Cromemco®

Cromix-Plus™ *User's*

Reference Manual

Cromemco®

Cromix-Plus™ *User's*

Reference Manual

023-5013

October 1987

Rev. F

CROMEMCO, Inc.
P.O. Box 7400
280 Bernardo Avenue
Mountain View, CA 94039

This manual was produced using a Cromemco System 400 computer running under the Cromemco UNIX Operating System. The text was edited with the Cromemco CE Editor. The edited text was formatted by the UNIX TROFF formatter and printed on a Texas Instruments Omnilaser 2108 Laser Printer.

CONTENTS

Chapter 1 - Introduction

This manual describes the Shell commands and utility programs supplied with the Cromix-Plus Operating System. For easy reference, commands are listed alphabetically. If you are unsure about the name of a particular command, consult chapter 2, a summary of available commands. Each command is discussed in detail in chapter 3.

To use this manual effectively, a basic understanding of the Cromix-Plus Operating System is required. Refer to the *Introduction to Cromix-Plus Manual*, in particular, chapters 4 and 6.

Chapter 2 - Summary of Commands and Utilities

The Cromix-Plus utility programs perform many necessary functions. They are similar to and used in conjunction with the Cromix Shell commands.

In contrast to the Shell commands, utility programs are not intrinsic to the Cromix-Plus Operating System but must be called from the disk when needed.

Write and **append** access for all utilities is limited to privileged users.

The following list summarizes the commands and programs described in detail in this manual.

access	changes access privileges of a file
bak	deletes files with **.bak** extensions
blink	links Z80 programs
boot	loads an operating system into memory and begins execution
ccall	calls another cromix system via a modem
cdoscopy	copies file to and from CDOS disks
cdosfix	strips CONTROL-Z's from the end of CDOS files
CE	Cromix-Plus Editor
check	runs the Dcheck and Icheck utilities
chowner	changes the owner or group owner of a file
clist	displays a file with page headings and line numbers

clock	executes a command line and reports the time needed to execute that command line
cmpasc	compares two ASCII files
compare	compares two files
config	configures memory for a **.bin** program
convert	converts file format (ASCII to EBCDIDIC, EBCDIC to ASCII, and so on)
convobj	converts **.obj** files to **.o68** format
copy	copies a file
cptree	makes a copy of a directory and its decendant directories
crc	computes the CRC of a file
create	creates a file
crogen	generates an operating system
cron	executes command files from the directory **/usr/cron/crontabs**
crontab	lists, removes, and changes the crontab file for the user
daemon	prints files into the **/usr/spool** directory
day	executes a command on the specified day
dcheck	verifies the integrity of a file system
ddt	used to trace user programs
delete	deletes a file
deltree	deletes a directory and all descendant files

directory	changes directories or displays the pathname of the current directory
diskinfo	prints hard disk information
dump	displays a file in hexadecimal and ASCII
ecc	manipulates the error-correcting memory controllers
echo	echoes arguments to the terminal
ed	enables the user to edit files
esdload	loads a program into an ESDI board
exit	exits from a shell
find	locates a file
fixsb	restores the superblock
flush	flushes system buffers
free	displays the amount of unused space on a device
ftar	creates and retrieves file archives
goto	transfers control within a command file
group	allows users to change their groups
help	displays the on-line manual
helpsys	displays information on functions in the **/usr/lib/syslib.obj** library
history	displays the last 10 executed Shell commands
icheck	checks the integrity of a file system
idump	displays the contents of an inode

if	conditionally executes another command
initflop	initializes a floppy diskette
inithard	initializes an STDC, ESDC, HD or SMD hard disk
inittape	initializes a floppy tape
input	reads a line from STDIN and writes it to STDOUT
ioload	loads a program into an I/O board
ipcrm	will remove one or more specified message, shared memory, or semaphore identifiers
ipcs	reports inter-process communications status
kill	sends a kill signal to a process
ksam	executes KSAM (Keyed Sequential Access Method) calls for users
ksclib	used to communicate with a KSAM running in the backround
l	lists directory or file information
link68	links 68000 programs; used with the **crogen** utility
logerr	accumulates errors detected and reported by the **ecc** utility
ls	lists directory or file information
mail	sends or displays mail
makdev	creates a device file
makdir	creates a directory file
make	constructs executable programs from separate modules

makfs	sets up the structure for a file system on disk
maklink	creates another name for an existing file
match	finds all occurrences of a string within a file
mode	displays or alters device modes
more	displays pages from pipes or from user files
mount	enables access to a file system
move	moves a file from one directory to another
msg	sends messages between users
ncheck	displays file information
newuser	displays information for new users
oldtape	initializes a floppy tape in old non-Ecc format
passwd	changes a user password; adds or deletes a user
patch	patches a file
patchbug	corrects a bug in **paslib.obj**
path	displays the pathname of executable files
priority	changes the priority of a process
priv	changes user status to that of a privileged user
prompt	changes the Shell prompt until the user next logs in
pstat	displays the status of a process
query	summarizes Shell commands and utility programs

ramdisk	creates, deletes, verifies, and checksums RAM disks
rcopy	copies a file or block device
readall	reads a device to check for errors
rename	changes the name of a directory or file
repeat	repeats a command
rewind	restores arguments within a command file to their original positions
rfile	allows binary files to be received over phone lines (used with **ccall**)
root	displays the name of the device containing the root directory
sc	calls the **screen** editor with the **-n** option, to create files compatible with the UNIX Operating System
scan	scans a directory tree
screen	the screen editor
set	set and displays Shell variables
setpri	changes the priority of a process
sfile	allows binary files to be sent over phone lines (used with **ccall**)
shell	creates a shell process
shift	shifts arguments in a command file
shutdown	shuts down the operating system
sim	allows CDOS programs to run under the Cromix Operating System
sleep	puts a process to sleep for a specified number of seconds

sort	sorts or merges files
split	splits a file into smaller pieces
spool	queues files and sends them to the printer
startup	contains commands executed every time the system is started up
stdload	loads a program into an STDC disk controller
stcrmp	tests for equality between the contents of STDIN and a set of strings
sync	provides a one-time flush of system buffers
sysdef	generates a configuration file for the **crogen** utility
tail	displays the last part of a file
tar	creates and retrieves file archives
tee	pipes output to a file as will as to STDOUT
term	displays or changes the terminal name
termcaps	defines terminal capabilities
testinp	tests for equality between the contents of a file and one or more strings
time	displays or alters the time and date
touch	changes the modification times of files to the current time (used with **make**)
type	displays an ASCII file
uboot	standalone boot program for the UNIX Operating System
unmount	disconnects a mounted file system from the current file system
update(1...2)	updates a Cromix-Plus system disk from system floppy disks

usage	displays directory-size information
vdt	controls a special terminal function
version	displays the version number of the operating system or a utility program. Calculates and verifies program integrity via CRC.
wait	waits until all detached processes have finished
wboot	writes the boot program to the boot area of the disk
wc	counts lines, words and characters
who	lists the users presently logged in
z80	used to run z80 programs
>	redirects the standard output to a file
>>	appends the standard output
<	redirects the standard output from a file
>*	redirects the standard output and standard error to a file
><	sequentially pipes the standard output only
><*	sequentially pipes the standard output and standard error
>>*	appends the standard output and standard error to a file
|	pipes the standard output only
|*	pipes both the standard output and standard error

Chapter 3 - Shell Commands and Utility Programs

An on-line version of this chapter is contained in the **/usr/help** directory, where it is accessible to all users via the **help** command. To display information about a particular command, use the **help** command, as in the following example:

```
jim[1] help match
```

This command displays detailed information about the **match** command. Information about any command can be obtained in the same way.

3.1 The Access Utility

utility:	ACCESS	
purpose:	This program changes the access privileges associated with a file.	
user access:	all users	files owned by the user
	privileged user	any file
summary:	access access-string file-list	
arguments:	access-privilege-specifier string	
	one or more pathnames	
options:	none	

Description

The **Access** utility allows a user to change file-access privileges.

The access-privilege-specifier string (first argument) contains three clusters of access flags separated by periods. The first cluster indicates owner permitted access, the second indicates group access, and the third indicates public access. Each cluster is composed of zero or more of the following flags, given in any order:

+	add the specified privileges
r	read access
e	execute access
w	write access
a	append access

The first cluster may be preceded by the shared text attribute:

s or +s	add shared text attribute
-s	remove shared text attribute

and/or the delete protect attribute:

p or +p	add delete protect attribute
-p	remove delete protect attribute

Access privileges under the Cromix-Plus system are discussed in detail in the following paragraphs.

File Protection

The Cromemco Cromix-Plus Operating System offers protection for files on many levels.

All files may be opened for exclusive or nonexclusive access. A file opened for exclusive access may not be opened by another process until it is closed by the process that originally opened it. If a file is opened for nonexclusive access, it may be simultaneously opened and accessed by more than one process.

File access privileges are divided into three population segments and four types of file accesses.

The first population segment is the owner of the file. This is normally the creator of the file. The second population segment is the group to which the owner belongs. A user's group number can be verified in the **/etc/passwd** file. The third population segment is the general public. This segment includes all system users.

There are four types of file access for each population segment. The first is read access. **Read** access allows the designated user to read the file. If a user has **read** access for a directory, the user may list the contents of the directory.

The second is **execute** access. **Execute** access allows the user to execute the file. If the user has execute access for a directory, the user may use the directory in a pathname.

The two remaining types of access are **write** access and **append** access. **Write** access allows the user to write to the file, meaning the user may write over or change data in the file.

Append access allows data to be added to the end of the file. Data may then be written to the file at a point past the end of file, and the end-of-file indicator is moved to the end of the newly added data. If **append** access is the only access specified, data written to the file may not be read.

Append access does not imply **write** access, but **write** access implies **append** access.

One type of access privilege for a population segment does not imply any other access privilege for that population segment. The categories of access privileges should be combined to provide meaningful data handling. For example, a user with **write** access to a file normally has **read** access.

One important point to consider when determining file access privileges is that the file's owner is a member of a group and a member of the public. Implicitly, the user has all access privileges granted to the public and to the group. Any member of the group enjoys all access privileges granted to the public.

All files are created with default access privileges as follows: **read**, **execute**, **write**, and **append** access privileges for the owner; **read** and **execute** access privileges for the group and public. The default owner is the user who executed the command creating the file. The system gathers its information on user name from the **/etc/passwd** file. (This default access may be changed by generating a new operating system with the **Crogen utility**.)

When files are created by programs that a user is running (e.g., CE, etc.), those files take on default attributes as described. These same programs can also alter existing files. In this case, the owner name is unchanged, but file access attributes may change. For instance, the CE Editor does not change file attributes after an editing session. However, since this effect may vary from command to command within a single program, and from program to program, users should be aware that file attributes are not immutable.

Access privileges take on a different meaning when applied to a directory. **Read** access for a directory means the user can use the **Ls** utility to see the contents of the directory. **Execute** access means the directory may be used in a pathname or that the user has access through the directory. **Write** access means the user can alter, create and delete files in the directory.

The **ls** utility program with the **-l** option may be used to check the access privileges associated with a given file. For example, the following command will list the access privileges of file **xyz**:

```
jim[1] ls -l xyz
312 1   rewa re-- re-- jim Mar-09 18:25 xyz
```

Reading this display from left to right, two items precede the access information: the numbers of bytes in the file (312) and the number of links to the file (1). The access information is displayed as three clusters of four characters. The four characters are **r** (read), **e** (execute), **w** (write) and **a** (append). The presence of one of these characters indicates that the specified population segment is endowed with the specified access privilege. The population segments are, from left to right owner, group, and public. Thus, in the above display, the owner has all four access privileges, while the group and public have only read and execute access privileges.

The last four items in the preceding display are: the name of the owner of the file (**jim**), the date and time the file was modified and, finally, the name of the file.

Users working within the Cromix-Plus file system must explicitly check the access attributes of files and directories they work in, use, or create for other users. Users must be aware of accessory files that may be required by programs they are running--help files, libraries, and so forth. Access and ownership of the accessory file--and access and ownership of parent directories all the way to the root--must be compatible with the operation of the program being executed.

For all errors implying access limitations, always check access privileges and the ownership of the directories, files, and ancestor directories involved.

Shared text attribute

Executable (68000) **.bin** files may have the shared text attribute. If an executable file has a shared text attribute, only one copy of the file will be loaded into memory even if more than one user is running it. Each invocation will get its own stack, all invocations will run on the same code in memory. The shared attribute should NOT be set for files that were not created accordingly. The rules are very complex and it is difficult to create a program that may be shared. Examples of such files are **Shell.bin** and **Gtty.bin** in the **/etc** directory.

Example

```
jim[1] access +s foo.bin
```

will declare the file foo.bin to have the shared text attribute.

Delete protect attribute

File names can be protected against deletion. If a file has the **delete protect** attribute set, the **Delete** command (or **deltree** command) will not delete it. Only the owner of the file or a privileged user can change the **delete protect** attribute. The **delete protect** attribute refers to file name, not to inode. This means that a file can have more than one name (link). Protecting one name does not protect other names. **Delete protect** attributes means only what it says: a file protected against deletion is not protected against rewriting unless other access attributes are used.

Example

```
jim[1] access P foo.bar
```

The file **foo.bar** cannot be deleted until the delete protect attribute is removed:

```
jim[2] access -P foo.bar
```

Notes

The **Access** utility allows the user to change file access privileges in several different ways. The first of these is to re-enter each access privilege for each population segment, making the desired changes. For example:

```
jim[1] access rewa.rw.a xyz
jim[2] ls -l xyz
312 1   rewa r-w- ---a jim Mar-09 18:25 xyz
```

The second method for specifying access privileges involves the use of the plus sign (+) in one or more of the access population clusters. When used in this manner, the plus sign means that the attributes for the specified population segment remain the same. The plus sign may also be followed by access privileges to be added for the given population segment.

```
jim[1] ls -l abc
517 1   rewa re-- re-- jim Mar-09 18:26 abc
jim[1] access +.+.+a abc
jim[2] ls -l abc
517 1   rewa.re.re-a jim Mar-09 18:26 abc
```

3.2 The Account Utility

utility:	**ACCOUNT**	
purpose:	This program displays account information.	
user access:	privileged user	
summary:	account [-lt] [file-name]	
arguments:	none	
options:	-l	list the current /etc/acc file
	-t	truncate the current /etc/acc file

Description

The **Account** utility sorts and displays the contents of the file **/etc/acc**. If the file **/etc/acc** exists, whenever a user logs in or logs out, a record describing that action is appended to the file **/etc/acc**. This is an addition to standard Cromix-Plus accounting.

The structure of the account records is described in the file **/usr/include/account.h**. The basis of account tracking is the job identification number (JOBID). When a **gtty** process is created a JOBID number is assigned to it. This number is then inherited by all processes started by this **Gtty** process. Each account record contains a JOBID. The JOBID numbers are counted from zero each time the system is started up. This means that JOBID numbers have no meaning across boot records.

The **-l** option can be used to inspect the **/etc/acc** file:

```
system[1] account -l
```

Each account record is listed in readable form. The fields listed are:

Job identification
Type of account record (see **/usr/include/account.h**)
Date of transaction
Time of transaction
User name (void except for login records)
Terminal name (void except for login records)
Time used (zero except for logout and exit records)

All records with the same JOBID (between two boot records) belong to the same user. A login record is generated when the user logs in and can be used to define the user name and the terminal name used to log in. A logout record is generated when the user logs out. It gives the total number of seconds of CPU time that the user used while logged in. Exit records are generated for detached processes that terminate after the user logs out.

The **Account** utility, when executed without options, reads the **/etc/acc** file, finds the login-logout pairs, and prints out the information for each login session. The information is printed in the form that allows convenient sorting if necessary. The records generated after the last startup are not printed as

there may be detached processes still running.

If the **-t** option is used, the **/etc/acc** file is cleaned up. Only the records generated after the last startup are retained. The original **/etc/acc** file is retained under the name **/etc/acc.bak.**

When the **-t** option is used it is implied that the system administrator is piping or redirecting the output of the **Account** utility to some other file, for example:

system[1] **account -t >> /etc/acc.total**

Such a command should be issued while no user is able to log in or log out. The **Init** command provides a convenient mechanism to do this.

The output of the **Account** command consists of lines such as:

guest	qtty2	1987-09-10 11:28:23	156	1,345

The meaning is:

> The user "guest" logged in on the terminal "qtty2" on September 10 at 11:28:23 and logged out 156 minutes later. He used up 1,345 seconds of system time.

3.3 The Bak Utility

utility:	**BAK**
purpose:	Deletes **.bak** files from the current directory
user access:	all users
summary:	bak
arguments:	none
options:	none

Description

The **Bak** utility deletes files in the current directory with the filename extension **.bak**.

3.4 The Blink Utility

utility: **BLINK**
purpose: This program links relocatable Z80 files.

user access: all users

summary: blink [-dinpqrxz] [-b outname]
filename [-s libname] . . .

arguments: one or more filenames

optional -s followed by a library name

options:

-b	output filename
-d	data section address
-i	IOP starting address
-n	no map
-p	program address
-q	do not display map
-r	relocatable binary
-s	search library
-x	bit-mapped
-z	size (use with -r)

Description

The **Blink** utility is a two-pass virtual linker of Z80 programs. One or more input files can be specified. An executable binary file is generated. **Blink** can be used to generate relocatable binary files that can share a bank of memory with other programs.

Options

The **-b** option may be used to specify the output filename. If used, the **-b** option must be followed by a space and the name of the binary file to be created. If this option is not used, the output file adopts the name of the first relocatable file specified on the command line. The output file has the filename extension **.bin**. This option may be used to force the output file to have a filename extension of **.com**. These are programs compatible with the CDOS Operating System only if they were written using CDOS system calls. The format for linking these files is:

jim[1] **blink -b filename.com modulenames**

The **-d** option is followed by a space and the hex value of the data section starting address.

The **-i** option is followed by a space and the hex value of the starting address for an IOP program. It allows relocation of the program above the memory area occupied by the IOP Monitor. The IOP Monitor occupies memory between addresses 0000 and 0800 hex in ROM, and between 7F00 and 7FFF hex in RAM. This option creates an automatic header for the program to be run in the IOP using

the **Iopload** utility program.

The **-n** option prevents creation of a link map. Otherwise, the link map is created and written to a file with the filename extension **.map**.

The **-p** option must be followed by a space and the hex value of the program starting address. If no starting address is specified, the program starts at 100 hex. A relocatable binary program is placed wherever there is space in a memory bank.

The **-q** option inhibits display of the link map on the terminal. Otherwise, the link map is displayed on the terminal.

The **-r** option causes generation of the output file in relocatable binary format. Programs in this format can be executed with another process in a single bank of memory. The **-r** option is used with the **-z** option discussed below.

The **-s** option precedes the filename of the library to be searched. The option applies only to the file immediately following it, and must be specified for each file to which it applies. **Blink** searches the **.rel** file for necessary functions. If no library is specified using the **-s** option, and there is no library in the current directory, the program looks into **/usr/lib**, which is the default system library directory.

The **-x** option makes the output file a bit-mapped self-relocating file. This option generates a self-relocating file which, when loaded into a user bank, loads in highest available memory and sets high memory to the byte just below itself. This option is used in linking the Cromemco **Debug** program.

The **-z** option allocates a specific size for the program segment. This switch is used only with the **-r** option, and only when free space (more than **Blink** normally allocates) is desired in the program area.

Notes

Blink manages memory so as to link programs up to the total amount of memory available. The memory area used by the linker during execution does not impose a restriction on the size of the program being linked. Thus, Cromix-Plus programs up to 64K, minus 1K of memory occupied by the Cromix-Plus Operating System in each user bank, can be linked by **Blink**.

Relocatable binary programs are treated as normal Z80 programs under the Cromix-Plus system.

CDOS programs running under the Cromix-Plus Operating System are limited to approximately 4K less memory than the 63K available to Cromix-Plus programs. This is because Sim, the CDOS simulator, must also be loaded.

COBOL programs using segmentation cannot be linked with **Blink**.

3.5 The Boot Utility

utility:	**BOOT**
purpose:	This utility loads an operating system into memory.
user access:	privileged user
summary:	boot [filename]
arguments:	filename (optional)
options:	none

Description

The **Boot** utility loads an operating system into memory.

If no argument is given, the file **/cromix.sys** is loaded, and execution begins. In this manner, the **Boot** utility can be used to warm boot the Cromix-Plus Operating System.

Example:

```
system[1] boot

System shutdown in progress
System shutdown complete
```

The raw console then displays the normal boot messages.

If **Boot** is followed by a filename, the file is assumed to have a .sys extension.

Notes

Because this program loads an operating system, it interrupts any active processes. Be sure that no one else is executing a program and that there are no detached processes running on the system before executing the **Boot** utility. Otherwise, data may be lost.

One quick method to determine if there are users on the system is to execute the **Ps** (Process Status) command:

```
system[1] ps -a

PID     State       Command
  1       S         -
112       R         Shell
```

```
105     R       ce letter
 18     S       gtty 19200 tty6 3102 12
 94     S       shell
 16     S       shell
 15     S       shell
 14     S       shell
 89     S       gtty 9600 tty1 C-40 1
```

Here the command is executed with the **-a** (for all) option. The display shows one user running the CE Editor program to edit a file named **letter**. If the **Boot** program were executed at this point, the user would lose all editing changes made during this session.

As long as all lines of the **Ps** display show a command of **shell** or **gtty**, no processes are running, and it is safe to load an operating system.

The **Boot** utility may be executed only by a privileged user.

3.6 The Ccall Utility

utility:	**CCALL**
purpose:	This program calls another Cromix (or non-Cromix) system using a modem.
user access:	all users
summary:	ccall [-q] [-d dev-name] [-b baud]
arguments:	none
options:	-q quiet (default is verbose) -d qtty device-name (default is /dev/modem) -b baud rate (default is current rate)

Description

The **Ccall** utility allows one Cromix-Plus system to act as a terminal to another Cromix-Plus system (or to another completely different system). For best performance, connect the modem to any serial port on the IOP/Quadart, or to any serial port on the new Octart. With some restrictions, **Ccall** can also run on a modem connected to the old OCTART or TU-ART. When connected to the IOP/Quadart, a 12-wire cable designed for this purpose should be used. When connected the the new Octart, the 5-wire cable as described in the Administrator guide should be used. The modems on each system must be compatible.

Using the **Maklink** utility, make a link from the appropriate **qtty** device in the **/dev** directory to **/dev/modem**.

Ccall provides no automatic calling or modem initialization. Once the baud rate on the **/dev/modem** device is set (if necessary), all characters typed on the terminal are sent to the modem, and any character received from the modem is echoed to the terminal. The following **Ccall** commands are also available (**Ccall** interprets lines beginning with a tilde (~) as escape sequences):

~.	Terminate Ccall
"~~ ..."	Send the line: "~ ..." to the remote system.
~< **filename**	Redirect the contents of a file to the remote system, as though the contents had been typed from the terminal.
~> **filename**	Redirect output from the remote system to the specified file. Output redirected in

	this way is written to an ordinary file and to the standard output (for display). To write output to the specified file only, type a colon after the redirect symbol: **~>: filename** To append output to a file, use the redirect-and-append symbol (>>) with either form of the command.
~>	The command ~> ends redirection.
~sh	Invoke an interactive shell on the local system. To exit the Shell type EXit or CNTRL-Z.
~sh command	Execute the command on the local system (via shell -c).
~put [-f] file-list	Copy the specified files from the local system to the remote system. If **~put** finds a file with the same name at the remote system, that file is not copied unless the -f option is used to overwrite the existing file. **~put** uses the **Sfile** and **Rfile** utilities to perform error-free block transfers.
~take [-f] file-list	Copy the specified files from the remote system to the current directory on the local system. The -f option overwrites files with the same name. **~take** uses the **Sfile** and **Rfile** utilities to perform error-free block transfers.
~h	Print a summary of the **Ccall** commands.

Options

The **-b** option sets the baud rate of the transmitting device. Possible settings are 110, 150, 300, 1200, 2400, 4800, 9600, and 19200 baud. Without this option, the baud rate set for the device **/dev/modem** is used; if a baud rate has not been defined (or the device was discarded), 1200 baud is used.

The **-d** option specifies a transmitting device (**qtty**) other than **/dev/modem**.

The **-q** (for quiet) option reduces the number of **Ccall** messages.

Notes

The remote system should have a compatible modem that can automatically answer the phone call. On a remote Cromix-Plus system, the **mtty** terminal connected to the modem must be enabled in the **/etc/ttys** file. If the **mtty** baud rate is set to automatic (a), the caller can establish the baud rate by repeatedly pressing RETURN until the remote system recognizes the baud rate and responds with a login prompt. **Ccall** sets many of the modes for both the local and remote systems. The optional mode settings are described below.

Remote system, mtty, SIGHUPall

The **mtty** driver enables SIGHUPall on all **mtty** devices (and reenables it each time a device is DIScarded). When the communication link is broken, all processes on the remote system that use **mtty** as the controlling terminal are aborted (Shell processes are logged out) by the SIGHANGUP signal (unless this signal is trapped or otherwise ignored). If you disable SIGHUPall after logging in, a broken communication has no effect on the remote precesses, and you can reestablish the phone link and proceed from where you left off. However, disabling SIGHUPall is a serious security risk, as another user can call in and inherit the previous user's shell, password, and so on.

Remote system, mtty, DIScard

The **mtty** driver enables DIScard on all **mtty** devices. Thus, when an **mtty** device is finally closed (i.e., when the user EXits the Shell or when his Shell is killed by the SIGHUP signal), the device is reinitialized.

Remote system, mtty, HUPenable

The **mtty** driver disables HUPenable on all **mtty** devices. Thus, the driver does not try to hang up the phone when the **mtty** device is finally closed. Note that the modem may be configured to hang up if the data carrier is lost. If you enable HUPenable, exiting from the Shell will hang up the phone.

Local system, modem

Ccall does not change the DIScard and HUPenable modes of the local modem (SIGHUPall is ineffective). If you enable HUPenable before using **Ccall**, the phone will hang up when Ccall exits.

If the phone is not automatically hung up upon exiting from **Ccall**, execute the command:

jim[1] **mode modem hup**

If DIScard is not enabled, the next caller can use **Ccall** at the previous baud rate without having to specify the **-b** option; if DIScard is enabled, the baud rate will default to 1200 baud unless the **-b** option is used.

3.7 The Cdoscopy Utility

utility:	**CDOSCOPY**	
purpose:	This utility copies files to and from CDOS disks.	
user access:	all users	
summary:	cdoscopy [-belvw] devname file-list	
arguments:	Cromix device name	
	name(s) of the file(s) to be copied	
options:	-b	binary file
	-e	erase file
	-l	list CDOS directory
	-v	verbose
	-w	write CDOS file

Description

The **Cdoscopy** utility copies files from a Cromemco Disk Operating System (CDOS) format disk to a Cromemco Cromix-Plus Operating System format disk, and vice versa. For example:

```
jim[1] cdoscopy fdb letter
jim[2] cdoscopy -w sfda notes
```

The first of these command lines copies a CDOS file named **letter** (located on a large floppy disk in drive B) into the user's current directory. The second command line copies the Cromix file named notes from the user's current directory to a small floppy disk in drive A. In the first case, the file is converted from a CDOS format to a Cromix format. A Cromix-format-to-CDOS-format conversion takes place in the second example.

The Cromix-Plus Operating System cannot read CDOS files. Programs to be executed and data to be read under the Cromix-Plus Operating System must be transferred from CDOS formatted disks to Cromix formatted disks before execution begins.

Where a file pathname is specified, CDOS considers the lowest level filename. This is the portion of the pathname to the right of the rightmost slash. For instance, the following command line puts the file named **memo** onto the CDOS format disk in drive B.

```
jim[1] cdoscopy -w fdb /usr/mary/memo
```

Options

The **-b** option copies binary files. When this option is used, the 1Ah (end-of-file mark) is not stripped from the end of the file.

The **-e** option erases the specified file(s) from the CDOS disk.

The **-l** option displays the contents of the CDOS directory.

The **-v** option displays files while they are copied to and from CDOS disks.

The **-w** option causes the file to be written to the CDOS disk.

Notes

When an ambiguous CDOS file reference is used, it must be enclosed in quotation marks.

CDOS filenames must also be legal Cromix filenames. If not, use the CDOS Operating System to rename the files, then use the Cdoscopy utility.

Examples:

```
jim[1] cdoscopy -v fda "*.txt"
jim[2] cdoscopy -l fdb
```

These examples assume that the disks in drive A (**fda**) and B (**fdb**) are CDOS disks. The first example copies all CDOS files on drive A having the filename extension **.txt** into the current directory. The ambiguous CDOS file reference was placed inside quotation marks.

The second example displays the directory of the CDOS disk in drive B (Cromix file designation **fdb**).

Refer to the Cromix-Plus System Administrators Guide for a list of device names.

3.8 The Cdosfix Utility

utility:	**CDOSFIX**
purpose:	This program strips the ^Z's from the end of CDOS files.
user access:	all users
summary:	cdosfix filename [filename ...]
arguments:	one or more filenames
options:	none

Description

The Cdosfix utility strips the ^Z's from the end of files created using the CDOS Operating System.

3.9 The CE Utility

utility: **CE**

purpose: This program is used to edit files.

user access: all users

summary: ce [-rucnetvifamsopjk1h] [-d logfile] [-x tfile]
[-z efile] [-w #][-l #] [-b #] filename [filename]

arguments: filename(s) to be edited

options:

-r	read-only mode
-u	update mode
-c	lines terminated by CR-LF pair
-n	lines terminated by LF only
-t	blanks in output replaced by TABs
-e	blanks in output not replaced by TABs
-i	auto indent mode on
-v	auto indent mode off
-a	display cursor address
-f	display name of file being edited
-m	multi-file session
-s	single file session
-x	declare local termcaps file
-z	declare environment file
-o	enable Xon/Xoff protocol
-p	disable Xon/Xoff protocol
-w	line width
-l	start address (line)
-b	start address (byte)
-d	replay editing session
-1	replay editing session one keystroke at a time
-h	do not change tabs
-j	turn keystroke recording ON
-k	turn keystroke recording OFF
-h	do not change tabs

The following is a chart of all environmental options and conditions and their interrelationship:

Function	**command option**	**key/variable**	**ce_env**
Line alignment (rigid,elastic)		[a] align	al
Bell enable		[b] bell	bl
Display cursor address	-a,-f	[c] pos	cu

TAB-key space substitution		[d] step	di
Left margin		[e] lmarg	lm
Right margin		[f] rmarg	rm
Paragraph indentation		[g] para	pi
Highlight command line		[h] head	he
Automatic indentation	-i,-v	[i] inden	in
Right margin justification		[j] just	ju
Case conversion (upper/lower)		[k] kase	ka
Line termination	-c,-n	[l] lterm	lt
Mode (update/read-only)	-u,-r	[m] mode	mo
Re-display line count		[n] rstl	rl
Program Editor/Word Processor		[p] type	pt
Keystroke recording	-j,-k	[r] recd	rs
Sync (no buffering on update)		[s] sync	sy
TAB character handling	-t,-e,-h	[t] tabs	ta
Xon/Xoff	-o,-p	[x] xonof	xo
Multi-file editing	-m,-s	[ ] file	
Declare environment file	-z		
Declare local termcaps file file	-x		
Line width	-w		
Start address (line)	-l		
Start address (byte)	-b		
Replay editing session	-d		
Single keystroke replay	-1		

COMMANDS

Case Sensitivity

Many CE commands are case sensitive. An uppercase version of a command might have a radically different effect than its lowercase counterpart. For example, lowercase "c" invokes the COPY command (copy text from one location in the file to another). Uppercase "C" causes the case of all characters entered from the keyboard to be converted either to upper or lowercase. **The alpha lock key should not be used with CE. The environmental variable "Kase" should be set to insert upper case text. (See the description of the "I" command).**

Word Processing/Program Editing

CE can be modified so that the action of two of its commands work more conveniently for either editing computer programs or for word processing.

If the environment variable **type** (as described in the previous section) is set to be PROGRAM, the command enabled by pressing the "b" key will be BRACKET. This provides assistance to C language programmers by matching the "bracket" characters "{","[","(" with their complements. With **type** set to PROGRAM, the "p" or PAGE command will advance the screen display by one page.

If the environment variable **type** is set to WORD, the command enabled by pressing the "b" key will be BEAUTIFY. This provides text formatting (justified or unjustified) on a paragraph by paragraph, or marker to marker basis. With **type** set to WORD, the "p" or PARAGRAPH command will advance the cursor to the next paragraph, the "P" or PAGE command will advance the screen display by one page.

Command Format

Many commands will prompt for some type of user input and also provide options which effect the command's behavior. Input is prompted for by the cursor moving to the command line and waiting. Options are indicated on the command line (after invocation) within brackets ([]) to the right of the command name. Commands which are direction sensitive will display ">" or "<" to the left of the command name to the indicate direction of their action. Many commands accept a "repeat factor" which is also displayed to the left of the command name. This value is entered prior to pressing the key which invokes the actual command.

```
>     3 Substitute: [eqvr] <old>,<new>
```

This line indicates that:

- the command will search in the forward direction
- 3 substitutions will be performed (if possible)
- four options are available (see command description)
- input is expected in the form "<old>,<new>".

The character "#" represents the value 65335. For example, typing "#" prior to invoking the SUBSTITUTE command ("s"), will result in up to 65335 substitutions.

Command Descriptions

@ for "SET MARKER"

The current cursor position can be "labelled" by a single digit number 1-9. In addition, four predefined numbers b(egin), e(nd), c(ursor), p(revious) can be used by commands like **JUMP**, **COPY**, **MOVE**, **ZAP**, etc.

a or A for "AGAIN"

Repeats the last **FIND** or SUBSTITUTE command. No action is taken if either command has not been used.

b or B for "BRACKET"
(PROGRAM mode only)

This command causes the cursor to jump to the next matching bracketing character if the cursor is positioned on any of the following characters: "(", "[", "{". If there is no matching bracket, the editor beeps and the cursor does not move. If the character at the cursor is not a bracketing character, no action is taken.

b for "Beautify"
(WORD processing mode only)

This command formats text. Paragraphs will be rearranged within the left, right and paragraph margins. The margin values are set via the environment variables **lmarg, rmarg and para**. They are set in either the **ce_env** environment file or interactively via the **Q** command. The formatting can be done with or without right margin justification. This is determined by the value of the **just** variable, which is also set in either the **ce_env** environment file or interactively via the **Q** command. A paragraph is defined as any sequence of non-blank lines, or as text separated by one or more blank lines.

The "b" command will format the paragraph in which the cursor is located. If the cursor is not located in a paragraph, it will format the first paragraph it encounters. This provides for convenient text formatting on a paragraph by paragraph basis by simply typing successive "b's". The PARAGRAPH command can be used to skip paragraphs which should not be formatted.

This command is case sensitive.

B for "Beautify"
(WORD processing mode only)

The "B" command will prompt for two markers to define the portion of text to be formatted. The markers define lines (beginning with the first and stopping before the second) and not character positions within the lines.

This command is case sensitive.

c for "COPY"

This command copies the block of text defined by two markers to the current cursor position. All characters between the stated markers will be copied. The optional parameter "r" (RECTANGLE) causes the marker positions to be interpreted as being in diagonally opposite corners of a rectangle of text to be copied. The dimensions of the marked rectangle may be verified using the "V" (Verify) command.

This command is case sensitive.

C for "CONVERT CASE"

Converts the case of all characters within the given range, according to the current value of the "kase" ENVIRONMENT variable (TO UPPER, TO LOWER or TOGGLE). The range is defined by

two file markers. (See "k" and "K" commands)

This command is case sensitive.

d or D for "DELETE"

While in "DELETE mode", text can be deleted horizontally one character at a time by pressing the space bar or the left/right arrow keys. Pressing "w" deletes all blank and non-blank characters from the cursor position to the next word. Text can be deleted vertically a line at a time by pressing the RETURN key or the up/down arrow keys. The entire line upon which the cursor resides will be deleted. Deleted text can be restored by using the arrow key in opposite direction of the deletion. Restoration is only available prior to typing ESCAPE to terminate the DELETE command.

CE is somewhat restrictive while in DELETE mode. Once the direction has been selected (horizontal or vertical) it cannot be changed. Consistent handling of "hidden" lines would otherwise be difficult without this restriction.

e or E for "EXIT"

There are four optional methods of exiting CE: Update, Quit, Terminate or Continue. CE prompts for confirmation if you press "q" to exit from a file that has been modified. A similar check is performed if "u" is selected, and the file has not been modified. Pressing "t" in EXIT mode quits the current file and aborts an entire muti-file editing session. Pressing "c" for "continue" will cause the file to be updated to the disk and the editing session to resume with the cursor at its position prior to update. **NOTE:** If keystroke recording is being used as a means of backup, use of the "EXIT-continue" option is not necessary. If, however, keystroke recording is activated and the "EXIT-continue" option is used, CE will update the ".bak" file and reset the keystroke log, thus providing replay from the point of the EXIT-continue command.

Since CE reads the entire file into memory prior to execution and then closes it, severing all connections with it, a security check is performed. CE keeps the last modification time of every file being edited. If that stored modification time does not match the same data for that file at the moment when Update is to be performed, a warning message is issued and execution is suspended. Two typical examples of such a situation: two users concurrently editing the same file; a single user leaving the editor via the "%" command and then inadvertently editing the same file from the new shell.

f or F for "FIND"

The FIND command prompts for a pattern of characters (strings) for which it should search. The pattern must be delimited by a pair of special (non-alphanumeric) characters (commonly '/', '.' or '?'). Pressing "e" prior to giving the search pattern of a FIND command causes the search to be case sensitive. Pressing "r" (for range) and typing a marker number before the pattern, will cause the search to be terminated at the given marker (the marker must exist prior to giving the command). If a matching pattern is not found, the cursor will return to its current location.

h or H for "HOME"

Moves the cursor to its home position (upper left corner).

i for "INSERT"

This command is used for the insertion of characters. When the line being typed exceeds 79 characters, CE will shift the display of the line to the left 16 characters. This allows for the insertion of lines as long as 1023 characters. Pressing RETURN restores the line to its original "window dependent" position.

CE also supports an automatic indentation feature in the INSERT mode. If auto-indent is enabled (the default, see section on the CE environment), the position of the beginning of the next line (after pressing RETURN), will be the same column as the first non-blank character inserted by the previous invocation of the INSERT command.

After terminating a line with a RETURN, the left arrow key can be used to return the cursor to the columns where previous insertions have occurred on multiple lines. This is valid for the duration of the current INSERT command only. This can be helpful in aligning indentation levels of programming statements.

Characters may be inserted automatically converted to lower or upper case if the environment variable "kase" is set either to be TO LOWER or TO UPPER respectively. No conversion is performed if the "kase" variable is set to TOGGLE (the default value).

This feature is very useful in preventing problems with CE commands that are case sensitive, since it enables the entry of upper case characters without using the ALPHA LOCK key.

This is a case sensitive command.

I for "INSERT"

An upper case "I" inserts one blank line and returns the cursor to the beginning of the line prior to insertion.

This is a case sensitive command.

j for "JUMP"

The "j" command must be followed by a destination. The destination can be either one of the special markers (Begin, Cursor, End, Previous) or defined "numeric" markers between 1 and 9. It is also possible to jump to a specified line by typing the letter "l" before the marker. When "l" has been selected, it has to be followed by either a destination line number (unsigned) or a number preceded by the sign ("+" or "-"). Signed numbers will cause a relative jump forward or backward from the current position, while an unsigned number indicates a jump to the designated line. The special marker, "Previous" is automatically set to be the position of the last "find" by the FIND command.

J for "JUMP"
(when CE is called by a compiler)

The "J" command is only available for use when CE is invoked directly by one of the specially equiped Cromemco compilers (see the section in the introduction of this document entitled "Interaction With Cromemco Compilers" for details).

CE is able to keep track of nine separate syntax errors and assigns special markers to their positions in the file. These markers can be reached via the uppercase version of the JUMP command. It is possible to move to those locations directly by specifying their numbers, or by moving to them relatively with the "next" or "previous" options to the JUMP command. After each JUMP, the editor command line will be replaced with the text of the associated compiler error message.

k for "CASE"

The "k" command toggles the case of the character at the cursor position. (See "C" and "K" commands)

This is a case sensitive command.

K for "CASE"

Pressing "K" toggles the case of all alphabetic characters from the cursor position to the next non-letter, non-digit character (i.e. toggles the current word). (See "C" and "k" commands)

This is a case sensitive command.

l or L for "LIST"

This command causes specified lines of text to be spooled and then printed by the device **/dev/prt** (Cromix-Plus) . The "n" option will cause the printed lines to be numbered, and each page to be headed by the name of the file. Note that lines are numbered in relation to the entire file. The line numbers are the same as they would be had the entire file been printed.

m or M for "MOVE"

This command moves the block of text defined by two markers to the current cursor position. All characters between the stated markers will be moved. The optional parameter "r" (RECTANGLE) causes the marker positions to be interpreted as being in diagonally opposite corners of a rectangle of text to be moved. The dimensions of the marked rectangle may be verified using the "V" (Verify) command.

n for "NEXT"

Concatenates the "next line" to the current line adding one space at the juncture of the lines.

This is a case sensitive command.

N for "NEXT"

Concatenates the "next line" to the current line.

This is a case sensitive command.

p or P for "PAGE"
(PROGRAM mode only)

This command advances the screen display by one page (number of lines as described in **/etc/termcaps**, minus one). If preceded by a number, the screen will advance that number of pages.

p for "PARAGRAPH"
(WORD processing mode only)

The "p" command advances the cursor to the next paragraph.

This is a case sensitive command.

P for "PAGE"
(WORD processing mode only)

The "P" command advances the screen display by one page (number of lines as described in **/etc/termcaps**, minus one). If preceded by a number, the screen will advance that number of pages.

This is a case sensitive command.

q or Q for "SET MODES OF OPERATION"

Please refer to the section titled "The CE Environment".

r or R for "READ EXTERNAL FILE"

This command prompts for the pathname of a file to be read from the disk into the file which is currently being edited. The additional text will be inserted beginning at the current cursor position.

s for "SUBSTITUTE"

This command is very similar to the FIND command. The string to be substituted is entered between delimiters, followed by the delimited string which is to be substituted for it. The "q" option will cause the user to be queried prior to each substitution. The "v" option will cause multiple substitutions to occur at approximately once per second and the string will be highlighted as the substitution occurs for user inspection. The "e" and "r" options function as in the FIND command.

