Section 3
SYSTEM ACCESS

In this section: See page

In this section (cont): See page

Summary

This section describes the way in which you access the system, beginning with system installation and proceeding through user registration, login, and the use of menus and commands to request system functions.

INTRODUCTION

You can request access to the system to perform a number of different functions, such as:

- System building - Configuring the system to the needs of its users.
- System administration - Registering users.
- Operation control - Starting up the system each day, controlling processing, managing peripheral devices, and monitoring system status.
- Program development - Compiling, testing, and debugging programs.
- Application execution - Interacting with a program to accomplish a particular task.

In a large installation, different individuals will perform different functions. In a small installation, one person may perform most or all of the functions.

Access to the system is restricted to authorized users by means of the registration and login processes. Access to system files is restricted to specified users through the access control process (described in Section 2). Access to the various system facilities is controlled through the menu or command environment.

Before any access to the system can be made, the system must first be configured.

SYSTEM CONFIGURATION AND DEFINITION

Creation of a system is a four-step procedure, consisting of:

1. Installing the operating system and application software onto your system disk. Note that this step is not necessary if you receive a disk with the operating system and application software already installed.

2. Bootstrapping a Honeywell Bull supplied system startup routine that provides a limited operating environment for building the files used in the third step.

3. Specializing the system startup procedure by configuring a system to correspond to the installed hardware and by defining the environment in which to prepare and execute application programs.

4. Making a backup copy of the disk containing your specialized system.

System Installation

Honeywell Bull delivers to you one of the following:

1. A disk containing an installed operating system and requested application software. You use this disk to bootstrap the startup routine.

2. A LOAD/SAVE Diskette and a disk or tape containing the operating system and other application software you ordered. The LOAD/SAVE software guides you in installing the operating system onto your system disk. The installation process is described in detail in the *Software Installation Guide*. Once the operating system software resides on your system disk, you bootstrap the startup routine.

Bootstrap

The bootstrap operation consists of turning on the power supply to the hardware, mounting the disk containing the MOD 400 system software, and pressing several keys on the control panel or System Control Facility device. (The procedure is described in the *System User's Guide*.) The bootstrap operation generates the initial configuration and startup operations. The resulting limited, one-user, on-line environment permits you to specialize system startup so that subsequent bootstraps will produce a multi-user environment that is adapted to your site's software requirements. This initial configuration may also be used to develop or execute application programs; however, the standard procedure is to generate a site-specific configuration first.

Configu-ration

To generate a site-specific configuration, you either invoke the Autoconfigurator (DPS 6/22 only) or you create a file (called the CLM_USER file) containing the Configuration Load Manager (CLM) directives that describe the operating environment that will exist at your installation. The CLM_USER file is created automatically by the DPS 6/22 Autoconfigurator. The configuration directives are described in the *System Building and Administration* manual.

START_UP.EC

To further define the environment, you can create a START_UP.EC file (>>START_UP.EC) containing the operator commands that perform installation-specific functions such creating buffer pools. A >>START_UP.EC file is created automatically by the DPS 6/22 Autoconfigurator. (START_UP.EC files are described later in this section.)

After the CLM_USER file and the system START_UP.EC file are created, you again bootstrap the system. This time, the directives in the CLM_USER file control the configuration, and the operator commands in the system START_UP.EC file further define the operating environment.

System Backup Copy

You should make a backup copy of the specialized system disk. Either the Copy utility or the Save and Restore utilities can be used for this purpose.

USER REGISTRATION

User Information

User registration is a process that protects the system from unauthorized access. Each person who is to be allowed on the system must be registered by the system administrator (that is, a user whose account is SYS_ADMIN). The administrator uses the Edit Profile command to specify user-specific information such as:

- User id. (Identifies the user for system purposes. Consists of two or three parts: person.account[.mode]. Uppercase and lowercase characters are equivalent; JONES.ADMIN and Jones.Admin are the same user id.)
- Login id. (Identifies the user for login purposes only. Contains no periods. Uppercase and lowercase characters are distinct; JONES and Jones are different login ids.)
- Default login line
- Login traits, including:
  - Single or multiuser profile
  - Secondary user profile
  - Login only with default login line
- Whether a password is required to log in
- For single-user profiles, whether statistics such as the following are to be kept for the user:
  - Number of sessions
  - Total elapsed time
  - Accumulated system resource usage.

