Technical Support Bulletin

Issue 1984-2
17 February 1984

Technical Support Bulletin

Sun Microsystems, Inc.

Issue 1984-2
17 February 1984

2.1) Correction to Item number 1.13, Issue 1984-1.

Description:

Item number 1.13 in Issue 1984-1 of the Technical Support Bulletin (13 January 1984) contained a typographical error. On page 8, in the second line from the top of the page, the phrase

> Sometimes upon checking the disk tables, ...

should read

> Sometimes upon checking the disk cables, ...

2.2) What to do when tip says "all ports busy"

Description:

Sometimes, when attempting to run the "tip" command, you will get the response "all ports busy". If this happens, here are a couple of things to try:

a) Do an "ls -l" on tip's output device (specified by the **dv** field in /etc/remote). Its permissions should be rw-rw-rw. If not, do "chmod 666 /dev/*dddd*" where *dddd* is the name of the device. Now try running **tip** again.

b) If **tip** terminated abnormally last time it was run (for example, the user did a "kill -9" on it, or the system crashed) then there may be a spurious lock file in /usr/spool/uucp. Look for a file name of the form "LCK..*dddd* where *dddd* is the name of tip's output device, as above. Become super-user and remove this file. Now try running **tip** again.

See Also:

tip(1) manual page
remote(5) manual page

2.3) A note about Multibus memory.

Description:

A few customers are confused about what happens to Multibus memory when a system is upgraded to Sun-2 hardware. On Sun 1.5 and earlier hardware (UNIX release 0.4 and earlier), systems with disk, 9-track tape, or other Multibus DMA devices were required to have at least one Multibus memory board. Usually this was either a Chrislin 128KB or 256KB Multibus memory board, or a Sun 256KB Multibus memory board (which is the same thing

as a Sun quarter-inch tape controller board).

When these systems are upgraded to Sun-2's (UNIX Release 1.0 and later), any Multibus memory in the system must be removed or disabled. On Sun-2 systems, Multibus DMA devices use DVMA (direct virtual memory access) to do DMA directly into the high-speed memory on the P2 bus. If Multibus memory is present, the user will be unable to boot. The Sun-2 *Upgrade Installation Guide* contains detailed instructions on how Multibus memory should be removed or disabled.

See Also:

Upgrade Installation Guide for Model 100 and Model 150
System Manager's Manual

2.4) Changing baud rates on serial lines.

Description:

The **stty** command can be used to change the baud rate on a serial line. For example, the command

```
stty 1200 >/dev/ttyb
```

should change the baud rate of ttyb to 1200. However, if you then type

```
stty >/dev/ttyb
```

you will find its baud rate unchanged. The reason is that when a tty device is closed, the kernel resets all of its characteristics, including baud rate, to standard settings. This resetting will not occur so long as any process still holds the serial port open. For example, the command

```
(stty 1200; sleep 100000) >/dev/ttyb &
```

will start a process in background which will set the baud rate of ttyb to 1200, then sleep for 100000 seconds while continuing to hold ttyb open. As long as this process continues to run in background, the characteristics of ttyb will not be reset by the kernel, and may be set to whatever the user desires.

This approach is not a good general solution, however, since if the sleeping process dies for any reason, you will again be unable to set tty characteristics. A better approach is to have any program which reads or writes the serial port first open the device, then set its characteristics (for example, by calling ioctl), finally doing the desired i/o. When the program is finished using the device it will close it, causing it to revert back to its default characteristics.

The UNIX programs **tip** (terminal interface protocol), **lpd** (the line printer daemon), and **getty** (which logs users in on serial lines) all work in this fashion. For **tip** the baud rate is specified by an entry of the form "br#1200" in the file /etc/remote. Similarly, for **lpd** an entry of the form "br#1200" is placed in the file /etc/printcap. In the case of **getty** (*in Release 1.0 and later*), the baud rate is specified in the file /etc/gettytab by the capabilities **is**, **os**, and **sp**.