This command is case sensitive.

S for "SWAP"

The current line will be swapped with the next one.

This command is case sensitive.

v for "VERIFY"

The current contents of the screen will be cleared and redisplayed.

This command is case sensitive.

V for "VERIFY"

The "V" command, is used to verify delimited (marked) areas for accuracy prior to acting upon them. This command prompts for a definition of mode (AREA/RECTANGLE) and markers (2 or 4) which define the block of text to be verified. The marked block will be highlighted. The concept of AREA and RECTANGLE are discussed under the ZAP command description.

This command is case sensitive.

w or W for "WRITE"

This command writes a marked area of text to a specified file. The command prompts for two markers to define the block of text. The "a" option causes the text to be appended to an existing file. The "r" (raw) option causes the text to be written to the file without any line terminators.

x for "EXCHANGE"

Typing "x" causes characters entered from the keyboard to be placed in the file, "writing over" or substituting any characters encountered.

This command is case sensitive.

X for "EXCHANGE WORD"

Typing "X" causes an "exchange word" operation. This involves deleting all characters from the cursor to the first blank character and then placing the editor in a limited INSERT mode. A space will be preserved at the end of the insertion. If the ESCAPE key is typed as the first character of the replacement, the word will be restored in its original form.

This command is case sensitive.

y for "INDENT A BLOCK"

The "y" command changes the horizontal position of a selected portion of text. This command functions logically like a "column insert" mode. This portion of text is defined either by two markers or by a repeat factor entered prior to invoking the command. Additionally, it is possible to insert or delete characters at the starting column (seam) throughout the entire marked block of text.

The "y" command accepts two different types of repeat factors. The first repeat factor serves as an alternate method of defining the selected portion of text to be moved. It must be entered prior to invoking the command (typing "y"). The repeat factor consists of either a positive or negative (preceded by a "-") number. This factor represents the number of lines relative to the current cursor position which should be "yanked". Negative numbers define lines above the line upon which the cursor resides, NOT including that line. Positive numbers define lines below the line upon which the cursor resides, including that line. It should be noted that "positive" and "negative" are defined relative to the currently defined editing direction which is indicated by either ">" or "<" at the left of the command line.

Upon invocation, the column at the "yank" point (seam), defined by the cursor's position, is highlighted.

The second repeat factor which this command accepts, is entered after defining the text block and just prior to actually moving the text with the arrow keys, etc. In this case, the repeat factor defines the number of column positions which the block of text should be moved (in either direction).

After using file markers to define the portion of text to be moved, place the cursor at the correct column position. It is not important on which line the cursor is located. It is however, best to place it on the column "yanked" text. The command is then invoked by typing "y", optionally preceded by the previously discussed repeat factor. If the repeat factor was not entered, then the block of text to be "yanked" must be described by entering the numbers of the file markers which define it. The marked block of text may now be moved horizontally either to the left or right using the left or right-arrow keys (replacing any existing characters). Any printable character entered from the keyboard at this point will be inserted at that column position on all of the lines within the defined block of text. If the starting position (seam) is located past the last character on a line within the defined block, the characters entered from the keyboard will be automatically concatenated to the end of that line. If this effect is not desired, the "p" option (padding) can be invoked prior to declaring the markers. Used in this manner the entered text will be uniformly entered at the same column position throughout the defined block. Repeat factors apply to character insertion as well. Please note that when entering numbers, that the first number entered will be interpreted as a repeat factor and not as the character to be inserted. Therefore, to enter one column of the character "4", you would type "1" (repeat factor) followed by "4" (character to be inserted).

The following example may serve as some illustration:

On a new file, enter "20" followed by "I" followed by (ESC) to enter the INSERT mode and create 20 empty lines. Type "y" to invoke the YANK command and then define your block to be the entire file (**Begin** and **End** ["b","e"]). Enter the characters "CROMEMCO" followed by typing the ESC key. The result will be 20 lines, beginning at column #1, which contain the characters "CROMEMCO". Place the cursor on the initial "C" in CROMEMCO and type "y", again followed by "be". By typing the right-arrow key, the entire column of CROMEMCO's will be moved. If the arrow commands are preceded by a repeat factor, the column will be moved that number of columns.

This command is case sensitive.

Y for "CHANGE WINDOW POSITION"

Pressing "Y" allows you to change the position of the editing window (horizontal scroll). The left and right arrow keys can be used to shift the window to the left or right. The "/" option will shift the window to the left enough to move the right most characters onto the screen. The "?" option will return the window to the position where column 0 is at the left edge of the screen. A repeat factor is also accepted.

If the terminal does not have the features required for horizontal scrolling (either "insert a column", or "insert/delete" a character in a "page" mode) the arrow keys will not be accepted. The only way of changing a window without these features is by supplying a new (numeric) value, for the location of the first column to. The screen will then be redisplayed.

This command is case sensitive.

z or Z for "ZAP"

ZAP deletes ("zaps") a marked area of text. ZAP has five optional parameters **[alrst]**:

- **a** - "zaps" a selected AREA
- **l** - "zaps" to the end of the line (left or right, depending on the direction)
- **r** - "zaps" the selected RECTANGLE
- **s** - "zaps" to the end of the current sentence (direction dependent)
- **t** - will remove any extraneous blank characters located past the end of any line within the specified range

The "r" and "a" options are described in more detail below. If none of the options are selected, the ZAP command will "zap" all characters between the two selected markers.

RECTANGLE "zap" interprets the two markers which must follow the selection of "r", as two diagonally opposite corners of a rectangle (see COPY command). Since this is a destructive command, to prevent possible mistakes, the selected block of text is displayed in inverse video (even if it is not present on the currently displayed screen) and confirmation is required for the action to be performed.

AREA "zap" is a generalization of the RECTANGLE ZAP in the sense that it expects four markers as definition of the area to be "zapped". The first two markers define the range in the vertical direction (lines), and the second two markers define the range in the horizontal direction (columns). In this case the marker "end" defines either the last line or last column. For example, to trim away all characters after column 72 in the range of lines defined by markers 1 and 2, place the cursor on any line at column 73 and enter "za12ce". As with the RECTANGLE version of the ZAP command, a verification of the action will be required.

The following characters have special meaning in EDIT mode:

%,! Fork a shell and exit to the operating system shell command line. The editing session may be re-entered by typing "exit", CNTL-Z (Cromix) or CNTL-D (UNIX).

- Sets the direction temporarily backward (for a single command).

+ Sets the direction temporarily forward (for a single command).

, Sets the direction backward.

. Sets the direction forward.

> In Edit mode, pressing ">" moves the cursor forward to the next word (the next string of non-blank characters). This character is NOT available as a "change of direction" command.

< In Edit mode, pressing "<" moves the cursor backward to the previous word (the last string of non-blank characters). This character is NOT available as a "change of direction" command.

/ In Edit mode, pressing "/" moves the cursor to the end of the line. If the right end of the line is off the screen, and the "align" variable is set to be RIGID, the cursor stops at the screen edge. With the "align" variable set to be ELASTIC, the line will be forced out of alignment with the current window position and the end of the line will become accessible.

\ Copies current line to the next line. Repeat factors accepted.

? In Edit mode, pressing "?" moves the cursor to the start of the line. If the left end of the line is off the screen, the cursor stops at the screen edge. With the "align" variable set to be ELASTIC, the line will be forced out of alignment with the current window position and the beginning of the line will become accessible.

' To clear existing markers, press "'" followed by either a marker number (to clear that marker) or an "*" (to clear all markers).

Control Characters

While in INSERT or EXCHANGE mode, the editor will normally not allow the insertion of control characters. Preceding it with a special character that announces the entry of a control character, will cause it to be accepted. That special character has the default value of CNTRL-V. It may also be defined using the **kc** termcaps capability.

Depending on the type of terminal, control characters will either be represented graphically (eg. Cromemco 3102, C-10 or C-5 terminals), or as a question mark ("?"). Terminals which support a "monitor mode" will display control characters using available special symbols, while the rest will display a "?". The actual value of any character represented by "?" will be displayed as a two digit Hex number at the upper right corner of the screen when the cursor is positioned upon that character and if the "display cursor address" mode is selected (see environment section). The **mo** and **mf** terminal capabilities define the escape sequences to enter and leave the monitor mode. Terminals that would display the **mf** sequence (while being in the monitor mode), should use the Boolean capability **mt**. In that case, the **mf** sequence should also contain the sequence to back the cursor up for the length of the "leave monitor mode sequence", followed by deleting the same number of characters. The actual implementation on the WYSE 85 terminal is a good example:

```
mo=\E[3h        # Go into monitor mode
mf=\E[3l        # Exit monitor mode
```

Once in monitor mode, the Wyse terminal will display (and interpret) all subsequent escape sequences (including control characters), therefore the **mf** sequence itself (the four characters "ESC" "[" "3" "l") will be sent to the screen - corrupting its contents. To resolve that problem, **mf** should actually be defined as:

```
mf=\E[3l\E[4D\E[4P
```

This means that after exiting monitor mode (and sending the characters "\E" "[" "3" "l" to the screen), the cursor should go back four character positions ("\E" "[" "4" "D") and erase four characters ("\E" "[" "4" "P"). Note that this discussion applies only if the **mf** sequence is displayed while in the monitor mode.

3.10 The Check Command

utility:	**CHECK**
purpose:	This program runs the **dcheck** and **icheck** utilities.
user access:	privileged user
summary:	check [-s] [devname]
arguments:	optional device name
options:	-s

Description

The **check** command runs the programs **dcheck** and **icheck** on a file system. Check should be run after rebooting the system or any time the integrity of the file system is in doubt. The **startup** command file program executed after every boot-up displays a message which indicates when the **check** program needs to be run. See the **startup** command file description in this manual for more information on **check**.

Options

The **-s** option is the salvage option used with **dcheck** and **icheck** to repair most file system problems. See the description of the **dcheck** and **icheck** utilities in this manual for more information. The system is rebooted after running **check** with the **-s** (for salvage) option.

Notes

In general, it is safer to run **check** with the **-s** option only on an unmounted device. When run on a mounted device, especially the root device, the file-structure problem you are attempting to correct may be immediately recreated.

3.11 The Chowner Utility

utility:	**CHOWNER**
purpose:	This program changes the owner or group of a file.
user access:	privileged user
summary:	chowner [-gv] ownername file-list
arguments:	name or number of the user to whom ownership is to be transferred or name or number of the group to which ownership is to be transferred and one or more filenames
options:	-g change group -v verbose

Description

The **Chowner** utility changes the owner or group associated with any type of file. If the file **abc** is in the current directory and is owned by **mark**, the **Ls** utility might display it as:

```
system[1] ls -l abc
27      1 rewa re-- re-- mark                Mar-11 19:59  abc
```

Using the **Chowner** utility, ownership can be transferred to cindy:

```
system[1] chowner cindy abc
system[2] ls -l abc
27      1 rewa re-- re-- cindy               Mar-11 19:59  abc
```

Options

The -g option allows the **Chowner** utility to change the group associated with the file. This option is used in the manner previously described, substituting the group name for the owner name.

The -v option displays the name of each file as its ownership is changed.

Notes

When the ownership of a file is changed, the group with which the file is associated changes to that of the new owner.

3.12 The Clist Program

utility:	**CLIST**
purpose:	This program displays files with page headings and line numbers
user access:	all users
summary:	clist [file-list]
arguments:	one or more file pathnames
options:	none

Description

The **Clist** program displays the files specified by its argument(s). When displaying a file, **Clist** numbers each line and adds a heading showing the filename and the time the file was last modified.

When called without an argument, **Clist** waits for input from the standard input.

3.13 The Clock Utility

utility:	**CLOCK**
purpose:	This program executes a command line and reports the time used to execute that command line.
user access:	all users
summary:	clock [-hec] command-line
arguments:	command line to be executed
options:	-h report time in the form hh:m:ss -e report time on standard error -c report time on controlling tty

Description

The **Clock** utility executes the given command line and reports the time, in seconds, used to execute that command line. The times reported are CPU time and real time.

Options

The **-h** option displays time used in the form "hh:mm:ss" instead of in seconds alone (the default).

The **-e** option causes the time usage to be displayed on the standard error device. This is useful if the output is redirected to **/dev/null**.

The **-c** option causes the time usage to be displayed on the controlling **tty**. The controlling **tty** is unaffected by redirection. This is useful if both standard output and standard error are redirected to **/dev/null**.

3.14 The Cmpasc Utility

utility:	CMPASC	
purpose:	This program compares two ASCII (text) files.	
user access:	all users	
summary:	cmpasc [-b #] [-m #] [-lrt] file1 file2	
arguments:	2 filenames	
options:	-b #	memory size, in bytes
	-l	print line numbers
	-m #	match this many lines
	-r	ignore RETURN preceding LINEFEED
	-t	expand TAB characters

Description

The **Cmpasc** utility compares two ASCII (text) files and displays differences.

Cmpasc displays lines from the first file (**file1**) with corresponding lines from the second file (**file2**).

If there are too many unmatched lines in a file (more than can be stored in the allotted memory), the **-b** option can be used to increase allotted memory.

Options

The **-b** option defines the amount of memory set aside for storing unmatched lines. (The default value is 32,768 bytes per file.)

The **-l** option adds line numbers to the display.

The **-m** option, followed by a number, defines how many successive lines from one file must match the corresponding lines in the second file to be considered a match. (The default value is 3.)

The **-r** option ignores a RETURN character preceding a LINEFEED. Thus, a line ended by the LINEFEED character alone is equal to a line ended by a RETURN-LINEFEED pair.

The **-t** option expands TAB characters before comparing lines.

Example:

```
jim[1] cmpasc -l fileone filetwo
-----> fileone
26 This file is sample file one.
```

```
-----> filetwo
26 This file is sample file two.
```

Notes

The **Cmpasc** utility compares characters with the parity bit reset. As a result, the bytes 0x8D and 0x0D are considered equal.

3.15 The Compare Utility

utility:	**COMPARE**
purpose:	This program compares two files.
user access:	all users
summary:	compare [-t] file1 file2
arguments:	2 filenames
options:	-t terse

Description

The **Compare** program compares two files and reports differences in length and content.

Compare lists differences between the files on a byte-by-byte basis. It displays an address in hexadecimal, then the byte in the first file at that address, followed by the corresponding byte in the second file. **Compare** does not adjust for offset, should one file lack one or more bytes in the middle (e.g., if part of a file was deleted).

Options

The **-t** option suppresses the list of differences. When this option is used, only a message is displayed to indicate whether the files are the same or different.

3.16 The Config Utility

utility:	**CONFIG**
purpose:	configures memory for any Cromix-Plus .bin program
user access:	all users
summary:	config filename [memory-size[kd]]
arguments:	Cromix-Plus **.bin** filename memory size, followed by: k memory size in kilobytes (the default) d memory size in decimal number of bytes

Description

The **Config** utility allows users to select the memory size that the Cromix-Plus Operating System allocates for any **.bin** program. The filename specified can be any Cromix-Plus **.bin** file (with or without the **.bin** filename extension). Memory size is assumed to be in kilobytes, unless followed by **d** to denote a decimal number of bytes. Although a valid memory size can range from 8K to 16000K, it will be rounded up to the nearest 4K.

If a filename is given, but no memory size, **Config** will report the number of kilobytes currently allocated to that file.

3.17 The Convert Utility

utility: **CONVERT**
purpose: Converts ASCII to EBCDIC and other transformations

user access: all users

summary: convert [-faslutbprge] [-i isize] [-o osize] [[inputfile] outputfile]

arguments: optional input filename
optional output filename

options:

-f	input lines are fixed length
-a	convert to ASCII (from EBCDIC)
-s	strip trailing blanks
-l	convert to lower case
-u	convert to upper case
-t	expand TAB characters
-b	compress spaces in TAB characters
-p	pad output line with blanks
-r	insert CR in front of NL
-g	output lines are fixed length
-e	convert to EBCDIC (from ASCII)
-i #	input line size
-o #	output line size

Description

The **Convert** utility, reads lines from an input file, transforms those lines as specified, and writes the converted lines to an output file. The available options determine how the input lines are transformed; the arguments, if any, define the input and output file names.

With no arguments, the **Convert** utility reads lines from standard input and writes the converted lines to standard output. If one argument is given, it is taken as the input filename (transformed lines are again written to standard output); if two arguments are given, they are taken as input and output filenames, respectively. In most cases a device file can be substituted for the input and/or output files.

As usual, a dash "-" substituted for the input or output filenames represents the standard input or standard output, respectively.

The **Convert** utility has no provisions for such functions as tape positioning. Use the **Mode** utility to perform such functions before calling the **Convert** utility. Since the general **Rcopy** utility can be piped into the **Convert** utility (or the output of the **Convert** utility can be piped into **Rcopy**), the **Convert** utility needs no specialized knowledge of I/O devices. It is best understood as a filter that transforms standard input to standard output.

Options

The **-f** option indicates that all input lines have a fixed length (determined by the **-i** option); without this option, lines are terminated by a "**\n**" character.

The **-a** option converts EBCDIC input to ASCII output. EBCDIC characters with no convenient ASCII counterpart are changed to spaces. Since end-of-line characters (**\n**) are not recognized, the **-f** option must be used with the **-a** option.

The **-s** option discards the trailing blanks in all input lines.

The **-l** option converts upper case characters to lower case characters.

The **-u** option converts lower case characters to upper case characters.

The **-t** option replaces each TAB character with enough spaces to reach the next TAB position. The TAB positions are 1, 9, 17, and so on.

The **-b** option compresses multiple spaces into TAB characters, based on TAB positions 1, 9, 17, and so on.

The **-p** option pads output lines with blanks to match the line size set by the **-o** option.

The **-r** option terminates each output line with **\r\n** rather than with **\n** alone. The **-r** option has no effect if the **-g** option is set.

The **-g** option indicates that all output lines are of fixed length (determined by the **-o** option), and have no line terminators (intended for use with the **-p** option).

The **-e** option converts ASCII input to EBCDIC output.

The **-i** option followed by a number (**i_size**) defines the size of the input lines. If the **-f** option is set, each input line will be exactly **i_size** characters long; if the **-f** option is not set, the **-i** option defines the maximum input line size (line terminators **\n** or **\r\n** are not counted). Without the **-i** option, input lines default to 80 characters (if the **-f** option is set) or to 512 characters (the **-f** option is not set). Input lines that are longer than the specified **i_size** are artificially broken into multiple lines, so a small **-i** value cannot be used to truncate long lines. To truncate long output lines, use an adequate **i_size** with the **-o** option.

The **-o** option followed by a number (**o_size**) defines the size of output lines. If the **-g** option is set, each output line will be exactly **o_size** characters long; if the **-g** option is not set, the **-o** option defines the maximum output line size (line terminators **\n** or **\r\n** are not counted). Without the **-o** option, output lines default to 80 characters (if the **-g** option is set) or to 512 characters (if the **-g** option is not set). Output lines that are longer than the specified **o_size** are truncated.

Notes

Lines are processed one at a time, in ten steps:

1. A line is read into the input buffer. If the **-f** option is set, then exactly **i_size** (defined by the **-i** option) characters are read; if the **-f** option is not set, reading of the line proceeds character by character until one of the following happens:

   - the \n character is read
   - the \0 character is read
   - the input buffer contains **i_size** characters.

   Any \r characters encountered are discarded. The terminating character is not stored in the buffer.

2. If the **-a** option is set, the characters in the input buffer are transformed from EBCDIC to ASCII.

3. If the **-s** option is set, the trailing blanks in the input buffer are discarded.

4. If the **-l** option is set, all upper case characters in the input buffer are changed to lower case characters.

5. If the **-u** option is set, all lower case characters in the input buffer are changed to upper case characters.

6. The characters in the input buffer are moved to the output buffer. If the **-t** option is set, all TAB characters are expanded during this process; if the **-b** option is set, multiple spaces (in the appropriate positions) are replaced by a TAB character. If neither **-t** not **-b** are set, the input characters are simply moved to the output buffer. In all cases, characters exceeding the **o_size** are discarded.

7. If the **-p** option is set, the output line is padded with blanks to **o_size** characters.

8. If the **-e** option is set, the characters in the output buffer are converted from ASCII to EBCDIC.

9. If the **-g** option is not set, a \n character (or a \r\n pair if the **-r** option is set) is added after every **o_size** characters in the output buffer.

10. All characters in the output buffer are written to the output file.

Examples

The first example creates a compressed version of the input file **text** (extra spaces are converted to TABs where possible) in the output file **text.new**. Lines are less than 80 characters each.

```
jim[1] convert -b text text.new
```

The next example creates an ASCII version of the EBCDIC input file **cobol.ibm** (containing card images, 80 characters each, of a cobol program) in the output file **cobol.asc.**

```
jim[1] convert -fas cobol.ibm cobol.asc
```

As the previous example, if the input is on 9-track tape:

```
jim[1] mode tp rewind
jim[2] convert -fas /dev/tp cobol.asc
```

As the previous example, if the input is on cartridge tape:

```
system[1] rcopy /dev/ftcd | convert -fas - cobol.asc
```

The last example creates a truncated (72 characters per line) version of the FORTRAN input file **prog.xxx** (120 characters per line terminated by **\n**) in the output file **prog.for**. The first half of the command truncates the output lines at 72 characters; the second half strips all trailing blanks.

```
jim[1] convert -o 72 prog.xxx | convert -s > prog.for
```

3.18 The Convobj Utility

utility:	**CONVOBJ**
purpose:	This program converts a **.obj** file to a .o68 file
user access:	all users
summary:	convobj [-jms] filename
arguments:	pathname of a **.obj** file
options:	-j build jump table -m write map -s create a single module

Description

The **Convobj** utility converts the **.obj** files generated by the C compiler/code generator to **.o68** format to be used with the **link68** linker. The conversion is intended for the cases when a user wants complete control over the linking process. The user must supply his own libraries and his own main module (probably written in assembler). For each module the **Convobj** utility creates up to three **psects**: CODE **psect** with REA and EXE attributes contains the code, IDATA **psect** with REA and WRI attributes contains initialized data, and UDATA **psect** with REA, WRI, and UNI attribute contains the uninitialized data. Global variables are entry points, not common **psects**. All percent signs (%) in the names are changed to dollar signs ($) because the Assembler does not allow the use of the percent sign in the names. Static functions are moved to the global module (with the name being the original file name). Static functions are not entry points. They are accessed through the global module entry point. The code of each module, including the jump table, is limited to 32 K bytes.

Options

The **-j** option generates a jump table to access the external functions. This ensures that the **Link68** utility will not generate the overflow errors. The **-j** option is not needed if the generated **.bin** program is small enough. Only if the **Link68** utility reports word overflow errors the **-j** option should be used to generate long references to external functions.

The **-m** option causes an entry point map to be generated on a file with the extension **.cmp**. The entry point map might be useful in case static functions are used, to find their location in the global module.

The **-s** option squeezes all the modules in one file into a single module. Non-static functions still have entry points, but those entry points are all located in the same CODE **psect**--the only one.

Example

```
jim[1] convobj -sm test
```

will convert **test.obj** file to **test.o68** file. The **test.o68** file will contain a single module with all functions concatenated together. The entry point map will be written to the file **test.cmp**.

3.19 The Copy Utility

utility:	**COPY**	
purpose:	This utility copies a file.	
user access:	all users	
summary:	copy	[-dftv] source-file destination-file [-dftv] file-list dirname
arguments:	two single file pathnames	
	or	
	one or more file pathnames	
	and	
	a directory pathname	
options:	-d	directory and device files ok
	-f	force
	-t	time
	-v	verbose

Description

The **Copy** utility copies one or more files. It does not alter the source file(s).

In its simplest format, copy duplicates file abc as file **xyz**, with both files residing in the current directory:

```
jim[1] copy abc xyz
```

To copy to or from a directory other than the current directory is more complex:

```
jim[1] copy abc /usr/fred/xyz
```

Here the pathname of the destination file is specified. The file **abc** exists in the current directory. It is being copied to the directory **/usr/fred**, and its name is to be **xyz** in that directory.

In the command:

```
jim[1] copy abc /usr/fred
```

the pathname of the destination directory is specified. The file **abc** exists in the current directory and is being copied to the directory **/usr/fred** without having its name changed.

The following form of the command can be used to create copies of all C language programs in the directory **/usr/archives**:

```
jim[1] copy *.c /usr/archives
```

This **copy** command copies all files in the current directory with filenames ending in **.c** to the directory **archives**. The files maintain their original names.

Options

The **-d** option allows directory and device files to be copied. If this option is not used, directory and device files are not copied. For example, a command such as:

```
jim[1] copy -d /dev/tty2 data
```

can be used to transfer all characters typed at terminal 2 into the file named **data** until a terminating character is received. The terminating character for console devices is CONTROL-Z.

The **-f** option makes the copied file overwrite an existing file with the same pathname. If this option is not specified and another file exists with the destination pathname, an error is reported.

The **-t** option causes a file to be copied only if:

1. The file does not exist in the destination directory; or

2. The source file has been modified more recently than the destination file. This comparison is performed on a file-by-file basis.

The **-v** option displays the name of each file as it is copied.

Notes

With the exception of "time dumped," which is automatically zeroed, files are copied with ownership and times preserved. If the system clock is reasonably accurate, the **-t** option can be very useful.

3.20 The Cptree Utility

utility:	**CPTREE**
purpose:	This program copies a tree.
user access:	all users
summary:	cptree [-ftv] source destination [file-list]
arguments:	source directory
	destination directory
	optional file list
options:	-f force
	-t time
	-v verbose

Description

The **Cptree** utility copies the source directory, and all its descendant directories and files, to the destination directory. Existing links within the source directory are preserved.

If a file list is specified, only files whose names match at least one of the names in the list are copied. Ambiguous filenames enclosed in quotation marks may be included in the file list.

Options

The **-f** option causes the copied files to overwrite any file with the same pathname. If this option is not invoked and another file exists with the destination pathname, an error is reported.

The **-t** option causes a file to be copied only if:

1. the file does not exist in the destination directory, or
2. the source file has been modified more recently than the destination file. This comparison is performed on a file-by-file basis.

The **-v** option causes display of the name of each file as it is copied.

Notes

With the exception of "time dumped," which is automatically zeroed, files are copied with ownership and times preserved. If the system clock is reasonably accurate, the **-t** option can be very useful.

3.21 The Crc Utility

utility:	**CRC**
purpose:	This program computes the CRC of a file.
user access:	all users
summary:	crc [-c value] file and/or directory list
arguments:	list of files and/or directories
options:	-c value append two bytes to make CRC equal to "value"

Description

The Crc utility will process every ordinary file given as an argument. If a directory pathname is given as an argument, all files in that directory will be processed (recursively). Device files are ignored.

When a file is processed, its CRC is computed and printed. The CRC polynomial used is:

```
x**16 + x**12 + x**5 + 1
```

Options

The -c option causes two bytes to be appended at the end of each file. These two bytes are computed so that the CRC of the modified file will be equal to the given value. The CRC value given must be in the range

```
0 <= value <= 65,535
```

Note that appending two bytes to a file might cause the file to be useless, so that the -c option may be used only under very special conditions.

3.22 The Create Command

Shell command:	**CREATE** or **CRE**
purpose:	This command creates a file.
user access:	all users
summary:	cre file-list
arguments:	one or more pathnames
options:	none

Description

The **Create** command is used to create one or more files.

The files are zero bytes in length and have default access privileges. They are owned by the user who created them and are in the domain of their creator's group.

If the specified pathname already exists, an error is reported.

Notes

This command makes a standard data file. Refer to the **Makdir** command or the **Makdev** utility program if you need to make a directory or device file.

3.23 The Crogen Utility

utility:	**CROGEN**
purpose:	This program generates a Cromix-Plus Operating System.
user access:	privileged user
summary:	Crogen pathname [sysdef]
arguments:	pathname of new system file optional system definition file
options:	-d include system debugger

Description

The **Crogen** utility generates a new operating system by creating a configuration file based on the data in a system-definition file.

The file **/gen/sysdef** is provided for this purpose. The **sysdef** file can be used as is or with possible modifications. A user's own system-definition file may be substituted.

After **Crogen** creates the configuration file, **Link68** generates the system file.

3.24 The Cron Utility

utility:	**CRON**
purpose:	This program executes command files from the directory /usr/cron/crontabs.
summary:	/etc/cron &
arguments:	none
options:	none

Description

The **Cron** utility resides in the /etc directory. Normally, it is executed from the **/etc/startup.cmd** file by executing the command

/etc/cron &

The **Cron** utility will inspect the directory **/usr/cron/crontab**. The files in this directory are maintained by the **Crontab** utility. If the file **/usr/cron/log** exists, the **Cron** utility will use it to record every executed command. The logging mechanism is disabled by removing the file **/usr/cron/log**.

Other files of interest are:

/usr/cron/cron.allow	The names of users that are allowed to use the **Crontab** utility.
/usr/cron/cron.deny	The names of users that are denied the use of the **Crontab** utility.
/usr/cron/cron.pid	The file where the **Cron** utility records its own process ID number.

Please see the description of the **Crontab** utility for additional details.

3.25 The Crontab Utility

utility:	**CRONTAB**	
purpose:	This program lists, removes, and changes the crontab file for the user.	
user access:	all users	
summary:	crontab [-lr] [crontab-file]	
arguments:	optional user crontab file	
options	-l	list current user crontab file
	-r	remove current user crontab

Description

The **Crontab** utility copies the specified file, or standard input if no file is specified, into a directory that holds all users' crontabs. The **-r** option will remove a user's crontab from the crontab directory. **Crontab -l** will list the crontab file for the invoking user.

A user is permitted to use the **Crontab** utility if his name appears in the file **/usr/cron/cron.allow**. If that file does not exist, the file **/usr/cron/cron.deny** is checked to determine if the user should be denied access to the **Crontab** utility. If neither file exists, only a privileged user can use the **Crontab** utility.

A crontab file consists of lines of six fields each. The fields are separated by spaces or tabs. The first five fields are integer patterns that specify the following:

minute (0-59)
hour (0-23)
day of the month (1-31)
month of the year (1-12)
day of the week (0-6 with 0=Sunday)

Each of these patterns may be an asterisk (meaning all legal values), or a list of elements separated by commas. An element is either a number, or two numbers separated by a minus sign (meaning an inclusive range).

The following examples of fields refer to the **hours** field. The other four fields are built according to the same rules:

*	any hour
9,17	at 9 AM and 5 PM
9-17	any working hour
18-8	except working hours !
0,4,8,12,16,20	every fourth hour starting at midnight

If any of the five time fields is illegal, the command will never execute.

The sixth field of a line in a crontab file is a string that is executed at the times that match all listed conditions. A percent character in this field (unless escaped by "\") is translated to a newline character. Only the first line (up to the % or end of line) of the command field is executed by Shell. The other lines are made available to the command as standard input.

If the crontab line is terminated by the "\" character, another line is read and appended to the previous one, up to 2048 characters.

The **Shell** command is invoked from the user's home directory. The standard input channel is **/dev/null** unless the % character is used to provide input lines. The standard output and standard error are redirected to the controlling **tty** device as determined from the **/etc/who** file. If the **/etc/who** entry is not found, the standard output and standard error are redirected to **/dev/null**.

Users can redirect their input and/or their output elsewhere. In particular, the standard output and/or standard error can be piped into the **Mail** utility.

3.26 The Daemon Command

Shell command:	**DAEMON**
purpose:	This command prints the files spooled by the spool utility.
user access:	all users
summary:	daemon device-name
arguments:	pathname of the device to be run by Daemon
options:	none

Description

The **Daemon** command is usually run detached. Typically, it is started by the **Spool** utility, though a user can start it himself.

The **Daemon** will examine the **/usr/spool** directory and select all files, one at a time, that are to be printed on the specified device. The files are selected in order of increasing priority. Files that were spooled with a forms number different than the current forms number are skipped. The **Daemon** keeps running until it runs out of eligible files, or until killed by the **Spool** utility or by the user. Only one **Daemon** can be running per device. If another **Daemon** is started to run the same device while the first one is still active, the new **Daemon** will die immediately.

Example:

```
jim[1] daemon /dev/typ &
```

This command starts a detached **Daemon** to print the files spooled on **typ** printer.

3.27 The Day Utility

utility:	**DAY**
purpose:	This program executes a command on the day specified.
user access:	all users
summary:	day [day-of-the-week command-line]
arguments:	day of the week
	command line
options:	none

Description

The **Day** utility executes a command on the day specified. **Day** checks the system clock for the specified day. This program is useful in applications that require certain tasks be done on certain days of the week.

Notes

When used without an argument, **Day** displays the name of the current day.

Example:

The following command line (if executed on Wednesday) will remind you of a weekly Wednesday meeting.

```
jim[1] day wed echo "This is Wednesday, remember your meeting"
```

If the command is executed on any other day, no action is taken. It is recommended that these commands be included in startup command files or other command files executed daily.

3.28 The Dcheck Utility

utility:	**DCHECK**
purpose:	This program verifies the integrity of a file system.
user access:	all users
summary:	dcheck [-s] [devname(s)]
arguments:	optional device name(s)
options:	-s salvage directory structure

Description

The **Dcheck** utility verifies the integrity of a file system's internal directory structure. If possible, **Dcheck** with the -s option should be run on an unmounted file system. If the file system that needs to be fixed is the root, **Dcheck** should be run by itself, with no other users or tasks running concurrently. If another task is writing to the disk, the results of **Dcheck** may be incorrect.

If the -s option is used while another task or user is using the disk, the directory on the disk may be irreparably damaged.

MESSAGES RETURNED BY DCHECK

Cannot read super block
The super block cannot be read.

Insufficient memory
There is not enough available memory to run **Dcheck**. Either free additional memory, or create a new disk with fewer inodes. (Use **Cptree** to transfer the contents of the disk to the new disk.)

Inode xxxxx, Cannot read inode
A disk I/O error occurred while trying to read the specified inode.

Inode xxxxx, error reading directory
A disk I/O error occurred while trying to read a directory.

Inode xxxxx, file "ffffff" has bad inode
Inode xxxxx is a directory inode that contains an active file with an impossible inode number. Use **Ncheck** to locate the directory. Then delete the offending file, and run **Dcheck** with the -s option.

Inode xxxxx, directory with more than 1 parent
A directory is linked to more than 1 parent directory. Use **Ncheck** to locate the names of the files, and delete all but one link. Then run **Dcheck** with the -s option.

Inode xxxxx, directory with wrong parent

This error indicates the inode is pointing to the wrong parent. Use the **Dcheck** utility with the -s option to correct this error.

Inode xxxxx, bad link count xxxxx, should be xxxxx
The number of names pointing to this inode from various directories is greater or less than expected. Use **Dcheck** with the -s option to correct this error.

Inode xxxxx, more than 255 links
There are more than 255 names for this inode. Use **Ncheck** to find all the names. Delete some names to bring the total number of names to 255 or less. Then run **Dcheck** with the -s option.

Inode xxxxx, bad inode number in inode
Each inode contains its own inode number. This error means the inode specified has the wrong number. Use **Dcheck** with the -s option to correct this error.

Inode xxxxx, unallocated inode with xxx links
Although this inode is unallocated, names point to it. Use **Ncheck** to find these names, then delete them.

Inode xxxxx, allocated inode with 0 links
This inode is still allocated, though there are no names for it. Use **Dcheck** with the -s option to correct this error.

Inode xxxxx, bad directory entry count
This inode is a directory. The number of directory entries in the inode differs from the actual number of directories. Use **Dcheck** with the -s option to correct this error.

End of dcheck (This is the last message)

The program has finished executing.

Options

The -s option fixes problems reported by **dcheck**. The program corrects an incorrect inode number when:

1. The inode is allocated;

2. The inode link is nonzero; and/or

3. The inode is being pointed to (i.e., is in use).

The program does not correct an incorrect inode number if the inode is unallocated.

Notes

Immediately after running **Dcheck** with the -s option, run **Icheck** with the -s option. After both programs are run, the system must be rebooted. Refer to the **Boot** utility for additional information.

It is not necessary to reboot if the -s option is not used or if **Dcheck** is run on an unmounted device.

3.29 The Ddt Utility

utility:	**DDT**	
purpose:	This program can be used to trace user programs.	
user access:	all users	
summary:	ddt [options] command arguments	
arguments:	full pathname of the program to be traced	
	arguments to the program to be traced	
options:	-i path	path name of STDIN for debugged program
	-o path	path name of STDOUT for debugged program
	-e path	path name of STDERR for debugged program
	-t term	terminal name (4 characters, Term variable)

Description

The **Ddt** utility should be called in the following manner:

jim[1] **ddt /bin/ls.bin -lrt**

The **Ddt** utility will fork the **/bin/ls.bin** program, the display will appear something like:

First = 052000
Next = 057000

indicating that the program was loaded at address 052000H and extends up to the address 057000H.

This initial information is followed by the **Ddt** prompt:

->

Whenever this prompt appears, a **Ddt** command can be entered. A command consists of a one or two character command abbreviation, followed by arguments. The number of arguments depends on the command used. Arguments are expessions which are described later. Multiple arguments must be separated by spaces or commas. Spaces are not required unless a command might be misinterpreted. Upper and lower case letters are indistinguishable.

In the description that follows optional arguments are shown in brackets.

Breakpoint

b [address, ...]

If the command is given with no arguments, the **Ddt** utility will list the declared breakpoints. If one or

more arguments are given they are taken as addresses of breakpoints to be added to the list. At no time can the **Ddt** utility handle more than eight breakpoints.

Breakpoint Remove

bx [address, ...]

If the command is given with no arguments, the **Ddt** utility will remove all declared breakpoints. If one or more arguments are given, they are taken as addresses of breakpoints to be removed from the list.

Call Trace

c

This command will trace one instruction unless the instruction happens to be a BSR or JSR instruction. In these cases a temporary breakpoint will be installed after the instruction and the Ddt utility will come back to life where the subroutine returns.

Display Memory

dm [address [lines]]

Display memory command can have zero, one, or two arguments. With no arguments the **dm** command will list 4 lines of memory starting where the last **dm** command left off. If one argument is given, it is taken as the beginning address and 4 lines starting at that address will be printed. If the second argument is also given, it is taken as the number of lines to be printed. Each line shows 16 bytes. Displaying memory can be aborted any time by pressing any key.

Display Registers

d

The command will first display the **D** and **A** registers. The next line will start with the **Ddt** status, followed by the processor status register, program counter, and the disassembled instruction at the program counter.

Exit

ex

The traced program and the **Ddt** utility will be terminated.

Go

g [address]

The traced program continues execution with the current register state, including the program counter (**PC**). If an address is specified with the command, a temporary break will be installed at the specified address. The debugger will not be activated again unless an installed breakpoint is reached.

Hex calculator

h expression

The **Ddt** utility will display the value of expression both in hexadecimal and in decimal notation.

Ignore

i

If the traced program is aborted by a signal, the **Ddt** will show it immediately. At that point the user can use the **i** command to cause the signal to be ignored. The go command will then proceed as if there were no signal. If the **i** command is not issued the traced process will be killed by the signal.

List instructions

l [address [lines]]

The list command can have zero, one, or two arguments. With no arguments, the **l** command will list 16 instructions starting where the last list command left off. If one argument is given, it is taken as the starting address and 16 instructions starting at that address will be printed. If the second argument is also given, it is taken as the number of instructions to be listed. Listing can be aborted any time by pressing any key.

Substitute Registers

s [regvar]

In this case the optional argument is limited to a register name or the name of a **Ddt** variable. If such an argument is given, only the specified register will be changed. If no argument is given, **Ddt** will keep prompting for the replacement of the next register, circularly, until a period is entered. The allowed registers are all processor registers and the Ddt variables **v0**, **v1**, ..., **v7**, and **vb**. Variables can be set to arbitrary values and then used in expressions.

Substitute Bytes

sb [address]

The command will start substituting bytes in memory starting at the specified address (or where the previous substitute command left off if no address is given). The command is an iterative one. It displays the next byte and waits for the user response, which might be:

nothing; whereupon the Ddt will leave the byte unchanged and proceed to the next byte

an expression whose low order byte replaces the current one; proceed to the next byte

a (singly or doubly) quoted string which will replace a number of consecutive bytes

an illegal expression which will cause the same byte to be displayed again

a minus that will cause the debugger to backspace to the previous byte

a period which terminates the substitution

Substitute Longs

sl [address]

The command will start substituting longs in memory starting at the specified address (or where the previous substitute command left off if no address is given). The command is an iterative one. It then displays the next long and waits for the user's response, which might be:

nothing; whereupon the debugger will leave the **long** unchanged and proceed to the next **long**

an expression whose value replaces the current one; proceed to the next **long**

an illegal expression which will cause the same **long** to be displayed again

a minus that will cause the debugger to backspace to the previous **long** word

period which terminates the substitution

Substitute Words

sw [address]

The command will start substituting words in memory starting at the specified address (or where the previous substitute command left off if no address is given). The command is an iterative one. It displays the next word and waits for the user response, which might be:

nothing; whereupon the debugger will leave the word unchanged and proceed to the next word

an expression whose low order word replaces the current one; proceed to the next word

an illegal expression will cause the same word to be displayed again

a minus that will cause the debugger to backspace to the previous word

a period which terminates the substitution

Trace

t

A single instruction will be executed and then command will be returned to the debugger.

Shell

% [command]

If a **Shell** command is given following the "%" sign, the **Shell** command will be executed. If no command is given, an interactive Shell is created. When the interactive Shell terminates (**Exit** command or CNTRL-Z) the **Ddt** utility will resume.

