System 8000™
ZEUS Administrator Manual

03-3246-04

October, 1983

SYSTEM 8000

ZEUS ADMINISTRATOR MANUAL

Software Release 3.2

Preface

This manual describes the steps involved in performing various system administrative functions: system startup (bootstrapping), system reconfiguration (software system regeneration), and system maintenance (adding or deleting users, file management, backing up files to tape, etc.).

Each section of this manual describes a major component of system administration. Certain sections may require a basic understanding of the ZEUS operating system. Appendix A provides an overview of the ZEUS operating system. Appendix B contains important considerations for changing the layout of disk filesystems.

The following are related manuals that are part of the standard manual set. Additional manuals describing particular boards and/or options can be ordered using the part numbers in the Zilog literature order form.

Related Software Manuals

System 8000 ZEUS Reference Manual (03-3255)
System 8000 ZEUS Utilities Manual (03-3250)
System 8000 ZEUS Languages/Programming Tools Manual (03-3249)

Related Hardware Manuals

System 8000 Hardware Reference Manual (03-3237)
System 8000 SADIE Reference Manual (03-3264)

Table of Contents

List of Illustrations

List of Tables

SECTION 1
INTRODUCTION

Almost all of the system administrator functions must be performed by the ZEUS super-user (called "zeus"). The super-user has the permission to read, write, and execute all files, no matter what the access permission bits indicate. Other functions exclusive to the super-user include changing the password of any user, changing the owner or access permission bits of any file, entering a user into the system, and making a file system on disk.

The system administrator becomes a super-user either by typing "zeus" as a response to the login message or by typing "su zeus" while logged in as another user. The system prompts for the super-user's password, and the appropriate response is "jupiter." This password should be changed promptly after system installation (see passwd(1)).

Command notations within the text of this manual are followed by a letter or number enclosed within parentheses; for example, sysgen(M). The parenthetical expression references a particular section of the ZEUS Reference Manual. Section M contains commands that are specifically used for system administration or maintenance. General-use commands are normally followed by (1) to denote that they can be found in Section 1 of the ZEUS Reference Manual.

NOTE

Some of the system administration commands are actually "csh" shell scripts that can be fouled by user-selected, environmental parameter settings. Particularly, do not set the csh "noclobber" option or some of the system administration commands, such as adduser(M), will not work.

Throughout this book, there are procedures which require the entry of either of the terms "zd" or "smd". These terms refer to a program interface parameter for a particular disk type. Before continuing, find the applicable disk type and its associated entries in Table 1-1 which follows.

Table 1-1. Entries Required for Each Disk Type

	ASSOCIATED ENTRIES	
	TYPE OF INTERFACE	BOOT or RESTART
*MODELS 20, 21, 30	zd	ZBOOT D or Z D
*MODEL 31	smd (see note also)	ZBOOT S or Z S

* With any model, it is always possible to boot from tape using "Z T" and a ZEUS Release Tape (or 3.1 or higher Update Tape).

NOTE

The Model 31 is shipped with "smd" type disk(s). However, it can be optionally configured with "zd" type disks as well as "smd" type disks. In the latter case, the appropriate entry must be specified for each of the individual disk drives in the procedures which follow.

SECTION 2
STARTUP AND SHUTDOWN PROCEDURES

2.1. Initial Startup Considerations

When the System 8000 is shipped from the factory, the system disk has been formatted, and is loaded with the ZEUS operating system programs and files.

The Initial Startup Procedure consists of an automatic boot followed by the Startup File Management procedure in Section 2.6. If the automatic boot portion fails, the ZEUS files and programs on disks may have been lost due to unusual handling during shipment and installation. To restore correct disk data, the software restoration from a Release tape should be performed as described in Section 3. However, other diagnotic actions should be taken first as detailed in the following startup procedure.

2.2. Initial Startup Procedure

Once the System 8000 hardware is set up, the system can be started using the following procedure.

(1) Turn ON the system power with the rear-panel power switches.

(2) Turn ON the front-panel keylock switch. This enables the front-panel RESET and START switches.

(3) Proceed to the Automatic Boot Procedure in Section 2.3.2. If the automatic boot is successful, turn the keylock switch to the LOCK position, remove the key, ignore the remaining steps of this procedure and continue to Section 2.6 instead. (Leaving the keylock switch in the LOCK position disables the RESET and START switches, guarding against unintended interruptions of computer operation.)

    If the automatic boot fails, return to this section and perform the remaining steps in this procedure.

(4) If the boot failed, turn off system power.

(5) Remove the pop-off front cover from the Processor module (with the RESET and START pushbuttons).

(6) Remove the CPU card from slot one.

(7) Switches 1 and 4 (bits D4 and D5) on the CPU card control the console terminal port baud rate. Ensure that they match the baud rate of the console terminal that is attached to the system. See Figure 2-1, Table 2-1, and refer to the System 8000 Hardware Reference Manual.

(8) Switches 2 and 3 (bits D7 and D6) on the CPU card indicate the boot disk device type (refer to Figure 2-1). As shown in Table 2-2, use the settings which correspond to the "zd" disk interface type for the Model 21; use the settings which correspond to the "smd" disk interface type for the Model 31.

(9) Re-install the CPU card.

(10) Ensure that the Disk/Tape module power cord is plugged in to the outlet at the rear of the Processor module. (The absence of the disk's steady whirring sound would indicate that power is not connected.)

(11) Refer to Section 3.4 of the System 8000 Hardware Reference Manual, and perform those steps necessary to ensure that the disk head is unlocked.

(12) Repeat steps 1-3. However, if the boot fails again, proceed to Section 3, particularly the Software Reinstallation Procedure in Section 3.2. Alternately, run the Stand-Alone Diagnostic Interactive Executive (SADIE) programs as described in the SADIE Reference Manual in order to verify proper system hardware operation.

Table 2-1. CPU Card Baud Rate Settings

DATA BITS:	D5	D4	
SWITCHES:	4	1	BAUD RATE
	ON	ON	300
	ON	OFF	1200
	OFF	ON	9600
	OFF	OFF	19,200

Table 2-2. CPU Card Disk Type Switch Settings

NOTE

These switches indicate the interface type for disk device Ø (default boot device).

DATA BITS:	D6	D7
SWITCHES:	2	3
DISK INTERFACE TYPE		
md*	ON	OFF
zd	OFF	OFF
smd	OFF	ON
reserved	ON	ON

* md is used only on the Model 11

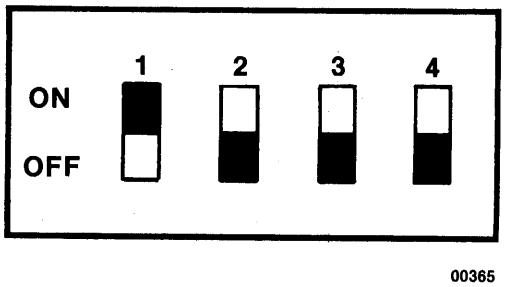

Figure 2-1 Baud Rate/Disk Type DIP Switches Shown for zd type disk; 96ØØ baud

2.3. Automatic Boot Procedure

The automatic boot method is the normal way the system is brought up since it requires only two keyboard inputs and it automatically performs file system integrity checks.

2.3.1. General

The automatic boot procedure is initiated by pressing the RESET and then START pushbuttons.

First, the power-on diagnostics are run, the bootstrap program is read from disk, the kernel is read, and then INIT is called (see Section 7). INIT executes the file /etc/rc (a "sh" shell script). This script in turn conditionally executes the /etc/rc_csh file (a "csh" shell script). Among other things, the rc_csh executes the fsck(M) program which checks the consistency of the (yet unmounted) filesystems. The execution of /etc/rc_csh displays status messages on the console to indicate the progress of the boot.

If the system encounters any problems that it cannot fix automatically, it indicates the fact and issues a prompt on the system console. The system is in single user mode, and is ready to accept commands. If the problem is due to a filesystem inconsistency, the system administrator should attempt to fix the problem by following instructions in Section 4.5, Managing File System Consistency.

After /etc/rc_csh checks for certain error conditions, it prompts for the date (see datem(M)). Then it mounts certain filesystems. A successful boot is indicated by the message

```
ZEUS login:
```

on the system console and on all terminals. At this point, the system administrator should login. Users can login as soon as the login message is displayed.

The file /etc/rc_csh can be modified to reflect conditions at each installation. For example, if a disk is reconfigured, the fsck command line in /etc/rc_csh should be changed accordingly. Other necessary changes are described in Section 4.2.

2.3.2. Procedure

1. Press the front-panel RESET switch. The console should display the following message:

   S8000 Monitor X.Y - Press START To Load System

   If it does not, ensure that the keylock is ON. Press RESET. If there is still no response, contact the nearest Zilog Field Service office. RESET initiates communication with the monitor. Monitor commands are listed in Section 4 of the System 8000 Hardware Reference Manual.

2. Press the START switch, as indicated in the above display. The console should display filesystem information for a short duration, followed by the last known date and time.

   Last known date and time: Fri Mar 5 08:13:19 1982

   Then a prompt provides the option of entering a new date:

   Enter Date (MM/DD/YY or <cr>):

   Enter the date in the following format, followed by a RETURN:

   MM/DD/YY

   where MM, DD, and YY are month, day, and year, respectively. See the example below.

   Next, the console prompts for the time:

   Enter Time (HH:MM):

   Enter the time in the following format, followed by RETURN:

   HH:MM

   where HH and MM are hours and minutes. Enter the hour from a 24-hour clock. See the example below. When the entry is complete, the console displays the new date and time. See also date(M).

   Example:

```
Last known date and time: Fri Mar 5 08:13:19 1982
Enter Date (MM/DD/YY or <cr>): 3/26/82
Enter Time (HH:MM): 16:48
Fri Mar 26 16:48:00 PST 1982
```

3. A successful boot is indicated when the system console displays the last date the system stored, followed by:

```
Going multi-user!
ZEUS login:
```

4. If this boot was preceded by reconfiguration of the disk (sysgen) or recovery of the original software from Release tape (Section 3), or if this was the initial boot after the system is first installed, then proceed to Section 2.6.

2.4. Manual Boot Procedure

The manual boot method requires explicit entry of parameters that indicate the names and locations of programs to be executed as part of system initialization.

The manual boot can be used to bring the system up in single-user mode as well as to perform certain stand-alone recovery functions from the secondary bootstrapper. The benefits of single-user mode are described in Single-User Mode Functions, Section 2.5.

2.4.1. General

The manual boot procedure allows user interaction with the secondary bootstrapper program. If booting from disk, the CPU Monitor prompts the user for the name of a secondary bootstrapper, which it reads from disk and executes. If booting from tape, the tape primary bootstrapper automatically reads the secondary bootstrapper from tape and executes it.

The secondary bootstrapper can read and execute any stand-alone program from tape or disk. After execution, the secondary bootstrapper regains control and can run another program.

The secondary bootstrapper program is provided on disk within the first filesystem and on tape as the second file on any Release Tape or 3.1 (or higher) Update Tape.

2.4.2. Procedure

Use the following procedure to bootstrap manually. Each command must be followed by a (carriage) RETURN.

1. Enter the monitor by pressing RESET. At this point, depending on the System 8000 model, it is possible to boot from either disk or tape (providing the disk is intact). To load from disk, go to step 2. To load from tape, go to step 3.

2. To boot from disk, enter either one of the following commands for Model 21 systems:

```
ZBOOT D
Z D
```

For Model 31 systems, use either one of the following commands:

```
ZBOOT S
Z S
```

The CPU Monitor prompts the user for the name of the secondary bootstrapper with:

```
BOOTING FROM DISK
>
```

At this point the user should enter the name of the secondary bootstrapper (the default is "boot") followed by RETURN.

```
> boot
```

This loads the secondary bootstrapper from the first filesystem on the disk, and executes it. Go to step 4.

3. To load the primary bootstrapper from tape and execute it, enter the command

```
ZBOOT T
```

or

```
Z T
```

The primary bootstrapper automatically loads and executes the secondary bootstrapper from the tape. The monitor displays:

```
BOOTING FROM TAPE
(Non-) Segmented Jumper Configuration
```

Continue to step 4.

4. For either procedure the secondary bootstrapper prompt:

```
(Non-) Segmented Jumper Configuration
Boot
:
```

should now be displayed on the console.

The secondary bootstrapper loads a program from tape or disk, and executes it. The command syntax for reading and executing a disk program is:

```
:xxx(n,m)name [ | xxx(n,m)fppname ]
```

where: xxx is either "zd" for Model 21 systems or "smd" for Model 31 systems where the boot disk is of "smd" type.
n is the drive number of the disk
name is the name of the file to be read in,
m is the starting block of the filesystem where the file resides (the first filesystem has index zero)
fppname is the name of the file containing the optional Floating Point Board Set microcode (only useful when the option has been properly installed)
[...] indicates that the expression contained within the brackets is optional (note that brackets are a textual notation only - actual entries would not include the brackets)

NOTE

The optional expression [within brackets], if left unspecified, defaults to that shown above if a properly installed Floating Point Board Set has been detected.

Further, an entry consisting of only a carriage return (blank line) causes the secondary bootstrapper to supply a default entry, similar to the operation of the automatic, primary bootstrapper. The default entry is valid for the default system disk layout that is initially provided for the first disk. Amy subsequent changes to the system disk layout necessitate the recreation of the secondary boot program (and primary boot defaults) in order to maintain valid default parameters. This task is described in the system generation chapter, Section 5.

Example:

If the user enters

```
zd(0,15200)zeus
```

the kernel is booted from drive 0 at a block offset of 15200, from a "zd" type disk (Model 21 system). Enter this to change the system to single-user mode from the secondary bootstrapper program.

The command syntax for a tape program is:

```
ct(n,m)
```

where: n is the drive number of the tape
m is the mth file on tape, beginning with zero (m must be greater than 1 since the first and second files on tape are the bootstrappers themselves).

5. Run the ZEUS kernel (refer to the example in the preceding step) from the secondary bootstrapper to bring up ZEUS in single-user mode. Now ZEUS has control of the system, which it keeps it rather than returning control to the secondary bootstrapper. When started in this manner, ZEUS automatically runs one csh(1) command interpreter program. It expects ZEUS commands, not secondary bootstrapper commands.

   The system administrator now issues commands directly to the ZEUS system, instead of to the secondary bootstrapper.

NOTE

Only a subset of the available ZEUS commands listed in the ZEUS Reference Manual is available from single-user mode unless all filesystems (not just the root) are manually mounted with the mfs(M) command.

All of the ZEUS commands become available once filesystems other than the root are mounted. This task is automatically performed when multi-user state is requested with the "INIT 2" command.

If the manual boot was preceded by reconfiguration of the disk (sysgen) or recovery of the original software from Release tape (Section 3), or if this was the initial boot after the system is first installed, then proceed to Section 2.6.

NOTE

Entering "INIT 2" after step 5 of the manual bootstrap procedure leaves the system in the same (multi-user) state as would an automatic boot.

2.5. Single-User Mode Functions

In this mode, the administrator can perform various maintenance activities that are often incompatible with a normal multi-user, time-sharing operation. These include the filesystem initialization tasks performed with labelit(M), mkfs(M), and makenewfs(M), which are part of the data restoration tasks as described in Section 3.

Single-user mode is also useful when troubleshooting and recovery functions must be performed. These tasks often require that the filesystems (except root) be unmounted (a single-user mode default). Entering single-user mode from multi-user mode automatically kills all user-processes and ongoing processes such as the spooler, any of which could be responsible for improper system operation. Refer to Section 6.4.

If serious errors are reported concerning disk files or disk devices that cannot be fixed simply, such as by freeing up space on a filesystem, enter single-user mode and then follow the detailed procedures in Sections 4.5 through 4.5.7. An automatic boot will also perform many of the same troubleshooting functions and attempt to repair disk files. So, this may be the first corrective action that should be taken (and often is required if the error "crashes" the system). If the system does not "crash" as the result of a serious disk problem, the super-user should bring the system down anyway, since any further work with files could jeopardize disk data even further.

To switch from multi-user mode to single-user, use the procedure in Section 2.8.

In single-user mode, the user at the console is automatically made the super-user (often referred to as zeus), providing unlimited access to files and programs (except if they are presently unmounted).

ZEUS issues a prompt for each command line in the form "#n", where n is the command number which can be referenced to re-request commands already entered (see csh(1)).

To change the operating mode to multi-user, enter:

```
INIT 2
```

This invokes /etc/INIT which causes the shell scripts /etc/rc and /etc/rc_csh to run.

Successful completion of the startup script is indicated when all terminals display the login prompt and the system console displays:

```
Going multi-user!
ZEUS login:
```

The most important things that /etc/rc_csh does are the following:

a) Runs the filesystem consistency check/fix program fsck(M). Problems that fsck cannot automatically fix must be fixed manually. See Section 4.5, Managing Filesystem Consistency.

b) Prompts for the date, as described in datem(M). A reply must be given within 2 minutes. See the procedure in Section 2.3.2, step 2. If this is not done, the system displays the last known date and time, and then enters multi-user mode.

c) Issues a mount command for each filesystem to be mounted. This is done by running the shell script, /etc/mfs, which contains the mount commands.

d) Background processes such as dqueuer(M) and cron(M) are started.

2.6. Startup File Management

This procedure is not required for boots subsequent to the initial boot (unless such boots follow file system restoration or system reconfiguration).

1. If in single-user mode, request multi-user mode by entering:

```
INIT 2
```

NOTE

Steps seven and higher require mounted filesystems. Entering multi-user mode with "INIT 2" automatically accomplishes this at the outset. Even though other terminals may display login prompts, their operation is suspect until this procedure has been completed.

On the system console, login as zeus to perform super-user (system administrator) functions. The password is "jupiter", unless it has already been changed. After initial startup procedures, the password should be changed. See passwd(1).

2. Check the file /etc/inittab to ensure that it contains desired parameters for terminals other than the console that have been attached to the system. The system reads this information when switching from single-user to multi-user mode.

NOTE

The console entry is a special one that should not require changing.

Terminals other than the console can be configured to run at a variety of baud rates by leaving the GETTY command in place but changing the numbers following the GETTY command as described in Section 7.3.3. To configure a terminal port for write-only operation (printers), eliminate the GETTY fields and do not replace them with any other interactive program invocation. The GETTY invocation is also required to establish the baud rates for terminal ports to be used with dialin modems.

The system console terminal may be used like any other terminal. It is not dedicated to console functions, although system error messages are displayed on it.

3. Configure the system's serial ports as modems or terminals with the ttyconfig(M) command by changing the ttyconfig command line in the /etc/rc_csh startup script. The ttyconfig command line (or lines) specifies which ports use the modem control lines, and which do not.

The ttyconfig command has two options:

```
ttyconfig -m ports
```

and

```
ttyconfig -t ports
```

where:

-m specifies a modem (7-wire line) configuration

-t specifies a terminal (3-wire line) configuration

ports is comma-separated TTY port numbers or ranges of port numbers

Example: To configure terminal ports 2-6 for terminals (3-wire lines) and port 7 for a modem (7-wire line), use the command

```
ttyconfig -t 2-6   -m 7
```

NOTE

To avoid having to reboot the system to establish these settings, directly enter these commands as the super-user. Check the results by entering the command "ttyconfig" with no arguments after the csh prompt.

If the system encounters a problem while opening an improperly configured port, it may become caught in a continuous retry loop. Follow the procedure at the end of Section 7.3.2.

4. Update the file /etc/ttytype. The login program reads this file to determine the user's terminal type. This information is then used by the visual editor **vi**. See ttytype(5) for correct parameter settings and step 2 in Section 7.4.2.

5. If the system has more than one cartridge tape drive, the correct entries must be placed in /dev for each additional drive. For each such drive, enter the command

```
/etc/mktape n
```

where n is between 1 and 3, and is the tape drive number. Tape drives are numbered from Ø through 3, but the correct entry is already in /dev for tape drive zero. Mktape(M) is a program located in /etc that makes the necessary special files in /dev. For example, if the system has three tape drives, the following commands would be executed:

```
/etc/mktape 1
/etc/mktape 2
```

After these commands are issued, they should never be issued again (unless, for some reason, the entries in /dev are removed or destroyed). See ct(4) and mktape(M) for more information about the cartridge tape.

6. If the system has one or more nine-track tape drives, the correct entries must be placed in /dev for each drive. For each, enter the command:

   /etc/mkmt mt<n>

   where n is between Ø and 3.

