Software Technical Bulletin
September 1987

Software Information Services

Part Number 812-8701-08
Issue 1987 – 8
September 1987

sun microsystems

Software Technical Bulletin
September 1987

Software Information Services

Part Number 812-8701-08
Issue 1987 – 8
September 1987

Software Technical Bulletins are distributed to customers with software/hardware or software only support contracts. Send comments or corrections to 'Software Technical Bulletins' at Sun Microsystems, Inc., 2550 Garcia Ave., M/S 2-312, Mountain View, CA 94043 or by electronic mail to *sun!stb-editor*. Customers who have technical questions about topics in the Bulletin should call Sun Customer Software Services AnswerLine at
800 USA-4-SUN.

Contents

1

NOTES & COMMENTS

1

NOTES & COMMENTS

Editor's Notes

Editor's Notes

The September editor's notes for the Software Technical Bulletin (STB) include the current Sun software products and release levels table, current customer service hotlines available in the United Kingdom and Europe, and a Browse program.

Current Sun Software Products and Release Levels Table

The September Software Technical Bulletin (STB) includes the current version table. The current release level is shown for each product.

Use this table along with STB articles that appear in one or two issues after a new current release is available for a particular product. You can then better determine what your software needs are, what functions are available in a new release, and whether the release you are using is down-level from the most current product release.

UK and Europe Hotlines

Look further into this Notes and Comments section for a listing of United Kingdom and European service hotlines. These phone lines are available for both software and hardware support questions.

The Hackers' Corner

Again, please note that such applications, scripts, or code are not offered as released Sun products, but as items of interest to enthusiasts wanting to try out something for themselves. They may not not work in all cases, and may not be compatible with future SunOS releases. Please consult your local shell script or programming expert regarding any application, script, or code problems.

Thanks.

The STB Editor

Sun Software Produce Releases

Current Software Sun Products and Release Levels

Product Name	Current Release
SunOS	3.4
Cross Compiler	1.0
SunLink BSC3270	4.0
SunLink Local 3270	4.0
SunLink SNA3270	4.0
SunLink IR	4.0
SunLink DDN	4.0
SunLink DNI	4.0
SunLink OSI	4.0
SunLink TE100	4.0
SunLink X.25	4.0
NeWS	1.0
Sun Common Lisp	2.0
Modula-2	1.0
SunAlis	2.1
SunGKS	2.0
SunINGRES	5.0
SunSimplify	1.0
SunUNIFY	2.0
TranScript	2.0
SunIPC	1.1
PC-NFS	2.0

Current Sun Software Products and Release Levels

The table appearing above contains a list of current Sun software products and their respective current release levels.

You will note that the Software Technical Bulletin (STB) contains articles from time to time that detail technical changes in a given software product's next available release.

Please contact your sales representative if you decide that you would like to update the release level of a Sun software product you already use, or wish to purchase another product. Use the table below to determine whether your release is the current release level.

This table appears monthly in the STB for your convenience.

European Hotlines

European Service Hotlines

Sun Customers in the United Kingdom and Europe have service hotlines available for both software and hardware support questions. The service hotlines are shown below.

United Kingdom	Camberley UK Headquarters	(44) 276 62111	
France	Central/Northern Regions	(33) 1 4630 2324	Paris HQ
	South West Region	(33) 6144 4477	Toulouse
	South East Region	(33) 7835 5141	Lyon
Germany	Munich Germany Headquarters	(49) 89 926 9000	
The Netherlands	Soest Netherlands Headquarters	(31) 2155 24888	

Errata

Errata

Two typographical errors occur in the June 1987 STB, on page 230, in the In Depth article entitled *ND Second Swap Space.*

In the second paragraph under the heading 'Step Two: Edit `/etc/nd.local`', please change `/etc.nd.local` to `/etc/nd.local` in two places.

2

ARTICLES

2 ARTICLES

Using USA-4-SUN

Using 800 USA-4-SUN

All Sun customers may call the **800 USA-4-SUN** phone line for assistance in the use of Sun software, hardware, and network products. This article explains what information you will need when you call, and how your call is routed to the service engineer who helps you. Your call will be routed to different support locations, depending on whether you have a support contract and on what type of product you are using that requires customer support.

When calling the **800 USA-4-SUN** number, you should always have the information listed below ready. If any of the information is not readily available, it may take longer to route your call properly.

- workstation model and serial number
- purchase order (PO) number (for those customers not holding support contracts)
- name
- company or organization name and address
- SunOS release number (See the June STB short subject, page 205, to find how to determine your SunOS release level.)
- problem description

Routing Your Calls

Many customers call after talking to their sales representatives. Others call 'cold'. In either case, you are prompted by a prerecorded message. It asks those not holding support contracts to have their PO number handy. The recording then asks you to dial a number, depending on the type of support needed. The current options are listed below.

Dial 1	for software support
Dial 2	for hardware support, including returning or exchanging parts
Dial 3	to schedule the installation of a new system
Dial 6	for telemarketing, to purchase customer service products or service contracts

After you select a number, a service dispatcher will ask you for the information listed above and for a brief description of your problem. The dispatcher then uses your problem description to route your call to a support engineer who specializes in the product that is the subject of your phone call.

The dispatcher logs your service call and will give you a service (SO) number that you may use as a reference to your call in future calls, mail, or email. Your call is now routed to specialists who answer calls for their particular subject matter area.

You can now expect an engineer to return your call that same day, or during the next normal working day.

SunIPC Logical Hard Disks

Creating a 30 Mbyte SunIPC Logical Hard Disk

Use the procedures shown in the *Sun IPC™ User's Guide*, part number 814-1002, chapter 4, 'Using Disks' to create logical hard disks up to 20 Mbytes in size.

Use the procedures contained in this article to create a 30 Mbyte SunIPC logical hard disk.

Background and Requirements

You may wish to increase the size of your SunIPC logical hard disk as your disk needs increase. Initially, your SunIPC logical hard disk occupies about 1 Mbyte of storage space. The name of this file(s) is `/usr/pctool/drive_C.pc0` through `/usr/pctool/drive_C.pc3`, depending on your having up to four SunIPC boards installed in your system. In this article the case of a single SunIPC board and file `/usr/pctool/drive_C.pc0` is considered.

The logical hard disk grows to approximately 10 Mbytes by default as users store more files or PC applications or both. The maximum disk size upper limit may be reset, allowing additional disk storage.

You *must have access* to a SunIPC floppy disk subsystem to change the SunIPC logical hard disk size. You will create a `boot`able floppy before beginning the procedures in this article. This is required since the existing drive C is destroyed when changing the logical hard disk size. Note that you *cannot* backup your logical hard disk to an NFS server since it is *not possible* to boot SunIPC from a network device.

Also note that it is best to change the logical hard disk size when you first receive the SunIPC board. Backup time at a later date may be greatly increased by your having many PC application programs stored on the logical disk.

Two Procedures

Use one of the two procedures shown in the following paragraphs, depending on whether you have an IBM AT Diagnostics diskette available. Use Procedure I if you have the disk, otherwise use Procedure II.

Procedure I: IBM AT Diagnostics Diskette Available

Use this procedure in the case that you have a copy of the *DOS User's Manual* and an IBM AT Diagnostics diskette.

1. Backup the SunIPC logical hard disk contents onto floppy disks. Use either the MS-DOS `copy` or `backup` command. See the *DOS User's Manual* for command definitions if needed. The logical disk, drive C, contains the MS-DOS, NFS, GWBASIC, and system utility files included with the SunIPC board, plus any user files.

   Note that additional backup procedures may be required, depending on your application programs. Some application programs create 'hidden' files that may not be copied unless you use a special backup procedure. This is part of some application programs' software protection schemes. Refer to your application program user manual for any special backup procedures.

2. Make a new system floppy disk. Insert a blank floppy disk in drive A. Move to directory `c:\msdos` and enter the command shown below.

   ```
   c:\msdos> format a:/s
   ```

   This command causes MS-DOS to copy the necessary system files from the SunIPC logical hard disk to the new system floppy disk.

3. Use the MS-DOS `copy` command to transfer the files listed below from drive C to the new system floppy disk in drive A. Note that you need to copy the `Restore.Com` file only if you used the `backup` command in step 1.

   ```
   COMMAND.COM
   FDISK.COM
   FORMAT.EXE
   RESTORE.COM
   ```

4. Remove the new system floppy disk from the SunIPC floppy disk drive A. Insert the IBM AT Diagnostics diskette and `reboot` the `PCTOOL`.

5. From the menu that appears, select option four, `setup`, and press <Return>.

6. When prompted, verify the correct date and time. Change the date and time as required.

7. When prompted with *The following options have been set:.... Are these options correct (Y/N)?*, press <N> and then press <Return>.

8. When prompted with *Are diskette drive types correct (Y/N)?*, press <Y> and then press <Return>. Do not change the floppy disk options.

9. When prompted with *Your fixed disk drive types are set to the following:.... Is this correct (Y/N)?*, press <N> and then press <Return>.

10. When prompted with *How many fixed disks are installed?*, press <1> and then press <Return>.

11. When prompted with *Enter fixed disk type (1-15) for fixed disk drive C.*, press <8> which signifies a 30 Mbyte hard disk.

12. Check the next screen to ensure that you have entered the correct disk type and then press <Y> if correct. Press <N> if incorrect and then repeat steps 10 through 12.

13. Do not change any subsequent options.

14. The final screen prompts you with the selected options and asks for verification that the options are correct. Check that the *Fixed Disks Drive C - Type* is type 8 for the 30 Mbyte hard disk. Also check that no other options were changed. Press <Y> if the options are correct and then press <Return>. Press <N> if the options are not correct, press <Return>, and then repeat steps 8 through 14.

15. Remove the IBM AT Diagnostics diskette from the SunIPC floppy disk drive A. Insert the new system floppy disk you created in steps 2 and 3.

16. Press <Return> or use the mouse to reset the `PCTOOL`.

17. `Reboot` the SunIPC from the new system floppy disk. You must `reboot` since you cannot change the disk size at the same time you are running SunIPC from that disk.

18. Run the MS-DOS `fdisk` utility from the new system floppy disk. This modifies the existing drive C to enlarge the logical hard disk.

19. Refer to the `fdisk` utility documentation in the *DOS User's Manual.*

20. First, select the third menu item, *Delete DOS Partition.* Second, select the first menu item, *Create DOS Partition.* Third, select the second menu item, *Changing the Active Partition.*

21. The `fdisk` utility forces you to `reboot` SunIPC again from the new system floppy disk once the utility has finished changing the logical hard disk partition.

22. Format the logical hard disk by entering the command shown below.

```
> format c:/s/v
```

23. Copy the files from the backup floppy disk(s) you created in step 1 onto the new, 30 Mbyte SunIPC logical hard disk. Use either the MS-DOS `copy` or the `restore` command, depending on whether you used the `copy` or the `restore` command to create the backup floppy disk(s).

24. `Reboot` the SunIPC from the logical hard disk on drive C.

The procedure is completed. You are now ready to use the SunIPC as usual.

Procedure II: IBM AT Diagnostics Diskette **not** Available

Use this procedure in the case that you *do not* have a copy of the *DOS User's Manual* and an IBM AT Diagnostics diskette. You will use the MS-DOS `debug` command to enlarge the size of the SunIPC logical hard disk.

Again, note that up to four SunIPC logical hard disks may be installed on your system. They use files `/usr/pctool/cmos_ram.pc0` through `/usr/pctool/cmos_ram.pc3`, respectively. In this article the case of a single SunIPC board and file `/usr/pctool/cmos_ram.pc0` is considered.

1. Backup the SunIPC logical hard disk contents onto floppy disks. Use either the MS-DOS `copy` or `backup` command. See the *DOS User's Manual* for command definitions if needed. The logical disk, drive C, contains the MS-DOS, NFS, GWBASIC, and system utility files included with the SunIPC board, plus any user files.

   Note that additional backup procedures may be required, depending on your application programs. Some application programs create 'hidden' files that may not be copied unless you use a special backup procedure. This is part of some application programs' software protection schemes. Refer to your application program user manual for any special backup procedures.

2. Make a new system floppy disk. Insert a blank floppy disk in drive A. Move to directory `c:\msdos` and enter the command shown below.

   ```
   c:\msdos> format a:/s
   ```

   This command causes MS-DOS to copy the necessary system files from the SunIPC logical hard disk to the new system floppy disk.

3. Use the MS-DOS `copy` command to transfer the files listed below from drive C to the new system floppy disk in drive A. Note that you need to copy the `Restore.Com` file only if you used the `backup` command in step 1.

   ```
   COMMAND.COM
   FDISK.COM
   FORMAT.EXE
   RESTORE.COM
   ```

4. From a UNIX window, copy file `/usr/pctool/cmos_ram.pc0` to a file named `cmos-tmp` in a directory that is both accessible and mountable via PC-NFS.

5. From a PCTOOL or a PC running PC-NFS on your network, continue with this procedure and perform the following steps.

6. Use the PC-NFS `NET USE` command to mount the UNIX directory containing the `cmos-tmp` file you made in step 4. An example is shown below.

   ```
   NET USE <?>: \\<host>\<dir>...
   ```

7. Change the current hard disk to the PC-NFS volume by issuing a `?:` where `?:` is the drive designation you used in the `NET USE` command example shown in step 6.

8. Type the DOS command `debug cmos-tmp` and then press <Return>.

9. You now see the `debug` prompt, a dash, on the left side of the screen. The next 16 steps (steps 10 through 25) are done from the `debug` prompt. Note that <sp> signifies typing a space using the space bar, and <Return> signifies pressing the <Return> key. Type each command exactly as shown in steps 10 through 25.

10. e <sp> 100 <Return>

11. 26 <Return>

12. e <sp> 102 <Return>

13. 16 <Return>

14. e <sp> 112 <Return>

15. 80 <Return>

16. e <sp> 114 <Return>

17. 33 <Return>

18. e <sp> 12F <Return>

19. 55 <Return>

20. e <sp> 142 <Return>

21. 45 <Return>

22. e <sp> 143 <Return>

23. 4A <Return>

24. w <Return>

25. q <Return>

26. Type the DOS command `debug cmos-tmp` and then press <Return>. You will again see the `debug` prompt, a dash.

27. d <sp> cs:100 <sp> L44 <Return>

28. Check that the screen obtained from step 27 contains the new values you entered in steps 10 through 23. A sample screen is shown below with the actual changes highlighted with asterisks (**).

```
-d cs:100 L44

33CC:0100  26 00 16 00 15 00 06 04-03 87 26 02 50 80 00 00   &.........&.P...
           **    **

33CC:0110  20 00 80 00 33 80 02 00-00 00 00 00 00 00 00 00    ...3...........
                 **    **

33CC:0120  00 00 00 00 00 00 00 00-00 00 00 00 00 00 01 55   ...............U
                                                        **

33CC:0130  00 00 19 80 00 00 00 00-00 00 00 00 00 00 00 00   ................


33CC:0140  20 4C 45 4A
                 ** **
```

29. If your screen obtained from step 27 matches the screen shown above, go to step 32, skipping steps 30 and 31.

30. If your screen obtained from step 27 *does not match* the screen shown above, from the `debug` dash prompt, type the `debug` command `q` and then press <Return>.

31. Type the MS-DOS command `del cmos-tmp` and then press <Return>. Go to step 4, and repeat this procedure by repeating steps 4 and 5. Then *skip* step 6, and repeat steps 7 through 29.

