Silicon Graphics Inc

System Administration

STUDENT WORKBOOK

Publication Number: 0000 R1 A

July 8, 1988

REVISION NOTICE
This is the First revision of the Student's Workbook

READER COMMENT FORM
A User's Comment Form or Class Survey is provided at the end of this publication. Comments and suggestions may be sent to: Silicon Graphics, Inc., Training Dept, 2011 Stierlin Rd, Mountain View, CA, 94043.

RESTRICTION ON USE

NOTICES

the Role of the System Administrator

Objectives:

- define system administrator
- identify administrative tasks
- categorize tasks

Support Users

- add new users
- remove users
- assist users
- set up and administer user programs

Support Devices

- set up terminals
- troubleshoot terminals
- set up printers
- troubleshoot printers
- set up other peripherals

Manage Disk Resources

- back up system
- add new disks
- reconfigure (partition) disks
- restore system
- update system
- recover system

Maintain Security

- define default file permissions
- manage directory permissions
- manage passwords
- eliminate security weaknesses

Maintain System

- add new software
- monitor resource usage
- plan and schedule new hardware
- keep system logbook

Administer Network

- set up communications
- set up network security
- configure networking products
- troubleshoot the network

System Documentation

Manual Sets

When your IRIS workstation arrives at your facility, a number of manuals arrive with it. These manuals include:

Standard Set

077-5001-001	Operating System Kit
077-5002-001	Owners Kit
077-5003-001	C Programming Kit
077-5004-001	Graphics Library Kit
007-7102-010	IRIS-4D Documentation Guide

Operating System Kit

007-0601-010	IRIS-4D Series Programmer's Guide, Volumes I and II *Basic intro to 4D programming/awk/lex*
007-0602-010	IRIS-4D Series Programmer's Reference Volumes I, II, and III *Terminfo/make/sccs/interprocess comm*
007-0603-010	IRIS-4D Series System Administrator's Guide *Basic tutorials on System Admin topics*
007-0604-010	IRIS-4D Series System Administrator's Reference Manual *The 1M man pages*
007-0605-010	IRIS-4D Series User's Guide *Editors, Shells, and terminal setup*
007-0606-010	IRIS-4D Series User's Reference Volumes I and II *Man(1) pages and section 6 - games*

Owner's Manuals Kit

007-7101-010	Getting Started with the IRIS-4D Series Workstation
007-5320-010	IRIS-4D Series Owner's Guide *How-to guide*
007-3301-010	4D Workstation Release Notes

C Programming manuals Kit

007-0903-010	Learning to Debug with edge, C Edition
007-0701-010	C Language Reference Manual
007-0901-010	Porting C Code to IRIS-4D Series Workstations
007-0730-010	Assembly Language Programmer's Guide
007-0905-010	IRIS-4D Series Compiler Guide

Graphics Library Kit

007-1201-010	Graphics Library User's Guide, Volumes I and II

FORTRAN Programming Kit

007-0904-010	Learning to Debug with edge, FORTRAN Edition
007-0902-010	Porting FORTRAN Code to IRIS-4D Series Workstations
007-0710-010	FORTRAN Language Reference Manual
007-3305-010	4D FORTRAN Release Notes

Documentor's Workbench Kit

007-0510-010	Documentor's Workbench Software User's Guide
007-0511-010	DWB Software Technical Papers and Reference Guide
007-3303-010	4D DWB Release Notes

NFS Kit

007-0850-010	NFS User's Guide
007-3302-010	4D NFS Release Notes

Laser Printer Kit

007-6203-010	Using the Laser Printer
007-3304-010	4D Laser Printer Release Notes

Input Devices

007-6205-010	Installing the Dial and Button Box
007-6206-010	Installing the Digitizer Tablet

Output Devices

077-5701-001	Laser Printer Kit
007-6204-010	Using the Color Printer

Series Communications

007-0550-010	Using the 6-Port RS-232 Option
007-0850-010	NFS User's Guide
007-0830-010	TCP/IP User's Guide

FORTRAN

77-5005-001	FORTRAN Programming Kit
007-0730-010	Assembly Language Programmer's Guide

On-line Resources

The IRIS contains a number of on-line resources for handy reference. These are in the form of on-line "man" pages and some interesting software tools and tutorials.

Man pages are located in the */usr/catman* directory and are broken into four main sections:

u_man, the user's man pages, section 1, and the demos, section 6

g_man, the Graphics Library man pages, section 3g

a_man, the System Administration manual pages, and section 7

p_man, the User's man pages, sections 2, 3, 4, and 5

The actual pages themselves are individual files in a compressed (*pack(1)*) form. Use *man*(1) to read them as usual.
To directly read them, you can use *pcat*(1) to unpack and print them to the screen. Be sure to use *^s* and *^q* to control the output unless you have 300msec eyes!

whereis

whereis(1) is a useful command for locating specific system programs and commands. For example:

```
whereis passwd
```

IRIS-4D User's Guide

The Administrator will usually find a goodly slice of his/her time dedicated to helping new users survive UNIX. Giving them a copy of this manual may save you significant amounts of time.

Important sections in this manual include:

- *Basics for UNIX* - how to login/out and deal with some problems.
- *Using the UNIX File System* - How to use basic commands for getting around, directory and file manipulations.
- *Ed and VI Tutorials* - Quite useful tutorials/references for the editors. The optional editor *Emacs* is covered in it's own manual.
- *Sh and Csh tutorials* - For those wishing to take advantage of tools like *batch, at, ps/kill, etc., and to write simple shell scripts.*
- *Summaries* - a series of reference pages and summaries of the foregoing sections.

IRIS-4D Programmer's Guide

Programmers will find much useful information about the UNIX programming environment in the two volumes that make up this guide.

Contents:

- Programming in the UNIX environment
- awk, lex, yacc
- File, Record Locking
- Interprocess Communication
- curses/terminfo
- make
- source code control system
- lint

Programmer's Reference Manual

The *IRIS-4D Programmer's Reference Manual* Contains sections 2, 3, 4, and 5 of the traditional AT&T UNIX manuals organized into three volumes:

Vol I Contains Section 2 - Standard System Calls. Here you will find the traditional system call man pages.

Vol II Contains Section 3 - Functions and Libraries. Program callable functions, subroutines, and libraries.

Vol III Contains Sections 4 & 5.

Section 4 - Special Files and miscellaneous. From an administrator's view, section 4 is the most important. It contains detailed descriptions of many of the configuration files. Execute *ls* on the */usr/catman/p_man/cat4* directory for a full list.

Section 5 contains special information about files and programs that do not easily fall into the other man page sections. For example, some *troff* macros and the ASCII table.

Manuals Related to Administration

Of the supplied manuals, the following provide information explicitly related to daily IRIS system maintenance. These are:

007-0603-010 IRIS-4D Series System Administrator's Guide
007-0604-010 IRIS-4D Series System Administrator's Reference Manual
007-0602-010 IRIS-4D Series Programmer's Reference Volume III
007-5320-010 IRIS-4D Series Owner's Guide
007-0830-010 TCP/IP User's Guide
007-0850-010 NFS User's Guide
007-3301-010 4D Workstation Release Notes
007-3302-010 4D NFS Release Notes

System Administrator's Guide

The *IRIS-4D System Administrator's Guide* is provided as a supplement to the Owner's Manual. It describes security and administrative tasks and suggests shell scripts to automate these procedures. Information is organized by major subject areas including the processor, the file system, and system procedures for UNIX Operating System Administration.

Contents:

- System Security
- User Services
- Processor Options
- Disk/Tape Management
- File System Administration
- Line Printer Use and Administration
- Basic Networking
- Directories and Files

System Administrator's Reference Manual

The *IRIS-4D System Administrator's Reference Manual* Contains section 1M manual pages of the AT&T manuals for the UNIX System V Operating System. These manual pages describe the commands directly related to system administration. Also included is section 7, which describes special files specific to hardware peripherals and system device drives.

IRIS-4D Series Owner's Guide

Traditional UNIX documentation was designed to be a set of references for the familiar and expert user. It was not designed to be a tutorial or "how to" reference though some tutorials were found in the old Volumes IIA and IIB. Unfortunately, the tutorials assume you have a good grasp of standard UNIX jargon. Consequently, novice UNIX users and administrators often find themselves spending inordinate amounts of time trying to piece together even the simplest of tasks using that material.

The Owner's guide is a cookbook designed to provide the IRIS administrator quick and simple procedures to deal with common administrative functions. Following is a short list of administrative functions covered by the Owner's Guide.

Contents:

- Booting the IRIS
- Installing the Workstation
- Setting up Your System
- General Administration
- Prom Monitor
- Introduction to *fx*
- Attaching Terminal, Modems, and printers
- System Crash Recovery
- Software Options - Tools and Installation

Additional References

A number of operations or tutorials are available but fall outside the scope of the regular manual set. This information is contained in special booklets shipped with the system, or with special options. These manuals include:

Peripherals

Using the Color Printer
Instruction on installation and use of the color printers supported by Silicon Graphics with the UNIX LP Spooler. Printing screen images, printer over the network, troubleshooting are covered. Contains applicable AT&T manual pages.

Using the Laser Printer
Instruction on installation and use of the Apple LaserWriter with the UNIX LP Spooler. Printing screen images, formatting text with AT&T Documenter's Workbench software and PostScript, printing over the network, and troubleshooting are covered. Contains applicable UNIX Operating System manual pages.

Laser Printer Release Notes
This document describes the software installation procedure, features, and known bugs of the Laser Printer option.

DWB Release Notes
This document describes the software installation procedure, features, and known bugs of the DWB option.

Using the 6-Port RS-232 Option
This document is designed for customers who have received the 6-Port RS-232 option either with their original system, or as an add-on. It shows them how to configure the ports.

Installing the Dial and Button Box
This document accompanies the Buttons Box and Dials peripheral option for the IRIS-4D Series workstation. It describes the installation and operation of the dial and button box hardware and software.

Installing the Digitizer Tablet
This document accompanies the both Digitizer Tablet peripheral options for the IRIS-4D Series workstation. It describes the installation and operation of the digitizer tablet.

Communications

TCP/IP User's Guide
This document is designed for customers who will be the TCP/IP system administrator on an IRIS-4D Series workstation. It is a reference guide that describes how to install and configure TCP/IP, and how to use various network utilities.

NFS User's Guide
This document is designed for customers who will be the NFS system administrator on an IRIS-4D Series workstation. It is a reference guide that covers these topics: understanding networking models, becoming an NFS server, mounting remote file systems, debugging the network, improving network security, and installing and administering the Yellow Pages.

NFS Release Notes
This document describes the software installation procedure, features, and known bugs of the NFS option.

Programming Languages

IRIS-4D Series Compiler Guide
Description of cc compiler and ld linker, including an explanation of all switches and options, as well as operating procedures.

C Language Reference Manual
Is a manual describing the general syntax of the C programming language, data structures, program flow control, etc.

Porting C Code to IRIS-4D Series Workstations
Contains lists of supported and unsupported cc and ld switches; addresses code compatibility and known cc, ld, and dbx bugs.

Assembly Language Programmer's Guide
Description of Mips processor architecture and instruction set. Explanation of lexical and linkage conventions. Instruction on writing assembly language programs and using the assembler.

Porting FORTRAN Code to IRIS-4D Series Workstations
Contains lists of supported and unsupported Series 2000/3000 f77 and ld switches; addresses code compatibility, wrappers, and known f77, ld and dbx bugs.