   Mkmt(M) is a program that makes the special files in /dev. For example, if the system has two tape drives, the following commands would be required:

   /etc/mkmt mtØ
   /etc/mkmt mtl

   After these commands are issued, they need not be executed again unless the entries in /dev are removed or otherwise destroyed. See mt(4) and mkmt(M) for more information.

7. If the default timezone mode of Pacific Standard Time is acceptable, proceed to step 8. The default is also for daylight savings time to be displayed. If a change in timezone mode is desired, perform the following steps once only; thereafter, when the system is rebooted the modified timezone becomes the new default.

   a) Edit the /etc/rc file, changing the TZ environ(5) variable to the appropriate value. Particularly, the string "PST8PDT" must be changed to reflect the appropriate timezone abbreviation (ie. PST), the correct number of hours west of Greenwich Mean Time (8 is the correct number for a Pacific timezone) and the daylight time abbreviation (ie. PDT). The original and new lines for changing the entry from Pacific daylight savings time to Eastern daylight time are as follows:

      WAS:
      setenv TZ PST8PDT

      IS:
      setenv TZ EST5EDT

   b) Make the same change as described above in three other files, /etc/cshprofile, /etc/inittab and /etc/profile.

   c) Run sysgen (Section 5) in order to change the system constants DSTFLAG and TIMEZONE. Set DSTFLAG to a value of 1 to enable daylight savings time. Set TIMEZONE to a value representing the number of minutes westward from Greenwich mean time (the product of 6Ø x <number of timezones west of GMT>).

d) After rebooting, if the system is in single-user mode, repeat step 1.

8. If the system has more than one disk drive, follow the procedure described in Section 4.2 for configuring them.

9. If the system has an additional line printer, or if the line printer has a Data Products (vs. Centronics) interface, refer to Section 7.1.

10. If any filesystems other than /usr, /tmp, /z and / have been created by changing the disk layouts or adding disks, then "lost and found" directories should be made for each of them as described in this step. (For the default configuration of disk drive 0, the directories mentioned have factory-installed lost and found directories.)

Make a directory called "lost+found" in all filesystems that do not already contain the directory. Fsck(M) uses the "lost+found" directories to fix certain filesystem consistency problems. For instance, if a file has no corresponding directory entry, then fsck puts that file in the "lost+found" directory for that filesystem. However, in this case, if there is no "lost+found" directory, it cannot be safely made, because the filesystem is already corrupt. The file will be lost.

The space for the directory "lost+found" must be allocated in advance. (Free blocks cannot be allocated to the directory when the filesystem is in the process of being fixed). Typically, about ten blocks are allocated to the directory. An easy way to do this is to make a directory, move some files into it, and remove them. The following csh script is automatically run by makenewfs(M):

```
#
foreach i (/tmp /z)
        set dir="$i/lost+found"
                if (! -e $dir) then
                        mkdir $dir
                        chmog 0750 zeus 0 $dir
                endif
        @ j = 318
        while $j
                echo > $dir/Z$j
                @ j--
        end
        @ j = 318
        while $j
                rm $dir/Z$j
                @ j--
```

```
        end
    end
```

With the standard disk configuration and one drive, this shell script is executed for the directories /tmp and /z. It takes about 5 minutes to complete.

11. To activate all of the parameters requested in the previous steps, redispatch multi-user mode by entering:

    ```
    INIT 2
    ```

12. On the system console, login as zeus to perform additional super-user functions as described in the following sections. The password is "jupiter", unless it has already been changed. After initial startup procedures, the password should be changed. See passwd(1).

2.7. Adding New Users and Terminals

New terminals and new users can be added after the initial power-on, boot, and file administration procedures have been completed. See Section 7 and adduser(M).

2.8. Bringing the System Down

Execute the utility down(M), which brings the system down without mishap. Then press RESET.

Down(M) is the recommended utility for bringing the system down; however, it can be done manually. Use the following procedure:

1. Use the wall(M) command, which sends a message to each terminal, notifying all logged-in users of the impending shutdown.

2. Issue the command

   ```
   INIT 1          OR

   kill -1 1
   ```

   This kills all multi-user related processes running on the system. This command must be issued by the super-user. The system is left running a single-user csh process that receives commands from the console.

3. Issue the commands

```
sync; sync
```

from the system console. This forces all outstanding I/O on the system to completion. Now, the system can either be powered down or rebooted. To shut the system down, press the front-panel RESET switch and turn the system power off.

SECTION 3
RESTORING THE SYSTEM DISK

3.1. General

The System 8000 is shipped with a preformatted disk that contains all operating system software. Consequently, the system should boot properly with the manual or automatic boot procedure.

NOTE

If the Initial Startup Procedure described in Section 2 has been successfully performed, Section 3 can be ignored.

If the system does not boot properly, then the backup copy of the system software on the release tape must first be restored to the disk using the procedures in this section. In general, the restoration procedure involves creating empty ZEUS filesystems and restoring data files and directories onto the empty filesystems. These procedures are combined into one general procedure in the subsection which immediately follows.

NOTE

The procedures in this section require a properly formatted disk.

Disk formatting is unnecessary at installation of a new system since it has been formatted at the factory. However, the formatting procedure may be a necessary part of disk repair or recovery. Disk formatting is performed with the Stand-Alone Diagnostic Interactive Executive (SADIE) program. Refer to the SADIE Reference Manual.

All of the programs and data files needed for this complete restoration are on the Release tape and are listed below. The number within parenthesis is the correct number to use to reference the location on the tape where the associated file or program resides.

(0) The primary bootstrap program -- a 512 byte program that automatically loads the next program (the secondary bootstrapper) into memory and executes it.

(1) The secondary bootstrap program -- a program that reads into memory and executes any stand-alone program from tape or disk.

(2) Disk formatting information -- informs the user about reformatting the system disks.

(3) A stand-alone version of mkfs(M) -- a program that makes filesystems on the disk.

(4) A stand-alone version of restor(M) -- a program that reads filesystems from the system tape to disk.

(5) A level Ø dump of the common root filesystem that is restored from tape to disk. This root filesystem is common to all System 8ØØØ models.

(6) A level 1 dump of Model 21 special root files.

(7) A level 1 dump of Model 11 special root files.

(8) A level 1 dump of Model 31 special root files.

(9) A level Ø dump of the common filesystem /usr that is restored from tape to disk. This filesystem is common to all System 8ØØØ models.

(1Ø) A level 1 dump of Model 21 and Model 31 special /usr files.

(XX) File locations higher than 1Ø contain tar(1)-recorded software packages including the accounting system, global optimizer, learn tutorials, source code control system (sccs), version 7 nroff, zmenu, etc.

NOTE

The files and/or programs after location 1Ø can be restored using the package(M) utility.

The common root is the set of files making up the root that are the same for all System 8ØØØ models. Though they may be identically named, two or more versions of other files are supplied to accommodate specific System 8ØØØ models. The model-specific files for each different model are grouped together in a single location on the tape, making it possible to select the correct files through tape location rather than filename.

The following is a complete list of the files specific to the Model 21 (where Y.Z is the current release number):

```
/zeus, /zeus2_Y.Z
/dev/zdØ,  /dev/zd2,  /dev/zd3,  /dev/zd4
/dev/rzdØ,  /dev/rzd2,  /dev/rzd3,  /dev/rzd4,
/dev/z, /dev/rz, /dev/usr, /dev/rusr, /dev/tmp, /dev/rtmp,
/dev/root, /dev/rroot, /dev/swap, /etc/group
```

The following is a complete list of the files specific to the Model 31 (where Y.Z is the current release number):

```
/zeus, /zeus2_Y.Z
/dev/smdØ, /dev/smd2, /dev/smd3, /dev/smd4,
/dev/rsmdØ, /dev/rsmd2, /dev/rsmd3, /dev/rsmd4
/dev/z, /dev/rz, /dev/usr, /dev/rusr, /dev/tmp, /dev/rtmp,
/dev/root, /dev/rroot, /dev/swap, /etc/group
```

Note that the following filenames are linked to the same file ("zd" should be "smd" for Model 31 systems):

```
/zeus and /zeus2_Y.Z

/dev/zdØ and /dev/usr
/dev/rzdØ and /dev/rusr

/dev/zd2 and /dev/root
/dev/rzd2 and /dev/rroot

/dev/zd3 and /dev/tmp
/dev/rzd3 and /dev/rtmp

/dev/zd4 and /dev/z
/dev/rzd4 and /dev/rz
```

3.2. Software Reinstallation Using Release Tape

CAUTION

Portions of this procedure destroy the contents of the disk.

The following procedure is a global procedure, incorporating all the other procedures in Section 3, and assumes a Model 21 system. At the end of the procedure, the system is in multi-user mode, all terminals display the login prompt, and the system is ready for use.

NOTE

Except for pressing the RESET and START buttons, all user entries MUST be concluded with a (carriage) RETURN.

The format for each of the steps in the following procedure is shown in Table 3-1. If any of the items of information is missing, then there is no pertinent information.

Table 3-1. Procedure Step Format Description

<step no.>	
display:	this line(s) contains the information displayed on the screen.
enter:	this line describes the information that must be entered by the user -- information that is entered literally is enclosed in quotes ("").
note:	explanation of any of the above is contained on this line.

NOTE

To boot from an "smd" type disk, enter "smd" in place of "zd" in steps 8, 14, 22, 25, 35, and 38 of the following procedure.

Initial Boot/File System Initialization Procedures

Step 1. Switch power on and press the RESET button on the front panel.

Step 2.

display: S8000 Monitor X.Y - Press START to Load System
enter: "Z T"
note: "ZBOOT T" can also be used.

Step 3.

display: BOOTING FROM TAPE
(Non-) Segmented Jumper Configuration
Boot
:
enter: "ct(0,2)"

Step 4.

display: ZEUS Disk Format Information
.
.
.
enter: <CR>
note: Disk formatting is unnecessary at installation of a new system, since it has been formatted at the factory. However, the formatting procedure may be a necessary part of disk repair. Disk formatting is performed with the SADIE diagnostic programs. Refer to the descriptions of SADIE operation in the SADIE Reference Manual.

Step 5.

display:
Boot
:
enter: "ct(0,3)"
note: **CAUTION! Steps 5 through 10 initialize the root ("/") filesystem.** Do not initiate this step unless you are prepared to wipe out all the information in the root ("/") filesystem.

Step 6.

display: (tttt/0xTTTT)+(dddd/0xDDDD) + (...)+(...) ...
note: Each pair of parenthetical expressions contains the size of text and data in bytes of a particular

segment, starting with segment Ø. Decimal digits are followed by a slash and the hex equivalent.

Step 7.

display: filesystem size:
enter: "6ØØØ"

Step 8. (See NOTE preceding this procedure)

display: filesystem:
enter: "zd(Ø, 15200)"

Step 9.

display: interleaving factor (m n):
enter: m n
note: m and n are interleaving factors; see Table 3-2.

Step 1Ø.

display: isize = 16ØØ m/n = mm nn
Exit called
note: mm and nn are the interleaving factors entered in the preceding step. The isize will vary with the filesystem size.

Step 11.

display:
Boot
:
enter: "ct(Ø,3)"
note: **CAUTION! Steps 11 through 16 initialize the ("/usr") filesystem.** Do not initiate this step unless you are prepared to wipe out all the information in the "/usr" filesystem.

Step 12.

display: (tttt/OxTTTT)+(dddd/OxDDDD) + (...)+(...) ...
note: Each pair of parenthetical expressions contains the size of text and data in bytes of a particular segment, starting with segment Ø. Decimal digits are followed by a slash and the hex equivalent.

Step 13.

display: filesystem size:
enter: "12ØØØ"

Step 14. (See NOTE preceding this procedure)

display: filesystem:
enter: "zd(Ø,Ø)"

Step 15.

display: interleaving factors (m n):
enter: m n
note: m and n are the interleaving factors; see Table 3-2.

Step 16.

display: isize = 16ØØ m/n = mm nn
Exit called
note: mm and nn are the interleaving factors entered in the preceding step. The isize will vary with the filesystem size.

Step 17.
display:
Boot
:
enter: "ct(Ø,4)"
note: This procedure loads and executes the stand-alone version of the restor program.

Step 18.

display: Will you be restoring from a factory-supplied Zilog release tape (y or n)?
enter: "y" or "n"
note: "y" answer initiates the interaction described in paragraph 3.5, step 2. Be sure to read this paragraph for complete instructions. An "n" answer skips to step 34 below.

Step 19.

display: Do you want instructions (y or n)?
enter: "y" or "n"
note: Instructions are described in paragraph 3.5, step 4. Enter "y" for instructions; else enter "n".

Step 2Ø.

display: Do you want to restore the root filesystem (y or n)?
enter: "y" or "n"

note: "y" requests restoration of the root filesystem; "n" skips steps 22-24 and 29.

Step 21.

display: Do you want to restore the /usr filesystem (y or n)?
enter: "y" or "n"
note: "y" requests restoration of /usr; "n" skips steps 25-27 and 30. "n" for both this step and the preceding step aborts the restor program, skipping to step 38.

Step 22. (See NOTE preceding this procedure)

display: Restoring the root filesystem
onto 'zd' type disk:
note: Press RETURN if disk type is zd; else enter disk type.

Step 23.

display: disk unit: 0
enter: RETURN or a number
note: Press RETURN if disk unit is 0; else enter disk unit number 0 through 3.

Step 24.

display: offset 15200:
enter: RETURN or offset size
note: Press RETURN if offset is 15200; else enter offset.

Step 25. (See NOTE preceding this procedure)

display: Restoring the /usr filesystem
onto "zd" type disk:
enter: RETURN or disk type
note: Press RETURN if disk type is zd; else enter disk type. See Table 3-2.

Step 26.

display: disk unit 0:
enter: RETURN or disk type
note: Press RETURN if unit is 0; else enter disk unit number 0 through 3.

Step 27.

display: offset Ø:
enter: RETURN or offset size
note: Press RETURN if offset is Ø; else enter offset.

Step 28.

display: Tape Unit number?
enter: Ø
note: Be sure to insert the cartridge tape in drive number Ø.

Step 29.

display: Restoring from tape drive #Ø
Onto device zd(Ø,15200):
Common / filesystem
Special / for 'zd' type drives
note: Console displays restor program constants from above steps.

Step 3Ø.

display: Onto device zd(Ø,Ø):
Common /usr filesystem
Special /usr for 'zd' type drives

Step 31.

display: OK to restore (y or n)?
enter: "y" or "n"
note: "n" causes program to exit. Start again at step 17. After an affirmative response the restoration takes about 3Ø minutes to complete.

Step 32.

display: Restoring / filesystem
Done.

Step 33.

display: Restoring /usr filesystem
Done.
note: If steps 19 through 32 have been performed, the program skips to step 38.

Step 34.

display: Tape?
enter: "ct(x,y)"

note: To restore from a dump tape not supplied by Zilog, enter the values for x and y as described in step 12 of Section 3.5.

Step 35. (See NOTE preceding this procedure)

display: Disk?
enter: "zd(Ø,15200)"

Step 36.

display: Last chance before scribbling on disk.
enter: RETURN
note: Allow about 15 minutes.

Step 37.

display: End of tape

Step 38. (See NOTE preceding this procedure)

display:

```
Boot
:
```

enter: "zd(Ø,15200)zeus"
note: Transfers control to the default system kernel restored in the preceding steps.

Step 39.

display:

```
(tttt/OxTTTT)+(dddd/OxDDDD) + (...)+(...) ...
Zilog Zeus Kernel-- Release y.y --Generated mm/dd/yy hh:mm:
Copyright 1981 Zilog, Inc.

System: SYS 8ØØØ, Node: ZEUS, Release: REL Y.Z Version

<Configuration Item>      = <Configuration Data>
                          .
                          .
                          .

SYS 8ØØØ ZEUS
Single-User Mode
```

note: Each pair of parenthetical expressions contains the size of text and data in bytes of a particular segment, starting with segment Ø through segment X, which is configuration dependent. Decimal digits are followed by a slash and the hex equivalent. Also displayed are the kernel, version, and node

default names. y.y is release number followed by release date.

Step 40.

display: #1
enter: "chmod 0700 /etc/makenewfs"
note: #1 is the first ZEUS prompt.

NOTE

Before proceeding to step 41, Model 20 and 30 users must modify the /etc/makenewfs script for different disk parameters as shown in Table 3-2.

Step 41.

display: #2
enter: "/etc/makenewfs"

Step 42.

display: **** WARNING ****
THIS PROGRAM WILL COMPLETELY REMAKE THE /z AND /tmp FILE SYSTEMS
Are you ready to proceed? (y or n)
enter: "y"

Step 43.

display: Initializing /z and /tmp file systems
/etc/mkfs /dev/rtmp 6000 12 96
/etc/mkfs /dev/rz 30400 12 96
isize=1280
m/n=mm nn
isize=1600
m/n=mm nn
The /z and /tmp filesystems are initialized
note: mm nn are interleaving factors; the isizes will vary with the filesystem size.

Step 44.

display: Labeling all filesystems...

All filesystems are being mounted temporarily to run the lost+found shell script as explained in the System Administrator Manual. The following shell script takes about 5 minutes to complete.

cp /boot /usr/boot

This file has been made non-executable.

chmod 700 makenewfs

before it can be used again

Step 45.

display: #3
note: Follow the directions in Section 2.6. This brings the system up in multi-user mode; system is ready for use. All terminals have the login prompt and are ready for use.

Table 3-2. System 8000 Disk Drive 0 Characteristics

System 8000 Model	Disk Drive Type	Disk Model	Interleaving Factors m	n	Logical Size (.5K blocks)
20 and 30 (24Mbyte)	8" Winchester (zd)	CDC Finch	12	72	43200
		BASF	12	72	43200
21 (32Mbyte)	8" Winchester (zd)	CDC CDC Finch	12	96	57600
31 (80Mbyte)	8" Winchester (smd)	Fujitsu	16	224	131936
		Memorex	16	224	131936

3.3. Disk Formatting Information

Disk formatting is unnecessary at installation of a new system, since it has been formatted at the factory. However, the formatting procedure may be a necessary part of disk repair. Disk formatting is performed with the Stand-Alone Diagnostic Interactive Executive (SADIE) program. Refer to the descriptions of SADIE operation in the SADIE Reference Manual.

In the rare event where system disk formatting is required, the individual procedures described in the remaining subsections must be followed instead of the overall procedure described in the preceding section.

3.4. Initializing the Root and /usr File Systems

In the case where the system disk has had to be formatted, the next step is to initialize the root and /usr filesystems as empty ZEUS filesystems. This procedure uses a stand-alone version of mkfs(M) (Make File System). While the stand-alone version of mkfs could also be used to initialize other filesystems on the disk, it is somewhat easier to initialize them under a running ZEUS system using the regular mkfs (as in the recommended procedures given in this section and Section 3.5).

(1) Press RESET to boot the primary bootstrapper. Then enter "Z T" or "ZBOOT T" to load the secondary bootstrapper. The following prompts should appear:

```
BOOTING FROM TAPE
(Non-) Segmented Jumper Configuration
Boot
:
```

(2) Enter:

```
ct(0,3)
```

after the colon. The secondary bootstrapper loads the stand-alone mkfs into memory and executes it. As part of the loading, the bootstrapper prints the sizes of the text (instructions), data, and bss (uninitialized data) areas of the program being loaded (in bytes).

(3) To initialize a ZEUS disk, two pieces of information are needed by mkfs: the size of the filesystem in 512 byte blocks, and the location of the filesystem on the disk. Location is specified as drive number and the offset, in blocks, of the first block of the filesystem.

Mkfs asks for each of these in turn. For example, the program prompts and user entries for the root (/) filesystem follow (for Model 31 systems, enter "smd" in place of "zd" at the "filesystem:" prompt).

```
file sys size: 6000
filesystem: zd(0,15200)
interleaving factor (m n): <Refer to Table 3-2>
```

where "zd" or "smd" is the disk type, zero indicates the physical disk unit, and 15200 is the block offset at which the filesystem starts. See Table 3-2 for interleaving factors. Mkfs prints internal information about the size of the inode pool and the interleaving factors. See Table 3-2.

Next, for /usr, repeat the above step, but enter 12000 for the filesystem size, zd(0,0) or "smd" for the filesystem, and enter the same interleaving factors as previously entered.

The stand-alone mkfs program runs for about a minute each time, and then exits and returns control to the secondary bootstrapper.