Profiles File

This user-specific registration information is stored in a profiles file. The profiles file contains the user profiles for all registered users. The system checks the user profile when monitoring the privileges and/or limitations of each user.

On the DPS 6/22, the Autoconfigurator creates a default profiles file that you can customize to meet your special requirements.

Profile Sections

User profiles are made up of sections, each of which describes a specific interface to some part of the system. A user profile always contains a registration section (consisting of the information listed above plus the optional user statistics, if specified). It can also contain one or more of the following sections:

- Comments section. Contains any comments the system administrator has about this user's registration.
- Subsystem sections. Contain user-specific information meaningful to individual subsystems. Examples of subsystem sections are:
  - The Office Processing section (WP), which contains information such as the default print queue assigned to the user. Users must be registered with this subsection to use the ONE PLUS menus.
  - The ONE PLUS Menu Subsystem section (OP), which contains information such as the first menu the user is allowed to see. Most users will be registered with this subsection.
  - The Menu User section (MU), which contains information about the User Productivity Facility menus. Users who will be creating their own menus, or who will be executing system commands through menus and forms, must be registered with this subsection.

Viewing Profiles

The List Profile command allows registered system users to view the contents of their user profiles. It allows a system administrator to view any registered user's profile.

Refer to the *ONE PLUS System Administration* manual for details on user registration.

ACCESSING THE SYSTEM

Access to the system requires:

- A physical connection between your terminal and the central processor
- A logical connection between you and the Executive.

Connecting to the Central Processor

Two types of terminal connections are possible: a direct connection and a dialup connection.

Direct Connection

In a direct connection, the terminal is connected to the central processor when the terminal's POWER switch is turned ON and its ONLINE/OFFLINE switch is set to ONLINE.

Dialup Connection

In a dialup connection, the terminal is connected to the to the central processor after a telephone number is dialed on the terminal's modem.

The same terminal can be used both as an operator terminal and as a user terminal.

Refer to the *System User's Guide* for further information on physical connections.

Connecting to the Executive

Listener

You can access a terminal in one of two ways, depending on whether or not the terminal is configured for login. Terminals configured for login are reserved for access to the system through a system component called the Listener. Such terminals cannot be directly reserved by system applications. Terminals not configured for login are not monitored by the Listener and can be directly reserved by system applications. (Refer to the *System Building and Administration* manual for details on configuring the Listener and Listener terminals.)

Systems are more secure when all terminals are monitored by the Listener.

Login Terminals

Login Types

When a terminal is monitored by the Listener, either a login form or a banner line is displayed when the connection to the central processor is made. You can connect to the Executive through one of the following login methods:

- Forms login. Fill out the login form and, if required, enter the designated password.
- Full login. Type the Login command and, if required, enter the designated password.
- Abbreviated login. Type the 1-character login abbreviation and, if required, enter the designated password.

Another login method, direct login, requires no action on your part beyond turning on the terminal and, if required, entering your designated password.

Terminals File

The system builder defines those terminals that the Listener monitors through entries in a file called the terminals file. (On the DPS 6/22, the Autoconfigurator builds a terminals file that you can modify for special requirements.) The terminals file is used to specify the following information:

- Maximum number of concurrently logged-in users to be allowed on the system.
- Whether the terminal is to display a login form or a banner. The Autoconfigurator selects the login form for all terminals.
- Name of each terminal that can be used as a login terminal (plus, for direct login terminals, the login line to be used when the terminal is turned on).
- Each 1-character login abbreviation and its associated login line.
- Other information about the system and specific terminals.

Non-Login Terminals

An application can be activated, with a terminal designated for the input of the commands and/or data required by the application. When this terminal is physically connected to the system, no banner or form is displayed. In most cases, you can start entering data immediately. The screen display and your response depend on the application.

A login terminal can be changed to a non-login terminal (and back again) by the Set Listen command.