FORTRAN Language Reference Manual
Is a manual describing the general syntax of the FORTRAN programming language; data structures, program flow control, etc.

FORTRAN Release Notes
This document describes the software installation procedure, features, and known bugs of the FORTRAN option.

General

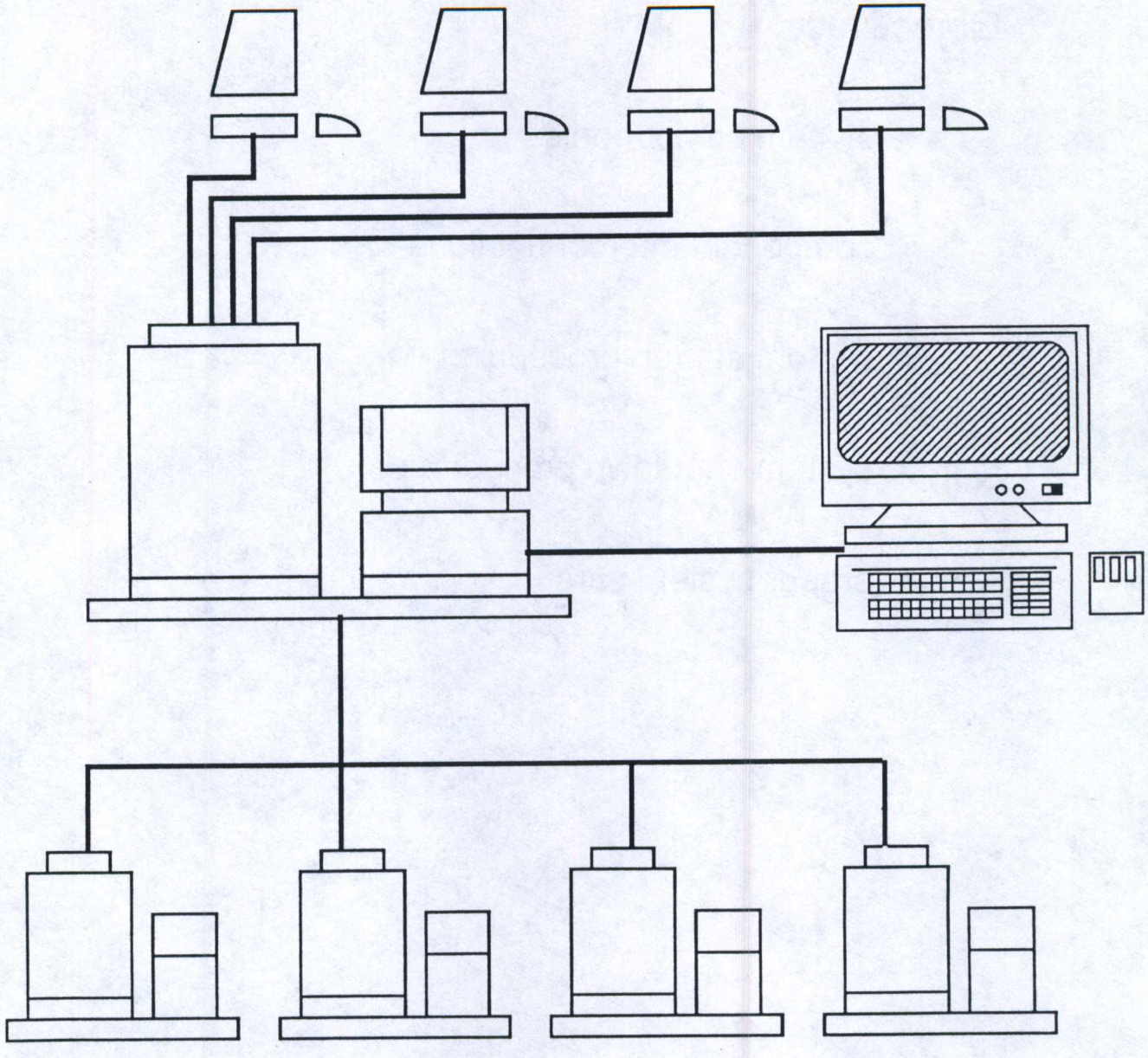
Workstation Release Notes
This document describes the software installation procedure, features, and known bugs of the standard system.

Hardware Familiarization

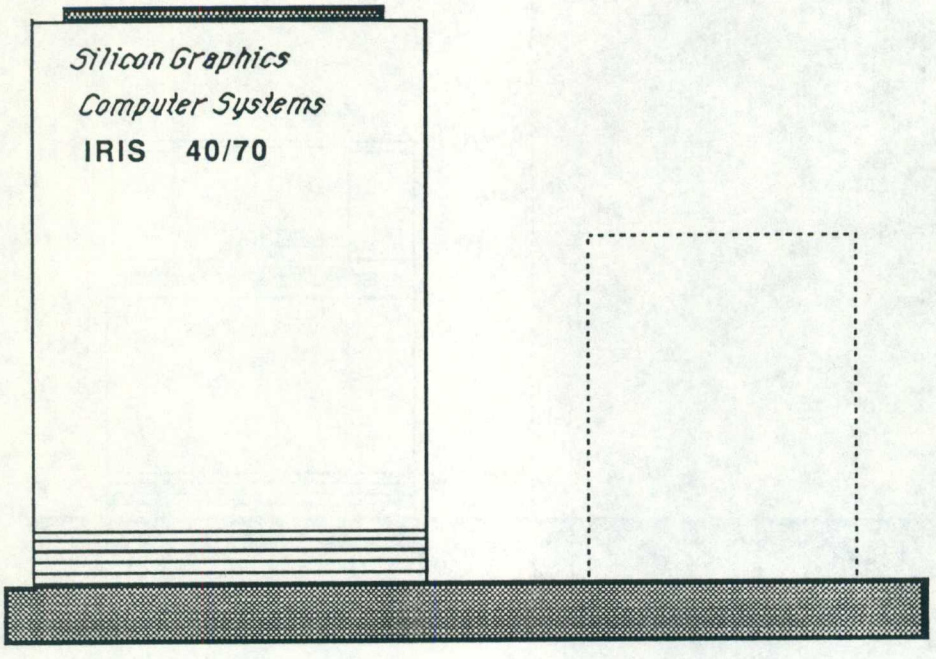
Objectives:

- system components
- component interconnections
- system startup procedures
- system shutdown procedures
- proper system care

the IRIS 4D System

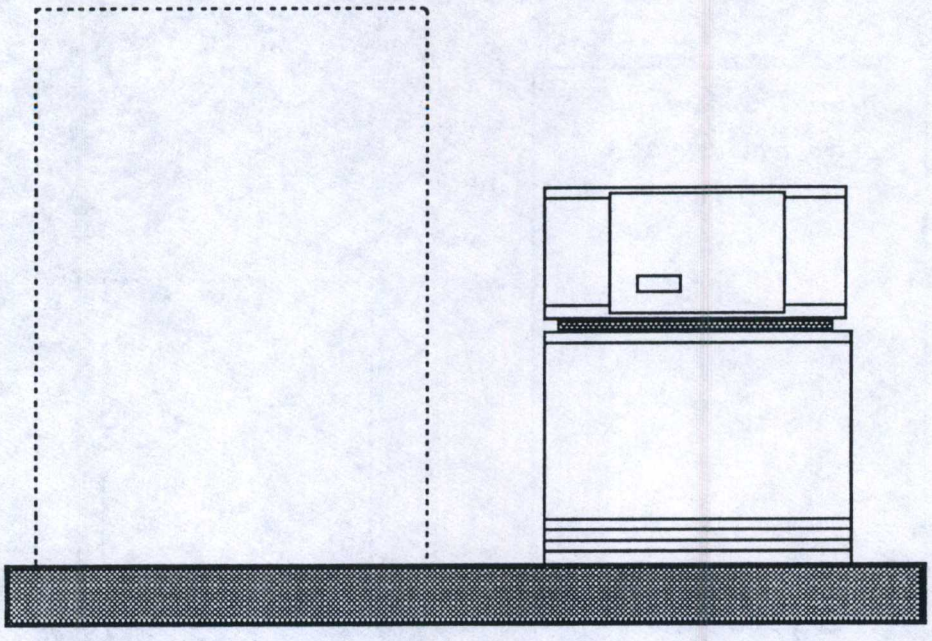


IRIS 4D CPU Stack



- processor boards
- memory boards
- graphics boards
- terminal ports
- miscellaneous options

IRIS 4D I/O Stack



ESDI:

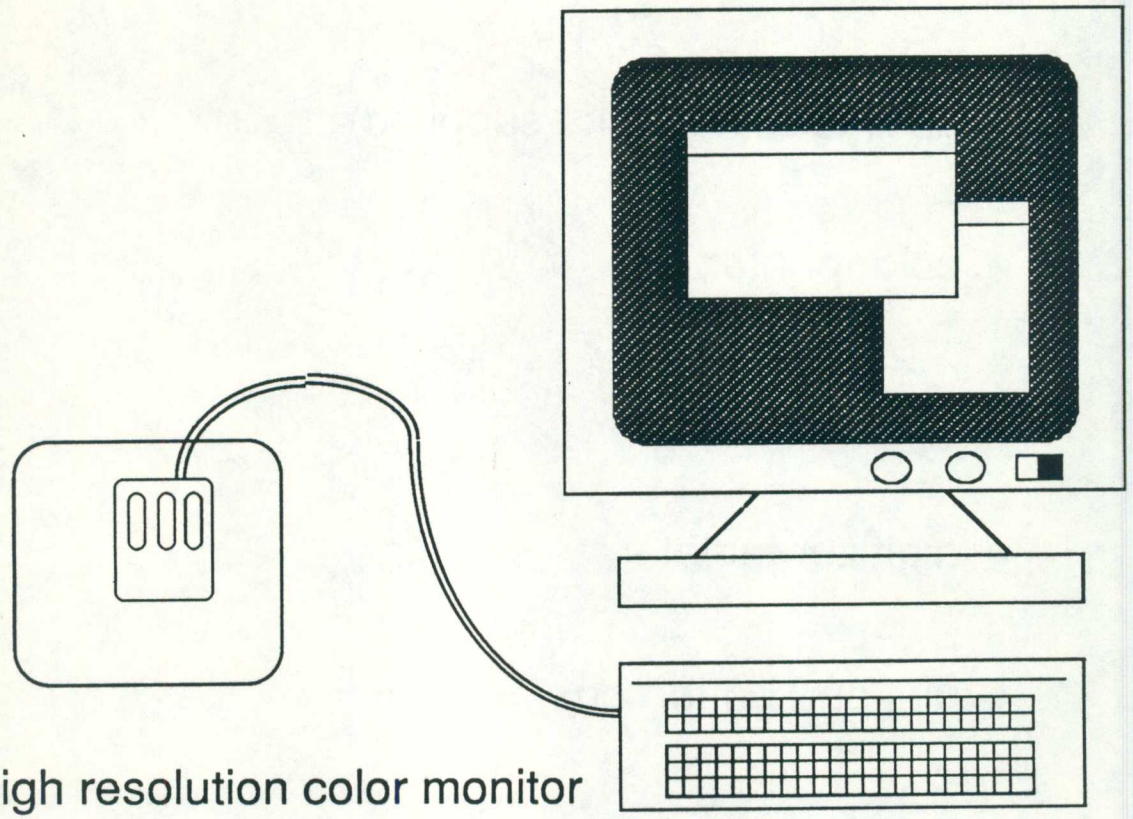
- up to two disk drives per controller
  (maximum of two controllers)