The syntax of the expressions that can be used is described in BNF form:

```
<expression> ::= <term> | <expression> + <term> |
                          <expression> - <term>
<term> ::= <factor> | <term> * <factor> | <term> / <factor> |
                          <term> & <factor>
<factor> ::= <unit> | @ <factor>
<unit> ::= <number> | <regvar> | ( <expression> )
<number> ::= <hex number> | <dec number>.
<hex number> ::= <hex digit> | <hex number> <hex digit>
<dec number> ::= <dec digit> | <dec number> <dec digit>
<hex digit> ::= <dec digit> | a | b | c | d | e | f
<dec digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

<regvar> ::= <d register> | <a register> | <control register> |
<variable>
<d register> ::= rd0 | rd1 | rd2 | rd3 | rd4 | rd5 | rd6 | rd7
<a register> ::= ra0 | ra1 | ra2 | ra3 | ra4 | ra5 | ra6 | ra7
<control register> ::= sr | pc
<variable> ::= v0 | v1 | v2 | v3 | v4 | v5 | v6 | v7 | vb

Semantics.

The @ operator is the dereferencing operator:

value of

@x

is the value read from memory location **x**. Note that it is the highest priority operator, so parentheses might be required occasionally.

All numbers are haxadecimal by default unless followed immediately by a period. In the later case the number is decoded decimaly.

Variables **rd0, ..., pc** refer to processor registers. Variables **v0, ..., v7** are general purpose variables. **vb** is another general purpose variable initialized to the program base address. All variables can be changed by the **sr** command. Note that the **h** command can be used to examine them.

The options of the **Ddt** utility can be used to debug a program that is running from another terminal. A typical example might be

ddt -i /dev/qtty2 -o /dev/qtty2 -t 3102 ce.bin file

In this case the **Ce** program is debugged while it is running from the 3102 terminal hooked to **qtty2**. The **Ddt** utility itself is running from the original terminal. All **Ddt** dialogue goes through the original terminal, all **Ce** dialogue (except messages to STDERR which were not redirected) run through the **/dev/qtty2** terminal.

To use this feature, the **qtty2** terminal must be disabled from the ttys file otherwise the **Shell** and the debugged program will fight for input characters.

3.30 The Ddump Utility

utility: **DDUMP**

purpose: This program converts and copies a file from one device to another. It can handle direct physical I/O from devices such as the tape driver.

user access: all users

summary: ddump options

Option		Function
if=pathname		specify input file pathname
-i pathname		specified input file pathname
of=pathname		specify output file pathname
-o pathname		specified output file pathname
ibs=n		input block size
obs=n		output block size
cbs=n		conversion buffer size
cbufsz=n		conversion buffer size
iskip=n		skip the first n input blocks before starting to copy
oskip=n		skip the first n output blocks before starting to copy
icount=n		copy only n input blocks
conv=	ascii	convert EBCDIC to ASCII
	ebcdic	convert ASCII to EBCDIC
	ucase	convert alphabetic characters to uppercase
	lcase	convert alphabetic characters to lowercase
	strip	strip trailing blanks in the conversion buffer
	nostop	do not stop processing on an error (such as a file read error)
		Several conversions, separated by commas, may be specified in one argument.

Description

Ddump converts and copies data from one file or device to another. Since the input and output block sizes can be specified, it is useful for gaining access to devices that store data in raw form.

Conversions are done in the conversion buffer. Each block read from the input file is transferred to

the conversion buffer, one buffer at a time. The conversions specified are performed there before writing the result to the output file. For example, if the strip conversion option is specified, trailing spaces are stripped and a newline added before sending the result to the output file.

Example:

jim[1] **ddump if=/dev/tp1 of=file1 conv=ascii,lcase,strip**

This example causes input to be read from **/dev/tp1** and written to disk file **file1**. EBCDIC characters are converted to ASCII, uppercase to lowercase, and trailing blanks are not copied to **file1**. The end of the tape file is indicated by an EOF tape mark written when the tape was created.

Notes

The following is a list of default values for options.

input file	standard input
output file	standard output
conversion buffer	80 bytes
disk input buffer	512 bytes
disk output buffer	512 bytes
tape input buffer	8192 bytes
tape output buffer	8192 bytes

3.31 The Delete Command

Shell command:	**DELETE** or **DEL**
purpose:	This command deletes a file.
user access:	all users
summary:	del [-v] pathname(s)
arguments:	one or more pathnames
options:	-v verbose

Description

The **Delete** command removes a link to a file. If there is only one link to the file, the file is no longer accessible, and the space it occupied is made available.

Options

The **-v** option displays the name of each file as it is deleted.

Notes

To remove all links to a file, making it inaccessible, use the **Ls** command with the **-i** option to find the inode number of the file in question. Use that inode number as an argument to **Ncheck**, and find the names of all files linked to the file.

A directory may be deleted by specifying a directory pathname.

In order to delete a directory, it must not:

1. Contain any files;
2. Be the current directory for any user; or
3. Be the root directory of a device.

Examples:

In the following example, the file named schedule is deleted from the current directory.

```
jim[1] ls -m
```

```
  3,016      1 letter
    200      1 memo
  1,408      1 schedule

jim[2] del schedule
jim[3] ls -m

  3,016      1 letter
    200      1 memo
```

If there is more than one link to a file and one of the links is deleted, the file is no longer accessible through that link. The file remains on disk and is accessible through the remaining links.

The following example concerns itself with part of the **/dev** directory. Suppose that the device file **prt** is linked to the dot-matrix printer device file **lpt1**. In the first listing that follows, the link is shown by the **2** preceding each filename. When the file **prt** is deleted, the file **lpt1** remains intact, and the number of links is reduced to one.

```
system[1] ls -m

5:5  C      2 lpt1
5:5  C      2 prt
6:5  C      1 typ1

system[2] del prt
system[3] ls -m

5:5  C      1 lpt1
6:5  C      1 typ1
```

3.32 The Deltree Utility

utility:	**DELTREE**
purpose:	This program deletes a tree, including all files and subtrees.
user access:	all users
summary:	deltree [-a] pathname [file-list]
arguments:	pathname
options:	-a suppresses user verification

Description

The **Deltree** utility deletes all files and subtrees in the tree (directory) specified. Normally, **Deltree** prompts the user with the file or directory name, followed by a question mark. If the user types **y**, the file or directory is deleted; otherwise it is not. If the **-a** option is used, **Deltree** asks once whether the user really wants to delete the entire tree, instead of prompting for verification of each file. If the user types **y**, all files and subtrees are deleted. If "n" is typed, **Deltree** returns control to the Shell.

If **Deltree** is called from within the specified directory, the program will not allow the deletion of that directory. All the files must be deleted from a directory before the directory itself is deleted.

If the directory pathname is followed by a list of filenames, only the corresponding files will be deleted.

Options

After asking for verification, the **-a** option deletes all files and subtrees automatically.

3.33 The Directory Command

Shell command:	**DIRECTORY** or **D**
purpose:	This command displays the name of or changes the current directory.
user access:	all users
summary:	d [dir name]
arguments:	optional directory pathname
options:	none

Description

When given without an argument, the **Directory** command displays the pathname of the current directory.

Given with a directory pathname, the **Directory** command makes the specified directory the current directory.

3.34 The Diskinfo Utility

utility:	**DISKINFO**
purpose:	This program prints information from the hard disk label, the partition table, and the alternate track table.
user access:	privileged user
summary:	diskinfo devicename [id_line]
arguments:	device name optional identification line
options:	none

Description

The **Diskinfo** utility prints information from the label (on block 0) of the specified Cromemco hard disk, as well as the partition table and the alternate track table. The devicename argument should be the device name of the entire disk (e.g., std31 on drive 0; or esd63 on drive 1). The optional id_line argument is printed on the first line of the printout for identification purposes.

The "Location of alternate tracks" is the starting cylinder number of the alternate tracks. The partition table (if any) is printed after the disk parameters, followed by the alternate track table. For example, the command:

system[1] **diskinfo std31** "MAXTOR Hard Disk"

prints the following:

```
MAXTOR Hard Disk

Device /dev/std31 6:31  STDC Hard Disk  Aug-14-1987 10:53:39

Disk parameters
        Number of heads..................... = 15
        Number of cylinders................. = 1224
        Number of alternate tracks.......... = 60
        Location of alternate tracks..... = 632
        Number of sectors per track...... = 20
        Bytes per sector.................... = 512
        Start of write precompensation... = 1225
        Starting cylinder of disk ....... = 1

Partition #      Starting Cylinder  Number of cylinders  Size (Kbytes)
    0                    1                 196              29,400
    1                  197                 196              29,400
    2                  393                 196              29,400
    3                  589                  43               6,450
```

4	632	196	29,400
5	828	196	29,400
6	1024	150	22,500
7	1174	46	6,900
31	0	1220	183,000

Alternate track table

#	bad track		alt track		#	bad track		alt track		#	bad track		alt track	
	cyl	hd	cyl	hd		cyl	hd	cyl	hd		cyl	hd	cyl	hd
0	63	10	632	12	7	244	0	632	2	13	681	8	632	11
1	64	10	632	13	8	385	12	633	2	14	772	10	632	14
2	70	7	632	8	9	429	5	632	6	15	989	10	633	0
3	106	13	633	3	10	488	2	632	5	16	1017	5	632	7
4	124	7	632	9	11	668	7	632	10	17	1117	1	632	4
5	242	0	632	0	12	678	0	632	3	18	1218	11	633	1
6	243	0	632	1										

3.35 The Dump Program

utility: **DUMP**
purpose: This program displays a file in hexadecimal and ASCII.

user access: all users

summary: dump [-b #] [file-list]
[-e #]
[-k #]
[-s #]
[-o #]

arguments: one or more optional file pathnames

options:
-b first byte
-e last byte
-k first block
-s width
-o offset address

Description

The **Dump** program displays the file(s) specified by the pathname(s). When called without an argument, **Dump** waits for input from the standard input.

Dump displays any type of file. The file is displayed in hexadecimal with an ASCII equivalent to one side. All numeric arguments to the **Dump** utility are assumed to be decimal numbers unless followed by an **h** (for hexadecimal).

Options

The **-b** option allows the user to specify the first byte of a file to be dumped.

The **-e** option allows the user to specify the last byte of a file to be dumped.

The **-k** option allows the user to specify the first block to be dumped.

The **-s** option allows the user to specify the swath width of the dump.

The **-o** option causes a specified offset to be added to all addresses displayed by **Dump.**

Example:

```
jim[1] dump -b 1000h -e 5000h filename
```

This command dumps the file **filename** starting with the 1000th (hex) byte and ending with the 5000th (hex) byte.

3.36 The Ecc Program

utility:	ECC	
purpose:	This program manipulates the error-correcting memory controllers.	
user access:	privileged user	
summary:	ecc [-e] [-t #] [ON] [OFF]	
arguments:	Optional keyword ON or OFF	
options:	-e	examine memory controller
	-t #	reexamine every # seconds

Description

The **Ecc** program can enable, disable, or examine the error-correcting memory controllers. When called without options or arguments, **Ecc** reports the controller status (enabled or disabled).

If **Ecc** is called with the **-e** option, it will examine the memory controllers, report any recorded errors to the standard output, and clear the error status. If **Ecc** is called with the **-t** option, the memory controllers will be scanned periodically as defined by the numerical argument following **-t**.

Options

The **-e** option requests the **Ecc** program to scan the controllers and report errors.

The **-t** option, followed by a number, defines the number of seconds between two consecutive scans.

Example:

```
system[1] ecc on
system[2] ecc -t 30 &
```

This sequence of commands enables the error-detection hardware and then scans it every 30 second as a detached process. The errors will be reported on standard output.

3.37 The Echo Program

utility:	**ECHO**
purpose:	This program echoes its arguments to the terminal.
user access:	all users
summary:	echo [-en] text
arguments:	any text
options:	-e send to STDERR -n do not print newline

Description

The **Echo** program echoes its arguments. Text may be enclosed within single or double quotation marks to insure correct interpretation by the Shell.

Options

The **-e** option echoes arguments to the standard error channel.

The **-n** option suppresses the echo of a newline.

Note

The **Echo** utility, like most of the utilities, objects to illegal options. The example:

```
jim[1] echo -1
```

does not echo the string "-1". The **Echo** utility interprets the string "-1" as an option, and such an option is illegal for the **Echo** utility. The intended result can be produced by the command:

```
jim[2] echo -- -1
```

This is particularly convenient for a command sequence like:

```
jim[3] ls nonexistent
jim[4] echo -- #err
```

3.38 The Ed Utility

utility:	Ed
purpose:	This program is used to edit files.
user access:	all users
summary:	ed [-cn] filename [filename]
arguments:	name of file to be edited
options:	-c CR-LF pair used as line terminator -n LF only used as line terminator

Description

The **Ed** utility program enables the user to edit files.

When called without an option or with the **-c** option, the **Ed** program (**/bin/ed.bin**) creates output files with a CR-LF (carriage RETURN-linefeed) pair at the end of lines. When called with the **-n** option, only a LF character is used as a line terminator. Such files can then be used with the UNIX Operating System.

Cromix-Plus utilities understand lines terminated by either the LF character or CR-LF pair. A few utilities, however, were originally written for the CDOS Operating System. These programs (notably, those using older versions of Z80 assembler) might require the CR-LF pair as the line terminator.

The **Sc** program (**/cmd/sc.cmd**) can be used to automatically call **Ed** with the **-n** option.

If you prefer to alter the **Ed** program so **-n** instead of **-c** is the default, patch the file **ed.bin** with a zero at location 0FAH. This patched version will be loaded slightly faster than **sc.cmd**. You may wish to call the patched version **sc.bin**, in which case, the file **sc.cmd** may be discarded.

3.39 The Esdload Utility

utility:	**ESDLOAD**
purpose:	This program loads a program into an ESDC board.
user access:	Privileged user
summary:	esdload filename device
arguments:	filename of program to be loaded
	ESDC device to be loaded
options:	none

Description

The **Esdload** utility loads a file into an Esdc board. (SCSI and hard disk controller). This utility is normally used to load the Esdc firmware (**/etc/esdcfirm**) into an Esdc. Esdload is found in the **/etc** directory.

Note

Only one device per controller need be downloaded, but that device can be either an SCTP device or an ESDC device.

3.40 The Exit Command

Shell command:	**EXIT** or **EX**
purpose:	This command exits from a Shell.
user access:	all users
summary:	ex [return-value]
arguments:	Optional value returned to the parent process
options:	none

Description

The **Exit** command exits from a shell. If no higher level shell is active, the Cromix-Plus Operating System logs the user off the system. If there is a higher level shell active, it receives from the exiting shell a return value as defined in the following paragraph.

If the **Exit** command is used without an argument (or if a command file runs to end-of-file), the return status will be **-1** if an error such as a syntax error was detected anywhere in the command file. If there was no error detected, the return value will be the value returned by the last executed command. An **Exit** command with an argument (a value) can be used to return that value.

Return values are used by the "**if -err**" command. Each shell keeps track of the value returned by the last command, and this value is used to determine the truth of the "**if -err**" condition. Any nonzero value indicates an error condition.

Example:

Here is the contents of a possible command file to compile (and, optionally, link) a number of C source files. A user-defined **-l** option requests linking. The command file will abort if any step returns an error.

```
if "#1" = "-l" shift                    % Get rid of -l
if "#1" = "" goto huh                   % If no arguments

%compile                                % Compile all source files
if "#1" = "" goto endcompile            % If none is left
c #1                                    % C compiler
if -err goto err                        % If errors in compilation
code #1.i #1.obj                        % Generate code
if -err goto err                        % If errors in code generation
del #1.i #1.scr >* /dev/null            % Delete scratch files
shift                                   % Next argument
goto compile
```

```
%endcompile

rewind
if "#1" != "-l" exit                    % No linking requested
shift                                   % Get rid of -l
crolinker #* /usr/lib/clib /usr/lib/paslib
exit

%err                                    % In case of errors
 del #1.i #1.scr >* /dev/null           % Delete scratch files
 exit 1                                 % Return error

%huh                                    % If no arguments
 echo "Huh?"
 exit 1                                 % Return error
```

If both "**exit 1**" commands were replaced by "**exit**" command, the return value would be the value returned by the last command (**Delete** or **Echo**).

3.41 The Find Utility

utility: **FIND**
purpose: This program locates files.

user access: all users

summary: find pathname [!] expression

arguments: pathname

[!]

expression(s)

options: File specifiers:
- -name
- -type x
- -links n
- -user name or number
- -group name or number
- -size n
- -blocks n
- -mtime n

Action Specifiers:
- -exec command-line
- -ok command-line
- -print

Logical Operators:
- -a
- -o

Description

The **Find** utility locates a file. The pathname is the pathname of the tree, directory, or file to be searched, and the expression is the string to be found and what is to be done with it.

Expressions are combinations of file criteria and operations. Refer to the following list.

Parentheses may be used to change the order of evaluation of the items in the **Find** expression. Used with parentheses, the expression must be enclosed within quotation marks so that the Shell passes them to the **Find** utility.

When one of the action specifiers is used to execute a program, the return value of that program can be evaluated and used within the expression.

The ! operator may precede the expressions to negate the sense of the tests.

Options

File Specifiers

-name file-list

The file-specifying keyword **name** is followed by a list of one or more unique or ambiguous filenames. If an ambiguous filename is used, it must be enclosed within quotation marks. The **Find** utility finds all files that match the file list.

-type	b	block device"
	c	character device
	f	file
	d	directory

The file-specifying keyword **type** is followed by either **b**, **c**, **f**, or **d**, as shown. The **Find** utility finds all files of that type.

-links n

The file-specifying keyword **links** is followed by a number, n. The **Find** utility finds all files with that number of links. If the number is preceded by a plus sign, all files with more than that many links are found; if a minus sign is used, all files with fewer than n links are found.

-user name

The file-specifying keyword **user** is followed number by a user name or number. The **Find** utility finds all files owned by the specified user.

-group name

The file-specifying keyword **group** is followed number by a group name or number. The **Find** utility finds all files owned by the specified group.

-size n

The file-specifying keyword **size** is followed by a number, **n**. The **Find** utility finds all files of the specified size, in bytes. If the number is preceded by a plus sign, all files with more than that number of bytes are found; if a minus sign is used, all files with fewer than **n** bytes are found.

-blocks n

The file-specifying keyword **blocks** is followed by a number, **n**. **The** Find utility finds all files using that number of blocks (actual number of blocks occupied by the file). If the number is preceded by a plus sign, all files occupying more than the specified number of blocks are found; if a minus sign is used, all files with fewer than n blocks are found.

-mtime n

The file-specifying keyword **mtime** is followed by a number, **n**. The **Find** utility finds all files modified **n** days ago. If the number **n** is preceded by a plus sign, all files modified **n** or more days ago are found; if a minus sign is used, all files modified fewer than **n** days ago are found.

Action Specifiers

-exec command-line

The action-specifying keyword **exec** is followed by a command line. This may be any valid command line, that is, any line that can be entered in response to the Shell prompt. This command line is then executed each time the **Find** utility finds a file meeting the find criteria. A pair of braces ({}) may be placed within the command line. They will be replaced by the name of the file found.

-ok command-line

The action-specifying keyword **ok** is used in the same manner as **exec**. When **ok** is used, the **Find** utility prompts the user prior to executing each command line. The user may respond with a **y** to execute the command line or **n** to prevent its execution.

-print

The action-specifying keyword **print** is used to display the pathnames of files found.

Logical Operators

-a The **-a** operator is used to logically **AND** two items in the **Find** expression.

-o The **-o** operator is used to logically **OR** two items in the **Find** expression.

Notes

The expression used with the **Find** command is evaluated from left to right. Items to be found and actions to be performed may be combined logically by use of the **-a** and/or **-o** logical operators. Either operator combines the sum of the expression to its left with the subsequent item in the expression. For example:

system[1] **find / -name ted -a -print**

system[2] **find / -name ted -o -name mary -a -print**

The first example finds all files with the filename **ted** and prints the pathnames of these files. If the print instruction is omitted, all of the correct files are found, but no action is taken: their names are not

displayed. The second example demonstrates the use of the logical OR. All files with the filename **ted** OR **mary** are found and their pathnames printed.

Examples:

The following example finds all subdirectories of the current directory, then executes an ls command with the **-d** and **-e** options.

```
jim[1] find . -type d -a -exec ls -de {}
```

The next example finds all entries with a .c extension, then lists the entry with the **-l** option.

```
system[1] find / -name "*.c" -a -exec ls -l {}
```

3.42 The Fixsb Utility

utility:	**FIXSB**
purpose:	This command file restores the Superblock.
user access:	privileged users
summary:	fixsb device
arguments:	device name to be corrected
options:	none

Description

The **Fixsb** utility file restores the Superblock, should it be destroyed accidentally. This command file has the same function as the **Makfs** utility used with the **-r** option, but without the possible risks associated with running an older version of the **Makfs** utility.

After restoring the Superblock, the **Fixsb** command automatically runs Icheck, to check inodes in the file.

Notes

Fixsb is only to be used on disks whose file systems were created with the default number of inodes. Refer to the **Makfs** utility for additional information.

3.43 The Flush Utility

utility:	**FLUSH**
purpose:	Writes system I/O buffers to disk
user access:	all users
summary:	flush # &
arguments:	number of seconds
options:	none

Description

The **Flush** utility is a system-maintenance program that flushes (i.e., writes to disk) system buffers by executing the **_update** system call every specified number of seconds.

Starting with release 158 the **Flush** utility is no longer necessary. The system will flush buffers by itself after four seconds of inactivity.

3.44 The Free Program

utility:	**FREE**
purpose:	This program displays the amount of unused space remaining on a device.
user access:	all users
summary:	free [devname1 ... devnameN]
arguments:	optional list of device names
options:	none

Description

The **Free** program displays the amount of unused space remaining on a specified device. If no device is specified, the free space is displayed for all mounted devices.

Example:

The following is a sample output of the **Free** utility. It shows the available free space in blocks, kilobytes, and bytes.

```
/dev/root      7,513 blocks      3,756K      3,846,656 bytes
```

3.45 The Ftar Utility

utility: **FTAR**
purpose: Creates and retrieves file archives.

user access: all users

summary: ftar options archive [file-list]

arguments: archive name
optional filenames

options: exactly one of the following per command:

-c	create files on new archive
-t	list files on archive
-x	extract files from archive
-y	compare files from archive

other options

-d	update the time dumped for each processed file
-i	get list of files from stdin
-u	use Unix compatible Tar format
-v	list the names of files processed
-w	wait for confirmation before processing
-r	turn on Read-after-write on block device
-n	turn off Read-after-write on block device
-k #	archive size in kilobytes
-b #	buffer size in kilobytes
-s #	number of volumes to skip

Description

The **Ftar** utility can create a file archive, retrieve files from an archive, or list the contents of an archive. An archive can be written to, or read from, any device or file. Before using **Ftar** to back up files onto a device, the device must be initialized.

The first argument is the archive file (or device) name. If the first argument is "-", the standard input (or standard output) can be used to "pipe" data to and from an archive. The remaining arguments are the names of files and directories to be read or written. If a directory name is specified, all the files in the directory are transferred (recursively).

Archives created by the **Tar** utility cannot be restored with **Ftar**.

Options

One of the following for each **Ftar** command:

-c Creates a new archive of the named files, device files, and/or directories. Any existing data in the archive is overwritten.

-t Lists the named files, device files, and/or directories in the archive. If no names are specified, the entire archive is listed.

-x Copies the named files, device files, and/or directories in the archive to the current directory (unless the extracted files are governed by absolute pathnames). The owner, modification time, and access privileges are restored. If no file arguments are given, the entire archive is extracted. If there are multiple files of the same name, the last one overwrites all of the previous.

-y Similar to the -x option, except that the named files, device files, and/or directories are compared with the identically named files in the current directory. Any differences are reported.

Other options

The **-d** option updates the time dumped for each file written by **Ftar**; the dump time is useful for selective **Ftar** operations (refer to the last example).

The **-i** option accepts filenames from the standard input (until a CONTROL-Z is encountered) and writes the files to an **Ftar** archive. The explicitly listed files are processed first.

The **-u** option prevents writing of device files and directories to an archive (to conform to the UNIX **Tar** utility). During restore, directories are generated as needed.

The **-v** option lists each file as it is processed. When used with the **-t** option, a fuller description of the file is given.

The **-w** option prompts for confirmation before processing each file. Enter "y" or "Y" to confirm the action; any other response cancels the action.

The **-r** option can be used on a block device to turn the Read-after-write mode ON for the duration of **Ftar**. The mode is restored to old value when **Ftar** terminates. If neither the **-r** or **-n** option is used, the current mode setting is used.

The **-n** option can be used on a block device to turn the Read-after-write mode OFF for the duration of **Ftar**. The mode is restored to the old value when **Ftar** terminates. If neither the **-r** or **-n** option is used, the current mode setting is used.

The **-k** option, followed by a number expressed in 1-kilobyte units, limits the archive to the specified size. The **-k** option works in all cases, but is necessary only for nine track tapes (sizes of standard peripherals are known to **Ftar**).

The **-b** option, followed by a number expressed in 1-kilobyte units, defines the size of the **Ftar** buffer. The **-b** option is ignored unless the buffer size specified is larger than the one calculated by **Ftar**.

The -s option can be used to restart an Ftar -c run in case **Ftar** was aborted due to a malfunction (e.g. the tenth floppy turned out to be uninitialized). With the -s option, **Ftar** will pretend it is writing information to a multi-volume device, but no actual writing will take place. After the specified number of volumes has been skipped, normal writing will resume. Note that the results will be incorrect if the source files have changed in between.

Notes

With the **-t**, **-x**, or **-y** options, directories are extracted, listed, or compared with all their descendent directories. When extracting a subtree, all directories above the one being extracted must exist prior to calling **Ftar**.

The **Ftar** utility permits multiple-volume archives if the archive is written to (or read from) a block device. On **Cflop** devices (non-uniform floppy disks), archives start at block 1; on Stdc devices (hard disks), archives start at block 20. On all other devices, archives start at block 0. There is no **Ftar** label, so there is no need to first create an empty archive. The only requirement is that the device must be initialized.

Except for **Cflop** and **Stdc** devices, archives written directly to a device are the same as archives piped into the **Rcopy** utility. For **Cflop** and **Stdc** devices, **Rcopy** must be told where to start writing. However, pipes cannot be used with multiple-volume archives.

Some filenames in archives created by UNIX may be illegal under Cromix-Plus. Cromix-Plus truncates filenames over 24 characters, and may create duplicate filenames in the process. To prevent Ftar from overwriting such files, each file must be extracted separately (using its full name) and renamed. The **Ftar** utility also converts upper case characters to lower case, and all illegal characters (e.g. -, [,], etc.) to the "$" character.

Examples

To view the progress of directory backups to a new archive of one or more small floppy disks:

ftar -cv /dev/sfdc directory_names

To view the progress of file backup to a new archive on floppy tape:

ftar -cv /dev/ftcd file1 file2 ...

To extract all files from the archive on a floppy tape:

ftar -xv /dev/ftcd

To compare files in the archive on a floppy tape with existing files:

ftar -yv /dev/ftcd

To list the contents of a tape archive in long form:

ftar -tv /dev/tp1

To write a floppy tape with the file names entered from standard input (or from a pipe):

ftar -cvi /dev/ftcd

To create a new archive on cartridge tape of all the ordinary files in the "**my**" directory that have been modified since being transferred (refer to the **Scan** utility):

scan my 'type == is_ordin && tmodify >
tdumped && print(path)' | ftar -divc /dev/ftcd

The example above will not work (all "**my**" directory files will be copied) unless the **-d** option was set when the files were first transferred by **Ftar**.

3.46 The Goto Command

Shell command:	**GOTO** or **GO**
purpose:	This command causes transfer of control within a command file.
user access:	all users
summary:	go label
arguments:	line label
options:	none

Description

The **Goto** command transfers control within a command file. Control is transferred to the line specified by **label**. This command is used to execute the same commands within a command file repeatedly. When used in conjunction with the **If** and **Shift** commands, the **Goto** command becomes part of a conditional loop with varying parameters.

A percent sign (%) anywhere on a line means the rest of the line is a comment. A comment at the beginning of a line--with no space after the percent sign--is a label.

The **Goto** command given with a nonexistent line label causes termination of command file execution.

Example:

```
%sample_label
x
y
z % this is a comment
goto sample_label
```

This sample command file causes repeated execution of the commands **x**, **y**, and **z**. The first line of the command file is a line label, as indicated by the leading percent sign.

The percent sign indicates a comment on the fourth line of the file. The fifth (last) line of the file transfers control to the specified label (**sample_label**).

3.47 The Group Utility

utility:	**GROUP**
purpose:	This program allows users to change their groups.
user access:	all users
summary:	group [group name]
arguments:	new group name
options:	none

Description

The **Group** utility examines the **/etc/group** file for a named group. If this group is not found, an error message is displayed, and execution of the utility is aborted.

If the named group is found and if there is a password associated with it, the **Group** utility prompts for the password. If the user responds with the correct password or if there is no password associated with the group, a new shell is formed in which the user belongs to the named group. Upon exiting from the newly created shell, the user's previous status is reinstated.

3.48 The Help Utility

utility:	**H** or **HELP**
purpose:	This program displays pages from the user manual on Shell commands and utility programs.
user access:	all users
summary:	help [-p] [-d directory] [topic ...]
arguments:	optional command or utility names
options:	-d name of directory with help files -p enable print command

Description

The **Help** utility program provides on-line descriptions of the Shell commands and utility programs (for information regarding other aspects of the system refer to the appropriate Cromix-Plus manual).

If you enter **help** and press RETURN, the **Help** utility lists the available topics and prompts you to select one. Entering **help** and a topic name will list the manual entry on that topic. **Help** displays the manual entry one page at a time. The percentage of the file yet to be viewed is displayed at the bottom of the screen.

The following functions will aid you in viewing the manual entry:

SPACE BAR	Display the next page.
u	Display the previous page.
RETURN KEY	Display the next line.
DOWN ARROW	Display the next line.
UP ARROW	Display the previous line.
b	Display the first page.
h	Display the list of available functions.
p	Print the topic (if called with **-p** option).
r	Return to list of available topics if the **Help** utility was called with no arguments.
f	Find a string.
e	Find a string (exact match - case sensitive).
a	Again - Find the same string again.
q	Exit the program.

Trying to move outside the file (e.g., pressing SPACE BAR when the last page is displayed) has no effect.

Modifying the On-line Manual

The text files for the on-line manual are in the **/usr/help** directory (unless the **-d** option selects an

alternate directory). Each topic is described in a file of the same name (plus the **.hlp** filename extension). To add more topic files to the **/usr/help** directory, the filenames must have the **.hlp** extension, otherwise the **Help** cannot access them.

3.49 The Helpsys Utility

utility:	**HELPSYS**
purpose:	This program displays information on functions in the **/usr/lib/syslib.obj** library
user access:	all users
summary:	helpsys [topic ...]
arguments:	optional system call function names
options:	none

Description

The **Helpsys** utility program provides on-line descriptions of the functions available in the library **/usr/lib/syslib.obj**.

If you enter helpsys and press RETURN, the **Helpsys** utility lists the available topics and prompts you to select one. Entering **helpsys** and a function name will list the entry on that topic. **Help** displays the entry one page at a time. The percentage of the file yet to be viewed is displayed at the bottom of the screen.

The following functions will aid you in viewing the entry:

SPACE BAR	Display the next page.
u	Display the previous page.
RETURN KEY	Display the next line.
DOWN ARROW	Display the next line.
UP ARROW	Display the previous line.
b	Display the first page.
h	Display the list of available functions.
p	Print the topic.
r	Return to list of available topics if the **Helpsys** utility was called with no arguments.
q	Exit the program.

Trying to move outside the file (e.g., pressing SPACE BAR when the last page is displayed) has no effect.

General usage of system call functions

Cromix-Plus system calls may be executed directly from a C program using the interface functions described in this document. These functions load the argument values into the appropriate registers, then execute the system call. Generally, the functions have the same names, less the starting underscores, as the system calls which they execute. For example, the function which calls the **_getdir**

system call is named **getdir**.

Channel numbers and file descriptors

Many Cromix-Plus calls require or return a channel number. This channel number is identical to the file descriptor of the C language.

Cromix operating system errors

When a system call reports an error, the interface function returns, as a rule, **-1** to the calling C function after saving the value of the Cromix-Plus error code in the external variable **errno**. The value of the variable **errno** is accessible for inspection by the C program if the declaration

```
extern int errno;
```

is made. In addition, the function **error (channel)** is available, which prints the system-defined error message to the specified channel.

Include files

The directory **/usr/include** contains files that a user program can include to define various types, structures, and constants.

The information about individual functions is available by the use of the **helpsys** utility.

3.50 The History Command

Shell command:	**HISTORY** or **HI**
purpose:	This command displays the last 10 executed Shell commands
user access:	all users
summary:	history
arguments:	none
options:	-a all stored commands

Description

The **History** command will display the last 10 executed Shell commands. Each interactive Shell keeps a 4K buffer of commands. If the buffer overflows, the oldest command is deleted from the buffer. This process is repeated until there is room enough to store the next command line. The excecuted command lines are numbered. Command numbers are the same as the numbers shown as part of each prompt, provided the prompt string contains the characters "**%d**".

The **History** mechanism is an extension of the Shell **retype** capability. Whenever the character CNTRL-R is entered (or the string identified by the "**kR**" termcaps entry capability) the history mechanism is invoked. The **History** mechanism inspects the characters immediately in front of the cursor. If one or more characters in front of the cursor is a decimal digit these characters are interpreted as the History number. If the command line with such a number is actually stored (the last command if there is no number) the characters forming the command number will be erased and the command line extracted from the history will be interpreted as if it were just typed in; it will be either inserted at the cursor position or it will overwrite the characters following the cursor, depending on whether the Shell editor is in insert or replace mode.

Shell manages the current command number. Whenever a command is executed, the current command number points to the last command -- the command just executed.

The UP and DOWN arrow keys (or keys described by "**ku**" and "**kd**" **termcaps entries**) provide for moving the current command number up or down together with the search mechanism. The stored commands are searched for a command that begins with the characters currently in the command line, from the beginning of the line up to cursor position. If the cursor is positioned at the beginning of the command line, every stored command line will match. The search goes in the direction identified by the arrow, starting with the current command number. If a command is found which matches the characters on the current line up to the cursor position, that command replaces the current command and the current command number points to that command.

Some terminals generate the '**\n**' character for the down arrow key. As the terminal should be set to mode CRDEV, mode CRDEV will cause the '**\r**' character generated by the RETURN key to be translated to the '**\n**' character. Under such conditions there is no way to tell the keys RETURN and DOWN ARROW apart. If the terminal driver supports mode INL, Shell will use it and mode INL will

cause the '**\n**' character to be translated to the CNTRL-X character. The character CNTRL-X is intended to be the "standard" down arrow key.

The **-a** option will cause History to display all stored lines.

Note: Non-interactive Shells (command files) do not have the **History** capability: you cannot put a **History** command in a command file. Moreover, you cannot pipe **History** into another process. The command:

jim[1] **hi | more**

does not work because the left hand side of the "|" symbol executes as a non-interactive Shell. You can REDIRECT **History** to a file:

jim[1] **hi > temp**

and by the same token you can use sequential pipes to pipe output into another process:

jim[1] **hi >< more**

3.51 The Icheck Utility

utility:	**ICHECK**
purpose:	This program verifies the integrity of a file system.
user access:	all users
summary:	icheck [-sm] [-b blk# ...] [devname ...]
arguments:	optional list of device names
options:	-s salvage -b blocks -m missing blocks

Description

The **Icheck** utility verifies the integrity of the file system's inode structure. After a power failure or after the computer has been reset, run **Icheck** on all devices that were mounted at the time of failure.

If no device names are specified, **Icheck** checks the integrity of all mounted devices. The list of mounted devices is obtained from the file **/etc/mtab**.

If no options are specified, **Icheck** produces a report on the file system, but does not alter it. A sample report and explanation follow.

If the -s option is used while another task or user is using the disk, the directory on the disk may be irreparably damaged.

```
jim[1] icheck

Device:  /dev/fda

Blocks missing:                     0
Bad free blocks:                    0
Duplicate blocks in free list:      0
Blocks in files and in free list:   0
Bad blocks:                         0
Duplicate blocks:                   0

Device files:                      16
Ordinary files:                   117
Directories:                       14
Blocks used in files:           1,462
Free blocks:                      845
Free inodes:                      358
```

Blocks missing

All disks (also referred to as block devices) are divided into allocation units called **blocks**. A block is 512 bytes. Every block should appear either in a file or in the **free list**. Blocks appearing in files include those permanently assigned as either system or inode blocks. The free list is a list of all blocks available for use.

A block is **missing** if it appears neither in a file nor in the free list. Missing blocks do not compromise the integrity of the file system, and the problem does not need to be corrected immediately. If a block is missing, it is simply not available for use.

The problem may be corrected by executing **Icheck** with the -s option.

Bad free blocks

This message pertains to blocks located in the free list. The term **bad** indicates that the block number is out of range. A block number can be out of range if it is:

1. Past the end of the disk;

2. In the system area of the disk; or

3. In the inode area of the disk.

Bad free blocks do compromise the integrity of the file system, and the problem should be corrected immediately by executing **Icheck** with the -s option. No files are affected.

Duplicate Blocks in Free List

This message means the same block number appears twice in the free list.

Duplicate blocks in the free list **do** compromise the integrity of the file system, and the problem should be corrected immediately by executing **Icheck** with the -s option. No files are affected.

Blocks in Files and in Free List

This message means some blocks appear both in files and the free list. Blocks in files and in free list do compromise the integrity of the file system, and the problem should be corrected immediately by executing **Icheck** with the -s option.

Bad Blocks

This is similar to Bad free blocks except that the bad blocks appear in files.

Bad blocks do compromise the integrity of the file system, and the problem should be corrected

immediately.

Icheck reports the inode number of the bad blocks. The **Ncheck** utility is then used to determine the names of the files containing bad blocks. These files must be deleted. The file may be copied to another file before it is deleted; the new file should be carefully checked because it will probably not be correct.

Duplicate Blocks

This is similar to Duplicate Blocks In Free List except that the duplicate blocks appear in files.

Duplicate blocks do compromise the integrity of the file system and the problem should be corrected immediately.

Icheck reports the inode number of the duplicate blocks. The **Ncheck** utility is then used to determine the names of the files containing duplicate blocks. At least one of these files must be deleted. The **Icheck** utility should then be run with the -s option.

The file may be copied to another file before it is deleted and should be carefully checked because it will probably not be correct.

MESSAGES RETURNED BY ICHECK

Cannot read super block
The super block cannot be read.

Insufficient memory
The disk contains too many inodes for **Icheck** to check. Free additional memory or create a new disk with fewer inodes and use **Cptree** to transfer the contents of the disk to the new disk.

Inode xxxxx, cannot read inode
A disk I/O error occurred while trying to read the specified inode.

Inode xxxxxx, bad usage count
This inode has an incorrect usage count. The usage count is used by the **Usage** utility program to calculate the amount of disk space used. This error can be corrected by running **Icheck** with the -s option.

Inode xxxxxx, cannot write to inode
This error message occurs when the **Icheck** utility is attempting to correct an inode and an error occurs.

Block xxxxxx, inode xxxxxx, block used in file
This is not an error message. This message is displayed when the **-b** option is used, indicating the number of the inode in which the specified block is used.

Block xxxxxx, inode xxxxxx, bad block number

Refer to the previous discussion of Bad blocks.

Block xxxxxx, inode xxxxxx, duplicate block in the system
Refer to the previous discussion of Duplicate blocks.

Block xxxxxx, inode xxxxxx, duplicate block in free list
Refer to the previous discussion of Duplicate blocks.

Block xxxxxx, inode xxxxxx, block in file system and free list
Refer to the previous discussion of Duplicate blocks.

Block xxxxxx, missing block
This message is printed when the **-b** option is used to find the status of a certain block and the block is missing. Refer to the previous discussion of Blocks missing.

Block xxxxxx, block in free list
This message is printed when the **-b** option is used to find the status of a certain block and the block is in the free list.

Block xxxxxx, bad free block
Refer to the previous discussion of Bad free blocks.

Block xxxxxx, cannot write free list block
When running **Icheck** with the **-s** option, the free list is recreated. This error message is printed when there is an error in writing the free list.

Block xxxxxx, cannot read block
This message is printed when a block cannot be read.

Options

The **-s** option salvages and recreates the free list.

The **-b** option displays information about blocks. The option must be followed by a list of block numbers. The numbers must be separated by commas, or they must be separated by spaces and the entire list enclosed in quotation marks.

The **-m** option prints the numbers of missing blocks.

Notes

When using the **-s** option, **Icheck** must be used in conjunction with the **Dcheck** utility. **Icheck** is run after **Dcheck**. Both utilities should be run using the **-s** option. After both programs are run, the system must be rebooted. It is not necessary to reboot if the **-s** option is not used or the device was not mounted. Refer to the **Boot** utility for additional information.

Do not execute the **Icheck** utility when other processes are being executed. This includes detached

processes as well as other user processes.

3.52 The Idump Utility

utility:	**IDUMP**
purpose:	This program displays the contents of an inode.
user access:	all users
summary:	idump [-x] [-i inode-list] blockdev [inode-list]
arguments:	block device name
	optional list of one or more inode numbers
options:	-x dump inode in hexadecimal
	-i inode-list list of inode number

Description

The **Idump** utility displays the contents of the specified inode(s). Unless the **-x** option is used, inodes are displayed in symbolic form.

Inode numbers can be listed as additional arguments following the device name. As an alternative, inode numbers can be given as the **-i** option parameter. In this case the list of inode numbers can be one or more inode numbers separated by commas or it can be one or more inode numbers separated by spaces with the entire list enclosed in quotation marks.

3.53 The If Command

Shell command: **IF**

purpose: This command is used to conditionally execute another command.

user access: all users

summary:
if -err command
-rewa filename command
string-1 = string-2 command
string-1 != string-2 command

arguments: error condition specifier

or

access method and a filename

or

two strings separated by the equal (=) or not equal (!=) relational operator

and

a command line

options: none

Description

The **If** command is used to place a condition on the execution of another command. It is frequently used in conjunction with the **Goto** command. Referring to the summary above, the **If** command has three basic forms.

The first form executes a command if the previous command returned an error. (Refer to the discussion of the **Exit** command.)

In its second form, the If command causes **commands** to be executed if a particular access method applies to the file specified.

The third and fourth forms of the If command cause **command** to be executed when the specified relational condition is true or false. Neither of these forms of the If command requires that the strings be enclosed in quotation marks. However, both forms do require a space on either side of the relational operator (= or !=).

3.54 The Init Utility

utility:	**INIT**
purpose:	Changes or displays run level
user access:	privileged user
summary:	init [level]
arguments:	optional run level
options:	none

Description

Run level is a mechanism that allows the privileged user to enable and disable selected terminals without changing the **/etc/ttys** file.

Run level is recorded in the file **/etc/level**. Run level can be any number in the range 1 .. 15. When the system starts running, the run level equals 1 (one), even if the **init** command has not been executed.

Each terminal has the run levels defined in the first field of every line in the **/etc/ttys** file. In releases prior to 159, the first field of every line in the the **/etc/ttys** file was:

0	terminal disabled
1	terminal enabled

In release 159 and later, the first field of every line in the **/etc/ttys** file can contain a list of run levels (1 .. 15), separated by spaces.

If the list contains the number zero, the terminal is disabled, independent of other run levels as well as the processor run level.

If the list contains the number one, the terminal is enabled for all processor run levels.

If the list contains neither the number zero nor the number one, the terminal is enabled only if the processor is running at a level which is included in the list of terminal run levels.

Note

Be sure you have at least one terminal running at level 1. This will ensure that you will not lock yourself out.

3.55 The Initflop Program

utility: **INITFLOP**
purpose: This program initializes a floppy disk.

user access: privileged user

summary: initflop [[-dsvz] [-c #[,#]] [-h #[,#]
[-l name] [-u #] [-n #] devname]

arguments: Cromix device name

options:

-c	CYLINDERS to initialize
-d	Single DENSITY
-h	HEADS to initialize
-l	CDOS disk label name
-n	Number of CDOS directory entries
-s	Single SIDED
-u	UNIFORM format disk sector size
-v	Verbose
-z	CDOS format disk

Description

The **Initflop** program is used to initialize floppy disks. Parameters may be passed from the command line or interactively from the terminal as the program executes.

Following is a sample script of a typical **Initflop** session to format a small (5-inch) Cromix floppy disk. The user's responses are printed in boldfaced type--everything else is displayed by **Initflop**. RETURN is pressed to select a default response (default values are displayed in angle brackets):