Activated Lead Task

When you gain access to the system, the executable code for the lead task (the controlling task of the application) is loaded and activated. The lead task can be designated to be the Command Processor, the Menu Processor, or an application. When the Command Processor is the lead task, you can control execution by issuing any user command described in the *Commands* manual. When the Menu Processor is the lead task, you can control execution by filling out screen forms. When an application is the lead task, neither the Command Processor nor the Menu Processor is part of the task group.

MENU ENVIRONMENT

The ONE PLUS Menu Subsystem provides an easy-to-use interface to MOD 400. You can access all MOD 400 components, including electronic mail, document processing, data base management, and system utilities from one master menu (called the PLUS menu).

PLUS Menu

The PLUS menu groups the system functions according to category. For example, the menu's Document Services category includes functions such as document processing, library operations, index operations, and queue operations. If you want to create a document, you select Document Services on the PLUS menu and, when the Document Services menu is displayed, you select Document Processing. You specify a particular document processing operation by selecting that operation on the Document Processing menu and filling out the fields of the form that is displayed as a result of your selection.

Messages

Each selection on a Honeywell Bull supplied menu, and each form field that may be filled-in, has an associated help message. To obtain help in filling in a field on a form, press the key designated as the help key for your terminal. When the help key is pressed, a help message is displayed below the menu or form. If you make a mistake in filling in a field, the system provides expanded error messages that list causes and corrective actions to aid in resolving the problem.

User Applications

The Menu Subsystem allows you to select user applications from the PLUS menu. To be able to do this, you must have added your applications to the Menu Subsystem. You can add, delete, and modify a menu, form, or message through the following tools:

- Menu Maintenance to create, modify, or delete menus.
- Forms Developer (VISION) to create, modify, or delete forms.
- Generalized Forms Processor (DFC) to use standard forms in applications.
- Add/Delete Message utility to update the message library.
- Table Maintenance utility to build or modify tables used to process forms.

The *Application Development Overview* describes the process of developing user menus and forms.

System Commands

The Menu Subsystem allows you to use menus to execute system commands. To operate in this way, you select User Productivity Facility from the System Services menu. The User Productivity Facility contains a set of menus arranged according to command category. You select system commands by choosing options listed on the menus and then filling in the fields on one or more associated forms. When all appropriate fields have been filled in, the command is generated and executed.

The Menu Subsystem also allows you to enter command lines directly. To operate in this way, you select Command Line from the System Services menu. (For information on command usage and format, refer to the *Commands* manual.)

Menu Processor

The Menu Processor is the system software component that reads the Honeywell Bull supplied menus and forms you fill in. After reading a form (or command line), the Menu Processor loads and executes a bound unit that fulfills the request represented by the form.

The Menu Processor uses the standard system files (command-in, user-in, user-out, and error-out). Refer to "Command Environment" for a complete description of these files.

Menu/Form Level

When the system is in a state in which it is capable of displaying a menu or form, it is said to be at menu/form level.

Achieving Menu/Form Level

The Listener can spawn task groups whose lead tasks are the Menu Processor. Note that the system is delivered with a Honeywell Bull supplied user task group ($H) whose lead task is the Command Processor (described later in this section).

The system indicates it is at menu/form level by displaying a menu or form.

Menu/Form Level Processing

When you fill out a Honeywell Bull supplied menu and press the EXECUTE key, the system displays another menu or a form. When a form is displayed, you enter the necessary data and press the EXECUTE key. The function you requested is executed.

Command Execution

When the system generates a command, it uses the data you entered. Once the command has been generated, the system performs the following steps:

1. Spawns a task naming the requested bound unit (the bound unit named by the generated command). Task spawning implies task creation, which consists of the allocation and initialization of any control structures and data areas required for task control.

2. Calls the Loader to load the requested bound unit.

3. Places a request for the execution of the bound unit against the created task. The Menu Processor enters the wait state to await completion of the requested task (command). At this point, the system leaves menu/form level, which can be returned to only by completion of command execution.

4. If the command is Enter Group Request, places a group request against an application task group. The Enter Group Request command terminates. The request is queued if there are other outstanding requests against the application task group from previous Enter Group Request commands.