32. From the `debug` dash prompt, type the `debug` command `q` and then press <Return>.

33. If steps 6 through 32 were issued from a `PCTOOL`, use the right mouse button to 'quit' the `PCTOOL`.

34. Begin working from a UNIX window on a Sun workstation.

35. Copy the `cmos-tmp` file edited in this procedure to file `/usr/pctool/cmos_ram.pc0`. Note again that this procedure assumes that only one SunIPC logical hard disk is on your system. Up to four IPC boards may be installed using files `/usr/pctool/cmos_ram.pc0` through `/usr/pctool/cmos_ram.pc3`, respectively.

36. Insert the new system floppy disk you created in steps 1 through 3 into drive A.

37. Open a SunIPC window by entering `pctool` and pressing <Return> which boots from the new system floppy disk.

38. Run the MS-DOS `fdisk` utility from the new system floppy disk. This modifies the existing drive C to enlarge the logical hard disk.

39. Refer to the `fdisk` utility documentation in the *DOS User's Manual.*

40. First, select the third menu item, *Delete DOS Partition.* Second, select the first menu item, *Create DOS Partition.* Third, select the second menu item, *Changing the Active Partition.*

41. The `fdisk` utility forces you to `reboot` SunIPC again from the new system floppy disk once the utility has finished changing the logical hard disk partition.

42. Format the logical hard disk by entering the command shown below.

```
> format c:/s/v
```

43. Copy the files from the backup floppy disk(s) you created in step 1 onto the new, 30 Mbyte SunIPC logical hard disk. Use either the MS-DOS `copy` or the `restore` command, depending on whether you used the `copy` or the `restore` command to create the backup floppy disk(s).

44. `Reboot` the SunIPC from the logical hard disk on drive C.

For Further Information

Regardless of whether you used procedure I or II, the resulting file `cmos_ram.pc0` (for one SunIPC board) will now expand to a maximum of 30 Mbytes.

See the *SunIPC™ User's Guide*, part number 814-1002, chapter 4, 'Using Disks' for a discussion that includes the additional topics appearing below.

- differences between logical and physical hard disks
- creating a board-independent `autoexec.bat` file

- installing PC applications
- using disk drives D through V to work with NFS files
- reducing the logical hard disk size
- changing the logical hard disk location

Client UNIX Status

Determining UNIX Status on a Client

Local network-related commands, such as `netstat` and `etherfind`, are not suitable for remote use. Therefore, when developing a program to monitor the status of UNIX of a given file server's clients, provide this information using a Remote Procedure Call (RPC) to specific software. Several methods can be used.

In addition to `traffic(1C)` with `ether(8C)`, as described in the *Commands Reference Manual*, part number 800-1295-04, the following can be used:

- `ping`
- `rpc.statd` (daemon)
- `rpc.etherd` (daemon)
- Portmap `pmap_rmtcall`

The information returned by each of these methods varies the meaning of the status of the remote machine and network or both.

`ping`

`ping` refers to the `imc echo` packet, which reports whether or not the kernel has gone through the initialization process enough to initialize the Inter-Process (IP) code. `ping` keeps trying to send the packet and to report the reply until either a specified timeout period has elapsed, or a reply is received. The default timeout period is 20 seconds.

Thus, `ping` can be used to determine whether a remote machine is halted or powered on/off. However, it cannot be used to determine whether or not a user can remotely log in to the machine, using `rlogin`. It cannot be used to determine whether or not the user can send or receive Network File System (NFS) requests or responses to the remote machine.

If the remote machine is operating in multi-user mode, the user is usually able to `rlogin`, as well as send and receive NFS requests responses. If `rlogin` is unsuccessful, the NFS server can be `ping`ed by calling the null procedure of the NFS server, similar to the function of `rcpinfo`.

Additional information on the use of `ping` and `rpcinfo` is included in the *Commands Reference Manual*, part number 800-1295-04.

rpc.etherd

rpc.etherd is a server which puts the appropriate interface into promiscuous mode, and keeps summary statistics of all packets received on that interface. It reports whether or not the host is sending or receiving packets, because much network traffic on clients consists of Network Disk (ND) and NFS requests generated in response to user programs running on those clients. rpc.etherd is used because the kernel does not specifically send out any status packets. The user must be root to use rpc.etherd.

rpc.etherd is useful only on local networks. For additional information, refer to the etherd(8C) description in the *Commands Reference Manual*, part number 800-1295-04, and the ether(3R) description in the *UNIX Interface Overview* manual, part number 800-1341-02.

rpc.rstatd

rpc.rstatd is used for obtaining performance statistics from the kernel, and are graphically displayed by the perfmeter. rpc.rstatd reports whether or not the rstat daemon is running. If the rstat daemon is running, this usually indicates that the remote machine is operating in multi-user mode. This also indicates that the network is functioning and can successfully respond to the user.

Additional information on the use of rpc.rstatd is included in the *Commands Reference Manual*, part number 800-1295-04.

pmap_rmtcall

The problem described in using rpc.rstatd, above, can be avoided by using pmap_rmtcall to determine whether or not the rstat daemon is running, and so indicate that the remote machine is operating in multi-user mode, as well as proper functioning of the network. pmap_rmtcall is a user interface to the portmap service, which instructs portmap residing on the host at the IP address *addr to make an RPC call on the user's behalf to an RPC procedure on that host.

One consideration to keep in mind is that pmap_rmtcall cannot notify the user if the network is having problems. For example, the following network status messages, normally appearing in the user's console screen as well as in the /usr/adm/messages file, are not returned by pmap_rmtcall.

```
ie0: no carrier
ie0: Ethernet jammed
```

Additional information on the use of pmap_rmtcall is included in the manual *Networking on the Sun Workstation*, part number 800-1324-03.

A Simple ping Script

The following ping script can also be used to determine the status of clients. This script is especially useful within smaller networks where gateways are not involved. Keep in mind that this script does not check for ypbind, nor does it allow for the user to specify the net number and timeout as optional arguments.

Note that the timeout on the call to `ping` is set at 15 seconds; you may prefer a shorter or longer timeout period.

```
#!/bin/csh -f
foreach host ( `ypcat hosts | grep 192.9.201 | awk '{print $2}'` )

  /usr/etc/ping $host  15>/dev/null
  if ($status == 0) then                #if host is up
        echo $host
  endif
 end
```

Due to a limitation with `ping`, this script should not be run in two or more windows simultaneously.

Disk/Controller Combinations

Identifying Controller and Disk Configurations

It is often necessary for users to determine the controller/disk configuration of their Sun hardware. Use the following guidelines to determine the existing hardware configuration on different systems.

Deskside Pedestal Label Information

Deskside pedestals have the following information included on a label located on the front cover of the pedestal. This label is visible after removing the gray faceplate.

DISK DRIVE CONFIGURATION

DRIVE	0	1
FUJITSU	☐	☐
VERTEX	☐	☐
MICROPOLIS	☐	☐
OTHER	☐	☐
OTHER	______________	

The appropriate disk(s) contained in the pedestal will be marked on this label.

The pedestal label does not specify the disk information listed below.

- Disk interface type -- Small Computer System Interface (SCSI), or Storage Module Disk (SMD)
- Disk capacity
- Disk model designation

The disk interface type can be ascertained as shown below.

- Disks contained within the CPU pedestal are SCSI-type disks.
- Disks within expansion pedestals are SMD-type disks.

The disk capacity and model designation or both can be usually ascertained from the following information.

Sun-3 'Shoebox' Disk Subsystem Label Information

Newer Sun-3 disk subsystems (commonly referred to as 'shoeboxes') have a small label affixed to the rear. For 71MB disk subsystems, the label appears as shown below.

DISK DRIVE CONFIGURATION

FUJITSU ☐ MICROPOLIS 1325 ☐

For 141MB disk subsystems, the label appears as shown below.

DISK CONFIG

☐ MICROPOLIS 1355
☐ TOSHIBA MK156FA

All Sun-3 disk subsystems utilize the SCSI interface.

Sun-3 System Controller and Disk Combinations

The following lists the controller and disk combinations used in Sun-3 systems.

Sun-3 'Shoebox' Disk Subsystems:	
71MB	Adaptec controller. Disks are primarily Micropolis 1325 and Fujitsu M2243AS.
141MB	Emulex controller. Disks are Micropolis 1355 and Toshiba MK156F.
Sun-3/160 with SCSI Disk(s) in the CPU Pedestal	
71 MB	Adaptec controller. Disks are primarily Micropolis 1325 and Fujitsu M2243AS.
141MB	Emulex controller. Disks are Micropolis 1355 and Toshiba MK156F.
Sun-3/160 and 3/260 with SMD Disks in an Expansion Pedestal:	
280MB	Xylogics 451 controller. Disk is the Fujitsu M2333.
Rack-mount SMD Disks:	
575MB	Xylogics 451 controller. Disk is the Fujitsu M2361 Eagle XT (also known as the 'Super Eagle').

Sun-2 System Controller and Disk Combinations

The following lists the controller and disk combinations used in Sun-2 systems.

Sun-2 'Shoebox' Disk Subsystems:	
71MB	Adaptec controller. Disks are primarily Micropolis 1325 and Fujitsu M2243AS.
100U with SMD Disk(s):	
'84MB'	Xylogics 450 controller. Fujitsu M2312K disk. This combination is also referred to as a 'FAT' box, for Fujitsu-disk and Archive Tape.
Sun-2/120 with SCSI Disk(s) in the CPU Pedestal:	
42MB	Adaptec controller. Disks are Micropolis 1325 and Maxtor XT-1050.
71MB	Adaptec controller. Disks are primarily Micropolis 1325 and Fujitsu M2243AS.
Sun-2/130 and Sun-2/160 with SCSI Disk(s) in the CPU Pedestal:	
71MB	Adaptec controller. Disks are primarily Micropolis 1325 and Fujitsu M2243AS.
Sun-2/120, Sun-2/130 and Sun-2/160 with SMD Disk(s) in an Expansion Pedestal:	
130MB	Xylogics 450 controller. Disk is the Fujitsu M2322.
Rack-Mount SMD Disks:	
'169MB'	Xylogics 450 controller. Disk is the Fujitsu M2284. This is only found in 150U and Sun-2/170 systems.
380MB	Xylogics 450 controller. Disk is the Fujitsu M2351 Eagle.

Read This First Purpose

Using the *Read This First* (RTF) Document

This article contains a discussion of the purpose and use of the *Read This First* (RTF) document provided with all Sun Microsystems software.

The primary purpose of the RTF is to provide the user with current, pertinent information about the corresponding software product. This includes installation considerations of importance to system administrators when installing a new product or upgrading an existing product. Additionally, details are provided of new or changed features of importance to product users.

Read the RTF thoroughly before beginning the installation or upgrade since much of this information should be kept in mind at that time.

The RTF Format

The format of the RTF is designed to include the items listed below.

- Software compatibility with Sun system hardware and operating system release levels
- Environmental requirements, such as physical space and minimum swap space needed for proper operation
- Product or release anomalies or both
- How to get help

The RTF is the designated document to include information describing installation and usage problems encountered during the final testing of the product. These descriptions usually include workaround methods. The RTF also describes any errors in the product documentation, as well as the revised form, reflecting the current state of the product.

Copying and Distributing the RTF to Other Users

In some locations, the individual responsible for installing product software does not actually use the product. In these situations, the person installing the software will want to have copies of the RTF made for internal distribution to the Sun Workstation end-users. This ensures that information affecting product use is provided to developers and users of the corresponding product and dependent applications.

The primary contact person or department may duplicate the 'master' RTF copy for all Sun Workstation end-users, as well as to those who need the information. So long as the copies are duplicated and routed internally to employees working for a company having the product license, there are no copyright infringement problems.

3

STB SHORT SUBJECTS

3

STB SHORT SUBJECTS

lockd and Dumping Core

Running /etc/rpc.lockd and /etc/rpc.statd Concurrently

Customers have observed a problem with /etc/rpc.lockd which fails in the case that /etc/rpc.statd is not running at the same time.

The network lock daemon (/etc/rpc.lockd) fails if the network status monitor (/etc/rpc.statd) is not also running. /etc/rpc.lockd enters a loop trying to contact /etc/rpc.statd and allocates memory which it never frees. This eventually causes /etc/rpc.lockd to terminate without any message. This may also affect other processes since swap space is used up by /etc/rpc.statd.

The Workaround

This problem has been reported as bug ID number 1004739 and will be fixed in a future SunOS release. The workaround is to ensure that both /etc/rpc.statd and /etc/rpc.lockd are running. The default startup is for both /etc/rpc.statd and /etc/rpc.lockd to be started. Generally, you should not expect to see this problem.

tty and Terminal Displays

Virtual tty Lines and Terminal Display Problems

Terminal displays may become garbled when someone logs onto a virtual tty line (/dev/ttyp?), usually via rlogin(1), and finds processes running from the *previous* user of the virtual tty line.

Such problems have been observed while running SunOS release 3.2. They interfere with proper terminal displays when using vi, for one example. Merely killing the 'left-over' processes does not correct the problem.

The Problem Defined and Solved

This problem is caused by some terminal size attributes being reset only when the last process closes the terminal. If someone logs in and kills all of the 'left-over' processes, the tty line is not reset and the new processes continue to use the old settings.

On SunOS release 3.2 and later systems, add the following code to the .login file to correct the problem. Note that this code implies /bin/csh.

```
if ($term != "sun") then
    stty everything |& fgrep -s columns
    @ setscreensize != $status
    if ($setscreensize) stty rows 0 cols 0
endif
```

Optimizing Read Times

tunefs(8) to Optimize Read Times

In case you find your disk read and write times a little slow for your application, consider resetting a default value for *tunefs(8)* to reduce overall read and write time.

The Default Value

You can use *tunefs(8)* to change certain dynamic parameters in the superblock which the kernel uses when laying out the file system.

The `rotdelay` flag `-d` specifies the time expected to service a transfer completion interrupt and to initiate a new transfer on the same disk. This flag is used to decide how much rotational spacing to place between successive blocks in a file.

The `rotdelay` default value is set to 4 msec. This optimizes the read time at the expense of the write time. In one case, this default value resulted in 30 seconds to write 10 Mbytes, and only 12 seconds to read the 10 Mbytes. This default setting resulted in an overall read and write rate of 341 kbytes/second.

Resetting the Default Value

By increasing the `rotdelay` parameter to 5 msec, the write time is reduced and the read time is increased. Both would then be about 17 seconds, resulting in a reduced overall read and write rate of 602 kbytes/second. Note that these transfer rates are based on the number of *unformatted* bits per second that pass under the read head. Your rates will vary.

Also note that *tunefs(8)* does not report a current default setting unless you try to change it. You might have a piece of paper handy to write down your old values before trying any experiments.

The `-o` Optimization Preference Option

You may find an undocumented option, `-o`, useful. This option sets the optimization preference to either 'time' to minimize allocation time or 'space' to minimize disk fragmentation and disk space used.

SunAlis 1.0 Support Ends

Discontinuation of Support for SunAlis Release 1.0

The short subject describes the plan to discontinue SunAlis Release 1.0 support.

New features, functions, and performance enhancements included in SunAlis Release 2.0 provide faster operation and greater product reliability than SunAlis Release 1.0. For these reasons, Sun strongly recommends that SunAlis customers upgrade from Release 1.0 to Release 2.0.

SunAlis Software Upgrade Program

SunAlis Release 2.0 has been automatically shipped free of charge to all Sun customers holding SunAlis support contracts. In addition, Sun is offering SunAlis Release 2.0 free of charge to SunAlis licensees without support contracts. This free upgrade offer to customers not holding SunAlis support contracts is a one-time offer. In the future, customers without SunAlis support contracts will be charged for upgrade releases of the product.