```
Initialize Floppy Disks          version          xx.yy

Press:    RETURN to supply default answers
          CTRL-C to abort program
Warning:  INITFLOP can destroy all disk data

Disk to initialize (devname)? sfdd

Testing:        Rotational speed:   300 RPM

Formatting
     Disk type (C=CDOS, X=CROMIX)? <X>          RETURN
     Single or double sided (S/D)? <D>          RETURN
     Single or double density (S/D)? <D>        RETURN
     Single or double tracked (S/D)? <S>        RETURN
```

```
First cylinder (0-27H)?  <0H>          RETURN
Last cylinder (0-27H)?   <27H>         RETURN
Surfaces (0-1,All)?  <All>             RETURN


Surface,Cylinder
0          00
1          00
0          01
1          01
:          :
:          :
```

The **Initflop** program first asks for the device name of the drive containing the disk to be formatted. Legal responses are the device names of the floppy disk drives connected to your system, such as **fda**, **fdbf1**, **sfda**, **sfdb**, **ufda**, **ufdb**, and so on. Absolute pathnames may also be used, for example **/dev/fda** or **/dev/sfdb**. Be sure to specify the device name correctly, as the **Initflop** program destroys all data on a floppy disk.

Initflop briefly tests the specified drive. Drive speed, which is particularly important when formatting, is displayed by the program.

Next, if the device is not a uniform floppy, **Initflop** prompts for the "type" of disk to be formatted. This determines whether the disk is to be used with the CDOS Operating System or with the Cromix-Plus Operating System. If the device is a uniform floppy, **Initflop** prompts for the sector size.

Initflop then prompts for information about disk sides and density. Double-sided, double-density is the default.

Next, **Initflop** asks for the numbers of the first and last cylinder to be formatted. On occasion, it may be necessary to format only a portion of a disk (for example, one track may be experiencing frequent errors). Selecting the default responses will format the entire disk.

Initflop then prompts for the surfaces to be initialized. To format only one surface, select the corresponding value from the prompt. Selecting the default by pressing RETURN initializes all surfaces.

At this point, **Initflop** proceeds to format the disk. As the disk is formatted, cylinder and surface numbers are displayed. For proper head positioning, the disk is formatted from the outermost cylinders inward.

When **Initflop** is called with a device name as an argument, the responses to the preceding prompts are taken from the command-line options. If no options are specified, double-density, double-sided Cromix disk, with all heads and cylinders formatted is selected by default. For uniform floppies, the default sector size is 512 bytes.

The **-d** option specifies a single-density disk.

The -s option specifies a single sided disk.

The -z option specifies a CDOS-format disk. When the -z option is used, the disk label name (up to 8 characters) may be specified with the -l option and the number of directory entries (64 to 512) may be specified with the -n option. The disk label is only written if the track containing the label has been formatted. These options are invalid with uniform floppies.

The **-u** option, with a sector size as an argument, specifies a uniform-density disk with the corresponding sector size. Valid sector sizes are 128, 256, 512, and 1024 bytes. This option is invalid if the device is not a uniform floppy.

The -c option can be used to specify the cylinders to be formatted. If only one argument is given, only that cylinder will be formatted. If two arguments are given, all cylinders from the first argument through the second argument will be formatted.

The **-h** option can be used to specify that only certain surfaces are to be formatted. If only one argument is given, only that surface will be formatted. If two arguments are given, all surfaces from the first argument through the second argument will be formatted.

When arguments are given on the command line, **Initflop** normally operates in the "quiet" mode. To display information about the type of disk being formatted, and **Initflop's** progress, use the -v (verbose) option.

Example

system[1] **initflop -zv -c 1,3 -l library -n 64 sfdc**

```
Device name:   sfdc
Rotational speed:    300 RPM
CDOS disk      Double sided      Double density
First cylinder:    1.
Last cylinder:    3.
All surfaces

surface,cylinder
0       00
1       00
0       01
:       :
:       :

Disk name:     library
Date on disk:    10/10/84      (current date is printed)
Number of directory entries:    64
```

Had the -v option been omitted, no information would have been displayed.

3.56 The Inithard Utility

utility:	**INITHARD**
purpose:	Initializes a hard disk
user access:	privileged user
summary:	inithard [-n]
arguments:	none
options:	-n new disk

Description

The **Inithard** utility initializes hard disks. It can also be used to declare disk partitions and alternate tracks.

The **Inithard** utility first asks for the device name. In case of devices that have disk partitions (all hard disks except WDI interface drives) you must enter the highest partition available (e.g. std31, esd63, and so on) which specifies the entire drive. When an acceptable device name is entered, initializing proceeds in four steps. In all cases the questions asked provide ranges for answers. No answer will be accepted unless it falls within the allowable ranges. All questions also offer the default answer that will be taken if only the return key is pressed.

Step 1

All hard disks contain some special information on track zero. This special information includes the partition table (how a large disk is broken into a number of smaller devices), and an alternate track table (a list of known bad tracks together with the replacement track for each bad track). Most disks also contain a label. The label contains information such as the number of available cylinders. In case of drives that do not have a label, all label information is known by the drive itself (e.g. ESDC drives).

If the **-n** option is not used, the Inithard utility will first read all pertinent information from track zero. If the **-n** option is used, the **Inithard** utility will not try to read the information from track zero. In the case of labeled drives, the **Inithard** utility will ask the operator label information entries. If the **-n** option was used, **Inithard** will display default values that are most likely to be wrong. The operator may need the drive manufacturer's manual to obtain the correct answers.

The questions asked will be:

Number of surfaces
Number of cylinders
Number of sectors
Interleave factor (optional)
Max number of alternate tracks
Starting cylinder of disk
Starting cylinder for write precomp (optional)

The items: number of surfaces, cylinders, and sectors are self explanatory. Only SMD drives use interleaving. "Max number of alternate tracks" defines the number of tracks that will be set aside. This number should be a multiple of the number of surfaces. All drives except WDI drives should use the number "one" as the starting cylinder of the disk. This will make accidental destruction of the label, partition table, and alternate track table less likely. Only some STDC disks use write precompensation. Here is the list of the most imporartant parameters for standard Cromemco STDC hard disks:

Drive type	**# heads**	**# cylinders**	**Write Precomp**
VERTEX V150	5	987	987
MICROPOLIS 1304	6	830	830
CDC	6	925	0
HITACHI	7	714	256
IMI HD21	6	306	214
MAXTOR XT2140	11	1225	1225
MAXTOR XT1140	15	918	918

When all the questions have been answered, the **Inithard** utility will initialize track zero and rewrite the label (if applicable) the partition table and the alternate track table.

Step 2

This step starts with the question:

Do you want to initialize any tracks?

If the answer is affirmative, the **Inithard** utility will ask for the range of cylinders and heads:

First cylinder
Last cylinder
First head
Last head

Therefore, the tracks (**cylinder**, **head**) with:

First cylinder	<= cylinder	<= Last cylinder
First head	<= head	<= Last head

will be initialized, thereby destroying any information that happens to be recorded on such tracks. Cylinder zero, head zero will not be reinitialized even if included in the range.

Step 3

This step starts with the list of current partitions. For each existing partition the partition number and the number of cylinders it occupies is listed. The list is followed by the question:

Do you want to change partition table?

If the answer is affirmative the **Inithard** utility will keep asking questions like:

Partition 0
Number of cylinders (1 - 689) <689> ?

The default answer is always the remaining number of cylinders. When all the cylinders are used up the new partition table will be written out.

Step 4

This step covers the declarations of alternate tracks. The first question asked is for the starting cylinder number of the alternate tracks:

Alternate track location

Warning: Moving the alternate tracks will jeopardize existing data.

(The number of alternate tracks has been defined in step 1). Next, the current list of alternate tracks is listed in the form

Alternate track table

#	bad track cyl	bad track hd	alt track cyl	alt track hd	#	bad track cyl	bad track hd	alt track cyl	alt track hd	#	bad track cyl	bad track hd	alt track cyl	alt track hd
0	50	3	377	0	2	243	3	377	2	4	320	0	377	1
1	113	2	377	5	3	288	2	377	4					

The first entry of this table indicates that the track on cylinder 50, head 3, is unusable and the system should use the track on cylinder 377, head 0, instead.

The listed table is followed by the question:

Do you want to add an alternate track?

If the answer is affirmative, the **Inithard** utility will ask the questions:

Bad cylinder number
Bad head number

The questions will be repeated until the pair 0, 0 is entered. (Cylinder zero, head zero, cannot be a bad track). The **Inithard** utility will now display the modified alternate track table and ask if the operator wants to add some more alternate tracks. A negative answer is followed by the question:

Do you want to delete an alternate track?

An affirmative answer is followed by the question:

Bad track index

The answer should be the index of the bad track. These indices can be read from the alternate track table under the heading "#". After an alternate track has been deleted, the alternate track table is displayed again. This is repeated until the operator says he does not want to either add or remove an alternate track. The final table will be now written to the disk.

3.57 The Inittape Program

utility: **INITTAPE**
purpose: This program initializes a floppy tape in error correcting format.

user access: privileged user

summary: inittape [[-cdefvx] [-p #] [-s #[,#] devname]]

arguments: Cromix device name

options:

-d	detached mode
-f	fast mode (no retension delay)
-p	number of verification passes
-s	STREAMS to initialize
-v	verbose

and at most one of the following:

-c	check mode (verify tape only)
-e	edit VTOC only
-x	examine VTOC only

Description

The **Inittape** program is used to initialize floppy cartridge tapes in the new error correcting format (SCSI tapes need not be initialized). Parameters may be passed from the command line or entered interactively as the program executes. **Inittape** disables interrupts during formatting or verifying of the entire stream (approximately 90 seconds) so the program can only be aborted between streams.

Options

The **-d** option skips the edit option of the error map.

The **-c** option verifies the tape and displays the VTOC (bad sector map), but skips initialization.

The **-e** option displays the existing VTOC and allows editing. No verification or initialization is performed. This option can be used to quickly convert an old style tape into the new error correcting format without having to run an initialization or verification pass. The old VTOC entries are preserved.

The **-f** option defeats the tape retension delay. Using it while the tape is being retensioned results in a device open error.

The **-p** option, followed by a number, specifies the number of verification passes (the default is 3).

The **-s** option, followed by a single number (0 through 5), formats the stream number specified. Using two numbers separated by a comma formats a range of streams from the first stream number through

the second.

The -x option skips initialization and only displays the VTOC.

The -v option displays a progress report of the initialization when **Inittape** is run from the command line.

Example

A typical interactive session to format a cartridge tape is shown below (user responses are boldfaced). Press RETURN to select the default response (displayed in angle brackets) of any prompt. To begin, enter **Inittape** (as a privileged user) and press RETURN.

```
Inittape version xx.xx

Press:    RETURN to supply default answers CTRL-C to abort program
Warning:  inittape will destroy all disk data

          Do you wish retension delay of 180 seconds (Y/N) <N> ? n
          Device name? ftp1
          Do you wish to examine VTOC only (Y/N) <N> ? n
          Do you wish to edit VTOC only (Y/N) <N> ? n
          Do you wish to VERIFY tape only (Y/N) <N> ? n
          Number of VERIFY passes (0-10) <3> ? 3
```

All tape cartridges are retensioned (run from beginning to end and back to the beginning) when first placed in the drive. If the retension cycle is over (the drive motor has stopped), press RETURN for the first prompt; otherwise enter "Y" and **Inittape** will sleep for 3 minutes.

The next prompt is for device name of the tape drive. Enter the device name of the drive connected to your system, such as **ftp0** or **ftp1**. Be sure to specify the device name correctly, as initialization destroys all data on the tape. Entering a device name while the tape is being retensioned results in a device open error and a repeated prompt for the device name.

Enter "Y" to the next prompt to skip initialization and display the VTOC (bad sector map). Enter "Y" to the edit VTOC only prompt to allow editing of the VTOC without doing a verify pass. Enter "Y" to the "VERIFY tape only" prompt to verify the tape's current formatting (initialization is skipped). Verifying a tape deletes the previous VTOC and generates a new one.

The last prompt above selects the number of verification passes (the default is 3). To verify a tape, select at least one pass; to initialize a tape without generating a VTOC, select 0 passes.

```
Formatting
First stream (0-5.)?  <0> RETURN
Last stream (0-5.)?  <5.> RETURN
```

Inittape prompts for the numbers of the first and last stream to be formatted. On occasion, you may want to format a portion of a tape (e.g., when a single stream has frequent errors). Press RETURN at

both prompts to format the entire tape. When **Inittape** is called with a device name as an argument, the responses to the preceding prompts are taken from the command-line options. If no options are specified, all streams are selected by default.

The stream numbers are displayed as the tape is formatted (the entire stream is formatted in streaming mode, with interrupts disabled).

```
Initializing stream 0
.
.
.
Initializing stream 5
Verify pass 1 on stream 0
.
.
.
Verify pass 3 on stream 5

********** ERROR MAP **********

Stream    Segment    Sectors

    0     6     1#     2*     3
Total bad sectors 3

Do you wish to edit error map (Y/N) <N> ? y
Command (Add, Delete, End) ? d
Enter side,segment,sector ? 0,6,3
Command (Add, Delete, End) ? e

********** VTOC **********

Stream    Segment    Sectors

    0     6     1     2
Total bad sectors 2
```

After the tape is formatted and/or verified, the list of bad sectors is printed. Each sector number is followed by a "#" (sector bad on all passes), a "*" (sector bad on more than one, but not all, passes), or a space (sector bad on only one pass).

Inittape allows you to add or delete entries in the error map. To delete or add a sector, enter the side (stream), segment, and sector numbers separated by commas. Any field (side, segment, or sector) entered as a negative number means all valid entries for that field.

Each tape may only have 199 bad sectors mapped out. If the error map contains more than 199 entries, an error message will be printed and you must either edit the table to contain 199 or less entries or abort **Inittape** and try to initialize the tape again. If you are using the **-d** option, and the error map contains more than 199 entries, the VTOC will be truncated to 199 entries. This means that there will

be unmapped bad spots on the tape and it should not be used.

3.58 The Input Utility

utility:	**INPUT**
purpose:	This program reads a line from standard input and writes it to standard output.
user access:	all users
summary:	input
arguments:	none
options:	none

Description

The **Input** utility reads a string from the standard input and, upon reading a newline character, writes that string to the standard output. This utility can be used to write interactive command language programs by redirecting its output to a file, and then testing the contents of the file with the **Testinp** utility.

Input stops reading when it reads a zero byte, a carriage return, or a line feed character. The terminating character is not written to STDOUT.

Example:

```
jim[1] input > temp
```

The above command reads one line from the standard input and writes it to the file **temp**.

3.59 The Ioload Utility

utility:	**IOLOAD**
purpose:	This program loads a program into an I/O board
user access:	privileged user
summary:	ioload [-r] filename device
arguments:	filename of program to be loaded
	I/O device name (io1 .. io4 oc1 .. oc8)
options:	-r Reset the board before loading

Description

The **Ioload** utility loads a file into an I/O board (IOP, Octart, or Biart). This utility is normally used to load the IOP driver **/etc/quadart.iop**, **/etc/tape8.iop**, or **/etc/tape16.iop** into an IOP, the Octart driver **/etc/octart.iop** into an Octart, or the driver for Z80 programs **/etc/zio.iop** into a Biart or Octart. **Ioload.bin** is located in the **/etc** directory.

The -r option should be used only with the devices that understand software reset sequence (Octart and Biart). The reset sequence will ensure that the board is listening to the host in a well defined manner.

3.60 The Ipcrm Utility

utility:	**IPCRM**
purpose:	This program removes a message queue, semaphore set, or shared memory identifier.
user access:	owner of the facility
summary:	ipcrm -{ik}{qms} identifier(s)
arguments:	IPC facility identifiers or keys
options:	-i IPC facilities are identified by identifiers -k IPC facilities are identified by keys -q remove message queues -m remove shared memory segments -s remove semaphore sets

Description

The Ipcrm utility will remove one or more specified message, shared memory, or semaphore identifiers. Exactly one of the options **-i** and **-k** must be used to specify that IPC facilities are to be identified either by identifier or by key. Exactly one of the options **-q**, **-m**, and **-s** must be used to specify removal of a message queue, shared memory segment, or semaphore.

The IPC identifiers and keys may be found using the **Ipcs** utility.

Options

The **-i** option specifies the arguments are IPC identifiers.

The **-k** option specifies the arguments are IPC keys.

The **-q** option requires removal of the message queue identifers.

The **-m** option requires removal of the shared memory identifers.

The **-s** option requires removal of the semaphore identifers.

3.61 The Ipcs Utility

utility: **IPCS**
purpose: This program reports inter-process communications status.

user access: all users

summary: ipcs [-qmsbcopta]

-m	shared memory segments
-s	semaphores
-b	biggest allowable size information
-c	creator's login name
-o	outstanding usage
-p	process number information
-t	time information
-a	shorthand for -bcopt

Description

Ipcs utility prints certain information about active inter-process communication facilities. Without options, information is printed in short format for message queues, shared memory, and semaphores that are currently active in the system. Otherwise the information that is displayed is controled by options.

Options

The **-q** option causes the information about the active message queues to be printed.

The **-m** option causes the information about the active shared memory segments to be printed.

The **-s** option causes the information about the active semaphores to be printed.

If any of the options **-q**, **-m**, or **-s** are specified, information about only those indicated will be printed. If none of these three is specified, information about all three will be printed.

The **-b** option causes the information regarding maximum allowable sizes to be printed. (Maximum number of bytes in messages on queue for message queues, size of segments for shared memory, and number of semaphores in each set of semaphores.)

The **-c** option causes the creator's login name and group name to be printed.

The **-o** option causes information on outstanding usage to be printed. (Number of messages on queue and total number of bytes in messages on queue for message queues and number of processes attached to shared memory segments.)

The **-p** option prints process ID information. (Process ID of last process to send a message, process ID of last process to receive a message on message queues, process ID of creating process and process ID of last process to attach or detach on shared memory segments.)

The **-t** option causes the time information to be printed. (Time of last control operation that changed the access permission for all facilities. Time of last **msgsnd** and last **msgrcv** on message queues, last **shmat** and last **shmdt** on shared memory, last **semop** on semaphores.)

The **-a** option stands as a shorthand notation for all print options (**-b**, **-c**, **-o**, **-p**, **-t**.)

Notes

The column headings and the meaning of the columns in an Ipcs listing are given below; the letters in parentheses indicate options that cause the corresponding heading to appear; "all" means that the heading always appears.

T (all) Type of facility:

- q message queue
- m shared memory segment
- s semaphore

ID (all) The identifier for the facility entry.

KEY (all) The key used as an argument to the **msgget**, **shmget**, or **semget** system call to create the facility entry. (**Note:** the key of a shared memory segment is changed to IPC_PRIVATE when the segment has been removed until all processes attached to the segment detach it.)

MODE (all) The facility access mode and flags. The mode consists of 11 characters that are interpreted as follows:

The first two characters are:

- R if a process is waiting on a **msgrcv**;
- S if a process is waiting on a **msgsnd**;
- D if the associated shared memory segment has been removed. It will disappear when the last process attached to the segment detaches it;
- \- if the corresponding special flag is not set.

The next 9 characters are interpreted as three sets of three bits each. The first set refers to the owner's permissions; the next one refers to permissions of others in the user's group; and the last to all others. Within each set, the first character indicates permission to read, the second character indicates permission to write or alter the facility entry, and the last

character is currently unused.

The permissions are indicated as follows:

r	if read permission is granted
w	if write permission is granted
a	if alter permission is granted
-	if the indicated permission is not granted.

OWNER	(all)	The login name of the owner of the facility entry.
GROUP	(all)	The group name of the group of the owner of the facility entry.
CREATOR	(a,c)	The login name of the creator of the facility entry.
CGROUP	(a,c)	The group name of the group of the creator of the facility entry.
CBYTES	(a,o)	The number of bytes in messages currently outstanding on the associated message queue.
QNUM	(a,o)	The number of messages currently outstanding on the associated message queue.
LSPID	(a,p)	The process ID of the last process to send a message to the associated queue.
LRPID	(a,p)	The process ID of the last process to recieve a message from the associated queue.
STIME	(a,t)	The time the last message was sent to the associated queue.
RTIME	(a,t)	The time the last message was received from the associated queue.
CTIME	(a,t)	The time when the associated entry was created or changed.
NATTCH	(a,o)	The number of processes attached to the associated shared memory segment.
SEGSZ	(a,b)	The size of the associated shared memory segment.
CPID	(a,p)	The process ID of the creator of the shared memory entry.
LPID	(a,p)	The process ID of the last process to attach or detach the shared memory segment.
ATIME	(a,t)	The time the last attach was completed to the associated shared memory segment.

DTIME	(a,t)	The time the last detach was completed on the associated shared memory segment.
NSEMS	(a,b)	The number of semaphores in the set associated with the semaphore entry.
OTIME	(a,t)	The time the last semaphore operation was completed on the set associated with the semaphore entry.

3.62 The Kill Command

Shell command:	**KILL**
purpose:	This command sends a signal to a process.
user access:	all users
summary:	kill [-12345678] [PID #s]
arguments:	process id
options:	-1 abort -2 user -3 kill -4 terminate (default) -5 alarm -6 broken pipe -7 modem hang up -8 reserved

Description

The **Kill** command sends a signal to the process specified. If the signal type is unspecified, **Kill** sends a terminate signal. When a signal is sent to process 0, the signal is also sent to all processes initiated from the user's terminal.

If the user is a privileged user, and a user signal is sent to process 1 (**kill -2 1**), system shutdown is initiated.

Kill 0 aborts all background jobs attached to the user's terminal.

Kill -1 1 consults the **/etc/ttys** file and allows any terminals o that have been added to be logged on. It also logs off any terminals that have been deleted from the file.

Options

The -l option causes an abort signal to be sent to the process. This option has the same effect as CONTROL-C from the keyboard, and aborts only interactive programs. Detached processes continue unaffected.

The -2 option sends a user signal to the process. It is generated by a character typed at the terminal. The character that generates the signal is determined by the mode.

The -3 option sends a kill signal to the process. This kill signal cannot be ignored or trapped. It is typically used to abort a program caught in an infinite loop.

The **-4** option sends a terminate signal to the process. The terminate signal kills both interactive and background processes. This is the default type of signal sent by the **Kill** command.

The **-5** option sends an alarm signal to the process.

The **-6** option is sent by the operating system when a pipe is used improperly.

The **-7** option is sent by the characters drivers if a modem detects a hangup condition.

The **-8** option is reserved for future use.

3.63 The Ksam Utility

utility:	**KSAM**
purpose:	This program executes KSAM (Keyed Sequential Access Method) calls for all users.
user access:	all users
summary:	/etc/ksam lockfile &
arguments:	pathname of the lockfile
options:	none

Description

The **Ksam** utility must be run in the background. It is commonly be started from the **/etc/startup.cmd** command file as a detached process.

The **Ksam** utility uses inter-process communications (IPC) facilities. This in means that it can only run on Cromix-Plus version 31.06 or higher. The system must be **Crogened** with parameters for messages and shared memory (semaphores are not used).

The **Ksam** utillity is normally started as a detached job, e.g., by the **/etc/startup.cmd** file. As long as it is running, the **Ksam** interface functions can be used from other user programs. The **Ksam** process can be killed normally, e.g. **kill <PID>**, or by using the **Ipcrm** utility to remove its message queue. Do not kill **Ksam** using the **Kill -3 <PID>** command as this will not allow **Ksam** to flush its buffers.

There might be more than one group of processes using the IPC facilities, and more than one (independent) instance of **Ksam** might be running concurrently. There must be a mechanism to prevent different users of the IPC facilities, or of **Ksam's** in particular, from getting mixed up. When **Ksam** is started it must be given one argument, for example:

```
jim[1] /etc/ksam lockfile &
```

The referenced file, **lockfile** in the above example, is an arbitrary file (on the root device). Its use is twofold:

- The inode number of this file is used to build the keys for the IPC facilities

- The access privileges of the file are used to build the access privileges to the message system. For a user program to be able to use **Ksam** calls he must have the Read/Write access to the message system. This is computed from the Read/Write access to the **lockfile**.

The IPC access mechanism allows for different accesses of the owner, group, or public. Access to the **lockfile** can be constructed so that some particular instance of **Ksam** can be used only by a particular user, by a particular group of users, or by the general public.

Each **Ksam** user must first call the function **ksstart** (explained in the **ksclib** helpfile) to establish communications with **Ksam**. The **ksstart** function must reference the same **lockfile** to allow the user to get access to the right IPC facility.

Usage of IPC facilities.

The **Ksam** utility itself maintains a message queue. Each command from a user sends an eight byte message, each completed command from **Ksam** sends back a four byte message. In addition, each **Ksam** user will establish a 4K shared memory segment **Ksam** is going to use while executing a **Ksam** command.

3.64 The Ksclib.obj Utility

library:	**KSCLIB.OBJ**
purpose:	Object time library of C callable functions to communicate with a Ksam process running in the backgroud.
user access:	all users
summary:	jlinker ... -lksc -lc

Description

The **Ksam** utility uses inter-process communications (IPC) facilities. This means that it can only run on Cromix-Plus version 31.06 or higher. The system must be "Crogened" with parameters for messages and shared memory (semaphores are not used).

For user programs to be able to use **Ksam** calls, the **Ksam** utility must be running (in the background).

Each **Ksam** user must first call the function **ksstart** (explained below) to establish communications with **Ksam**. The **ksstart** function must reference the the **lockfile** which **Ksam** was given as an argument when it was started.

The functions from this library return:

0	Execution successful
> 0	**Ksam** error number
< 0	System error number in extern int errno

User's viewpoint of Ksam

From the user's viewpoint, **Ksam** is a set of functions which can be linked into his program. At present these functions are available only to 68000 C programs. The functions are stored in the library **/usr/lib/ksclib.obj**. In the future there will be interface functions available for use from other languages.

Each **Ksam** function returns an error number. Zero indicates no error. Numbers less than 1000 are Cromix error numbers; number greater or equal to 1000 are **Ksam** error numbers (described in **Ksam.h**).

Most functions listed below have a filename as the first argument. This filename identifies the data base **Ksam** is to work on.

```
ksstart(lockfile)
char *lockfile;
```

This is the first function that has to be called from a user program to establish communications with the **Ksam** process. The lockfile given must be the same file the **/etc/ksam** was started with.

```
ksbuild(filename,recordsize,access)
char *filename;
int recordsize;
int access;
```

This function builds an empty data base. This means that the files **filename.dat** and **filename.btr** are created and initialized to an empty data base with NO indices. The data base records will be recordsize bytes each. The files will be created with access privileges defined by access. The integer access is understood as 12 bits in the usual sense:

```
11
1098 7654 3210
rewa.rewa.rewa

ksaddkey(filename,keypos,dupsarray)
char *filename;
short keypos[7,2];
char *dupsarray;
```

A key defined by **keypos** is added to the data base. A data base can have up to seven keys. Each key is defined by up to seven pairs of (short) integers. Each pair quotes the starting and ending byte position of a part of the key. The bytes are counted from zero. Unused pairs should be set to (-1, -1). If the string **dupsarray** contains the string "dups" (or "DUPS" and the like), records with identical keys are allowed.

```
ksdelkey(filename,keypos)
char *filename;
short keypos[7,2];
```

The index key identified by **keypos** is removed from the data base. The space occupied by the removed index entries goes into the index free list.

```
ksrename(filename,newname)
char *filename;
char *newname;
```

Rename files forming the data base. **Filename** is the old name, **newname** is the new name.

```
kserase(filename)
char *filename;
```

The data base called filename is destroyed.

```
ksopen(filename)
char *filename;
```

The data base identified by **filename** is open for processing. The data base must be open before it can be read or modified.

```
ksread(filename,keypos,keyvalue,keylength,record)
char *filename;
short keypos[7,2];
char *keyvalue;
int keylength;
char *record;
```

This function will find the record identified by the **keyvalue** string of **keylength** bytes. The **keyvalue** will be compared to the keys stored in the index file. The index will be identified according to **keypos**. **Keylength** can be smaller than the size of key computed from **keypos**. The first record that has a matching key in the first **keylength** characters will be found. The records are sorted by increasing key. If there are a number of records with identical keys, they are sorted by age, the oldest first. The record will be stored in record.

If a matching key is not found the record with the smallest key which exceeds the given key will be returned. The **ksread** function will return the KS_NEXT error in this case.

```
ksnext(filename,record)
char *filename;
char *record;
```

This function will read the next record in the index order.

This call must be preceded by a **ksread**, **ksfirst**, **ksnext**, **ksprevious**, **ksretcurrent**, or **ksposition** call.

```
ksprevious(filename,record)
char *filename;
char *record;
```

This function will read the previous record in the index order.

This call must be preceded by a **ksread**, **ksfirst**, **ksnext**, **ksprevious**, **ksretcurrent**, or **ksposition** call.

```
ksfirst(filename,keypos,record)
char *filename;
short keypos[7,2];
char *record;
```

This function will read the first record by the key identified by **keypos**.

```
kslast(filename,keypos,record)
char *filename;
short keypos[7,2];
char *record;
```

This function will read the last record by the key identified by **keypos**.

```
ksphfirst(filename,record)
char *filename;
```

```
char *record;
```

This function will read the first record in the physical order. No keys have to be defined.

```
ksphnext(filename,record)
char *filename;
char *record;
```

This function will read the next record in the physical order. No keys have to be defined.

This call must be preceded by a **ksread**, **ksfirst**, **ksnext**, **ksprevious**, **psfirst**, **phnext**, **ksretcurrent**, or **ksposition** call.

```
ksaddrec(filename,record)
char *filename;
char *record;
```

The record given in record will be added to the data base. All index trees will be updated.

```
ksdelrec(filename)
char *filename;
```

The current record will be deleted from the data base.

This call must be preceded by a **ksread**, **ksfirst**, **ksnext**, **ksprevious**, **ksretcurrent**, or **ksposition** call.

```
ksretcurrent(filename,record)
char *filename;
char *record;
```

The current record will be returned.

This call must be preceded by a **ksread**, **ksfirst**, **ksnext**, **ksprevious**, **ksretcurrent**, or **ksposition** call.

```
ksretuniqueid(filename,idptr)
char *filename;
long *idptr;
```

This call returns in ***idptr**, a unique number to be used for building lock sequences for the **lock/unlock** system calls. This number will be incremented with each successful **addrec** call.

```
ksposition(filename,keypos,keyvalue,keylength)
char *filename;
short keypos[7,2];
char *keyvalue;
int keylength;
```

This call does all the actions the call **ksread** would do without actually reading the record. Following this call, the current record is defined so that, for example, **ksdelrec** or **ksupdate** can be called.

```
ksupdate(filename,record)
char *filename;
char *record;
```

The current record will be deleted and the record identified by record will be inserted. The call is strictly equivalent to the sequence:

ksdelrec followed by **ksaddrec**

treated as an indivisible operation.

This call must be preceded by a **ksread**, **ksfirst**, **ksnext**, **ksprevious**, **ksretcurrent**, or **ksposition** call.

```
ksclose(filename)
char *filename;
```

This call will flush all **Ksam** buffers belonging to the data base and close the files. Whenever **Ksam** receives a command, it will close all files of dead users. However, if other users have the data base open it will not be flushed this way.

```
ksstop()
```

This call will break communications with **Ksam** and issue a **ksclose** for files opened by this user.

```
kslock(string,length,type)
char *string; int length, type;
```

Identical to the **lock** system call except for rearranged arguments.

```
ksunlock(string,length,type)
char *string; int length, type;
```

Identical to the **lock** system call except for rearranged arguments.

Ksam interface information

Ksam communications can be described as follows:

The **Ksam** process is supposed to be started as a detached process. This process creates a message queue and then waits until a message (with a message number KSAMMSG, which is not a valid process ID) is received.

When a user program wants some service from the **Ksam** process, it opens the **Ksam** message queue. The user process then sends the first message to **Ksam**. All messages sent by the user program to **Ksam** have the same format: the message number is KSAMMSG, the contents of the message are two integers: the process's own PID, and the **Ksam** call number. The first **Ksam** call sent over the message system must be **ksstart**. The user process sends this message and then waits for a message with his own PID as the message number.

When **Ksam** recieves a message it takes the process PID from the message body. The first call from each user is supposed to be **ksstart**, which causes **Ksam** to initialize itself to serve another user (the user with this PID). **Ksam** then creates a shared memory segment for the user, and sends back a message to the the user. All messages sent from **Ksam** have the destination process PID as the message number and **Ksam** error code as the body of the message.

The user process receives this message next, and in the case of 0 errors, communications are established. For each service call, the user can attach the shared memory segment (created by **Ksam**), move all data into the shared memory segment, and send a command to **Ksam**.

Ksam receives the command, attaches the shared memory segment itself, pretends he is the user himself, and executes the normal **Ksam** function. The return data is put into the shared memory segment, the shared memory is detached, and a message which contains the error code is sent to the user.

The user code receives the response message, moves the results from the shared memory to normal memory, and detaches the shared memory segment.

This sequence of events can go on indefinitely. The **Ksam** process is intended to run continuously. If it gets killed, the user interface is informed accordingly.

3.65 The L Utility

utility:	**L**
purpose:	This program lists directory or file information in the old Cromix-20 style.
user access:	all users
summary:	l [-adrst] [file-list]
arguments:	optional file or directory pathname(s)
options:	-a all -d directory information -r reverse order -s summary -t time modified

Description

The **L** utility lists directory or file information in alphabetical order. If no pathname is specified, **L** lists the contents of the current directory. If a directory pathname is given, the contents of that directory are listed. If a file pathname is given, information about that file is listed.

Options

The **-a** option lists the names of all files, including invisible files (those files whose names begin with a period).

The **-d** option lists information about the directory, rather than the contents of the directory.

The **-r** option performs the sort specified in reverse order. Thus, an alphabetical listing is given in reverse alphabetical order, and a time-date listing is listed most recent file first.

The **-s** option generates a summary of listed files.

The **-t** option sorts the file list in order of time last modified. This order is from oldest to most recent unless the **-r** option is used.

Notes

This utility is a simple command file, which uses the **Ls** utility to produce the result. Refer to the description of the **Ls** utility.

3.66 The Link68 Utility

utility: **LINK68**
purpose: This program links .o68 files.

user access: all users

summary: link68 [options] [-o outname] filename . . .
filename [-s libname] . . .

arguments: one or more filenames

optional library name, preceded by -s

options:

-a0	value of A0 register
-a1	value of A1 register
-a2	value of A2 register
-a3	value of A3 register
-a4	value of A4 register
-a5	value of A5 register
-a6	value of A6 register
-a7	value of A7 register
-n	no map
-o	output file name
-q	do not display map
-s	search library
-x	lsect alignment size
-y	unformatted output
-z	memory size
-r	produce ROMable code
-d	data address

Description

The **Link68** utility is a two-pass virtual linker. From one or more input files, **Link68** generates an executable binary file with a "**.bin**" filename extension. **Link68** can be used to generate absolute images of standalone programs, such as the Cromix-Plus Operating System.

Options

The **-an** option, where 0 <= **n** <= 7, specifies the address to be loaded into the **An** register. **Link68** uses this address to generate the A register with offset types of effective addresses. This option should be used only if the "**.o68**" files being linked were generated by nonstandard programs.

The **-n** option prevents creation of a link map. If -n is not specified, a link map is created and written to a file with the filename extension "**.map**".

The **-o** option, with a filename as an argument, specifies the output filename. If **-o** is not specified, the output file has the name of the first relocatable file specified on the command line, with the filename extension "**.bin**".

The **-q** option inhibits display of the link map on the terminal. If **-q** is not specified, the link map is displayed.

The **-s** option, with a library filename as an argument, specifies a library to be searched. To search more than one library, use multiple **-s** options. **Link68** searches the "**.o68**" file for necessary functions.

The **-x** option, with a hex number as an argument, defines the padding size of each linked **lsect**. The size of each **lsect** will be a multiple of this padding size. If **-x** is not specified, the value 4096 is used.

The **-y** option, with a hex number as an argument, specifies the starting address of an absolute image. When **-y** is specified, the generated output file has no internal structure. **cromix.sys** is an example of a file linked in this way.

The **-z** option defines the amount of free memory to be added to the end of the binary file. This memory can be used for stack or any other purpose.

The **-r** option is used to produce code that can be written into ROM, (ROMable code). With the **-r** option, only three types of **Psects** are allowed:

REA,EXE	Code Psect
REA,WRI	Initialized data Psect
REA,WRI,UNI	Uninitialized data Psect

The **-r** option has the effect that the initialized data **Psect** is included twice: The initializing data is appended to the end of the code **Psect**, the initialized data itself is referenced at the end of uninitialized data **Psect**. This implies that only two sections are finally produced, the code section that also contains the initializing data, and the uninitialized data section than also provides room for a copy of the initialized data. The idea is that the first thing such a program must do is to copy the initialized data from the code section to the data section and, if necessary, zap the rest of uninitialized data. In order for the program to be able to do that, **Link68** provides eight entry points with the following meaning:

_cbegin	points to the first byte of code
_cend	points to the first byte following code
_ibegin	points to the first byte of the copy of the initialized data
_iend	points to the first byte following the copy of the initialized data
_ubegin	points to the first byte of unitialized data
_uend	points to the first byte following the unitialized data
_dbegin	points to the first byte where initialized data has been copied
_dend	points to the byte following the initialized data

If the program is linked with the -r option, the first thing the program must do is

to zap (_uend - _ubegin) bytes at address _ubegin
(if not zapped already)
to copy (_iend - _ibegin) bytes from address _ibegin
to address _dbegin

This scheme applies to any program linked with the -r option, indepedent of other options (like -y and -d).

The -d option, used together with the -r and -y options, can be used to specify the address of the data section. The -d characters must be followed by a space and the data address in hexadecimal.

If the code is intended to go into ROM, in addition to the -r option,

the -y option has to be used to get rid of Lsect headers and to
specify the code address (ROM address),
the -d option has to be used to assign the data segment to a RAM address.

3.67 The Logerr Program

utility:	**LOGERR**
purpose:	This program accumulates errors detected and reported by the Ecc program.
user access:	privileged user
summary:	logerr [#]
arguments:	Optional time period in seconds
options:	none

Description

If called without an argument, **Logerr** displays the contents of the file that accumulates errors reported by the **Ecc** program. If **Logerr** is called with an argument, the argument defines how often the **Ecc** program should scan the error-correction hardware and append any errors to a file for later display by **Logerr**.

Example:

```
system[1] logerr 30
```

The **Ecc** program will scan the error-correcting hardware every 30 seconds.

The following example is a typical display when **Logerr** is called without argument.

```
system[1] logerr
*ECC started Jul-21-1984 16:45:22
 Single bit error - MCU 1, Addr Ocxxxx, Row 3, Column 18
 Single bit error - MCU 1, Addr Odxxxx, Row 1, Column 18
```

3.68 The Ls Program

utility:	**LS** (List)
purpose:	This program lists directory or file information.
user access:	all users
summary:	ls [-abdeilmrst] [file-list]
arguments:	optional file or directory pathname(s)

options:

-a	all
-b	brief
-d	directory information
-e	everything
-i	inode number
-l	long list
-m	medium list
-r	reverse order
-s	summary
-t	time modified

Description

The **Ls** program lists directory or file information in alphabetical order. If no pathname is specified, it lists the contents of the current directory. If a directory pathname is given, the contents of that directory are listed. If a file pathname is given, information about that file is listed.

Options

The **-a** option lists the names of all files, including invisible files (those files whose names begin with a period).

The **-b** option makes a brief list, which contains only filenames.

The **-d** option lists information about the directory, rather than the contents of the directory.

The **-e** option lists everything about a file.

The **-i** option lists an inode number, rather than the file size.

The **-l** (long) option makes a long list of information. This option does not display as much information as the **-e** option.

The **-m** option makes a medium list (more information than **-b**, less than **-1**), containing file size, number of links, and name.

The -r option performs the sort specified in reverse order. Thus, an alphabetical listing is given in reverse alphabetical order, and a time-date listing is listed as the most recent file first.

The -s option generates a summary of disk space used.

The -t option sorts the file list in order of time-last-modified. This order is from oldest to most recent unless the -r option is used.

Notes

The meaning of the file-size information displayed by the Ls utility is as follows. If the file listed is a regular (data) file, the number associated with the file is its size in bytes (or number of characters). If the file is a directory, the number is the number of files stored in that directory. If the file is a device file, the numbers are the major and minor device numbers.

When options are combined on the command line, the most extensive option prevails.

Example:

Samples of the output from the Ls utility follow. Each is preceded by a note as to the option utilized. Without options, Ls displays a sorted, columnar list of filenames.

The following shows an output of Ls with the -b option, containing only filenames.

```
apa
apa1
apb
apc
apd
ape
```

The following shows an output of Ls using the -m option. For a filename, the field on the extreme left contains the number of bytes in the file. This is followed by the number of links to the file, and the filename. If the entry represents a directory, as in the first entry shown, the leftmost number shows the number of files in the directory. The D indicates it is a directory. The last two fields show the number of links and the directory name.

