UNIX[†] 4.3BSD SOURCE RELEASE NOTE

For Source Licensees

Release 5.0

490277 Rev. A

About this Release Note

This Release Note describes the current source release from Integrated Solutions. This Note accompanies the general Release Note that describes binary releases. Refer to the general Release 5.0 Release Note for information on release contents and installation procedures. When you are finished with this Note, please insert it in the back of your *UNIX 4.3BSD System Administrator Guide* (SMM:1)[††] for future reference.

This Note contains the following subsections:

1.1 Format of the Source Release Tape

The source release tape contains two tape files, as shown in Table 1-1: the */usr/src* file system and the X window system, a public-domain package for graphics workstations. Graphics customers receive both graphic files and the X window system. Non-graphics customers receive only the X window system.

NOTE

> The source release tape includes some files related to the Network File System (NFS [†††]). The files on the source release tape do not allow you to use NFS without additional software under separate license from ISI.

NOTE

> The source release tape does not include ISI graphics source files, other than the public-domain X sources. ISI graphics sources are on a separate tape. Use the following procedure to install the graphics source release tape.

[†] UNIX is a registered trademark of AT&T in the USA and other countries. 4.3BSD was developed by the Regents of the University of California (Berkeley), Electrical Engineering and Computer Sciences Departments.

[††] References of the form (XXX:N) refer to a section of the seven-volume UNIX 4.3BSD Reference Set. SMM:1, for example, refers to the first section of the *UNIX System Manager's Manual* (SMM).

[†††] NFS (the Sun Network File System) is a product created and developed by Sun Microsystems, Inc. NFS is a separate product option.

Table 1-1. Format of the UNIX Source 5.0 Release Tape

File	Contents	Description
0	**src**	**tar** (1) of */usr/src* file system
1	**X**	**tar** (1) of X window system sources

1.2 Installation of the Source Release Tape

Use this procedure to install the new source release from the source release tape.

1. Load the source release tape and rewind it with the command

```
mt rewind
```

2. Enter these commands:

```
cd /
tar xp
```

The tar files on the source release tape are created relative to the root (/) directory. These commands will extract the */usr/src* files and install them on your system.

3. If you want to extract the X window system sources, load the source tape and enter these commands:

```
cd /
mt rew
mt fsf 1
tar xp
```

These commands extract the */usr/src/graphics/X* files and install them on your system.

4. Kernel sources reside in the directory */usr/src/sys.* The binary distribution includes kernel configuration files in */usr/sys.* The */usr/sys* directory is redundant after you have installed sources.

If you have edited or added any files in */usr/sys*, move them now to */usr/src/sys.* Remove the directory */usr/sys* and, in its place, make a link to the */usr/src/sys* directory. Use these commands:

```
rm -r /usr/sys
ln -s /usr/src/sys /usr/sys
```

This completes the installation procedure.

1.3 The Kernel Debugger

The kernel debugger is a kernel-resident machine-level debugger. This is a special ISI debugger, with syntax unlike other UNIX debuggers.

The kernel debugger runs from the system console. When the kernel debugger runs, it turns off all interrupts; therefore, there should be no other system activity during debugging.

The DEBUGGER option in the kernel configuration file enables the kernel debugger. To enter the debugger, type "^_" (CTRL-_); hold down the CTRL key and type an underscore. You should now see the debugger prompt (d:) and you can issue debugger commands. To exit the debugger, type "G" or another "^_".

The debugger, when enabled, will come up automatically after panics. The system will reboot after you exit the debugger.

The **config** (8) program causes a command **ksymbol** (8) to be invoked when making a kernel. **ksymbol** configures the kernel's internal symbol tables to allow symbolic address interpretation during debugger sessions. Any address that begins with an alphabetic character will be interpreted as a symbolic address. The address "A2", for example, should therefore be expressed as "0A2" to force numerical interpretation.

In addition, the debugger has a limited capability for expression evaluation. Note that precedence is right to left in the evaluation.

The debugger will catch invalid kernel accesses. However, the kernel may panic if you write into text and data space incorrectly.

Table 1-3 shows the commands for the kernel debugger.

Table 1-2. Kernel Debugger Commands

Command	What It Does
I *address*	Insert Breakpoint. Insert a breakpoint in the kernel text space. The debugger supports up to 16 breakpoints.
K *address*	Delete Breakpoint. Delete a breakpoint. "**K** *" deletes all breakpoints.
?	Display Breakpoints. Show all existing breakpoints.
A *address*	Disassemble. Display the contents of *address* interpreted as 68010/68020 instructions.
!	Continue to Disassemble. Same as **A**, moving to the next memory address. Successive "**!**" commands will move through memory.
D *address*	Dump Memory. Display 256 bytes of memory at *address*, both in hex and as ASCII characters.

(continued on next page)

Table 1-3. Kernel Debugger Commands *(continued)*

Command	What It Does
E *address*	Examine/Change Memory. Examine memory according to the display format (F), and allow change if desired.
W *address*	Write Memory. Write into memory, without examining first.
F *format*	Set Display Format. Set the default display format to one of the following: **F B** Byte format **F W** Word format **F L** Long format
G *address*	Go. Continue execution at *address*.
R[*modifier*] *value*	Display/Set Register(s). Show the contents of the registers. These are the available modifiers: **R** displays register contents. **RP** loads *value* into the pc register. **RS** loads *value* into the status register. **RD***n* loads *value* into data register *n* (0-7). **RA***n* loads *value* into address register *n* (0-7). *value* can be any of the following: • a hexadecimal number • *@address*—the contents of the given *address* • **R**[**A**,**D**]*n*—the contents of the given register Examples: **RP 1C+2** loads the number value 1E into the pc register. **RS @0D2** loads the contents of memory address D2 into the status register. **RD2 RA4** loads the contents of address register 4 into data register 2.
S <ESC>	Single Step. Displays memory addresses consecutively.
B	Reboot. Reboots the system from within the debugger.