SCSI:

- one controller, up to seven SCSI devices
  (maximum of four drives on I/O stack)

Cartridge Tape Unit

IRIS Graphics Monitor

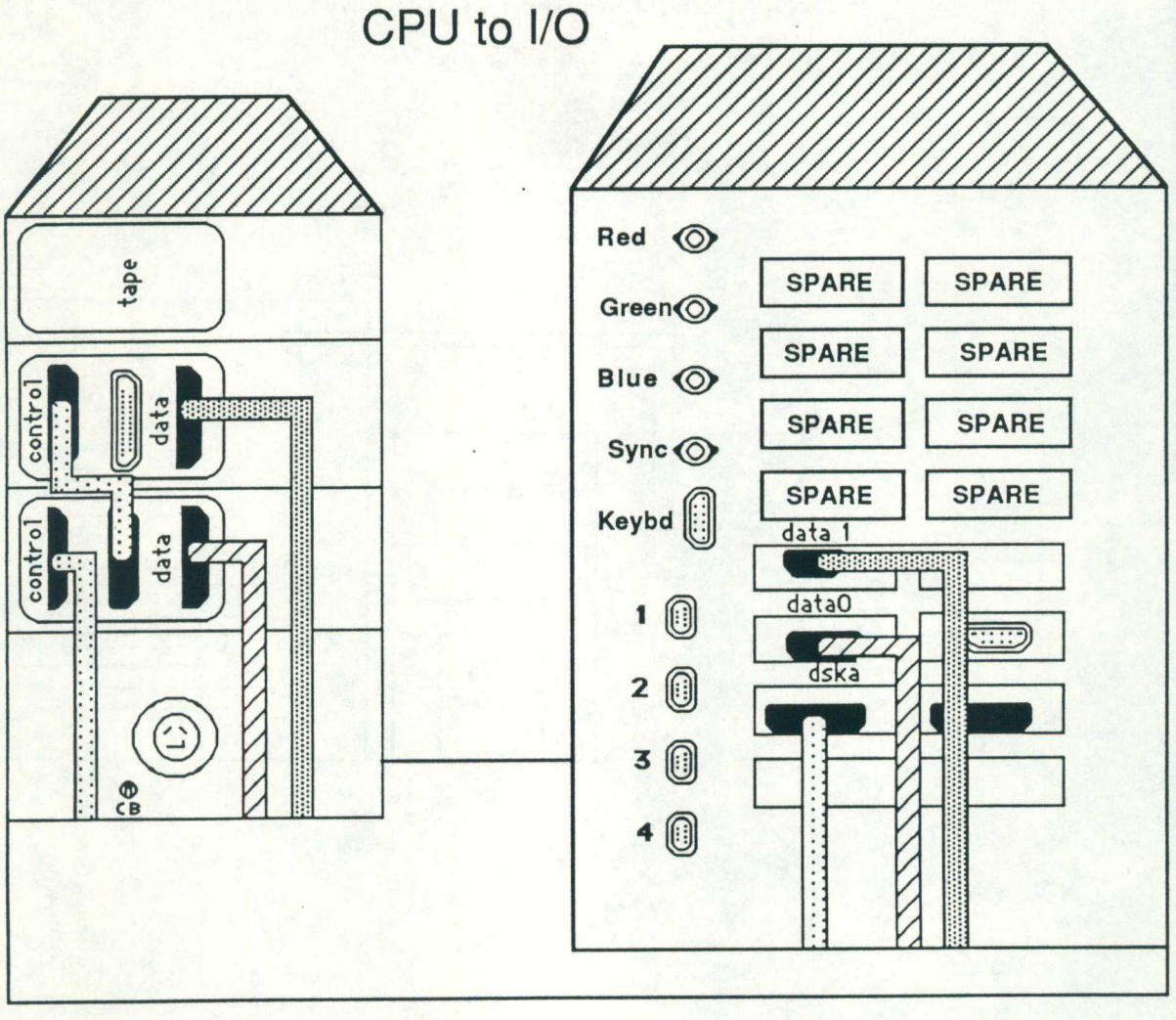


- high resolution color monitor
- graphics keyboard interface
- infrared mouse controller

IRIS Peripherals

- ascii terminals (not supplied)
- color printer
- color plotter
- digitizer tablet
- dial and button box
- laser printer