Mkfs initializes an empty ZEUS filesystem. A single directory is created that is the root directory for the filesystem. All the inodes (accounting of disk blocks) are marked as free, and all of the other disk blocks are placed on the free list. Continue to Section 3.5.

3.5. File System Restoration

(1) The next step is to fill the filesystem with the program and data files needed by ZEUS for normal operation. This is done by a stand-alone version of restor(M) that is in the fifth file of the system tape. So, in response to the prompt:

```
Boot
:
```

from the secondary bootstrapper, enter:

```
ct(0,4)
```

This reply executes the stand-alone restor. This reply, and the following replies to the prompts are standard. When first booting the system, be sure to use these replies as shown. Do not change these replies without careful prior consideration.

The bootstrapper prints the text, data, and bss sizes as it works.

(2) The restor program prompts

```
Will you be restoring from a factory-
supplied Zilog release tape (y or n)?
```

The answer "n" skips to step 12, below; the answer "y" continues to step 3.

(3) The standard system tape has a common root dump image in the sixth file. For the initial boot, leave this tape in the tape drive.

The program now prompts

```
Do you want instructions (y or n)?
```

The answer "y" continues to step 4 below; the answer "n" skips to step 5 below.

(4) The console displays:

This program builds filesystems from a release tape. It asks a series of questions to find out which filesystems you want to restore from what cartridge tape drive.

The program displays information about the drive type, unit number, and offset for building each filesystem. It waits for a response from you. A RETURN means that the display is correct; otherwise, the program changes these parameters to those you enter.

The program reads some switches on the CPU board to determine the disk type. The model numbers of the various systems correspond to the default disk types as follows:

```
Model 11      'md' type disk
Model 21      'zd' type disk
Model 31      'smd' type disk
```

When a yes/no answer is required, the affirmative answer must contain a "y" as the first character. A negative response must contain an "n" as the first character. On any other response, the question repeats.

(5) At this point, the program reads the CPU switches, as described in Section 2, to determine the type of boot drive device.

(6) The program prompts

```
Do you want to restore the root
filesystem (y or n)?
```

The answer "y" continues to step 7 below; the answer "n" causes step 8, and part of the display in step 11 to be skipped.

(7) The program prompts

```
Do you want to restore the /usr
filesystem (y or n)?
```

The answer "y" continues to step 8 below; the answer "n" causes step 9 and part of the display in step 11 to be skipped. A "n" to both questions aborts the restor program.

(8) The display prompts (for Model 31, "zd" becomes "smd")

```
Restoring the root filesystem onto
                        'zd' type disk:
                        disk unit Ø:
                        offset 15200:
```

in three steps. Following each prompt, either enter the correct information; or, if the default information is correct, press RETURN, not "y".

(9) The display prompts (for Model 31, "zd" becomes "smd")

```
Restoring the /usr filesystem onto
                        'zd' type disk:
                        disk unit Ø:
                        offset Ø:
```

in three steps. Following each prompt, either enter correct information; or if the default information is correct, press RETURN.

(10) The display prompts

```
Tape unit number?  Ø
```

If the boot drive device is Ø, press RETURN; otherwise enter the correct unit number, in the range 1 through 3.

(11) The display shows (for Model 31 "zd" is replaced by "smd"):

```
Restoring from tape drive #Ø
Onto device zd(Ø,15200);
Common / filesystem
Special / for 'zd' type drives
Onto device zd(Ø,Ø);
Common /usr filesystem
Special /usr for 'zd' type drives

OK to restore (y or n)?  y
```

The answer "n" aborts the restore operation. Repeat the above procedure starting at step 1 by entering

```
ct(Ø,4)
```

The answer "y" continues the restore operation. As the operation progresses, the display prints

```
Restoring / filesystem
Done.
Restoring /usr filesystem
Done.
```

(12) If you are restoring from a factory-supplied release tape, skip to step 14. To restor the / filesystem from a dump tape, reply "n" to the question in step 2. The restor program then prompts

```
Tape?
```

NOTE

When the display prompts "Tape?", any dump tape may be used.

To restore the root, remove the system tape, and insert the root dump tape in the drive. Specify:

```
ct(x,y)
```

for the tape, where x is the tape drive number, and y is the file location on the tape relative to zero, the first location. For example, if the restore is made from drive Ø, and is the first file (file Ø) on the tape, then the entry would be

```
ct(Ø,Ø)
```

It is strongly recommended that periodic dump tapes of the root and /usr filesystems be made to allow this type of stand-alone restoration. A dump of a 6ØØØ block root takes about ten minutes, and should be done about every month, or more frequently if there is very much activity on the root filesystem. Restor then prompts:

```
Disk?
```

Enter:

```
zd(Ø,15200)
```

Restor then prints the message:

```
Last chance before scribbling on the disk.
```

Press RETURN to continue, or RESET to abort. The message is redundant since formatting and mkfs have already destroyed whatever data was present on the filesystem. The restor takes about 30 seconds to decide exactly which files in the filesystem it restors (in this case, all of them), then it proceeds to read from the tape and write to the disk. The entire process takes about ten minutes. When restor completes, it prints the message:

End of tape

and transfers control back to the secondary bootstrapper.

(13) To restore the /usr filesystem from a dump tape, insert the system tape in the tape drive and enter:

ct(0,4)

The restor program prompts:

Will you be restoring from a factory supplied Zilog release tape (y or n)?

Respond by typing "n".

The restor program prompts:

Tape?

Remove the system tape from the tape drive and insert the dump tape. Then enter:

ct(x,y)

where x is the tape drive number
y is the position (relative to zero) of the file on tape that is to be restored.

Restor then prompts:

Disk?

Enter:

zd(0,0)

Restor then prints the message:

Last chance before scribbling on the disk.

Press RETURN to continue, or RESET to abort. The message is redundant since formatting and mkfs have already destroyed whatever data was present on the filesystem. The restor takes about 30 seconds to decide exactly which files in the filesystem it restors (in this case, all of them), then it proceeds to read from the tape and write to the disk. The entire process takes about fifteen minutes. When restor completes, it prints the message:

```
End of tape
```

and transfers control back to the secondary bootstrapper.

(14) Boot ZEUS with the tape secondary bootstrapper. In response to the prompt of:

```
Boot
:
```

Enter:

```
zd(0,15200)zeus
     OR
smd(0,15200)zeus
```

This directs the secondary bootstrapper to load the program "zeus" from the disk filesystem starting at block 15200 on drive 0. The bootstrapper prints the text, data, and bss sizes.

The first actions of the operating system are to print the kernel release message and date:

```
Zilog ZEUS Kernel --Release n.n-- Generated <date>
Copyright 1981 Zilog, Inc.
```

where n is the release variable.

The kernel then determines how much memory is available for normal mode (user) programs. This is (roughly) the amount of real memory on the system, less about 100 Kbytes for the operating system itself. The operating system prints the size of memory in decimal.

Finally the operating system is brought up in "single user mode" and is indicated by the system prompt: "#n", where n is the command number. This mode allows the system administrator to perform various maintenance actions that cannot be done conveniently (or at all) under normal time-sharing operation. The system is running the C shell (csh(1)) as

the user "zeus", the super-user.

(15) Now the filesystems on disk should be initialized. Enter the commands:

```
chmod 0700 /etc/makenewfs
/etc/makenewfs
```

NOTE

Before performing this step, Model 20 and 30 users must modify the /etc/makenewfs script for different disk parameters as shown in Table 3-2.

This causes the shell script makenewfs(M) to be run, which takes less than 15 minutes. As it executes, makenewfs asks for interleaving factors, and then prints statements on the console indicating what it is doing. For interleaving factors, see Table 3-2. Among other things, makenewfs executes the regular (i.e. not the stand-alone) version of mkfs. Most important, it executes the following commands (the values differ for models other than Model 21; refer to Table 3-2):

```
/etc/mkfs /dev/rtmp 6000 9 96
/etc/mkfs /dev/rz 30400 9 96
```

The interleaving factor of mkfs controls the order in which the blocks are placed on the free list. A gap is left between contiguous logical blocks of a file, so that ZEUS can initiate an I/O operation before the next block passes under the disk read/write heads. If the interleaving were not present, and the logically contiguous blocks of a file were physically contiguous on the disk, then ZEUS could not start an I/O operation to access the next block of the file before it had already passed by. This necessitates a delay of one full disk revolution to access the block. Interleaving is only effective on the initial creation of the free list. This means that files are allocated optimally only when they are allocated space from this optimal free list. With time, as files are removed and other files are created, the disk allocation becomes more and more suboptimal. Consequently, the best practice is to dump, mkfs, and restore the filesystems on a periodic basis. Following the initialization of the filesystems, makenewfs continues by making the lost+found directories.

Then, the secondary bootstrapper is copied from the root

filesystem to the first filesystem of drive 0, /usr. It is important to note that if makenewfs is not executed during the restoration or if the system has a non-standard configuration, the user must copy the secondary bootstrapper into the first filesystem of drive 0.

(16) If this is the initial boot, continue to Section 2.6. Otherwise, enter "INIT 2"; this brings the system up in a multiuser mode, and all terminals display the login message.

SECTION 4
FILE SYSTEM MANAGEMENT

4.1. Layout of the System Disk

At initial power-on, the system must be booted as described in Section 2 preceding. The initial boot procedure configures the disk associated with drive Ø. Configuration is determined by the set of mkfs(M) and restor(M) commands. The default configuration layout is listed in Table 4-1 below.

Table 4-1. Default Disk Configuration Parameters

System 8ØØØ Model No.	File System Device Name**	File System Name**	Default Size (in blocks)
21	/dev/zdØ	/usr	12ØØØ
	/dev/swap		32ØØ
	/dev/zd2	/	6ØØØ
	/dev/zd3	/tmp	6ØØØ
	/dev/zd4	/z	3Ø4ØØ
31	/dev/smdØ	/usr	12ØØØ
	/dev/swap		32ØØ
	/dev/smd2	/	6ØØØ
	/dev/smd3	/tmp	6ØØØ
	/dev/smd4	/z	1Ø4739

** File system and device names are linked.

The filesystem device name is the name the system associates with that area on disk. This name is actually the name of a special file located in /dev. Thus, the root filesystem initially supplied on the system tape has a directory in it called /dev with entries for, among other things, the special files /usr, /swap, /root, /tmp, and /z. Entries such as these were created by the mknod(M) command. For example:

```
/etc/mknod /dev/root b Ø 2
```

The filesystem name is the pathname component associated with that area on disk. For example, the file /usr/foo/file is located in filesystem /dev/usr. The association between the

filesystem and the system name is made by issuing a mount command. For example:

```
/etc/mount /dev/usr /usr
```

This is usually done at boot time for all normally mounted filesystems.

The filesystem /dev/z is meant to be used for development work. For instance, when user "stan" logs in, the home directory for that user might be /z/stan.

4.2. Adding Disks

The following paragraphs describe the recommended software modification procedure for adding extra disks. Any specific instructions supplied with a particular disk upgrade should be performed first.

4.2.1. General

Model 2Ø, 21, 3Ø, and 31 systems can operate with three additional disk drives. Unless the disk configuration for the additional drive is changed, the default configuration (assumed by ZEUS) is the same as the default for the first drive.

Before configuring the additional disk, the extra disk should be formatted, which physically sets up the disk for access by the software. Refer to Section 3.3. Do not format drive Ø!

The configuration of each additional disk is at the user's option; the only constraint is that each disk is limited to a maximum of ten filesystems. Recommended practice is to configure additional disks so that the most active filesystems are together near the center of the disk. This minimizes head travel and access time.

To change a disk configuration, invoke sysgen(M) and generate a new kernel. Sysgen prompts the user for the size of each filesystem on the drive, in order. After generating this new kernel, boot it. This procedure is described in the following section.

4.2.2. Procedure

To add another disk drive to a System 8000, use the following procedure. This procedure includes an example for adding a second disk to a system that previously had only one disk. The example is continued through the procedure.

In the example, a second disk drive is being added to a System 8000 Model 21 that presently has only one disk. The format of the new disk includes two filesystems, z1 and z2; the size (in blocks) of each will be 20,000 and 37,600.

1. Make a new zeus kernel that includes the layout of the new disk format(s). See sysgen(M) and Section 5. Enter the commands

```
cd /usr/sys/conf
/etc/sysgen
```

and answer the questions as prompted. The questions and answers for the example are shown below. The newly created zeus kernel recognizes the new filesystems on the second disk.

```
Do you wish to change any system constants? (y or n) : n

Is this kernel for a Model 11 system? (y or n) : n

Do you have ZD type drives on your system? (y or n) : y

   How many disk drives do you have? (1-3) : 2
   Do you wish to change the disk layout? (y or n) : y
   Do you wish to change drive 0? : n
   Do you wish to change drive 1? : y
Drive number : 1
      Number of virtual disks? : 2
       virtual disk number : 0     size: 20000
       virtual disk number : 1     size: 37600
   Is the swap area on these disks? (y or n) : y
   Enter the size of the swap area: 3200
   Would you like to see the disk layout? (y or n) : y
```

(Virtual) Disk	DRIVE 0 Offset	Size	DRIVE 1 Offset	Size
0	0	12000	0	20000
1	12000	3200	20000	37600
2	15200	6000	0	0
3	21200	6000	0	0
4	27200	30400	0	0

```
   Will you boot from any of these drives? (y or n) : y
   Do you want a new secondary boot made? (y or n) : n

Do you have SMD type drives on your system? (y or n) : n

Do you have a nine-track drive? (y or n) : n
Do you have any line printers? (y or n) : y
Do you have any cartridge tape drives? (y or n) : y
Do you have any ICP 8/Ø2 Intelligent Serial Processors?
(y or n) : n

Do you wish to add any of your own
device drivers? (y or n) : n

The major device numbers for the ZD drives is : Ø
The major device number for the SMD drives is : 8
The major device number for the MODEL 11 drives is : 1Ø

Enter the major, minor numbers for the root
filesystem, n,n : Ø,2
Enter the major, minor numbers for the swap
filesystem. m,n : Ø,1
Enter the major, minor pair for temp storage
device for pipes : Ø,2

What system name would you like to call this system?
 (8 characters max): xxxxxxxx

What network node name would you like to call this system?
 (8 characters max): xxxxxxxx

What version label would you like to call this system?
 (8 characters max): xxxxxxxx

making zeus
           Version X.Y
           4 lines
sld -Ns -o zeus -e start -X -i -Ms62 z.o event.o mch.o
 ver.o fpe.o conf1.o conf2.o ../dev/mt.d.o ../dev/lpr.o
 UNET.D.LIB ../dev/zd.o ../dev/smd.d.o ../dev/md.d.o
 ../dev/ct.o ../icp/zpd/icp.d.o ../icp/zpd/itty/itty.d.o
 ../icp/zpd/ilp/ilp.d.o ../icp/zpd/x25/x25.d.o
 ../icp/zpd/bsc/bsc.d.o ../icp/zpd/sna/sna.d.o
 ../icp/zpd/oly/oly.d.o ../icp/zpd/acu/acu.d.o
 ../icp/zpd/apt/apt.d.o ../icp/zpd/bs3/bs3.d.o
 ../sys/LIB1 ../dev/LIB2

new kernel: zeus
size of zeus: 82686 + 11776 + 56064 = 150526 = Øx24bfe
```

2. Move the old zeus to a file ozeus as shown,

```
mv /zeus /ozeus
```

and replace it with the new zeus:

```
mv /usr/sys/conf/zeus /zeus
```

3. Change the mode and owner with the command

```
chmog 644 bin 0 /zeus
```

4. Use the following procedure to reboot the system:

a) Enter

```
INIT 1          OR
kill -1 1       OR
/etc/down
```

(the down command is recommended if in multiple-user mode) and wait for the new zeus super-user prompt, which will be

```
#1
```

b) Enter the commands

```
sync; sync
```

This completes all current system I/O.

c) Press the RESET button on the System 8000 front panel, and wait for the prompt:

```
S8000 Monitor X.Y - Press START to load system
```

d) Enter

```
Z D
```

followed by RETURN.

The CPU monitor prompts the user for the name of the secondary bootstrapper with:

```
BOOTING FROM DISK
>
```

e) Enter

```
boot
```

The system console will display

```
(Non-) Segmented Jumper Configuration
Boot
:
```

f) Enter

```
zd(0,15200)zeus
```

to boot the kernel. The console will display the kernel message (with the date and release number), along with the new zeus prompt:

```
#1
```

The system is now in single user mode with the new zeus kernel.

5. Modify the directory /dev to include a new entry for each new filesystem. Use the command

```
/etc/mknod /dev/<devname> y <major device number>
     <minor device number>
```

where: devname is the device name, and is either zdxx for the non-raw device or rzdxx for the raw device; and xx is a two digit decimal number that is the same as the minor device number.

y is either b for the non-raw device, or c for the raw device.

major device number is 0 for zd type drives for both cases preceding (or 8 for block smd and 2 for character smd). It is used to access the disk device driver.

minor device number is a two digit number that identifies each filesystem on the disk. Minor device numbers are assigned sequentially for each filesystem on each disk, as indicated below:

system (first) disk:	0 through 9	(Unit 0)
second disk:	10 through 19	(Unit 1)
third disk	20 through 29	(Unit 2)
fourth disk	30 through 39	(Unit 3)

For the example, the block device names for the new files are zd1Ø and zd11 for zd type disks. Each filesystem must have one raw, and one non-raw device, respectively. The raw device has an "r" prefix to the device name. The raw device is a character device, and the non-raw device is a block device. To add these four files, the commands would be:

```
/etc/mknod /dev/zd1Ø b Ø 1Ø
/etc/mknod /dev/zd11 b Ø 11
/etc/mknod /dev/rzd1Ø c Ø 1Ø
/etc/mknod /dev/rzd11 c Ø 11
```

The same commands for smd type disks would be:

```
/etc/mknod /dev/smd1Ø b 8 1Ø
/etc/mknod /dev/smd11 b 8 11
/etc/mknod /dev/rsmd1Ø c 2 1Ø
/etc/mknod /dev/rsmd11 c 2 11
```

All entries in /dev are made with the command mknod(M).

6. Set the mode and owner of the new files properly. For the example above, use the following command for zd type disks:

```
chmog 64Ø bin Ø /dev/zd1Ø /dev/zd11
chmog 64Ø bin Ø /dev/rzd1Ø /dev/rzd11
```

To check the entries, use the command:

```
ls -l /dev
```

The result is a list of all the files in /dev. Among them, would be the following:

```
.
.
.
crw-r----- 1 bin    system Ø,  Ø Dec 31 16:39 rusr
crw-r----- 1 bin    system Ø, 1Ø Dec 31 19:5Ø rzd1Ø
crw-r----- 1 bin    system Ø, 11 Dec 31 19:5Ø rzd11
crw-r----- 1 bin    system Ø,  2 Dec 31 18:54 rroot
crw-r----- 1 bin    system Ø,  3 Dec 31 16:4Ø rtmp
crw-r----- 1 bin    system Ø,  4 Dec 31 14:26 rz
.
.
.
```

```
brw-r----- 1 bin    system 0,  0 Jul 31 19:01 usr
brw-r----- 1 bin    system 0, 10 Dec 31 19:49 zd10
brw-r----- 1 bin    system 0, 11 Dec 31 19:49 zd11
brw-r----- 1 bin    system 0,  2 Jul 31 19:01 root
brw-r----- 1 bin    system 0,  3 Jul 31 19:01 tmp
brw-r----- 1 bin    system 0,  4 Jul 31 19:01 z
```

7. Use **vi** to add new file entries for both mount(M) and umount(M) routines in /etc/mfs and in /etc/umfs. Name the new directories at this time. For the example above, use the command:

```
vi /etc/mfs
```

A typical file is displayed as follows:

```
: "@[$]mfs  <whatstr version> <date>  <time>   - Zilog Inc"
        case $0 in
                /etc/mfs | mfs)
                        /etc/devnm / |grep root |/etc/setmnt
                        /bin/echo 'mounting:'
                        /etc/mount /dev/z /z
                        /etc/mount /dev/tmp /tmp
                        /etc/mount /dev/usr /usr
                        /etc/mount ;;

                /etc/umfs | umfs)
                        /bin/echo 'un-mounting:'
                        /etc/mount |grep -v root
                        /etc/umount /dev/z
                        /etc/umount /dev/tmp
                        /etc/umount /dev usr ;;

                *)
                        /bin/echo "$0: Unknown command" ;;
        esac
```

The new directories are to be named z1 and z2. Add the following entries to the file following lines 8 and 15, respectively:

```
/etc/mount /dev/zd10 /z1
/etc/mount /dev/zd11 /z2

/etc/umount /dev/zd10
/etc/umount /dev/zd11
```

When this step is finished, the completed file will appear as follows:

```
: "@[$]mfs  <whatstr version> <date>  <time>   - Zilog Inc"
        case $0 in
                /etc/mfs | mfs)
                        /etc/devnm / |grep root |/etc/setmnt
                        /bin/echo 'mounting:'
                        /etc/mount /dev/z /z
                        /etc/mount /dev/tmp /tmp
                        /etc/mount /dev/usr /usr
                        /etc/mount /dev/zd10 /z1
                        /etc/mount /dev/zd11 /z2
                        /etc/mount ;;

                /etc/umfs | umfs)
                        /bin/echo 'un-mounting:'
                        /etc/mount |grep -v root
                        /etc/umount /dev/z
                        /etc/umount /dev/tmp
                        /etc/umount /dev/usr
                        /etc/umount /dev/zd10
                        /etc/umount /dev/zd11;;

                *)
                        /bin/echo "$0: Unknown command" ;;
        esac
```

8. Add new files to the startup script in /etc/rc_csh. For the example, the following line would be added to the check for filesystem consistency if a zd type disk has been added:

```
/etc/fsck -y /dev/rzd10 /dev/rzd11
```

When the script is checked, the lines for filesystem consistency typically would appear as shown:

```
# "@[$]rc_csh   <whatstr version> <date>  <time>   - Zilog Inc"
        echo
        uname -sn
        echo 'RC STARTUP SCRIPT'
        echo 'Multi-user Startup'
        echo

# Check for filesystem consistency
         /etc/fsck -y /dev/root
         /etc/fsck -y /dev/rusr /dev/rtmp
         /etc/fsck -y -t /scratch /dev/rz
         /etc/fsck -y /dev/rzd10 /dev/rzd11
```

NOTE

Both the -t option to fsck and a "scratch" filename argument within a filesystem other than the one to be checked should be used when checking a filesystem of greater than 30,000 blocks. This will avoid the need for any extra user intervention when the start-up script is invoked.