```
    3   D       1 cromix.doc
1,559           1 default.fm0
```

A sample of the output of Ls using the -e option is shown below. This is the most complete display. The name of each file in the directory is displayed on the extreme left. To the right, on the same line, is the number of bytes in the file. The first column of the remaining lines lists the operations performed on the file: created, modified, accessed, or dumped. To the right of each operation is the date and time the operation was last performed.

The rightmost column contains additional information. At the top the read, execute, write and access privileges for the owner, group, and all other users are shown. The second line is the login name of the file owner. The third entry lists the number of links to the file, and the final entry is the inode number.

To the extreme right of the owner's login name is an entry showing the group name of the user: in this case, **pubsl**.

```
Directory:  cromix.doc
locktest                                        9     directory
        created:        Oct-21-1984     13:56:57      rewa re-- re--
        modified:       Oct-21-1984     13:56:57      karen                pubs1
        accessed:       Nov-19-1984     12:49:41      links:   1
        dumped:         000-00-1900     00:00:00      inode:   734

pipetest                                       10     directory
        created:        Oct-21-1984     13:56:13      rewa re-- re--
        modified:       Oct-21-1984     13:56:13      karen                pubsl
        accessed:       Nov-19-1984     12:49:33      links:   1
        dumped:         000-00-1900     00:00:00      inode:   781

system.c                                    1,641
        created:        Oct-21-1984     13:56:10      rewa re-- re--
        modified:       Oct-21-1984     13:56:11      karen                pubsl
        accessed:       Nov-31-1984     12:17:05      links:   2
        dumped:         000-00-1900     00:00:00      inode:   782
```

The following example shows the **Ls** program output using the **-i**, or inode number, option. This display first shows the directory name. The names of all files and directories within the subject directory are listed on the right. The inode number associated with the file is shown to the left.

```
Directory:      cromix.doc
    734         locktest
    781         pipetest
    782         system.c
```

The following example shows the **Ls** program output using the **-l** option. If the second field in the entry is a **D**, for directory, the leftmost field indicates the number of files in that directory. If the second field is blank, the entry is a file, and the leftmost field shows the number of bytes in the file. Moving to the right, the third field indicates the number of links to the file or directory.

The next field shows the delete protect attribute, shared text attribute, read, execute, write, and append access of the directory or file for the owner, group, and all other users, in that order. Immediately to the right of the access privileges is the login name of the owner. The three rightmost fields in this format are the most recent date and time when the file was last modified, and the file or directory name. If the file was last modified not in this year the time will be replaced ty the year.

```
Directory:  cromix.doc
        9       D   1   rewa re-- re-- karen          Oct-21 13:56    locktest
       10       D   1   rewa re-- re-- karen          Oct-21 13:56    pipetest
```

```
1,641        2 PS  rewa re-- re-- karen        Oct-12 1984        test.bin
```

The following is a sample of Ls program output using the -s and -m options. This display is the same as that obtained using the -m option, except that the last line of the display is a summary showing, from left to right, the number of files, number of blocks, and total bytes in the directory.

```
Directory:      cromix.doc
   9 D          1 locktest
  10 D          1 pipetest
1,641           2 test.bin
3 files         6 blocks          2,313 bytes
```

What follows is a sample of Ls program output using the -t and -m options. These files are listed in order of the time last modified.

```
Directory:      cromix.doc
1,641           2 system.c
  10 D          1 pipetest
   9 D          1 locktest
```

3.69 The Lstat Utility

utility:	**LSTAT**
purpose:	This program reports status of the lock table.
user access:	all users
summary:	lstat
arguments:	none
options:	none

Description

The **Lstat** utility prints certain information about active locks installed currently in the system.

The column headings and the meaning of the columns in an **Lstat** listing are given below.

PID — The process ID of the process that installed the lock.

SHARED — This column contains the word "Yes" or "No". If the lock is shared, some other process may install the same lock, as long as it is also shared.

WANTED — This column contains the word "Yes" or "No". The lock is "wanted" if there is a process waiting for the lock to be unlocked.

KEY — This column contains the full lock sequence. The sequence is displayed in hexadecimal as it may contain non ASCII characters. Note that the preferred form of the lock sequence is:

Device number	2 bytes
Inode number	2 bytes
Other	up to 12 bytes

The device number and the inode number are used to ensure a unique locking sequence (device number and inode number identify a unique file in all the mounted file systems).

3.70 The Mail Utility

utility:	**MAIL**	
purpose:	This program sends or displays mail.	
user access:	all users	
summary:	mail [-aegv] [-f file] [user-name ...]	
arguments:	optional list of user names	
	or	
	optional list of group names	
options:	-a	all
	-e	check mail
	-g	group
	-v	verbose
	-f file	mail file to read

Description

Given without arguments, **Mail** displays mail sent to the user. After each section of mail is displayed, the **Mail** utility will display a question mark. Enter one of the commands described in the Commands section.

Given with one or more user names as arguments, the **Mail** utility sends mail to one or more users. To send mail, enter the message after pressing RETURN at the end of the command line. A CONTROL-Z terminates the message and returns control to the Shell.

Options

The **-a** option sends mail to all users. The list of users for this option is obtained from the **/etc/passwd** file.

The **-e** option checks if the user has mail. No mail is read. The **Mail** uility returns

0	There is mail
1	There is no mail

The **-g** option sends mail to members of a specified group(s). Group members are defined in the **/etc/group** file.

The **-v** option displays the list of users who received mail.

The **-f** option must be followed by a path name. The **Mail** utility will read the mail from the specified

file instead of the file "**/usr/mail/user-name**".

Commands

After a section of mail is displayed, the user is prompted for what to do with the mail. A single line is read and interpreted as follows:

d	Delete the message and go on to the next section of mail.
+	Keep the current section of mail and display the next section of mail. Do nothing if there is no next section.
-	Keep the current section of mail and display the previous section of mail. Do nothing if there is no previous section.
p	Print the same section of mail again.
s [files]	Save the current section of mail into named files. The mail is appended to all the named files. The **Mail** utility called with the **-f** option can be used to inspect such saved mail. If there are no files specified, the mail is saved to the file "**../mbox**".
w files	The section of mail is written to all named files. The files should not previously exist The **Mail** is written without the **Mail** time stamp.
m users	Forward the section of mail the the named users. If there is no user named, the mail is forwarded to you.
!	command The **Mail** utility will execute this command.
q	Put back all undeleted mail and stop.
x	Put back all mail, deleted or not, and stop.
CONTROL-Z	Same as q.
CONTROL-C	Same as x.
Any other	Show a short help file.

Notes

Upon logging in to the system, a user is informed if there is mail.

Command examples.

! mail person

Immediatelly send the reply to the person.

! ls

Execute the Ls utility.

! shell

Start an interactive Shell.

3.71 The Makdev Utility

utility:	**MAKDEV**
purpose:	This program creates a device file.
user access:	all users
summary:	makdev [-c] devname b/c majornum minornum
arguments:	device name
	block or character device specification
	major device number
	minor device number
options:	-c conditional

Description

The **Makdev** utility associates a device drive number with a name. After the program is executed, references to the device name refer to the device indicated by the device number.

Options

The **-c** option displays an error message if no device driver corresponds to the specified device number.

Notes

Makdev calls for two numbers in its arguments: a major device number, which is the driver number, and a minor device number, which is the device number.

Some utilities demand that certain devices be owned by **bin**. For example, **Spool** expects the print devices to be owned by bin. Use the **Chowner** utility to change device ownership as needed.

3.72 The Makdir Command

Shell command:	**MAKDIR** or **MAKD**
purpose:	This command creates a directory file.
user access:	all users
summary:	makdir dir1 [... dirN]
arguments:	directory pathname(s)
options:	none

Description

The **Makdir** command creates directory files.

3.73 The Make Utility

utility:	**MAKE**
purpose:	This utility automates the construction of executable programs from separate modules.
user access:	all users
summary:	make [-vdf] makefile [arg1 arg2 ...]
arguments:	instruction file with possible arguments
options:	-v verbose -d debug -f force

Description

Most complex programs are constructed from a number of separate modules. **Make** provides the means for automatically executing the necessary steps (i.e., compilations, assemblies, linkages) to construct a finished program. It also provides selective execution of just those steps necessitated by the modification of any of the constituent files.

Since the actions of **Make** are predicated on date and time, it is very important to keep the system time and date reasonably accurate.

Make takes its instructions from a text file, which must have the suffix **.mak**. The **.mak** file consists of two kinds of lines:

conditional statements--those with a colon somewhere in the line.
commands--those without a colon.

Make scans each conditional statement line. If any of the files following the colon have been modified later than any of the files preceding the colon, the commands on the command lines following the conditional statement are executed.

The following is an example based on the Cromemco 68000 FORTRAN-77 environment:

```
prog1.obj:      prog1.for progdata.for
        fortran prog
        code prog1.i prog1.obj
prog2.obj:      prog2.for progdata.for
        fortran prog2
        code prog2.i prog2.obj
prog.bin:       prog1.obj prog2.obj
        jlinker -oprog prog1 prog2 -lftn
```

The preceding instructions tell **Make** which parts of the program construction should be executed if

any of the constituent files have been modified.

Multiple files on either side of the conditional colon are permitted.

```
prog2.obj:      prog2.for progdata.for
```

is equivalent to:

```
prog2.obj:      prog2.for
prog2.obj:      progdata.for
```

and

```
foo.obj:        foo.for /usr/lib/ftnlib.obj
bar.obj:        foo.for /usr/lib/ftnlib.obj
```

is equivalent to:

```
foo.obj bar.obj: foo.for /usr/lib/ftnlib.obj
```

If the files to the right of the colon have not been modified more recently than those to the left of the colon, none of the commands are executed, and the **Make** utility scans for the next conditional statement.

The commands before the first condition line are executed unconditionally. Also, a line with only a colon forces the following commands to be executed unconditionally.

The **Make** utility allows the use of the **DIRectory** command to change the current directory. Also, arguments to the **Make** utility can be referenced anywhere in the **.mak** file in a shell-like fashion. The name of the make file may be referenced as **#0**.

Options

The **-v** option will display the program's progress.

The **-d** option will display the times that Make used for its conditional comparisons.

The **-f** option will cause all actions to be taken regardless of time consideration.

3.74 The Makfs Utility

utility:	**MAKFS**	
purpose:	This program sets up the structure for a file system on disk.	
user access:	privileged user	
summary:	makfs [-b #] [-i #] [-r #] [-s #] devname(s)	
arguments:	device name	
options:	-b #	first inode block number
	-i #	number of inodes
	-r	restore Superblock
	-s #	number of blocks used

Description

The **Makfs** utility sets up a structure for a file system on block devices. It establishes the number of inodes, the blocks dedicated to those inodes, blocks dedicated to the system, and blocks dedicated to the user.

Makfs is run on all floppy disks and on some hard disks before the disk is mounted for the first time.

The **Makfs** utility destroys any existing data on the device. It warns and prompts the user before destroying data.

The **Makfs** utility stores the inode number in all of the inodes created.

Options

The **-r** option restores the Superblock should it be accidentally destroyed. Use this option with caution. If you previously ran **Makfs** with the **-i**, **-s**, or **-b** option, give the **-r** option with the same argument given **-i**, **-s**, or **-b**. **Failure to do so will destroy the file structure.**

After running **Makfs** with the **-r** option, run **Icheck** to complete the restoration process.

The **-b** option specifies which block should be the first to contain inodes. Except for block 1 (Super block), preceding blocks are not used. The argument given **-b** cannot be less than 2.

If **-b** is not used, block 20 is selected by default, with blocks 0 and 2 through 19 allocated as follows: boot program (0, 2-17), partition table (18), alternate track table (19).

The **-i** option establishes a file system with a nonstandard number of inodes. This option is used only if you need more files than the default allows. Otherwise, **Makfs** decides how many inodes are needed and uses that number. **Makfs** rounds the number of inodes specified down to the nearest multiple of four (the number of inodes in a block).

The -s option specifies the number of blocks, from the beginning, to be used for the file structure. Blocks beyond the point specified by the argument to -s are not used.

If -s is not used, the maximum number of blocks is selected by default.

Notes

In lieu of running **Makfs** with the -r option, a more prudent method of restoring the Superblock is to use the **Fixsb** utility. **Fixsb** restores the Superblock and then runs **Icheck** automatically.

3.75 The Maklink Program

utility:	**MAKLINK**	
purpose:	This program creates another name for an existing file.	
user access:	all users	
summary:	maklink	[-fv] file-list dirname [-fv] source-file destination-file
arguments:	filenames followed by a directory name	
	or	
	source file followed by destination file	
options:	-f	force
	-v	verbose

Description

Every file you create has one link from its pathname to an inode. Thus, the Cromix-Plus system can access that file when you specify its pathname. The **Maklink** program creates additional links. In effect, **Maklink** creates another name (or pathname) for an existing file.

Options

The **-f** option causes the new link to overwrite another file with the same pathname if one exists. If the **-f** option is not used, and another file exists with the same name, an error results and **Maklink** is aborted.

The **-v** option displays the names of files as they are being linked.

Notes

No link is possible between two different file systems. That is, links cannot extend between two different devices (disks).

3.76 The Match Utility

utility: **MATCH**
purpose: This program finds all occurrences of a string within a file.

user access: all users

summary: match [-bceilqr] string file-list

arguments: string

file list

options:

-b	block numbers
-c	count
-e	exact match
-i	file names from STDIN
-l	line number
-q	quiet
-r	reverse match

Description

The **Match** utility searches through the specified files for all occurrences of the string and displays each line containing a match. Unless the **-e** option is used, **Match** is not case sensitive. If no file is specified, input is accepted from the standard input device.

Options

The **-b** option displays the block number with the matching line.

The **-c** option prints a count of the matching lines. The lines themselves are not displayed.

The **-e** option displays only lines that match the given string exactly - a case sensitive match.

The **-i** option causes that the file names of the files, on which **Match** is to work, are read from STDIN.

The **-l** option displays the line number together with the matching line.

The **-q** option does not display filenames where no match occurred.

The **-r** option reverses the sense of the match, displaying only lines that do **not** contain a match to the given string.

Notes

Strings of more than one word and ambiguous strings may be specified on the command line, surrounded by quotation marks. The same characters represent ambiguous strings as are used by the Cromix Shell (*, ?, []).

In addition, \n may be specified at the beginning or end of a string to force the match of that string at the beginning or end of a line of text, respectively. The search for the string is case insensitive unless the -e option is used. If the ambiguous characters * or ? are used, the string should be enclosed in quotation marks (").

If **Match** is used to search a file that is not a text file, control characters may be sent to the terminal. This may lock up the terminal; if you are using a Cromemco 3102 terminal, press CONTROL-Reset, or turn the terminal OFF and then ON again to restore terminal operation.

Example:

```
jim[1] who
jim             tty1        Apr-13-1987 15:40:08        200     2
roger           qtty2       Apr-13-1987 15:50:47        100     2

jim[2] who|match roger
roger           qtty2       Apr-13-1987 15:50:47        100     2

jim[3] scan . 'ext == ".c" && print(path)' | match -qei string
```

3.77 The Mode Utility

utility:	**MODE**
purpose:	This program displays or alters device modes.
user access:	all users
summary:	mode [devname] [characteristic(s)]
arguments:	optional device name optional characteristic(s)
options:	-v verify

Description

The **Mode** utility displays or alters the operational characteristics (or modes) of a device. If **Mode** is run without arguments, the modes of the device that issued the **Mode** command are displayed (e.g., your terminal).

To display the characteristics of another device, specify the device name as the first argument (you must have **read** access to the device, or else a "File not accessible" error occurs). If no modes are specified, **Mode** displays the characteristics of the selected device without changing them.

Device modes can be altered by specifying the desired settings as arguments (you must have **execute** access to the device, or else a "Channel access error" occurs). For example:

system[1] **mode lpt1 width 132 -tabexpand**

Some characteristics can be turned on by simply stating the appropriate name, and turned off by preceding the name with a dash (e.g., **-tabexpand**). Other characteristics must be followed by a number (e.g., **width 132**). All numbers are assumed to be decimal unless followed by an "**h**" for hexadecimal (e.g., **delaycode 7fh**).

Some characteristics use ASCII characters as values. A character value may be entered by pressing the ASCII key, or by typing its hexadecimal value. Control characters may also be entered by pressing the caret key (^) followed by the character. For example, the default line-kill character on terminal devices (CONTROL-U) can be changed to CONTROL-A in any one of three ways:

1. Type "**mode lkill**", then hold down the CONTROL key and press A

2. Type "**mode lkill 01h**"

3. Type "**mode lkill ^A**"

All commands are entered by pressing the RETURN key. Using method 2 or 3 above, you could make the RETURN key the user-signal key ("**mode sigchar 0dh**" or "**mode sigchar ^M**").

When displayed, the first part of each mode name is capitalized (e.g., Pause). The capitalized part of the name must be used when changing the characteristic (e.g., both "**mode tty1 -pa**" and "**mode tty1**

-**pause**" turn off the pause mode of tty1).

Option

The -**v** option displays the new mode settings after you change them.

Notes

In CBREAK, RAW, and BINARY modes, calls that read characters (**.rdbyte**, **.rdline**, or **.rdseq**) do not wait for a line terminator; they return after a single byte is entered. The Shell sets the mode to nonCBREAK, nonRAW, and nonBINARY each time it prompts for a new command line. A program, PROG, can be run in BINARY mode, by typing

jim[1] mode binary; prog

The function keys on a 3102 terminal are disabled when the Cromix-Plus Operating System is booted up. The mode command "**mode fn**" enables the function keys ("**mode -fn**" disables them again). Each function key allows you to send a 2-byte sequence (CONTROL-B (^B) and some other character) directly to a program. For example, CONTROL-B and "**p**" are sent when function key 1 is pressed. These 2-byte sequences can be used by application programs to perform special functions.

The following table shows the available modes for each character device. The modes are then described in detail, followed by a similar discussion for block devices.

CHARACTER DEVICES

	u t t y	t t y	o t y	q t y	s y d e v	t i m e r	u l p t	l p t	o s l p t	q s l p t	f p	c n e t	t y p	u s l p t	s l p t	t a p e	s c	s c t p
ABortenable	+	+	+	+	-	-	-	-	-	-	-	-	-	-	-	-	+	-
Append	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+
Baud	+	+	+	+	-	-	-	-	+	+	-	-	-	+	+	-	+	-
BINary	+	+	+	+	-	-	+	-	+	-	-	-	-	+	-	-	+	-
BLKsize	-	-	-	-	-	-	-	-	+	-	-	-	-	+	+	-	-	-
BLKSwrt	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	-	-
Block	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	-	+
BMargin	+	+	+	+	-	-	+	+	+	+	-	-	+	+	+	-	-	-
CBreak	+	+	+	+	-	-	+	-	+	-	-	-	-	+	-	-	+	-
CHECK	-	-	-	-	-	-	-	-	-	-	-	-	+	-	-	-	-	-
Correction	-	-	-	-	-	+	-	-	-	-	-	-	-	-	-	-	-	-
Corr error	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	-	-
CRDEVice	+	+	+	+	-	-	+	+	+	+	-	-	+	+	+	-	+	-

CRIGNore	-	-	-	-	-	-	-	-	+	-	-	-	-	+	+	-	-	-
CWidth	-	-	-	-	-	-	-	-	-	-	-	-	+	-	-	-	-	-
DELAY	+	+	+	+	-	-	+	+	+	+	-	-	-	+	+	-	+	-
DELECho	+	+	+	+	-	-	-	-	-	-	-	-	-	-	-	-	+	-
DIScard	+	+	+	+	-	-	+	+	+	+	-	-	+	+	+	-	+	-
ECho	+	+	+	+	-	-	-	-	-	-	-	-	-	-	-	-	+	-
End of tape	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	-	+
EOFclose	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	-	+
Erase	+	+	+	+	-	-	-	-	-	-	-	-	-	-	-	-	+	-
ESCreturn	+	+	+	+	-	-	-	-	-	-	-	-	-	-	-	-	+	-
EVENparity	+	+	+	+	-	-	-	-	+	+	-	-	-	+	+	-	+	-
FFexpand	+	+	+	+	-	-	+	+	+	+	-	-	-	+	+	-	+	-
File	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	-	+
File mark	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	-	-
FLUSHtime	-	-	-	-	-	+	-	-	-	-	-	-	-	-	-	-	-	-
FMark	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	-	+
FNkeys	+	+	+	+	-	-	-	-	-	-	-	-	-	-	-	-	+	-
FORMs	-	-	-	-	-	-	+	+	+	-	-	-	+	+	+	-	-	-
GMToffset	-	-	-	-	-	+	-	-	-	-	-	-	-	-	-	-	-	-
HANDshake	-	-	+	-	-	-	-	-	+	-	-	-	-	-	-	-	-	-
Hard error	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	-	-
High speed	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	-	-
HUPenable	-	-	+	+	-	-	-	-	-	-	-	-	-	-	-	-	+	-
IBsize	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	-	-
INL	+	+	+	+	-	-	-	-	-	-	-	-	-	-	-	-	-	-
IMmediateecho	+	+	+	+	-	-	-	-	-	-	-	-	-	-	-	-	+	-
LCase	+	+	+	+	-	-	+	-	+	-	-	-	-	+	-	-	+	-
Length	+	+	+	+	-	-	+	+	+	+	-	-	+	+	+	-	+	-
LHeight	-	-	-	-	-	-	-	-	-	-	-	-	+	-	-	-	-	-
LKill	+	+	+	+	-	-	-	-	-	-	-	-	-	-	-	-	+	-
LMargin	-	-	-	-	-	-	-	-	-	-	-	-	+	-	-	-	-	-
Load	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+
Load point	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	-	+
NLECho	+	+	+	+	-	-	-	-	-	-	-	-	-	-	-	-	+	-
OBsize	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	-	-
ODDparity	+	+	+	+	-	-	-	-	+	+	-	-	-	+	+	-	+	-
ON LINE	-	-	-	-	-	-	-	-	-	-	-	-	+	-	-	+	-	+
PAPER OUT	-	-	-	-	-	-	-	-	-	-	-	-	+	-	-	-	-	-
PAuse	+	+	+	+	-	-	+	-	+	-	-	-	-	+	-	-	+	-
PSthimble	-	-	-	-	-	-	-	-	-	-	-	-	+	-	-	-	-	-
RAW	+	+	+	+	-	-	+	-	+	-	-	-	-	+	-	-	+	-
READY	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	-	+
RESume	+	-	+	+	-	-	+	-	+	-	-	-	-	+	-	-	+	-
Rewind	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	-	+
Rewinding	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	-	-
RIBBON OUT	-	-	-	-	-	-	-	-	-	-	-	-	+	-	-	-	-	-
Secure	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	-	+
SIGAllchars	+	+	+	+	-	-	-	-	-	-	-	-	-	-	-	-	+	-
SIGChar	+	+	+	+	-	-	-	-	-	-	-	-	-	-	-	-	+	-

SIGenable	+	+	+	+	-	-	-	-	-	-	-	-	-	-	-	-	+	-	
SIGHUPall	-	-	+	+	-	-	-	-	-	-	-	-	-	-	-	-	+	-	
SOFTerr	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	
TABexpand	+	+	+	+	-	-	+	+	+	+	-	-	-	+	+	-	+	-	
TANdem	+	+	+	+	-	-	-	-	-	-	-	-	-	-	-	-	+	-	
TDelay	+	-	+	-	-	-	+	-	+	-	-	-	-	+	-	-	-	-	
Unload	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	-	+	
Width	+	+	+	+	-	-	+	+	+	+	-	-	+	+	+	-	+	-	
WRAParound	+	+	+	+	-	-	+	+	+	+	-	-	+	+	+	-	+	-	
Wrt protect	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	-	+	

ABortenable

When the ABortenable switch is on ("**mode ab**"), pressing CONTROL-C sends a SIGABORT signal to all processes controlled by the terminal; when off ("**mode -ab**"), CONTROL-C is treated like any other character.

Append

Position a tape to end of recorded data to allow appending to it.

Baud

The Baud parameter, followed by a number, sets the baud rate of a serial device (e.g., "**mode b 9600**" changes the baud rate of your terminal to 9600).

The baud rates designated **Auto**, **Nochg**, and **Ctswait** are special cases. If **Auto** mode is used with a terminal, the driver tries different baud rates until it reads a RETURN from the input. If **Nochg** mode is used, the baud rate has been previously established (e.g., by RDOS). If **Ctswait** is used, the driver waits for a CTS signal from the modem.

BINary, CBreak, and RAW

If any of these parameters are enabled, any read from the device returns after each input character. These parameters serve to disable the action of various other parameters, as shown in the following table (+ means that the parameter causes the given effect, a space means it does not.)

Effect	CBreak	RAW	BINary
Return after each character input	+	+	+
No erase, linekill, or EOF (CONTROL-Z) characters	+	+	+
No output PAuse or output Width			

truncation	+	+
Treat XOFF (CONTROL-S), XON (CONTROL-Q) as regular input	+	+
No tandem mode - no input buffer flow control		+
Treat CONTROL-C and SIGChar key as regular input		+
No checking or changing of input parity bit		+
No delays after any output control characters such as tabs		+
No echoing of input		+
No function key decoding		+
No character transformations - ignore the LCase, CREDEVice, and TABexpand settings		+

BLKsize

A serial printer can accept only a limited number of characters without some kind of handshaking. With the ETX/ACK protocol, the driver sends one block of characters followed by an ETX character and waits until the printer returns an ACK character before sending the next block. The number of characters in a block never exceeds BLKsize characters. If an escape sequence is detected, the current block is immediately terminated by an ETX character, thereby handling escape sequences correctly provided no escape sequence exceeds BLKsize characters. Refer to your printer documentation for the value of BLKsize.

BLKSwrt

BLKSwrt is the number of the blocks written when a tape file was last written to a tape device. It cannot be changed with the **Mode** utility.

Block

Block is the command for tape positioning. It should be followed by a space and the block number. Tape blocks are numbered 1, 2, 3, ..., up to the number of blocks in the file. If the specified block is larger than the total number of blocks in a file, the tape moves to the beginning of the next file.

BMargin

If a printer device is within BMargin lines of the bottom of the page, a formfeed takes the device to the top of the next page. The length of a page is determined by the parameter Length (see below).

CBreak

See BINary.

CHECK

CHECK indicates a malfunction, which can only be resolved by DISCARD or a power OFF/ON sequence.

Correction This is the number of seconds per 100 days to be added to or subtracted from the system timer.

Corr error

Corr error is a flag that is set when the tape formatter detects a correctable error. The flag cannot be set by the **Mode** utility.

CRDEVice

The CRDEVice switch must be on for a carriage return device and off for a newline device. If CRDEVice is on, a RETURN character read from the device is translated into a newline character before being passed to the calling program (and a RETURN, linefeed sequence is echoed to the device). On output, newlines are translated into RETURN, linefeed sequences. If CRDEVice is off, no translations are made.

The single newline character (0Ah) performs a RETURN, linefeed sequence.

CRIGNore

Some printers, such as the CLQ, need only a RETURN character (0Dh) to move to the start of a new line. If CRIGNore is on, the driver ignores all RETURN characters and translates newline characters into RETURN characters.

CWidth

The value of CWidth defines the width of each character in units of 1/120th of an inch. The default is 12, or 10 characters per inch.

DELAY

The DELAY is the decimal equivalent of a byte determining the amount of delay inserted after certain characters are sent to the output. If the TDelay switch is set, the output process is suspended for some multiple of one-tenth of a second. If the TDelay switch is reset, the delay is implemented by sending a number of NULL characters. The old drivers do not recognize the TDelay switch. The Tty driver behaves as if TDelay is always off, the Qtty driver behaves as if TDelay was always on.

The bit assignments for DELAY are:

Character	DELAY Bits	TDelay ON (seconds)	TDelay OFF (chars)
newline	0 and 1	0, .1, .2, .3	0, 4, 8, 12
tab	2 and 3	0, .1, .2, .3	0, 4, 8, 12
carriage return	4 and 5	0, .1, .2, .3	0, 4, 8, 12
formfeed	6	0, .8	0, 128
backspace	7	0, .1	0, 4

For example, "mode qtty1 delay a3h" sets the QTTY1 newline delay to 0.3 seconds, the RETURN delay to 0.2 seconds, the backspace delay to 0.1 seconds, and the TAB and formfeed delays to zero.

DELECho

On TTY, QTTY, and MTTY devices, the character following DELECho is echoed in response to any one of the delete characters. If the letter R follows DELECho, the echoed response to a delete character is: backspace, space, backspace.

DIScard

When a driver is first used, a data area for that device is initialized. Mode settings are determined at this time. Values are changed or not changed as the **Mode** calls dictate, and these values remain even if the device is closed. If DIScard is set, closing a device discards all information, and all tables are reinitialized when the device is next opened.

TTY and LPT drivers define the actual devices during system generation (refer to **Crogen**). To attach a parallel printer to a connector that was used for a TTY terminal, you must generate (and reboot) a properly configured Cromix-Plus Operating System.

ECho

If the ECho switch is on (no preceding dash), characters entered on the terminal are echoed to the screen. ECho cannot be changed by the **Mode** utility.

End of tape

End of tape is a flag describing the tape device status. It cannot be set by the Mode utility.

EOFclose

If EOFclose is set, the tape controller writes a double file mark when the device is closed.

Erase

The character following Erase serves as an extra erase character, along with DEL (7Fh) and CONTROL-H (08h, also referred to as backspace). For example, the command

```
jim[1] mode erase _
```

selects the underscore as an auxiliary delete character. Note that DEL and backspace still function as delete characters.

ESCreturn

If this switch is enabled, the ESCAPE character terminates the .rdline and .rdseq functions as if RETURN had been pressed. EVenparity and ODDparity

ODDparity and EVenparity

Control the handling of the parity bit, as follows (+ means enabled, - means disabled).

EVenparity	ODDparity	Function for Input Characters
-	-	does not check parity but strips parity bit
+	-	checks for even parity before stripping parity bit
-	+	checks for odd parity before stripping parity bit
+	+	leaves parity unchecked and unchanged

EVenparity	ODDparity	Function for Output Characters
-	-	strips parity bit
+	-	makes character even parity
-	+	makes character odd parity
+	+	leaves parity bit unchanged

FFexpand

If FFexpand is on, each formfeed character (0bh) sent to a printer device is converted to enough newlines to move the paper to the top of the next page. The length of a page is determined by the

Length mode. If FFexpand is off, the formfeed character is sent without conversion.

File

On a tape device, a file number following the File mode moves the tape to the start of that file. Tape files are numbered from 1 through the number of files on the tape. The following command moves the tape to the sixth file on TP1:

```
jim[1] mode tp1 file 6
```

If the specified file number is larger than the total number of files recorded on the tape, the tape moves to the end of tape reel. This motion may be aborted by taking the tape drive off-line and entering CONTROL-C at the terminal keyboard.

File mark

File mark is a flag describing the tape device status. It cannot be set by the **Mode** utility.

FLUSHtime

The system will write out all modified disk buffers after FLUSHtime seconds of no activity.

FMark

FMark is a command to write the file mark on a tape device (e.g., "mode tp1 fm").

FNkeys

If FNkeys is on, the driver echoes a CONTROL-B for each of the two bytes sent when a function key on the Cromemco 3102 terminal is pressed. This allows the 2-byte function key sequences to be passed to a program.

FORMs

A printer driver that understands the concept of paper forms can be told what paper form is loaded into the printer (or made to recognize some other physically distinct change). The printer daemon prints only those files that were spooled with the same forms number (refer to the -z option of the Spool utility.)

When the operator needs to change paper forms (print thimble, etc.), the FORMs number can be set to any unused value. Then, after the current file is printed, the operator can make the change, set any necessary modes, and select a new FORMs value. After the printer daemon is restarted, all files with the new FORMs value will be printed.

The actual FORMs numbers are left to the System Administrator.

GMToffset

This is the number of hours to be added to GMT time to get local time.

HANDshake

This hardware handshaking (RTS - CTS signals). If this mode is enabled, the driver will drive and monitor the RTS/CTS lines. The other part of the communications system must also drive and monitor its side of RTS/CTS lines.

Hard error

The Hard error flag is set on a tape device when the tape formatter finds an error that it cannot correct. This flag cannot be set by the **Mode** utility.

High speed

High speed is a flag describing the tape device status. It cannot be set by the **Mode** utility.

HUPenable

If this switch is on and an IOP/OCTART terminal device closes, the modem on the IOP device is hung up.

IBsize

IBsize defines the length in bytes of the first block of the last file read from a tape device. It cannot be changed by the **Mode** utility.

INL

If INL is ON, the RETURN character will not be translated into a NEWLINE character, even if the device is a carriage return device. Output is not affected.

IMmediateecho

If IMmediateecho is on, characters typed ahead are echoed immediately, and echoed again when they are read. If IMmediateecho is off, characters are echoed only at the time they are read.

LCase

If LCase is on, terminal devices TTY, QTTY, and MTTY convert uppercase alphabetic input characters to lowercase.

Length

This is the page length in lines of the designated device. When the Mode utility displays the page length, the word length is followed by the specified page length. To change the page length, use the argument length followed by a space and the desired page length.

LHeight

The value of LHeight defines the height of a printed line in units of 1/48th of an inch. The default value is 8 (6 lines per inch).

LKill

The LKill character deletes the current input line for terminal drivers. This performs multiple deletes back to the last prompt character.

LMargin

The value of LMargin defines the first printable position, expressed in units of .1 inches (the default is 0).

Load

Load a new tape after the old tape has been unloaded.

Load point

Load point is a flag describing the tape device status. It cannot be set by the **Mode** utility.

NLECho

Determines if a newline character will be echoed.

OBsize

OBsize is a command to set the block length of files written to the tape device. Keyword OBsize

should be followed by a blank and the desired size. The following command sets the output block length of TP3 to 4096 bytes:

```
jim[1] mode tp3 ob 4096
```

ODDparity

See EVENparity.

ON LINE

Signifies the device is on-line (powered on or loaded).

PAPER OUT

Signals the printer is out of paper.

PAuse

If PAuse is on, terminal devices pause after the number of lines specified by Length has been output. The output resumes only after a character is entered on the keyboard. Mode PAuse can also be used on new printer drivers. As a printer does not have a keyboard to enter XON, the same effect can be achieved by the "Mode prt resume" command.

PSthimble

When using the Typ driver to operate a "spinwriter" printer, the setting of PSthimble (proportional spacing) must agree with the print thimble in use. The default setting (-PSthimble) supports the normal (non-proportional) print thimble. If printed copy is unreadable (or readable, but erratically spaced), check the setting of PSthimble.

RAW

See BINary.

READY

READY is a flag describing the tape device status. It cannot be set by the **Mode** utility.

RESume

If a printer driver allows the mode to be set to PAuse, the printer will stop after a full page. Mode RESume will have cause the printer to continue printing.

Rewind

Rewind is a command to rewind the tape device (e.g., "mode tp1 r").

Rewinding

Rewinding is a flag describing the tape device status. It cannot be set by the **Mode** utility.

RIBBON OUT

Signals the printer is out of ribbon.

Secure

Secure is a command to erase the tape at high speed (e.g., "mode tp1 s").

SIGenable, SIGChar, and SIGALLchars

If SIGenable is on and SIGALLchars is off, pressing the SIGChar key causes terminal devices to send a SIGUSER signal to all processes controlled by the terminal. The SIGChar key character is not put into the input stream. If SIGenable is off, then the SIGChar key is treated in the same manner as any other key.

The terminal which controls a process is the terminal on which the owner of the process logged on to the system.

If SIGenable and SIGALLchars are both on, pressing the SIGChar key causes the SIGUSER signal to be sent to all processes controlled by the terminal, but the SIGChar key character is also put into the input stream.

If SIGALLchars is on but SIGenable is off, every terminal keystroke pressed before a system call to read input has been made sends the SIGUSER signal to all controlled processes. (Only characters typed-ahead send signals.) The characters are also put into the input stream.

Note that shells are set up to ignore SIGUSER signals, so that a user is not logged off by them. Any program running in a nondetached mode that does not either ignore or trap SIGUSER signals is aborted by them. The .signal system call provides a means for ignoring or trapping signals.

SIGHUPall

If this switch is on and the modem of an IOP terminal device hangs up, the signal SIGHANGUP is sent to all processes controlled by the device. A process is controlled by the terminal from which the user who initiated the process logged in. For example, a user who has logged in on QTTY1 and hangs up without logging out is logged off by the resulting SIGHANGUP signal, provided SIGHUPall is enabled.

SOFTerr

Number of soft (nonfatal) tape errors. Can be reset by "mode stp1 soft 0".

TABexpand

If TABexpand is on, every TAB character (09H) output is converted to enough spaces to bring the output to the next standard TAB stop. Standard tab stops are multiples of 8 at columns 1, 9, 12, etc., on the terminal.

TANdem

Tandem mode allows a receiving Cromix system to control the rate of input data using the DC1/DC3 handshaking protocol. The device sending data may be a Cromix system or another computer. When used to communicate between two Cromix systems, the ttys on the sending and receiving systems should not be selected in the ttys files. Both drivers should be set to the same Baud rates, have RAW mode enabled, and ECho and CRDevice disabled.

The receiving system should have TANdem mode enabled, and the receiving program or command file should already be executing before sending begins. In TANdem mode, the receiving system sends a DC3 (XOFF) character when its tty driver buffer is full, and sends a DC1 (XON) character when the driver is ready to accept more characters.

TDelay

If the TDelay switch is on the delay after ceratin control characters is implemented by waiting some time after the character is sent out. If the TDelay switch is off, the delay is implemented by a number of NULL character being sent out after the control character.

See the description of Mode DELAY.

Unload

Unload is a command to unload the tape device (e.g., "mode tp1 u")

Width

The Width function specifies the number of columns displayed before truncation or wrap-around. If Width = 0, no truncation or wrap-around occurs.

WRAParound

If WRAParound is on, and the device output column reaches the page Width, an extra newline is sent to the device. This allows the remainder of the output line to be printed on the next line. If WRAParound is off, the remainder of the line is truncated. If Width = 0, no truncation or wrap-around occurs.

Wrt protect

Wrt protect is a flag describing the tape device status. It cannot be set by the **Mode** utility.

BLOCK DEVICES

	c f l o p	u f l o p	a l f m e m	l s t d c	r a m d s k	t f l o p	s m d	h d	a e m	e s d i
ADDress	-	-	-	-	+	-	-	-	-	-
BSTEP	-	-	-	+	-	-	-	-	-	-
CDOS	+	+	-	+	-	-	-	+	-	+
CROMIX	+	+	-	+	-	-	+	+	-	+
CYLinders	+	+	-	+	-	+	+	+	-	+
DDensity	+	+	-	-	-	-	-	-	-	-
DSide	+	+	-	-	-	-	-	-	-	-
Free	-	-	-	-	-	-	-	-	+	-
HARDerr	+	+	-	+	-	+	+	+	-	+
ICACHE	-	-	-	-	-	-	-	-	+	-
MFree	-	-	-	-	-	-	-	-	+	-
MOUNTed	+	+	-	+	+	-	+	+	-	+
MSys	-	-	-	-	-	-	-	-	+	-
MTot	-	-	-	-	-	-	-	-	+	-
READonly	+	+	-	+	+	-	+	+	-	+
RETENsion	-	-	-	-	-	+	-	-	-	-
RETRY	+	+	-	+	-	+	+	+	-	+
RPM	+	+	-	+	-	-	-	+	-	-
SECSIZ	+	+	-	+	-	+	+	+	-	+
SECtors	+	+	-	+	-	+	+	+	-	+
SIZE	+	+	-	+	+	+	+	+	-	+
SOFTerr	+	+	-	+	-	+	+	+	-	+

SURfaces	-	-	-	+	-	+	+	+	-	+
VERIFY	+	+	-	+	+	-	+	+	-	+
VERsion	+	+	-	+	+	+	+	+	-	+
VOICEcoil	+	+	-	-	-	-	-	-	-	-
WRITEprotect	+	+	-	+	-	+	+	+	-	+

ADDress

This value indicates, in hexadecimal notation, the starting address of the RAM disk device.

BSTEP

This switch indicates whether the driver is using buffered step mode.

CDOS

This switch is set if the device is a CDOS device.

CROMIX

This switch is set if the device is a Cromix device.

CYLinder

This value indicates, in decimal notation, how many cylinders are on the device.

DDensity

This switch is set if the device is a double-density device.

DSide

This switch is set if the device is a double-sided device.

ECache

This flag describes the state of external cache memory. It can be turned on or off. Setting this switch is ineffective on the DPU or XPU board.

Free

The Free value is the total amount of unused memory (not necessarily continuous).

HARDerr

This is the total number of hard errors reported on the raw console. To change the accumulated total number, use the argument harderr followed by a space and the desired total.

ICache

This flag describes the state of the 68020 on-chip cache memory. It can be turned on or off. Setting this switch is ineffective on the 68000 or 68010 processor.

MFree

The Mfree value is the remaining amount of continuous unused memory.

MOUNTed

This switch is set after the device has been mounted (_**mount** system call).

MSha

The Msha value is the amount of memory shared by many users (Shared code and shared data).

MSys

The amount of memory occupied by the operating system proper.

MTot

The total amount of memory the system is running on.

READonly

This switch is set if the device is mounted for read only.

RETENsion

This value indicates the number of tape repositions before a retension is performed on the floppy tape device. Setting this value to 0 (e.g., "**mode ftcd reten 0**") disables the retension feature. Setting this

value to a negative number causes an immediate retension, but no change to the old value.

RETRY

This value indicates the number of times a read operation will be retried.

RPM

RPM value defines the rotational speed of the device (in revolutions per minute).

SECSIZ

This field defines the number of bytes per sector.

SECtors

This field defines the number of sectors per track.

SIZE

SIZE defines the size of the device, in kilobytes.

SOFTerr

This is the total number of soft errors (retries). To change the accumulated total number, use the argument softerr followed by a space and the desired total.

SURfaces

This field defines the number of surfaces on the devices.

VERIFY

This switch is set if the device reads back data after writing it to check for errors.

VERsion

VERsion indicates the version number of the firmware (if applicable) or the version number of the iolib library where the driver was introduced. The iolib library itself may have a higher version number if other components of the library have been changed.

VOICEcoil

This switch is set if the device uses a voice coil positioning mechanism.

WRITEprotect

This switch is set if the device is physically write protected.

XMM

This switch can be used to turn XMM memory management unit ON or OFF. Cromix version 31.13 and later will always try to use the XMM unit provided the XMM unit is present in the system. (XMM cannot be turned on if there is no XMM board in the system). With XMM enabled any user access to illegal memory will cause the offending process to be aborted with a Bus error. Details will be displayed on the raw terminal.

3.78 The More Utility

utility:	**MORE**
purpose:	This program displays pages from pipes or from user files.
user access:	all users
summary:	more [-cfs] [-n number] [file-list]
arguments:	optional list of files
options:	-c draw pages from top of screen -f count true lines (not screen lines) -s do not display multiple blank lines

Description

The **More** utility allows examination of continuous text, one screenful at a time. Under normal circumstances, **More** will display a screenful of text and wait for a command. When the user presses the SPACE bar, the next screenful will be displayed (See the **Help** utility description).

The **More** utility will display all given files one after another. If no file is specified, the **More** utility will display text from STANDARD INPUT. This allows results of a program to be piped into the **More** utility to be viewed at leisure. As the **More** utility can receive the text to be displayed from STANDARD INPUT, it also uses **/dev/tty** (user's terminal) for reading commands and displaying data.

Following the screenful of data, the More utility prints the text **--More--** at the bottom of the screen and waits for a command to be typed. If **More** is displaying an ordinary file, it will also print the percentage of the total text displayed so far.

More reads the commands in CBREAK mode with no echo. This means that the command typed in will not be echoed and will be acted upon the moment it is recognized. Usually, a single keystroke will suffice.

All commands can be preceded by a decimal integer which defaults to value one when not given explicitly. This number, if it has any meaning at all, is the replication factor. In the description below the replication factor is denoted by **[n]**.

The available commands are:

[n]SPACE	Skip **n**-1 screenfuls of data and display next screen of data.
[n]RETURN	Advance display by **n** lines.
=	Print current line number (and file name if applicable).
DELETE	Cancel partially typed command and redisplay the **--More--**

prompt.

h or H	Print short overview of available commands.
b or B	Move back to the beginning of file. Illegal if viewing piped input.
[n]n or N	Go to the **n-th** next file in the list. If this will try to move past the last file, the last file will be displayed.
[n]p or P	Go to the **n**-th previous file in the list. If an attempt is made to move past the first file, the first file will be displayed.
q or Q	Terminate the **More** session.

Options

Without the **-c** option, text is displayed by simply writing out lines and scrolling old lines from the screen. With the **-c** option the More utility will start writing at the top of the screen and erase each line before writing to it. This option will only be effective if the terminal has the **ce** (clear to the end of line), and **ho** (cursor home) terminal capabilities.

The **-f** option will change the way lines are counted. Without the **-f** option each screen line will count as one. This means that a long source line will be displayed as multiple lines also and counted as a multiple line. With the **-f** option, only the new line character increments the line count.

The **-s** option will cause multiple blank (screen) lines to be replaced by a single screen line.

The **-n** option must be followed by an integer specifying the number of lines that form a screenful. Without the **-n** option the screenful size is defined to be **li** - 1 where **li** is the **termcaps** value of the number of lines on the screen.

Notes

The **More** utility uses the following **termcaps** values. None are mandatory-**More** can work with dumb terminals.

li	Number of lines on the screen (set to 24 if not defined).
co	Number of columns on the screen (set to 80 if not defined).
ce	Clear to the end of the line (**-c** option ineffective if not defined).
ho	Cursor home (**-c** option ineffective if not defined).
so	Start highlighting (not used if not defined).

se Stop highlighting (not used if not defined).

3.79 The Mount Utility

utility:	**MOUNT**
purpose:	This program enables access to a file system.
user access:	privileged users
summary:	mount [-r] [devname dummyname]
arguments:	device name
	file pathname
options:	-r read only

Description

The **Mount** utility enables access to a file system.

When called without arguments, **Mount** lists the currently mounted devices.

The **Mount** utility looks on the disk to be mounted for the file **/etc/passwd**. Finding that file, it looks for the special user name *mount*. If this name is present and has a password associated with it, **Mount** prompts the user for the password before mounting the disk. Thus, it is possible to protect disks from being mounted by an unauthorized user.

User Access

By editing the file **/etc/mutab**, a privileged user can allow selected, non-privileged users to **mount** specified devices. The **Mount** utility consults this file whenever a non-privileged user gives the **Mount** command.

Each line in **mutab** has up to four fields, separated by colons:

flag : device : username : groupname

Flags can be zero (entry disabled) or one (entry enabled). The **device** field must be the name of a block device in the **/dev** directory. **Username** is the user allowed to mount that device. If **groupname** is specified (field 4), all members of that group may mount the device. If **username** or **groupname** is not specified, all users may mount the device.

A typical **mutab** file is shown here:

```
0:fda
1:fdb : tom
1:fdb : : users
```

Line 1: Because a username or groupname is not specified, any user can mount device **fda**. In the example, this entry is disabled by a zero flag in field 1.
Line 2: User **tom** may mount device **fdb**
Line 3: All members of the users group may mount device **fdb**.

Options

The **-r** option causes the file system to be mounted for read-only access. The **Mount** utility will fail if the user tries to mount a write protected device without the **-r** option.

Notes

A file system that has been mounted must be unmounted by the use of the **Unmount** utility before the mounted disk is removed from the system. If this is not done, the integrity of the data on the mounted system cannot be assured.

Do not attempt to mount a file system on a nonexistent device. Devices which do not exist may be deleted from the **/dev** directory.

Example:

```
jim[1] create newfilesys
jim[2] mount fdb newfilesys
jim[3] ls -m
.
.
.
145 D newfilesys

jim[4]
```

In the example above, the user first creates a dummy file. After mounting, the name of this dummy file becomes the root directory name of the file system to be mounted. After unmounting, this name becomes a dummy filename once again.

The **Mount** command is given with the device name where the file system is located. Refer to the *Cromix-Plus System Administrator's Guide* for a complete list of device names.

The **Ls** utility shows that the new file system has been mounted and gives the name of the root directory.

3.80 The Move Program

utility:	**MOVE**
purpose:	This program moves file(s) from one directory into another.
user access:	all users
summary:	move [-ftv] file-list dirname [-ftv] srcfile destfile
arguments:	two single file pathnames **or** one or more file pathnames **and** a directory pathname
options:	-f force -t time -v verbose

Description

The **Move** program moves one or more files from one directory to another directory. This program removes the source file(s). The **Move** program does not change the access privileges of the moved files. If files are transported from directory **A** to directory **B**, the owner of directory **B** may not have full access privileges for the files. The program **Chowner** must be run to change the owner of these files.

Options

The **-f** option causes the moved file to overwrite another file with the same pathname if one exists. If this option is not used and another file exists with the destination pathname, an error is generated and the **Move** program aborted.

The **-t** option causes a file to be moved **only** if:

1. The file does not exist in the destination directory; or

2. The source file was modified more recently than the destination file. This comparison is performed on a file-by-file basis.

The **-v** option displays the names of the files being moved.

Notes

With the exception of "time dumped", which is automatically zeroed, files are moved with ownership and times preserved.

3.81 The Msg Utility

utility:	**Msg**
purpose:	This program sends messages between users.
user access:	all users
summary:	Msg [-any2] [user-name or devname]
arguments:	text terminated by CONTROL-Z
options:	-a all -n disable -y enable -2 Cromemco 3102 terminal

Description

The **Msg** utility sends messages between users or from a user to a device. Sending a message to a device is useful when a device is online but no user is in attendance.

If **Msg** is typed and immediately followed by a RETURN, then a message is displayed to inform the user of the status of incoming messages. Incoming messages may be disabled or enabled by using the **-n** and **-y** options. Terminating a message with CONTROL-Z automatically sends the message **End of message** to the receiving user.

The **Msg** command followed by a user or device name allows a message to be entered. The message is transmitted to the destination user after each RETURN is pressed. A CONTROL-Z terminates the message and returns the originating user to the Shell.

Options

The **-a** option broadcasts a message to all users currently logged on to the system. This can be used by the privileged user to warn other users of interruptions to system usage such as rebooting. This message is sent to all users whether or not they have message receiving enabled. The message is preceded by the warning **Broadcast message**. Only privileged users are permitted to use this option. A message sent with the **-a** option is not transmitted until the entire message is given. Hence, when the **-a** option is specified, it may be followed on the command line by the name of a file that contains a broadcast message.

The **-n** option causes incoming messages to be disabled.

The **-y** option allows incoming messages to be received.

The **-2** option sends messages to the status line of a Cromemco 3102 terminal.

Notes

To clear the status line of a Cromemco 3102 terminal after receiving a message transmitted using the **-2** option, type **CONTROL-shift** followed by **CONTROL-1**.

If two-way communication is desired, a protocol should be established to prevent the confusion that arises when two messages are transmitted simultaneously. A suggested protocol follows: One user transmits at a time. A single **o** (short for over) is transmitted on a line by itself to indicate the end of the message. Upon seeing the **o**, the other user responds, terminating the message with an **o**. When the entire communication is finished, one user transmits **oo** (short for over and out) followed by a **CONTROL-Z**. The other user should type a **CONTROL-Z** also.

Two-way communication can be established by the **Msg** utility. When a user receives a message:

Message from xxxx

the receiving user should type:

Msg xxxx

This allows users to send each other messages. In the example above, **xxxx** represents a user name.

3.82 The Ncheck Utility

utility:	**NCHECK**
purpose:	This program displays file information.
user access:	all users
summary:	ncheck [-i inode-#'s] [dirname or filename]
arguments:	directory or file pathname
options:	-i list of inode numbers

Description

The **Ncheck** program displays the inode number, link count, and pathname of all files contained in the specified directory and all subdirectories. If no arguments are supplied, **Ncheck** uses the root directory.

Options

The **-i** option, which displays information about specified inodes only, requires an argument. This argument can be one or more inode numbers separated by commas or it can be one or more inode numbers separated by spaces with the entire list enclosed in quotation marks.

3.83 The Newuser Utility

utility:	**NEWUSER**
purpose:	This program displays information for new users.
user access:	all users
summary:	newuser
arguments:	none
options:	none

Description

The **Newuser** utility displays the file **newuser.msg**, which contains important information about the present version of the Cromix-Plus Operating System.

3.84 The Passwd Utility

utility:	**PASSWD**
purpose:	This program changes the passwd and group files
user access:	all users
summary:	passwd [-dgnp] [user1 user2...]
arguments:	user1 user2...
options:	-d delete -g group -n new user -p prompt

Description

The **Passwd** utility has four functions. It may be used to change a user's own password or prompt. A privileged user may use it to add and delete from the list of users permitted to log on to the system. By using the delete function followed by the add function, the privileged user may change the login status of any user.

In any one of these four modes of operation, user name(s) are specified either on the command line or during the execution of the **Passwd** program.

To change the password only, enter the command **passwd** followed by a RETURN. The **Passwd** program prompts for a user name and a new password.

Options

The **-d** option deletes a specified user or group.

The **-g** option alters the **/etc/group** file (instead of the **/etc/passwd** file).

The **-n** option adds new user(s) or group(s).

The **-p** option will cause the program to change the prompt string instead of the password.

Establishing a New User

A new user may be added using the **Passwd** program. In the following example, the user logs on as the privileged user **system** and creates a new user **fred** with the password **mountain**:

Login: **system**

```
Logged in system May 18-1986 17:12:15 on console
system[1] passwd -n

Name: fred
Password:  xxx
User number: 5
Group number: 0
Directory: /usr/fred
Starting Program:
Prompt string: [%d]->

Name:
system[2]
```

The **Passwd** program prompts for a user name. The response to this prompt is the user name typed in response to the **Login:** prompt. Press RETURN after entering the name.

Next, the program prompts for a user password. If no password is desired, press RETURN in response to the prompt. If you do enter a password, it is encrypted, and the encrypted password displayed on the screen. When a user logs on, this password must be entered after the password prompt.

The program prompts for the user and group identification numbers. Each of these is an unsigned integer between 0 and 65535. A zero in the user field indicates a privileged user. A zero in the group field indicates the user is not a member of any group. Any other number has significance only within a given system.

The **Directory:** prompt allows specification of an initial directory, which is the user's home directory. If this directory does not exist, the system creates one. The user is the owner of this directory. If the home directory already exists, the **Passwd** utility prints this information.

Next, the **Passwd** program prompts for a **Starting program**. If RETURN is pressed in response to the prompt, the user has full use of the Shell program. If the name of a program is entered here, the user is brought up running the program specified and is logged off upon exiting from the program. Any valid Shell command line may be entered in response to this prompt.

Finally, the **Passwd** program prompts for the Prompt string. If RETURN is pressed in response to the prompt, the login name followed by the command number in brackets will be used as a prompt string. Any other string may be entered instead. If the string contains the characters "**%d**", these characters will be replaced by the command number when the prompt is displayed.

Deleting a User

A user is deleted from the list of users (**/etc/passwd** file) by running the **Passwd** program with the **-d** option. In the following example, the user **fred** is deleted:

```
system[1] passwd -d
```

```
Name: fred

Name: RETURN
system[2]
```

Note that only a privileged user may delete a user.

Changing a Password and Prompt

When called without any options, the **Passwd** program allows the privileged user to change any user's password and any user to change his or her own password. When called with **-p** option, the **Passwd** program allows the privileged user to change any user's prompt and any user to change his or her own prompt string.

To change a password, call the **Passwd** program as follows:

```
system[1] passwd
Name: fred
Password: xxx

Name: RETURN
system[2]
```

Notice that the password encryption is displayed only after the password and a RETURN have been entered.

To change a prompt, call the **Passwd** program as follows:

```
system[1] passwd -p
Name: fred
Prompt string: xxx

Name: RETURN
system[2]
```

Changing User Characteristics

If the privileged user has occasion to change user characteristics other than the password or prompt, the user must be deleted and added again with the new characteristics.

3.85 The Patch Utility

utility:	**PATCH**
purpose:	This program patches a file.
user access:	all users
summary:	patch [filename]
arguments:	optional filename
options:	none

Description

The **Patch** utility displays and alters specified bytes within a file. Enter the command name plus a filename and press RETURN. When you see a greater-than sign (>), enter any of the available commands: "**d**" for display, "**e**" for exit, "**h**" for calculate, "**q**" for query, "**s**" for substitute, "**t**" for truncate, or "**z**" for zap. If **Patch** is called without a filename, only the "**h**" and "**e**" commands are valid.

Generally, if you give a command without arguments **Patch** uses the arguments from the previous command. Start addresses and swath values can be arbitrary expressions (refer to the "**h**" command).

Display

The "**d**" command displays part of a file. The possible forms are:

d start S swath	Display **swath** bytes from start address.
d start	Display the same number of bytes from start
d S swath	Display swath bytes from where last d command ended.
d	Display the same number of bytes from where last **d** command ended.

If no start address is given, the **d** command wraps to the beginning of the file when the end of the file is reached.

Exit

The "**e**" command (no arguments) incorporates the changes from the current **Patch** session and exits the program. To exit from **Patch** without saving the changes, press CNTRL-C.

Calculate

The "**h**" command serves as a calculator, and has the form:

h expression

The syntax of **expression** can be described in BNF form as follows ("::=" means "defined as" and "|" means "or"):

```
<expression> ::= <term> | +<term> | -<term> |
        <expression>+<term> | <expression>-<term>
<term> ::= <factor> | <term>*<factor> | <term>/<factor>
<factor> ::= <number> | (<expression>)
<number> ::= <hex string> | <dec string>.
<hex string> ::= <hex digit> | <hex string><hex digit>
<dec string> ::= <dec digit> | <dec string><dec digit>
<hex digit> ::= <dec digit> | a | b | c | d | e | f |
                                    A | B | C | D | E | F
<dec digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

The expression result is printed in decimal and hexadecimal form.

Query

The "**q**" command searches for a pattern of bytes. The possible forms are:

q start S swath pattern	Look for **pattern** through **swath** bytes from start address.
q start pattern	Look for **pattern** through the same number of bytes from **start**.
q S swath pattern	Look for **pattern** through **swath** bytes from where last **q** command ended.
q start S swath	Look for same pattern through **swath** bytes from **start**.
q start	Look for same pattern through the same number of bytes from **start**.
q S swath	Look for same pattern through **swath** bytes from where last **q** command ended.
q	Look for same pattern through the same number of bytes from where last **q** command ended.

If no start address is given, the **q** command wraps to the start of the file when the end of the file is reached.

The pattern, which is always a non-empty sequence of bytes, can be described in BNF form as follows:

```
<pattern> ::= <item> | <space><item> | <pattern><space><item>
<space> ::= SPACE | TAB | <space>SPACE | <space>TAB
```

```
<item> ::= <number> | <string>
<string> ::= '<simple string>'
<simple string> ::= <empty> | <single string><character>
<empty ::=
<character> ::= <ASCII character other than backslash> |
                    <escape character>
<escape character> ::= \n | \r | \t | \f | \b | \0 | \ddd | \x
```

The escape characters are:

\n	0A
\r	0D
\t	09
\f	0C
\b	08
\0	00
\ddd	ddd should be three octal digits
\x	if x is a non-escape character, it is taken verbatim.

If the **q** command finds the pattern, it displays 16 bytes of the file, starting with the pattern sought.

Substitute

The "**s**" command displays a file word by word and allows you to change selected bytes. The possible forms are:

s start	Display word at start address.
s	Display word from where the last s command ended.

The s command displays one word, and waits for one of the following responses:

.	Exit substitute command
RETURN	Display next byte
-	Display last byte
pattern	Replace this byte (and the following bytes, as needed) with the specified pattern. Pattern has the same form as in the **q** command.

Truncate

The "**t**" command truncates a file to the specified length, and has the form:

```
t size
```

The file is truncated (or extended) to the "**size**" number of bytes.

Zap

The "z" command overwrites the specified number of bytes with a given pattern. The available forms are:

z start S swath pattern	Fill **swath** bytes with **pattern** beginning at **start**.
z start pattern	Fill the same number of bytes with **pattern** beginning at **start**.
z S swath pattern	Fill **swath** bytes with **pattern** beginning where the last **z** command ended.
z start S swath	Fill **swath** bytes with zeros beginning at **start**.
z start	Fill the same number of bytes with zeros beginning at **start**.
z S swath	Fill **swath** bytes with zeros from where the last **z** command ended.

3.86 The Patchbug Utility

utility:	**PATCHBUG**
purpose:	This program corrects a bug in paslib.obj
user access:	all users
summary:	patchbug file-list
arguments:	file list
options:	none

Description

Paslib.obj version 02.45 and earlier contains a bug which prevents any program "crolinked" with this library from running successfuly with XMM enabled. The **Patchbug** utility can be used to enable such **.bin** (or **.obj**) files to work with the XMU.

Example:

```
system[1]  patchbug c.bin jcode.bin /usr/lib/paslib.obj
```

3.87 The Path Command

Shell command:	**PATH** or **PA**
purpose:	This command finds the full pathname of an executable file.
user access:	all users
summary:	path file-list
arguments:	filename
options:	none

Description

The **Path** utility identifies the command which would be executed if entered at the Shell prompt.

Path allows you to ensure you are running the correct version of a program.

3.88 The Priority Command

Shell command:	**PRIORITY** or **PRI**	
purpose:	This command changes the priority of a process.	
user access:	all users	(priorities 0 through +40)
	privileged user	(priorities +40 through -40)
summary:	pri [+priority-number][command-line]	
arguments:	priority number (optional)	
	command line (optional)	
options:	none	

Description

The **Priority** command establishes the priority of a process. **Priority** numbers range from -40 (highest) to +40 (lowest). The highest priority a nonprivileged user may specify is 0, the lowest is +40. A privileged user may specify any priority.

If the **Priority** command is executed without a priority number, the default value is +10. All processes run without using the **Priority** command are assigned a priority of 0.

If a command line is given as an argument, the priority specified applies to the process(es) initiated by the command line. If no argument is given, the priority applies to the current shell and all children of the current shell created after execution of the **Priority** command.

All ready processes are handled in a "round-robin" fashion. The higher the priority, the longer the process is allowed to run before the next process is executed. A -40 priority gets a "time-slice" of 576 milliseconds, a +40 priority gets 16 ms, and a 0 priority gets 96 ms (distribution varies exponentially).

3.89 The Priv Utility

utility:	**PRIV**
purpose:	This program allows any user the status of a privileged user.
user access:	all users
summary:	priv [command]
arguments:	optional command
options:	none

Description

The **Priv** utility examines the /etc/passwd file for a user named **system**. If this user is not found, an error message is displayed and execution of the utility is aborted.

If the user named **system** is found and there is a password associated with the user, the **Priv** utility prompts for the password. If the user responds with the correct password or if no password is associated with the user **system**, a new **Shell** is formed in which the user has the status of a privileged user.

If **Priv** was called without an argument, the new Shell is an interactive one. If there was a command supplied as an argument to the **Priv** command, the specified command will be executed by the privileged Shell.

Upon exiting from the new Shell, the user's previous status is reinstated.

3.90 The Prompt Command

Shell command:	**PROMPT**
purpose:	This command changes the prompt.
user access:	all users
summary:	prompt [prompt_string]
arguments:	any string
options:	none

Description

The **Prompt** command changes the system prompt. **Prompt_string** is the new string that the Shell is to use as a prompt. It must be a string of up to 23 characters. If no string is specified, the prompt is changed to the default prompt "**user_name[%d]**". The new prompt may be any string. For example:

jim[1] **prompt "Yes, dear,"**

Note the quotes were used to enforce the whole string to be considered a single argument. The prompt string may contain the "**%d**" substring. The "**%d**" characters will be replaced by the current command number.

The prompt command inspects the **/etc/who** file to find the "user_name" of the user on the current console. The effect is twofold:

- different users with the same user number will get a different prompt.

- execution of the **Priv** command will not change the prompt as the **Priv** command does not change the **/etc/who** file.

If there is no **/etc/who** file or if it does not contain the entry for the current terminal (this will be true if the user is rooted on a write protected floppy), the prompt will be "#" or "%", depending on whether or not the user is privileged.

3.91 The Pstat Command

Shell command:	**PSTAT** or **PS**
purpose:	This command displays the status of a process.
user access:	all users
summary:	pstat [-abl]
arguments:	none
options:	-a all -b brief display -l long display -e full display

Description

The **Pstat** command displays the following information on the status of a process:

PID	process identification number	
state	state of process:	Sleeping
		Ready
		Terminated
user id #		
group id #		
Ctty	controlling **tty**, the **tty** from which the process was started	
Priority	priority of the process	
Base	page number of the first memory allocated to the process	
Size	amount of memory allocated to the process	
Seconds	number of seconds the process has been executing	
command line	command line which invoked the process	

Options

The **-a** option lists the status of all processes. If the **-a** option is not selected, only those processes with the ID of the user giving the **Pstat** command are displayed.

The **-b** option displays a list of processes and their brief status.

The **-l** option displays a list of processes and their long status.

The **-e** option displays a list of processes and their full status.

3.92 The Query Program

utility:	**Q** or **QUERY**
purpose:	This program displays a short description of a specified utility program or Shell command.
user access:	all users
summary:	query [-s] [name]
arguments:	the names of one or more utility programs or Shell commands
options:	-s system function lists system call data as well as commands and utilities.

Description

The **Query** program searches a file containing one line descriptions of all of the utility programs and Shell commands for the name given as an argument.

When using **Query** without an argument, a listing of all one line descriptions of utilities and Shell commands is displayed.

The **Query** program considers names that are part of other keywords. When the name **fil** is given, **Query** finds all occurrences of the name **file** as well. This is helpful when the correct spelling of a name is unknown.

After using **Query** to find the name of the desired command, additional information is obtained by entering **help**, followed by the name of the command. For further details, refer to the **Help** utility.

The **Query** program uses the file **/usr/query/query_data** as a database. This file may be edited using the Screen Editor.

Options

The **-s** option searches the file **/usr/query/sys_data**, **/usr/query/jsys_data**, and **/usr/query/mode_data** before searching the default file, which gives information on the programs only.

The **/usr/query/sys_data** file gives a list of system calls associated with the command. The **/usr/query/jsys_data** and **/usr/query/mode_data** are linked to the files **/equ/jsysequ.z80** and **/equ/modeequ.z80**, respectively.

Example:

The following example demonstrates the use of the **Query** program.

```
jim[1] query delete

query_data
delete      -   removes a file or directory from a file system
deltree     -   deletes a directory and its descendents
passwd      -   change a user password, add or delete a user
```

In the above example, the **Query** program has displayed all descriptions of Shell commands and utility programs that contain the word **delete** in their descriptions.

3.93 The Ramdisk Utility

utility:	**RAMDISK**	
purpose:	This program creates, deletes, verifies, and calculates checksums on RAM disks.	
user access:	privileged user	
summary:	ramdisk [-c #] [-d] [-r] [-s] device-name(s)	
arguments:	RAM disk device names	
options:	-c #	create RAM disk of given size
	-d	delete RAM disk (return memory)
	-r	read RAM disk to show checksum errors
	-s	salvage RAM disk (recompute checksums)

Description

The **Ramdisk** utility should be called with one option and one or more RAM-disk device names as arguments. **Ramdisk** performs different actions, depending on the specified option.

Options

The -c option must be followed by the size of RAM disk (must be a multiple of four) to be created. The size is expressed in K (1024 bytes). The requested amount of memory will be allocated by the system and used as RAM disk. The contents of RAM disk will be all zeros. The **Mode** utility used on such a RAM disk will show a smaller number of available blocks, as one or more blocks, depending on size, will be used for checksums on individual blocks.

The -d option deallocates the RAM disk created by the -c option. The memory occupied by the RAM disk is returned to the system's memory pool.

The -r option reads all of the RAM disk. Any checksum errors are reported on the raw console.

The -s option recomputes all checksums. After running **Ramdisk** with the -s option, the RAM disk driver will report no checksum error. Because checksum errors should not be ignored, use the -s option with discretion.

Notes

The **Mode** utility, when applied to a RAM-disk device, shows various characteristics, including the size, in blocks, and the verify flag. Although **Mode** can be used to turn off the verify flag, resulting in increased speed, this is not recommended. Turning the verify flag off defeats a valuable checking mechanism. While the flag is off, write operations do not compute a checksum. Thus, when the flag is turned on again, the RAM disk driver may indicate numerous errors. (The checksums are no longer up

to date.)

When using **Makfs** to create a file system on a RAM disk, **makfs -b 2** forces the beginning of the inode area to the block following the Superblock. This is acceptable because there is no need for a boot area on a RAM disk.

Example

The following example is a typical command file that creates a RAM disk, loads it with some often-used programs, and mounts it to **/ram**.

```
ramdisk -c 128 rd0          % Create 128K RAM disk
makfs -b 2 rd0              % Create file structure without
                            % boot area
mount rd0 /ram              % Mount RAM disk
dir /bin                    % Copy useful files from /bin
copy copy.bin version.bin mode.bin ls.bin /ram
dir /cmd                    % Copy useful commands from /cmd
copy bak.cmd /ram
```

Execution of a similar command file might be specified in the file **startup.cmd.**

3.94 The Rcopy Utility

utility:	**RCOPY**
purpose:	This utility copies a file or block device
user access:	privileged user
summary:	rcopy [options] arguments
arguments:	two file pathnames **or** one or two block device pathnames **or** a file pathname and a block device pathname

options:

-b #	first block on input device
-e #	last block on input device
-d #	first block on output device
-s #	block count (swath)
-l #	I/O buffer size
-c	compare mode
-f	forced write mode
-p	physical mode (Cflop only)
-r	read-only mode
-t	terse mode
-v	verify mode

Description

Rcopy (for privileged users) copies a file or block device to another file or block device. **Rcopy** is primarily intended for copying a hard disk onto cartridge tape or another hard disk. Files saved with **Ftback** (68000 Cromix version 20.65) cannot be restored with **Rcopy**. **Rcopy** can be used with version 20.08 or higher of the Tar utility. A "-" in place of a file or device name indicates the standard input or standard output.

Options

-b The number following **-b** is the first block of the input device to be read; without this option, reading begins at block 0. This option is useful for copying specific data, such as a hard disk label (block 0) or alternate track table (block 19).

-e The number following **-e** is the last block of the input device to be read; without this option, reading stops at the last block (unless the end of the output device is reached first).

-d The number following **-d** is the first block of the output device to be written; without this option, writing begins at block 0.

-s The number following **-s** is the total number of blocks to be read.

-l The number following **-l** is the length of the I/O buffer in units. If the CTD is used for input or output, a unit is 17K; otherwise a unit is 1K. Without this option, the buffer is 10 units (170K for the CTD and 10K for all other files/devices). A buffer of 510K (**-l 30**) yields an optimum speed of 2 Mbytes/minute when copying between a CTD and a hard disk. This option is ignored when copying between uniform (UNIX) floppy disks.

-c After copying to or from a cartridge tape, repeat the **Rcopy** command with the **-c** option to verify that the input and output are identical. For all other files or devices, use the **-v** option to copy and verify on the same pass.

-f Overwrites the existing output file (use only when output is to a file).

-p Reduces copying time when both input and output devices are Cromix-Plus floppy disks (large or small, single- or double-sided, single- or double-density). In this case the buffer size equals the cylinder size, regardless of the **-l** setting.

-r Reads the input file or device for errors, but creates no output (only one argument is required).

-t Provides a terse progress report as copying proceeds.

-v Compares the copied output with the input and sends an error message if they are not the same. Do not use the **-v** option for cartridge tapes; verification of cartridge tapes requires a second execution of **Rcopy** with the **-c** option.

Examples

When copying to an entire hard disk (std31 or std63), be sure not to overwrite the disk label (block 0) or alternate track table (block 19). To save blocks 0 and 19 on floppy disk files **label** and **alttable**, enter (for an 8" diskette mounted in drive A as **/a**):

```
system[1] rcopy -t -s 1 /dev/std0 /a/label
system[2] rcopy -t -b 19 -s 1 /dev/std0 /a/alttable
```

To restore these blocks in the event of a hard disk crash, enter:

```
system[1] rcopy -t -s 1 /a/label /dev/std0
system[2] rcopy -t -d 19 -s 1 /a/alttable /dev/std0
```

If the CTD is the input or output device, copying and verifying requires separate **Rcopy** commands. To copy partition std3 to floppy tape, and then verify the copy, enter:

```
system[1] rcopy -t /dev/std3 /dev/ftcd
system[2] rcopy -tc /dev/std3 /dev/ftcd
```

Since the tape is rewound after each **Rcopy** command, it is correctly positioned for the second pass.

To create and verify a **Tar** archive of directory **/usr/data** on a tape cartridge, enter:

```
system[1] tar -cvf - /usr/data | rcopy - /dev/ftcd
system[2] tar -cvf - /usr/data | rcopy -c - /dev/ftcd
```

Tar writes directory **/usr/data** to the standard output (the dash after the **f** option). **Rcopy** takes the data from standard input (the dash for the input file), and writes it to the tape. In this case, **Tar** issues the progress reports (**v** option selected).

To restore a specific file or directory from a tape created with the **Tar** utility, enter:

```
system[1] rcopy /dev/ftcd - | tar -xvf - filename
```

To list the contents of a tape created with the **Tar** utility, enter:

```
system[1] rcopy /dev/ftcd - | tar -tvf -
```

To dump the contents of a tape created by **Rcopy** alone, enter:

```
system[1] rcopy /dev/ftcd - | dump
```

3.95 The Readall Utility

utility:	**READALL**
purpose:	This program reads a device to check for errors.
user access:	privileged user
summary:	readall [-at] [-c #[,#]] [-h #[,#]] [-l #] devicename
arguments:	device name
options:	-a do not read declared bad tracks -t do not write progress report -c cylinder numbers to read -h surfaces (heads) to read -l number of passes

Description

Readall checks for bad tracks on an STDC hard disk, ESDC hard disk, or any other block device that supports the "primitive read" **setmode**. The device name must be in the **/dev** directory, and the device must already be initialized. For an STDC or ESDC drive, use the device name for the entire disk (**std31, esd31** for drive **0**; **std63, esd63** for drive **1**).

Error messages (reports of bad tracks) are sent to the standard error device; status messages are sent to the standard output device. Numbers for the **-c** and **-h** options are assumed to be decimal unless followed immediately by an "**h**" for hexadecimal.

Options

The **-a** option skips reading of tracks declared bad in the alternate track table (valid only for hard disk drives).

The **-t** option suppresses progress messages and displays only error messages (useful when redirecting output). This option should be used for floppy tape devices to allow reading in streaming mode.

The **-c** option, followed by a number, selects a single cylinder to be read; if **-c** is followed by two numbers separated by a comma, a range of cylinders is read (from the first number through the second). Without this option, all cylinders are read.

The **-h** option, followed by a number, selects a single surface (head) to be read; if **-h** is followed by two numbers separated by a comma, a range of surfaces is read (from the first number through the second). Without this option, all surfaces are read.

The **-l** option selects the number of passes (each track is read once on each pass). Without this option, only one pass is made.

Examples:

To make one pass through cylinders 5 thru 16 (10h) on device **/dev/std31**, enter:

system[1] **readall -c 5,10h std31**

To make 5 passes through the entire device **/dev/std31** (except for tracks declared bad in the alternate track table), enter:

system[1] **readall -a -l 5 std31**

To send error messages to both a file and the terminal, use a command such as:

system[1] **(readall -a -l 5 std31 > /dev/tty) |* tee error_file**

3.96 The Rehash Command

Shell command:	**REHASH**
purpose:	This command builds (or discards) a table of executeble files.
user access:	all users
summary:	rehash [any-string]
arguments:	optional arbitrary string
options:	none

Description

To execute a command the **Shell** utility must locate the file to be executed. This means checking all the directories defined in the "**path**" Shell variable, in general three times, once for each possible executable extension (**.cmd**, **.bin**, or **.com**). If the affected directories are very extensive this might take a long time.

The **Rehash** function of the Shell utility, when invoked without an argument, can be used to contruct a full list of executable files. If such a list is built, a command can be found much faster. The difference is especially noticable in the case of large directories and a busy system. If the **Rehash** function is invoked with any argument, the list of executable files will be scrapped.

The list of executable files will be used provided all of the following is true:

> The command was entered without the directory information in front of it and without an extension.
>
> The **Rehash** function was used to build the list of executable files and the list was not subsequently scrapped.

In such a case, the **Shell** utility will try to execute the file without any searching. If the list of executable files does not contain the named command the message "Command not found" will appear.

When a Shell is started there is no list of executable files. The **Rehash** function must be invoked to build it first.

The list of executable files will become invalid in the following cases:

> The current directory was changed using the "**Directory**" command.
>
> An executable file (file with the exension **.cmd**, **.bin**, or **.com**) was created or deleted.
>
> The search path stored in the "**path**" and "**ext**" Shell variables

was changed.

If the list of execuble files becomes invalid the following may result:

A valid command will result in the message "Command not found".

A deleted command is still in the list of executable files
resulting in the message "File not found".

The wrong file will execute (for example, **.cmd** instead of **.bin**,
or a file from the wrong directory will execute).

In all such cases, the **Path** function will correctly identify the command Shell will actually execute. Any problems may be resolved by any of the following means:

Rebuild the list of executable files ("**Rehash**" command).

Scrap the list of executable files ("**Rehash x**" command).

Identify the file to be executed more precisely by specifying
either the directory and/or the extension.

Hint

Do not add your own files into the **/bin** and **/cmd** directories. These directories, especially **/bin**, are big enough already. Store your executable files into the **/usr/bin** directory. This strategy has an added benefit of providing a simpler back up procedure.

Note:

If you have moved a large number of files from the **/bin** or from the **/cmd** directory to the **/usr/bin** directory, the original directories may end up with many empty slots will still be large. You can verify this state of affairs by simply executing the command

jim[1] **dump /bin**

Every other line must show a command name. If there are many empty slots you should consolidate the directory. You can do this by executing the the following sequence of commands (as a privileged user):

jim[1] **d /**
jim[2] **makd newbin**
jim[3] **chowner bin newbin**
jim[4] **cptree bin newbin**
jim[5] **ren bin oldbin**
jim[6] **ren newbin bin**
jim[7] **deltree -a oldbin**

Note that the system is very vulnerable between both **Rename** commands: for a short period of time there is no **/bin** directory.

3.97 The Rename Command

Shell command:	**RENAME** or **REN**
purpose:	This command changes the name and/or directory of a file.
user access:	all users
summary:	ren oldfile1 newfile1 [... oldfileN newfileN]
arguments:	one or more pairs of file pathnames (existing pathname first, followed by new pathname)
options:	none

Description

The **Rename** command changes a filename and/or the directory where it is located.

This command does not move a file from one device to another.

3.98 The Repeat Command

Shell
command: **REPEAT** or **REP**
purpose: This command repeats a command.

user access: all users

summary: rep count command

arguments: a count of the number of repetitions

command

options: none

Description

The **Repeat** command is used to repeat a command a specified number of times.

Example:

```
jim[1] repeat 5 echo "this line is displayed five times"
this line is displayed five times
this line is displayed five times
this line is displayed five times
this line is displayed five times
this line is displayed five times
jim[2]
```

Notes

The **Repeat** command may be terminated by a semicolon and in this case any command(s) following a semicolon are executed only once. This means that the following command displays the current directory three times and then displays the time once:

```
jim[1] repeat 3 dir; time

/usr/jim
/usr/jim
/usr/jim
        Wednesday, July 16, 1986        12:23:42
```

Multiple commands may be enclosed in parenthesis to cause the sequence of commands to be repeated. The following command would display both the current directory and time 3 times:

```
jim[1] repeat 3 (dir; time)

/usr/jim
        Wednesday, July 16, 1986        12:23:42
/usr/jim
        Wednesday, July 16, 1986        12:23:42
/usr/jim
        Wednesday, July 16, 1986        12:23:43
```

3.99 The Rewind Command

Shell command:	**REWIND** or **REW**
purpose:	This command restores the arguments used to call a command file.
user access:	all users
summary:	rew
arguments:	none
options:	none

Description

The **Rewind** command restores the arguments used to call a command file. It nullifies the effect of any **Shift** commands given within the command file. After execution of the **Rewind** command, **#1** represents the first argument of the original command file, **#2** the second, and so on.

3.100 The Rfile Utility

utility:	**RFILE**
purpose:	This program allows binary files to be received from users over the phone lines with error-free results.
user access:	all users
summary:	rfile [-q] [-f] [-d device-name] [-b baud-rate] destination-directory
arguments:	destination directory pathname (must already exist)
options:	-q quiet (default is verbose) -f force -d qtty device name (default is STDIN/STDOUT) -b baud rate (default is current baud rate)

Description

The **Rfile** utility allows binary disk files to be received by a user on one Cromix system from a user on another Cromix system that is using the **Sfile** utility to transmit files.

Rfile may operate at either 300 or 1200 baud. It must use an asynchronous modem such as the Cromemco MDM-1200, a Bell 212A, or a Bell 103 type. The modem used must be compatible with the modem the **Sfile** utility is using to transmit the data. The modem can be connected to any serial port on the Quadart using a 12-wire cable constructed for this purpose.

The **qtty** driver is used to connect **Rfile** with the IOP/Quadart and modem. The Cromix-Plus system being used must include the IOP/Quadart drivers (refer to **Crogen**), and a device file for the **qtty** should be set up in the **/dev** directory using **Makdev**. The corresponding entry in the **/etc/ttys** file should have a 0 in the first column.

Example

```
jim[1] rfile -d qtty2 -b 300 recvtemp
```

This command line sets the reception rate at 300 baud and stores all of the data received from **qtty2** into the existing directory named **recvtemp**.

Messages returned by Rfile

Waiting for phone call

Rfile is in an idle state waiting for a connection from the **Sfile** utility. **Rfile** will remain in this mode indefinitely if **Sfile** fails to make a valid connection.

Receiving [filename] from [devname] into [directory-name]

A valid connection to the **Sfile** utility has been established, and data is being transferred from the **qtty** device to the specified disk file in the specified directory.

Connection lost -- ABORT RFILE

The line to **Sfile** was prematurely disconnected.

ABORT RFILE

A CONTROL-C character was received from the user at the keyboard; **Rfile** does an orderly exit back to the Cromix Shell.

Options

The **-q** option specifies a different set of **Rfile** console messages. (Used by **Ccall** and when **Rfile** is running on a remote machine.)

The **-f** option causes **Rfile** to overwrite any existing file with the same pathname as the file sent by **Sfile**. If this option is not specified and another file exists with the destination pathname, an error is reported.

The **-d** option specifies which **qtty** device is to be the receiver.

The **-b** option sets the baud rate of the receiving device.

Notes

Rfile is used in conjunction with **Sfile**. Refer to the Sfile utility for additional information.

When used without an argument, **Rfile** displays a summary of the command line syntax.

3.101 The Root Utility

utility:	**ROOT**
purpose:	This program displays the name of the device containing the root directory.
user access:	all users
summary:	root
arguments:	none
options:	none

Description

The **Root** program displays the root directory's device pathname.

Example:

```
system[1] root

/dev/std0
```

3.102 The Sc Utility

utility:	SC
purpose:	This program allows the user to edit files.
user access:	all users
summary:	sc filename [filename]
arguments:	name(s) of file to be edited

Description

When called without an option or with the -c option, the **Screen** program (**/bin/screen.bin**) creates output files with a CR-LF (carriage RETURN-linefeed) pair at the end of lines. When called with the **-n** option, only a LF character is used as a line terminator. Such files can then be used with the UNIX Operating System.

The Sc program (**/cmd/sc.cmd**) calls the **Screen** editor with the **-n** option.

Note

If you prefer to alter the **Screen** program so **-n** instead of **-c** is the default, patch the file **screen.bin** with a zero at location 0FAH. This patched version will be loaded slightly faster than **sc.cmd**. You may wish to call the patched version **sc.bin**, in which case, the file **sc.cmd** may be discarded.

3.103 The Scan Utility

utility:	SCAN
purpose:	This program scans a directory tree
user access:	all users
summary:	scan pathname ['expression']
arguments:	directory pathname optional expression in single quotes
options:	none

Description

The **Scan** utility is a modernized version of the **Find** utility. It scans all the files in the specified directory structure for **expression**. The **expression** may be given from the command line or from STDIN (the standard input). If the **expression** is given from the command line, it should be enclosed in single quotation marks. If the **expression** is given from STDIN, quotation marks are not required. **Scan** reads the expression until CONTROL-Z (end-of-file) is typed from the terminal.

The **expression** is similar to an expression in the C programming language.

Types

Each value is either an integer value or a string value.