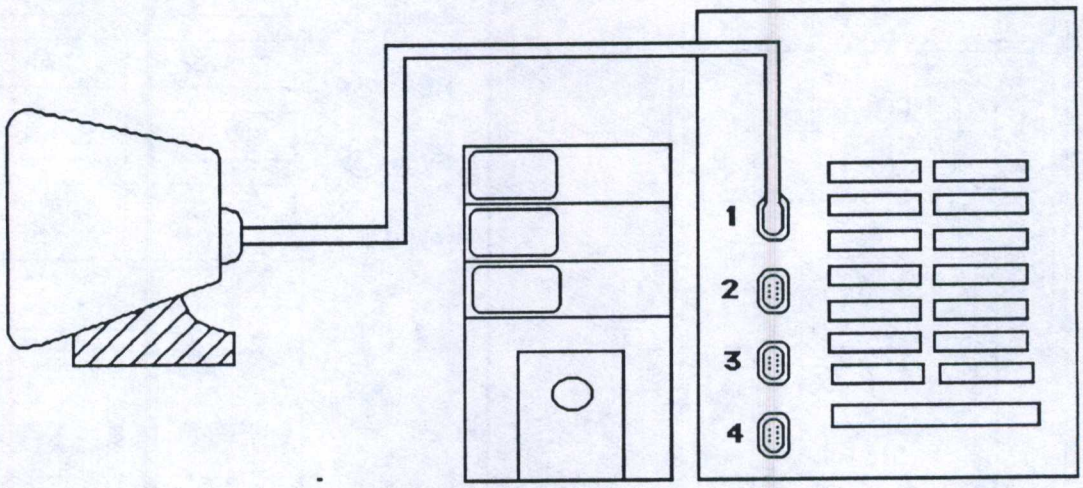
System Communication



- ESDI has external cables
- SCSI has internal cables

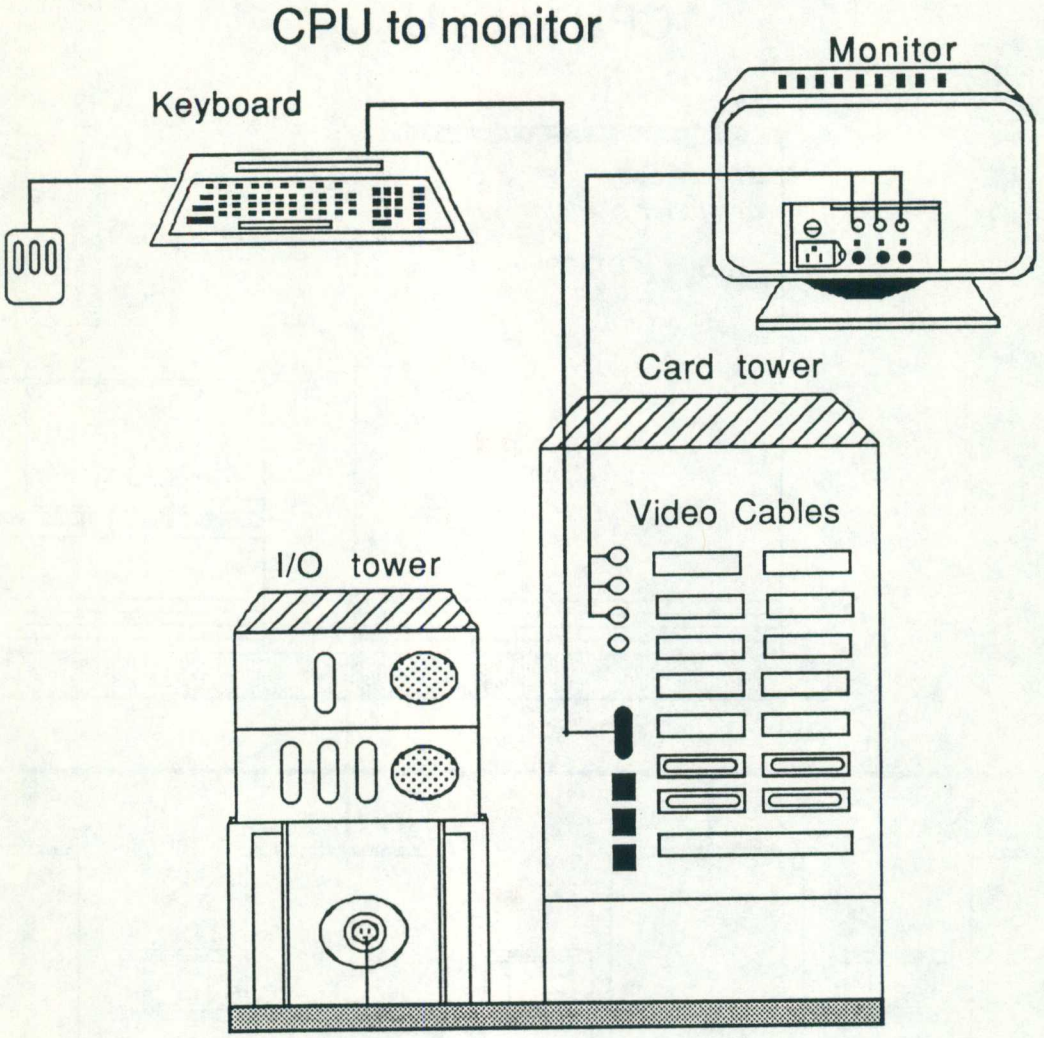
System Communication

CPU to terminal



- RS-232 "null modem" cable

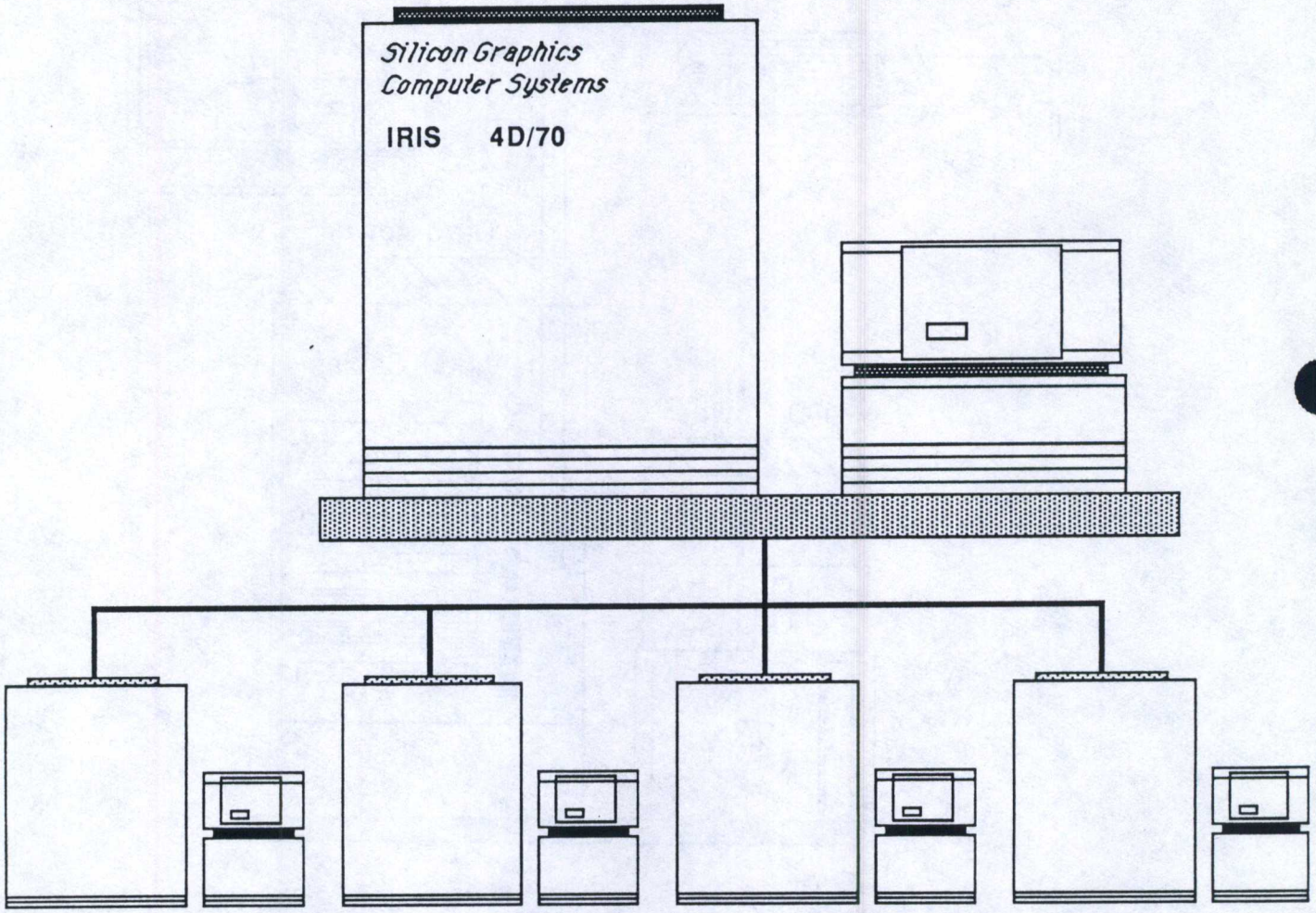
System Communication



- RGB to monitor
- graphics keyboard interface

Inter-System Communication

CPU to CPU



- ethernet with TCP/IP

System Startup Procedure

1. connect all components

2. power up console, (terminals)

3. power up CPU

   wait one minute for disk to come up to speed

4. at >> prompt, type: *auto*

5. wait for system prompt: *login:*

6. log on

System Shutdown Procedure

1. obtain root permission

2. warn all users

3. change to root directory

4. issue shutdown command

5. wait for prompt: >>

6. power off system

System Precautions

- don't power off except for special occasions
- don't move system while powered up
- avoid electrostatic discharge
- keep skins and covers on
- never remove skin or cover while system is running

Intro to Device Communication

- unix kernel contains device drivers
- drivers control operation of peripherals:

  character devices

  block devices
- device files link driver with kernel processes
- writing to most devices is same as writing to file

System Devices

- /dev/console, /dev/syscon, /dev/systty
- /dev/ttyd(1-12), /dev/ttym(1-12), /dev/ttyq(1-99)
- /dev/root, /dev/rroot
- /dev/usr, /dev/rusr
- /dev/dsk, /dev/rdsk
- /dev/tape, /dev/nrtape
- /dev/mt, /dev/rmt

System Startup and Shutdown

Objectives:

- overview of startup process
- prom and sash environments
- automatic boot process
- manual boot process
- boot problems
- shutdown process

Boot Process Overview

- power on (or reset)
- diagnostics
- load prom
- load sash
- load operating system

prom and sash low level utilities

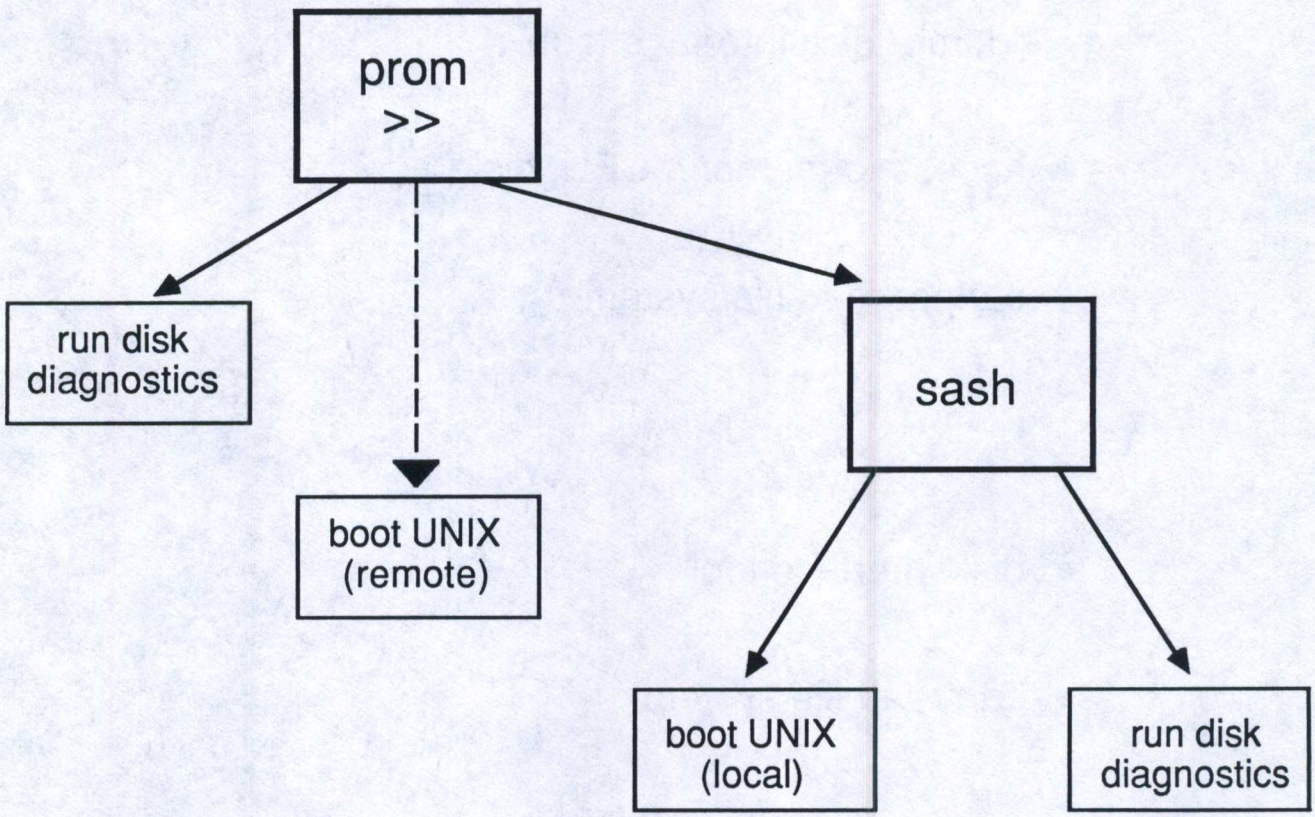
prom monitor:

- "dumb" disk interface tool
- access to memory, CPU, disk
- cannot read file system *

sash: - standalone shell.

- disk interface tool
- can read file system *

Boot Options



prom and sash Functions

prom monitor:

- boot sash from disk or tape.
- boot unix remotely from other disks (or sash)
- run diagnostics MIPS diags
  fx diags
- display and change environment variables

sash: standalone shell

- boot unix
- access file system (*cat*)
- load and run diagnostics (fx)

Automatic Boot Sequence

>> auto

ENV Variable

- prom checks bootfile for location of sash

  default is *dkip(0,0,8)sash* (ESDI)

  default is *dksc(0,1,8)sash* (SCSI)

- sash is loaded

- sash checks volume header for location of operating system

  default is *dkip(0,0,0)unix* (ESDI)

  default is *dksc(0,1,0)unix* (SCSI)

- operating system is loaded

prom and sash Review

- disk interfaces
- prom on prom, sash on disk
- used to boot unix, other system, diagnostics
- control system environment prior to boot-up

Useful prom Commands

- auto — autoboot.
- help (?)
- printenv
- setenv
- unsetenv

(printenv, setenv, unsetenv) — ENV variables.

- init - update values in PROM
- boot — manual boot.

(MIPS Supplied)

prom and sash Environment Variables

- netaddr - net address for booting over network
  default = 97.0.0.255
- dbaud - baud rate of alternate console
  ascii terminal on PORT 1
- rbaud - not used
- bootfile - location of sash
- bootmode - type of boot (d, m) defines powerup diags.
  extensive default (limited)
- path - list of devices for bootfile (or sequence if first one failed)
- console - which device is the console g, G, d, a
  (Port 1) (both devices)
- gfx - graphics subsystem status flag
- root - specifies default boot device
  ips = ESDI
  dks = SCSI

Manual Boot Sequence

manual boot needed to:

- stop at sash
- override sash's bootfile (on volume header)

process:

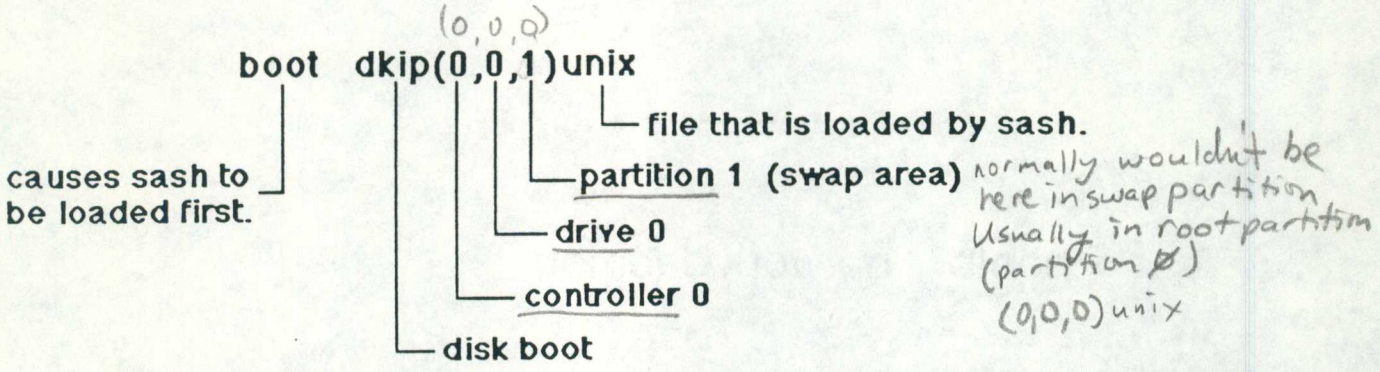
- issue *boot* command
- prom checks bootfile for location of sash
- load sash
- sash checks command argument for system to load

from volume header (boot block)

the boot Command

format:

boot [-f] device(address)file



notes:

boot loads sash (reference prom bootfile)

sash loads *file* from *address* on *device*

-f flag required to bypass sash

* → ( ) defaults to (0,0,0) for ESDI and (0,1,0) for SCSI

not available (a system bug)