9. Add the new filesystem(s) to the fsck checklist in /etc/checklist. For the above example (zd type disk), the following lines would be added to the end of the file:

```
/dev/rzd10
/dev/rzd11
```

The complete file would then appear as:

```
/dev/rusr
/dev/root
/dev/rtmp
/dev/rz
/dev/rzd10
/dev/rzd11
```

10. Add new directories to the root filesystem (/) for each new filesystem. Use the following commands with the new file names:

```
mkdir /z1 /z2
```

11. Construct the new filesystems on the additional (new) disk. Use the system mkfs(M) command program (/etc/mkfs). This step will take considerably longer for larger filesystems.

Enter the command

```
/etc/mkfs <device id> <file size> m n
```

where: <device id> is the device name (for example, /dev/rzd10)

<file size> is the number of blocks that are to be in the filesystem

<m n> are interleaving factors. <m n> are given in Table 3-2.

For the example above (zd type disk), use the commands

```
/etc/mkfs /dev/rzdlØ 2ØØØØ 12 96
/etc/mkfs /dev/rzdll 37ØØ 12 96
```

where the interleaving factors are 12 96.

12. Label the filesystems using the labelit(M) command. The first argument to this command is the /dev filename entry, the second is the mounted name of the filesystem, and the third is the volume number. For the example case, the following commands should be entered:

```
/etc/labelit /dev/rzdlØ /zl 1
/etc/labelit /dev/rzdll /z2 1
```

13. Mount the new filesystems on directories zl and z2. Enter the commands for a zd type disk:

```
/etc/mount /dev/zdlØ /zl
/etc/mount /dev/zdll /z2
```

14. Write a new shell script /etc/mlf to add "lost and found" directories for each of the new filesystems. Use the filesystem names zl and z2. This new script will make lost and found files lØ blocks long for any files in "foreach". The completed file should look like:

```
#
foreach i ( /zl /z2 )
set dir="$i/lost+found"
        if (! -e $dir) then
                mkdir $dir
                chmog Ø75Ø zeus Ø $dir
        endif
        @ j=318
        while $j
                echo > $dir/Z$j
                @ j--
         end
        @ j=318
        while $j
                rm $dir/Z$j
                @ j--
        end
end
```

15. Enter the command:

    chmod 0700 /etc/mlf

16. Start the execution of the shell script by entering:

    /etc/mlf

    The script takes a few minutes to complete.

17. Change the mode and owner of the new "lost+found" directories. Use the command:

    chmog 644 bin 0 /z1/lost+found /z2/lost+found

18. Check for successful completion of the previous steps using the ls command:

    ls -l /z*

    The display should show the following for z1 and z2:

```
/z1:
total 10
drw-r--r-- 2 bin  system   5120 Dec 31 21:20 lost+found

/z2:
total 10
drw-r--r-- 2 bin  system   5120 Dec 31 21:29 lost+found
```

19. To enter multiple-user mode, enter the commands:

```
sync
INIT 2
```

    This completes the software modification procedures for the addition of a new disk.

4.2.3. Reconfiguration of the Original Disk

With the addition of a disk, it may also become desirable to change the layout (sizes and composition) of the first disk. To optimize the system, however, the default disk layout should be followed wherever possible (such as leaving the "command" programs within / in the central area of the disk to minimize head travel for frequently accessed programs).

Use the following procedure to change the first (system) disk layout:

NOTE

Note that the root, /tmp, and /usr should not be smaller than the default sizes, since the restor may not work if they are.

1. Run sysgen to generate a new kernel for the new configuration. When the new kernel is created, copy the old file, /zeus, to /zeus.old and move the new kernel to the root filesystem with the name /zeus. If a new secondary bootstrapper was created, move it to the filesystem which will be the first filesystem on disk in the new configuration. Name it "boot".

2. Dump each filesystem to a separate tape.

3. Bring the system down.

4. Insert the system distribution tape, press RESET and type in "Z T".

5. Type in

   ct(Ø,3)

   to invoke the stand-alone mkfs(M) program. The prompts and appropriate Model 21 replies are:

   ```
   file sys size: x
   filesystem: zd(Ø,y)
   interleaving factor (m n):
   ```

   where x is the new size of the root; x should be greater than or equal to 6ØØØ, since the new root will be restored from the dump of the root just taken. y is the block offset on disk where the new root filesystem starts.

   mm and nn are the interleaving factors. See Table 3-2.

6. Type in

   ct(Ø,4)

which invokes the stand-alone restor(M) program.

The restor program prompts

```
Will you be restoring from a factory-supplied
Zilog release tape (y or n)?
```

Enter

```
n
```

and insert the tape with the dump image of the root on it. The prompts and appropriate replies for Model 21 are:

```
Tape? ct(Ø,Ø)
Disk? zd(Ø,n)
```

where n is the block offset of the new root filesystem. This command restores the root filesystem from the dump just taken.

7. Type in (change "zd" to "smd" for Model 31 systems)

```
zd(Ø,n)zeus
```

where n, again, is the block offset of the root filesystem that was restored. The name of the kernel that was generated using sysgen(M) should be "zeus". The system should now boot.

8. Issue appropriate mknod(M) and mkfs(M) commands as described above, and follow steps 1 through 7, above. Then restore and label the filesystems dumped previously, using restor(M) and labelit(M).

9. If it is desirable to have a maintenance kernel to boot (see Section 4.3), /tmp should be no smaller than the root filesystem. A new kernel should be sysgened. When the major, minor numbers for the root are requested, reply with the major, minor number for /tmp. The major, minor number for temporary storage for pipes should also be that of /tmp. The new kernel should be called "zeus.maint" and moved to the root. Finally, dump the contents of this new root to tape for use when necessary to boot from the maintenance filesystem.

4.3. Maintenance File System

If fsck cannot fix a corrupted root filesystem, it may be necessary to boot using a maintenance filesystem to clean up an unmounted root. In the configuration above (which is enforced at boot time), the filesystem /dev/tmp is the same size as the root filesystem, /dev/root. Thus /dev/tmp can be used as the root filesystem if the real root is corrupted. This is possible because /tmp, the file name linked to /dev/zd3 or /dev/smd3, is not used by the commands that are ordinarily issued to clean up /dev/root, and thus need not be separately mounted. Use the following procedure to boot (Model 21) using /dev/tmp as the maintenance filesystem.

NOTE

To boot using /dev/tmp as the maintenance system for the Model 31, substitute "smd for "zd" in steps five, six, and seven of the following procedure. The device files are named /dev/smd<X> rather than /dev/zd<X> for the Model 31.

1. Bring the system down.

2. Insert the system tape in the tape drive.

3. Press RESET.

4. Enter

```
ZBOOT T
```

5. In response to the secondary bootstrapper's prompt, enter "ct(0,3)". This executes the stand-alone mkfs program. The prompts and replies for this program for a Model 21 are:

```
file sys size: 6000
filesystem: zd(0,21200)
```

   This remakes the filesystem /dev/zd3 (or /dev/smd3 for Model 31 systems).

6. Use the following procedure to restor the root filesystem. In this case, the root filesystem, on tape, is being restored in the filesystem on disk usually associated with /tmp. The restore should take about 15 minutes.

Step 1.

display: Will you be restoring from a factory supplied Zilog release tape (y or n)?
enter: "y"

Step 2.

display: Do you want instructions (y or n)?
enter: "n"
note: Instructions are described in paragraph 3.5, step 4. Enter "y" for instructions; else enter "n"

Step 3.

display: Do you want to restor the root filesystem (y or n)?
enter: "y"

Step 4.

display: Do you want to restor the /usr filesystem (y or n)?
enter: "n"

Step 5.

display: Restoring the root filesystem onto 'zd' type disk:
note: Press RETURN if disk type is zd; else enter disk type.

Step 6.

display: disk unit: Ø
enter: RETURN or a number
note: Press RETURN if disk unit is Ø; else enter disk unit number Ø through 3.

Step 7.

display: offset 15200:
enter: 21200
note: Entering 21200 writes on /tmp instead of /.

Step 8.

display: Tape Unit number?
enter: Ø
note: Be sure to insert the cartridge tape in drive Ø.

Step 9.

display:
Restoring from tape drive #0
Onto device zd(0,21200):
Common / filesystem
Special / for Model 21
note: Console displays restor constants from above steps.

Step 10.

display: OK to restor (y or n)?
enter: "y"

Step 11.

display:
Restoring / filesystem
Done

7. The secondary bootstrapper prompts again when the restore is done. The prompt and appropriate reply in the Model 21 are:

```
Boot
:zd(0,21200)zeus.maint
```

This causes the maintenance kernel, which considers /dev/tmp as the root, to be booted.

Now /dev/root can be repaired in single-user mode either by running the program fsck or by running individual commands such as icheck(M) and dcheck(M). To use fsck, extract from /etc/rc_csh the line which invokes fsck, and issue that line as a command. Alternately, refer to Section 4.5 and follow the instructions carefully.

When /dev/root is repaired, manually reboot the system with the old root filesystem (/dev/root), and, in single user mode, issue the command:

/etc/mkfs /dev/tmp 6000 <m> <n>

WHERE FOR MODEL 21:

m=12 n=96

WHERE FOR MODELS 20, 30 and 31:

refer to Table 3-2 for the values of m and n

This remakes the filesystem associated with /tmp. The system can now be brought up multi-user.

4.4. Dumps

To dump the contents of a filesystem to tape as a precautionary backup, the user usually issues a command like

```
/etc/dump 0u /dev/rz
```

This copies the files from disk filesystem /dev/rz onto tape.

The complementary command restor(M) can be used to restore the files from the tape back to the disk. The command:

```
/etc/restor r /dev/rz
```

restores the files from tape back onto the /dev/rz filesystem on disk.

Only unmounted filesystems should be dumped, because activity in a filesystem being dumped can make it an unreliable backup.

The characteristics of each installation determine how frequently dumps should be made. For most installations, a weekly level 0 dump and daily level 1 dumps are advisable for active filesystems. For less active filesystems, the weekly level 0 dump is still a good idea, but the level 1 dumps need not be taken so frequently.

To minimize the impact on the user community, dumps can be made at some "off" time through the use of the at(1) command. A tape is left in the transport and the dump is scheduled to be taken at some later time.

It may be desirable to do a tape dump, a mkfs(M), and a tape restoration of a filesystem occasionally (eg. every three to six months). This dump/mkfs/restor re-allocates all of the blocks in each file in an optimal way. Active filesystems become fragmented over time, so the performance improvement can be noticeable.

4.5. Managing File System Consistency

The fsck(M) program is used to check filesystem consistency as part of the /etc/rc_csh script that is automatically executed after a manual or automatic boot when bringing the system into a multiple-user state. It can also be entered manually.

However it is run, fsck(M) performs the necessary repairs in almost all cases. However, occasionally the fsck program may encounter a problem that requires manual intervention to correct.

This section discusses various file consistency problems and recovery procedures. The following subsections each describe one of the possible errors and the corrective action that must be taken.

Fsck actually performs the equivalent of an icheck(M) and dcheck(M) on (mostly unmounted) filesystems. Icheck checks for block consistency of a filesystem. For each filesystem, it returns messages of the form:

```
files    149 (r=128, d=13, b=3, c=5)
used     1721 (i=57, ii=0, iii=0, d=1664)
free     3245
missing  0
```

This is interpreted as follows:

- Files: there are 149 files in the filesystem; 128 are regular files, 13 are directories, 3 are block special files and 5 are character special files.
- Used: there are 1721 blocks used in the filesystem, there are 57 indirect blocks, no doubly or triply indirect blocks, and 1664 direct blocks.
- Free: there are 3245 blocks on the free list.
- Missing: there are no missing blocks. Blocks are missing when they are not in a file and not on the free list.

Dcheck checks the consistency between link counts in inodes and directory entries. If dcheck is successful, it returns nothing except the name of the device being checked.

Icheck and dcheck can report different types of errors. These errors and appropriate recovery procedures are discussed below.

NOTE

Whenever possible, make all error recoveries on unmounted or quiescent filesystems. If the root filesystem is inconsistent, make all fixes to the raw device /dev/rtmp. Then reboot, but do not issue any sync commands prior to rebooting. Likewise, before rebooting, do not issue a regular icheck or dcheck of the root, because these commands may issue sync commands. An alternate method is to boot from a maintenance filesystem and work on an unmounted root.

4.5.1. Missing Blocks

The icheck(M) program detects an error involving missing blocks. Recover by doing an icheck with the "-s" option, which restructures the free list. Enter:

```
/etc/icheck -s /dev/...
```

where ... is the filesystem in error.

Then do a regular icheck:

```
/etc/icheck /dev/...
```

This ensures there are now no missing blocks. Missing blocks are not too serious and can be reclaimed at the user's convenience.

4.5.2. Duplicate Blocks

The icheck(M) program detects an error involving duplicate blocks:

```
53311 dup;  inode=81  class = free
```

The word "dup" indicates that logical block #53311 is both on the free list and in a file. Recover by doing an icheck with the "-s" option, as described above. Then do a regular icheck. This removes the blocks in question from the free list. This is a serious error and should be repaired immediately.

4.5.3. Bad Freeblock

The icheck(M) program detects an error in addressing a block. This indicates that a block number outside the available space was encountered in the free list. To recover, do an icheck with the "-s" option.

4.5.4. Bad Block

The icheck(M) program detects an error in addressing a block and is indicated by a message similar to a "duplicate block" message:

```
53312 bad;  inode = 58, class = free
```

In this example, block 53312 lies outside the filesystem on the disk. Since the class indicates that the block is on the free list, an "/etc/icheck -s" should be performed. Issue an icheck

and a dcheck to check filesystem consistency. Recovery is important and should be done promptly.

4.5.5. No Directory Entries

The dcheck(M) program detects an error in the accounting information associated with a file. Part of this message is:

```
/dev/...
          entries        link cnt
4348        Ø                1
```

In this example, the link count in inode 4348 is one but there are no directory entries for this file. Therefore, this file will never be deallocated. To recover run clri(M) on the inode. For example:

```
/etc/clri /dev/z 4348
```

Then execute an icheck on the filesystem:

```
/etc/icheck -s /dev/z
```

This puts the disk blocks back on the free list. Finally, perform an icheck and dcheck on the filesystem.

If the fields for both the entries and link count fields are zero, use the same recovery procedure.

If the dcheck indicates at least one directory entry for the file, but the link count is still greater than the number of entries, then there are two options:

1. Do nothing. When all the directory entries are removed for that file, the inode will not be deallocated. Some disk space will be lost, but the situation does not degenerate.

2. Remove the directory entries by removing the file(s), and clear the inode. The method for doing this, and for obtaining the ASCII file name(s) for the file(s), is described for Too Many Directory Entries, below.

4.5.6. Too Many Directory Entries

The dcheck(M) program detects an error in the accounting information associated with a file. Part of this message is:

```
/dev/...

          entries       link cnt
4348        2             1
```

This message indicates that a particular inode has more directory entries than links, as indicated by the link count. Eventually, one or more directory entries may point to an inode that is either not allocated; or worse, may point to a different file. To recover:

1. Discover the file name associated with the inode in question by doing an ncheck(M). For example:

   ```
   /etc/ncheck -i 2001 /dev/zd5
   ```

   Ncheck returns the ASCII name of the file corresponding to inode 2001.

2. Remove the file from the filesystem. If the file is important, first attempt to copy it to another file in a clean filesystem first. To remove a file:

   a. Mount the filesystem. For example:

      ```
      /etc/mount /dev/z /z
      ```

      See /etc/mfs to determine the correct mount command.

   b. Remove the file with the rm(1) command.

   c. Finally, issue the command:

      ```
      /etc/umount /dev/z
      ```

3. Issue the following commands to complete the recovery:

   ```
   /etc/clri /dev/z 2001
   /etc/icheck -s /dev/z
   /etc/icheck /dev/z
   /etc/dcheck /dev/z
   ```

   No errors should be found.

4.5.7. File Block Outside the File System

The icheck(M) program detects an error in addressing an allocated block and is indicated by a message similar to "Bad Block":

```
53312 bad;  inode = 25, class = data (small)
```

In this example, block 53312 lies outside the filesystem on the disk. It is a block that is part of a file, and it is a direct block because it is described as "small". (The adjectives "large", "huge" and "garg" indicate that the block is accessed singly, doubly or triply indirect, respectively.) To recover:

1. Remove the file from the filesystem. Use the procedure for Too Many Directory Entries, above.

2. Because the file's blocks were returned to the free list, the free list may have a block that is outside the available filesystem space. Perform an:

   ```
   /etc/icheck -s /dev/...
   ```

3. Check filesystem consistency with an icheck and a dcheck.

4.5.8. Duplicate Blocks in Files

The icheck(M) program detects an error in addressing an allocated block. Part of this message is:

```
53312 dup;  inode = 25, class = data (large)
```

In this example, block 53312 is in two files, and the inode for one of these files is 25. The inode and file name for the other file should be retrieved in order to correct the problem. In order to retrieve them, first issue an icheck command with the -b option. This produces the inode numbers of all the files that contain the duplicated block. After this is done, find the file names from the inode numbers as described above for a file block outside the filesystem. One of these files must be removed. The administrator must use discretion in deciding which file to remove. Usually, this file was open at the time of the system crash. Use adb(1) to examine the contents of the duplicate block. Sometimes the contents are obviously part of one file. In any case, if the file is important, try to copy it to another filesystem before removing it. After one file is removed, perform an icheck -s, and then an icheck, and finally, dcheck to ensure consistency.

SECTION 5
SYSTEM GENERATION (``sysgen'')

System generation (sysgen) is an on-line tool for modifying and generating a new version of the ZEUS kernel. The user can modify certain system constants, change the configuration of the disk(s), modify the location of the root filesystem, or add user-written device drivers to the system. Most installations use sysgen(M) only to adjust the timezone information to reflect the location of the system. The other tuning parameters should not need adjustment in normal circumstances. It is possible to seriously degrade system performance or even to create a non-functional kernel by inappropriate adjustments of parameters.

Many of the parameters are used as bounds on arrays. Their initial settings have been chosen to make maximum use of the kernel address space. In most cases, the number of big (disk block) buffers (NBUF) is the last parameter adjusted. Its value is usually set to occupy the remaining address space. For example, decreasing the number of processes (NPROC) from the default setting may free enough memory to allow the number of buffers (NBUF) to be increased. Although it is possible to create a slightly smaller kernel by setting NBUF to a smaller value, system performance will probably be degraded.