5. Deletes the spawned task when the command terminates.

COMMAND ENVIRONMENT

The command environment is that environment in which you communicate with the Executive through command lines entered at a terminal or read from a command file.

When you use the menu environment, you make a choice from a menu and enter data into the fields of the form that is displayed in response to your menu selection. In the command environment, however, you directly enter a command line that contains all the necessary information in the form of arguments.

The description of the command-in, user-in, user-out, and error-out files given below also applies to the menu environment. In addition, the information under the heading "Command Format" applies equally to commands entered under the menu environment.

Command Processor

The Command Processor is the system software component that reads your command lines. After reading a command line, the Command Processor loads and executes a bound unit that fulfills the request represented by the command line.

The essential parts of the command environment are the Command Processor and the command-in file. Three other files are also involved with the command environment. These are the user-in file, the user-out file, and the error-out file. The four files associated with the Command Processor are also referred to as the standard I/O files.

Command-In File The command-in file is the file from which the command lines are read. It can be a terminal device, as in the case of an interactive user, or a command file residing on a disk, as in the case of a non-interactive user.

User-In File The user-in file is the file from which a command, during its execution, reads its own input. When a task group request has been processed, and as long as no alternate user-in file is specified as an argument in the subsequent command, the user-in file remains the same as the command-in file. At the termination of a command that names an alternate user-in file, the user-in file reverts to its initial assignment.

For example, the directives submitted to the line editor following the entry of the Editor command are submitted through the user-in file. No specific action is required on your part to activate or connect to the user-in file unless the directives are to be read from a previously created disk file. You simply invoke the line editor and begin entering line editor directives through the same terminal. The attaching of the terminal to the user-in file is invisible to you.

User-Out File

The user-out file is the file to which a task group normally writes its output. However, certain system components (for example, compilers) also write to list files (path.L) or to the output file defined in the -COUT argument of the command that invokes the component.

The user-out file is initially established by the -OUT argument of the Enter Group Request or Spawn Group command. Originally, it is the same device as the error-out file device. It can be reassigned to a file or another device by use of the File Out command or New User Out system service macrocall. Such a reassignment remains in effect for the task group until another reassignment occurs.

Again using the line editor as an example, any responses from the line editor, such as the printing of a line of the file being edited, are issued through the user-out file. As in the case of the user-in file, you need not perform any special action to attach your terminal to the user-out file. The only time such action would be required is if you wanted to direct the output from the command to some device other than the terminal.

Error-Out File

The error-out file is used by the system to communicate any error condition that may be detected during the interpretation of a command or its subsequent execution. Such a condition could be a missing command argument (reported by the Command Processor) or a file-not-found condition (reported by the invoked command).

The error-out file is originally the same as the original user-out file; it can be reassigned to a file or another device by use of the File Out command.

Command Level

When the system is in a state in which the Command Processor is capable of accepting a command from the command-in file, it is said to be at command level.

Achieving Command Level

You can achieve command level by creating or spawning a user task group whose lead task is the Command Processor. The Listener can also spawn task groups whose lead tasks are the Command Processor. The system is delivered with a Honeywell Bull supplied user task group ($H) whose lead task is the Command Processor.

The system indicates it is at command level by issuing a "ready" prompter message at your terminal. (This assumes that you have not disabled the ready message by issuing a Ready Off command. If Ready Off has been issued, the system comes to command level without informing you.) If you are working in the system task group ($S) at the operator terminal, no ready prompt message appears unless you issue an EC !CONSOLE command followed by a Ready On command.

Command Level Processing

When you enter a command, the system performs the following steps:

1. Spawns a task naming the requested bound unit (the command name). Task spawning implies task creation, which consists of the allocation and initialization of any control structures and data areas required for task control.
2. Calls the Loader to load the requested bound unit.
3. Places a request for the execution of the bound unit against the created task. The Command Processor enters the wait state to await completion of the requested task (command). At this point, the system leaves command level, which can be returned to only by completion of command execution or by pressing the BREAK key.
4. If the command is Enter Group Request, places a group request against an application task group. The Enter Group Request command terminates. The request is queued if there are other outstanding requests against the application task group from previous Enter Group Request commands.
5. Deletes the spawned task when the command terminates. Optionally issues a ready message to indicate a return to command level.