The following information is required to obtain the Release 2.0 upgrade.

- Your company name, contact name, address, and phone number
- The original sales order number for SunAlis Release 1.0
- The date of purchase
- The machine type(s) running SunAlis Release 1.0

Mail this information to the address below.

Ms. Carol Adams
SunAlis Marketing Manager
Sun Microsystems, Inc.
M/S A4-40
2550 Garcia Avenue
Mountain View, CA 94043

Discontinuation of Telephone Support

Effective November 1, 1987, Sun will not provide telephone support for SunAlis Release 1.0, since improvements in Release 2.0 correct most of the problems reported with Release 1.0. Therefore, if you are using SunAlis Release 1.0 and encounter any problems with the product, please upgrade to Release 2.0 to verify the existence of the problem with Release 2.0 before calling for assistance.

IN DEPTH

4

IN DEPTH

SunView 2: A New Platform

An Introduction to SunView 2

This in-depth article gives a general overview of Sun Microsystem's plans to port the SunView user interface toolkit to a new window system platform. You will find a summary of the changes this new platform will introduce to SunView's *programmatic* interface. This article provides an early direction for application developers working with the current SunView.

Overview

The SunView graphical, window-based environment has become familiar to both end-users and application developers over the past year. Sun Microsystems has a strong commitment towards maintaining SunView as a stable basis for continuing development in the future. This commitment is reflected in the decision to move the SunView[2] environment from its current SunWindows base to the newly announced X.11/NeWS platform. The new version of SunView will be called SunView 2.

Audience

This and future SunView 2 compatibility documents are intended for use by the experienced SunView application developer who is familiar with the current SunView and its underlying platform, SunWindows. Application developers new to SunView should read the *SunView Programmer's Guide*, part number 800-1345, prior to investigating the changes described in this article.

SunView and SunView 2 Differences

Attribute names do not change, the current event structure is maintained, and `Pixfont` continues to be supported.

SunView 2

SunView 2 is a new version of the current SunView user interface toolkit designed to run on the X.11/NeWS window system. It provides a migration path for moving existing SunView applications onto X.11/NeWS, and delivers network flexibility for SunView 2 applications.

[2] In this document, 'SunView' without a '2' after it refers to the current SunView product, which first shipped with SunOS release 3.0. SunView has been enhanced and included in each release since SunOS release 3.0.

When?

SunView 2 on the X.11/NeWS platform will become available on Sun workstations in the spring of 1988. SunView 2 will be the basis for future SunView enhancements. However, in addition to SunView 2, SunView on SunWindows will continue to be included with Sun workstations, in order to allow Sun customers to make a graceful transition from the earlier to the later version.

Network SunView

X.11/NeWS is a *server*-based window system. The display manager, or window server, is a single, user-level process on the machine with the physical display. Applications ask the server to draw on the screen and to notify them of user-input events.

SunView 2 applications will inherit many of the benefits provided by a server-based window system. They will be able to run on one machine and display their output in windows on another machine anywhere on the network. They will also be able to run on different X.11/NeWS servers using a variety of display hardware.

Why Are There Changes?

Moving from SunWindows to X.11/NeWS involves changing from a kernel-based to a server-based architecture, and from a machine-specific to a portable platform.[3] The programmatic interface to SunView must change -- the underlying system is fundamentally different. For example, in a server-based system, window applications cannot access the display themselves, since the display may reside on another machine. They must ask the window server to draw on the display for them.

The *SunView 2 Changes* paragraphs that follow include a discussion of the planned changes and likely incompatibilities between SunView and SunView 2.

Future window-system technology will be server-based. Sun will give your applications the ability to run applications across many machine types on a heterogeneous network. This will be accomplished by developing a new version of an existing toolkit. This will be a full-featured user interface which will use the new generation of window servers. By migrating SunView onto the X.11/NeWS platform, Sun lets you move your existing SunView applications onto a state-of-the-art window system. The applications which are included with SunView (`textedit`, `cmdtool`, `mailtool`, `perfmeter`, `dbxtool`, and so forth) will continue to provide both users and developers a productive, integrated working environment.

SunView 2 Changes

Compatibility with the current SunView is the primary goal of SunView 2. Compatibility has been preserved except where change was necessary to make SunView work well in the X.11/NeWS environment.

[3] See the *Technical Overview*, part number 800-1498-05, for a discussion and comparison of different window system technologies.

Overview

While SunView 2 is not completely compatible with the current SunView, large areas of the programmatic interface will remain unchanged. Much of the *SunView Programmer's Guide* will remain unchanged as noted below.

- The structure of applications need not change.

  Programs will look the same as before -- first object creation, then installation of notify procedures, and then `window_main_loop()` to begin processing.

- The philosophy and organization of SunView does not change.

  The same basic window types (frames, canvases, panels, text subwindows, and tty subwindows) and objects (menus, icons, cursors and scrollbars) will carry over with most of the same attributes.

Most of the changes that have been made result from moving the SunView interface to a server-based window system. These changes are summarized below.

- Windows are no longer pseudo-devices (with FDs) that you open, since the underlying window system is not kernel-based.
- You no longer have access to some data structures, since they now reside in the server.
- You cannot access the screen directly, since it is controlled by the server. Note that it may even be on another machine.
- SunView 2 does not include many of the routines from SunOS releases prior to release 3.0. Such routines predate SunView. Further, SunView 2 does not include many of the lowest-level routines that support the current SunView. Functionality at this level is the responsibility of the X.11/NeWS server.

Compatibility with the Current SunView

SunView 2 is highly compatible with the current SunView at the higher level of the SunView packages. The higher-level packages in SunView are implemented in SunView 2. Such packages include the window types and objects mentioned above, and the notifier.

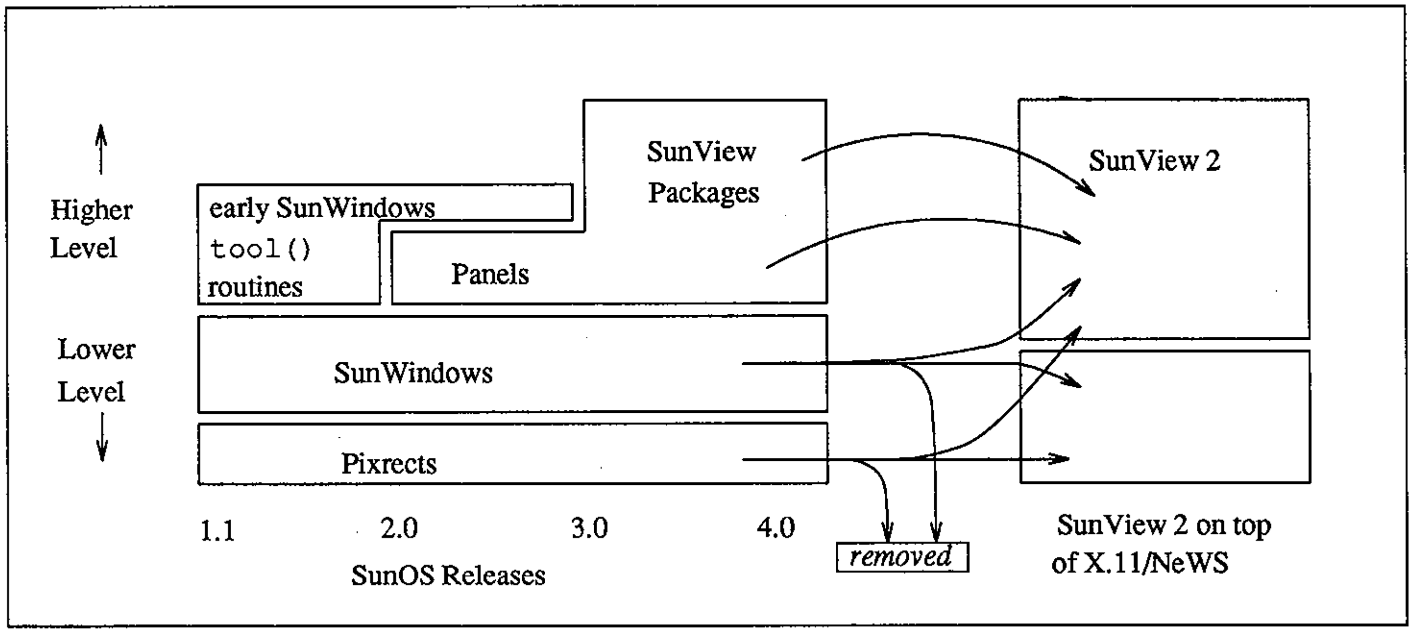
Some Compatibility with Lower Levels

The current SunView is based on the SunWindows kernel-based window system, and the Pixrect drawing library. Some of the lower-level SunWindows and Pixrect routines are implemented by the X.11/NeWS server, some by SunView 2, and some are not supported. The programmatic interface to directly access X.11/NeWS functionality from SunView 2 is being designed at this time.

Incompatibility with Pre-SunView Routines

Many pre-SunView window routines will no longer be supported. These are routines documented in the outdated *SunWindows Reference Manual* from SunOS releases 1.X; and the *Programmer's Reference Manual for SunWindows*, part number 800-1167, for release 2.X. In particular, compatibility does not extend to low-level SunWindows features that predate SunView such as `SIGWINCH`, `struct tool`, and `struct toolio`.

See the diagram shown below for the relationships among SunView, SunView 2 and SunOS releases.



Relationship of SunView, SunView 2, and SunOS Releases

Differences and Incompatibilities

The following paragraphs describe specific areas of incompatibility between SunView 2 and the current SunView. Packages not mentioned here (panels, menus, and the like) are generally fully compatible, although other areas of incompatibility may arise as implementation progresses.

Windows

There is no access to windows by their `WIN_FD`s, since windows are not devices in a server-based window system. One advantage is that the UNIX limitation on the number of file descriptors per process no longer restricts the number of windows in an application.

Some SunWindows routines that use window FDs will not be supported. In SunView 2, windows have an opaque window *ID* in place of an FD-number. For compatibility, the `WIN_FD` attribute will retrieve the window ID. Wherever possible, routines that are currently passed a window FD will work with this window ID. See the *SunView System Programmer's Guide*, part number 800-1342, for such routines.

Cursor and Icon `structs`

You will be able to create cursors and icons dynamically only. You will not be able to create them statically using the `#DEFINE_CURSOR_FROM_IMAGE` and `#DEFINE_ICON_FROM_IMAGE` routines.

You will not be able to access the fields in the `cursor` and `icon` structs.

⇒ You should convert access to these structs into `icon/cursor create()`, `set()`, and `get()` calls now.

Cursors

The X.11/NeWS server supports a mask-type cursor. Therefore, not all cursor RasterOp logical operations can be provided by the X.11/N server.

Crosshairs are not supported by the X.11/NeWS server.

Stacking Menus

Old-style, stacking menus from SunOS release 1.1 will not be converted to SunView 2. Since SunOS release 3.0, Sun has provided a full-featured, *walking menu* package, with much greater functionality. In SunOS release 4.0, users will get the new-style menus by default. Note that the old-style menus will continue to be available, for downward compatibility.

⇒ You should convert to the new menu package as soon as possible to save conversion effort when SunView 2 becomes available.

`menu_prompt()`

Support for the old SunWindows `menu_prompt()` routine will be removed in SunView 2. Sun is introducing the *alerts* package in SunOS release 4.0 as a replacement for `menu_prompt()`. Alerts provide improved functionality compared with `menu_prompt()`. As with old-style menus, `menu_prompt()` will continue to be available, again for downward compatibility.

⇒ When SunOS release 4.0 becomes available, you should convert all uses of `menu_prompt()` to the new alerts package.

Input

The X.11/NeWS server does not support SunView's current *click-to-type* model, since it does not have separate pick and keyboard input masks.

Pixwins

The `pixwin struct` is not included in SunView 2. Pixwins in SunView 2 will be strictly opaque objects. You will not be able to access their fields.

However, SunView 2 has the same imaging model as in the current SunView, and most `pw_*()` routines will be supported. Some new window attributes will be available to manipulate pixwins.

SunView 2 will not support pixwin regions since the `pw_region()` will no longer exist. You will create multiple windows instead.

Other unsupported pixwin calls include routines for locking, batching, and double-buffering. Functionality at this level is the responsibility of the X.11/NeWS server.

In SunView 2, there is no pixrect associated with a pixwin that you can use to draw directly on the screen. Pixwins are not based on pixrects in SunView 2. They are an interface to drawing routines implemented on the server.

Pixrects

You will not be able to use `pr_open(/dev/fb)` to open a pixrect which represents a remote screen. Also, you will not be able to use pixrect calls to draw on the X.11/NeWS server's screen.[4] Drawing at the pixrect level usually implies 'going around' the window system anyway.

However, the Pixrect package will remain unchanged. You can continue to build and manipulate pixrects as you do now. Pixrects can be created from files produced by `iconedit`, just as before, and the current rasterfile formats will continue to be supported. A similar interface will be provided for *remote pixrects* -- pixrects whose bit images reside in the X.11/NeWS server, not in your program.

Pixwin routines that take one or more pixrect arguments such as `pw_rop()`, `pw_replrop()`, and `pw_batchrop()` will only be able to use memory pixrects as arguments.

Fonts

You can load only the fonts that are available to the X.11/NeWS server. However, after loading a font, you can continue to access the bits in its glyphs since the `Pixfont struct` remains unchanged.

Repaint

You can ask the server to make a window retained. However, there is no guarantee that it will have the resources to retain all the pixels of the window in memory.

⇒ Applications that use canvases to draw on must be prepared to repaint themselves. This has always been recommended in the current SunView, and now in SunView 2 as well.

Different Internals

The X.11/NeWS server has a different architecture than kernel-based SunWindows. Thus most of the low-level routines giving clients access to the internals of the current window system do not apply to the new platform. The following listed chapters of the *SunView System Programmer's Guide* cover the internals of SunView on SunWindows. Much of their information, therefore, will not apply to SunView 2.

- *SunView System Model*
- *The Agent and Tiles*
- *Windows*
- *Desktops*
- *Workstations*

[4] If the screen is on the same machine, client programs not using SunView 2 can continue to use the raw Pixrect package to draw on the screen as before.

- *Advanced Imaging*
- *Window Management*

Similar facilities are provided by the X.11/NeWS server. Similar functionality will be available for clients who access the window system at this low level.

Virtual User Input Devices

The X.11/NeWS server provides support for input devices. The SunView VUID interface layer is not supported.

5

QUESTIONS, ANSWERS, HINTS, AND TIPS

5

QUESTIONS, ANSWERS, HINTS, AND TIPS

Q&A, and Tip of the Month

Hints & Tips #6

This is the sixth in a continuing series of this column which I have created for two purposes.[5] First, some questions are asked regularly on the AnswerLine. I feel everyone can benefit from distributing discussions of these problems as widely as possible. Second, a large and constantly growing body of information, hints, and tips are not documented anywhere.

I will collect and distribute these information nuggets in this continuing column so that we can all learn from them. I will cover unusual topics, but this column should not be used as an alternative to contacting your support center or using the AnswerLine.

If you have a question that you would like answered in this column, please mail your question to 'Software Technical Bulletins' at Sun Microsystems, Inc., 2550 Garcia Avenue, M/S 2-312, Mountain View, CA 94043. You can also send in your question by electronic mail to *sun!stb-editor*. U. S. customers can call Sun Customer Software Services AnswerLine at **800 USA-4-SUN** for technical questions on this column or any other article in this bulletin. I look forward to hearing from you!