See Also:

System Manager's Manual (Release 1.1 and later) on connecting a printer.
stty(1) manual page.
printcap(5) manual page.
remote(5) manual page.

gettytab(5) manual page.

2.5) Sun UART initialization can be changed by a PROM patch.

Applies to: 0.4 only.
Description:

Certain very low-level serial line characteristics (such as number of stop bits transmitted) are initialized by the PROM monitor at power-up and cannot be changed by UNIX system calls. However, the UART registers are set from an array in the PROM which can actually be patched (e.g., by reading-out the PROM on a PROM burner, changing the affected locations, and burning a new PROM). For Revision J/K PROMs, starting at location 0x3256 you find the following byte sequence:

02 00 04 44 03 e1 05 ea 01 00

These locations can be examined on a Sun 1.5 workstation with the monitor command "e203256". Each pair of bytes indicates a control register in the Intel 8274 dual UART, followed by the value which is loaded into it. For example, 0x44 is loaded into register 4; 0xe1 is loaded into register 3; and so on. Referring to the Intel data sheet for the 8274, the user can change the initial state of the UART by changing these values.

Similar patches are possible in the Sun-2 PROMs; it is even possible to change the default baud rate. If anyone actually needs this information, contact Technical Support and we will get the details from Engineering.

2.6) Why you get "Mount device busy" at boot.

Applies to: Network disk servers only.
Description:

When booting a workstation configured as a network disk server, you probably have noticed the message

/dev/xy0g on /pub: Mount device busy

which is normal on networked systems. The first line of /etc/rc contains a command of the form:

/etc/mount -r /dev/xy0g /pub

which was inserted by the *setup* command during system installation. This line is necessary because many of the commands used by /etc/rc have been moved to /pub in a networked system, so the /pub file system must be mounted. Later on in /etc/rc is the command:

/etc/mount -a

which means "mount every file system listed in /etc/fstab". It is this command which generates the "mount device busy" message.

See Also:
mount(8) manual page
The file **/etc/rc**

2.7) Take plastic bag off of older D84 drives.

Applies to: Some D84 (Fujitsu M2312K) disk drives shipped in late 1982/early 1983.
Description:

Back in the "good old days" when we began to ship D84 disk drives, a list of media defects in a plastic bag was taped directly onto the top of the disk drive, inside the outer sheetmetal cabinet. The information in this defect list is based on a type of disk controller other than SMD, so it is not usable on Suns. Unfortunately, the plastic bag obstructs vents on the drive which has caused a few drives to overheat and malfunction. Most of these have already been caught and corrected by Field Support, but any time a customer is opening up the enclosure on an old D84 you should advise him to remove and discard the defect list.

2.8) Restoring a file from distribution tape — some examples.

Description:

Sometimes a customer accidentally removes or damages a system file which he wishes to restore off of the UNIX boot tape. If the file normally resides in root file system directories like /sys, /lib, /, /bin, or /etc, it is extracted during the system installation step called *Loading the Root File System*, immediately after booting the mini-UNIX system. (This is true even on a network disk server; these files are extracted during this step and later moved to /pub). The shell script which extracts the files is called /xtr and is normally found only in the mini-UNIX file system.

If the file normally resides in /usr file system directories like /usr/bin, /usr/lib, or /usr/ucb, then it is extracted by the *setup* program using one of several shell scripts: /etc/xtrusr, /etc/rxtrusr, or /etc/xtrusrnd.

By examining these shell scripts, it is fairly straightforward to figure out how these file systems are stored on the boot tape(s) and how to extract them.