When executing a command function, you can return to command level in any of the following ways:

- Normal command termination. At normal termination of a command function, the task group returns to command level and awaits the entry of another command. (If your terminal is not monitored by the Listener, you should not enter the Bye command.)

- Command interruption. You can interrupt the execution of an invoked command by pressing the BREAK key (which may be labeled with BREAK or BRK) on the terminal. The system responds to this action with the break message **BREAK**. At this point, you can enter other commands or can enter the Start command to resume processing at the point of interruption. (Refer to the *Commands* manual for details.)

Command Format

A command line is a string of up to 252 ASCII characters in the form:

```
command_name [arg ...arg ];command_name [arg ...arg ] ...
                 1      n                   1      n
```

where command_name is the pathname of the bound unit that performs the command's function. Arguments, designated by arg, are described below.

Multiline Commands

A command line can span one or more physical lines. A command line is concatenated with the next line by ending it with an ampersand (&). A command line consisting of two or more concatenated lines can be canceled by entering a single ampersand on the next physical line.

Multicommand Lines

More than one command can be included in a command line by ending each command (except the last) with a semicolon (;). If any command in the command line is prematurely terminated (interrupt or error), the remaining commands are not processed. Note that only one operator command can appear on a line. (An operator command is a command that can be entered only by the person at the operator terminal.)

Arguments

An argument of a command is an individual item of data passed to the task of the named command. Some commands require no arguments; others accept one or more arguments as indicated in the syntax of each command description. The types of arguments used are positional and control.

Positional Argument

A positional argument is an argument whose position in the command line indicates to which variable the item of data is applied. The argument can occur in a command line immediately after the command name, or as the last argument following the control arguments, as in the List Names command.

Control Argument

A control argument is a keyword whose value specifies a command option. A keyword is a fixed-form character string preceded by a hyphen (for example, -ECL). It can be alone, as in -WAIT, or it can be followed by a value, as in -FROM xx.

Except for -ARG, or when the last argument of a command line is a positional argument, keywords of control arguments can be entered in any order in the line, following the initial positional arguments. The keyword -ARG must be the last argument of the command line. The arguments following -ARG are passed to the activated (application) task.

Parameters

Arguments are the user-selected items of data passed to a task. In the activated task, which is written in a generalized manner to handle any set of data passed to it, these data items are known as parameters. If the activated task expects positional parameters, the command line arguments passed to it must be in the same order as the task's positional parameters.

Spaces in Command Lines

Arguments in command lines are separated from each other by spaces. Unless otherwise indicated, a space in command line syntax represents one or more space characters, one or more horizontal tab characters, or a combination of space and horizontal tab characters. You can embed spaces within an argument by enclosing the argument in apostrophe (') or quotation mark (") characters. Note that a file name supplied in an argument can be shown to have a trailing space if the argument is bounded by quotation marks.

Protected Strings

Some characters (called reserved characters) have special significance to the Command Processor. If you want to use these characters without their special significance, you must protect them by placing them between protected string designators.

Reserved Characters

Special significance is attached to the following reserved characters:

- Space (blank)
- Horizontal tab
- Quotation mark (")
- Apostrophe (')
- Semicolon (;)
- Ampersand (&)
- Vertical bar (|)
- Left bracket and right bracket ([])

Designators

It is occasionally necessary to use a reserved character without its special meaning (for example, a blank could be used in a command argument). The protected string designators (apostrophe and quotation mark) are reserved for this purpose. Reserved characters within a protected string (one surrounded by protected string designators) are treated as ordinary characters. For example, in the argument:

```
    -ARG "ALPHA 2" ALPHA
```

the space in ALPHA 2 is treated as part of the name.

Unless the Uppercase command specified that uppercase was off, lowercase characters that are not protected by quotation marks are interpreted as upper case characters. For example, if uppercase is off, the string abcd will be passed to a command as ABCD and the string "abcd" will be passed as abcd.