Boot Devices

- dkip - ESDI disk
- dksc - SCSI disk

- bootp - network daemon

- tpsc - SCSI tape
- ~~tqic~~ - VME-QIC tape

Can only boot diags or sash off of tape. NOT UNIX

mistake → ~~tqic~~ tpqic
ok →

Disk Location

ESDI:

- controller 0 or 1
- drive 0 or 1

SCSI:

- controller always 0
- drive 1 or 2 or 3 or 4

Partitions:

- 0 - root file system
- 1 - swap space virtual memory
- 8 - volume header - disk label - has disk info, partition maps, sash, fx diags
- 2,5,6 - usr file system

Manual Boot Examples

- boot dksc(0,1,0)unix
- boot dkip(0,0,0)unix
- boot -f bootp()sgi:/usr/local/boot/unix
- boot -f tpqic()fx.ip4 → for single-board computer
  tpqic()fx.R230 → for multi-board computers
- boot

FROM SASH:

exit - goes to PROM from sash

C+(-D)

Manual vs Auto Bootup

manual boot:

- boot - loads sash from bootfile location
  -f (overrides default & boots from another file)
- sash - loads system from supplied location

automatic boot:

- boot - loads sash from bootfile location
- sash - loads system from default location

Boot Lab I

Objectives:

- prom and sash
- prom environment variables
- automatic boot
- manual boot

4D-System Adminstration

Booting Notes

*** Booting unix

Automatically :

```
>> auto
```

With defaults :

```
>> boot dkip()unix       (ESDI)
>> boot dksc(0,1,0)unix  (SCSI)
```

With direction :

```
>> boot dkip(0,0,0)unix (ESDI)
>> boot dksc(0,1,0)unix (SCSI)
```

With a remote disk drive :

```
>> boot -f bootp()server:/unix
```

Note : make sure that netaddr environment variable is set, and that the remote system has the bootp daemon running.

*** Booting Sash

Automatically :

```
>> boot
```

With direction :

```
>> boot dkip(0,0,8)sash   # MBC or SBC with ESDI
   boot dksc(0,1,8)sash   # SBC with SCSI
```

With a second drive :

```
>> boot dkip(0,1,8)sash   # MBC or SBC with ESDI
   boot dksc(0,2,8)sash   # SBC with SCSI
```

With a remote workstation's disk drive over the network:

```
>> boot -f bootp()server:/stand/sash
```

Note : make sure that netaddr environment variable is set, and that the remote system has the bootp daemon running.

With local Product Distribution tape:

```
>> boot -f tpqic()sash.R2300    # MBC with VME-QIC
   boot -f tpqic()sash.IP4      # SBC with VME-QIC
   boot -f tpsc(0,7,0)sash.IP4  # SBC with SCSI
```

With a remote tape drive :

```
>> boot -f bootp()two:/dev/tape(sash[.IP4])
```

Note : typing 'boot' will use default set by bootfile parameter. Sash can also be booted from alternate drives, and the /stand directory.

*** Booting FX

System Startup and Shutdown - Lab 1

Introduction

This lab provides practice booting and shutting down the system. It also provides practice using several prom commands and setting the environment variables.

If you are sharing a system, only one person can issue the commands given in the lab. Even if you are not the one typing in the commands, make sure you are completely familiar with the commands described.

Lab Outline

`Step 1. Booting sash and unix`

In this step you will practice booting unix from prom using the *auto* command. You will also practice booting sash, and booting unix from sash, using several variations of the *boot* command.

Automatic Boot

Perform the following steps:

- Make sure your console and ascii terminals are turned on, then turn on the power to the system.
- When you see the "prom" prompt, type **auto** to automatically boot the system, and notice the events that occur.

  This is the normal appearance of the boot process when booting into *single-user* state. (Your own system may display slightly different information than the ones you use in lab. When you return to your site, make sure you become familiar with the behavior of your own boot-up process).

  At this point you should see the root prompt (#) on your console.

- Make sure you are in the root directory (**pwd**), then type **init 0** to return to the prom monitor.

  init 0 is a convenient method for quickly returning to prom from the unix operating system. However, *you should not use* **init 0** *when other users are logged on, as they will be unceremoniously dumped from the system with no warning.*

Manual Boot

Perform the following steps:

- Type **printenv** and display the prom environment variables. Write down the values of the following variables in the space provided:

  bootfile: dkip(,,8)sash

  console: g

  bootmode: m

  The *bootfile* identifies the location to look for *sash*, and it also gives you an idea about the type of disk drive on your system. For ESDI disks, the boot file begins "dkip...", and for SCSI disks, the bootfile begins "dksc...".

  Please note which device you have, as this will be important throughout the remainder of this lab, and for the remainder of the course.

- Boot unix manually, using the proper disk device and address for your system:

boot dkip(0,0,0)unix (ESDI, drive 0)

or

boot dksc(0,1,0)unix (SCSI, drive 1)

Circle which of the above boot commands you used.

- Type **init 0** to return to prom.
- Next, from prom, boot to sash *without* continuing on to unix:

  boot (no other arguments)

  Where is sash being loaded from?

  disk

  Where is the location of sash specified?

  dkip(0,0,0)sash

  What would you expect to happen if this variable were changed?

  can't find sash program on disk

  NOTE: you should never change the location specified in the bootfile for sash. If you need to boot sash from a tape or a remote system, use the *boot -f* option to override the bootfile variable.

- From sash, boot unix by issuing the complete boot command you used earlier.

  Notice that this attempted to sash to load again, then stopped.

  To boot unix from sash, all you need to type is the second half of the complete boot command:

dkip(0,0,0)unix (ESDI system, drive 0)

or

dksc(0,1,0)unix (SCSI system, drive 1)

Issue the appropriate command for your system that will boot unix.

Type **init 0** to return to prom.

- Boot sash one more time.

  Now, type **auto**

  Describe what happens:

  tries to reload sash.

  The **auto** boot command can only be issued from the *prom* monitor, and not from *sash*. Once you are in *sash* you can only return to prom by pressing your system's "reset" button. - or Ctl-D or exit

Review questions for manual boot:

- when in prom, typing **boot** will:

  load sash

- to boot from prom to sash to unix with a single command, type:

  auto

- typing **boot** or **auto** in sash will:

  io error
  can't overload sash.

- to boot unix from sash, type:

  d---(0,-,0)unix

Step 2. prom Environment Variables

In this step you will experiment setting several of the prom environment variables and discovering how they affect the boot process.

Perform the following steps:

- Return to prom.

  Now, change the console boot device to 'd':

  printenv (displays current variable values)

  setenv console d (sets the console to the alternate)

  init (required to update the prom variable values)

  Where is the prom prompt located?

  on the alternate console

  Type **printenv** from the alternate console.

  The above command sequence (**printenv, setenv, init, printenv**) is always recommended when changing the prom environment variables. By first displaying the current settings, then setting the new variable value, reinitializing the prom monitor, and finally displaying the new settings, you can easily catch any errors you may have made *before* they cause you any problems.

- Now, boot unix from the alternate console, and note any differences from what you would normally expect to see.

  Return to prom and reset the console variable to "g" or "G".

- Power down the computer and disconnect the graphics keyboard.

- Power up the computer.

  Where is the prom prompt?

  PORT 1

- Display the environment variables and identify the gfx flag:

  gfx = dead

- Boot unix from the ascii terminal.
- Return to prom.
- Power off and reconnect the graphics console
- Verify that the system boots properly from the graphics console.

In the last few steps you saw how the system will *automatically* boot from the alternate console when the graphics subsystem appears to be "dead".

You also learned how you can manually set the console device to the ascii terminal by setting your *console* prom variable to "d".

At times you may want to do this to avoid booting from the graphics console.

Let's try one more prom environment variable:

- Return to prom, then set the *bootmode* variable to "d".

  Now, press the reset button on your computer.

  Set the *bootmode* variable to "m", and press the reset button on your computer.

  Describe any differences you notice (don't worry about writing down all the details)

Bootmode "m" runs a brief set of cpu diagnostics, while "d" runs much more extensive diagnostics. Although you will not learn how to interpret these messages in this course, you should become familiar with what is "normal" on your own system. You should also know how to run the diagnostics, in case a problem

arises that requires on-site service by your FE (Field Engineer).

`Step 3. using sash`

So far, you have only used sash to boot unix. Sash is also very useful for checking out your file system if you have problems booting.

Suppose your boot command fails because it can't load unix. If you are sure your command syntax (device, address, file) is correct, perhaps your kernel or an important file is corrupted or missing.

While sash cannot tell you if unix or any other program is corrupted, it can confirm the existence of the file. And for important text files such as */etc/passwd* and */etc/inittab*, you can use sash to display the file and check for obvious errors.

Here's how:

- boot sash from the prom monitor
- now, use the **cat** command to display a file from the disk:

  cat dksc(0,1,0)/etc/passwd (SCSI)

  or

  cat dkip(0,0,0)/etc/passwd (ESDI)

  Try this on your system.

- Try using **cat** to confirm the existence of the file *unix*.

  (*NOTE:* if the file is present you will see "garbage" displayed on your screen. Press control-c to stop the display.)

- The following files are essential to the boot process:

  - /etc/inittab
  - /etc/passwd
  - /etc/init

- /unix

All of these files are located on your *root* partition (partition 0), and if any are missing you can expect problems booting up.

Return to your seats for discussion and review.

Booting with Showconfig

default to 0,0,0

```
>> boot dkip()unix showconfig

109328+22752+187040 entry: 0xa0300000
MIPS Standalone Shell Version 4D1-1.0  MIPS OPT Fri Apr 3 13:53:18 PST 1987 SGI


Loading dkip(0,0,0)unix
549376+58800+172464 entry: 0x80028000

UNIX System V Release 3.2 4D/60 Version 2
Total real memory = 8388608
Available memory = 7311360

ttyc0: missing          6-port option
if_en1: missing         SGI Ethernet Controller #1
if_en0: missing         SGI Ethernet Controller #0
if_enp1: missing        CMC Ethernet Controller #1
if_enp0: hardware address 02:cf:1f:10:40:18        CMC Controller #0
xmt0: missing           1/2" Controller
ikc0: missing           Ikon Color Printer Controller

NOTICE: GF3 pipe at 0x1C900000

ips0 firmware prom id 0x6526.D (9/18/1986), ipl 1, vec 0xA5
2457600 bytes of cache, 75 buffers
malloc virtual region: 0xC05B2000 - 0xC0631FFF
ips0d0: 823/10/32, 1:1 interleave       170Meg disk drive 1
ips0d1: 823/10/32, 1:1 interleave       170Meg disk drive 1
swapping on dev 0x401, 32800k bytes
root on dev 0x400 (fstyp is efs)
Available memory = 7233536
```

Boot Troubleshooting

no prom prompt on console:

- power off system
- disconnect graphics keyboard
- power on, check for prompt on alternate console
- printenv - gfx flag = "dead"
- boot

Boot Troubleshooting

prom prompt on console, but can't boot:

- setenv console d
- init
- auto

if *still* can't boot,

- power off system
- disconnect graphics keyboard
- power on, check for prompt on alternate console
- printenv -(gfx flag = "dead")
- boot

Introduction to User Modes

single user mode:

- only console is active
- only root user exists (no login: prompt)
- system ready for maintenance

only root mounted

multi user mode:

- all enabled ports active
- login prompt
- usr file system is mounted
- system ready for general use

When to Shut Down

- add system software
- move system
- add disk drive, memory
- service system
- troubleshoot file system

Shutdown States

- return to single user mode
- return to prom
- power off

Two Commands to Shut Down

reboot (pre 3.0)

- impolite
- immediate, no grace period

shutdown (preferred)

- flexible
- broadcast message
- grace period

Shutting Down with reboot

- become superuser, cd to root directory
- notify specific user with *write*
- *who, rwho, whodo, ps -ef*
- *showmount* for remote file systems
- *wall* to all local users
- *rwall* to all remote users
- verify all users off
- type *reboot*
- wait for prompt (>>)
- (power down)

Shutting Down with shutdown

- become superuser, cd to root directory
- notify specific user with *write*
- *who, rwho, whodo, ps -ef*
- *showmount* for remote file systems
- *wall* to all local users
- *rwall* to all remote users
- verify all users off
- type *shutdown -y -g30 -i0*
- wait for prompt (>>)
- (power down)

Shut Down to Single-user Mode with shutdown

- become superuser, cd to root directory
- notify specific user with *write*
- *who, rwho, whodo, ps -ef*
- *showmount* for remote file systems
  showmount -e
- *wall* to all local users
- *rwall* to all remote users (systems)
- verify all users off
- type *shutdown -y -g30 -i1*

  or

- type *shutdown -y -g30 -is*

Rules for System Reset

from multi or single.

never press *reset* button unless:

- the system is hung, and...
- you cannot use other terminal, and...
- you cannot remotely log in, and...
- you have waited two minutes for the kernel to update the disk

Boot Lab II

Objectives:

- practice shutdown, reboot commands
- differentiate between single, multi-user modes

from single:
"multi"
"init 2"

System Startup and Shutdown - Lab 2

Introduction

This lab provides practice shutting down the system. It also explores some of the differences between single and multi-user mode.

If you are sharing a system, only one person can issue the commands given in the lab. Even if you are not the one typing in the commands, make sure you are completely familiar with the commands described.

Lab Outline

Step 1. Booting into Multi-User mode

In the previous lab, all of your boot commands ended with the system in *single-user* mode. In *single-user* mode, only the root user exists, and there is no *login* prompt.

You system has been set up to boot into single user mode. Later, you will learn how to change the system configuration so that it automatically boots in to *multi-user* mode, so that other users can access the system.

In the meantime, you will force the boot command to boot to *multi-user* mode by including the *initstate* option to the boot command.

Perform the following steps:

- Make sure you are at the prom monitor, then type the appropriate boot command for your system. Make sure you include the *initstate* argument:

  boot dkip()unix initstate=2 (ESDI system)

  or

  boot dksc(,1,)unix initstate=2 (SCSI system)

  Watch the console display carefully, and notice the difference during boot up.

When the system boots up, you should have the login prompt displayed on your console, and on each of your ascii terminals.

Welcome to *multi-user* state! Your system can now be accessed by any system user, simply by logging on with a valid user id and password.

Log on as *root*. Your instructor will provide the password.

- Now, type the command **ps -ef**, and notice the system daemons that are running.

  Type the command **df**, and notice the file systems that are mounted.

- Try logging on as the user *guest* on each of the ascii terminals. Your instructor will provide the password.

Step 2. Shutting down

In this step you will practice following the recommended shutdown sequence.

When your system is up and running in multi-user mode, and users are logged on, it is socially unacceptable to shut down to prom by using the *init 0* command. Instead, you should follow a complete sequence that involves notifying (begging?) your users, identifying any user processes that are running, etc., before you proceed with a *graceful* shutdown using the *shutdown* command.

Perform the following steps:

- make sure you are the *root* user, and are in the root directory:

  id

  pwd

- identify your system users, and see what they are doing:

  who

  whodo

(*Note:* if someone is running *vi*, you may not want to write a message to their terminal)

if you are part of a network, check for remote users:

rwho

- find out if any remote file systems are mounted:

  showmount

- Warn your users that you are going to shut down the system:

  write *user* <press return>

  SYSTEM GOING DOWN IN 5 MINUTES. PLEASE LOG OFF.

  <press ctl-d>

  You must repeat the above command for each user logged on.

  You can warn *all* users with a single command:

  wall <press return>

  Shutting down in 5 minutes. Please log off

  Press <ctl-d> to send the message.

  Note: You may want to repeat your message after a few minutes.

- Go ahead and log the user *guest* off of the ascii terminals.
- verify that all users are logged off:

  ps -ef

(At this time, you may want to compare the process list to the normal running processes and see if any "extra" processes are running - someone may have used **nohup** to keep a process running after they log off.)

- Type the shutdown command:

  shutdown -y -g0 -i0 (you may use a different grace period)

At this point your system should take a couple minutes to terminate all running processes, dismount any file systems, and return to the prom monitor.

Return to your seat for discussion and review.

System Initialization

Objectives:

- overview of init process
- distinguish run states
- /etc/init and the /etc/inittab file
- inittab details
- boot to run state 2, s, 1
- run level directories and *init*
- init to runstate 0, 1, s

System Initialization

/etc/init:

- first process spawned
- spawns system processes
- sets up console, terminal communication
- prepares system for use

User Modes and Run States

- single user mode
- multi user mode
- run states

  1 - single user mode

  s, S - "special" single user mode all daemons running all filesystems mounted.

  2 - multi user mode

  0 - shutdown mode

/etc/init and the /etc/inittab file

```
is:2:initdefault:
fs::sysinit:/etc/bcheckrc </dev/console >/dev/console 2>&1
cons::sysinit:/etc/gl/setupcons </dev/console >/dev/console 2>&1
link::wait:/etc/lnsyscon > /dev/console 2>&1 < /dev/null
mt:23:bootwait:/etc/brc </dev/console >/dev/console 2>&1
s0:056:wait:/etc/rc0 >/dev/console 2>&1 </dev/console
s1:1:wait:/etc/shutdown -y -iS -g0 >/dev/console 2>&1 </dev/console
s2:23:wait:/etc/rc2 >/dev/console 2>&1 </dev/console
s3:3:wait:/etc/rc3 >/dev/console 2>&1 </dev/console
of:0:wait:/etc/uadmin 2 0 >/dev/console 2>&1 </dev/console
fw:5:wait:/etc/uadmin 2 2 >/dev/console 2>&1 </dev/console
RB:6:wait:echo
rb:6:wait:/etc/uadmin 2 1 >/dev/console 2>&1 </dev/console
```

default - can be "S" or "1"

- /etc/init spawns system processes
- /etc/inittab designates which processes are run

initdefault entry determines default run state

corresponding /etc/inittab entries are run

/etc/inittab overview

the /etc/inittab file is read:

- when system is booted
- when root issues *init* command
- when root issues *telinit* command

telenit q (kill -1 1)

- when system receives powerfail signal

/etc/inittab entries

id : run states : actions : process

- if run state field is null, all states apply
- if run state is x, ignore
- if run state equals specified run state, spawn process

actions:

sysinit - at initialization, before console is accessed

boot, bootwait - only at boot-time

respawn - restart if process dies

once - run only once

wait - suspend inittab execution until process finishes

```
#! /bin/sh
#       Copyright (c) 1984 AT&T
#         All Rights Reserved

#       THIS IS UNPUBLISHED PROPRIETARY SOURCE CODE OF AT&T
#       The copyright notice above does not evidence any
#       actual or intended publication of such source code.

#ident  "@(#)initpkg:./inittab.sh       1.16"
#ident  "$Header: /clone/att/usr/src/cmd/initpkg/RCS/inittab,v 1.23 87/08/0

is:2:initdefault:
fs::sysinit:/etc/bcheckrc </dev/console >/dev/console 2>&1
cons::sysinit:/etc/gl/setupcons </dev/console >/dev/console 2>&1
link::wait:/etc/lnsyscon > /dev/console 2>&1 < /dev/null
mt:23:bootwait:/etc/brc </dev/console >/dev/console 2>&1
s0:056:wait:/etc/rc0 >/dev/console 2>&1 </dev/console
s1:1:wait:/etc/shutdown -y -iS -g0 >/dev/console 2>&1 </dev/console
s2:23:wait:/etc/rc2 >/dev/console 2>&1 </dev/console
s3:3:wait:/etc/rc3 >/dev/console 2>&1 </dev/console
of:0:wait:/etc/uadmin 2 0 >/dev/console 2>&1 </dev/console
fw:5:wait:/etc/uadmin 2 2 >/dev/console 2>&1 </dev/console
RB:6:wait:echo \"\\\nThe system is being restarted.\" >/dev/console 2>&1
rb:6:wait:/etc/uadmin 2 1 >/dev/console 2>&1 </dev/console
#
gcon:234:once:/etc/gl/grcond -i
co:234:respawn:/etc/getty console console none LDISC0   # the console
t1:234:respawn:/etc/getty -s console ttyd1 co_9600 none LDISC0  # alt consc
t2:x:respawn:/etc/getty ttyd2 co_9600 none LDISC0       # port 2
t3:x:respawn:/etc/getty ttyd3 co_9600 none LDISC0       # port 3
t4:x:respawn:/etc/getty ttyd4 co_9600 none LDISC0       # port 4
#
# Optional graphics peripherals
diad:x:once:/etc/gl/diald /dev/dials > /dev/null 2>&1 < /dev/null
tabd:x:once:/etc/gl/tabletd /dev/tablet > /dev/null 2>&1 < /dev/null
#
c0:x:respawn:/etc/getty ttyd5 co_9600 none LDISC0       # serial option por
c1:x:respawn:/etc/getty ttyd6 co_9600 none LDISC0       # serial option por
c2:x:respawn:/etc/getty ttyd7 co_9600 none LDISC0       # serial option por
c3:x:respawn:/etc/getty ttyd8 co_9600 none LDISC0       # serial option por
c4:x:respawn:/etc/getty ttyd9 co_9600 none LDISC0       # serial option por
c5:x:respawn:/etc/getty ttyd10 co_9600 none LDISC0      # serial option por
c6:x:respawn:/etc/getty ttyd11 co_9600 none LDISC0      # serial option por
c7:x:respawn:/etc/getty ttyd12 co_9600 none LDISC0      # serial option por
```

Boot to Run State 2

process overview:

- system initialization
- prepare for run state 2
- make system public

System Initialization

run *sysinit* scripts:

- check root file system, ensure /dev exists

```
fs::sysinit:/etc/bcheckrc </dev/console >/dev/console 2>&1
```

- connect "console" to either graphics port or ttyd1

```
cons::sysinit:/etc/gl/setupcons </dev/console >/dev/console 2>&1
```

- create device file for *console* and *syscon*

```
link::wait:/etc/lnsyscon > /dev/console 2>&1 < /dev/null
```

Prepare for Run State 2

- check */etc/fstab* for file systems to mount:

```
mt:23:bootwait:/etc/brc </dev/console >/dev/console 2>&1
```

- mount file systems, start system daemons:

```
s2:23:wait:/etc/rc2 >/dev/console 2>&1 </dev/console
```

rc2 Scripts

- mountfsys - mount file systems
- rmtmpfiles - remove files from /tmp, /usr/tmp
- syssetup - print configuration name, set system name
- perf - print copyright notice
- tcp - start TCP daemons *portmap, inetd, rwhod, bfsd*
- lp - remove *SCHEDLOCK* file, start lp scheduler
- cron - start *cron* daemon
- display message "system is ready"

Make System Public

- start graphics subsystem:

```
gcon:234:once:/etc/gl/grcond -i
```

mex windowing system.