For example, suppose you have accidentally deleted the file /usr/ucb/lpr and wish to restore it from tape. Using the 1.0 Release version in this example, you examine the script /etc/xtrusr and find the following commands (here in condensed form):


```
. . .
: Make the file system and restore files with tar
. . .
cd /usr
mt -f /dev/nr${tape}0 rew
if [ "$tape" != "mt" ]
then
        echo "Insert second tape and hit RETURN."
        read x
        mt -f /dev/nr${tape}0 rew
        mt -f /dev/nr${tape}0 fsf 1
        bs=200
else
        mt -f /dev/nr${tape}0 fsf 8
        bs=20
fi
echo Extracting the /usr files
tar xfbp /dev/nr${tape}0 $bs
. . .
echo "DONE.  /usr loaded."
```

The shell variable appearing as "$tape" or "${tape}" expands to "mt" if you have 9-track tape, or to "ar" or "st" (as appropriate) for cartridge. Based on the above script, you would enter the following commands to restore the file from 9-track tape:

```
cd /usr
mt -f /dev/nrmt0 rew
mt -f /dev/nrmt0 fsf 8
tar xfbp /dev/nrmt0 20 ucb/lpr
```

Notice the additional argument to **tar** specifying the name of the file you want to extract. If you have SCSI cartridge tape, again referring to the above script you would use the following commands:

```
cd /usr
(Insert the second cartridge.)
mt -f /dev/nrst0 rew
mt -f /dev/nrst0 fsf 1
tar xfbp /dev/nrst0 200 ucb/lpr
```

As a final example, if you want to restore the file /etc/fsck from quarter-inch tape (device "ar") you could first follow the instructions in the *System Manager's Manual* for loading the mini-UNIX file system (note that this will *not* damage any existing files), then find the following lines in /xtr:

```
. . .
/etc/mount /dev/${disc}0a /a
mt -f /dev/r${tape}0 rew
cd /a
/etc/restore rsf 6 /dev/r${tape}0
. . .
echo 'Root filesystem extracted'
```

"${disc}" is the name of your disk (like xy or sd) while "${tape}" is the name of your tape, in this example assumed to be "ar". Your next step (if you had loaded mini-UNIX) is to abort and do a normal UNIX boot. It is not necessary for you to do the **mount** since your root file system is already mounted on / as a normal part of the boot. You would enter the following commands:

```
mt -f /dev/rar0 rew
cd /
/etc/restore xsf 6 /dev/rar0 etc/fsck
```

Note that the 'x' option is used in restore rather than the 'r' option, since we only want to restore one file. Note also the use of the 's' option (skip to specified file on tape) ***which is undocumented in the Release 1.0 manual page*** for restore.

Now that you know about the 's' option, you can avoid all the bother of loading mini-UNIX. Simply refer to the chapter called "Installing UNIX for the First Time" in the *System Manager's Manual.* The first section, titled "What is on the Distribution Tape?", tells you that File 6 (the sixth file on the tape) is the complete root file system in *dump* format. You can then compose the above *restore* yourself.

See Also:

System Manager's Manual
tar(1) manual page
dump(8)
restore(8)

2.9) Interpreting disk errors "restore" or "retry" vs. "failed".

Description:

Occasionally, a customer may see an error message on the system console which includes one of the words "retry" or "restore" plus the name of a disk device, a block number, and some other information. For example:

```
xy0g: read restore blk 128fe ....
xy0a: reset retry blk 1e06 ....
```

If one of these messages occurs in isolation, it indicates a "soft" error in the sense that the disk controller was able to recover from the error by successfully retrying the i/o operation. In the case of a "hard" unrecoverable error, you will get a sequence of retry and restore messages (perhaps ten of them) followed by a similar message with the word "failed" instead of "retry".

If a customer is experiencing an occasional soft error (say once a week) it is probably no cause for concern. However, if soft errors occur frequently or if hard errors ever occur, this should be treated as a hardware problem and should be corrected as soon as possible. The cause often involves (in order of decreasing frequency): bad or loose disk cables; disk drive improperly set up or grounded; defective disk controller or wrong firmware; defective disk drive. Note that unreadable blocks may still be unreadable after correcting the hardware problem; in this case, the *diag* utility should be used to reformat the offending blocks (or, in worst case, the whole disk).

See Also:

Item 1.13, "Fixing bad disk blocks due to loose cables"
in *Tech Support Bulletin* Issue 1984-1.