To use sysgen, change directory to /usr/sys/conf, and then execute /etc/sysgen.

See sysgen(M). If the -d option is not used, sysgen asks the user a number of questions, creates some temporary files, and then builds a new kernel from the available object files. The questions and the procedures are described below. See Section 4.2.2 (step 1) for an example of sysgen operation.

1. System Constants

The first sysgen question is:

Do you wish to change any system constants? (y or n):

Either "y" or "yes" invokes the editor **vi** (see vi(1)), which allows the user to edit the file /usr/sys/h/sysparm.h. This file contains all the modifiable system constants. The recommended response is "n", for no (unless simply changing the time zone quantity within the TIMEZONE constant definition).

CAUTION

Do not change any of the system constants unless necessary.

Do not change the value of any system constant to be a nonsense value (e.g. setting the size of the open file table to Ø).

To return from vi, enter ":x" or "ZZ". The system will then prompt for disk information (next step).

Table 5-1 contains a typical list of constants, default values and meanings.

Table 5-1 Typical System Constants

```
#define CANBSIZ    256  /* max size of input line from terminal  */
#define DSTFLAG      1  /* Daylight Savings Time flag            */
#define MAXMEM     512  /* max memory per process                */
                        /* (256 byte blocks)                     */
#define MAXUPRC     2Ø  /* maximum number of processes per user  */
#define NBUF1       44  /* number of big buffers in buffer pool 1*/
#define NBUF2        1  /* number of big buffers in buffer pool 2*/
#define NBUF3        1  /* number of big buffers in buffer pool 3*/
#define NBUF4        1  /* number of big buffers in buffer pool 4*/
#define NBUF5        1  /* number of big buffers in buffer pool 5*/
#define NCLIST     1ØØ  /* number of small buffers for term io   */
#define NFILE      175  /* size of system open file table        */
#define NINODE     2ØØ  /* number of in-core inodes              */
#define NMOUNT      2Ø  /* number of mountable filesystems       */
#define NPROC       7Ø  /* max number of active processes        */
#define NTEXT       4Ø  /* max number of shared text segments    */
#define TIMEZONE (8*6Ø)/* minutes westward from GMT              */
#define NFLOCK     1ØØ  /* number of lockable regions in a file  */
#define NESLOT      2Ø  /* number of slots for error logging     */
```

2. Disk Information

   a. Different types and sizes of disks can be used with various models of the System 8ØØØ. These drives are listed in Table 3-2 in Section 3.

      Sysgen asks if the system model number is 11:

      Is this kernel for a Model 11 system? (y or n) :

If the response is "y" or "yes", then sysgen asks for data pertinent to the disk units on those models only. The proper response for Model 21 and 31 systems is "n" or "no". If the answer is "n" or "no", then sysgen asks if the system has ZD type disk drives (non -SMD interface):

Do you have ZD type drives on your system? (y or n) :

If the answer is "n" or "no", then sysgen asks for data pertinent to SMD interface disks:

Do you have SMD type drives on your system? (y or n) :

One of the three questions must have an affirmative answer.

b. Sysgen first asks for the number of disk drives in the system:

How many disk drives do you have? (1-4):

Enter the number of physical drives in the system configuration. The maximum number is four for Model 21 and eight for the Model 31 (consisting of four "zd" and four "smd" type drives).

c. Sysgen then asks for data relevant to disk layout:

Do you wish to change the disk layout? (y or n):

A default disk layout is assumed if the answer to this question is "n" or "no". Avoid "fancy" disk layouts; it is very easy to do something wrong, or leave something undone. The default partitioning of the first and second disks should be appropriate for most ZEUS users. The default layouts for the System 8000 drives are shown in Table 4-1.

This partitioning is optimized to increase disk access performance. There should be a good reason to change the layout of the first two physical drives. For 3 drives, the layout should probably be changed to improve performance and accommodate local needs (e.g. putting two heavily used filesystems on separate drives, or, if on the same drive, close together).

Remember to create the proper special device files, modify the /etc/rc_csh script, and modify the /etc/mfs after installing the new filesystem.

NOTE

If changing the disk layout, be sure to save a copy of the old boot program first (/usr/boot).

If the layout does need to be changed, sysgen steps through each of the physical drives, asking whether the layout should be changed. If so, sysgen asks for the number of virtual disks (partitions) on that drive, and for the size of each partition. The maximum number of blocks for a given physical disk is in Table 3-2. All blocks on a given disk must be allocated. The maximum number of partitions on a disk is 10.

d. Sysgen next asks:

Is the swap area on these disks? (y or n):

Answer "y" unless you have both smd and zd disks and you wish to place the swap area on a different kind of disk.

e. Sysgen next asks for the size of the swap area, in blocks. This size must match one of the virtual disk sizes entered above, or for the default size in Table 4-1.

f. Next sysgen asks:

Would you like to see the disk layout? (y or n):

If the answer is "y", the layout is shown for all disks configured in the system.

NOTE

Request a new bootstrapper only if the location of the root has changed, and the root is on the disk(s) being changed.

g. The sysgen program creates a new secondary boot program, if requested, that looks for /zeus in the correct place when booting is automatic. Next, sysgen asks for the location of the root, i.e. the physical drive number and block offset on that disk (the default

values are Ø and 15200, respectively). The sysgen program checks to ensure that the offset given for the root also exists in the disk layout tables. Then it prints the messages:

```
making new boot
new boot in current directory : boot
```

To install this new secondary bootstrapper for the new kernel, move it to /usr/boot (default disk layout), or, the first filesystem on drive Ø (non-default layout). Do this only after the new kernel is installed, because the new boot might not work automatically with the old kernel.

3. Nine-Track Tape Option

Sysgen prompts:

```
Do you have a nine track magnetic tape drive? (y or n) :
```

Enter "y" or "yes" if a nine-track magnetic tape drive is installed with the system.

4. Line Printers

Sysgen next asks:

```
Do you have any line printers? (y or n) :
```

5. Cartridge Tape

Next sysgen asks:

```
Do you have any cartridge tape drives? (y or n) :
```

Since the standard model configurations always include a cartridge tape, the answer should be "y".

6. Intelligent Communications Processor (ICP)

Sysgen asks:

```
Do you have any ICP 8/Ø2 Intelligent Serial Processors?
(y or n):
```

If you answer "y" sysgen then prompts:

How many ICP 8/Ø2's do you have? (1 to 3) :

NOTE

The following questions appear only if files have been properly installed using the Zilog release tape associated with these options.

Do you wish to have the Intelligent Tty protocol included? (y or n):

Do you wish to have the Intelligent Line Printer protocol included? (y or n):

Enter the correct quantity in response to the first question, and "y" for each of the desired protocols.

7. Device Drivers

Sysgen next asks:

Do you wish to add any of your own device drivers? (y or n) :

To add user-written device drivers to the kernel, answer "y" or "yes". Sysgen asks for the .o file name of the compiled device driver. The user can add up to six drivers in this manner. Continue to the next question by pressing RETURN.

Writing a device driver and making it work are not trivial tasks. The basic shells for all the required routines are provided in /usr/sys/dev, in the files udev1.c, udev2.c, etc.

Use the mknod(M) command to set up the appropriate special files in the directory /dev. The major device numbers for the six user devices are 16-21, respectively. Minor device numbers must be determined by the user, as they are device dependent.

8. File System Location

If the disk layout has been changed from the default, then sysgen(M) asks the user to enter the major, minor number pair for three separate areas on disk: the root filesystem, the swap area, and the temporary storage area for pipes. The major device

numbers differ for the smd and zd type disk drives as indicated in Section 4.2.2.

The swap filesystem is where all processes are swapped. The temporary storage area for pipes should be located in the same filesystem as the root, to allow pipes to be used in single-user mode. Again, these should not be changed without careful prior consideration.

9. Naming Your System

Sysgen prompts:

> What system name would you like to call this kernel? (8 characters max) :

This question helps identify the kernel via the uname(2) system call. This name may be up to eight characters and may include blanks, numbers and special characters. It replaces the default system name of "ZEUS" appearing in several prompts including the login prompt.

10. Network Node Name

Sysgen prompts:

> What network node name would you like to call this kernel? (8 characters max) :

This question helps identify the kernel via the uname(2) system call. This name may be up to eight characters and may include blanks, numbers and special characters. The network node name is used by uucp(1) to identify itself to other systems. For this purpose, imbedded blanks should be avoided.

11. Kernel Version Name/Number

Sysgen prompts:

> What version label would you like to call this kernel? (8 characters max) :

This question helps identify the kernel via the uname(2) system call. This name may be up to eight characters and may include blanks, numbers and special characters.

12. Kernel File Creation

Sysgen then invokes the make(1) utility to build a new kernel in the current directory. The display prints:

```
making <filename>
     Version <version number>
     4 lines
```

The version and line messages are generated by the chkout(1) program in the process of making ver.o. The next message is the input to the ZEUS loader, which links the various pieces of kernel object code together and resolves external references. The next message is:

```
new kernel : zeus
```

The filename will be "zeus" unless the -f option was used when sysgen(M) was executed.

The final message states the sizes of the various pieces of kernel. For example, the message:

```
size of zeus: 82686 + 11776 + 56Ø64 = 15Ø526 = Øx24bfe
```

tells the user that the size of the kernel text area is 82,686 bytes, the kernel data area is 11,776 bytes, and the bss area is 56,Ø64 bytes.

To install the new zeus, move /zeus to /ozeus, move /usr/sys/conf/zeus to /zeus, and then reboot the system.

The most common mistakes in running sysgen are:

[1] Not having write permission in /usr/sys/conf. The user must be the owner of /usr/sys/conf or the super-user, because sysgen creates some temporary files there.

[2] Undefined symbols from the loader. These will probably be from user-defined device drivers. All the necessary driver routines must be defined.

[3] The generated kernel does not boot. In this case possible causes can include giving the wrong major,minor pair for the root to sysgen; defining a bad disk format; or overlooking installation of a new boot when changing the disk layout. If the old kernel does not boot manually, then it may be necessary to boot from tape or a backup filesystem.

[4] Upon booting, the fsck(M) program removes many files in a filesystem. This is usually a result of an incorrect declaration of the filesystem size during sysgen. If this occurs, interrupt the fsck program immediately and

try booting an old kernel. (All user files should have been backed up to tape, so their recovery is possible.)

SECTION 6
SYSTEM CRASHES AND OTHER PROBLEMS

6.1. System Crashes

If the system crashes, note the cause (if possible) from the message on the system console. If there is a message, it starts with the word "panic:", and is followed by other information. Panic messages and their meanings are listed below. All of these are "should never happen" messages, and indicate a serious hardware or software problem. If the kernel has been recently changed, it is a prime suspect. If not, there is probably a hardware problem.

6.2. Panic Messages

blkdev
: The kernel was called to get a block from a filesystem represented by a nonexistent major device number.

devtab
: Block in nonexistent device requested.

EIT on ICP<x>
: An Extended Instruction Trap occurred on ICP<x>, where x is a number from Ø through 2. To recover, stop the ICP, and re-load the ICP protocol tape. This is a fatal (disabling) error only for those devices connected to ICP number x.

EPU instruction
: An EPU instruction trap occurred while in system mode.

ICP<x> Parity Error
: The system detects corrupted memory on ICP<x>, where x is a number from Ø through 2. To recover, stop the ICP, and re-load the ICP protocol tape. This is a fatal (disabling) error only for those devices connected to ICP number x.

ICP<x> Uninitialized Vector Entry
: A critical error is detected on ICP<x>, where x is a number from Ø through 2. To recover, stop the ICP, and re-load the ICP protocol tape. This is a fatal (disabling) error only for those devices connected to ICP number x.

init
: An I/O error occurred during initialization while reading the super-block for the root filesystem.

IO err in swap
An I/O error occurred while swapping.

Kernel segmentation violation
An MMU complained.

Kernel system call
System call routine was entered from system mode.

<zd> or <smd>: fatal error
An unrecoverable error status was returned from the disk controller, indicating the system disk is unusable. To recover, reboot. If the reboot fails, turn power off for a few minutes and then reboot again.

no fs
A device has disappeared from the mounted device table in the kernel.

no imt
Same as above but produced from a different kernel routine.

no procs
Internal system fork can't find process entry in table.

Nonvectored interrupt
Spurious interrupt.

Out of swap
A program needs to be swapped out, and there is no more swap space.

out of swap space
Same as above.

Privileged instruction
Privileged instruction interrupt happened from system mode.

restart
Control erroneously jumped to location Ø.

Running a dead proc
A context switch is made to a bad process.

Unexpected interrupt
Spurious interrupt.

zero wchan
A process is sleeping on the wrong internal channel. The system must be rebooted. If the boot does not work (for example, the prompt is never issued) the files needed to boot are probably not intact (most likely, /etc/init). To correct, restor(M) the root filesystem from tape. In this case, boot from a maintenance filesystem. Then check the root filesystem consistency with fsck or icheck and dcheck. When it is consistent, mount the old root on a new directory, and check the files necessary for booting. The following commands are an example:

```
mkdir /t
/etc/mount /dev/root /t
ls -l /t/etc/INIT /t/dev/console /t/bin/csh /t /zeus*
<here, move files over to root, if necessary>
/etc/umount /dev/root
```

The directory /t is made in the (maintenance) root filesystem, which is currently on /tmp. zeus should be linked to the file zeus3_Y.Z. Thus, the file /t/zeus2_Y.Z and the link /t/zeus should exist. Note that the file zeus_maint is linked to zeus3_Y.Z - where Y.Z is the release number.

6.3. Other Messages

The kernel also prints a number of other messages indicating serious, but non-fatal problems. These messages are:

bad block on dev x/y

A block in a file is not located between the i-list and the end of the filesystem on disk. x is the major device number and y is the minor device number. Minor device numbers are the individual filesystems on disk. The file /etc/mfs shows what filesystems are mounted under which names. For the typical system with "zd" type disks, these are:

Major (raw)	Major (block)	Minor	File System
Ø	Ø	Ø	/usr
Ø	Ø	1	swap area
Ø	Ø	2	root (/)
Ø	Ø	3	/tmp
Ø	Ø	4	/z

To recover, bring the system down and reboot, having fsck(M) fix the filesystem consistency problem.

bad count on dev x/y

A consistency check on the super block of a mounted filesystem failed. Chances are, the filesystem super block has been corrupted. To recover, dump to tape (dump(M)), do a mkfs(M), and restore (restor(M)).

Bad free count on dev x/y

The super block of the filesystem associated with major device x, minor device y is full, or corrupted. The size of the free block list, kept in the super block, is bad. To recover, try to delete some of the files on the x/y device. If this does not clear the trouble, dump the filesystem onto tape, remake using mkfs(M), and restore the filesystem using restor(M).

err on dev x/y bn=n er=Øxm,Øxo

A device error occurred when accessing the device associated with major/minor device pair x/y. (Other information given by the message is for Zilog internal use only.) To recover, bring the system down, power down the system, and then power up and reboot. If the error still occurs, contact the nearest Zilog Field Service office.

Out of inodes on dev x/y

There are no more inodes available in the filesystem associated with major device number x, minor device number y. To recover, either delete some files from the filesystem or increase the size of the filesystem.

Out of space on dev x/y

There is no space on the filesystem associated with major/minor device numbers x/y. To recover, the filesystem must be "cleaned up" to make more space. Because the occurrence of this condition can cause seriously incorrect actions to occur (e.g. unsuspected truncation of a file), it should be addressed before there is no space left. As a general rule, at least ten percent of the space in a filesystem should be free. If the count drops below ten percent the administrator should take steps to get users to clean up. The programs df(M) (disk free space) and fsck(M) can be used to indicate the amount of available free space in a filesystem. The find(1) and quot(M) commands can then be used to find users who have excessive amounts of space in

use, or files that are candidates for deletion or archiving to tape (see tar(1)).

Inode table overflow

The in-core inode table is full. Wait until some processes naturally die (or close files) or, alternately, "kill" uncritical processes that may have files open. If the tunable parameter NINODE was changed with sysgen(M), increase the size of this parameter. If this is not the case, and the error occurs with some regularity, use sysgen(M) to increase the size of NINODE.

<zd> or <smd>: unexpected intrpt Øxn ignored
<zd> or <smd>: fatal error on unit n
<zd> or <smd>: unrecoverable error on unit n

In all of the above cases, the disk driver encountered an error that couldn't be corrected. In all cases, reboot the system. If the error continues to occur, contact the nearest Zilog Field Service office.

no file

The global open file table is full. Given the large number of slots available, this is a very unusual condition. If desirable, "kill" some processes that may have open files, or wait until some processes die before trying again. Alternately, if the size of the open file table was diminished by running sysgen(M), it may have been made too small; the size should be increased. In any case, it may be desirable to increase the tunable parameter NFILE using sysgen(M).

Nonmaskable interrupt

An NMI was received, due either to a manual press of the START button, an impending power fail or a memory error. In all cases but the first, the system will go down, and should be rebooted. In the case of a memory error, if the rebooted system generates the same error, contact the nearest Zilog Field Service office.

no procs

There are no free processes. This message indicates an abnormal condition. Use ps(1) to get a list of all processes, and use kill(1) to get rid of the offenders. It may be desirable to use sysgen(M) to generate another kernel with a new value for NPROC.

proc on q

A request was made by the operating system to put a process on the run queue. However, the process appears to already be on the run queue. If this happens, reboot the system.

out of text

There are no more shared text table entries. If this happens more than very occasionally, use sysgen(M) to create a kernel with a larger table by increasing the parameter NTEXT. As with all changes to tunable parameters, be aware that some other parameter may need to be decreased to stay within the address space limits.

Warning: ECC error count high

See Section 6.5, Error Checking and Correction.

6.4. Troubleshooting

Two commands, kill(1) and ps(1) are important debugging tools. For example, if a user locks up the terminal or cannot kill a program with RUB or DEL, the system administrator can issue the command

```
ps -leaf
```

This produces a detailed display of the status of every process currently running in the system. Among the items shown is the process identification (pid) number. To delete an ongoing process, a kill command can be issued with the pid as an argument. If the pid that is specified in a kill command argument is that of the user's login shell, that command interpreter shell is aborted and the user is issued a new login prompt. This allows the creation of a newly initialized login shell. Such a procedure may be necessary if the original shell corrupts and is unable to communicate with the terminal. To make it a "sure kill," the argument -9 can be used as an option (some processes refuse to die otherwise). For example:

```
kill -9 <pid>
```

where pid is the process number of the doomed process as obtained from the ps(1) display.

6.5. Error Checking and Correction (ECC)

The System 8000 memory controller employs memory error checking and correcting circuitry that detects and corrects one-bit per byte errors. Correctable one-bit errors are referred to as "soft" errors. The ECC can also detect, but it cannot correct, two-bit (or "hard") errors.

The ECC also counts the soft errors. The operating system interrogates the ECC hourly, and if the error count exceeds 128, the operating system prints the following message on the system console:

```
Warning: ECC error count high
Soft errors encountered in the last hour = ddd
Memory bank error bit map = 0xhh
```

where: ddd is a decimal number not exceeding 255 (the ECC count stops at 255), and 0xhh is hexadecimal number.

If there have been less than 128 errors, no message is printed. If the error message is printed, there may be a memory malfunction. Although the system will continue to operate, for best results the malfunction should be located and corrected.

The memory bank error bit map indicates the sections of memory where one or more errors may have occurred. Each bit corresponds to one 256K byte section of memory, and the least significant bit corresponds to the lowest section of memory. If a bit is set in the bit map, it indicates that there was at least 1 soft error in the corresponding 256K byte section of memory.

The ECC also detects errors greater than one bit, but it cannot correct them. If such an error occurs, the ECC circuit responds with a non-maskable interrupt (NMI). When the operating system receives the NMI, it determines that the interrupt was caused by the ECC controller, performs a "panic" stop, and it prints a panic message:

```
panic: Uncorrectable ECC error
```

If an uncorrectable memory error causes a panic halt, repair the memory malfunction and re-boot the system.