Use of the protected string designators may also be required when the & character is followed by a number in a command-in file. If &1 does not represent a substitutable parameter, it must be written as &'1' or &"1" (not "&1"). Substitutable parameters are discussed in the *Commands* manual and in the *System User's Guide*.

Suppressing Designators

Since the protected string designators themselves are reserved characters, it may be desirable at times to suppress their special meaning. To do this, you must enclose the string containing the reserved character in quotation marks or apostrophes. If the reserved character to be protected is the same as the characters enclosing the string, it must be entered twice. For example, to pass the string A"B to a command, you can enter either of the following:

```
"A""B"
'A"B'
```

To pass the string A'B to a command, you can enter one of the following:

```
'A''B'
"A'B"
```

Active Strings and Active Functions

An active string is a part of a command and is evaluated (executed) immediately by the Command Processor. The resulting value is then substituted for the active string characters in the command line. Any command can be used within an active string. Commands explicitly designed to be used within active strings are called active functions. Active strings and active functions are delimited by left and right brackets.

An example of an active function is [EQUAL a b], which returns TRUE if a is equal to b and FALSE if a is not equal to b. Another example is [LHD], which returns the full pathname of your home directory. If you issued the command:

```
MENU_PR [LHD]>MENUCAT.EN -LC
```

the system would list all the menus in the menu catalog in your home directory.

Active functions can have arguments of their own, and active strings can be nested. For example, the TIME active function (which returns the current time as hour and minute) can be nested in the SUBSTR function (which returns a substring of characters beginning at a specified position and including a specified number of contiguous characters). If TIME returns 10:15, the active string:

```
[SUBSTR [TIME] 4 2]
```

returns the substring 15 (begin at fourth character and return a substring of two characters).

An active string can consist of any number of valid active functions separated by semicolons. The value of the active string is the concatenation of the values of the active functions. For example, if the active string [act_fnc1 x] returns the value TURN, and the active string [act_fnc2 yz] returns the value OUT, the active string:

```
    [act_fnc1 x; act_fnc2 yz]
```

returns the value TURNOUT.

Active strings and active functions are described in detail in the *Commands* manual.

Command Abbreviations

Abbreviation Processor

The abbreviation processor provides a way of relating a short, user-defined character string to another string of arbitrary length. Suppose, for example, that a program existed in a directory other than your working directory and had several entry points (called ALPHA, BETA, and OMEGA) that performed different functions. Entering the full pathname of the program and its entry point would require you to type an involved command line such as:

```
    >PROJECTA>SMITH>SUB_DIR>WIDGET?ALPHA
```

each time you wished to invoke function ALPHA in program WIDGET in the directory identified as >PROJECTA>SMITH>SUB_DIR. The abbreviation processor allows you to define a simple name such as ALPHA and equate it to the full command pathname. Similar abbreviations can be defined for functions BETA and OMEGA.

Abbreviation File

You can create and maintain your own standard sequential file of abbreviations for commonly used commands, control arguments, and pathnames. When the abbreviation processor has been activated by the Abbreviation command, it intercepts a command line, uses your abbreviation file to expand any abbreviations in the command line to their predefined character strings, and then passes the full command line to the command processor.

COMMAND ACCOUNTING

Command accounting is an optional facility that logs all user commands entered through the command language, menus, and the Process Command Line system service macrocall. Commands entered from system groups (those whose first group id character is $) are not logged.

The system records the command's elapsed time and resource usage as well as the group id and user id of the issuer. Refer to the *System Building and Administration* manual for information on requesting command accounting and obtaining command accounting reports.

COMMAND BEAMING

Command beaming allows you to execute commands in another computer system (node). (A task group capable of processing the commands you enter is automatically created in the remote node.) When you issue a Beam command, the system's remote file access facility (described in Section 2) reads your command-in and user-in files and sends the data to the node specified in the Beam command. The output generated at this node is written to your user-out and error-out files. All subsequent commands you issue will be executed at the specified node until you issue another Beam command to return to your node.

All memory space, processor time, and disk space required to execute the commands are distributed to the remote node.

Since command beaming allows you into another computer, you can enter commands to find out the status of users, applications, devices, and so forth on that node. You can queue requests, send messages to local users, and update the node's remote file catalog.