Avoiding Quota Delays

Is it sometimes very slow to log onto your Sun? Does it seem to take forever from typing the password to getting your prompt? If your answers are yes, you are likely getting delayed by quota checking done at login time.

Even if you are not running the quota system and quotas are not configured into your kernel, the program `/usr/ucb/quota` is executed by `/bin/login`. 'Quota' will take a look at every mounted file system, both local and NFS. For each NFS file system, it sends an RPC request to the server and requests that it verify that you are not over quota.

[5] This continuing column is submitted by Chuq Von Rospach, Customer Software Services.

This checking can take considerable time if you have a large number of file systems mounted on your machine. The checks are done sequentially. This can cause additional delays if quota hangs until its check request times out. This can be caused by any of the three following conditions.

- one of the mounted file systems is down
- the NFS server is running a SunOS release older than release 3.0
- `rpc.quotad` is not running

It is possible on a machine with a large number of NFS mounts or a down server to cause a login procedure to take two or three minutes in the worst case!

Fortunately, there are two ways around this problem. If you do not run quotas anywhere in your organization, it is possible to disable the `/usr/ucb/quota` program completely by replacing it with a symbolic link to the `/bin/true` program as shown in the example below.

```
# cd /usr/ucb
# mv quota quota.hold
# ln -s /bin/true quota
```

Please note, *do not* simply delete or move `/usr/ucb/quota`, or the login program will fail.

This example does not work if some of your machines are running quotas since it removes the quotacheck completely. If you want to check quotas on some machines but not on others, you can use the `noquota` option in `/etc/fstab`. If you add `noquota` to the mount options on all of the file systems that are not under quota control, the `/usr/ucb/quota` program will skip them and not try to verify the quota over the network. An example `fstab` entry for this is shown below.

```
blurfl:/usr/blurfl /usr/blurfl nfs rw,noquota,soft,bg 0 0
```

By removing the quota checks from the file systems that are not under quota control, you can remove most of the time delays you are seeing while logging in.

Tip of the Month (TOM)

This month's Tip of the Month includes a few hints on making your mouse more responsive. There are a number of variables you can set with `Defaultsedit` that can make your mouse work more to your liking. If you start up `Defaultsedit` and go to the Input section, you will see the options 'Jitter_Filter' and 'Speed_Enforced'. By default, these options are On and Yes, respectively. However, these options are useful only to very early machines such as the 100U. For Sun2, Sun3, and Sun4 machines, these two options should be turned off.

Another improvement you can make is to modify the Mouse Motion Scaling. By default, the distance you move the mouse corresponds directly to the distance the mouse-arrow is moved on the screen. If you change the scaling, though, you can cause the mouse-arrow to move faster on the screen than you move the mouse on the mouse tablet. A good set of defaults to get you started are shown below.

```
/Input/1st_ceiling  "1"
/Input/1st_factor   "1"
/Input/2nd_ceiling  "16"
/Input/2nd_factor   "2"
/Input/3rd_ceiling  "32"
/Input/3rd_factor   "3"
/Input/4th_ceiling "65535"
/Input/4th_factor "5"
```

Once you do this, you need to make sure that `input_from_defaults` is run when you log onto your workstation. Note to run it only when you are on the console, since you do not want to change the defaults for someone else accidentally! Put this the following in your `.login` file.

```
if ('tty' == /dev/console) input_from_defaults
```

Your mouse customization will now be active next time you log in.

6

THE HACKERS' CORNER

6

THE HACKERS' CORNER

Browsing Mail Conveniently

Browser, A Mail Program

The Browser program introduced in this article has been developed by several programmers, consultants, and enthusiasts interested in tool development. Browser is a graphical, SunView-based application for viewing files and directories. It has been developed over a period of time and has been widely tested, especially in the United Kingdom.

The Script

The Browser program is submitted to the STB from Sun Europe. The program appears at the end of this article.

Comments on Browser program features, improvements, or the user interface are welcomed. Please send them to *sun!stb-editor*. I will then forward them to Sun Europe.

Please consult your local shell script or programming expert regarding any script or code problems. The Browser source archive is not offered as a supported Sun product, but as an item of interest to enthusiasts wanting to try out something for themselves. Note that the program may not work in all cases, and may not be compatible with future SunOS releases.

Browser

Browser has evolved over time. The most recent features are shown below.

- takes an optional directory argument, and passes window arguments; robust to problems with the initial directory
- stretches text display horizontally, has scrollbars in any orientation; and incorporates any plane-group combination for all Sun 3 systems
- understands small fonts, though it enforces a maximum font size
- the tool icon now labeled with the directory name when closed

- uses `cmdtool` if TERM is `sun-cmd` for SunOS release 3.4, otherwise uses SHELLTOOL from the environment for shells
- allocates the canvas and directory 'cache' dynamically, reducing memory overhead
- removes the restriction on the maximum permitted directory entries
- contains scrollbars that reflect the directory size

Notes on Usage

Browser is a window-based application containing a canvas on which a graphical representation of the current directory is displayed. The archive contains icons for the different types of objects in the file system.

Selecting an object now selects that object only. Selection is now integrated with the SunView selection mechanism. You must 'double-click' or use the menu to actually view the object.

You can select an item by pointing at it and clicking the left mouse button. The name of the file or directory then becomes the SunView selection as well. This is the relative pathname, except in the case of '.'. In this case it is more useful to provide the full, absolute pathname.

'Double-clicking' the left mouse button lets you view the selected object. Directories cause Browser to change directory into them and display their contents on the canvas. Files then are displayed as text sub-windows. Objects are displayed in reverse-video upon selection, and then are displayed in mid-grey while being accessed.

A menu is displayed when you click the right mouse button, allowing editing or deleting of files (not directories). Greyed-out items on the canvas or menu signify that you do not have sufficient file-access permissions.

Pressing and holding the control key while 'double-clicking' or selecting from the menu starts a separate `textedit` process for the file, in lieu of using a pop-up window. You can then view multiple files easily. In the case of a directory, the control key starts another Browser process in that directory.

Choices on the menu quickly take you to the root directory (/0, home ($HOME), and the previous directory just as if you had followed a symbolic link. Note that the previous directory is not the same as '..'. Another accelerator prompts you, asking for the name of the directory you want. You can also take the desired directory name from a source outside the tool. For example, you may select the directory name from `pwd` in a `shelltool`, and then choose 'view' from the menu.

Browser Program Installation

Follow the steps listed below to install the Browser application.