6.6. Troubleshooting Aids

Pressing the START switch on the processor module (with the key switch away from the LOCK position), causes the system to print various status information at the console. The status corresponds to the state of the machine precisely when the button was pressed. Most of this information is unfamiliar to the average user. However some of it may be useful, if not to users, then to Zilog Technical Support personnel. The format of the status information is as follows:

<BANNER>
<Event id> <FCW> <pcseg> <pcoff> <state pointer>
<Contents of general purpose registers>

<Last Interrupt Handled>
<Pending Interrupts>
<Scheduler state>

<Last interrupted process, and its owner> OR
<Scheduling or Idle>

<Status of various segments of the last interrupted process>

<Last 16 or less items at the top of the system stack>

where:

BANNER indicates the mode of the CPU

eventid is the last vectored interrupt

FCW is the Flag and Control Word

pcseg and pcoff are the Program Counter

SECTION 7
SYSTEM MAINTENANCE

7.1. Line Printer Information

The kernel supplied with the system contains a line printer driver that handles up to three parallel-port printers. Each can have either a Centronics or Data Products interface. (Additional letter-quality or serial printers can be attached to unused tty ports, and all printers can be spooled together under the control of one spooling program.) The parallel-port line printer driver has been tested with the following line printers:

Anadex (Model DP-9501 - Centronics interface)

Printronix (Model P600 - Centronics interface, and also Data Products Interface. These interfaces do not support the nroff separate underline character.)

Centronics (Model 703/704)

Data Products (Model B600 - Data Products interface)

Data Products (Model 2230 - Data Products interface)

NEC Spinwriter (Model 5500 Series - Centronics interface)

The driver should work with most printers that have either a Centronics or a Data Products interface. The System 8000 hardware is configured for a Centronics printer interface. For a Data Products printer interface, disconnect the following shunts for the Centronics interface:

E13 from E14
E17 from E18

Connect the following shunts:

E14 to E15
E16 to E17

The supplied /dev entries for the line printer(s) are correct if both line printers have a Centronics interface; otherwise they should be changed before using the line printer (as described following).

When the system is shipped, there are two entries in /dev; one for each line printer. The command

```
ls -l /dev/lp*
```

produces the following type of output:

```
crw------- 1 zeus  system  9,  Ø May  6 ØØ:32 /dev/lp
crw------- 1 zeus  system  9,  1 May  6 ØØ:33 /dev/lp2
```

The first line above refers to the first line printer, and the second to the second line printer (connected via the terminal expansion board). Both define printers with a Centronics interface. If either printer has a Data Products interface, the entry for that device must be changed from that which is supplied. For example, if the line printer associated with the expansion board (the second line printer) has a Data Products interface, issue the commands

```
ls -l /dev/lp2
rm /dev/lp2
/etc/mknod /dev/lp2 c  9 5
```

which changes the entry for /dev/lp2 to reflect a minor device number of five rather than one. The minor device number determines the port number (Ø, 1, or 2), the interface type (Centronics or Data Products) and whether the interface is in raw or normal mode. The minor device number can be set to a value between one and fourteen where the value of each of the bits in a binary representation has the following meaning:

```
MSB          LSB
 Ø Ø Ø Ø r t a a

where:  r=raw interface if Ø
        r=normal interface if 1
        t=Centronics if Ø
        t=Data Products if 1
       aa=decimal Port Number Ø, 1 or 2
```

In decimal values, the minor device number is interpreted as follows:

MINOR VALUE	MEANING
Ø	normal Centronics interface on port Ø
1	normal Centronics interface on port 1
2	normal Centronics interface on port 2
4	normal Data Products interface on port Ø
5	normal Data Products interface on port 1
6	normal Data Products interface on port 2
8	raw Centronics interface on port Ø
9	raw Centronics interface on port 1
1Ø	raw Centronics interface on port 2
12	raw Data Products interface on port Ø
13	raw Data Products interface on port 1
14	raw Data Products interface on port 2

Raw mode is currently unsupported; it is intended to be a buffered, interrupt-driven parallel port that is different from normal mode because it allows any byte value to be written to it without generating a segmentation violation. Also it does not attempt to re-interpret tabs, carriage returns and so forth.

If a third printer is to be configured, enter the command:

```
/etc/mknod /dev/lp3 c 9 2
```

for a Centronics interface or

```
/etc/mknod /dev/lp3 c 9 6
```

for a Data Products interface.

The command "ls -l /dev/lp*" displays the owner and protection bits of the printer device files; change them as appropriate.

Disable the skip perforation and automatic line feed switches on any line printer connected to the system. Skip perforation, when enabled, causes the printer to skip several lines at the end of the page. Automatic line feed causes a line feed to be printed when a carriage return character is received. The system assumes 66 lines per page; set the associated hardware switch appropriately.

The special file associated with the second line printer is /dev/lp2. To print a file "filename" on the first line

printer using the queuer, enter:

```
nq -q lpr:1 filename
```

To print the file on the second line printer, enter:

```
nq -q lpr:2 filename
```

The option arguments, -q lp:n, specify the queue "lp" and the device "n". If the option is omitted, the first available device in the printer queue is chosen. The default is the first queue described in the configuration file (see Section 7.2.3).

If there is no second line printer attached to the system, issue the following command:

```
chmod ØØØ  /dev/lp2
```

Be sure to remove the device entry in the spooler configuration file (see Section 7.2.3). This makes the device inaccessable.

7.2. Printer Spooler Considerations

Maintenance of the queuer program primarily consists of providing and updating the configuration data regarding the kind and number of printers attached to the system.

7.2.1. Introduction

The line printer spooler outputs data, upon request, when a device (such as a line printer or a text-quality printer) is free. This has two advantages: first, a user program can output data any time, whether a printer is free or not. If a printer is not free, the request is remembered until a printer becomes free. Second, if there is more than one printer on the system, the spooler can output to either one. This allows the spooler to keep two devices busy, and is more efficient.

The ZEUS printer spooler consists of the program nq(1) and lpr(1) (which are linked together), for generating print requests; xq(1) for displaying and modifying previous requests; and dqueuer(M), with the backends /usr/lib/lp and /usr/lib/text for printing the requests. The system administrator has other functions available from the program xq. These are described in xq(M). See also backend(M). The /usr/spool/queuer directory holds the necessary configuration, status and request files to operate the spooling system.

The dequeuing process, dqueuer(M), runs at all times, and is started by /etc/rc_csh. Subsequent requests by nq(1), lpr(1), and xq(1) signal the dqueuer for service. The dqueuer then sets its process ID in a specific area in the active configuration file, /usr/spool/queuer/activeconfig. With this method, and various record locking techniques followed by all programs in the spooling system, all programs can inter-communicate without conflict. In addition, multiple copies of the dqueuer detect the existence of other copies, and immediately exit.

The dqueuer scans all devices and requests each time it is started. It dispatches child processes (called backends) to print a specific file on a specific device. If there are no available devices, or no ready requests, it goes to sleep, via pause(2), until it receives a signal from another process. Upon that signal, it restarts the scanning process. Nq(1) and lpr(1) both generate signals when they create a request. Xq(1) can generate signals if it needs to indicate a change in system status. The backends, through dqueuer(M), also generate signals when they finish printing a request. The above interprocess communication keeps the dqueuer printing when a device is open and when a request is ready to be printed, and quiescent when nothing is to be done. One exception to this occurs when a device is offline. In that case, the dqueuer polls the device once a minute to see if it came online.

Devices (printers) can be attached to the serial tty ports or the parallel printer ports on the System 8000. Then the configuration file (/usr/spooler/queuer/config) must be edited to link the device to the port, as described in Section 7.2.3.

7.2.2. Miscellaneous Spooler Commands

If a file is printing, and the printer fouls, the command

```
xq -q que:dev  -sd
```

stops ("-s") the current job and prevents (trailing "d") any further access to the printer device specified. The command

```
xq -q que:dev  -Ud
```

reenables access to the printer. See xq(M) and dqueuer(M) for other spooler control commands.

To delete a request in the queue that is not currently running, use the command

```
xq -d xxx
```

where xxx is the sequence number of the request to be cancelled. Xq(1) lists files and sequence numbers.

The printer spooler can also spool a text-quality printer that is attached to a tty port. To do so, use an editor program to change the first character in the third (action) field of the appropriate line of /etc/inittab to "k" (refer to Section 7.3.2). For example, if communications line Ø is to be used as the printer, the state 2 entry in /etc/inittab would change from this:

```
2:ØØ:c:/etc/GETTY ttyØ 2
```

to this:

```
2:ØØ:k:/etc/GETTY ttyØ !
```

Then use the command

```
kill -2 1
 OR
INIT 2
```

to reparse the /etc/inittab file and disable that tty port for logins. Then edit the spooler configuration file to reflect the new device (as described below).

7.2.3. Configuration File Description

The ZEUS printer spooler requires information about the device configuration of the system on which it runs. This information is contained in the configuration file, /usr/spool/queuer/config. The configuration file contains three types of records: comments, queue descriptors, and device descriptors. These are illustrated below.

The following example is for a system with two line printers (line printers 1 and 2), which are in the first queue; and one text-quality printer, which is in the second queue, and is connected to tty port /dev/tty6.

```
#
# Sample Configuration File
#
#      This file describes a system with 2 line
#      printers and one text quality printer.
#
#      first queue ... lpr has two line printers
#
Qlpr,R,N,S
```

```
#     line printer 1
D1,R,/dev/lp,/usr/lib/lp
#     line printer 2
D2,R,/dev/lp2,/usr/lib/lp
#
#     second queue ... text has the spinwriter
#
Qtext,R,N,S
#     text quality printer (typical option
#     shown ... this example for a qume)
D1,R,/dev/tty6,/usr/lib/text -T "9600 -n1"
#
#
```

The principal components of the above example are:

1. Comments. A comment line starts with a "#".

2. Queue descriptors. A queue descriptor starts with a "Q". The information on each queue record is:

queue name	1-8 characters
status	R=ready, D=down
minimum priority	R=rush, N=normal, D=deferred
selection criteria	F=first in first out, S=smallest file first

   One queue definition should be used for each type of physical device; for example, "lp" for a line printer, or "text" for a text-quality printer.

3. Device descriptors. A (physical) device descriptor starts with a "D." The device descriptor record follows the queue record with which it is associated. There should be a device descriptor for each physical device in the system configuration. The device descriptor record contains the following information:

device name	1-5 characters
status	R=ready, D=down
special file	entry in /dev associated with this device
backend name	entry in /usr/lib
options passed to backend	

   Backend options are described in backend(M).

To view a summary of the configuration file, use the command

```
/etc/dqueuer -n
```

After the configuration file has been changed, issue the command

```
/etc/dqueuer -r
```

This enables the dqueuer to recognize these changes.

Note that changing the status, priority or selection criteria in the configuration file does not effect the dqueuer. To change the dqueuer, use xq(M).

7.3. From Boot to Login - A Commentary

This section describes system operations from the initial boot to multi-user login.

Booting ZEUS is a two-step process. The first step is done by the ROM monitor (at the user's request). The primary bootstrapper loads the secondary bootstrapper from the disk or tape and then transfers control to it. It is the secondary bootstrapper that prints the boot prompt.

If the primary bootstrap is invalid, not much can be done. The symptom is that nothing happens after pressing START (NMI). The primary and secondary bootstrap are somewhat more verbose if something goes wrong during execution.

7.3.1. Kernel Initialization

Following a successful boot and various internal initializations, the kernel copies a small program from inside itself into process one. This small program is equivalent to the following C program:

```
main( ) {
exec ("/etc/INIT",0);
while (1);
}
```

/etc/INIT is a process that ultimately forks off processes that monitor user terminals for a login request. If the exec fails, the program loops, which appears as a continuous "on" condition of the user mode light. The only recourse is to try stand-alone restoration of a maintenance filesystem from tape, followed by a boot of the maintenance kernel.

7.3.2. /etc/INIT

After a manual boot, INIT starts by forking a process that runs /bin/csh with standard input, output, and error reporting to /dev/console. This state is called "single user mode." It allows checking and repair of filesystems and other operations that should be completed before users are on the system.

The system is switched to multiple-user mode by entering "INIT 2"; INIT executes the shell procedures found in /etc/rc and /etc/rc_csh.

The system is usually brought up with the automatic boot by pressing RESET and then START. This causes /etc/INIT to skip the single-user mode, and immediately begin to execute /etc/rc. Section 2.5 includes a discussion of the functions of /etc/rc_csh. This file may be locally modified to include any other functions desired.

If /etc/rc is not found, INIT ignores it and moves on.

Before running /etc/rc, INIT reads the file /etc/inittab. This file lists the actions to be taken on entry to each state (numbered one through nine), and the commands to be executed while in it. These actions and commands are organized according to group IDs, which by convention are associated with tty IDs. Each entry has the following format:

```
state:id:action(s):command
```

where:

state is an integer value between 1 and 9, with 1 conventionally associated with single-user mode and 2 with multiple-user mode.

id is an arbitrary unique identifier for this entry, conventionally associated with a tty terminal.

action(s) is identified by the letters "t", "k", or "c", which specify the disposition of any ongoing processes of the same type as the following: k=kill, t=terminate, c=reinvoke the process continuously. The kill (k) and terminate (t) actions can both be requested within the action field.

command is a string that can be given to a "sh" shell that is performed through an exec(1).

Typically entries for state 1 appear as follows:

```
1:00:k:/etc/GETTY tty0 !
```

Typically entries for state 2 appear as follows:

```
2:00:c:/etc/GETTY tty0 2
```

If an entry for an ID is not present for some state, nothing is done to any processes running under it, nor is any new login listener invoked. To temporarily disable a line that is being used for something other than user logins, set the action field to k, delete the command field, and re-enter your current state using the INIT command. For example, to kill the GETTY on communication line 3, change the entry for that line 3 to be:

```
2:03:k:
```

NOTE

There is no constraint requiring only login shells to be invoked on each line; states could be defined that run application programs on any or all terminals as the default.

If INIT fails in opening a configured terminal, the child process of INIT terminates and INIT forks another copy. This cycles through processes very quickly (hundreds per minute), and consumes a great deal of system resources. To recover, login as zeus, edit the /etc/inittab file and issue the command "INIT 2". Logging in under such circumstances takes a long time. Do not give up and start over. The same thing occurs if the terminal line is very "noisy."

7.3.3. /etc/GETTY

The GETTY program is started with the terminal type as one of its arguments. This argument to GETTY controls the initial values for terminal speed and various terminal parameters.

Currently GETTY supports nine different terminal types. Aside from minor variations in parameters, the distinguishing factor is the line speed. GETTY handles variable speed (typically dialup) lines by sensing the user transmitting "break," and trying another speed until the user does not send a "break." Thus, the user presses BREAK (or DELETE) until a recognizable input prompt is received.

Table 7-1 below summarizes GETTY terminal types:

Table 7-1. Terminal Type Codes

Type	Speed(s)	Uses
Ø	3ØØ,12ØØ,15Ø,11Ø	dialup
-	11Ø	old Mod 33/Mod35 teletype
1	15Ø	
2	96ØØ	glass tty`s (Zilog standard)
3	12ØØ,3ØØ	dialup
5	3ØØ,12ØØ	dialup
4	3ØØ	3ØØ baud terminals
6	24ØØ	lower-speed glass tty
7	192ØØ	high-speed glass tty
8	48ØØ	med-speed glass tty

GETTY prints the login message on configured terminals. GETTY then listens character-by-character for user input. It listens in raw mode to determine whether the terminal session is to be in uppercase or lowercase.

If GETTY receives all uppercase characters in the login name, it (naively) assumes that the user has an uppercase-only terminal. All user input and system output characters are translated and displayed as uppercase characters. If a user terminal gets into uppercase mode, the easiest way to recover is to type an invalid password; then answer the second user name request with CTRL-d. This starts a new GETTY.

7.3.4. Login

Login is executed by GETTY with the user name as an argument. If the name starts with "-", no message of the day is printed. After GETTY receives the user's name, it invokes /etc/login which is "linked" to /bin/login, which asks for the user's password, encrypts it, and compares it to the appropriate entry in the password file, /etc/passwd. If this file cannot be found, no one can login. To recover, reboot, come up in single-user mode, and try to salvage the password file or restore it from a backup tape. If the password is correct, login attempts to update the database for logged in users, and the login accounting database. Nothing happens if these two files (/usr/adm/wtmp and /usr/adm/utmp) do not exist.

Login then does a number of things. It attempts to change its working directory to the user's home directory. If this fails, "No directory" is printed and another name and password are solicited. Typically, this situation occurs when the user filesystem has not been mounted. Next, it changes the owner of the terminal and the process to the user logging in. It initializes the environment vector with the home directory, shell, and terminal type. The terminal type is used by **vi**, and comes from /etc/ttytype, which consists of lines in the form:

```
cc <device name>
```

For example:

```
vz console
```

means that /dev/console is a VTZ 2/10. In this file, the tty type is restricted to no more than seven characters. The file /etc/termcap should contain an occurence of the same terminal name followed by descriptive data.

If the device name is not found in this table, the type is set to "xx", which has no meaning for **vi**. After this, login optionally prints the contents of /etc/motd, the message of the day. It checks the file /usr/spool/mail/<name> to see if the user has mail. If the mailbox does not exist, no check is made. Finally, it executes the shell specified in the /etc/passwd entry with the argument "-". This causes the shell to take commands from $HOME/.profile (if it is /bin/sh) or from $HOME/.cshrc (if it is /bin/csh). /bin/csh then interprets $HOME/.login. If the shell invocation fails, the "No Shell" message is printed, and another name/password pair is solicited.

7.3.5. Logout

When the user logs out, /bin/csh executes $HOME/.logout and exits, (/bin/sh simply exits). Meanwhile, INIT has a wait(2) outstanding. If the wait is satisfied by one of INIT's immediate descendents (the processes initially running GETTY), and the inittab entry for this descendant in this state specifies continuous re-invocation of GETTY, INIT forks another GETTY and the cycle repeats indefinitely.

7.4. Adding Terminals

Use the following procedure to add new terminals.

The standard System 8000 accommodates up to eight terminals. These terminals connect to the connectors on the distribution

panel, which is on the rear panel of the CPU module. Each terminal connector is marked with the following terminal identifiers:

TTY Ø
CONSOLE
TTY 2

through

TTY 7

where CONSOLE is TTY 1. When more than eight terminals are used with the system, the additional terminals are connected to a secondary distribution panel that is similar to the primary panel. The connectors on the secondary panel are labeled TTY 8 through TTY 15. The secondary panel is located below the primary panel.

NOTE

The system console must be connected to the connector labeled CONSOLE.

When any terminals are added to the system, the software changes described in Section 7.4.2, below, are also required. Terminal devices (except the system console) are referred to as

tty<n>

where <n> is a numeral from Ø through 15, but excluding ttyl. The system console is identified as either "console," or as "ttyconsole." These designations correspond to the distribution panel identifiers.

The distribution panel connectors are female 25-pin D connectors that mate with standard male connectors. The connectors and other interface characteristics are compatible with EIA RS-232C.

Note that the operating system enforces limits on the number of users contained in the ZEUS software license agreement.

7.4.1. Preliminary Instructions

Installation of the additional hardware supplied with the Field Upgrade Kit is performed before changing the system software. Perform the instructions accompanying the Upgrade Kit, then the following steps.

1. Double check all communications cable connections.

2. Turn on system power, boot the system, and remain in single-user mode.

3. Login and remain logged in as the super-user (zeus) until the procedure in section 7.4.2 is completed.

NOTE

If installing terminals using an Intelligent Communications Processor (ICP) Board option, skip to Section 7.5 (the instructions supplied with the board include the correct Software Modification Procedure for this option).

7.4.2. Software Modification Procedure

1. Edit the file

   /etc/inittab

   by adding one entry for each new terminal to be added. Each entry should have the form (refer to Section 7.3.2 for a detailed description):

   <state>:<id>:<action(s)>:/etc/GETTY tty<n> <m> <o>

   where: <m> specifies the terminal type as shown in Table 7-1.

   <n> is the tty number.

   <o> is the number of seconds of delay before a dial-up port is disconnected if the user does not respond to the login request. Use only with dialup ports.

   Each line in this file describes a terminal; thus, there must be a line entry for each terminal attached to the system. This allows different types of terminals to be added to the system. Examples:

   a. Suppose an additional terminal is connected to the terminal three port and is to be initialized to run the normal ZEUS login (csh) shell in multiple user state

(state 2). The file /etc/inittab must have new or changed entries for tty3 that describe the terminal connected to the terminal three port.