Constants

Integer constants can be decimal, hexadecimal, or octal, as in C. String constants are (doubly) quoted strings. In addition, there are a few predefined constants:

is_ordin	type int	value 0
is_direct	type int	value 1
is_char	type int	value 2
is_block	type int	value 3
is_pipe	type int	value 4
ac_read	type int	value 0x01
ac_exec	type int	value 0x02
ac_writ	type int	value 0x04
ac_apnd	type int	value 0x08
days	type int	value 60*60*24

Variables

The following variables are defined at the beginning of the program. There is no assignment, so they cannot be changed by the user.

getuser	type int	current user number
getgroup	type int	current group number
now	type int	current time (number of seconds from 00:00:00 on March 1, 1960)

The following variables are defined after each file is scanned. They provide information from the inode. Again, there is no assignment, so they cannot be changed by the user.

path	string	Full pathname
dir	string	Directory pathname
name	string	File name without directory and extension
ext	string	Extension Example: If path = "/usr/user1/test.c" then dir = "/usr/user1/" name = "test" ext = ".c"
owner	int	File owner
group	int	File group
type	int	File type
aowner	int	Owner access
agroup	int	Group access
aother	int	Other access
nlinks	int	Number of links
size	int	File size
inode	int	Inode number
parent	int	Parent inode
dcount	int	Directory count
usage	int	Usage (blocks)
tcreate	int	Time created (see now)
tmodify	int	Time modified (see now)
taccess	int	Time accessed (see now)
tdumped	int	Time dumped (see now)

Functions

There are a number of predefined functions. These functions yield a value of predefined type, and the arguments are of predefined type.

int shell(s) string s;	Execute command s. Return exit value from the command.
int ok	Get a single character from **/dev/tty**. If it is "Y" or "y", return 1, else zero.
int strequ(s,t) string s, t;	Compare string s to string t. Return 1 if they match, else zero. (The string t can contain ambiguous shell characters, such as "*" or "?".
int print(s) string s;	Print string s to a line by itself. Always return 1.
int printn(s) string s;	Print string s without newline character. Always return 1.
string username(user) int user;	Return string corresponding to given user number, empty string if not found.
string groupname(group) int group;	Return string corresponding to given group number, empty string if not found.
int usernum(name) string name;	Return user number corresponding to given name, -1 if not found.
int groupnum(name) string name;	Return group number corresponding to given name, -1 if not found.

Operators

Operators are listed in order of decreasing priority. In general, operators can be used only on a pair of integers unless explicitly allowed on a pair of strings. Their meaning is the same as in C.

Operator	Associativity	Notes
( )	left to right	
! ~ -	right to left	Integers only.
* / %	left to right	Integers only.
+ -	left to right	Integers only.
<< >>	left to right	Also on strings. Shifting a string means

		shifting out.
< <= > >=	left to right	Also on strings. On strings, a lexicographic comparison.
== !=	left to right	Also on strings. On strings, a comparison for strict equality or inequality.
&	left to right	Integers only.
^	left to right	Integers only.
|	left to right	Also on strings. The resulting string is the concatenation of the left and right argument, (in this order).
&&	left to right	Integers only.
||	left to right	Integers only.
? :	left to right	Integers and strings.
,	left to right	Integers and strings.

Examples

For brevity, each sample expression is written as though entered from STDIN, without quotes.

Print all filenames that include **ted**:

```
name == "ted" && print(path)
```

Print all filenames that include **ted** or **mary**:

```
(name == "ted" || name == "mary") && print(path)
```

List -e all directories:

```
type == is_direct && shell("ls -e "|path)
```

List -l all files with extension .c:

```
ext == ".c" && shell("ls -l "|path)
```

Create forced links of all **jsysequ.asm** files that are not already linked to the correct file. If files do not compare, just print the differences:

```
(namelext) == "jsysequ.asm" && inode != 234 &&
shell("cmpasc -rt /equ/jsysequ.asm "|path) == 0 &&
shell("maklink -vf /equ/jsysequ.asm "|path)
```

Delete, after verification, all files that have not been accessed for more than a year:

```
type == is_ordin && taccess < now - 365*days &&
printn("Delete "|path|"?  ") && ok && shell("del "|path)
```

Delete all files that belong to the user **temp**:

```
owner == usernum("temp") && shell("delete -v "|path)
```

or

```
username(owner) == "temp" && shell("dele -v "|path)
```

Compare all files in directory **foo** to files in the directory **bar**:

```
type == is_ordin && shell("compare -t foo/"|namelext|" bar/"|namelext)
```

To create a command file where directories are given as arguments, replace **foo** with **#1** and **bar** with **#2**.

3.104 The Screen Utility

utility:	SCREEN
purpose:	This program is used to edit files.
user access:	all users
summary:	screen [-cn] filename [filename]
arguments:	name of file to be edited
options:	-c CR-LF pair used as line terminator -n LF only used as line terminator

Description

The **Screen** utility program enables the user to edit files.

When called without an option or with the **-c** option, the **Screen** program (**/bin/screen.bin**) creates output files with a CR-LF (carriage RETURN-linefeed) pair at the end of lines. When called with the **-n** option, only a LF character is used as a line terminator. Such files can then be used with the UNIX Operating System.

Cromix-Plus utilities understand lines terminated by either the LF character or CR-LF pair. A few utilities, however, were originally written for the CDOS Operating System. These programs (notably, those using older versions of Z80 assembler) might require the CR-LF pair as the line terminator.

The Sc program (**/cmd/sc.cmd**) can be used to automatically call **Screen** with the **-n** option.

If you prefer to alter the **Screen** program so **-n** instead of **-c** is the default, patch the file **screen.bin** with a zero at location 0FAH. This patched version will be loaded slightly faster than **sc.cmd**. You may wish to call the patched version **sc.bin**, in which case, the file **sc.cmd** may be discarded.

3.105 The Set Command

Shell command:	**SET**
purpose:	This command sets and displays Shell variables
user access:	all users
summary:	set [variable [value]]
arguments:	optional variable name optional variable value
options:	none

Description

The **Set** command, when given without arguments, displays the names and values of all variables known to the Shell. The **Set** command given with a single argument removes the specified name from the list of variables. The **Set** command called with two arguments defines (or redefines) the value of the Shell variable identified by the first argument to the value of the second argument.

The names of Shell variables are arbitrary sequences of upper or lower case letters and digits, starting with a letter. The underscore character is considered to be a letter. All letters in Shell variable names are automatically converted to lowercase.

Shell variables are referenced by:

```
#name
```

or

```
#{name}
```

The second form is always correct. It must be used if the name is followed immediately by another alphanumeric character to prevent the remaining characters from being treated as part of the name.

Each Shell starts with an empty list of Shell variables. The files:

```
/etc/sh_env
../sh_env
```

if they exist, are then read in this order. If the Shell is invoked using the **-e** option and its filename argument, then that file is also read.

The Shell environment files are used to define (or redefine) Shell variables. Empty lines, lines consisting of white space only, and lines starting with the '**%**' character, are ignored. Each of the remaining lines contains a definition of a Shell variable. The definition consists of the variable name

and optional string. The variable name and the string must be separated by white space. The string terminates with white space. If necessary, the string may be enclosed in quotes ("). In such a case the string may contain white space.

See the file **/etc/sh_env** for an example.

Examples.

```
jim[1] set abra cadabra                    % Define abra
jim[2] set hokus pokus                     % Define hokus
jim[3] set foo bar                         % Define foo
jim[4] echo #abra                          % #abra is replaced
cadabra
jim[5] echo #hokus #pokus #foo #bar        % Try something weird
pokus #pokus bar #bar                      % Pokus and bar are not
                                           % variables
jim[6] set                                 % List all variables
foo = "bar"                                % Here they are:
hokus = "pokus"
abra = "cadabra"
jim[7] set abra                            % Delete abra
jim[8] set hokus                           % Delete hokus
jim[9] set                                 % See what is left
foo = "bar"                                % As expected
jim[10] set foo ""                         % Define foo as an empty
                                           % string
jim[11] set                                % List all strings
foo = ""
jim[12] set foo                            % Discard foo
jim[13] set                                % See that nothing is
                                           % left
jim[14] set x ../one/two/very_long_name    % Let x be a directory
                                           % path
jim[15] ls #x                              % List the directory
File not found: "../one/two/very_long_name" % Nonexistent
jim[16] echo #xy                           % There is no variable xy
#xy
jim[17] echo #{x}y                         % Use variable x
../one/two/very_long_namey                 % Note y at the end
jim[18]
```

Note:

Shell variables are not passed from one Shell to another. When a command file is started it is run as a shell of its own. This means that each command file starts with an empty set of variables.

3.106 The Setpri Utility

utility:	**SETPRI**	
purpose:	This command changes the priority of a process.	
user access:	all users	(priorities 0 through +40)
	privileged user	(priorities +40 through -40)
summary:	setpri process-id priority	
arguments:	process id number (PID) priority number	
options:	none	

Description

The **Setpri** utility assigns a priority number to a given process. Priority numbers range from **-40** (highest priority) to **+40** (lowest priority). A non-privileged user can assign only non-negative priorities to his own processes; a privileged user can assign any priority to any process.

Example

To assign a priority of 20 to the process whose PID number is 456, enter:

```
jim[1] setpri 456 20
```

3.107 The Sfile Utility

utility:	**SFILE**
purpose:	This program allows binary files to be sent between users over the phone lines with error free results.
user access:	all users
summary:	sfile [-q] [-d device-name] [-b baud-rate] file-list
arguments:	one or more filenames
options:	-q quiet (default is verbose) -d qtty device name (default is STDOUT) -b baud rate (default is current baud rate)

Description

The **Sfile** utility allows binary disk files to be transmitted from a user on one Cromix system to a user on another Cromix system using the **Rfile** utility to receive files.

Sfile operates at either 300 or 1200 baud. It must use an asynchronous modem such as the Cromemco MDM-1200, a Bell 212A, or a Bell 103 type. The modem used must be compatible with the modem the **Rfile** utility is using to receive the data. The modem can be connected to any serial port on the Quadart using a 12-wire cable constructed for this purpose.

The **qtty** driver is used to connect **Sfile** with the IOP/Quadart and modem. The Cromix-Plus system being used must include the IOP/Quadart drivers (refer to Crogen), and a device file for the **qtty** should be set up in the **/dev** directory using **Makdev**. The corresponding entry in the **/etc/ttys** file should have a 0 in the first column.

Messages returned by Sfile

Now waiting for call to complete ...

Sfile is waiting for a connection to be made with another modem over the phone lines. This call can either be dialed manually using Bell equipment, or the MDM-1200 can dial the number and establish the connection automatically.

No answer -- Call aborted

If a connection is not established within 60 seconds, **Sfile** exits to the Cromix Shell.

Transmitting [filename] to [devname]

When a valid connection is established with the **Rfile** utility at the other end, the specified file is transmitted through the specified **qtty** device.

Rfile not responding -- Sfile aborted

If a connection is established with another modem, **Sfile** determines if the **Rfile** utility is ready at that end. If **Rfile** is not running at the other end, or if **Rfile** is running at an incompatible baud rate, **Sfile** disconnects the line and exits to the Cromix Shell.

Connection lost -- ABORT SFILE

The line was disconnected in the middle of a transmission; **Sfile** exits to the Cromix Shell.

ABORT SFILE

A CONTROL-C character was received from the user at the keyboard; **Sfile** does an orderly exit to the Cromix Shell.

Options

The **-q** option specifies a different set of Sfile console messages. (Used by **CCall** and when **Sfile** is running on a remote system.)

The **-d** option specifies which **qtty** device is to be the transmitter.

The **-b** option sets the baud rate of the transmitting device.

Notes

Sfile is used in conjunction with **Rfile**. Refer to the **Rfile** utility for additional information. When called without an argument, **Sfile** displays a summary of the command-line syntax.

3.108 The Shell Command

Shell command:	**SHELL** or **SH**	
purpose:	This command creates a Shell process.	
user access:	all users	
summary:	[shell] [cmd file]	
arguments:	optional command file	
option:	-c	complete input line
	-p	parsed input line
	-q	quiet
	-z	do not terminate on CONTROL-Z
	-h string	prompt string
	-e pathname	Shell environment file

Description

Given without an argument, the **Shell** command creates an interactive shell process; given with the name of a command file, **Shell** executes that file as it echoes each line of the file to the terminal. Entering only the name of a command file implies the use of the **Shell** command. Executed in this manner the source lines of the command file are not echoed to the terminal.

On Cromix-Plus release 94 and higher, the **Shell** recognizes the variable:

#err

as the decimal value of the last error number.

Cromix-Plus release 149 and higher can manipulate **Shell** variables. If a command procedure sets the variable "**abort**" to an arbitrary value, the command procedure will abort on the first error. Execute the "**help set**" command for details.

On Cromix-Plus release 153 and higher, the Shell recognizes the variables:

#path and #ext

The variable **#path** should contain a list of directories, separated by colons. The Shell will search each directory in turn for the files with the extensions stored in the variable **#ext**. If the variable **#path** is not defined, the Shell will supply its own list of directories which is equivalent to the definition

path = ":/ram:/bin:/cmd"

with the meaning that the directories will be searched in the following order:

Rfile not responding -- Sfile aborted

If a connection is established with another modem, **Sfile** determines if the **Rfile** utility is ready at that end. If **Rfile** is not running at the other end, or if **Rfile** is running at an incompatible baud rate, **Sfile** disconnects the line and exits to the Cromix Shell.

Connection lost -- ABORT SFILE

The line was disconnected in the middle of a transmission; **Sfile** exits to the Cromix Shell.

ABORT SFILE

A CONTROL-C character was received from the user at the keyboard; **Sfile** does an orderly exit to the Cromix Shell.

Options

The **-q** option specifies a different set of Sfile console messages. (Used by **CCall** and when **Sfile** is running on a remote system.)

The **-d** option specifies which **qtty** device is to be the transmitter.

The **-b** option sets the baud rate of the transmitting device.

Notes

Sfile is used in conjunction with **Rfile**. Refer to the **Rfile** utility for additional information. When called without an argument, **Sfile** displays a summary of the command-line syntax.

The **-c** option indicates that a full command line is being passed to the Shell (the line is not parsed into arguments).

The **-p** option indicates that the command being passed to the Shell has been parsed.

The **-q** option requests that lines from a command file not be echoed to the terminal (STDOUT).

The **-z** option should be used to start an interactive Shell that is not aborted by CNTRL-Z (End-of-File) entered from the terminal. The Shells started by **Gtty** always have this option selected.

The **-e** option can be used to add a third shell environment file. Shell always reads the files "**/etc/sh_env**" and "**../sh_env**", in this order. The file defined by the **-e** option is the third evironment file read. Note the subsequent shell environment files are superimposed over existing Shell variables.

Examples:

As the value of **#err** might be a negative number, the "--" sign notifies the **Echo** utility that there are no more options.

```
bad_command
if #err != -1 echo -- #err error number
```

A command file:

```
command1
command2
set abort 1
command3
command4
set abort
command5
command6
```

is equivalent to the command file:

```
command1
command2
command3; if -err exit #err
command4; if -err exit #err
command5
command6
```

```
current     directory
/ram        directory
/bin        directory
/cmd        directory
```

Within each directory the executable files will be searched according to the extension list stored in the **#ext** variable. The extension list consists of individual extensions separated by a colon. The Shell will ignore illegal extensions so that the only reason to change the **#ext** variable is to alter the order in which the different extensions are searched. If the **#ext** variable is not defined, the Shell will use its own list which is equivalent to the definition:

ext = "bin:com:cmd"

Use the "**help set**" command for further information.

Command files can use redirection from the body of the command file. If the command is followed by the sequence:

<< label

the lines following the command up to the line starting with:

%label

will be submitted to the command as its standard input. This means that the lines:

```
ty << eof
Line 1
Line 2
%eof
```

are equivalent the the command:

ty < temp_file

where the file **temp_file** contains the text:

```
Line 1
Line 2
```

In addition, if a line submitted for input has the "/" (backslash) character as the last character, this backslash character and the following (optional RETURN and) NEWLINE character are discarded.

Options

Most options are needed only when a program is calling a Shell; the **-c** and **-p** options have no meaning when the Shell is called from the terminal.

3.109 The Shift Command

Shell command:	**SHIFT**
purpose:	This command shifts the arguments in a command file.
user access:	all users
summary:	shift
arguments:	none
options:	none

Description

The **Shift** command is used to shift the arguments in a command file. After execution of the **Shift** command, **#1** represents the second argument from the original command line, **#2** represents the third, and so on. After another execution of the **Shift** command, **#1** represents the third argument, etc.

The **Rewind** command nullifies the effects of the **Shift** command.

For examples of the **Shift** and **Rewind** commands, refer to the sample command file under the discussion of the **Exit** command.

3.110 The Shutdown Utility

utility:	**SHUTDOWN**
purpose:	This program shuts down the system.
user access:	privileged user
summary:	shutdown
arguments:	none
options:	none

Description

The **Shutdown** program (**/cmd/shutdown.cmd**) contains commands to shut down the operating system by killing all processes, flushing buffers, and logging off all users. It first warns users and provides a five-second countdown.

Shutdown also has a facility that works with the **Startup** command to detect inadvertent system terminations.

Run the **Shutdown** program whenever system operation is to be terminated.

3.111 The Sim Utility

utility:	**SIM**
purpose:	This utility allows CDOS programs to run under the Cromix-Plus Operating System.
user access:	all users
summary:	(sim) progname arg0,arg1,...,argn
arguments:	program name and arguments to the program to be run
options:	none

Description

The **Sim** program allows CDOS programs to run under the Cromix-Plus Operating System. The CDOS simulator is automatically loaded when a file with the extension **.com** is executed.

Notes

The **Cdoscopy** utility program is the only way to read files from or write files to CDOS format disks from the Cromix-Plus Operating System.

Drive/File Access From CDOS Programs

For CDOS programs to gain access to files on various drives, the CDOS Simulator converts disk specifiers to directory names. For example:

B:Filename becomes **/B/Filename**

If no disk specifier or the disk specifier A is used (as in **A:Filename**), the file is assumed to be in the current directory.

To take full advantage of this scheme, Cromemco recommends a file structure be constructed as follows:

1. Create files B, C, D, etc. in the root directory. Each file corresponds to one of the disk drives in the system.

2. Mount each disk on the appropriate drive using the Mount utility:

system[1] **mount fdb /b**

Note that these **must be** Cromix format disks.

3. The files on those disks may be read and written from CDOS programs. The CDOS simulator, running under the Cromix-Plus Operating System, automatically converts the CDOS drive specifiers to the appropriate directory names.

4. Each disk mounted must be unmounted before it is physically removed from the system. To do this, use the Unmount utility:

system[1] **unmount fdb**

Disks created in this manner are in the Cromix Operating System format and not CDOS compatible.

The permanent mapping of CDOS directories to Cromix directories is defined by a table in **sim.bin**. This table contains a 16-byte entry for each CDOS directory. The entry is the pathname of the Cromix directory to which the CDOS directory maps. Using **Dump**, you can display the contents of the table. The entry for CDOS directory **A** starts at location 509H, the entry for CDOS directory **B**, at location 519H, and so on.

To change the mapping of CDOS directories, the **Patch** utility can be used to change the contents of the table. When changing an entry:

1. Each pathname must end with a slash mark ("/").

2. Each string must be terminated with a null character.

3. The pathname, including the final "/" cannot be longer than 15 characters.

"A" refers to the current directory, and should not be changed.

Users may easily define the mapping of CDOS directories to Cromix directories when calling **Sim** explicitly.

Example:

jim[1] **sim -c /usr/lib program.com arguments**

refers all references to the CDOS C directory to the Cromix directory **/usr/lib**. Thus, one or more CDOS directories may be remapped to specific Cromix directories by giving the letter of the CDOS directory as a flag with the pathname of the Cromix directory as the next argument.

3.112 The Sleep Command

Shell command:	**SLEEP**
purpose:	This command suspends program execution.
user access:	all users
summary:	sleep time
arguments:	time in seconds
options:	none

Description

The **Sleep** command suspends execution of a process for the number of seconds specified. **Sleep** can be used to execute a command after a certain amount of time. For example:

```
jim[1] sleep 60 ; command
```

This example executes **command** after 60 seconds.

3.113 The Sort Utility

utility: **SORT**
purpose: This utility sorts or merges files.

user access: all users

summary: sort [-muvbdfirt?] [-o path] [-n lines] [-l size]
[+x[.y[bdfirv]] ...] [file_list]

arguments: input filename(s)

options:

-m	merge sorted input files
-u	unique records only
-v	evaluate fields numerically
-b	leading spaces and tabs ignored
-d	dictionary order
-f	consider uppercase as lowercase
-i	ignore all control characters and 7Fh
-r	reverse order
-t?	use ? as field separator
-o	output file
-n	number of lines in memory
-l	maximum line size
+x.y	sort on keys

Description

The **Sort** utility sorts the lines in one or more files in ASCII order. ASCII order is: nonprinting characters, blanks, punctuation, digits, uppercase alphabetic characters, and lowercase characters.

Each line (or record) in a file is a string of characters terminated by a newline (0Ah). The "+x.y" option allows you to begin sorting on any character in the line; without the +x.y option, sorting begins on the first valid character.

When called without arguments, **Sort** takes input from STDIN and sends output to STDOUT. An ASCII table is included in the **Cromix-Plus Programmer's Reference Manual.**

Options

The **-b** option ignores leading tabs and spaces, and sorts lines or fields according to their first nonblank character. If the first column shown below is the input file, a sort with no options produces column two, while a sort with the **-b** option produces column three.

INPUT FILE	SORT WITH NO OPTIONS	SORT WITH -b
maser	McKinley	MacDowell
McCormack	MacDowell	MacLeish
MacDowell	McCormack	McCormack
McKinley	MacLeish	McKinley
mace	mace	mace
MacLeish	make	make
make	maser	maser

In column two, the record with the most white space comes first because blank spaces precede characters and letters in an ASCII sort. The last column is closer to alphabetical order, but uppercase entries are first (blanks and tabs are retained, though they do not affect the order of the file).

The **-d** option sorts by letters, numbers, and blanks, but ignores special and nonprinting characters (dictionary order). If the first column shown below is the input file **db.in**, a sort with no options produces column two (standard ASCII sort), while a sort with the **-d** option produces column three. In a dictionary sort, lines with only special characters are sorted in standard ASCII order, but in lines with both special and alphabetic characters, special characters are treated as blanks.

INPUT db.in	SORT WITH NO OPTIONS	SORT WITH -d
a	**a	+++
+++	+++	.444
aaa	+C	BBB
BBBBB	.444	BBBB
C	BBB	+C
+C	BBBB	C
BBB	C	**a
.444	a	a
**a	aAa	aAa
aAa	aaa	aaa

To create a standard ASCII sort in output file **db2.out**, enter:

```
jim[1] sort -o db2.out db.in
```

To create a dictionary sort in output file **db.out**, enter:

```
jim[1] sort -d -o db.out db.in
```

Note that the output of a dictionary sort is not in simple alphabetical order (uppercase letters precede lowercase letters).

The **-f** option treats upper- and lowercase letters equally. Special and nonprinting characters retain their order of precedence. If the first column shown below is the input file **db.in**, a sort with the **-f** option

produces column two.

INPUT db.in	SORT WITH -f
a	**a
+++	+++
aaa	+C
BBBBB	.444
C	a
+C	aAa
BBB	aaa
.444	BBB
**a	BBBBB
aAa	C

The **-r** option reverses the ASCII order of precedence (e.g., "**z**" precedes "**a**"). If the first column shown below is the input file **db.in**, a sort with the **-r** option produces column two.

INPUT db.in	SORT WITH -r
cat	scat
hat	sat
scat	pat
sat	hat
pat	cat
gnat	1
splat	!Wo Fat
slat	gnat
1	splat
!Wo Fat	slat

To create a reversed sort in output file **db.out**, enter:

```
jim[1] sort -r -o db.out db.in
```

Note that the **-r** option completely reverses the ASCII ordering scheme for blank space, letters, and numbers.

The **-m** option merges previously sorted input files to create an ordered output (provided that both input files are sorted according to the same scheme). As shown below, the input file **db.in** (previously sorted without options) can be merged with itself using the **-m** option, as follows:

```
jim[1] sort -m -o db.out db.in db.in
```

INPUT db.in	**SORT WITH -m**
Bat	Bat
Fat	Bat
cat	Fat
rat	Fat
	cat
	cat
	rat
	rat

The merged output is in ASCII order, as are the input files. With adequate disk space, you can merge any number of files.

The **-u** option deletes the duplicate records in a file, leaving one copy of each record. For example, using the output file db.out from the previous merge operation as an input file, the command

jim[1] **sort -u -o u.out db.out**

creates an output file **u.out** that is identical to **db.in** of the previous example (no duplicate entries). For the **-u** option to identify two records as a match, they must be identical in all respects.

The -v option causes all lines and fields to be sorted numerically. If this option is used, the program assumes all fields are numeric and all non-numeric characters are discarded. Valid numeric characters are: "+", "-", ".", and the digits (0 - 9). The numeric option can also be appended to individual fields. For example the command:

jim[1] **sort +3.0vr filename**

will cause the program to skip three fields and sort the next field numerically in reverse order.

The command:

jim[1] **sort +1.0v data.in**

will produce the following action:

INPUT data.in		**SORTED OUTPUT**	
aaaa	1.01	e	-12
bbbb	-1.0	bbbb	-1.0
cccc	1.0	dddd	zero
dddd	zero	fff	.1234
e	-12	cccc	1.0
fff	.1234	aaaa	1.01

The -i option ignores all control characters and 7Fh (DEL) when sorting (e.g., "**ab^Ic**" would be

identical to "**abc**").

The **-l** option, followed by a number, defines the maximum line length; without this option, lines are limited to 80 characters. If an input line is longer than the maximum line length, the program will abort.

The **-n** option, followed by a number, indicates the number of lines that are kept in memory; without this option, 1000 lines are kept. If all input lines cannot be stored in memory, excess lines are stored on disk and merged later in a single result file. For faster operation when sorting large files, use the **-n** option to increase the number of memory-resident lines.

The **-t** option, followed immediately by a character, indicates that the specified character will be taken as the field separator (rather than space or tab) wherever it occurs in the input file. This option is used only with the **+x.y** option.

The **+x.y** option specifies the first character on the line where sorting begins; without this option, sorting begins on the first valid character on the line. The **x** indicates the number of fields to be skipped (counting from the left end of the line); **y** indicates the number of characters to be skipped within the selected field. If **x=0**, the first field on the line is selected.

Without the **-t** option, each field (except the first one) starts with a space or tab following the last nonblank character of the previous field. If the **-t** option is used, each field ends with the specified character.

If one or more of the **-b**, **-d**, **-i**, **-v**, or **-r** options immediately follow the "**y**" value, they affect only the selected field; otherwise, they affect the entire sort. As shown below, the input file **alpha** has four records, each with eight fields of 3 bytes each. Note that all fields except the first one start with a blank.

```
abc def ghi jkl mno pqr stu vwx
Abc dEf ghI Jkl Mno pQr sTu Vwx
aBC DeF gHI Jkl mNo Pqr StU vWx
ABc dEF GHi jKL mnO pQR sTu VWx
```

To sort the file in dictionary order, starting with the first character of the second field ("**def**"), enter:

```
jim[1] sort -d +1.0 -o alpha.out alpha
```

The output file **alpha.out** would be:

```
aBC DeF gHI Jkl mNo Pqr StU vWx
ABc dEF GHi jKL mnO pQR sTu VWx
Abc dEf ghI Jkl Mno pQr sTu Vwx
abc def ghi jkl mno pqr stu vwx
```

To sort alpha using only the second and third characters of the second field (not counting the spaces), enter:

```
jim[1] sort +1.1 -db -o alpha.out alpha
```

The **d** and **b** options affect only the second field.

In a more complex example, the file who contains information on immigrants to California, as follows:

Lopez	Jack	Spain	Bower	Orange
McNiff	John	England	Rose	Sonoma
Rizzo	Jill	Italy	Bly	Kings
Ross	Jerry	Wales	Green	Placer
Mcniff	John	England	Greer	Placer

To sort the file by country of origin, last name, first name, and county (starting with the first character of each field and ignoring upper and lowercase letters in the last-name field), enter:

```
jim[1] sort -t^I +2.0 +0.0 -f +1.0 +4.0 -o who.out who
```

Since each field is separated by a tab rather than blanks, the **-t** option selects the tab (CONTROL-I) as the field separator. The output file **who.out** would be:

Mcniff	John	England	Greer	Placer
McNiff	John	England	Rose	Sonoma
Rizzo	Jill	Italy	Bly	Kings
Lopez	Jack	Spain	Bower	Orange
Ross	Jerry	Wales	Green	Placer

Because of the **-f** option, the difference in capitalization between Mcniff and McNiff is ignored, and the two records are sorted by county.

3.114 The Split Utility

utility:	**SPLIT**
purpose:	This program can split a file into smaller pieces.
user access:	all users
summary:	split [-f] [b #] [-l #] input-file [output-file]
arguments:	file to be split optional mold for resulting files
options:	-f overwrite target files -b # make resulting files # blocks (512 bytes) each -l # make resulting files # lines (up to 512 bytes) each

Description

The **Split** utility can be used to break a large file into smaller chunks. Output file names are formed from the second argument by appending extensions:

.aa
.ab
.ac
....
.zz

in this order. If the second argument is not given, the first argument is used to form output file names.

Each chunk will have a prescribed number of lines (if the **-l** option is used) or a prescribed number of blocks (if the **-b** option is used). For the purpose of this discussion, lines are limited to 512 characters, blocks are 512 bytes each. The last file can have a smaller number of lines (blocks). In the absence of the **-l** and **-b** options the resulting files will have up to 1000 lines each.

If a "-" is given as the first argument, standard input will be split. In this case the second argument is mandatory.

Once a file has been split it can be concatenated together using the **Type** utility.

Options

The **-f** option forces the output files to be overwritten.

The **-b** option, followed by a number, will force output files to contain at most the specified number of blocks.

The **-l** option, followed by a number, will force output files to contain at most the specified number of

lines.

Example

Suppose the file **x** is 2000 bytes long. The command

```
jim[1] split -b 1 x
```

will split the file **x** into:

x.aa	512 bytes
x.ab	512 bytes
x.ac	512 bytes
x.ad	464 bytes

Provided there are no other files with names **x.***, the command

```
jim[2] ty x.* > y
```

will create the file **y** which is an exact copy of **x**.

3.115 The Spool Utility

utility: **SPOOL**

purpose: This utility queues files and sends them to a printer.

user access: all users

summary: spool [[devname] pathname(s)]

arguments: device name

If no device name is specified, output is directed to **/dev/prt**. The device name may be used to direct the output of the **Spool** program to any of the system's printers.

pathname

Filenames must be used to add files to the printing queue. Filenames or the sequence numbers assigned by the Spool program may be used to delete or change priority. If no pathname is given, input from STDIN (terminated by CONTROL-Z) will be spooled.

options:

-a	all files
-d	enter and delete
-h	header
-f	FORTRAN filter
-v	verbose
-j	append NEWLINE at the end
-g	do not append FORM FEED at the end
-w [ON/OFF]	wrap around
-r [ON/OFF]	CRDEVICE flag
-p #	priority of spooled files
-m #	multiple copies
-b #	bottom margin
-s #	character width
-t #	line height
-u #	left margin
-z #	forms number
-n name	define file name

Commands

-l	list
-c #	change priority
-e #	change forms
-k	kill
-q	quit

Description

The **Spool** utility allows one or more users to send printing jobs to one or more printers in an orderly sequence that may be changed at any time.

If no file is specified, input is taken from STDIN. This means the **Spool** utility can be used with redirected input or pipes.

When the **Spool** utility is called to add files to the printing queue, the files are copied into a directory named **/usr/spool**.

After the execution of the **Spool** program with any of its options, the specified files are sent to the printer. This is accomplished by a function intrinsic to the Cromix Operating System.

Output from the **Spool** program may be directed to any character device located in the current directory or in the device table (**/dev**).

If no device is specified, **/dev/prt** is assumed. The Cromix Operating System, as shipped, assumes a dot-matrix printer as the system printer. If a different printer is to be used as the system printer, change the printer type (refer to the **Cromix-Plus System Administrator's Guide**).

As requests are made to print additional files, the **Spool** program forms a print queue. Each file entered into the queue is assigned a unique sequence number. Once in the printing queue, files may be referenced by their filename or sequence number.

If two or more files in the queue have the same filename, a reference to that filename refers to all files with the same name. For example, if the **-k (kill)** option is used with a filename that appears more than once in the queue, all files with that name are deleted from the queue. The sequence number can always be used to refer to a specific file.

Each file added to the printing queue is assigned a priority number ranging from 0 to 9. Zero is the highest priority and is reserved for a privileged user. If no priority is specified, a value of 5 is assigned automatically. A priority number must be specified when using the **-c** (change priority) option.

If two users request a print job with the same priority, the requests are serviced on a first-come, first-serve basis.

A user other than a privileged user has access only to files that the user placed in the printing queue. The priority of a file in the printing queue can be changed by the user who initiated the printing request or by a privileged user. In a similar manner, only the privileged user or the user who added a file to the printing queue can delete the file from the queue by using the **-k (kill)** option. Any user can list all of the files in the printing queue by using the **-la (list all)** options.

Ambiguous file references must be used with caution. When an ambiguous file reference is expanded, it generates a list of filenames matching files in the current directory. An ambiguous file reference can be used when giving the **Spool** program files to add to the printing queue.

Ambiguous filenames are expanded from a directory, and not from a spool queue. An ambiguous file reference does not work properly when killing or changing the priority of files in the printing queue if

files with the same name do not exist in the current directory. This is the case when the **-d (delete)** option is used to add files to the printing queue, or if the current directory is changed by the user.

If **Spool** is interrupted for any reason, such as a power failure, jobs are left in the queue. There are three ways to restart **Spool**. (Before restarting **Spool**, the printer should be manually brought to top-of-form.)

The first method is to spool another job. This restarts printing at the beginning of the first job in the queue (the job that was interrupted).

The second method, used when there are no more jobs to be spooled, is to enter the command line:

system[1] **daemon /dev/yyy**

where yyy is the device name of the printer being spooled to (usually prt). This also restarts printing at the beginning of the interrupted print job.

The third method is to delete all spool jobs using the command line:

system[1] **spool -qa**

and then respool all unprinted jobs.

Options for Spooling Files

The **-d** option adds all specified files to the spool queue and deletes them from the directory in which they reside. This option may include a device name, and must include a list of one or more filenames.

The **-f** option interprets the first character of each line according to FORTRAN conventions:

0	double vertical space before printing
1	formfeed before printing
+	no vertical movement
other	single vertical space before printing

The **-j** option causes the **Daemon** to append the NEWLINE character at the end of the user file.

The **-g** option prevents the **Daemon** from appending the FORMFEED character at the end of the user file.

The **-h** option causes all specified files to be preceded by a one-page header. The first line of the header page contains the name of the user who spooled the file, the date and time, and the name of the file. This is followed by the same information displayed in large characters. The large character portion of the header page truncates the user and filenames to eight characters. Note that the header uses the full width of standard 132-column paper.

The **-m** option prints files a specified number of times. The maximum number of copies is 255.

5. Priority of printing job
6. Pages in printing job
7. Lines in printing job
8. Copies to be printed
9. Forms number to be used

A privileged user always gets a list of all jobs in the printing queue.

The **-la** option lists all printing jobs in a table. Refer to the list option.

Options for Changing Priority

The **-c** option sets the priority of all specified files in the spool queue to the specified value. This option is followed by a priority number, and must include a list of one or more filenames or sequence numbers.

Options for Changing Forms

The **-e** option sets the forms number of all specified files in the spool queue to the specified value. This option is followed by a forms number, and must include a list of one or more filenames or sequence numbers.

Options for Removing Files from the Spool Queue

The **-k** option deletes all specified files from the spool queue. If a specified file is printing, the printing is aborted. This option must include a list of one or more filenames or sequence numbers.

The **-q** option deletes all files that have been directed to the specified device from the spool queue.

The **-qa** option may be used only by a privileged user. It deletes all files that have been directed to the specified device from the spool queue.

Notes

Where no option is specified, the files specified by the pathname are added to the printing queue. A device name may be specified.

If more than one option is used, and one or more of the options requires an argument, the following syntax should be followed:

```
jim[1] spool -hv -m 3 -p 1 filename
```

The options that do not require arguments (**h** and **v** above) are grouped, preceded by a hyphen (-), and followed by a space. This group is followed by the option(s) that require arguments. Each option is preceded by a hyphen and followed by a space, a number, and another space. Additional option and

argument pairs may follow. Finally, the filename(s) of the file(s) to be spooled are entered.

In the following examples, assume the print files **t, u, w, x, y,** and **z** exist in the current directory. First, place each of these files in the printing queue:

```
jim[1] spool -v t u w x y z
t
u
w
x
y
z
jim[2]
```

Because the verbose option is used, the **Spool** program listed each file as it was copied to the spool directory. The list option is then used to display the printing queue:

```
jim[1] spool -l
```

Filename	User	Seq	Device	Pri	Pages	Lines	Copies	Form
-> t	fred	36	5:5 prt	5	2	95	1	0
u	fred	37	5:5 prt	5	2	107	1	0
w	fred	38	5:5 prt	5	1	42	1	0
x	fred	39	5:5 prt	5	2	115	1	0
y	fred	40	5:5 prt	5	2	115	1	0
z	fred	41	5:5 prt	5	3	160	1	0

The arrow at the upper left of the listing indicates the file currently being printed. All jobs have a priority of 5 because no priority was indicated when the jobs were put in the queue.

Next, change the priority of file **y** to 2 and change the priority of the file with the sequence number 39 (file **x**) to 3. Then, delete the file **u** from the queue using the **-k** option. Finally, add a "message" to the printing queue, and display the revised printing queue. A "message" is input from the terminal (STDIN), which the user sends to the printing queue by typing CONTROL-Z (end-of-file).

```
jim[1] spool -c 2 y
jim[2] spool -c 3 39
jim[3] spool -k u
jim[4] spool
this is a message
^Z
jim[5] spool -l
```

Filename	User	Seq	Device	Pri	Pages	Lines	Copies
-> t	fred	36	5:5 prt	5	2	95	1
y	fred	40	5:5 prt	2	2	115	1
x	fred	39	5:5 prt	3	2	115	1
w	fred	38	5:5 prt	5	1	42	1

z	fred	41	5:5 prt	5	3	160	1
----	fred	42	5:5 prt	5	1	2	1

To spool multiple copies of a job, use the -m option.

Example:

The command:

jim[1] **spool -m 3 pay7000**

prints 3 copies of the report **pay7000**.

The command:

jim[1] **spool -hm 3 pay7000**

prints 3 copies of **pay7000** with one header page at the beginning of each copy.

A pipe can be used to redirect output from a program to the printer. The following command line prints a list of the files in the current directory:

jim[1] **ls | spool**

3.116 The Startup Utility

utility:	**STARTUP**
purpose:	This file contains commands that are executed whenever the system is started up.
user access:	all users
summary:	startup
arguments:	none
options:	none

Description

As shipped, the **Startup** program (**/etc/startup.cmd**) executes the **Time** program to set the system clock and date. Additional commands may be added to **/etc/startup.cmd** at the System Administrator's discretion.

After the system is booted, **Startup** notices whether the system was last shutdown properly. If it was not, **Startup** informs the user the check program should be run to verify file system integrity.

3.117 The Stdload Utility

utility:	**STDLOAD**
purpose:	This program loads a program into an STDC.
user access:	privileged user
summary:	stdload filename device
arguments:	filename of program to be loaded
	std device to be loaded
options:	none

Description

The **Stdload** utility loads a file into an STDC (hard disk controller). This utility is normally used to load the STDC firmware **/etc/stdcfirm** into an STDC. **Stdload** is found in the **/etc/** directory.

NOTE: The device must be connected and be able to be opened or the firmware will not be downloaded.

3.118 The Strcmp Utility

utility:	**STRCMP**	
purpose:	This program test for equality between the contents of stdin and a set of strings.	
user access:	all users	
summary:	strcmp [-efr] [-m #] string(s)	
arguments:	one or more strings	
options:	-e	exact
	-f	compare first characters
	-m #	compare first # characters
	-r	reverse sense of test

Description

This utility compares the string **read** from **stdin** to the set of strings on the command line and sets an error return code if none of the strings matches the contents of the input string.

The test made by **Strcmp** is case-insensitive unless the **-e** option is used. **Strcmp** tests for equality between the strings. To locate text strings embedded in the text of a file, use the **Match** utility.

Options

The **-e** option forces the match to be case sensitive.

The **-r** option reverses the sense of **Strcmp** by setting the error code if a match is not made.

The **-m** option must by followed by a number. The **Strcmp** utility will check the specified number of leading characters.

The **-f** option checks only the first character of the input string against the first character of each of the control strings.

Example

```
echo -n "Do you want to shut down the system?"
input | strcmp YES OUI SI
if -err goto noshutdown
kill -2 1
%noshutdown
```

The example above is a typical command file that uses **Strcmp** and **Input**. The first line sends the

string within quotation marks to the standard output. The second line uses the **Input** utility to send the user's response into the pipe to be analyzed by the **Strcmp** utility. The **Strcmp** then tests for occurrences of the strings YES, OUI, or SI. If any of the control strings was entered, the system is shut down using the **Kill** command. If the user did not enter any of the control strings, **Strcmp** sets an error code. The command that follows passes control to the label **noshutdown**. If the user answers no to the question, the system is not shut down.

3.119 The Synchronize Command

Shell: command:	**SYNCHRONIZE** or **SYNC** This program provides a one-time flush of system buffers.
user access:	all users
summary:	sync
arguments:	none
options:	none

Description

System buffers should be occasionally written to the disk to prevent a loss of data in case of system failure. System buffers are written to the disk:

- at shutdown
- at unmount (unmounted device only)
- whenever a user logs out
- whenever the system is idle for more than four seconds
- whenever the flush utility (if it is running) says so
- whenever the **Sync** command is issued

3.120 The Sysdef Utility

utility:	**SYSDEF**
purpose:	This program generates a configuration file for the Crogen utility.
user access:	all users
summary:	Sysdef output_file input_file
arguments:	Pathname of system configuration file followed by the pathname of configuration description file.
options:	none

Description

The **Sysdef** utility is a specialized compiler which generates the configuration module used by the **Link68** utility to generate a new **cromix.sys** file (See the **Crogen** utility description). The input file, usually called **Sysdef**, contains a Shell like description of the system parameters. The following are general rules:

Each line has a number of fields separated by white space. The first field is the name of the parameter being defined, the remaining fields are the arguments. Their precise form depends on the parameter being defined. A percent sign marks the end of useful information. The text in a line following the percent sign is a comment. All numbers are entered in decimal form. The order of definition of individual system parameters is arbitrary.

The remainder of this text describes the precise meaning of the system parameters.