- spawn *getty* on console:

```
co:234:respawn:/etc/getty console console none LDISC0
```

- spawn *getty* on alternate console:

```
t1:234:respawn:/etc/getty -s console ttyd1 co_9600 none LDISC0
```

- spawn *gettys* on other terminal ports:

```
t2:x:respawn:/etc/getty ttyd2 co_9600 none LDISC0
t3::respawn:/etc/getty ttyd3 co_9600 none LDISC0
t4::respawn:/etc/getty ttyd4 co_9600 none LDISC0
```

↓ points to /etc/gettydefs

Boot to Run State s

- open virtual console for writing (*/dev/console*)
- execute /bin/su
- only main system processes are running
- only root file system exists
- no login required - automatically root user

Boot to Run State 1

- perform initialization steps
- only main system processes are running
- only root file system, root user exists
- execute inittab entry for *shutdown*:

```
s1:1:wait:/etc/shutdown -y -iS -g0 >/dev/console 2>&1 </dev/console
```

Forcing Boot to Alternate Run State

boot dkip(0,0,0)unix initstate=*N*

- overrides *initdefault* entry, performs initialization for run state *N*

4D 20 - All 4D running SCSI @ 3.1D
boot dksc(0,1,0)unix initstate=1

Init Lab I

Objectives:

- explore /etc/init and the /etc/inittab file
- experiment with boot to different run states
- manually boot to a specific run state

Changing Run States with init

init *runstate*

- init consults inittab, executes appropriate scripts
- init from 2 to 0 or 1 requires complete shutdown routine

System Initialization - Lab 1

Introduction

This lab explores the process of booting to different init states, and explores the differences between the init states achieved after booting.

Lab Outline

`Step 1. booting to single and multi user mode`

Perform the following:

- boot your system into single user mode, by typing **auto**.
- Identify the processes that are running and the file systems available:

  ps -ef

  df

  Processes:

  File Systems:

- Now, bring your system up into multi user mode:

  multi

  or

  init 2.

- Log on as *root*, and identify the processes that are running and the file systems that are available (don't write them all down, just notice the difference between multi and single user modes):

  ps -ef

  df

  (don't write them all down, just notice the difference).

`Step 2. booting to Multi-user mode`

In this step you will modify your system so that it automatically boots into multi-user mode.

Perform the following steps:

- use *vi* to modify the */etc/inittab* file so that "2" is the default init state.
- Return your system to the prom monitor, then type **auto** to boot automatically into multi-user mode.

  Log on the the system as *root* and use the *shutdown* command to return the system to the prom monitor:

  shutdown -y -i0

  (this will use the default grace period of 60 seconds).

By setting the *initdefault* field to "2" you avoid having **auto** boot into single-user mode. *An automatic boot into single user mode is a potential security problem, as any user can type* **auto** *and gain access to your system as "root".*

Step 3. booting to init state 1

Now that your */etc/inittab* file lists init state "2" as the default run state, you will need to manually override the *initdefault* field to boot to single user mode.

Try the following:

- boot to initstate 1 using the manual boot command:

  boot dksc(,1,)unix initstate=1 (SCSI)

  or

  boot dkip()unix initstate=1 (ESDI)

Step 4. Boot to initstate S

Perform the following steps:

- Return your system to the prom monitor, then manually boot into initstate "S" (upper-case "S"):

  boot dkip()unix initstate=S (ESDI)

  or

  boot dksc(,1,)unix initstate=S (SCSI)

  Describe the appearance of the system once your bootup completes.

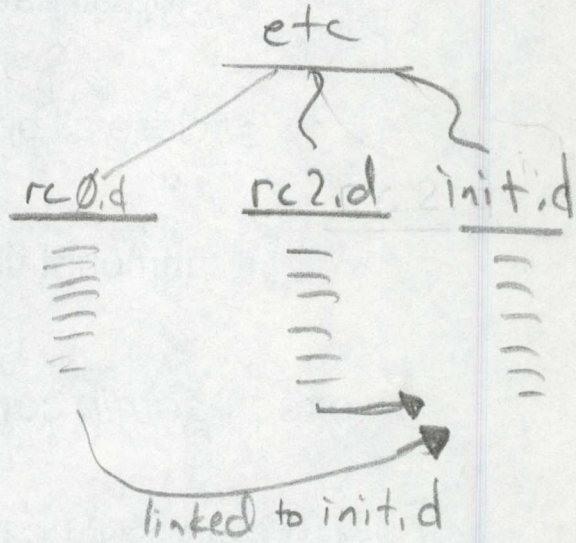
  What run state are you in?

Return to your seat for the next section.

Run Level Directories

consulted for run states 0 and 2

- contain "start" and "kill" scripts that control the system environment
- run state 2 uses */etc/rc2.d*
- run state 0 uses */etc/rc0.d*
- format:



S00*name* - run "start" scripts /etc/init.d/*name*

K00*name* - run "kill" scripts /etc/init.d/*name*

note: rc2, rc0 scripts linked to /etc/init.d

init to Run State 0

run rc0 scripts:

```
s0:056:wait:/etc/rc0 >/dev/console 2>&1 </dev/console
```

- stop all daemons and services
- close all open files and stop user processes
- unmount usr file system

exec *uadmin* command:

```
of:0:wait:/etc/uadmin 2 0 >/dev/console 2>&1 </dev/console
```

- unmount root file system
- halt system (return to prom)

init to Run State 1

exec *shutdown* command:

```
s1:1:wait:/etc/shutdown -y -iS -g0 >/dev/console 2>&1 </dev/console
```

- checks for root authority, root directory
- validate run state
- checks backup schedule
- broadcasts warning, prompt for verification
- runs rc0 scripts
- executes *init* to specified run state

init to Run State s, S

note: init to 's' is different than boot to 's'!

- no entries in /etc/inittab
- links /dev/console to user's terminal
- enter "single user" state (no gettys)
- file systems remain mounted
- all other processes remain running

Init Lab II

Objectives:

- explore /etc/init and the /etc/inittab file
- experiment with init to different run states
- characterize run states 0, 1, s, and 2
- explore rc2 and rc0 scripts

1. Boot to multi.
2. init to S
3. init to 2
4. init to 1

```
#! /bin/sh
#ident  "$Header: /clover/root/att/usr/src/cmd/initpkg/RCS/shutdown.sh,v 1.8 87/04/

#       Sequence performed to change the init stat of a machine.

#       This procedure checks to see if you are permitted and allows an
#       interactive shutdown.  The actual change of state, killing of
#       processes and such are performed by the new init state, say 0,
#       and its /etc/rc0.

#       Usage:  shutdown [ -y ] [ -g<grace-period> ] [ -i<init-state> ]

#!      chmod +x ${file}

if [ `pwd` != / ]
then
        echo "$0:  You must be in the / directory to run /etc/shutdown."
        exit 1
fi

#       Check the user id.
if [ -x /usr/bin/id ]
then
        eval `id  |  sed 's/[^a-z0-9=].*//'`
        if [ "${uid:=0}" -ne 0 ]
        then
                echo "$0:  Only root can run /etc/shutdown."
                exit 2
        fi
fi

grace=60
askconfirmation=yes
initstate=s
while [ $# -gt 0 ]
do
        case $1 in
        -g[0-9]* )
                grace=`expr "$1" : '-g\(.*\)'`
                ;;
        -i[Ss0156] )
                initstate=`expr "$1" : '-i\(.*\)'`
                ;;
        -i[234] )
                initstate=`expr "$1" : '-i\(.*\)'`
                echo "$0:  Initstate $i is not for system shutdown"
                exit 1
                ;;
        -y )
                askconfirmation=
                ;;
        -* )
                echo "Illegal flag argument '$1'"
                exit 1
                ;;
        * )
```

```
                echo "Usage:  $0 [ -y ] [ -g<grace> ] [ -i<initstate> ]"
                exit 1
        esac
        shift
done

if [ -n "${askconfirmation}" -a -x /etc/ckbupscd ]
        then
        #       Check to see if backups are scheduled at this time
        BUPS=`/etc/ckbupscd`
        if [ "$BUPS" != "" ]
                then
                echo "$BUPS"
                echo "Do you wish to abort this shutdown and return to
command level to do these backups? [y, n] \c"
                read YORN
                if [ "$YORN" = "y" -o "$YORN" = "Y" ]
                        then
                        exit 1
                fi
        fi
fi

if [ -z "${TZ}"  -a  -r /etc/TIMEZONE ]
then
        . /etc/TIMEZONE
fi

echo 'Shutdown started.     \c'
date

sync
cd /

trap "exit 1"  1 2 15

a="`who  |  wc -l`"
if [ ${a} -gt 1  -a  ${grace} -gt 0 ]
then
        if [ -x /etc/wall ]
        then
                su adm -c /etc/wall<<-!
                        The system will be shut down in ${grace} seconds.
                        Please log off now .
                !
        fi
        sleep ${grace}
fi

if [ -x /etc/wall ]
then
        /etc/wall <<-!
                THE SYSTEM IS BEING SHUT DOWN!  Log off now.
        !
fi
sleep ${grace}
```

```
if [ ${askconfirmation} ]
then
        echo "Do you want to continue? (y or n):   \c"
        read b
else
        b=y
fi
if [ "$b" != "y" ]
then
        if [ -x /etc/wall ]
        then
                /etc/wall <<-\!
                        False Alarm:  The system will not be brought down.
                !
        fi
        echo 'Shut down aborted.'
        exit 1
fi
case "${initstate}" in
s | S )
        . /etc/rc0
esac
/etc/init ${initstate}
```

User Support

Objectives:

- add new user accounts
- maintain system security
- monitor system resources
- maintain *crontab* utility
- maintain *news* bulletin board

User Setup

Process Overview:

- identify user account characteristics
- add account to */etc/passwd*
- set up user's login directory
- establish group memberships
- have user verify setup, create password
- update system logbook

User Account Information

- unique user name
- unique user identification number
  0 or -2 reserved
- group identification number
  # 20 is default
- home directory
  /usr/people/*
- login shell
  sh or csh
- configuration files
  .login > csh.
  .cshrc
  .profile > sh

Add Account to */etc/passwd*

```
root::0:0:0000-Admin(0000):/:/bin/csh
setup::0:0:general system administration:/usr/admin:/bin/rsh
powerdown::0:0:general system administration:/usr/admin:/bin/rsh
sysadm::0:0:general system administration:/usr/admin:/bin/rsh
daemon:*:1:1:0000-Admin(0000):/:
bin:*:2:2:0000-Admin(0000):/bin:
uucp::3:5:0000-uucp(0000):/usr/lib/uucp:
sys:*:4:0:0000-Admin(0000):/usr/src:
adm::5:3:0000-Admin(0000):/usr/adm:
lp:*:9:9:0000-lp(0000):/usr/spool/lp:
nuucp::10:10:0000-uucp(0000):/usr/spool/uucppublic:/usr/lib/uucp/uucic
man:*:99:995:0000-Admin(0000):/:
diag::0:996:Hardware Diagnostics:/usr/diag:/bin/csh
tutor::994:997:Tutorial User:/usr/tutor:/bin/csh
demos::993:997:Demonstration User:/usr/demos:/bin/csh
guest::998:998:Guest Account:/usr/people/guest:/bin/csh
games:*:999:999:Games:/usr/games:/bin/sh
john::990:990:John Smith:/usr/people/john:/bin/csh
```

defaults

- one line per user account
- supplied file has default user accounts, special accounts
- writeable only by root, readable by everyone

/etc/passwd Entries

account : passwd : userid : groupid : username : home dir : login shell

```
john::990:990:John Smith:/usr/people/john:/bin/csh
```

- account name must be unique
- passwd empty until user creates password
- user identification number must be unique
- group identification defines default group membership
- user name for information only
- home dir is login directory
- login shell defines Shell process invoked at login

Supplied Accounts

user account	used for
root	root user
guest	sample user account
demos	graphics demo account
tutor	graphics tutorials

special account	used for
setup	restricted root shell for admin functions
powerdown	reserved account for powering down
sysadm	special account for admin scripts (restricted)
daemon	account for system daemons
bin	references path for /bin directory
uucp	used for uucp processes
sys	system account, references /usr/src directory
adm	administrative tasks, references /usr/adm
lp	used by line printer spooler
nuucp	used for uucp logins
man	points to root directory...a starting path
diag	special account for diagnostics

Set up Login Directory

- create directory:

  mkdir /usr/people/*username*

- create configuration files: in that directory.