The second argument to GETTY describes the terminal type (Table 7-1 lists terminal type codes). Thus, if terminal three is a high-speed CRT-type (glass) tty (19.2 K baud), then the type code would be "7", and the entire entry would be:

```
2:03:c:/etc/GETTY tty3 7
```

Another line is added to specify the disposition (killing) of processes initiated at this terminal when single user state (state 1) is entered:

```
1:03:k:/etc/GETTY tty3 !
```

b. To add entries for eight new terminals where the specific terminals are tty8 through tty15 and where the baud rate is 9600 insert the following lines into the /etc/inittab file.

```
1:08:k:/etc/GETTY tty8 !
2:08:c:/etc/GETTY tty8 2
1:09:k:/etc/GETTY tty9 !
2:09:c:/etc/GETTY tty9 2
1:10:k:/etc/GETTY tty10 !
2:10:c:/etc/GETTY tty10 2
1:11:k:/etc/GETTY tty11 !
2:11:c:/etc/GETTY tty11 2
1:12:k:/etc/GETTY tty12 !
2:12:c:/etc/GETTY tty12 2
1:13:k:/etc/GETTY tty13 !
2:13:c:/etc/GETTY tty13 2
1:14:k:/etc/GETTY tty14 !
2:14:c:/etc/GETTY tty14 2
1:15:k:/etc/GETTY tty15 !
2:15:c:/etc/GETTY tty15 2
```

2. Edit the file

```
/etc/ttytype
```

by adding a new entry for each new terminal. This file is like the one described in step 1 above. It contains one entry for each terminal used by the system. Each entry tells the screen editor what type of terminal is connected to each line. Each entry has the form

<typecode> tty<n>

where: <typecode> specifies the type of terminal equipment. The typecode is a maximum of seven characters that must be the same as found in the /etc/termcap file entry. For example, the VTZ 2/10 terminal is identified by "vz"; the Concept 100 terminal is identified by the characters "c1" (c-one). Note that the entries in /etc/termcap and /etc/termcap.others are followed by longer entries that can be more convenient for a reader.

<n> is the tty number. This number consists either of the word "console," or the connector number on the distribution panel. For example, if the connector designation is TTY 3, then <n> would be 3, and the file entry would be tty3.

Examples:

a. For a VTZ 2/10 terminal connected to connector TTY 5, the entry would be:

```
vz tty5
```

b. For an additional eight VTZ 2/10 terminals, the new entries would be:

```
vz tty8
vz tty9
vz tty10
vz tty11
vz tty12
vz tty13
vz tty14
vz tty15
```

3. Check the file

/etc/termcap

for an entry corresponding to the terminal(s) to be used with the system. When the visual editor (vi) is invoked by a user, it reads file /etc/termcap for a description of the terminal that is to be used for the editing session. The file describes the terminal features, so that the system responds properly to the various terminal control functions. Usually there is already an entry in the file that describes the terminal type. However, if there is not, then an appropriate entry must be added. For this purpose, there is a

secondary, or backup file:

/etc/termcap.others

This file contains the same entries that are in /etc/termcap, plus some others that may be needed. Thus, while /etc/termcap is read by **vi**, the file /etc/termcap.others is not; it is simply used for reference in case another addition must be made to /etc/termcap. In this case, the information in /etc/termcap.others can be repeated in etc/termcap. The file /etc/termcap is deliberately kept small so that it can be read quickly.

If the information for any specific terminal is not in either /etc/termcap or /etc/termcap.others, then check with the Zilog Systems Division Technical Support Department. Also refer to termcap(5).

4. Use the mknod(M) command once for each terminal tty<n> to be added in order to create a new device file for each new terminal:

   /etc/mknod /dev/tty<n> c 8 <n>

   where <n> in the tty number. See the command mknod(M).

   Examples:

   a. If a new terminal, tty5, is to be added, then the appropriate command would be:

      /etc/mknod /dev/tty5 c 8 5

   b. To add new terminals 8 through 15, enter the mknod command eight times, as shown below:

      ```
      /etc/mknod /dev/tty8 c 8 8
      /etc/mknod /dev/tty9 c 8 9
      /etc/mknod /dev/tty10 c 8 10
      /etc/mknod /dev/tty11 c 8 11
      /etc/mknod /dev/tty12 c 8 12
      /etc/mknod /dev/tty13 c 8 13
      /etc/mknod /dev/tty14 c 8 14
      /etc/mknod /dev/tty15 c 8 15
      ```

5. Connect the new terminal to line power and turn it ON.

6. The final step can be performed in two ways, as described in either step a or b following.

a. Execute the command

INIT 2

as described in init(M). When the INIT command is used, the login message is displayed on all new terminals when they are powered-up.

b. Re-boot the system. When the system is rebooted, and goes multi-user, all new terminals will display a login message.

7.5. Adding and Removing Users

The system has two shell scripts, adduser(M) and rmuser(M) (see below) that prompt the super-user for the information necessary to add or remove a user (respectively) from the system.

7.5.1. Adduser

Adduser looks for a new user ID (userid) by scanning the file /etc/passwd (see passwd(5)) for a uid that is one greater than the highest one in /etc/passwd. This file contains information about all users on the system; it is scanned upon login for the password, and it is used for mapping the userid number to the password, for mapping the userid number to the string that identifies the user, etc. When adduser finds a new userid, it prompts for a home directory for the new user; i.e. the directory at which the user will be logged in. An attempt is made to create the home directory. If this attempt fails, adduser asks for another home directory name. The next prompt is for the login shell (the program that receives input from the terminal upon login). Most installations use the C shell, /bin/csh, because it provides more features than the standard shell, /bin/sh.

Adduser then prompts for information regarding the group or groups to which the new user will belong. By convention, most users are a member of one group, called "other." The groups and the associated group-id are found in /etc/group (see group(5)). If a user is a member of more than one group, it is possible for that user to change groups (see newgrp(1) and setgid(2)). The default login group is the group in which a given user automatically becomes a member after login.

Finally, adduser prompts for the user's initial password. This password should probably be changed by the user after the initial login.

Optionally, the administrator or the new user may want to put a profile file in the new user's home directory. This file is called .cshrc if the login shell is the C shell (/bin/csh), and .profile if it is the standard shell (/bin/sh). A minimal .cshrc file might look like the following example:

```
#
set path = ( . /bin /usr/bin )
```

This file is actually a shell script that is evaluated by the shell when the user logs in. The first line (#) tells the shell that the file is to be interpreted by the C shell, and not the standard shell. (All shell scripts must contain this line if they are to be interpreted by the C shell.) The second line lists the directories that are searched for the commands entered by the user. In this example, the search starts in the user's current directory, followed by /bin, then /usr/bin. This is actually the default directory search, but it can be changed. For example, to check the user's private bin directory before it checks /bin, the path would be changed as shown:

```
set path = ( . ~/bin /bin /usr/bin )
```

The directory /bin contains the most frequently used commands; /usr/bin contains less frequently used commands, so it is usually searched after /bin. Setting up paths carefully ensures faster access to frequently used programs.

For more information on the C shell and .cshrc files, refer to the ZEUS Utilities Manual.

7.5.2. Rmuser

Rmuser is a shell script that, when executed, removes a user from the system. Rmuser removes the user from all groups (see group(5)) and allows the administrator to change the password for that user, making it impossible for that user to login. The entry for the user is not removed from /etc/passwd, so that any files that remain in the system owned by that user will be associated with the appropriate user name. Finally, rmuser scans the system for any files owned by the deleted user, and leaves the list in files.<user>. The system administrator can handle them as desired. kf

APPENDIX A
AN OVERVIEW OF ZEUS

A.1. General

This Appendix describes the ZEUS kernel. The discussion has four parts:

a. Process Control

b. The I/O System

c. The ZEUS file system, and

d. The ZEUS command interpreter (the "shell")

For an introduction to the ZEUS, refer to the ZEUS Reference Manual, Section Ø.

A.2. Process Control

Users execute programs in an environment called a user process, which is simply a program in execution that includes those components that make the program run. These include a memory image, general register values, and the status of open files. When a user process requires that a system function be performed, it calls the system as a subroutine (system call). At some point in this call, there is a distinct switch of environments. After this, the process is said to be running in system mode, because it actually executes code within the ZEUS kernel. Thus, in the normal definition of processes, the user and system processes are different phases of the same process (they never execute simultaneously). For protection, the system part of the process has its own stack, distinct from the user part of the process. (See Figure A-1.)

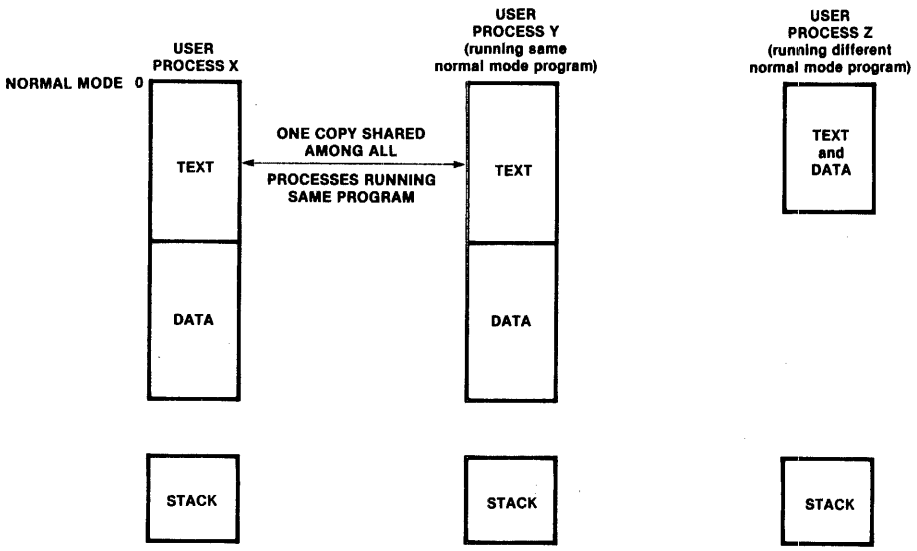

Figure A-1 User Process Control

A process has up to seven distinct segments:

1. a user mode text (program) segment containing only instructions and immediate operands,

2. a user mode data segment,

3. a user mode stack segment,

4. a system mode text segment that has the instructions of the operating system,

5. a system mode data segment containing operating system data global to all processes,

6. a system mode process local stack segment,

7. a system mode process local data segment (often called the "u vector" from the name of the structure defining the segment).

In this context, "segment" means a logical entity, not a memory segment. The user mode text and data segments may be combined into one segment. If they are separated, the text segment can be shared by all processes currently executing the same program. This read-only, sharable feature is specified at compile time, using the "-i" option of the linker.

All current read-only text segments in the system are maintained from the "text table." A text table entry holds the location of the text segment in secondary memory. If the segment is loaded in primary memory, that table also holds the primary memory location and the count of the number of processes sharing this entry. A text table entry is allocated when a process first executes a read-only text segment. At that time the segment is loaded into secondary memory. Subsequent processes executing the same text segment cause only the reference count in the text table entry to be incremented. When this count is reduced to zero, the entry is freed along with any primary and secondary memory holding the segment.

If the "-i" compile-time option is used, the user data segment is separate from the code, in a 64K byte address space. Otherwise, the data and code are in the same 64K byte address space. The data segment referred to also includes the user stack; the user's data segment does not hold system data. In particular, there are no system level I/O buffers in the user address space.

The user data segment has two growing boundaries. One, increased automatically by the system as a result of memory faults, is used for a stack. The second boundary is only grown (or shrunk) by explicit requests (see brk(2)). The contents of newly allocated primary memory is initialized with zeroes.

Also associated and swapped (i.e. moved temporarily to disk and back again) with a process is a small fixed-size system data segment. This segment contains all the data about the process that the system needs when the process is active. Examples of the kind of data contained in the system data segment are: saved central processor registers, open file descriptors, accounting information, scratch data area, and the stack for the system phase of the process. The system data segment is not addressable from the user process and is therefore protected.

Processes are listed in a process table with one entry per process. Each entry contains data needed by the system when the process is not active -- such as the process name, the location of the other segments, and scheduling information. The process table entry is allocated when the process is created, and freed when the process terminates. This process entry is always directly addressable by the kernel. It is, in a sense, the definition of all processes because all the data associated with a

process may be accessed starting from the process table entry. Figure A-2 shows the relationships between the various process control data.

A.2.1. Process Creation and Program Execution

Processes are created using the fork(2) system call. This system call causes a copy of the process invoking it to be created; that is, it causes a second image of the same process to be made, and execution begins on both images. The two processes have a hierarchical relationship, that of parent and child, even though they are identical in terms of their memory images, open files and the like.

If the fork(2) system call returns a value of -1, this indicates that the fork was unsuccessful. Anything greater than -1 indicates success; the value zero is returned to the child, and the process-id of the child is returned to the parent.

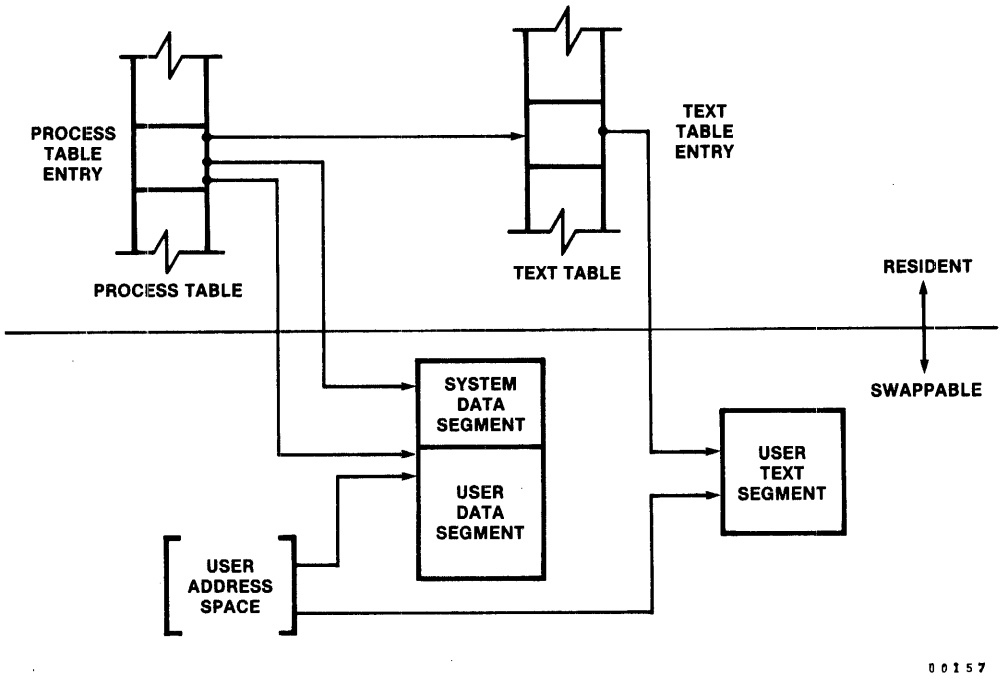

Figure A-2 Process Control Data Structure

The fork(2) system call is not very useful alone. However, when it is combined with the system call, exec(2), there is a full process creation/execution mechanism. "Exec" causes the process image to be replaced by that of another program, which is specified as a file name to exec(2). Although an explicit process

switch does not occur, this latter program is overlaid, and it becomes the process. Execution now proceeds from the beginning of the latter program. In other words, as soon as a process issues a successful system call exec(2), it is no longer executing the same code; its text and data segments are replaced by new ones, and the new ones are executed.

If a program, (such as the first pass of a compiler) has instructions to overlay itself with another program (such as the second pass), then it issues an exec(2) call for the second program. This is analogous to a "goto". If a program wishes to regain control after the "exec" of the second program, it "forks" a child process, has the child "exec" the second program, and then issues the system call "wait" to wait for the child to complete.

A.2.2. Swapping

The major data associated with a process (the user data segment, the system data segment, and the text segment) are swapped to and from secondary memory, as needed. The user data segment and the system data segment are kept in contiguous primary memory to reduce swapping latency. Allocation of both primary and secondary memory is performed by the same simple first-fit algorithm. When a process grows, a new piece of primary memory is allocated. The contents of the old memory are copied to the new memory. The old memory is freed and the tables are updated. If there is not enough primary memory, secondary memory is allocated instead. The process is swapped out onto the secondary memory, ready to be swapped in with its new size.

The scheduler, a separate process in the kernel, swaps other processes in and out of primary memory. It examines the process table looking for a process that is swapped out and is ready to run. It allocates primary memory for that process and reads its segments into primary memory, where that process competes for the central processor with other loaded processes. If no primary memory is available, the swapping process makes memory available by examining the process table for processes that can be swapped out. It selects a process to swap out, writes it to secondary memory, frees the primary memory, and then goes back to look for a process to swap in.

The scheduler uses two specific algorithms in making swapping decisions. One of these determines which process to swap in, as a function of secondary storage residence time. The one with the longest time out is swapped in first. There is a slight penalty for larger processes. The second algorithm determines which process to swap out. Processes that are not running or are waiting for slow events (i.e., waiting for teletype I/O or time-of-day events) are picked first, by age in primary memory (again with a

size penalty). The other processes are examined by the same age algorithm, but are not taken out unless they are at least of some age. This prevents total thrashing.

Swapping does not affect the execution of the resident processes. However, if the device used for swapping is used for file storage, the swapping traffic may affect the file system traffic, and slow down the system.

A.2.3. Synchronization and Scheduling

Processes are synchronized by having processes sleep while waiting for a particular event. Events are represented by addresses, usually of table entries associated with the events. For example, a process that is waiting for any of its children to terminate will wait for an event that is the address of its own process table entry. When a process terminates, it signals the event represented by its parent's process table entry. Signaling an event for which no process is waiting has no effect. Signaling an event for which many processes are waiting wakes them all up. Only one process gets the needed resource, and the others go back to sleep.

The event-wait code in the kernel is like a co-routine linkage. At any time, all but one of the processes has called event-wait. The remaining process is the one currently executing; when it calls event-wait, a process whose event has been signaled is selected to be run and the process running returns from its call to event-wait.

The process priority helps determine which process runs next. Each process has a priority; user process priority is determined by the recent ratio of compute time to real time consumed by the process. A process that has used a lot of compute time in the last real-time unit is assigned a lower user priority. Thus, because interactive processes are characterized by low ratios of compute to real time, interactive response is maintained without any special arrangements.

The compute-to-real-time ratio, which helps determine user process priority, is updated every second. Thus, in general, looping compute-bound user processes are scheduled round-robin with a 1-second quantum. A high-priority process waking up will preempt a running, low-priority process. Consequently, the scheduling algorithm has a very desirable negative feedback character. A process with a high priority cannot use an unfair share of the processor because its priority drops. At the same time, if a low-priority process is ignored for a long time, its priority rises.

A.3. I/O System

There are two types of I/O performed on the system: block I/O and character I/O, often called "raw" I/O. Block I/O always causes data to be read and written into the system's 512-byte internal buffers before being transferred to the final destination; character I/O, on the other hand, is unconcerned with blocks and involves the direct transfer of a specific number of bytes between memory and the device.

I/O devices are characterized by a major device number, a minor device number, and a class (block or character). For each class, there is a table consisting of one entry per device. Each entry is an array of entry points into the device drivers. The major device number is used to index into the table when calling the code for a particular device driver. The minor device number is passed to the device driver as an argument. The minor number has no significance other than that attributed to it by the driver. Usually, the driver uses the minor number to access one of several identical physical devices; e.g. separate disk drives on the same controller.

A.3.1. Block I/O System

The model block I/O device consists of randomly addressed, secondary memory blocks of 512 bytes each. The blocks are uniformly addressed Ø, 1, ... up to the size of the device. The block device driver has the job of emulating this model on a physical device.

The block I/O devices are accessed through a layer of buffering software. The system maintains a list of 52 buffers (this number can be changed by the user), each assigned a device name and a device address. This buffer pool constitutes a data cache for the block devices. On a read request, the cache is searched for the desired block. If the block is found, the data are made available to the requester without any physical I/O. If the block is not in the cache, the least recently used block in the cache is renamed, the correct device driver is called to fill up the renamed buffer, and then the data is made available. Write requests are handled similarly. The correct buffer is found and relabeled if necessary. The write is performed simply by marking the buffer as "dirty". The physical I/O is then deferred until the buffer is renamed. Note that if the system stops unexpectedly, it is possible that there will be logically complete, but physically incomplete, I/O in the buffers.