1. Save the mail you receive in a file as *filename*.
2. Edit the file to remove everything above the line containing `#!/bin/sh`. This saves the `shar` portion of the mail message.
3. sh <*filename*> <return>
4. make browser <return>
5. browser <return>

```
#! /bin/sh
# this is a shell archive, meaning:
# 1. Remove everything above the #! /bin/sh line
# 2. Save the resulting text in a file.
# 3. Execute the file with /bin/sh to create the files:
#   Makefile
#   browser.h
#   br_canv.c
#   br_main.c
#   br_menu.c
#   br_seln.c
#   br_text.c
#   confirm.c
#   bbad.icon
#   bblk.icon
#   bcha.icon
#   bdir.icon
#   bexe.icon
#   bfil.icon
#   brca.icon
#   brok.icon
#   brow.icon
# This archive created: Tue Jun 30 12:08:05 BST 1987
export PATH; PATH=/bin:$PATH
#
if [ -f Makefile ]
then
  echo shar: will not over-write existing file Makefile
else
  echo shar: extracting 'Makefile',      326 characters
  cat > Makefile <<'Funky_Stuff'
#
# browser - makefile
#
LIBS = -lsuntool -lsunwindow -lpixrect

BROBJS = br_main.o br_canv.o br_text.o br_seln.o br_menu.o confirm.o

BRSRC = br_main.c br_canv.c br_text.c br_seln.c br_menu.c confirm.c

browser : $(BROBJS)
    $(CC) $(CFLAGS) $(LDFLAGS) $(BROBJS) $(LIBS) -o browser

sources : $(BRSRC)

$(BRSRC) :
    sccs get $@
Funky_Stuff
  len=`wc -c < Makefile`
  if [ $len !=      326 ] ; then
    echo error: Makefile was $len bytes long, should have been      326
  fi
```

```
fi # end of overwriting check
if [ -f browser.h ]
then
  echo shar: will not over-write existing file browser.h
else
  echo shar: extracting 'browser.h',     3301 characters
  cat > browser.h <<'Funky_Stuff'
/*
 * browser.h - include file for browser global data
 *
 *
 */

#include <suntool/sunview.h>        /* SunView header files */
#include <sunwindow/notify.h>
#include <suntool/canvas.h>
#include <suntool/scrollbar.h>
#include <suntool/textsw.h>

#include <stdio.h>          /* system header files */
#include <sys/stat.h>
#include <sys/dir.h>
#include <sys/file.h>

extern  int errno;

extern  char br_name[];         /* version status */
extern  char br_version[];

extern  struct stat sbuf;       /* buffer for file stats */
extern  struct stat *sp;

extern  Frame       base_frame,     /* directory frame */
            view_frame;     /* edit pop-up */
extern  Canvas      canvas;     /* directory window */
extern  Pixwin      *pw;        /* pixwin of that window */

extern  Scrollbar   vertical_sb,    /* for scrolling directory */
            horizontal_sb;
extern  Menu        action_menu;    /* file menu */
extern  Textsw      viewsw;     /* edit window */
extern  Textsw      scratch;    /* scratch window */

#define SEMI_SCROLL_FILES 100
#define SEMI_SCROLL_COLS 8
#define SPACE_WIDTH 10
#define IMAGE_SIZE 64
#define NAME_OFFSET 47          /* ((IMAGE_SIZE/2)+15) */

#define BR_SCROLL_TO_TOP    TRUE
#define BR_DONT_SCROLL      FALSE

#define BR_CONTINUE     TRUE
```

```
#define BR_GETVALUE     2

extern  int maxcols;        /* width of canvas in images */
extern  int noindir;        /* number of directory entries */
extern  char    home_dir[];     /* users home directory */
extern  char    root_dir[];     /* root directory */
extern  char    real_dir[];     /* current directory */
extern  char    last_dir[];     /* last current directory */
                    /* not .. if followed symbolic link */
extern  char    text_dir[];     /* where textsw contents came from */
extern  char    sel_dir[];      /* current selection */
extern  char    name_stripe[];      /* frame stripe buffer */

#ifdef DEBUG
extern int debug;           /* trace status */
#endif

struct dir_disp {           /* our record of dir entry */
    char dname[256];        /* file name */
    int dmode;          /* file mode, 0 = bad stat */
    int dx;             /* pixel co-ord of image */
    int dy;
    Pixrect *dicon;         /* associated image */
};

extern  struct dir_disp *d_start;
extern  struct dir_disp *dptr;

extern  char shelltool[];       /* user preferred shelltool */

#define highlight(aptr) image_rop(aptr, PIX_NOT(PIX_DST), (Pixrect *)0)

extern  Pixrect bdir_pr;
extern  Pixrect bblk_pr;
extern  Pixrect bcha_pr;
extern  Pixrect bexe_pr;
extern  Pixrect bfil_pr;
extern  Pixrect bbad_pr;

/* from br_main.c */

extern  int     sort_ents();
extern  void        scan_dir();
extern  void        sort_dir();
extern  void        do_delete();
extern  void        do_dir();
extern  int     good_dir();

/* from br_menu.c */

extern  void        init_menu();
extern  Menu_item   do_menu();
extern  void        do_default_action();
extern  void        do_action();

/* from br_canv.c */

extern  void        image_rop();
extern  void        clear_canvas();
extern  void        init_canv();
extern  void        resize_canvas();
extern  void        resize_canvas_window();
extern  struct dir_disp *identify();
extern  int     double_click();
extern  void        show_action_started();
extern  void        show_action_stopped();
extern  void        track_selection();
extern  void        select_proc();
extern  void        draw_dir();
extern  void        name_rop();

/* from br_text.c */

extern  void        init_text();
extern  int     ok_to_reset();
extern  void        check_done();
extern  Notify_value    check_quit();
extern  void        do_edit();
extern  void        do_process();


Funky_Stuff
  len=`wc -c < browser.h`
  if [ $len !=     3301 ] ; then
    echo error: browser.h was $len bytes long, should have been     3301
  fi
fi # end of overwriting check
if [ -f br_canv.c ]
then
  echo shar: will not over-write existing file br_canv.c
else
  echo shar: extracting 'br_canv.c',     9596 characters
  cat > br_canv.c <<'Funky_Stuff'
/*
 * br_canv.c - canvas routines for browser
 *
 *
 */
#include "browser.h"

#include <sys/time.h>

char br_canv_sid[] = "@(#)br_canv.c 1.5 6/30/87";

int tracking = FALSE;           /* state variables - ugh */
int spawning = TRUE;
```

```
long click_timeout;
int click_space;
int last_x = 0;
int last_y = 0;

struct timeval tnow;
struct timeval tlast;
struct timezone tzone;

char *rindex();

                    /* image definitions */

static short brow_image[] = {       /* icon for program */
#include "brow.icon"
};
static mpr_static(brow_pr, 64, 64, 1, brow_image);

static short bfil_image[] = {       /* regular file image */
#include "bfil.icon"
};
static mpr_static(bfil_pr, 64, 64, 1, bfil_image);

static short bdir_image[] = {       /* directory image */
#include "bdir.icon"
};
static mpr_static(bdir_pr, 64, 64, 1, bdir_image);

static short bexe_image[] = {       /* executable image */
#include "bexe.icon"
};
static mpr_static(bexe_pr, 64, 64, 1, bexe_image);

static short bbad_image[] = {       /* unknown image */
#include "bbad.icon"
};
static mpr_static(bbad_pr, 64, 64, 1, bbad_image);

static short bblk_image[] = {       /* block device image */
#include "bblk.icon"
};
static mpr_static(bblk_pr, 64, 64, 1, bblk_image);

static short bcha_image[] = {       /* character device image */
#include "bcha.icon"
};
static mpr_static(bcha_pr, 64, 64, 1, bcha_image);

                    /* mask definitions */
static short gray25[16]  =  {       /*  25 % gray pattern*/
#include <images/square_25.pr>
};
static mpr_static(gray25_pr, 16, 16, 1, gray25);
static short gray_out[16] = {        /* 50 % gray pattern */
#include <images/square_50.pr>
};
static mpr_static(gray_out_pr, 16, 16, 1, gray_out);

Pixfont *br_font;            /* font in use */
int f_width, f_height;           /* max character size in pixels */
int name_split;              /* where to split long name */

void
image_rop(ptr,op,mask)
struct dir_disp *ptr;
int op;
Pixwin *mask;
{
    /* draw or modify a file image */

    if ( ptr != NULL )
        pw_rop(pw,ptr->dx,ptr->dy,IMAGE_SIZE,IMAGE_SIZE,op,mask,0,0);
#ifdef DEBUG
    else
    {
        if (debug)
            fprintf(stderr,"image_rop: null pointer (ignored)\n");
    }
#endif
}

void
clear_canvas(scroll_to_top)
int scroll_to_top;
{
    /* clear the canvas and scroll to top left */

#ifdef DEBUG
    if (debug)
        fprintf(stderr, "clearing canvas\n");
#endif
    pw_writebackground(pw, 0, 0, window_get(canvas, CANVAS_WIDTH),
        window_get(canvas, CANVAS_HEIGHT), PIX_CLR);

    if (scroll_to_top)
    {
        scrollbar_scroll_to(vertical_sb, 1);
        scrollbar_scroll_to(horizontal_sb, 1);
    }
}

void
init_canv()
{
    /* hook icons to frames - this is not really canvas stuff
        but all other icon material is in here */
```

```
Icon brow_icon;

brow_icon = icon_create(
        ICON_IMAGE, &brow_pr,
        ICON_LABEL, "",
        0);

window_set(base_frame, FRAME_ICON, brow_icon, 0);

/* create scrollbars for canvas */

vertical_sb = scrollbar_create(SCROLL_LINE_HEIGHT, 20,
        0);
horizontal_sb = scrollbar_create(SCROLL_LINE_HEIGHT, 20,
        0);

canvas = window_create(base_frame, CANVAS,
        CANVAS_FAST_MONO, TRUE,
        CANVAS_AUTO_SHRINK, FALSE,
        CANVAS_FIXED_IMAGE, FALSE,
        WIN_VERTICAL_SCROLLBAR, vertical_sb,
        WIN_HORIZONTAL_SCROLLBAR, horizontal_sb,
        WIN_CONSUME_PICK_EVENTS,
            WIN_MOUSE_BUTTONS,
            LOC_DRAG,
            0,
        WIN_EVENT_PROC, select_proc,
        0);

/* check the font is okay */

br_font = (Pixfont *)window_get(canvas, WIN_FONT);

if ( br_font == NULL
    || ( f_width = br_font->pf_defaultsize.x ) > 8
    || ( f_height = br_font->pf_defaultsize.y ) > 16
   )
{
    br_font = pf_open("/usr/lib/fonts/fixedwidthfonts/screen.r.14");

    if ( br_font == NULL )
    {
        perror("screen.r.14");
        exit(1);
    }
    f_width = br_font->pf_defaultsize.x;
    f_height = br_font->pf_defaultsize.y;
}

name_split = IMAGE_SIZE / (f_width + 1);

pw = canvas_pixwin(canvas);
```

```
    /* determine and check double-click options (use text ones) */

    click_space = defaults_get_integer_check(
                "/Text/Multi_click_space",
                3,       /* default */
                0,       /* min */
                IMAGE_SIZE, /* max */
                0);
    click_timeout = 1000L * (long)defaults_get_integer_check(
                "/Text/Multi_click_timeout",
                390,        /* default */
                100,        /* min */
                1000,       /* max */
                0);
}
void
resize_canvas(width,height)
int width,height;
{
    window_set(canvas,
        CANVAS_WIDTH, width,
        CANVAS_HEIGHT, height,
        0);
}
void
resize_canvas_window()
{
    window_set(canvas,
        WIN_WIDTH, (int)window_get(canvas, CANVAS_WIDTH)
            + (int)scrollbar_get(vertical_sb, SCROLL_WIDTH),
        WIN_HEIGHT, (int)window_get(canvas, CANVAS_HEIGHT)
            + (int)scrollbar_get(horizontal_sb, SCROLL_HEIGHT),
        0);

}
struct dir_disp *
identify(x,y)
{
    /* map pixel selection co-ordinates to a file */

    int srow,scol;
    struct dir_disp *retptr = NULL;

    srow = (y - SPACE_WIDTH) / (IMAGE_SIZE + SPACE_WIDTH);
    scol = (x - SPACE_WIDTH) / (IMAGE_SIZE + SPACE_WIDTH);

    retptr = d_start + ( srow * maxcols + scol );

    if ( scol >= maxcols || retptr >= d_start+noindir )
    {
        retptr = NULL;
    }
#ifdef DEBUG
```

```
    if (debug)
        fprintf(stderr, "active file %x %s\n", retptr,
            (retptr != NULL)?retptr->dname:"void");
#endif
    return(retptr);
}
int
double_click(event)
Event *event;
{
    int d_clicked = FALSE;
    long usecs;
    int now_x, now_y, dis_x, dis_y;

    now_x = event_x(event);         /* where are we ? */
    now_y = event_y(event);

    gettimeofday(&tnow, &tzone);        /* what is the time ? */

    usecs = tnow.tv_usec;

    dis_x = (now_x > last_x)?(now_x - last_x):(last_x - now_x);
    dis_y = (now_y > last_y)?(now_y - last_y):(last_y - now_y);

    /* check that time elapsed since last click is less than timeout
        and that movement is less than maximum allowed */

    if ( tnow.tv_sec == tlast.tv_sec + 1 )
        usecs += 1000000;
    if ( ! ( tnow.tv_sec > tlast.tv_sec + 1 )
        && ( usecs - tlast.tv_usec < click_timeout )
        && dis_x <= click_space && dis_y <= click_space )
            d_clicked = TRUE;

    tlast.tv_sec = tnow.tv_sec;     /* remember time */
    tlast.tv_usec = tnow.tv_usec;
    last_x = now_x;             /* remember position */
    last_y = now_y;

    return( d_clicked );
}
void
show_action_started()
{
    if ( dptr != NULL )
    {
        highlight(dptr);

        pw_replrop(pw, dptr->dx, dptr->dy,
            IMAGE_SIZE,IMAGE_SIZE,
            PIX_SRC ^ PIX_DST,
            (Pixrect *)&gray25_pr,
            dptr->dx, dptr->dy);
    }
}
void
show_action_stopped()
{
    if ( dptr != NULL )
    {
        pw_replrop(pw, dptr->dx, dptr->dy,
            IMAGE_SIZE,IMAGE_SIZE,
            PIX_SRC ^ PIX_DST,
            (Pixrect *)&gray25_pr,
            dptr->dx, dptr->dy);

        highlight(dptr);
    }
}
void
track_seln(event)
Event *event;
{
    struct dir_disp *olddptr = NULL;

    olddptr = dptr;                   /* where we were */

    dptr=identify(event_x(event), event_y(event));/* where we are */

    if ( olddptr != dptr )                /* moved? */
    {
        if ( olddptr != NULL )
            highlight(olddptr);

        if ( dptr != NULL )
            highlight(dptr);
    }
}
void
select_proc(window, event, arg)
Window window;
Event *event;
caddr_t arg;
{
    /* called by notifier when events occur on canvas */

#ifdef DEBUG
    if ( debug )
        fprintf(stderr,"event %d received\n", event_id(event));
#endif

    spawning = event_ctrl_is_down(event)?TRUE:FALSE;

    switch ( event_id(event) )
    {
    case MS_RIGHT:
        if ( event_is_down(event) )
        {
            menu_show(action_menu,
                window,
                canvas_window_event(canvas,event),
                0);
        }
        break;

    case LOC_DRAG:
        if ( tracking )
            track_seln(event);      /* continue tracking */
        break;
    case MS_LEFT:
        if ( event_is_down(event) )
        {
            tracking = TRUE;    /* start tracking */

            unset_selection();

            track_seln(event);
        }
        else if ( tracking )
        {
            /* left button up */

            if ( dptr != NULL )
            {
                if ( double_click(event) )
                    do_default_action();

                /* dptr is stale if we did a cd */

                if ( dptr != NULL )
                    set_selection(dptr->dname);
            }
            tracking = FALSE;   /* stop tracking */
        }
        break;
    case LOC_RGNEXIT:
        if ( tracking )
        {
            if ( dptr != NULL )
            {
                highlight(dptr);
                dptr = NULL;
            }
            tracking = FALSE;
        }
        break;
    }
}
void
```

```
draw_dir(scroll_to_top)
int scroll_to_top;
{
    /* draw_dir() actually does the drawing.