2.10) User bus errors vs. parity errors vs. panics

Description:

Perhaps the most commonly-seen error message on a Sun (especially on 0.3 and 0.4 systems, which contained more software bugs) is the phrase USER BUS ERROR accompanied by several lines of seemingly cryptic debugging information. The error message may alternatively say "Protection exception", "Segmentation violation", or "Bus error". Often the error message is output to the system console, even if the instigator is logged into another port or running in background. If the phrase "core dumped" appears, then the user will find a file named **core** has been created in the working directory of the process which encountered the error. This file can be used in conjunction with a debugger (like *adb*) to diagnose the error. In release 1.0 and later, the lines of cryptic-looking debugging information is suppressed from the console error message, but is still available in the core file. However, in all these cases UNIX keeps running. These messages do *not* indicate a UNIX system crash, but rather that a particular process has encountered a fatal error which is almost always due to a programming bug. The user should look for the name of a program followed by a colon occurring at the beginning of the line, to indicate which program encountered the error. For example, many VAX programs attempt to dereference a null (zero) pointer, which works by accident on the VAX but is illegal on the Sun. These errors are characterized by the phrase ACCESS ADDRESS 0 (or a small number like 8) appearing in the error message. Most such bugs in UNIX commands have been fixed by Release 1.0.

A second class of error messages look similar to those described above, but contain a line beginning with the word "Panic:", for example "Panic: freeing free inode". These errors originate from the UNIX kernel itself. UNIX does not continue running; instead the system resets and reboots itself. One of the side effects of the reboot is that the screen blacks out for perhaps ten seconds (like a power-on reset) and its contents are lost. You can still recover the "Panic:" message by running */etc/dmesg* (*/usr/etc/dmesg* in Release 1.0) immediately after reboot before the next crash or shutdown, or by looking through /usr/adm/messages. This message indicates the cause of the panic and can be useful for diagnosing the problem. Call Technical Support if you need help.

Another class of error messages contain the word "parity". These *always* indicate a hardware failure (a memory parity error) which should be corrected. Refer these to Field Service.

Unfortunately, an exhaustive discussion of possible errors and how to debug them is beyond the scope of this Bulletin. If you encounter any of the above situations, try to write down as much of the context as possible, and call Technical Support for further assistance.

2.11) "_sobuf" no longer defined

Applies to: 1.0 and beyond
Description:

In VAX versions of UNIX, *stdio* (standard i/o) defines a global variable called **_sobuf** (standard output buffer) which, as its name suggests, is used as a buffer for *stdout*. On the Sun, **_sobuf** no longer exists. Instead, the standard output buffer is allocated dynamically (by a call to *malloc*) the first time that output is actually done.

These facts have led to two classes of complaints from customers:

1) Many VAX programs contain the line

```
setbuf(stdout, _sobuf);
```

which fails on the Sun since _sobuf is undefined. The solution to this is simply to remove the *setbuf* call; it's not needed.

2) Some users are upset because *stdio* now invokes *malloc*. They previously have assumed that successive calls to *sbrk* will allocate contiguous chunks of memory, and the call to *malloc* invalidates this assumption. The solution is to explicitly call *setbuf* at the beginning of the program (before **stdout** is written to), passing *setbuf* a user-declared buffer. For example:

```
#include <stdio.h>
static char my_sobuf[BUFSIZ];
main()
{
        setbuf(stdout, my_sobuf);
        /* Now output to stdout won't call malloc */
        ...
}
```

2.12) A note about "pw" (page width) in *printcap*

Description:

The documentation for *printcap*(5) indicates that the *pw* field sets the page width (i.e., line length) in units of characters. However, this in itself does not affect the width of your output. The specified width is passed to your output filter (if you have one) as a command line option of the form **-w***width*. It is up to the output filter to do any additional processing on lines exceeding the specified width. If you don't have an output filter, lines will be output "as is" regardless of their length and regardless of your *pw* specification.