Maxmem

The **Maxmem** variable defines the size of the system memory in 256K units. This unit is chosen because the smallest memory card is 256K bytes. The number of such units supported must follow the command name. One memory unit is the smallest reasonable value and defines a minimal Cromix-Plus system which would be quite limited. During the initialization process, Cromix-Plus will verify all **maxmem** memory units in 16K units. Each 16K unit will be reported as present (+), absent (-), or erroneous (E). This means that a Cromix-Plus system can be built with an arbitrarily large maxmem (62 being the largest reasonable value). This would cause Cromix-Plus to check a large amount of nonexisting memory. Memory units above **maxmem** will not be referenced, memory units below maxmem will be used if they exist and are usable. Example:

```
maxmem    2
```

means 1/2 Megabyte of memory, a reasonable amount of memory in which to run Cromix-Plus.

CDEV

Cromix supports 16 character drivers, numbered from 1 to 16. These numbers are also used as (character device) major device numbers. The CDEV command must be followed by the major device number. If a device number is not used, further data is not required. If the device number is to be used, the device number must be followed by the driver name and if necessary, by the arguments the driver requires. If a driver requires arguments, they must be integers separated by white space. The list of driver names supplied in the **iolib.o68** file, together with the explanation of the meaning of the arguments if applicable, can be found at the end of the **sysdef** file supplied with Cromix-Plus in the **/gen** directory. Example:

```
CDEV   01      tty     0 2 5
CDEV   02
CDEV   03      sysdev
.....
```

says that major device number 1 is the **tty** driver which will support the minor device numbers 0, 2, and 5, (eg. the FDC terminal and two TUART terminals hooked the first TUART). Next, the major device number 2 will not be supported, major device number 3 will be the system device (required), and so on.

BDEV

Cromix-Plus supports 12 block device drivers, numbered from 1 to 12. These numbers are used as (block device) major device numbers. The BDEV command must be followed by the major device number. If a device number is not to be used, further data is not required. If the device number is to be used, the device number must be followed by the driver name and by any arguments the driver might require. If a driver requires arguments, they must be integers separated by white space. The list of driver names, supplied in the **iolib.o68** file, together with the explanation of the meaning of the arguments, can be found at the end of the **sysdef** file supplied with Cromix-Plus in the **/gen** directory. Example:

```
BDEV   01      cflop
BDEV   02      uflop
BDEV   03
.....
```

This says that the major device number 1 is the (Cromix) floppy driver which can support up to four 8 inch or 5 inch floppies. Major device number 2 will support up to four uniform style floppies, major device number 3 will not be used, etc.

RAW

Cromix-Plus uses the concept of a raw character driver. This is a very simple driver which is not accessible to user programs. It is used by the system only for:

- memory test display during startup

- definition of root device during bootstrap
- display of error detailed error messages such as disk I/O errors
- display of catastrophic error messages when the normal system functions are considered to be too dangerous to use (e.g. unexpected interrupt)

Example:

```
RAW     raw_fdc
```

defines the FDC terminal (terminal connected to 64FDC/FDX controller) to be (also) a RAW terminal.

The raw driver is used as an input device only to determine the root device if the root device was not defined in the **sysdef** file when the system was generated. This means that Cromix-Plus can run with the RAW terminal not connected at all. However, this practice is not recommended. A normal solution is to use the FDC terminal both as a RAW terminal and as a normal terminal (**tty1**).

Cromix 11.XX and Cromix 20.XX did not have a sufficient number of process tables and/or Shell buffers since both were bound to be allocated in the single 64K bank which was occupied by the kernel and the drivers. A gain of a few K worth of memory was important. Cromix-Plus does not have this limitation: all of the available memory can be used for process tables if a user so decides.

Tty terminals were inferior to **qtty** terminals as the **tty** terminals could lose characters when interrupts were disabled for substantial periods of time. This problem has been greatly reduced and a **tty** terminal is nearly as reliable in this respect as a **qtty** terminal.

ROOT

The file **cromix.sys** which is to be booted may be read from any device (automatically from **Reset** or explicitly using the **Boot** command). When the system file begins execution it must determine which device is going to be the root device. The ROOT command offers 3 possibilities.

```
ROOT    none
```

means that Cromix will ask the operator (using the RAW device) to determine the root device.

```
ROOT    boot
```

means that the root device should be the same as the device from which the file **cromix.sys** was read.

```
ROOT    6 0
```

For example, means that the root device is to be the device with the major device number 6 and the minor device number 0 (presumably std0).

LOGMSG

During the login procedure, the system identifies itself. Identification consists of three lines:

Version of the operating system
Copyright message
LOGMSG message

The third line is determined by whatever follows the LOGMSG command. For example:

LOGMSG My Customized Version from Nov 3, 1984

It is recommended that the user always modify the LOGMSG message whenever a **sysdef** file different from the distributed version is used.

ACCESS

The ACCESS command defines the access permissions which newly created files will be given by Cromix-Plus (until the access permission is changed). For example:

ACCESS rewa.r.r

The argument following the ACCESS keyword has the same form as for the **access** utility.

Bufcnt

Cromix-Plus keeps **Bufcnt** disk blocks in the memory. This implements the idea of a virtual disk: blocks are read from the main memory and written to the main memory. In the case where a requested block is not residing in main memory, it will actually be read from the disk (or written to the disk to make room for other blocks). A bigger **Bufcnt** might substantially increase throughput. The trade-off is that more memory is used (each block takes 544 bytes) and more widespread damage may occur in the case of a power failure. Note that the Shell contains the **Sync** command which will flush buffers at request. Also, the **Flush** utility should be run in the background.

Inocnt

Cromix-Plus keeps **Inocnt** inodes in memory. This implements the idea of a virtual disk: inodes are read from main memory and are written to main memory. In the case where a requested inode is not residing in main memory it will actually be read from the disk (or written to the disk to make room for other inodes). A bigger **Inocnt** might substantially increase throughput. The trade-off is that more memory is used (each inode takes 144 bytes) and more widespread damage may occur in the case of a power failure.

Filcnt

Filcnt defines the total number of files that can be open simultaneously. Each file structure takes 18 bytes.

Chcnt

Chcnt defines the number of files a process can open simultaneously. Each channel takes four bytes in every process table so that the total amount of memory taken by channels is **4*chcnt*usrcnt** bytes.

Usrcnt

Usrcnt is the number of process tables. **Usrcnt** defines the maximum number of processes which can exist at any time. Each process table takes 1,554 bytes. The maximum number of processes allowed is 255.

Ptbcnt

Ptbcnt defines the number of page tables being used for memory management. The page tables are used as follows. Each page table covers a segment (512K) of memory positioned on a segment boundary. Each process must get a page table for every segment it has access to. In general, one page table will do unless a process uses a lot of memory or is unluckily split across segment boundaries. The kernel itself will also take a page table for each segment that belongs to it. Also the shared code management will make a copy of page tables for shared code. Each page table takes 256 bytes.

Shtmni

Shtmni defines the number of different shared text that might be used simultaneously. Unless the user (carefully) writes a shared text himself, only the following programs use this possibility: **Shell**, **Gtty**, and **Z80**.

Mntcnt

Mntcnt defines the number of devices that can be mounted at the same time. Each entry has a price of 22 bytes.

Lckcnt

Lckcnt defines the number of locks (See **_lock** system call) that can be installed in the system. The price is 24 bytes for each lock.

Freecnt

The system has a small **alloc-free** mechanism used to obtain small chunks of memory. **Freecnt** is its size. Do not change it unless you understand the ramifications completely.

Argvcnt

All Cromix releases up to release 153 inclusive, severly restricted the size of arguments passed to a forked program. The limit was 1024 bytes. Releases 154 and later allow **argvcnt** bytes of arguments to passed to forked programs. The **Shell** utility, source version 1.23 and higher, will expand any number of arguments, limited only by the amount of available memory. However, if the generated list exceeds argvcnt limitation, the program will not execute.

Charcnt

Every character device uses a number of character queues, usually three. Each character queue is limited to 20 character buffers. (Each character buffer uses 16 bytes and can hold up to 12 characters.) If a character driver cannot find a free character buffer when needed, the process is put to sleep until one is available. All Cromix releases up to release 154 inclusive, use 64 character buffers. Release 155 introduced the **Charcnt** parameter to define the number of character buffers.

Maxlev

Occasionally users want to write very unusual programs that want to have interrupts disabled while they are running. The correct solution in such a case is to write a driver and let the driver fiddle with interrupts. As this seems to be an unpopular solution, Cromix-Plus provides the **setlev** system call to set the interrupt level of the processor to a prescribed level. **Maxlev** value is the maximum value accepted by the **setlev** system call. Users are strongly encouraged to set the **Maxlev** parameter to zero. This has the effect that setting the interrupt level to **zonzero** value is an illegal operation, setting it to level zero has no effect.

Msgcnt

Msgcnt is the number of bytes that the system will set aside for the formation of message queues. The **_msgsnd** system call will go to sleep or return an error if there is not a sufficient number of bytes left in the message pool.

Msgmax

Each message send by the **_msgsnd** system call can have at most **Msgmax** bytes.

Msgmnb

One message queue can contain at most **Msgmnb** bytes of messages.

Msgmni

The system can support up to **Msgmni** message queue identifiers. (Each message queue identifier takes

up 46 bytes).

Msgtql

The total number of messages the system can support is equal to **Msgtql**. (Each message header takes 18 bytes of memory).

Shmmax

One shared memory segment can be at most **Shmmax** bytes. The system will always assign shared memory segments in full 4K pages.

Shmni

The system will support up to **Shmni** shared memory identifiers. (A shared memory identifier takes 42 bytes of memory).

Shmseg

One process can request up to **Shmseg** shared memory segments.

Shmall

The total memory size used by shared memory segments can be up to **Shmall** bytes.

Semcnt

The semaphore pool will have **Semcnt** bytes. Each semaphore needs eight bytes from this pool.

Semmni

The system will support up to **Semmni** semaphore identifiers. Semaphore identifiers take 46 bytes each.

Semmsl

Each semaphore group, that is, each semaphore identifier, can have up to **Semmsl** semaphores.

Semopm

The **_semop** system call can do a number of semaphore operations in parallel. This number of simultaneous operations is limited to **Semopm** operations.

Semmnu

The system has room for **Semmnu** semaphore **undo** structures. The memory usage is 6 + 8 * **Semume** bytes per **undo** structure.

Semume

One process can use up to **Semume** undo entries.

END

The END command is a delimiter which indicates that the rest of the file is considered to be comment. In this section you will find the names of drivers together with descriptions of arguments (if applicable).

3.121 The Tail Utility

utility:	**TAIL**
purpose:	This program displays the last part of a file.
user access:	all users
summary:	tail [-c #] [-b #] [-l #] [-f] [filename]
arguments:	optional file name
options:	-c # display # characters -b # display # blocks of 512 characters -l # display # lines -f follow up

Description

The **Tail** utility displays the last part of the file. If there is no filename given, the standard input will be displayed. By default, the **Tail** utility will display the last 10 lines.

Options

The **-l** option followed by a number will cause the **Tail** utility to display the specified number of lines.

The **-c** option followed by a number will cause the **Tail** utility to display the specified number of characters.

The **-b** option followed by a number will cause the **Tail** utility to display the specified number of blocks. A block is 512 characters.

With the -f ("follow") option, if the input file is not a pipe, the program will not terminate after the tail of the input file has been copied, but will enter an endless loop, wherein it sleeps for two seconds and then attempts to read and copy further records from the input file. Thus, it may be used to monitor the growth of a file that is being written by some other process.

3.122 The Tar Utility

utility: **TAR**
purpose: Creates and retrieves file archives.

user access: privileged users

summary: tar -rxtc[vwfblmkoi] [archive block kilobytes] [filename(s)]

arguments: one per command, from the following list:

-r	add files to end of archive
-x	extract files from archive
-t	list files on archive
-c	create files on new archive

options: options, grouped (as applicable) after one of the preceding arguments.

v	list the name of the file being processed
w	wait for user confirmation before processing
f	next argument is the name of the archive
b	next argument is blocking factor
l	unresolved links reported
m	modification times not restored
k	next argument is archive volume size (in kilobytes)
o	do not check for overwriting of existing archive
i	cancels use of file system identifier on floppy disks

Description

The **Tar** utility can be used to create file archives on tape, floppy disk, or an ordinary file. Before using **Tar** to back up files onto floppy disks, the disks must be initialized for the Cromix Operating System. (Refer to the **Init** utility.) Single- or double-sided, single- or double-density, disks can be used.

Once an archive is created, its contents can also be retrieved using **Tar**.

Arguments

One of the following for each **Tar** command:

-r The named files and/or directories are added to the end of an existing archive.

-x The named files and/or directories are extracted from the archive and transferred to the current directory (unless the extracted files are

governed by absolute pathnames). The owner, modification time, and access are restored for ordinary files and directories--unless a directory already exists, in which case, the existing characteristics are retained. If no file argument is given, the entire archive will be extracted. For tapes, if there are multiple entries for the same file, the last will overwrite all previous entries.

-t The named files and/or directories in the archive are listed.

-c Creates files on a new archive, but prompts for continuation if existing data will be overwritten.

Options

v Lists each file as it is processed. When used in conjunction with the **-t** argument command, a fuller description of the file is given.

w Prompts for confirmation of each file to be processed. If the user responds by typing **y**, the action is performed; otherwise, it is cancelled.

f Takes the next argument as the archive name. The default archive is **/dev/tp1**. If the name of the file is '-', **Tar** writes to the standard output or reads from the standard input, as appropriate.

b Takes the next argument as the blocking factor (the number of **Tar** blocks per tape block) on a new tape archive. **Tar** blocks are 512 bytes each. The maximum (and default) blocking factor is 16 blocks. Since tape archives are limited to 65,535 tape blocks, the maximum size of a tape archive can vary from 32.7 megabytes to 524.3 megabytes. The blocking factor is determined automatically for existing tape archives.

l Reports when links to the files are not resolved (for use with the **-c** and **-r** arguments). If **Tar** runs out of memory for the link table, the message:

"No room to check links for file : <file name>"

is displayed, and this file and the files linked to it are written to the archive. (Normally, only one copy of the file is saved).

m Modification times for extracted files will be changed to the time of extraction.

k Takes the next argument as the size of the archive, in kilobytes. This option is useful for splitting large files into separate "volumes" on fixed-size devices such as floppy disks. When creating a multi-volume archive, **Tar** will prompt for the next volume. When extracting from a

multi-volume archive, **Tar** only prompts for a new volume if a split file has been partially restored. Without the **k** option, **Tar** will not check for exceeding the disk size. (For use with floppy disks and ordinary files.)

o When used with the -c argument, omits the check for overwriting an existing archive. The **o** option must be used when creating a new tape archive over an existing archive of less than 512 bytes.

i When creating an archive on floppy disk, **Tar** normally puts an 8-byte file system identifier into the first 8 bytes of block 1 (2 bytes for the Cromix version, 3-byte string "tar," 1 zero byte, and 2 bytes for the **Tar** version). **Tar** checks for this identifier when adding, extracting, or listing, and prompts for continuation if it is incorrect. The **i** option cancels use of the **Tar** identifier.

Examples

To view the progress of directory backups to a new archive of one or more small floppy disks, each limited to 390 kilobytes:

```
jim[1] tar -cvfk /dev/sfdc 390 directory_names
```

To view the progress of a file backup to an existing tape archive:

```
jim[1] tar -rfv /dev/tp1 file1 file2 ...
```

To list the contents of a tape archive in long form:

```
jim[1] tar -tvf /dev/tp1
```

Notes

If disk I/O errors occur while reading or writing a floppy disk archive, **Tar** will attempt to recover. On read errors, **Tar** will write the block to the Cromix file and display the location where the questionable block was written. On write errors, **Tar** will stop writing the file, back up to where the file began, and write an end-of-archive at that point. Thus, the volume's prior contents will be intact. **Tar** will then prompt for a new disk and try rewriting the file.

If the message "Cheksum error" appears, the integrity of the file just processed is questionable.

Tar cannot write to an uninitialized tape. To create an archive on an uninitialized tape, write a dummy file of size 512 bytes or greater to the tape and use the **Ddump** utility as follows:

```
jim[1] ddump if=[filename] of=/dev/tp1
```

After the tape is reloaded, **Tar** can write to the tape.

3.123 The Tee Command

Shell command: **TEE**

user access: all users

summary: tee [-a] pathname

argument: pathname

options: -a append

Description

Tee takes input from the standard input file and sends it to the standard output, as well as to the file specified by the pathname provided in the argument.

Options

The **-a** option appends output to the specified file.

Example:

```
jim[1] sort short | tee sort0
```

This command sorts the file **short**, and sends the sorted output to the terminal (standard output) and to the file named **sort0** in the current directory.

3.124 The Term Command

Shell command:	**TERM**
purpose:	This command displays or changes the terminal name.
user access:	all users
summary:	term [terminal_name]
arguments:	optional terminal name
options:	none

Description

Each process is associated with a terminal name, and the name of each terminal is initially defined in the **/etc/ttys** file. To display the name of the current terminal, enter the **Term** command with no arguments; to change the terminal name, enter the **Term** command with the name of the new terminal.

Terminal names are case sensitive, and should be four characters long. The terminal name must be defined by a set of terminal attributes in the **/etc/termcaps** file. If your terminal is not defined in the **/etc/termcaps** file, you must update the file (refer to the section on TERMCAPS).

3.125 The Termcaps Utility

Data base: **TERMCAPS**
purpose: The file **/etc/termcaps** is a data base describing various Terminal capabilities.

Description

The file **/etc/termcaps** describes the Terminal capabilities of various Terminals. A user program can always get the name of its own Terminal using the **ustat** system call. The **Term** command prints it out or, if called with an argument, changes it. The initial Terminal name is obtained from the **ttys** file and is passed from one process to another.

User programs may inspect the **/etc/termcaps** file to find the descriptions of Terminal capabilities for their own Terminal. Once the capabilities have been extracted, the user program can take the differences between Terminals into account.

Structure of the /etc/termcaps file

Please be aware of the difference between the use of the words "Line" and "line".

The **/etc/termcaps** file consists of a number of Lines, each Line describing one Terminal type. In practice it turns out that Lines are too long. A Line consists of the beginning line followed by a number of continuation lines. If a line is Terminated by the backslash character (), the next line is considered its continuation. So: a Line consists of a sequence of lines, each of them except the last is Terminated by the backslash character.

Comment lines are ignored. Comment lines are either blank lines or lines having the pound sign (#) character in the first column.

A Line consists of a number of fields Terminated by colons. The characters in fields are case sensitive, white space should not be used to increase readability. To increase readability empty fields can be inserted.

The first field in each Line identifies the Terminal type and has a special structure. It consists of three subfields separated by vertical bars. The first subfield has two characters and can be used to identify the Terminal type. The second subfield is the official Terminal type. It should be four characters long, as it must match the Terminal type coming from the system as described above. The third subfield is arbitrary and can describe the Terminal type in human readable form.

The remaining fields are used to describe one Terminal capability each. The field must start with the capability name. The capability name consists of two case sensitive alphanumeric characters. The capability names are otherwise arbitrary. As some software relies on finding particular capabilities in the **Termcaps** file some rules must be observed.

Capabilities listed in the **UNIPLUS System V User's Manual** are reserved. Cromix intends to use the capabilities with Unix names to mean the same thing. In addition, the capabilities listed in this

document are also reserved. They are either identical to Unix capabilities or Cromix specific capabilities used by Cromemco software. Note that the list of reserved capabilities will change with time.

The rest of the capability field depends on the capability form. Each capability is of some form. These forms are:

boolean
numeric
string

The boolean form has no further information in the field. The idea is that if such a capability name is included in the Line, the Terminal is believed to have such a capability. If it is not listed, the Terminal is believed not to have such a capability.

The numeric form is followed by the "#" sign which in turn is followed by the numeric value, encoded decimally.

The string form is followed by an equal sign which in turn is followed by the string forming the capability. The string extends to the end of the field (colon).

If the first character after the equal sign is a decimal digit, it does not belong to the string. The string should be padded with that number of NULL characters at the end.

Special characters may be imbedded in the string:

\n	means newline character
\r	means RETURN character
\t	means TAB character
\b	means backspace character
\f	means form feed character
\E	means ESC character
\xxx	means the character with the octal value xxx
\<char>	means character <char>. This allows to put the backslash or up arrow character in to string.
^<char>	means character <char> ANDed with 0x1f.

To be able to describe escape sequences such as cursor movement, it is intended that such escape sequences are first encoded by a function similar to **printf**. This means that arguments can be inserted in the string at appropriate places. The following forms should be recognized:

%d	same effect as in **printf**
%2	same effect as %2d in **printf**
%3	same effect as %3d in **printf**
%.	same effect as %c
%+x	add x to value then %.
%>xy	if value > x add y; will take effect at next %.
%r	reverse next two arguments; will take effect at

	next two arguments
%i	increment all arguments by 1
%%	single %

WARNING: Some capabilities define character sequences that are transmitted by certain keys. Note that you cannot have two character sequences such that one sequence is the leading subsequence of another sequence. For example, if the "**kl**" sequence for the "left arrow" key is defined to be "E[D", you cannot declare the editor escape "**kE**" sequence to be "E". If you do, the characters "E[D" transmitted by the left arrow key will be interpreted as the "**kE**" character followed by the '[' and the 'D' character.

The **Termcaps** decoding function, **tgread**, understands the "**tc**" Terminal capability. The capability "**tc**" is a string capability. Its value must be the Terminal name of some other Terminal (four characters). If this capability is used, it must be the last one. The "**tgread**" function will erase the "**tc**" entry and append the contents of the Terminal description at the end of the current Terminal. This process may be recursive. The final result returned by "**tgread**" is one long string which is the concatenation of all referenced Terminals. If this mechanism is used, it may happen that some Terminal capability is defined more than once. The "**tgread**" decoding functions will always find the first one.

If Terminal capabilities of one Terminal refer to capabilities of another Terminal, it may be necessary to delete a Terminal capability. A Terminal capability, followed by a "@" character, e.g.

```
:dl@:
```

means that all definitions of the "**dl**" capability, which occur textually after the "delete-capability" entry, will be deleted.

Example:

Terminal description:

```
xx|xxxx|terminal xxx:\
:co#132:\
:ho@:\
:tc=yyyy:
```

should be interpreted as follows:

Take description of Terminal "yyyy"
Delete all definitions of capabilities "**ho**" and "**co**"
Add definition of capability "**co**"

This interpretation is correct independently of how the description of the "yyyy" was built.

Here is the list of currently defined capabilities. Each capability is given its name, type and a short description.

al str	Insert line at cursor.
am bool	Terminal has automatic margins
ba bool	Terminal has automatic screen blank-out feature
cd str	Clear to the end of screen.
ce str	Clear to the end of line.
cl str	Clear screen, including move cursor to home position
cm str	Cursor addressing sequence.
co num	Number of columns on the screen.
ct str	Toggle cursor on-off.
dc str	Delete character at cursor position.
dl str	Delete line at cursor.
do str	Move cursor down one line.
ei str	Exit insert line mode.
fc str	Turn cursor off (make it invisible).
ho str	Send cursor to home position.
Ic str	Insert a space (page mode) at cursor position.
ic str	Insert a space (line mode) at cursor position.
il str	A string specific to C-05. It must be followed by character count and that number of characters. These characters will overwrite the leftmost column of the screen.
im str	Enter insert line mode.
ir str	A string specific to C-05. It must be followed by a character count and that number of characters. These characters will overwrite the rightmost column of the screen.
is str	Terminal initialization string. It is written to the Terminal whenever the Terminal type is selected.
kE str	Gives the string used to turn off insert mode for Shell editing capability, or to exit any CE mode.

kI str	Gives the string used to turn on insert mode for Shell editing capability.
kR str	Gives the character used for the Shell **History** capability. The **History** capability (**Retype** capability in previous versions) is always enabled and can work even on a "dumb" Terminal.
kc str	Defines the "Enter control character" key. If not defined, CE will use CNTRL-V.
kd str	String sent by down arrow key.
kl str	String sent by left arrow key.
kr str	String sent by right arrow key.
ku str	String sent by up arrow key.
li num	Number of lines on the screen.
mo str	Go to the monitor mode sequence. This capability is used by CE to display control characters on Terminals that do not provide the "write invisible cursor" mode that is used on the 3102, C-10, C-5.
mf str	Exit monitor mode (see above). If the **mo** capability is defined, **mf** has to be defined as well.
mt bool	If the **mf** sequence gets echoed (like on Wyse Terminals), **mt** should be set so that CE can take corrective action. For more details on that subject check the CE help file section on control characters.
nb str	Nondestructive backspace (move cursor left).
nd str	Nondestructive space (move cursor right).
NR bool	No **Retype** capability.
oc str	Turn cursor on.
pe str	String to turn off insert mode entered with the **pi** sequence.
pi str	String to enter page insert mode. This causes subsequent typing to wrap the rest of the characters to the next line, all the way to end of screen.
pd str	String to delete character at cursor position in page mode. This causes the rest of characters all the way to end of screen to move up.

rs str	Remove standout.
sd str	Select a screen for display
se str	Exit standout.
so str	Enter standout. Sequence to turn on some highlighting.
sq str	Inquiry - which screen is currently active
sw str	Select a screen for write
tc str	Continue with the definition of the Terminal identified by string **str**. This capability, if used, must be the last one.
up str	Move cursor up one line.
wi str	Write with invisible cursor at given position. If **wi** is defined, so **we** must be as well.
we str	Terminating string for write with invisible cursor. The whole invisible write consists of the **wi** string, string to be written, and the **we** string.
wf str	Turn wrap off. Insert in page mode will not shift the last character of each line to the first position of next line.
wo str	Turn wrap on. Insert in page mode will shift the last character of each line to the first position of the next line. If the **wo** string is defined, **wf** must be as well.
xs bool	Standout not erasable by writing over it.

3.126 The Testinp Utility

utility:	**TESTINP**
purpose:	This program tests for equality between the contents of a file and a particular string.
user access:	all users
summary:	testinp [-dfr] file string(s)]
arguments:	file pathname
	one or more strings
options:	-d delete
	-f compare file after test first characters
	-r reverse sense of test

Description

This utility compares the contents of a file to a string or strings and sets an error return code if one of the strings does not match the contents of the file specified.

The test made by **Testinp** is case insensitive. **Testinp** tests for equality between the file and the string. To locate text strings embedded in the text of a file, use the **Match** utility.

Options

The **-r** option reverses the sense of **Testinp** by setting the error code if a match does occur.

The **-f** option checks only the first character of the file passed as an argument against the first character of each of the control strings.

The **-d** option deletes the file passed as an argument after the test. This option is useful in many command files using a temporary file created during the command file execution.

Example:

```
echo "Do you want to shut down the system?"
input > temp
testinp -d temp YES OUI SI
if -err goto noshutdown
kill -2 1
%noshutdown
```

The example above is a typical command file that uses **Testinp** and **Input**. The first line sends the

string within quotation marks to the standard output. The second line uses the **Input** utility to send the user's response to the file **temp**. On the third line, **Testinp** tests the contents of the file **temp** for occurrences of the strings YES, OUI, or SI. **Testinp** then deletes **temp**. If the file contains one of the control strings, the system is shut down using the **Kill** command. If the file temp does not contain one of the control strings, **Testinp** sets an error code. The command that follows passes control to the label noshutdown. If the user answers no to the question, the system is not shut down.

3.127 The Time Utility

utility:	**TIME**
purpose:	This program displays or alters the time and date.
user access:	all users for display privileged user for changes
summary:	time [-se2]
arguments:	none
options:	-s set system values -2 set 3102 clock -e European style display (dd/mm/yy)

Description

The **Time** program displays or changes the time and date. If the -s option is not specified, the current date and time are displayed. If the -s option is used, the user is prompted for the date and then the time. Although the date is displayed with the "/" separator, and time is displayed using the ":" separator, any convenient separator character (such as a space or a period) can be used when entering the date and time. The **Time** utility displays a new prompt until a valid date or time is entered.

Options

The **-s** option causes the user to be prompted for a new date and time. Only a privileged user can specify the **-s** option.

The **-e** option causes the time to be displayed in the European style, with the day and month reversed.

The **-2** option allows the user to set the clock in the user's terminal. The terminal must be a Cromemco 3102, C-5, or C-10 terminal.

Notes

The **Time** utility allows shortcuts when the date and time are entered.

To enter date:

No value given	Keep old values.
One value	Used for day. Month and year unchanged.
Two values	Used for month and day (day and month in the European style). Year unchanged.
Three values	Month, day, and year (day, month, and year in the European style).

To enter time:

No value given	Keep old values.
One value	Used for minutes. Keep old hours, set seconds to zero.
Two values	Used for hours and minutes, set seconds to zero.
Three values	Hours, minutes, seconds.

3.128 The Touch Utility

utility:	**TOUCH**
purpose:	This program changes the modification times of files to the current time.
user access:	all users
summary:	touch [-c] filename(s)
arguments:	pathnames of one or more files
options:	-c do not create files if they do not exist

Description

The **Touch** program changes the modification time of a file to the current time. If the file does not exist, **Touch** creates the file unless called with the -c option.

The **Touch** program is for use with the **Make** program.

Options

The -c option prevents **Touch** from creating files if they do not exist.

3.129 The Tr Utility

utility: **TR**
purpose: Translate characters.

user access: all users

summary: tr [-cds] [string1 [string2]]

arguments: none

options:

-c	complement the set of characters in string1
-d	delete input characters in string1
-s	squeeze multiple output characters from string2

Description

The **Tr** utility copies standard input to standard output.

If the **Tr** utility is called with no options, any input character found in **string1** is replaced by the corresponding character in **string2**, provided the **string2** is long enough. If **string2** is too short, the input character remains unchanged.

The **-c** option complements **string1**. This means that the **string1** used by the program is a string formed of all ASCII characters from 01 to 377 that are not found in the **string1** supplied by the caller.

The **-d** option causes all input characters in **string1** to be deleted.

The **-s** option squeezes all strings of repeated output characters, that are found in **string2**, to a single character.

The characters in strings **string1** and **string2** can contain the following special combinations:

\n	the NEWLINE character
\r	the RETURN character
\t	the TAB character
\f	the FF character
\b	the BACKSPACE character
\ddd	where ddd are three octal digits, means the character whose octal ASCII representation is octal ddd
\any character	means "any character"; this is useful to put special characters like "\" or "[" into strings.

The following abbreviation conventions may be used in strings **string1** and **string2** (these are examples):

[a-z]	all characters from 'a' to 'z' inclusive

[x*10]	sequence of ten characters 'x'
[x*]	as many characters 'x' as needed; this form can be used only in **string2** to pad it to the same length as **string1**.

Example

```
jim[1] tr -cs "[A-Z][a-z]" "\n*"
```

writes to standard output a list of all words coming from standard input, where a word is taken to be a maximal string of alphabetic characters.

The strings are quoted to protect the special characters from interpretation by the Shell.

String1 consists of 52 alphabetic characters. Due to the -c option, the string used by the **Tr** utility will be a string of 255 - 52 = 203 non-alphabetic characters. **String2** consists of 203 NEWLINE characters. Therefore all non-alphabetic characters are replaced by the NEWLINE character. Because of the -s option, all multiple NEWLINE characters are replaced by a single NEWLINE character.

3.130 The Type Command

Shell command:	**TYPE** or **TY**
purpose:	This command displays an ASCII (text) file.
user access:	all users
summary:	ty [file-list]
arguments:	optional file pathnames
options:	none

Description

The **Type** command displays the file(s) specified by the pathname(s). **Type** can be used only to display ASCII (text) files. To display other kinds of files, use the **Dump** utility.

Type uses STDIN when called without an argument, and sends output to STDOUT.

Example:

```
jim[1] ty /dev/qtty5 > diskfile
```

This command line accepts data from **/dev/qtty5** and sends it to **diskfile**.

3.131 The Uboot Utility

file:	**UBOOT.SYS**
purpose:	Standalone boot program for the UNIX Operating System
user access:	privileged user
summary:	boot uboot
arguments:	uboot
options:	none

Description

The file **uboot.sys** is a standalone boot program for the UNIX Operating System. **Uboot** reads the UNIX kernel from the default device (std(2,0)unix). If you press the ESC key immediately after giving the "boot uboot" command, you will be prompted for the device name, the major and minor device numbers, and the filename. Currently supported devices are:

std	STDC disk
sfd	small floppy
fd	large floppy

Uboot.sys may be patched to use another default device. For example to change the default device to std(33,0) use the following sequence of commands: (User input is in boldface).

```
system[1] patch uboot.sys
>q Osfffff 'std('
0000XXXX: 73 74 64 28   20 32 2c 30 - 29 75 6e 69 78   00 73 74 std( 2,0)unix.st
> s XXXX
0000XXXX:  20 '33'
0000XXXX:  2c .
> e
system[2]
```

The value entered in the substitute command (s XXXX) is the value displayed by the query command plus 4.

3.132 The Unmount Utility

utility:	**UNMOUNT**
purpose:	This program disconnects a mounted file system from the current file system.
user access:	privileged user
summary:	unmount [-x] devname [devname ...]
arguments:	device names
options:	-x do not eject disk

Description

The **Unmount** utility program disables access to a file system. A file system that has been mounted must be unmounted by use of the **Unmount** utility before the mounted disk is removed from the system or the system is powered-down. If this is not done, the integrity of the data on the mounted system cannot be assured.

Options

The **-x** option causes a floppy disk not to be ejected when it is unmounted (PERSCI drives only).

3.133 The Update Utility

utility:	**UPDATE(1...2)**	
purpose:	These command files update an existing Cromix file system from new release diskettes.	
user access:	privileged user	
summary:	update1	device_name
	update2	device_name
options:	none	

Description

The **Update** programs update an existing Cromix file system from new system diskettes.

After booting the system from Disk 1 and rooting on Disk 1, run **Update1**, using the following command syntax:

 # update1 drive

where drive is the destination drive.

Update1 will reboot the system. Boot your newly updated drive and run **Update2** for all additional new system disks using the following command syntax:

 # update2 drive

where drive is the source drive.

Because the **Update** programs are command files, you can refer to the files themselves for information about how they work. Command files are ordinary text files. Thus, you can display them with **Type** or even change them with the **CE** editor.

3.134 The Usage Utility

utility:	USAGE
purpose:	This program displays directory size information.
user access:	all users
summary:	usage [file-list]
arguments:	directory or file pathname(s)
options:	none

Description

The **Usage** utility displays the physical disk space (in blocks) and the logical file space (in bytes) occupied by a directory and all of its descendants. If only a single file is specified, the size of that file is reported. If no pathname is given, the current directory is assumed.

Knowing the number of blocks occupied by a directory is useful before using the **Cptree** utility.

3.135 The Vdt Utility

utility:	**VDT**
purpose:	Special terminal functions
user access:	all users
summary:	vdt function
arguments:	keyword which identifies function
options:	none

Description

The argument to the **Vdt** utility is a keyword which identifies the actual command. The commands available are:

clear	Clear screen.
cursor	Turn cursor on.
cursoff	Turn cursor off.
wrapon	Turn wrap around on.
wrapoff	Turn wrap around off.
clock	Set hard clock.

If the keyword is followed by a message, the message is loaded into status line of the terminal. if there is no message, the status line is cleared.

Note

The **Vdt** command is intended to be used only on a C-5 or a C-10 terminal.

3.136 The Version Utility

utility:	**VERSION**
purpose:	This program displays the version number of the Cromix Operating System or a utility program.
user access:	all users
summary:	version [file and/or directory list]
arguments:	optional file and/or directory list
options:	-c calculate CRC -v verbose

Description

When called without an argument, the **Version** utility displays the version of the Cromix-Plus Operating System currently executing. A simple check for internal consistency is also performed. If the consistency check reveals problems, an appropriate message is displayed instead of the version number. Reboot the system as soon as possible.

When called with the name of a utility program, **Version** displays the version number of that utility. When called with a directory name, **Version** displays the version number of each of the programs in the directory. The following command displays the version numbers of all of the programs in the **/bin** directory:

```
jim[1] version /bin
```

The characters "RB" appearing in an entry indicate that the file is a Relocatable Binary file. Programs written for Z80 Cromix (including relocatable binary files) are allocated 16 pages (64k) of memory by the Cromix-Plus Operating System.

The characters "68" appearing in an entry indicate the file is a 68000 binary file. For these programs, the operating system allocates as many 4k pages of memory as required.

For example:

```
jim[1] version /bin/ls.bin
```

will display something like:

```
68 CRC OK 68020 XXU 149 Ls source 1.8
```

The meaning of individual fields is:

68	It is a 68xxx program.
CRC OK	The file has the correct CRC as computed when the

	file was distributed.
68020	The instructions used in the program are 68020 instructions. This means that this program will not execute on a 68000 or 68010 processor. The programs which utilize 68000 or 68010 instructions will, in general, execute on 68000, 68010, or 68020 processor.
XXU 149	The program was distributed with XXU Cromix, Release 149. The utility is guaranteed to work only with this Release of XXU Cromix. Executed under any other release or processor type, it may or may not work as expected.
Ls	Name of the utility.
source 1.8	Release version of the source used to compile the utility.

The **Version** utility (version 0.10 and higher) searches for the following string of bytes in a file: 0FDH 0EDH 0FDH 0EDH. The bytes immediately following are assumed to contain CRC information, version number, release number, and an optional login message.

To illustrate, consider the following portion of a file:

```
dc.b    0FDH 0EDH 0FDH 0EDH
dc.b    0, 0, 0, 0
dc.b    version, revision
dc.b    'login message  '
```

When called with the **-c** option, **Version** computes a CRC value for the file. The first two bytes immediately following the FDEDFDED pattern contain the version number of the **Version** program; the next two bytes contain the CRC.

When called without the -c option, Version reports good or bad file integrity by comparing a newly calculated CRC with the previously calculated CRC value.

Options

The -c option causes the CRC value calculated for the file to be placed in the file for future comparison. The -v option causes the pathnames of files to be printed.

3.137 The Wait Command

Shell command:	**WAIT**
purpose:	This command suspends execution and waits for the PID-specified process to terminate.
user access:	all users
summary:	wait [PID]
arguments:	optional PID number
options:	none

Description

The **Wait** command causes the Cromix Operating System to suspend operation until the process specified by the process id number (PID) has terminated. If no process is specified, **Wait** suspends execution of the current process until all detached processes belonging to that user have terminated.

3.138 The Wboot Utility

utility:	**WBOOT**
purpose:	Writes the boot program to the boot area of a disk.
user access:	privileged user
summary:	wboot devname [pathname]
arguments:	device name where the boot program is to be written.
	optional pathname of the boot program to be written.
options:	none

Description

The **Wboot** utility writes the boot program into the boot area of a device. This is necessary for a device to be bootable. Devices available are floppy disks, WDI-II disks, and STDC hard disks (though RDOS prior to version 03.12 cannot access them). The boot program to be written is selected by the **Wboot** utility. The programs required must reside in the **/etc** directory (**fdboot**, **sfdboot**, **hdboot**, **stdboot**). As an alternative, a user can write his own boot program if he specifies the appropriate filename as the second argument.

Wboot uses the system drivers to access the device, so verify that there is an entry in **/dev** for the device name, and that you have run **Crogen** to include the system drivers.

3.139 The Wc Utility

utility:	WC (Word Count)
purpose:	This program counts lines words and characters.
user access:	all users
summary:	ls [-lwc] [file-list]
arguments:	optional list of files
options:	-l count lines -w count words -c count characters

Description

The Wc utility counts lines, words, and characters in the named files, or in the standard input if no names appear. It also keeps a total count for all named files. A word is a maximal string of characters delimited by white space characters.

Options

The options can be used in any combination. Calling Wc with no options is equivalent to calling it with all options.

The -l option count lines.

The -w option counts words.

The -c option counts characters.

3.140 The Who Utility

utility:	**WHO**
purpose:	This program displays a list of users who are currently logged in.
user access:	all users
summary:	who [/etc/account] [am i]
arguments:	optional /etc/account
	or
	optional "am i"
options:	-t display also time used

Description

When the **Who** utility is called without an argument, the **/etc/who** file is consulted and a report is displayed showing the users currently logged on, together with the time each one logged on.

When followed by "am i", the name of the user calling the **Who** utility is displayed.

If the **Who** utility is called followed by **/etc/account**, the information contained in the account file is displayed.

If the **-t** option is used together with the **/etc/account** argument, the time (in seconds) used by the terminal when it is logged out, is displayed in addition to the normal contents of the **/etc/account** file.

3.141 The Z80 Utility

utility:	**Z80**	
purpose:	This program is used to run Z80 programs.	
user access:	all users	
summary:	z80 [-s] [-d device] Z80-program-pathname arguments	
arguments:	Z80 program full pathname	
	Arguments to Z80 program	
options:	-s	prints system call usage statistics
	-d device	device name where to run Z80 program

Description

The **Z80** utility is used to run Z80 programs (**.bin** and **.com** files) on any device which contains a Z80 processor and that has the Z80 driver built into Cromix-Plus.

The shell will automatically call the **Z80** utility whenever it has to execute a program that is not a legal 68000 **.bin** file. There is no need for the user to explicitly call the **Z80** utility unless a **Z80** program has to be forked from a 68000 user program or if for some reason the **Z80** program has to run on a particular device.

The **Z80** utility will scan the **/dev/z80** directory to find an unoccupied Z80 device. The **Z80** utility will handle the system calls for the **Z80** program that will be run by a Z80 driver for one of the device types that are supported. For Z80 programs to run successfully

- the **Z80.bin** program must be in the **/bin** directory;
- the **/dev/z80** directory must exist;
- the **/dev/z80** directory must contain a number of Z80 device files;
- device files must have "execute" permission
- the drivers for these device files must be built into Cromix-Plus.

Example:

To run the Z80 Screen editor from the console, enter:

jim[1] **screen file**

or

jim[1] **z80 /bin/screen.bin file**

To use the Z80 assembler, enter:

jim[2] **asmb file**

or

jim[2] **z80 /bin/asmb.com file**

To execute the assembler from a 68000 program, fork a process with the pathname:

"/bin/z80.bin"

and with arguments:

argv[0] = "z80"
argv[1] = "/bin/asmb.com"
argv[2] = filename
argv[3] = 0

or fork a shell with arguments:

argv[0] = "shell"
argv[1] = "-c"
argv[2] = "asmb filename"
argv[3] = 0

Gromemco®

280 Bernardo Ave.
P.O. Box 7400
Mountain View, CA 94039