        The pixwin calls are batched for efficiency. */

    Icon brow_icon;

#ifdef DEBUG
    if ( debug )
        fprintf(stderr, "drawing directory\n");
#endif
    dptr = d_start;

    pw_batch_on(pw);

    clear_canvas(scroll_to_top);

    while ( dptr < d_start+noindir )
    {
        image_rop(dptr, PIX_SRC, dptr->dicon);

        name_rop(dptr->dx, dptr->dy+NAME_OFFSET, dptr->dname);

        if ( access(dptr->dname,R_OK ) != 0 )
        {
            pw_replrop(pw, dptr->dx, dptr->dy,
                IMAGE_SIZE,IMAGE_SIZE,
                PIX_SRC & PIX_DST,
                (Pixrect *)&gray_out_pr,
                dptr->dx, dptr->dy);
        }
        dptr++;
    }

    pw_batch_off(pw);

    sprintf(name_stripe, "%s %s - %s (%d entries)",
        br_name,
        br_version,
        real_dir,
        noindir);

    brow_icon = (Icon)window_get(base_frame, FRAME_ICON);
    icon_set(brow_icon, ICON_LABEL, (rindex(real_dir, '/')+1), 0);

    window_set(base_frame,
        FRAME_LABEL, name_stripe,
        FRAME_ICON, brow_icon,
        0);

    dptr = NULL;    /* ensure no file is active */
    unset_selection();
}
void
name_rop(x,y,onp)
int x,y;
char *onp;
{
    char c = '\0';
    int l,twice_split;
    char nnp[256];
    char *np,*lp;

    strcpy(nnp,onp);        /* save name in local buffer */

    np = nnp;           /* may need to offset base */

    l = strlen(np);

    twice_split = name_split * 2;

    if ( l > twice_split )      /* too long for only two lines */
    {
        np += l - twice_split;
        *np = '>';
    }

    lp = np + name_split;       /* second half */

    if ( l > name_split )       /* write first half */
    {
        c = *lp;
        *lp = '\0';
    }
    pw_text(pw, x + 4 , y, PIX_SRC, br_font, np);

    if ( c != '\0' )        /* write second half */
    {
        *lp = c;
        pw_text(pw, x+4, y+15, PIX_SRC, br_font, lp);
    }
}
Funky_Stuff
  len=`wc -c < br_canv.c`
  if [ $len !=     9596 ] ; then
    echo error: br_canv.c was $len bytes long, should have been     9596
  fi
fi # end of overwriting check
if [ -f br_main.c ]
then
  echo shar: will not over-write existing file br_main.c
else
  echo shar: extracting 'br_main.c',     6867 characters
  cat > br_main.c <<'Funky_Stuff'
/*
```

```
 * browser - a graphical tool for viewing/editing files and directories
 *
 *
 */
#include "browser.h"

#include <sys/param.h>      /* for NOFILE */

char br_main_sid[] = "@(#)br_main.c 1.5 6/30/87";

char br_name[] = "Browser"; /* version status */
#ifdef MERGED
char br_version[] = "1.6(M)";
#else
char br_version[] = "1.6";
#endif

struct stat sbuf;       /* buffer for file stats */
struct stat *sp = &sbuf;

Frame       base_frame,     /* directory frame */
        view_frame;     /* edit pop-up */
Canvas      canvas;     /* directory window */
Pixwin      *pw;        /* pixwin of that window */

Scrollbar   vertical_sb,    /* for scrolling directory */
        horizontal_sb;
Menu        action_menu;    /* file action menu */
Textsw      viewsw;     /* edit window */
Textsw      scratch;    /* scratch window */

int maxcols = 0;            /* width of canvas in images */
int noindir = 0;            /* number of directory entries */
char    home_dir[256] = "";     /* users home directory */
char    root_dir[256] = "/";        /* root directory */
char    real_dir[256] = "";     /* current directory */
char    last_dir[256] = "";     /* last current directory */
            /* not .. if followed symbolic link */
char    sel_dir[256] = "";      /* current selection */
char    name_stripe[256];       /* frame stripe buffer */

#ifdef DEBUG
    int debug = FALSE;      /* trace status */
#endif

struct dir_disp *d_start;       /* base of directory cache */
struct dir_disp *dptr;          /* current entry in cache */

char    shelltool[256] = "shelltool";   /* user preferred shelltool */

char *malloc();
```

```
#ifdef MERGED
browser_main(argc, argv)
#else
main(argc, argv)
#endif
int argc;
char **argv;
{
    /* initialisation */

    int i;
    char *hp,*getenv();

    if ( (hp = getenv("HOME")) != NULL )/* where is home */
        strcpy(home_dir, hp);

    if ( (hp = getenv("TERM")) != NULL && strcmp(hp, "sun-cmd") == 0 )
        strcpy(shelltool, "cmdtool");
    else if ( (hp = getenv("SHELLTOOL")) != NULL )
        strcpy(shelltool, hp);

    /* if we exec from another window based tool, all kinds of
        material is left open; close it all to avoid
        running out of fds when we exec ourselves */

    for ( i = 3 ; i < NOFILE ; i++ )
        close(i);

    base_frame = window_create(0, FRAME,
            FRAME_SHOW_LABEL, TRUE,
            WIN_HEIGHT, 0,
            WIN_WIDTH, 0,
            WIN_ROW_HEIGHT, IMAGE_SIZE + SPACE_WIDTH,
            WIN_COLUMN_WIDTH, IMAGE_SIZE + SPACE_WIDTH,
            FRAME_ARGC_PTR_ARGV, &argc, argv,
            0);

    view_frame = window_create(base_frame, FRAME,
            FRAME_SHOW_LABEL, TRUE,
            FRAME_DONE_PROC, check_done,
            0);

    init_seln();

    init_text();

    init_canv();

    init_menu();

    notify_interpose_destroy_func(base_frame, check_quit);

    if ( argc > 1 )
    {
        if ( good_dir(argv[1]) )
        {
            do_dir(argv[1]);
        }
        else
        {
            perror(argv[1]);
            exit(1);
        }
    }
    else
    {
        if ( good_dir(".") )
        {
            do_dir(".");
        }
        else
        {
            perror(".");
            exit(1);
        }
    }

    resize_canvas_window();

    window_fit(base_frame);

    window_main_loop(base_frame);

    exit_seln();

    exit(0);
}

int
sort_ents(p1,p2)
struct dir_disp *p1;
struct dir_disp *p2;
{
    /* called from qsort to compare two entries */

    return( (int)strcmp( p1->dname, p2->dname ) );
}
void
scan_dir()
{
    /* read and sort the directory */

    struct dir_disp *dptr;

    DIR *dir_header = opendir(real_dir);
    struct direct *dp = readdir(dir_header);
#ifdef DEBUG
    if ( debug )
        fprintf(stderr,"scanning %s\n",real_dir);
#endif

    if ( d_start != NULL )
        free(d_start);

    if ( ( d_start =
        (struct dir_disp *)malloc(sizeof(struct dir_disp)) ) == NULL )
    {
            user_confirm("malloc", TRUE, errno);
            return;
    }

    dptr = d_start;

    noindir = 0;

    while ( dp != NULL )
    {
        if ( dp->d_fileno != 0 )
        {
            strcpy(dptr->dname, dp->d_name);/* name */

            if ( stat( dptr->dname, sp ) < 0 )/* mode */
            {
                dptr->dmode = 0;
            }
            else
            {
                dptr->dmode = sp->st_mode;
            }

            switch( dptr->dmode & S_IFMT )
            {
            case S_IFDIR:
                dptr->dicon = &bdir_pr;
                break;
            case S_IFBLK:
                dptr->dicon = &bblk_pr;
                dptr->dmode = 0;
                break;
            case S_IFCHR:
                dptr->dicon = &bcha_pr;
                dptr->dmode = 0;
                break;
            case S_IFREG:
                if ( dptr->dmode & S_IEXEC )
                    dptr->dicon = &bexe_pr;
                else
                    dptr->dicon = &bfil_pr;
                break;
            default:
                dptr->dicon = &bbad_pr;
                dptr->dmode = 0;
                break;
            }
            noindir++;

            d_start = (struct dir_disp *)realloc(d_start,
                    sizeof(struct dir_disp)*(noindir+1));

            if ( d_start == NULL )
            {
                user_confirm("realloc", TRUE, errno);
                return;
            }
            dptr = d_start + noindir;
        }
        dp = readdir(dir_header);
    }
    closedir(dir_header);
}
void
sort_dir()
{
    struct dir_disp *dptr;

    int row = 0;
    int col = 0;
    int width,height;

    /* sort the entries */

    qsort( (char *)d_start, noindir, sizeof(struct dir_disp), sort_ents);

    /* set required canvas size */

    if ( noindir < SEMI_SCROLL_FILES )
        maxcols = SEMI_SCROLL_COLS;
    else
        maxcols = (1152 - 20 - SPACE_WIDTH)/
                (IMAGE_SIZE + SPACE_WIDTH);

    width = ((IMAGE_SIZE + SPACE_WIDTH) * maxcols) + SPACE_WIDTH;

    height = ((noindir / maxcols) +
            ((noindir % maxcols)?1:0) )
            *(IMAGE_SIZE + SPACE_WIDTH)
            + SPACE_WIDTH;

    resize_canvas(width, height);

    /* allocate image positions */
    dptr = d_start;

    while ( dptr < d_start+noindir )
    {
        dptr->dx = col * (IMAGE_SIZE + SPACE_WIDTH)
                        + SPACE_WIDTH;
        dptr->dy = row * (IMAGE_SIZE + SPACE_WIDTH)
                        + SPACE_WIDTH;
        col++;

        if ( col >= maxcols )
        {
            col = 0;
            row++;
        }
        dptr++;
    }

}

void
do_delete()
{
    char mybuf[256];

    struct dir_disp *nptr = dptr + 1;

    sprintf(mybuf, "Confirm that you wish to delete %s", dptr->dname);

    if ( user_confirm(mybuf, FALSE, FALSE ) )
    {
        if ( unlink(dptr->dname) < 0 )
        {
            sprintf(mybuf, "unlink %s", dptr->dname);
            user_confirm(mybuf, TRUE, errno);
        }
        else
        {
            while ( nptr < d_start+noindir )
            {
#ifdef DEBUG
    if (debug)
        fprintf(stderr, "compress %x %x %s\n", dptr, nptr, nptr->dname);
#endif
                strcpy(dptr->dname,nptr->dname);
                dptr->dmode = nptr->dmode;
                dptr->dicon = nptr->dicon;
                dptr++;
                nptr++;
            }
            noindir--;
            draw_dir(BR_DONT_SCROLL);
        }
    }
}
do_move()
{
    user_confirm("Option not currently available",BR_CONTINUE,FALSE);
}

void
do_dir(new_dir)
char *new_dir;
{
    /* cd to named directory and display */

    char mybuf[80];

    if ( chdir(new_dir) < 0 )
    {
        sprintf(mybuf,"chdir %s", new_dir);
        user_confirm(mybuf, TRUE, errno);
    }
    else if ( strcmp(new_dir, ".") != 0 )
    {
        strcpy(last_dir, real_dir);
    }
    if ( getwd(real_dir) < 0 )
    {
        user_confirm("getwd", TRUE, errno);
    }
    else
    {
        scan_dir();
        sort_dir();
        draw_dir(BR_SCROLL_TO_TOP);
    }
}

/* check a name is an existing, readable, directory */
int
good_dir(tname)
char *tname;
{
    int good_name = TRUE;

    if ( tname == NULL
        || *tname == '\0'
        || stat(tname, sp ) < 0
        || ((sp->st_mode&S_IFMT) != S_IFDIR)
        || access(tname, R_OK) != 0 )
          good_name = FALSE;

    return(good_name);
}
```

```
Funky_Stuff
  len=`wc -c < br_main.c`
  if [ $len !=     6867 ] ; then
    echo error: br_main.c was $len bytes long, should have been     6867
  fi
fi # end of overwriting check
if [ -f br_menu.c ]
then
  echo shar: will not over-write existing file br_menu.c
else
  echo shar: extracting 'br_menu.c',     4191 characters
  cat > br_menu.c <<'Funky_Stuff'
/*
 * browser - br_menu.c - menu handling for browser
 *
 *
 */
#include <suntool/sunview.h>        /* SunView header files */

#include "browser.h"

char br_menu_sid[] = "@(#)br_menu.c 1.5 6/30/87";

#define ACT_VIEW    1
#define ACT_EDIT    2
#define ACT_DEL     3
#define ACT_ROOT    4
#define ACT_HOME    5
#define ACT_BACK    6
#define ACT_PROMPT  7
#define ACT_SHELL   9
#define ACT_TRACE   8
#define ACT_MOVE    10

extern int spawning;

Menu action_menu;

void
init_menu()
{
    /* initialise the canvas menu */

    action_menu = menu_create(

        MENU_INITIAL_SELECTION_SELECTED, TRUE,

        MENU_ITEM,
            MENU_STRING,    "View",
            MENU_GEN_PROC,  do_menu,
            MENU_VALUE, ACT_VIEW,
            0,
        MENU_ITEM,
                MENU_STRING,    "Edit",
                MENU_GEN_PROC,  do_menu,
                MENU_VALUE, ACT_EDIT,
                0,
            MENU_ITEM,
                MENU_STRING,    "Delete",
                MENU_GEN_PROC,  do_menu,
                MENU_VALUE, ACT_DEL,
                0,
#ifdef MOVING
            MENU_ITEM,
                MENU_STRING,    "Move",
                MENU_GEN_PROC,  do_menu,
                MENU_VALUE, ACT_MOVE,
                0,
#endif
            MENU_ITEM,
                MENU_STRING,    "Root",
                MENU_GEN_PROC,  do_menu,
                MENU_VALUE, ACT_ROOT,
                0,
            MENU_ITEM,
                MENU_STRING,    "Home",
                MENU_GEN_PROC,  do_menu,
                MENU_VALUE, ACT_HOME,
                0,
            MENU_ITEM,
                MENU_STRING,    "Previous",
                MENU_GEN_PROC,  do_menu,
                MENU_VALUE, ACT_BACK,
                0,
            MENU_ITEM,
                MENU_STRING,    "Prompt",
                MENU_GEN_PROC,  do_menu,
                MENU_VALUE, ACT_PROMPT,
                0,
            MENU_ITEM,
                MENU_STRING,    "Shell",
                MENU_GEN_PROC,  do_menu,
                MENU_VALUE, ACT_SHELL,
                0,
#ifdef DEBUG
            MENU_ITEM,
                MENU_STRING,    (debug)?"Trace Off":"Trace On",
                MENU_GEN_PROC,  do_menu,
                MENU_VALUE, ACT_TRACE,
                0,
#endif
            0);
}
Menu_item
do_menu(mi,op)
Menu_item mi;
```

```
Menu_generate op;
{
    /* menu generate proc */

    int inactive = FALSE;
    int mval = (int) menu_get(mi, MENU_VALUE);
#ifdef DEBUG
    if ( debug )
        fprintf(stderr, "do_menu op = %d, item = %d\n", op, mval);
#endif
    switch( op )
    {
    case MENU_DISPLAY:
        switch( mval )
        {
        case ACT_VIEW:
            if ( dptr == NULL )
            {
                if ( ! good_dir(get_selection()) )
                    inactive = TRUE;
            }
            else
            {
                if ( dptr->dmode & S_IFDIR )
                {
                    if ( ! good_dir(dptr->dname) )
                        inactive = TRUE;
                }
                else
                {
                    if ( dptr->dmode == 0 ||
                         access(dptr->dname, R_OK) != 0 )
                        inactive = TRUE;
                }
            }
            break;
        case ACT_MOVE:
        case ACT_EDIT:
        case ACT_DEL:
            if ( dptr == NULL
                || dptr->dmode == 0
                || (dptr->dmode & S_IFDIR)
                || access(dptr->dname,W_OK) != 0 )
                    inactive = TRUE;
            break;
        case ACT_BACK:
            if ( ! good_dir(last_dir) )
                inactive = TRUE;
            break;
        case ACT_HOME:
            if ( ! good_dir(home_dir) )
                inactive = TRUE;
            break;
#ifdef DEBUG
        case ACT_TRACE:
            menu_set(mi,
                MENU_STRING, debug?"Trace Off":"Trace On",