2.13) A note about using ttya for system console

Applies to: Systems using an ASCII terminal on serial port A for console.
Description:

When a terminal is used as the system console, there are two UNIX files which you need to look at:

- In */etc/ttys*, be sure that logins are enabled only for /dev/console, not for both /dev/console and /dev/ttya. If this is not done, the system will fail to come up multi-user. If necessary, boot single-user and do the edit.
- In */etc/ttytype*, you should set the correct terminal type for /dev/console. This will permit screen editors such as *vi* to function properly.

If you want to switch back to using the Sun keyboard and screen, you will have to edit these files again. You may not be able to run *vi* in single-user mode (unless /usr/ucb is mounted); in this case use *ed*. If you do run *vi* in single-user mode, you may have to set the terminal type to "sun" or whatever terminal you're using. In the Bourne shell (which is what runs in single-user mode) use the following commands to set the terminal type:

```
TERM=sun    ### or whatever your terminal type is
export TERM
```

The equivalent in the c-shell is "set term=sun" (although in normal use it is unnecessary to set the terminal type).

See Also:

ttys(5), ttytype(5) manual pages
Item 1.8, "getty baud rate table for 0.4"
in *Tech Support Bulletin* Issue 1984-1.

2.14) rexec broken until 1.1 — use rcmd instead

Applies to: Releases 0.4 and 1.0
Description:

The *rexec* library function does not work properly in releases before 1.1. In the meantime, users can use *rcmd* instead, although it is not entirely equivalent.

See Also:

rcmd(3N) manual page
rexec(3N) manual page

2.15) Making room on file systems

Description:

A user who gets "Device full" messages on the system console should suspend or kill the currently-executing program and try to make more room on the affected file system, either by deleting some files or moving some files to a different file system. Here are some hints that may be useful:

1) The command **ls -s** gives the size of each file in kilobytes. To find the ten largest files in the current working directory, type

```
ls -s | sort -nr | head
```

2) If UNIX crashes, it saves a core dump (roughly equal in size to the amount of physical main memory on your system) in a portion of the swap area on the disk. At the next reboot, *savecore* (which is run from /etc/rc) will copy this core dump into files in the directory */usr/crash*. If you have a problem which leads to a series of system crashes, then /usr/crash will keep accumulating dump files until the file system fills up. The

affected file system will typically be the /usr partition (xy0g or sd0g) on standalone systems or the root partition on servers and diskless clients. There is a convenient way to avoid this problem: *savecore* reads from a file named *minfree* in /usr/crash. If the file system contains less free space in kilobytes than the number contained in *minfree*, the core file will not be saved.

3) The directory */usr/adm* contains accounting information. Check there to see if any of the files are growing inordinately large. In particular, */usr/adm/lastlog* can grow enormously large if large user numbers (conjecture: > 32K) are used in */etc/passwd*. Advise customers to avoid using very large user numbers. You can disable login accounting by removing the file */usr/adm/wtmp*. You can disable process execution accounting by removing the file */usr/adm/acct*. Re-enable accounting by recreating the respective file with length zero.

4) You can look for obsolete junk in */tmp* and */usr/tmp*. Be careful not to delete temporary files from */tmp* which are in use by currently active processes (like file names starting with **Ex** or **Rx**, used by *vi*). Also look for obsolete files in the subdirectories of /usr/spool such as /usr/spool/mail and /usr/spool/uucppublic.

5) To find out which files were most recently created, use **ls -t** (lists files in order of most-recently-created first instead of alphabetically). This can be used in combination with other *ls* options like **l** and **s**.

6) Use the *du* command to see where disk space is being used. For example, **du /usr** gives the disk usage (in kilobytes) of every directory underneath **/usr**.

7) If you want to move a file or directory but programs expect to find it in a particular place, you can make a symbolic link. For example, if you want to move the mail spool directory */usr/spool/mail* onto a spare disk partition mounted on **/usr2**, you could do the following (as super-user):

```
# mkdir /usr2/mail
# chmod 777 /usr2/mail
# cd /usr/spool
# mv mail mail.bak
# ln -s /usr2/mail
# mv mail.bak/* mail
# rmdir mail.bak
```

Warning: some system files like */vmunix* and */dev* shouldn't be symbolically linked. Moving them from their accustomed place can cause confusion or havoc. If you're not sure a file can be safely moved, don't move it.