EC AND START_UP.EC FILES

The Command Processor is able to read commands from a source other than an interactive user terminal. One example is an Execute Command (EC) file that you construct through an editor. An EC file is a text file that contains command lines (for input to the Command Processor) and/or Execute command directives. An EC file is read by the Command Processor when:

- The Command Processor is invoked by an Execute command.
- A task group is activated with the Command Processor as its lead task and the EC file is specified as the task group's user-in file.

Absentee Task

When you enter a request to have a task run in absentee mode, you specify an EC file that is to be read by the Command Processor (refer to the *System User's Guide* for further details).

EC Files

An EC file might contain a series of commands that you execute on a frequent basis, such as commands to execute a set of application programs that run at the end of the month to summarize inventory, sales, and accounts receivable. EC files can range from simple to complex. An example of a simple EC file is:

```
ED -PT
FORTRANA AREA -LE
LINKER AREA -IN LNKDR
DPRINT AREA.M
AREA
```

This EC file is made up of commands that are most often used in developing a FORTRAN program called AREA. The ED command invokes the line editor, FORTRANA invokes the the FORTRAN compiler, LINKER invokes the Linker, DPRINT prints the link map, and AREA executes the program.

A more complex EC file uses active functions and substitutable parameters. The following file could be used to create, compile, and link any program. (The lines beginning with the & character are Execute command directives.)

```
& CREATE, COMPILE, AND LINK A &1 PROGRAM
&P BEGIN EDITOR SESSION
ED -PT
&P COMPILATION BEGINS
&1 &2
&IF [EQUAL [RETCODE] 0000] &THEN &ELSE &> ERROR1
&P LINKER SESSION BEGINS
&A
LINKER &2
LINK &2
QT
&D
&IF [EQUAL [RETCODE] 0000] &THEN &ELSE &> ERROR1
&P LINK COMPLETE
&> FINISH
&L ERROR1
&P ERROR ENCOUNTERED IN DEVELOPMENT SEQUENCE
&P EC TERMINATED
&Q
&L FINISH
&Q
```

Assuming that this file is named PROG_DEV.EC, you could execute it for a COBOL program development session by entering:

```
EC PROG_DEV COBOLM PAYROL
```

The pathname element PROG_DEV is substituted for all occurrences of &0 (none in this example), COBOLM is substituted for all occurrences of &1, and PAYROL is substituted for all occurrences of &2.

The EC command is described in the *Commands* manual; EC files are discussed in the *System User's Guide*.

START_UP.EC Files

A special application of EC files is their use at system initialization and at task group activation.

System START_UP.EC File

After configuration (after the CLM_USER file of configuration directives is executed), the system searches for a user-written command file named START_UP.EC in the system root directory. If this file is present, it is executed.

A typical system startup EC file might contain operator commands used to establish an application environment for the installation. An example of such a file is:

```
CBP BUFF1 -BUF 10 -CISZ 1024
CBP BUFF2 -BUF 5 -CISZ 512
CBP BUFF3 -BUF 5 -CISZ 256
CBP BUFF4 -BUF 20 -DIR
START_MAIL
EC >>GROUP$L
&Q
```

This startup EC file creates several buffer pools of various common sizes, activates the local mail/message facility, and activates Listener monitoring of terminals.

User START_UP.EC File

When a task group whose lead task is the Command Processor is activated, the Command Processor searches for an EC file named

```
home_directory>START_UP.EC
```

where home_directory is the pathname specified in the -HD argument of the user registration (or the -WD argument of the Spawn Group command). If such a file is present, the Command Processor executes it before performing any other action.

When a task group whose lead task is the Menu Processor is activated, the Menu Processor searches for an EC file named

```
home_directory>START_ONE.EC
```

and, if such a file is present, executes it before performing any other action.

The user startup EC file could contain commands to direct the execution of the tasks of the job and/or perform certain housekeeping tasks. An example of a user startup EC file is:

```
ST 2 -EFN PROGA
ST 3 -EFN PROGB
ST 4 -EFN PROGC
```

This user startup EC file causes three tasks to be activated.