                0);
            break;
#endif
        }
        menu_set(mi, MENU_INACTIVE, inactive, 0);
        break;
    case MENU_NOTIFY:
        do_action(mval);
        break;
    }
    return(mi);
}
void
do_default_action()
{
    /* action required, but menu based checks have not been done */

    if ( dptr != NULL
        && dptr->dmode != 0
        && access(dptr->dname,R_OK) == 0 )
            do_action( ACT_VIEW );
}
void
do_action(action)
int action;
{
    show_action_started();

    switch ( action )
    {
    case ACT_VIEW:
        if ( dptr == NULL )
        {
            do_dir(sel_dir);
        }
        else if ( dptr->dmode & S_IFDIR )
        {
            if ( spawning )
                do_process("browser", dptr->dname);
            else
                do_dir(dptr->dname);
        }
        else  /* assume ordinary file */
        {
            if ( spawning )
                do_process("textedit", dptr->dname);
            else
                do_edit(real_dir, dptr->dname, "view");
        }
```

```
            break;
        case ACT_EDIT:
            if ( spawning )
                do_process("textedit",dptr->dname);
            else
                do_edit(real_dir, dptr->dname, "edit");
            break;
        case ACT_DEL:
            do_delete();
            break;
        case ACT_ROOT:
            do_dir(root_dir);
            break;
        case ACT_HOME:
            do_dir(home_dir);
            break;
        case ACT_BACK:
            do_dir(last_dir);
            break;
        case ACT_PROMPT:
            if ( user_confirm("Give new directory name: ",
                BR_GETVALUE, FALSE) )
                    do_dir(user_value());
            break;
        case ACT_SHELL:
            do_process( shelltool, 0 );
            break;
        case ACT_MOVE:
            do_move();
            break;
#ifdef DEBUG
        case ACT_TRACE:
            debug = !debug;
            break;
#endif
        }
        show_action_stopped();
}

Funky_Stuff
  len=`wc -c < br_menu.c`
  if [ $len !=      4191 ] ; then
    echo error: br_menu.c was $len bytes long, should have been      4191
  fi
fi # end of overwriting check
if [ -f br_seln.c ]
then
  echo shar: will not over-write existing file br_seln.c
else
  echo shar: extracting 'br_seln.c',      3336 characters
  cat > br_seln.c <<'Funky_Stuff'
/*
 * browser - br_seln.c - selection handling for browser
 *
 *
 */
#include "browser.h"

#include <suntool/seln.h>

static char br_seln_id[] = "@(#)br_seln.c1.4 5/29/87";

static Seln_client s_client;

static char s_buffer[256];

void fkey_proc();
Seln_result reply_proc();

init_seln()
{
#ifdef DEBUG
    if (debug )
        fprintf(stderr,"initialise selection client\n");
#endif
    s_client = seln_create(fkey_proc, reply_proc, (char *)0);

    if ( s_client == NULL )
        user_confirm("unexpected error creating selection client",
            TRUE,
            errno);
}
exit_seln()
{
#ifdef DEBUG
    if (debug )
        fprintf(stderr,"destroy selection client\n");
#endif

    seln_destroy(s_client);
}

char *
get_selection()
{
    Seln_holder holder;
    Seln_request    *sel_buf;

    holder = seln_inquire(SELN_PRIMARY);

    sel_buf = seln_ask(&holder,
            SELN_REQ_CONTENTS_ASCII,
            0,
            0);

    strncpy(sel_dir, sel_buf->data + sizeof(Seln_attribute), 256);
```

```
#ifdef DEBUG
    if (debug )
        fprintf(stderr,"get selection returns %s\n",sel_dir);
#endif
    return(sel_dir);
}
void
set_selection(value)
char *value;
{
#ifdef DEBUG
    if (debug )
        fprintf(stderr,"set selection to %s\n",value);
#endif

    if ( seln_acquire(s_client, SELN_PRIMARY) != SELN_PRIMARY )
        user_confirm("unexpected error acquiring selection",
            TRUE,
            errno);

    if ( strlen(value) == 1 && *value == '.' )
        strcpy(s_buffer, real_dir);
    else
        strcpy(s_buffer, value);
}
void
unset_selection()
{
#ifdef DEBUG
    if (debug )
        fprintf(stderr,"unset selection\n");
#endif

    *s_buffer = '\0';
}

void
fkey_proc(cdata, args)
char *cdata;
Seln_function_buffer *args;
{
    Seln_holder *holder;

#ifdef DEBUG
    if (debug)
        fprintf(stderr,"fkey_proc: activated\n");
#endif
    switch ( seln_figure_response(args, &holder) )
    {
    case SELN_IGNORE:
        break;
    case SELN_REQUEST:
        break;
    case SELN_SHELVE:
        break;
    case SELN_FIND:
        break;
    case SELN_DELETE:
        break;
    }
}
Seln_result
reply_proc(item, context, length)
Seln_attribute item;
Seln_replier_data *context;
int length;
{
    int size, needed;
    char *destp = NULL;

    switch ( context->rank )
    {
    case SELN_PRIMARY:
        break;
    case SELN_SECONDARY:
        break;
    case SELN_SHELF:
        break;
    }

    switch ( item )
    {
    case SELN_REQ_CONTENTS_ASCII:

#ifdef DEBUG
    if (debug )
        fprintf(stderr,"reply_proc: give ascii selection\n");
#endif
        context->context = s_buffer;

        size = strlen(context->context);
        destp = (char *)context->response_pointer;

        needed = size + 4;

        if ( size % 4 != 0 )
            needed += 4 - size % 4;

        strcpy(destp, context->context);
        destp += size;

        while ( (int)destp % 4 != 0 )
            *destp++ = '\0';

        context->response_pointer = (char **)destp;
        *context->response_pointer++ = 0;
        return( SELN_SUCCESS );

    case SELN_REQ_YIELD:

#ifdef DEBUG
    if (debug )
        fprintf(stderr,"reply_proc: yield selection\n");
#endif
        if ( dptr != NULL )
        {
            highlight(dptr);
            dptr = NULL;
        }
        unset_selection();

        *context->response_pointer++ = (char *)SELN_SUCCESS;

        return( SELN_SUCCESS );

    case SELN_REQ_BYTESIZE:

#ifdef DEBUG
    if (debug )
        fprintf(stderr,"reply_proc: give selection size\n");
#endif
        *context->response_pointer++ = (char *)strlen(s_buffer);

        return( SELN_SUCCESS );

    case SELN_REQ_END_REQUEST:

        return( SELN_SUCCESS );

    default:

        return( SELN_UNRECOGNIZED );
    }
}
Funky_Stuff
  len=`wc -c < br_seln.c`
  if [ $len !=     3336 ] ; then
    echo error: br_seln.c was $len bytes long, should have been     3336
  fi
fi # end of overwriting check
if [ -f br_text.c ]
then
  echo shar: will not over-write existing file br_text.c
else
  echo shar: extracting 'br_text.c',     2626 characters
  cat > br_text.c <<'Funky_Stuff'
/*
 * br_text.c - text routines for browser
 *
 * Alistair Skinner - March 1986 Sun Microsystems Europe Inc.
 *
 */
#include "browser.h"

char br_text_sid[] = "@(#)br_text.c 1.5 6/30/87";

void
init_text()
{
    scratch = window_create(view_frame, TEXTSW,
            TEXTSW_DISABLE_CD, TRUE,
            TEXTSW_DISABLE_LOAD, TRUE,
            WIN_ROWS, 1,
            WIN_X, 0,
            0);

    viewsw = window_create(view_frame, TEXTSW,
            WIN_BELOW, scratch,
            WIN_X, 0,
            TEXTSW_BROWSING, TRUE,
            TEXTSW_DISABLE_CD, TRUE,
            TEXTSW_DISABLE_LOAD, TRUE,
            0);
}
int
ok_to_reset()
{
    /* check there are no edits outstanding */

    int modified = TRUE;

    if ( ! (int)window_get(viewsw, TEXTSW_MODIFIED)
        || user_confirm("** This action will destroy unsaved edits **", \
                FALSE,FALSE) )
    {
        textsw_reset(viewsw, 500, 500);
        modified = FALSE;
    }

    return( !modified );
}
void
check_done(donef)
Frame donef;
{
    /* called by the notifier when user selects "done" */

    if ( ok_to_reset() )
    {
        window_set(view_frame, WIN_SHOW, FALSE, 0);
    }
}
```

```
Notify_value
check_quit(quitf, dstatus)
Frame quitf;
Destroy_status dstatus;
{
    /* called by the notifier when user selects "quit" */

    if ( dstatus == DESTROY_CHECKING && !ok_to_reset() )
    {
        notify_veto_destroy(base_frame);
        return(NOTIFY_DONE);
    }
    textsw_reset(scratch,500,500);

    return( notify_next_destroy_func(quitf, dstatus) );
}
void
do_edit(t_directory,t_file,t_action)
char *t_directory;
char *t_file;
char *t_action;
{
    /* edit or view the currently active file */

    char full_name[256];
    int len = 0;

#ifdef DEBUG
    if ( debug )
        fprintf(stderr, "%sing %s\n", t_action, t_file);
#endif
    if ( ok_to_reset() )
    {
        /* use full path name in case we change directory later */

        strcpy(full_name, t_directory);
        len = strlen(full_name);
        full_name[len++] = '/';
        strcpy(&full_name[len],t_file);

        window_set(viewsw,
            TEXTSW_FILE, full_name,
            TEXTSW_BROWSING, (*t_action=='v'),
            0);

        sprintf(name_stripe, "%s %s (%s) - %s  (%d bytes)",
            br_name,
            br_version,
            t_action,
            t_file,
            window_get(viewsw,TEXTSW_LENGTH)
            );
        window_set(view_frame,
            FRAME_LABEL, name_stripe,
            WIN_SHOW, TRUE,
            0);
    }
}
void
do_process(pname,parg)
char *pname;
char *parg;
{
    /* spawn a new, detached process */

    switch ( vfork() )
    {
    case 0:
        switch( vfork() )
        {
        case 0:
            execlp(pname, pname, parg, 0);
            perror("browser: could not exec process");
            _exit(-1);
            break;
        case -1:
            perror("browser: could not dettach process");
            _exit(-1);
            break;
        default:
            _exit(0);
        }
        break;
    case -1:
        user_confirm("cannot fork process", TRUE, TRUE);
        break;
    default:
        wait(0);
        break;
    }
}
Funky_Stuff
  len=`wc -c < br_text.c`
  if [ $len !=     2626 ] ; then
    echo error: br_text.c was $len bytes long, should have been     2626
  fi
fi # end of overwriting check
if [ -f confirm.c ]
then
  echo shar: will not over-write existing file confirm.c
else
  echo shar: extracting 'confirm.c',     4369 characters
  cat > confirm.c <<'Funky_Stuff'
/*
 * confirm.c - user confirmation routines
 *
 *
 * isanerr now contains errno - 15 Apr 87 AES
 *
 * Based on example code in the SunView Programmers Guide.
 *
 * These routines provide a confirmer pop-up which the
 * user must respond to. The interface is:
 *
 *  int
 *  user_confirm(prompt, isacont, isanerr)
 *  char *prompt;
 *  int isacont;
 *  int isanerr;
 *
 * It returns TRUE if the user said yes/continue
 *        or FALSE if the user said no
 *
 * Use in one of three ways, in the first two cases, the
 * only choice is to continue.
 *
 * (1) For informational messages after a system error:
 *
 *  user_confirm("my message", TRUE, errno);
 *
 *  in this case the message appears with the system
 *  error message appended in the style of perror(3)
 *
 * (2) For informational messages when there is no system error:
 *
 *  user_confirm("my message", TRUE, FALSE);
 *
 * (3) For situations where the user must confirm an action:
 *
 *  user_confirm("my message", FALSE, FALSE);
 *
 * The remaining case, where there has been a system error
 * and the user must confirm an action, will work but the
 * format of the output will probably not be suitable, and
 * there will probably be too much information for the user.
 *
 */

#include <suntool/sunview.h>
#include <suntool/panel.h>

#define MAX_MSG 80
#define BR_CONTINUE TRUE
#define BR_GETVALUE 2

extern int  errno;
extern int  sys_nerr;
extern int  *sys_errlist[];
```

```
static short ok_image[] = {
#include "brok.icon"
};
mpr_static(ok_button, 64, 64, 1, ok_image);

static short ca_image[] = {
#include "brca.icon"
};
mpr_static(ca_button, 64, 64, 1, ca_image);


static void
yes_no(item, event)
Panel_item item;
Event *event;
{
    window_return(panel_get(item, PANEL_CLIENT_DATA));
}

Frame conf;
Panel panel;
Panel_item file_spec;
char uvalue_store[80];

static Frame
init_conf(prompt,isacont,isanerr)
char *prompt;
int isacont;
int isanerr;
{
        char msgbuf[MAX_MSG];
    char *cp;
    Panel_item msg;

    int left, top, width, height;
    Rect *r;
    struct pixrect *pr;

    conf = window_create(0, FRAME,
            FRAME_SHOW_LABEL, FALSE,
            0);

    panel = window_create(conf, PANEL, 0);

    pr = &ok_button;

    cp = msgbuf;

    if ( prompt == NULL || *prompt == '\0' )
        strcpy(cp, "(null message)");
    else
        strncpy(cp, prompt, MAX_MSG);
```

```
cp = msgbuf + strlen(msgbuf);

if ( isanerr )
{
    if ( errno < sys_nerr )
        sprintf(cp," : %s",sys_errlist[isanerr]);
    else
        sprintf(cp, " : Error Number %d",isanerr);
}

if ( isacont == BR_GETVALUE )
{
    file_spec = panel_create_item(panel, PANEL_TEXT,
        PANEL_ITEM_Y, ATTR_ROW(1),
        PANEL_LABEL_STRING, msgbuf,
        PANEL_VALUE_DISPLAY_LENGTH, 20,
        PANEL_VALUE, "",
        0);
}
else
{
    msg = panel_create_item(panel, PANEL_MESSAGE,
        PANEL_LABEL_STRING, msgbuf,
        PANEL_ITEM_Y, ATTR_ROW(1),
        0);
}


panel_create_item(panel, PANEL_BUTTON,
        PANEL_ITEM_X, 25,
        PANEL_ITEM_Y, 50,
        PANEL_LABEL_IMAGE, pr,
        PANEL_CLIENT_DATA, 1,
        PANEL_NOTIFY_PROC, yes_no,
        0);

panel_create_item(panel, PANEL_MESSAGE,
        PANEL_ITEM_X, 95,
        PANEL_ITEM_Y, 75,
        PANEL_LABEL_STRING, "Continue with current action",
        0);

if ( isacont != BR_CONTINUE )
{
    panel_create_item(panel, PANEL_BUTTON,
        PANEL_ITEM_X, 25,
        PANEL_ITEM_Y, 120,
        PANEL_LABEL_IMAGE, &ca_button,
        PANEL_CLIENT_DATA, 0,
        PANEL_NOTIFY_PROC, yes_no,
        0);

    panel_create_item(panel, PANEL_MESSAGE,
                PANEL_ITEM_X, 95,
                PANEL_ITEM_Y, 145,
                PANEL_LABEL_STRING, "Abort current action",
                0);
        }

        window_fit(panel);
        window_fit(conf);

        r = (Rect *) window_get(conf, WIN_SCREEN_RECT);
        width = (int) window_get(conf, WIN_WIDTH);
        height = (int) window_get(conf, WIN_HEIGHT);
        left = (r->r_width - width)/2;
        top = (r->r_height - height)/2;
        if ( left < 0 )
            left = 0;
        if ( top < 0 )
            top = 0;
        window_set(conf, WIN_X, left, WIN_Y, top, 0);

        return(conf);
}
int
user_confirm(prompt,isacont,isanerr)
char *prompt;
int isacont;
int isanerr;
{
        Frame conf;
        int ans;

        conf = init_conf(prompt,isacont,isanerr);

        ans = (int) window_loop(conf);

        if ( isacont == BR_GETVALUE )
            strcpy(uvalue_store, (char *)panel_get_value(file_spec));
        else
            *uvalue_store = '\0';

        window_set(conf, FRAME_NO_CONFIRM, TRUE, 0);

        window_destroy(conf);

        return(ans);
}
char *
user_value()
{
        if ( *uvalue_store == NULL )
            return( NULL );
        else
            return( uvalue_store );
```

```
}
Funky_Stuff
  len=`wc -c < confirm.c`
  if [ $len !=     4369 ] ; then