8) The directory /sys contains the kernel configuration files. If you know you won't be needing these for awhile, you can remove /sys and save about 1.5 megabytes in the root partition. You may want to back it up on a *tar* tape or in another disk partition.

9) On a disk server, *setup* does not allocate much space for user files on the server itself. By default partition **g**, which is mounted on */usr* in a stand-alone system, is dedicated to *nd* partitions on a server. Let us assume that your server has a Xylogics controller and a large disk. If you know you will want to have substantial user files on the server, you can create a separate partition for /usr by one of the following two methods:

   a) When labelling the disk in *diag*, use the **partition** command to create a non-standard label for your disk. In this label, reduce the size of partition **xy0g** and re-allocate the space to another label partition (like **xy0h**) which will be mounted on /usr. After bringing up UNIX, add an entry to /etc/fstab for mounting

/dev/xy0h on /usr. Then run *newfs* on **/dev/rxy0h**. If you have set up /etc/fstab correctly then the next time you boot, partition **h** will be checked and mounted on /usr.

b) When running *setup*, don't allocate all of partition **g** to /pub and client partitions. *Setup* will put an entry for the left-over space in /etc/nd.local. This left-over space can be mounted as /dev/ndl*x* similarly to partition **h** in case (a) above.

Note that you cannot run *newfs* on an nd partition; after editing /etc/nd.local and running /etc/nd, you must use *mkfs* to create the file system. More details can be found in the Technical Note entitled "A Few Words About Network Disks", available from Tech Support.

Note also that you cannot put an entry for this *ndl* partition in /etc/fstab, because at the time *fsck* is run during booting, */etc/nd* has not yet been run, so the *ndl* partition is not yet defined to the kernel. The solution is to add lines like the following to the end of /etc/rc.local:

```
echo Checking /dev/ndl6 > /dev/console
/etc/fsck -p /dev/ndl6 > /dev/console
/etc/mount /dev/ndl6 /usr > /dev/console
```

This assumes, of course, that *ndl6* is the correct *ndl* partition number for the left-over space as defined in /etc/nd.local.

Both of these approaches (a) and (b) assume that you allocated a spare partition at installation time. If you didn't, you will have to save your files on tape and restore them after reallocating the disk as described above.

See Also:

System Manager's Manual
ln(1), ls(1), df(1), du(1), nd(4), fstab(5), ac(8), sa(8), savecore(8) manual pages
"A Few Words About Network Disks (nd)" (Sun Microsystems Technical Note)

2.16) A couple of hints on mail installation

Applies to: Release 0.3 and later
Description:

The mail delivery program *sendmail* uses a configuration file named /usr/lib/sendmail.cf. If you look in /usr/lib, you will also see files named sendmail.generic.cf and sendmail.domain.cf. These are model configuration files, one of which is to be copied to sendmail.cf (possibly with some editing). The domain version is normally installed on only one machine on a Sun network, the "lord of the domain" machine which has all the other machines in its hosts.equiv file and may be capable of reaching outside hosts by uucp. All the other hosts on the network would use the "generic" version of sendmail.cf. Please read the section titled "Setting Up the Mail System" under "System Set-up and Operation" in the Release 1.0 *System Manager's Manual.* This information also applies to releases 0.3 and 0.4.

When you are debugging mail system problems, it is useful to run **mail** with the **-v** option. This causes *mail* to invoke *sendmail* with the **-v** (verbose) flag, which displays useful information about how the message is being routed to the addressee. Note: this only works with Berkeley mail (/usr/ucb/mail), not with /bin/mail. Alternatively, you can invoke "sendmail -v" explicitly.