There is a system primitive (system call "sync") to flush all outstanding I/O activity from the buffers. Normally, this call is made every 3Ø seconds by a process (see update(M)) that is in-

voked when the system goes multi-user. However, this does not guarantee that the file systems will be consistent if the system goes down unexpectedly. In such a case some repair work may have to be done; this is done either automatically by fsck(M), or manually, as described in Section 6 of this manual.

A.3.2. Character I/O System

The character I/O system includes the "classical" character devices such as communications lines and line printers. It also includes tapes and disks when they are accessed directly (i.e. the buffer cache is bypassed); for example, for track-at-a-time disk copies. User I/O requests are sent (essentially unaltered) to the device driver for implementation.

Zilog follows the convention that raw I/O is specified by fixing the letter "r" to the file name associated with the device. Thus if a block device is referred to as "/dev/ctØ", the raw device will be referred to as "/dev/rctØ". Using the raw device is usually faster than using the block device. The user is cautioned, however, to make sure that the raw disk device is used only when the corresponding file system is unmounted.

A.4. The File System

In the ZEUS system, a file is a (one-dimensional) array of bytes. No other structure of files is implied by the system. Files are attached anywhere (and possibly multiply) onto a hierarchy of directories. Directories are simply files that users cannot write.

A.4.1. File System Layout

The canonical view of a "disk" is a randomly addressable array of 512-byte blocks. This "array" is usually divided up into several contiguous variable-sized partitions. Most of these partitions are occupied by "file systems" although ZEUS requires that one partition (and only one) be reserved for swapping. A file system on disk (accessed either through the block or character I/O system) has four distinct self-identifying regions:

1. The first consists of one block (logical address Ø) that is used only for booting.

2. The second consists of one block (logical address 1) called a "super-block". The super-block contains, among other things, the size of the file system, the boundaries of the other regions, and the list of pointers to free inodes.

3. The third is the list of inodes. An inode is a 64-byte structure that defines the characteristics of a file (location, type, access permission, owner, etc.). The offset of a particular inode within the i-list is its i-number. The combination of device name (major and minor numbers) and i-number identifies a particular file to ZEUS.

4. The fourth section, from the i-list to the end of the disk space allocated to the file system, includes free storage blocks; these are available for file storage. (See Figure A-3.)

The free space on a disk is maintained by a linked list of available disk blocks. Every block in this chain contains the address of the next block in the chain. The remaining space in the block contains the address of up to 50 disk blocks that are also free. Thus, with one I/O operation, the system obtains 50 free blocks and a pointer to more. Since all blocks have a fixed size, and there is strict accounting of space, there is no need to compact or "garbage collect." However, as disk space becomes dispersed, latency gradually increases. Some installations may choose to occasionally compact disk space to reduce latency.

A.4.2. Directory Files

A logical directory hierarchy is added to the flat physical file system structure simply by adding a new type of file, the directory. A directory contains 16-byte entries consisting of a 14-byte name and an index, or i-number. The i-number is the index to an inode. The root of the hierarchy is at a known i-number, 2; that is, it is the second inode in the i-list. The file system structure allows an arbitrary, directed graph of directories with regular files linked at arbitrary places in this graph. Directories are accessed the same way as ordinary files.

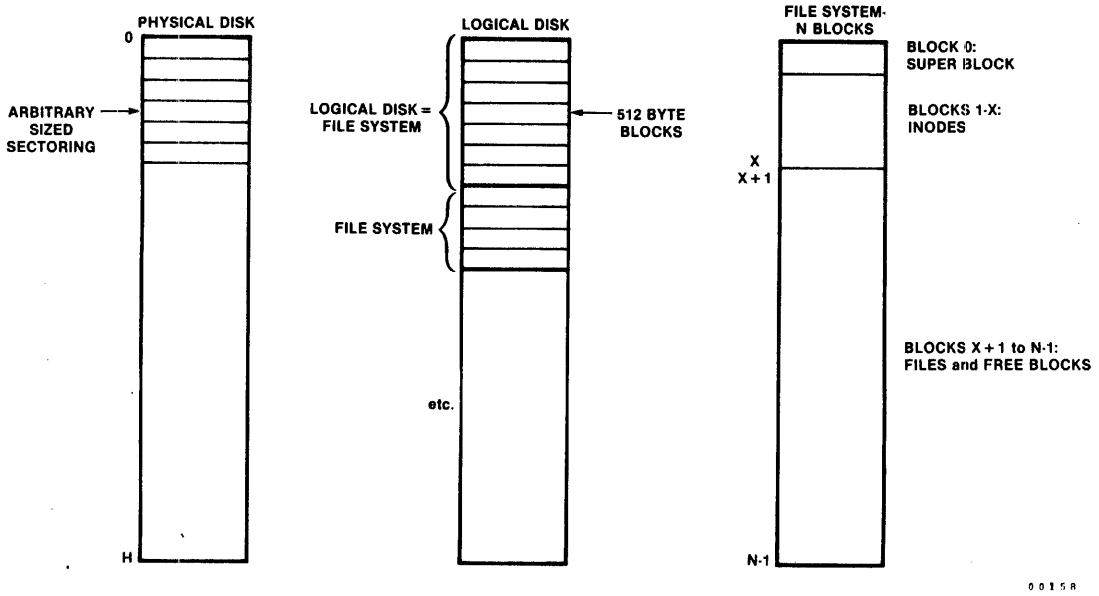

Figure A-3 File System Layout

A.4.3. Inodes

When a file is accessed, the associated i-number is used to index into the i-list region of the file system for that directory. The entry found thereby is the inode, which contains, among other things, the description of the file:

1. The user and group-ID of its owner.
2. Its protection bits.
3. The physical disk addresses for the file contents.
4. Its size.
5. Time of creation, last use, and last modification.
6. The number of links to the file, that is, the number of times it appears in a directory.

7. A code indicating whether the file is a directory, an ordinary file, or a special file.

The system maintains a table of all active inodes. As a new file is accessed, the system locates the corresponding inode, allocates an inode table entry, and reads the inode into the table in primary memory. This is the current inode; changes to the inode are made to the table entry. When the last access to the inode finishes, the table entry is copied back to the secondary store i-list and the table entry is freed.

All I/O file operations are carried out with the aid of the corresponding inode table entry. Inodes and i-numbers are transparent to the user. References to the file system are made in terms of path names of the directory tree. Converting a path name into an inode table entry begins with a known inode (the root or the current directory of some process). The next component of the path name is searched by reading the directory. This gives an i-number and an implied device (corresponding to the file system that the device is in). Thus the next inode table entry can be accessed. If that was the last component of the path name, then this inode is the result. If not, this inode is the directory needed to look up the next component of the path name, and the algorithm is repeated.

An inode normally contains 13 disk addresses. The first ten of these addresses point directly at the first ten blocks of a file. If a file is larger than ten blocks, then the eleventh address points at a block that contains the addresses of the next 128 blocks of the file. If the file is still larger than this, then the twelfth block points at up to 128 blocks, each pointing to 128 blocks of the file. Files yet larger use the thirteenth address for a "triple indirect" address. Thus files can grow to [(10 + 128 + 128 SQUARED + 128 CUBED) TIMES 512] bytes.

If an inode indicates that the file is special, the last 12 device address words are immaterial, and the first specifies an internal "device name"; that is, a major and a minor device number.

The user process accesses the file system with certain primitives. The most common of these are system calls open(2), creat(2), read(2), write(2), lseek(2), and close(2). The data structures maintained are shown in Figure A-2. In the system data segment associated with a user, there is room for a list of 20 open files. The list consists of pointers that can be used to access corresponding inode table entries. Associated with each of these open files is a current I/O pointer. This is a byte offset of the next read/write operation on the file. The system treats each read/write request as random with an implied seek to the I/O pointer. The user may, of course, perform random I/O by

setting the I/O pointer before read or write operations using the "lseek" system call.

With file sharing, it is necessary for related processes to share a common I/O pointer, and yet have separate I/O pointers for independent processes that access the same file. With these two conditions, the I/O pointer cannot reside in the inode table nor can it reside in the list of open files for the process. A new table, the open file table, holds the I/O pointer. Processes that share the same open file (the result of a "fork") share a common open file table entry. A separate open of the same file shares only the inode table entry, and has distinct open file table entries.

The main file system primitives are implemented as follows.

1. open(2) converts a file system path name given by the user into an inode table entry. A pointer to the inode table entry is placed in a newly created open file table entry. A pointer to the file table entry is placed in the system data segment for the process.

2. creat(2) first creates a new inode entry, writes the i-number into a directory, and then builds the same structure as for an "open."

3. read(2) and write(2) just access the inode entry as described above.

4. lseek(2) simply manipulates the I/O pointer. No physical seeking is done.

5. close(2) just frees the structures built by open(2) and creat(2)

Reference counts are kept on the open file table entries and the inode table entries to free these structures after the last reference goes away. Making a link to an existing file involves creating a directory entry with the new name, copying the i-number from the original file entry, and incrementing the link-count field of the inode. Removing (unlinking) a file is done by decrementing the reference count for the given inode. When the last reference to an inode is removed, then the file is deleted and the inode entry is freed. Note that if a file is removed while it is still open, the reference count will not be zero until the file is closed, since there is a count for the directory reference to the file, as well as a count for the "open" on the file.

A.4.4. Mounted File Systems

A file system is associated with some designated block device, which is formatted to contain a hierarchy, as described above. One file system always contains the root directory, and is usually called the root file system. Another file system may be mounted at any node (usually a leaf) of the current hierarchy. This logically extends the current hierarchy.

Mounting is achieved by issuing a "mount" system call, which takes three parameters: the name of a special file, the name of an ordinary file (usually a directory), and a flag that indicates whether the mounted file system is to have read-only status. After a "mount" has been issued, references to files with a pathname that includes the ordinary file name cause the system to look for that file on the mounted device. Any files previously associated with the sub-tree defined by the ordinary file are invisible.

Allocation of space for a file is taken from the free pool of a file system on disk. The hierarchy that the user usually sees consists of many mounted devices, and does not have a common pool of free secondary storage space. This separation of space on different devices is necessary to allow easy unmounting of a device.

A.5. The Shell

The shell, or system command interpreter, is the program with which most users interact after logging into the system. Most of the time, the shell is waiting for the user to type a command. When the newline character ending the line is typed, the shell's read(2) call returns. The shell analyzes the command line, putting the arguments in a form appropriate for exec(2). Then fork(2) is called. The child process, whose code is still that of the shell, attempts to perform an "exec" with the appropriate arguments. If successful, this will bring in and start execution of the program whose name was given. Meanwhile, the other process resulting from the fork(2), which is the parent process, "waits" for the child process to die. When this happens, the shell knows the command is finished, so it types its prompt and reads the keyboard to obtain another command.

Given this framework, the implementation of background processes is trivial; whenever a command line contains "&", the shell merely refrains from waiting for the process that it created to execute the command.

This mechanism meshes well with the notion of standard input and output files. When a process is created by the fork(2) primitive, it inherits not only the memory image of its parent but

also all the files currently open in its parent, including those with file descriptors Ø, 1, and 2. The shell, of course, uses these files to read command lines and to write its prompts and diagnostics. In the ordinary case, its children, the command programs, inherit them automatically. When an argument with "<" or ">" is given, however, the offspring process, just before it performs an "exec" of the program or command, makes the standard I/O file descriptor (Ø or 1, respectively) refer to the named file. This is easy because, by agreement, the smallest unused file descriptor is assigned when a new file is "opened" (or "created"); it is only necessary to close file Ø (or 1) and open the named file. Because the process in which the command program runs simply terminates when it is through, the association between a file specified after "<" or ">" and file descriptor Ø or 1 is ended automatically when the process dies. Therefore, the shell does not have the actual names of the files that are its own standard input and output, because it never reopens them.

In ordinary circumstances, the main loop of the shell never terminates. The main loop includes the branch of the return from fork(2) belonging to the parent process; that is, the branch that does a "wait", then reads another command line. The one thing that causes the shell to terminate is discovering an end-of-file condition on its input file. Thus, when the shell is executed as a command with a given input file, as in:

```
sh <comfile
```

the commands in comfile are executed until the end of comfile is reached; then the instance of the "shell" invoked by sh terminates. Because this shell process is the child of another instance of the shell, the wait(2) executed in the latter will return, and another command may then be processed.

Most systems use the C shell (see csh(1)), rather than the shell described in sh(1). The C shell, although bigger, has more features and thus, is easier to use.

A.6. Other Programs as Shell

The shell as described above is designed to allow users full access to the facilities of the system, because it can invoke the execution of any program with an appropriate protection mode. Sometimes, however, a different interface to the system is desirable, and this feature is easily arranged.

Ordinarily, after a user has successfully logged in, INIT(M) invokes the shell. However, the user entry in the /etc/passwd file may contain the name of another program to be invoked after login. This program is free to interpret user input in other ways.

For example, the /etc/passwd file entries for users of a secretarial editing system might specify that an editor is to be used instead of the shell. Thus, when users of the editing system login, they are inside the editor and can begin work immediately. They can also be prevented from invoking programs not intended for their use.

APPENDIX B
REDISTRIBUTING FILES WITHIN THE CURRENT DISK CONFIGURATION

B.1. General

This appendix contains instructions for relocating files within the current disk configuration. This procedure is easier and safer than reconfiguring the disk layout itself, which is described in Section 4.2.3. Directions can vary from system to system, depending upon disk capacity and additional software purchased.

B.2. File Relocation Requirements

File relocation can be necessary:

1. Before adding new files.
2. After running out of space on a device.
3. When is it desired to speed up disk access times.

B.2.1. Adding New Files

Before adding large files or large numbers of new files to a filesystem, use df(M) to check the number of free blocks on the target filesystem. If adding new files will decrease the number of free blocks below ten percent of the total blocks, reorganization is recommended. To avoid running out of space on that device (a potentially dangerous problem), perform this check.

B.2.2. Running Out of Space On a Device

The / and /usr filesystems should not run out of space unless the number of available blocks have not been checked before transferring large numbers of files. However, /usr may run out of space if many files are spooled to the printer. If this condition occurs frequently, it may be desirable to create more free space by relocating files as the following sections describe.

B.2.3. Optimizing Disk Access Times

As delivered, the ZEUS system is set up so that the most frequently used programs reside on /, and less frequently used programs on /usr and /z. Under the default disk configuration, / is physically in the middle of the disk, and is thus faster to access; whereas /usr and /z are somewhat slower to access. Normally, path variables in .login or .cshrc files are set up so that the current directory is searched first; then /bin, which resides on /; and finally /usr/bin, which resides on /usr. If languages are placed under /z/bin, the path variables in the users' .login or .cshrc files should be changed to reflect this.

To prevent software failure, certain files must not be moved. These include files in /lib and /usr/lib. Sh, csh, and commands needed in single user mode must not be moved out of /bin.

B.3. Removing Unwanted Files

Unused files can be removed to make space for more useful files. They can be moved to a less frequently accessed filesystem using mv(1), or they can be removed with rm(1). It is advisable to tar(1) the unwanted files to a tape before removing them, in case they are needed later. To avoid running out of space on the target filesystem, check the number of free blocks on that device with df(M) before moving a large number of files. The following are possible candidates for removal, because they are seldom needed:

1. /usr/lib/asz2 and /usr/lib/asz2d, which are MCZ-compatible versions of the assembler.

2. /etc/termcap.others, which contains terminal descriptions for nonstandard terminals.

For more information, refer to as(1), and termcap(5).

B.4. Distribution Tapes

For ease of custom-tailoring systems and because disks come in different sizes, the ZEUS system now comes on multiple tapes. These are:

1. The "boot" tape, including the operating system and utilities (dumps(M) of / and /usr specific to Model 11, 21 and 31).

2. Separate tapes containing tar(1) images of the languages Pascal, Cobol, Z8 Assembler, Z8Ø Assembler, Z8Ø C and Fortran.

Organizing files within filesystems is simplified if the languages are on separate tapes to allow selective restoration of languages. For example, it is possible for a 40 MB disk to have a large /usr filesystem, which could accommodate all the manual pages and the extra languages. On the other hand, if the system had a smaller capacity disk, and everything were on one tape (under /usr/doc and /usr/bin, as configured on a 40 MB disk), the /usr filesystem might run out of space upon restoring(M) the utilities. The number of languages on the system and the size of the system disk determine the number of languages which can reside in /usr/bin, and whether the online ZEUS Reference Manual pages may be included in /usr/man.

Additional languages will be placed in /z/bin. If languages reside in /z/bin, the .login files in the users' home directories must be modified to include this pathname. For example,

```
set path = (. /bin /usr/bin /z/bin)
```

Remember to logout, and then login again to activate the new .login file.

B.5. Suggested File Distribution

Since all the extra software and documentation probably will not fit on one filesystem, files should be distributed as follows:

1. Languages from the additional language tapes in /z/bin.

2. Manual page entries in /z/man or /z/usr/man if /usr is low on space.

Jun 26 15:49 1984 badBoot Page 1

```
8000 Monitor 3.0 - Press START to Load System

POWER UP DIAGNOSTICS

ACTIVE PERIPHERALS:
MDC
TCC
ECC
5000
FPP

COMPLETE
SEGMENTED JUMPERS
MAXSEG=<UP>
> boot
Segmented Jumper Configuration

boot
: md(0,15200)zeus
(65024/0xFE00)+(13312/0x3400)+(18688/0x4900)+(55862/0xDA36)
|md(0,15200)fpo.u.code
(12199/0x2FA7)+(83/0x53)
----------------------------------------------------------------------------
Zilog Zeus Kernel -- Release 3.2 -- Generated 06/20/84 00:08:16
Copyright  1981  Zilog, Inc.

System:SYS 8000       Node:ZEUS        Release:REL:3.2        Version:MOD 21P

number of users            = 16
fpp                        = 1
size of user struct        = (1394/0x572) bytes
address of user struct     = 0x3E00F600
kernel memory size         = (155648/0x26000) bytes
user memory size           = (892928/0xDA000) bytes
type of memory             = ecc
----------------------------------------------------------------------------

Nonmaskable Interrupt (manual)   ****  INTERRUPTED IN SYSTEM MODE   ****

STATE 3E00FCA4 eventid=FFF1, fcw=E020, pcseg=8000, pcoff=5808
r0:0 r1:0 r2:F000 r3:A878 r4:308 r5:C986 r6:3E00 r7:FCE6
r8:0 r9:558C r10:39 r11:E020 r12:3E00 r13:FCE6 r14:0 r15:FECA
System sp= 3E00FC8A
Last vectored interrupt serviced: real-time clock.
Pending I/O interrupts=  md.
Scheduler-    st: 1 fl: 3 wchan: C8F2
Command: INIT.  Non-segmented program.   Real user id= 0
Proc ptr: 300A874  Break addr: 1  tsize: 0  dsize: 40.
        Code segment-base:600 size:50 attr:0
        Data segment-base:50A size:1 attr:14
        Stack segment:base:308 size:1 attr:14
```

Floating Point Processor Microcode V1.0
Installation Instructions

This software package is distributed on a tape cartridge (Zilog P/N 14-0106-00) in tar format.

Before attempting to install this package please check to see if you already have the correct microcode on your system. To do this simply determine the checksum as follows:

```
sum /fpp.u.code
```

The expected value is "49209 48". Any other checksum value returned indicates an older, preliminary version and the microcode should be replaced as follows:

Change directories to "/z" with the command:

```
cd /z
```

Extract the tape files into the disk directory with the following command:

```
tar xv
```

Because the v (verbose) option is used, the names and sizes of the files are displayed as they are extracted. Now the current directory contains the following two files:

```
READ_ME                    fpp.u.code
```

If you wish to save the old version of fpp.u.code, you can move the old copy to fpp.u.code.o with the following command:

```
mv /fpp.u.code fpp.u.code.o
```

The new version of fpp.u.code must now be moved to the root directory ("/") by typing the following commands:

```
cd /
mv /z/fpp.u.code .
```

You may remove the READ_ME file from "/z" by typing the following command:

```
rm /z/READ_ME
```

The new version of fpp.u.code is now installed on disc. You now must reboot the system in order to download and initialize the FPP H/W option for subsequent use.

Remember to perform this procedure again whenever you restore the ZEUS 3.2 Operating System from the standard release tape supplied by Zilog. The microcode for the floating point processor H/W option supplied on the ZEUS 3.2 release tape was preliminary in nature and is known to have bugs.

3-1D: 1

Zilog, Inc. 1315 Dell Ave., Campbell, California 95008 Telephone (408)370-8000 TWX 910-338-7621

Printed in USA