    echo error: confirm.c was $len bytes long, should have been     4369
  fi
fi # end of overwriting check
if [ -f bbad.icon ]
then
  echo shar: will not over-write existing file bbad.icon
else
  echo shar: extracting 'bbad.icon',     1933 characters
  cat > bbad.icon <<'Funky_Stuff'
/* Format_version=1, Width=64, Height=64, Depth=1, Valid_bits_per_item=16
 */
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0003,0xFFFF,0xFFFC,0x0000,0x0002,0x0000,0x0004,0x0000,
    0x0002,0x0000,0x0004,0x0000,0x0002,0x0000,0x0004,0x0000,
    0x0002,0x0000,0x0004,0x0000,0x0002,0x0000,0x0004,0x0000,
    0x0002,0x0000,0x0004,0x0000,0x0002,0x001F,0xC004,0x0000,
    0x0002,0x003F,0xE004,0x0000,0x0002,0x0060,0x3004,0x0000,
    0x0002,0x0060,0x3004,0x0000,0x0002,0x0060,0x3004,0x0000,
    0x0002,0x0000,0x3004,0x0000,0x0002,0x0000,0x3004,0x0000,
    0x0002,0x0001,0xE004,0x0000,0x0002,0x0003,0xC004,0x0000,
    0x0002,0x0006,0x0004,0x0000,0x0002,0x0006,0x0004,0x0000,
    0x0002,0x0006,0x0004,0x0000,0x0002,0x0006,0x0004,0x0000,
    0x0002,0x0006,0x0004,0x0000,0x0002,0x0006,0x0004,0x0000,
    0x0002,0x0000,0x0004,0x0000,0x0002,0x0000,0x0004,0x0000,
    0x0002,0x0006,0x0004,0x0000,0x0002,0x0006,0x0004,0x0000,
    0x0002,0x0000,0x0004,0x0000,0x0002,0x0000,0x0004,0x0000,
    0x0002,0x0000,0x0004,0x0000,0x0003,0xFFFF,0xFFFC,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000
Funky_Stuff
  len=`wc -c < bbad.icon`
  if [ $len !=     1933 ] ; then
    echo error: bbad.icon was $len bytes long, should have been     1933
  fi
fi # end of overwriting check
```

```
if [ -f bblk.icon ]
then
  echo shar: will not over-write existing file bblk.icon
else
  echo shar: extracting 'bblk.icon',      1933 characters
  cat > bblk.icon <<'Funky_Stuff'
/* Format_version=1, Width=64, Height=64, Depth=1, Valid_bits_per_item=16
 */
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x007F,0xE000,0x0000,0x0000,0x0180,0x1800,0x0000,
    0x0000,0x0600,0x0600,0x0000,0x0000,0x1800,0x0180,0x0000,
    0x0000,0x2000,0x0040,0x0000,0x0000,0x3800,0x01C0,0x0000,
    0x0000,0x2600,0x0640,0x0000,0x0000,0x2180,0x1840,0x0000,
    0x0000,0x207F,0xE040,0x0000,0x0000,0x2000,0x0040,0x0000,
    0x0000,0x2000,0x0040,0x0000,0x0000,0x2000,0x0040,0x0000,
    0x0000,0x2000,0x0040,0x0000,0x0000,0x2000,0x0040,0x0000,
    0x0000,0x2000,0x0040,0x0000,0x0000,0x2000,0x0040,0x0000,
    0x0000,0x2000,0x0040,0x0000,0x0000,0x2000,0x0040,0x0000,
    0x0000,0x2000,0x0040,0x0000,0x0000,0x2000,0x0040,0x0000,
    0x0000,0x2000,0x0040,0x0000,0x0000,0x2000,0x0040,0x0000,
    0x0000,0x2000,0x0040,0x0000,0x0000,0x2000,0x0040,0x0000,
    0x0000,0x1800,0x0180,0x0000,0x0000,0x0600,0x0600,0x0000,
    0x0000,0x0180,0x1800,0x0000,0x0000,0x007F,0xE000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000
Funky_Stuff
  len=`wc -c < bblk.icon`
  if [ $len !=      1933 ] ; then
    echo error: bblk.icon was $len bytes long, should have been      1933
  fi
fi # end of overwriting check
if [ -f bcha.icon ]
then
  echo shar: will not over-write existing file bcha.icon
else
  echo shar: extracting 'bcha.icon',      1933 characters
  cat > bcha.icon <<'Funky_Stuff'
/* Format_version=1, Width=64, Height=64, Depth=1, Valid_bits_per_item=16
```

```
 */
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x1FFF,0xFFF8,0x0000,
    0x0000,0x1000,0x0008,0x0000,0x0000,0x11FF,0xFF88,0x0000,
    0x0000,0x1200,0x0048,0x0000,0x0000,0x1400,0x0028,0x0000,
    0x0000,0x1400,0x0028,0x0000,0x0000,0x1400,0x0028,0x0000,
    0x0000,0x1400,0x0028,0x0000,0x0000,0x1400,0x0028,0x0000,
    0x0000,0x1400,0x0028,0x0000,0x0000,0x1400,0x0028,0x0000,
    0x0000,0x1400,0x0028,0x0000,0x0000,0x1400,0x0028,0x0000,
    0x0000,0x1400,0x0028,0x0000,0x0000,0x1400,0x0028,0x0000,
    0x0000,0x1400,0x0028,0x0000,0x0000,0x1400,0x0028,0x0000,
    0x0000,0x1400,0x0028,0x0000,0x0000,0x1200,0x0048,0x0000,
    0x0000,0x11FF,0xFF88,0x0000,0x0000,0x1000,0x0008,0x0000,
    0x0000,0x1FFF,0xFFF8,0x0000,0x0000,0x0004,0x2000,0x0000,
    0x0000,0x0004,0x2000,0x0000,0x0000,0xFFFF,0xFFFF,0x0000,
    0x0000,0x8000,0x0001,0x0000,0x0000,0x8000,0x0001,0x0000,
    0x0000,0xFFFF,0xFFFF,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000
Funky_Stuff
  len=`wc -c < bcha.icon`
  if [ $len !=     1933 ] ; then
    echo error: bcha.icon was $len bytes long, should have been     1933
  fi
fi # end of overwriting check
if [ -f bdir.icon ]
then
  echo shar: will not over-write existing file bdir.icon
else
  echo shar: extracting 'bdir.icon',     1933 characters
  cat > bdir.icon <<'Funky_Stuff'
/* Format_version=1, Width=64, Height=64, Depth=1, Valid_bits_per_item=16
 */
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x01FE,0x0000,0x0000,0x0000,0xFE01,0xFFFF,0x0000,
    0x0000,0x8400,0x8001,0x0000,0x0003,0xFC00,0xFFFD,0x0000,
    0x0002,0x0000,0x0005,0x0000,0x000F,0xFFFF,0xFFF5,0x0000,
    0x0008,0x0000,0x0015,0x0000,0x0008,0x0000,0x0015,0x0000,
    0x0008,0x0000,0x0015,0x0000,0x0008,0x0000,0x0015,0x0000,
	0x0008,0x0000,0x0015,0x0000,0x0008,0x0000,0x0015,0x0000,
	0x0008,0x0000,0x0015,0x0000,0x0008,0x0000,0x0015,0x0000,
	0x0008,0x0000,0x0015,0x0000,0x0008,0x0000,0x0015,0x0000,
	0x0008,0x0000,0x0015,0x0000,0x0008,0x0000,0x0015,0x0000,
	0x0008,0x0000,0x0015,0x0000,0x0008,0x0000,0x0015,0x0000,
	0x0008,0x0000,0x0015,0x0000,0x0008,0x0000,0x0015,0x0000,
	0x0008,0x0000,0x0015,0x0000,0x0008,0x0000,0x0015,0x0000,
	0x0008,0x0000,0x0015,0x0000,0x0008,0x0000,0x0017,0x0000,
	0x0008,0x0000,0x0014,0x0000,0x0008,0x0000,0x001C,0x0000,
	0x0008,0x0000,0x0010,0x0000,0x000F,0xFFFF,0xFFF0,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000
Funky_Stuff
  len=`wc -c < bdir.icon`
  if [ $len !=     1933 ] ; then
    echo error: bdir.icon was $len bytes long, should have been     1933
  fi
fi # end of overwriting check
if [ -f bexe.icon ]
then
  echo shar: will not over-write existing file bexe.icon
else
  echo shar: extracting 'bexe.icon',     1933 characters
  cat > bexe.icon <<'Funky_Stuff'
/* Format_version=1, Width=64, Height=64, Depth=1, Valid_bits_per_item=16
 */
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0003,0xFFFF,0xF000,0x0000,0x0002,0x0000,0x1800,0x0000,
	0x0002,0x0000,0x1C00,0x0000,0x0002,0x0000,0x1E00,0x0000,
	0x0002,0x0000,0x1F00,0x0000,0x0002,0x0000,0x1F80,0x0000,
	0x0002,0x0000,0x1FC0,0x0000,0x0002,0x0000,0x1FE0,0x0000,
	0x0002,0x0000,0x1FF0,0x0000,0x0002,0x0000,0x1FF8,0x0000,
	0x0002,0x0000,0x1FFC,0x0000,0x0002,0x0000,0x0004,0x0000,
	0x0002,0x0000,0x0004,0x0000,0x0002,0x0000,0x0004,0x0000,
	0x0002,0x0000,0x0004,0x0000,0x0002,0x0000,0x0004,0x0000,
	0x0002,0x0000,0x0004,0x0000,0x0002,0x0000,0x0004,0x0000,
	0x0002,0x0000,0x0004,0x0000,0x0002,0x0000,0x0004,0x0000,
	0x0002,0x0000,0x0004,0x0000,0x0002,0x0000,0x0004,0x0000,
	0x0002,0x0000,0x0004,0x0000,0x0002,0x0000,0x0004,0x0000,
    0x0002,0x0000,0x0004,0x0000,0x0002,0x0000,0x0004,0x0000,
    0x0002,0x0000,0x0004,0x0000,0x0002,0x0000,0x0004,0x0000,
    0x0002,0x0000,0x0004,0x0000,0x0003,0xFFFF,0xFFFC,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000
Funky_Stuff
  len=`wc -c < bexe.icon`
  if [ $len !=     1933 ] ; then
    echo error: bexe.icon was $len bytes long, should have been     1933
  fi
fi # end of overwriting check
if [ -f bfil.icon ]
then
  echo shar: will not over-write existing file bfil.icon
else
  echo shar: extracting 'bfil.icon',     1933 characters
  cat > bfil.icon <<'Funky_Stuff'
/* Format_version=1, Width=64, Height=64, Depth=1, Valid_bits_per_item=16
 */
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0003,0xFFFF,0xF000,0x0000,0x0002,0x0000,0x1800,0x0000,
    0x0002,0x0000,0x1400,0x0000,0x0002,0x0000,0x1200,0x0000,
    0x0002,0x0000,0x1100,0x0000,0x0002,0x0000,0x1080,0x0000,
    0x0002,0x0000,0x1040,0x0000,0x0002,0x0000,0x1020,0x0000,
    0x0002,0x0000,0x1010,0x0000,0x0002,0x0000,0x1008,0x0000,
    0x0002,0x0000,0x1FFC,0x0000,0x0002,0x0000,0x0004,0x0000,
    0x0002,0x0000,0x0004,0x0000,0x0002,0x0000,0x0004,0x0000,
    0x0002,0x0000,0x0004,0x0000,0x0002,0x0000,0x0004,0x0000,
    0x0002,0x0000,0x0004,0x0000,0x0002,0x0000,0x0004,0x0000,
    0x0002,0x0000,0x0004,0x0000,0x0002,0x0000,0x0004,0x0000,
    0x0002,0x0000,0x0004,0x0000,0x0002,0x0000,0x0004,0x0000,
    0x0002,0x0000,0x0004,0x0000,0x0002,0x0000,0x0004,0x0000,
    0x0002,0x0000,0x0004,0x0000,0x0002,0x0000,0x0004,0x0000,
    0x0002,0x0000,0x0004,0x0000,0x0002,0x0000,0x0004,0x0000,
    0x0002,0x0000,0x0004,0x0000,0x0003,0xFFFF,0xFFFC,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000
Funky_Stuff
  len=`wc -c < bfil.icon`
  if [ $len !=     1933 ] ; then
    echo error: bfil.icon was $len bytes long, should have been     1933
  fi
fi # end of overwriting check
if [ -f brca.icon ]
then
  echo shar: will not over-write existing file brca.icon
else
  echo shar: extracting 'brca.icon',     1933 characters
  cat > brca.icon <<'Funky_Stuff'
/* Format_version=1, Width=64, Height=64, Depth=1, Valid_bits_per_item=16
 */
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x00FF,0xFFFF,0xFFFF,0xFF00,0x01FF,0xFFFF,0xFFFF,0xFF80,
	0x03FF,0xFFFF,0xFFFF,0xFFC0,0x07C0,0x0000,0x0000,0x03E0,
	0x0780,0x0000,0x0000,0x01E0,0x0700,0x0000,0x0000,0x00E0,
	0x0700,0x0000,0x0000,0x00E0,0x0700,0x0000,0x0000,0x00E0,
	0x0700,0x0000,0x0000,0x00E0,0x0700,0x0000,0x0000,0x00E0,
	0x0700,0x0000,0x0000,0x00E0,0x0700,0x0000,0x0000,0x00E0,
	0x0700,0x0000,0x0000,0x00E0,0x0700,0x0000,0x0000,0x00E0,
	0x0700,0x0000,0x0000,0x00E0,0x0700,0x0000,0x0000,0x00E0,
	0x0700,0x0000,0x0000,0x00E0,0x0700,0x0000,0x0000,0x00E0,
	0x0700,0x0000,0x0000,0x00E0,0x0707,0x0422,0x38F9,0x00E0,
	0x0708,0x8432,0x4481,0x00E0,0x0708,0x8A32,0x4481,0x00E0,
	0x0708,0x0A2A,0x4081,0x00E0,0x0708,0x0A2A,0x40F1,0x00E0,
	0x0708,0x1126,0x4081,0x00E0,0x0708,0x9F26,0x4481,0x00E0,
	0x0708,0x9122,0x4481,0x00E0,0x0707,0x1122,0x38F9,0xF0E0,
	0x0700,0x0000,0x0000,0x00E0,0x0700,0x0000,0x0000,0x00E0,
	0x0700,0x0000,0x0000,0x00E0,0x0700,0x0000,0x0000,0x00E0,
	0x0700,0x0000,0x0000,0x00E0,0x0700,0x0000,0x0000,0x00E0,
	0x0700,0x0000,0x0000,0x00E0,0x0700,0x0000,0x0000,0x00E0,
	0x0700,0x0000,0x0000,0x00E0,0x0700,0x0000,0x0000,0x00E0,
	0x0700,0x0000,0x0000,0x00E0,0x0700,0x0000,0x0000,0x00E0,
	0x0700,0x0000,0x0000,0x00E0,0x0700,0x0000,0x0000,0x00E0,
	0x0700,0x0000,0x0000,0x00E0,0x0780,0x0000,0x0000,0x01E0,
	0x07C0,0x0000,0x0000,0x03E0,0x03FF,0xFFFF,0xFFFF,0xFFC0,
    0x01FF,0xFFFF,0xFFFF,0xFF80,0x00FF,0xFFFF,0xFFFF,0xFF00,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000
Funky_Stuff
  len='wc -c < brca.icon'
  if [ $len !=     1933 ] ; then
    echo error: brca.icon was $len bytes long, should have been     1933
  fi
fi # end of overwriting check
if [ -f brok.icon ]
then
  echo shar: will not over-write existing file brok.icon
else
  echo shar: extracting 'brok.icon',     1933 characters
  cat > brok.icon <<'Funky_Stuff'
/* Format_version=1, Width=64, Height=64, Depth=1, Valid_bits_per_item=16
 */
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x00FF,0xFFFF,0xFFFF,0xFF00,0x01FF,0xFFFF,0xFFFF,0xFF80,
    0x03FF,0xFFFF,0xFFFF,0xFFC0,0x07C0,0x0000,0x0000,0x03E0,
    0x0780,0x0000,0x0000,0x01E0,0x0700,0x0000,0x0000,0x00E0,
    0x0700,0x0000,0x0000,0x00E0,0x0700,0x0000,0x0000,0x00E0,
    0x0700,0x0000,0x0000,0x00E0,0x0700,0x0000,0x0000,0x00E0,
    0x0700,0x0000,0x0000,0x00E0,0x0700,0x0000,0x0000,0x00E0,
    0x0700,0x0000,0x0000,0x00E0,0x0700,0x0000,0x0000,0x00E0,
    0x0700,0x0000,0x0000,0x00E0,0x0700,0x0000,0x0000,0x00E0,
    0x0700,0x0000,0x0000,0x00E0,0x0700,0x0000,0x0000,0x00E0,
    0x0700,0x0000,0x0000,0x00E0,0x0700,0x001C,0x4400,0x00E0,
    0x0700,0x0022,0x4800,0x00E0,0x0700,0x0022,0x5000,0x00E0,
    0x0700,0x0022,0x6000,0x00E0,0x0700,0x0022,0x5000,0x00E0,
    0x0700,0x0022,0x4800,0x00E0,0x0700,0x0022,0x4800,0x00E0,
    0x0700,0x0022,0x4400,0x00E0,0x0700,0x001C,0x4400,0x00E0,
    0x0700,0x0000,0x0000,0x00E0,0x0700,0x0000,0x0000,0x00E0,
    0x0700,0x0000,0x0000,0x00E0,0x0700,0x0000,0x0000,0x00E0,
    0x0700,0x0000,0x0000,0x00E0,0x0700,0x0000,0x0000,0x00E0,
    0x0700,0x0000,0x0000,0x00E0,0x0700,0x0000,0x0000,0x00E0,
    0x0700,0x0000,0x0000,0x00E0,0x0700,0x0000,0x0000,0x00E0,
    0x0700,0x0000,0x0000,0x00E0,0x0700,0x0000,0x0000,0x00E0,
    0x0700,0x0000,0x0000,0x00E0,0x0700,0x0000,0x0000,0x00E0,
    0x0700,0x0000,0x0000,0x00E0,0x0780,0x0000,0x0000,0x01E0,
    0x07C0,0x0000,0x0000,0x03E0,0x03FF,0xFFFF,0xFFFF,0xFFC0,
    0x01FF,0xFFFF,0xFFFF,0xFF80,0x00FF,0xFFFF,0xFFFF,0xFF00,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000
Funky_Stuff
  len='wc -c < brok.icon'
  if [ $len !=     1933 ] ; then
    echo error: brok.icon was $len bytes long, should have been     1933
  fi
fi # end of overwriting check
if [ -f brow.icon ]
then
  echo shar: will not over-write existing file brow.icon
else
  echo shar: extracting 'brow.icon',     1933 characters
  cat > brow.icon <<'Funky_Stuff'
/* Format_version=1, Width=64, Height=64, Depth=1, Valid_bits_per_item=16
 */
        0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,
        0xC000,0x0000,0x0000,0x0003,0xC000,0x0000,0x0000,0x0003,
        0xC000,0x0000,0x0000,0x0003,0xC000,0x7C00,0x0000,0x0003,
        0xC03F,0x83FC,0x3FFF,0xC003,0xC020,0x0004,0x2000,0x6003,
        0xC0FF,0xFFF4,0x2000,0x5003,0xC080,0x0014,0x2000,0x4803,
        0xC080,0x0014,0x2000,0x4403,0xC080,0x0014,0x2000,0x4203,
        0xC080,0x0014,0x2000,0x7F03,0xC080,0x0014,0x2000,0x0103,
        0xC080,0x0014,0x2000,0x0103,0xC080,0x0014,0x2000,0x0103,
        0xC080,0x0014,0x2000,0x0103,0xC080,0x0014,0x2000,0x0103,
        0xC080,0x0014,0x2000,0x0103,0xC080,0x0014,0x2000,0x0103,
        0xC080,0x001C,0x2000,0x0103,0xC080,0x0010,0x2000,0x0103,
        0xC0FF,0xFFF0,0x3FFF,0xFF03,0xC000,0x0000,0x0000,0x0003,
        0xC000,0x0000,0x0000,0x0003,0xC0FF,0xFF00,0x3FFF,0xC003,
        0xC080,0x0180,0x2000,0x6003,0xC080,0x0140,0x2000,0x7003,
        0xC080,0x0120,0x2000,0x7803,0xC080,0x0110,0x2000,0x7C03,
        0xC080,0x0108,0x2000,0x7E03,0xC080,0x01FC,0x2000,0x7F03,
        0xC080,0x0004,0x2000,0x0103,0xC080,0x0004,0x2000,0x0103,
        0xC080,0x0004,0x2000,0x0103,0xC080,0x0004,0x2000,0x0103,
        0xC080,0x0004,0x2000,0x0103,0xC080,0x0004,0x2000,0x0103,
        0xC080,0x0004,0x2000,0x0103,0xC080,0x0004,0x2000,0x0103,
        0xC080,0x0004,0x2000,0x0103,0xC0FF,0xFFFC,0x3FFF,0xFF03,
        0xC000,0x0000,0x0000,0x0003,0xC000,0x0000,0x0000,0x0003,
        0xC000,0x0000,0x0000,0x0003,0xC000,0x0000,0x0000,0x0003,
        0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000
Funky_Stuff
  len='wc -c < brow.icon'
  if [ $len !=     1933 ] ; then
    echo error: brow.icon was $len bytes long, should have been     1933
  fi
fi # end of overwriting check
```

7

CUMULATIVE INDEX: 1987

Index

Revision History

Revision	*Date*	*Comments*
FINAL	September 1987	Eighth issue of Software Technical Bulletin (Software Information Services).

Bulk Rate
U.S. Postage
PAID
Permit No. 515
Mountain View, CA

Corporate Headquarters
Sun Microsystems, Inc.
2550 Garcia Avenue
Mountain View, CA 94043
415 960-1300
TLX 287815