See Also:

System Manager's Manual, "System Operation" section:
"Setting Up the Mail System"
System Manager's Manual, "Tutorials" section:
SENDMAIL — An Internetwork Mail Router
SENDMAIL Installation and Operation Guide

2.17) When kernel name is changed, ps breaks

Description:

Sometimes you will have occasion to boot a kernel with a name other than */vmunix*. Note that the command *ps* (with no arguments) will not work correctly on such a kernel unless it happens to have the same symbol table as */vmunix*. To make *ps* work properly, give it the name of the kernel you booted as its third argument, e.g., "ps ax vmunix.new". For routine use, it is best to name the kernel you normally use "/vmunix".

See Also:

ps(1) manual page

2.18) Vi uses smaller screen at slow baud rates

Description:

Several people have reported the "bug" that *vi* uses only a few lines of the screen when run at slow baud rates. For example, when used over a modem at 300 baud, *vi* displays only eight lines. This is actually an intentional feature of *vi*. The idea is to minimize the amount of screen updating necessary at slow baud rates. Once you get used to it, this allows you to edit files more quickly over dial-in lines. If you want, you can change back to a full-screen display by entering the *ex* command ":set window=23" (assuming, say, a 24 line screen). You will also want to say ":set scroll=12" (half the screen size) to make ^D and ^U behave as expected.

2.19) Use large buffer size with Archive tape drive.

Applies to: Systems with Archive cartridge tape drive, release 0.3 and later.
Description:

Starting with Release 1.0, the optimum blocking factor to use when writing or reading an Archive cartridge tape is 126 blocks. With the *tar* command, use the "b 126" option (see examples below). With *dd*, use "bs=126b". Using this buffer size, *tar* will run up to five times faster than with the default buffer size of 20b. When reading or writing large files (>>100k bytes), you will achieve a net speed equal to about one-half of the full streaming speed of the Archive drive. In releases 0.3 and 0.4, any large blocksize (like "b 100") will work well. On V7/UniSoft systems, the larger block size does not improve the speed of the Archive drive.

2.20) More miscellaneous information about "tar"

Description:

When making a *tar* tape, use relative pathnames for files rather than absolute path names. For example, use

```
cd /usr
tar cvbf 126 /dev/rar0 demo
```

rather than

```
tar cvbf 126 /dev/rar0 /usr/demo
```

This permits you to restore the files from the tape into any chosen directory rather than forcing you to restore them to the same place they came from.

One customer called us with a problem: he had created a large *tar* tape with absolute pathnames of the form "/user/...". When he tried to restore from tape, *tar* created a directory called /user in the root file system and began to place the files there. This quickly filled the root file system. The work-around for this problem was to make a symbolic link in the root pointing into another file system where there was more room. In this case,

```
cd /
rm -r user
ln -s /usr/newstuff user
tar xvbfp ....
```

Now anything stored by *tar* in /user actually goes into /usr/newstuff.

Another problem that might arise is a *tar* tape containing one huge directory that you don't want to restore, along with many small files or directories that you do want to restore. If you create a (non-directory) file with the same name as the directory you don't want off the tape, then *tar* will not extract that directory.

When creating a *tar* tape, you can use the option **-C** to change directories, allowing *tar* to use shorter relative path names. For example, suppose you want to make a *tar* tape containing all the files in /usr/fred/bin, /usr/bill/1.0/cmds, and /usr/nancy/bin. Rather than copying all the files into one directory, you could type the following:

```
cd /usr/fred/bin
tar cvbf 126 /dev/rst0 * -C /usr/bill/1.0 * -C /usr/nancy/bin *
```

Note that the use of "*" assumes that none of the file names begin with ".".

See Also:

ln(1), tar(1) manual pages
Item 1.17, "Miscellaneous information about tar"
in *Tech Support Bulletin* Issue 1984-1.

2.21) Jumpering Xylogics 450 disk controller

Description:

Some customers who took the hardware training class may have gotten incorrect information on jumpering the Xylogics 450 disk controller. The jumpering information in the *System Manager's Manual* is correct.