  .login

  .cshrc

  .profile

- change ownership, group:

  chown *username* /usr/people/*username*

  chgrp *groupid* /usr/people/*username*

Configuration Files

*templates located in /etc/std**

customize user's login environment:

- set environment variables, command aliases
- set umask for file permissions
- miscellaneous functions

default scripts:

- /etc/login
- /etc/cshrc
- /etc/profile

```
stty erase ' ' kill '^U' intr '^C' echoe
set     ignoreeof noglob
set     tmp = (`tset -S -Q`)
setenv  TERM    $tmp[1]
unset   noglob
setenv  PATH .:~:~/bin:/bin:/usr/bin:/etc:/usr/etc:/usr/sbin:/usr/bsd:/usr
set     path = (. ~ ~/bin /bin /usr/bin /etc /usr/etc /usr/sbin /usr/bsd /usr)

umask 333
set history = 100
set savehist=100
stty erase ^h
tset

set prompt="<\! `/usr/bsd/hostname`.`who am i | cut -d' ' -f1`:`pwd`> "
set whom=`who am i | cut -d' ' -f1`
alias cd 'cd \!*; set prompt="<\! `/usr/bsd/hostname`.`echo $whom`:`pwd`> "'
alias pushd 'pushd \!*; set prompt="<\! `/usr/bsd/hostname`.`echo $whom`:`pwd`> "'
alias popd 'popd \!*; set prompt="<\! `/usr/bsd/hostname`.`echo $whom`:`pwd`> "'
alias c 'clear'
alias h 'history'
alias ls 'ls -CF'
alias la 'ls -a'
alias ll 'ls -al'
alias lm 'ls -am'
alias lsd 'll \!* | fgrep '/''
alias lsf 'll \!* | fgrep -v '/''
alias lsl 'lsd \!*; lsf \!*'
alias lsm 'lsl \!* | more'
```

```
# guest's sh profile

# "$Header: /clover/root/att/usr/src/cmd/adm/RCS/guest.prof,v 1.1 87/03/25 16:37:13

umask 002

PATH=:/usr/sbin:/usr/bsd:/usr/bin:/bin:/etc:/usr/etc
export PATH

stty line 1 erase '^H' kill '^U' intr '^C' echoe
eval `tset -s -Q 2> /dev/null`

# list directories in columns
ls()    { /bin/ls -C $*; }
```

```
# root's csh login profile

# "$Header: /clover/root/att/usr/src/cmd/adm/RCS/root.login,v 1.4 87/03/24 11:22:45

if ( -d /usr/lib/terminfo) then
    set noglob; eval `tset -s -Q`; unset noglob
endif
stty line 1 erase '^H' kill '^U' intr '^C' echoe



if ($?prompt) then
        if ( -o /bin/su ) then
                set prompt="`hostname`: $user \!# "
        else
                set prompt="`hostname`: $user \!% "
        endif
endif
setenv  SHELL   /bin/csh
set     ignoreeof
setenv  EPATH /usr/local/lib/emacs/maclib
set path = (. /usr/bsd /usr/local/bin /etc /bin /usr/bin /usr/sbin \
        /bin /usr/people/guest/tools /usr/people/guest/sample)
umask 022

if ( 'tty' == /dev/console || 'tty' == /dev/syscon ) then
    setenv TERM iris-ansi
else
    setenv TERM vt100
endif
```

Shell Variables

- TERM (terminal type)
- PATH (search path)

avoid .

- MAIL (mail directory)
- history (size of C shell history)
- prompt (change default prompt)

Establish Group Membership in *etc/group*

```
sys::0:root,bin,sys,adm
root::0:root
daemon::1:root,daemon
bin::2:root,bin,daemon
adm::3:root,adm,daemon
mail::4:root
uucp::5:uucp
rje::8:rje,shqer
lp:*:9:
nuucp::10:nuucp
user::20:john
diag::996:
demos:*:997:
guest:*:998:
games:*:999:
john::990:mike,jls,joe
```

- default group defined in */etc/passwd*
- "pseudo" group defined in *etc/group*
- change group affiliation with *newgrp*:

/etc/group Entries

group name : passwd : groupid : user1, user2, ...userN

```
jls::1000:john,joe,andy
```

- group name is familiar name
- user names required to *newgrp* to the group
- current problems with */etc/group*:
  - no convenient way to set group password
  - must *chmod 4755 /etc/newgrp*
  - must *chown root /etc/newgrp*
  - must *chgrp sys /etc/newgrp*

Verify and Document

- user verifies account
- user sets passwd
- administrator updates logbook
- administrator backs up files

User Support Lab I

Objectives:

- set up two new users
- define group memberships
- verify functionality

System Security

- use root authority properly
- require passwords on all accounts
- employ group membership
- set proper file ownership, permissions
- watch for set-UID, set-GID
- watch for trojan horses
- consider restricted shell
- promote proper user attitudes

Rules for Root

- only user root when required
- don't let *anyone* use root
- don't leave root logged on
- invoke *su* as */bin/su*
- don't run user programs as root

Require Passwords

- all users must have passwords
- passwords must not be published
- secure root password
- employ password "aging"

Employ Group Membership

- prevents indescriminate access to files
- segregates users by function
- creates flexible work environment

File Ownership, Permissions

- /bin, /usr/bin, /etc writeable only by root
- only root access to *mount* command
- all set-UID and set-GID programs writeable only by root

set-UID and set-GID

- part of file protection code
- user executing program has same privilege as user or group

  give certain programs "privileged" mode
- gaping hole in security if not write protected
- example:

  /bin/newgrp has mode 4777

  user copies /bin/csh to /bin/newgrp

  user executes /bin/newgrp (now has root id)

a Trojan Horse

- user search path begins with .
- root su's to the user
- root su's back to root
- example:

```
%cat su

set prompt ""
stty -echo
echo passwd:
read pass
echo $pass > grab
stty echo
```

Restricted Shells

user may not:

- change directory
- change PATH
- use slash (/) in a pathname
- redirect output

User Attitudes Towards Security

users:

- view security as an aggravant
- consider security trivial
- aren't very careful

System Monitoring

- du -s
- who
- whodo

```
Tue Jun 14 12:27:43 1988
doc

ttyd3    joe        8:47
    ttyd3   14516    0:02 csh
    ttyd3   17393    0:00 whodo

ttyd1    john       9:47
    ttyd1    3146    0:03 csh

console  smith      9:31
    console 15392    0:00 csh

ttyq3    rlogin     9:31
    ttyq3   15997    0:00 csh
    ttyq3   16055    0:00 csh
```

Process Monitoring with ps -ef

- ps -ef

```
 UID   PID  PPID  C    STIME TTY      TIME COMMAND
 root     0     0  0  Jun  7 ?        0:00 sched
 root     1     0  0  Jun  7 ?        2:34 /etc/init
 root     2     0  0  Jun  7 ?        0:01 vhand
 root     3     0  0  Jun  7 ?        0:35 bdflush
smith 15392     1  0 18:09:52 console  0:00 -csh
  joe 14516     1  0 15:01:43 ttyd3    0:02 -csh
 root    50     1  0  Jun  7 ?        0:02 /usr/etc/syslogd
 root    68     1  0  Jun  7 ?        0:59 /usr/etc/routed
 root    72     1  0  Jun  7 ?        0:13 /usr/etc/portmap
 root    71     1  0  Jun  7 ?        0:13 /usr/etc/inetd
 root    76     1  0  Jun  7 ?        0:03 /usr/etc/ypbind
 john  3146     1  0  Jun  8 ttyd1    0:03 -csh
   lp   129     1  0  Jun  7 ?        0:03 /usr/lib/lpsched
 root   137     1  0  Jun  7 ?        0:13 /etc/cron
 root   144     1  0  Jun  7 ?        0:01 /usr/lib/sendmail -bd -q15m
  joe 17402 14516 18 12:31:52 ttyd3    0:00 ps -ef
```

the motd File

- message stored in /etc/motd
- displayed when user logs in
- reserve for important (and short) messages

the news Facility

- files stored in /usr/news
- maintained by system administrator
- the news command:

  news (read unread news)

  news -a (read all news)

  news -n (number of unread news)

UNIX Process Scheduler cron

- system daemon wakes up every minute
- checks control file for jobs to execute
- jobs execute regularly at specified interval
- facility controlled by system administrator
- entries stored in /usr/spool/cron/crontabs
- users can add entries with *crontab* command

cron Administration

/usr/lib/cron

- users given access via entry in *cron.allow* or cron.deny
- one crontab file per user
- cron activity recorded in */usr/lib/cron/log*
- crontab command:

  crontab *file* (places user's file in cron table)

  crontab -r (removes user's file from cron table)

  crontab -l (lists contents of user's file)

- crontab format:

  minute hour day_of_month month day_of_week event

the Login Process

Objectives:

- identify the processes involved
- identify the files involved
- describe sequence of events

the Function of Login

- prompt the user
- validate the user
- start up the session

csh, sh or program
put in home directory

- initialize the terminal environment

the Processes

init spawns the *getty*

getty:

- displays a prompt
- opens port for terminal communication
- reads login id
- execs *login* and passes login id

login

- prompts for password and validates
- execs the login shell, user's config files

user's config files initialize terminal

the Files

- */etc/inittab* contains entries for ports:
  - which ports are active (no "x" in second field)
  - which process to spawn (getty)
  - which line to open (tty)
  - pointer to */etc/gettydefs* for the port
- */etc/gettydefs* defines baud rate, line conditioning, prompt
- */etc/ttytype* defines terminal type
- */etc/terminfo* defines communication for term types
- */etc/passwd* contains user login information

the Login Sequence

init to login

- init spawns getty on ports defined by inittab
- inittab entries direct getty to proper gettydef line
- getty opens port, using line characteristics and prompt from gettydefs
- user responds to prompt
- getty execs login, with user name as argument

the Login Sequence

login to user shell

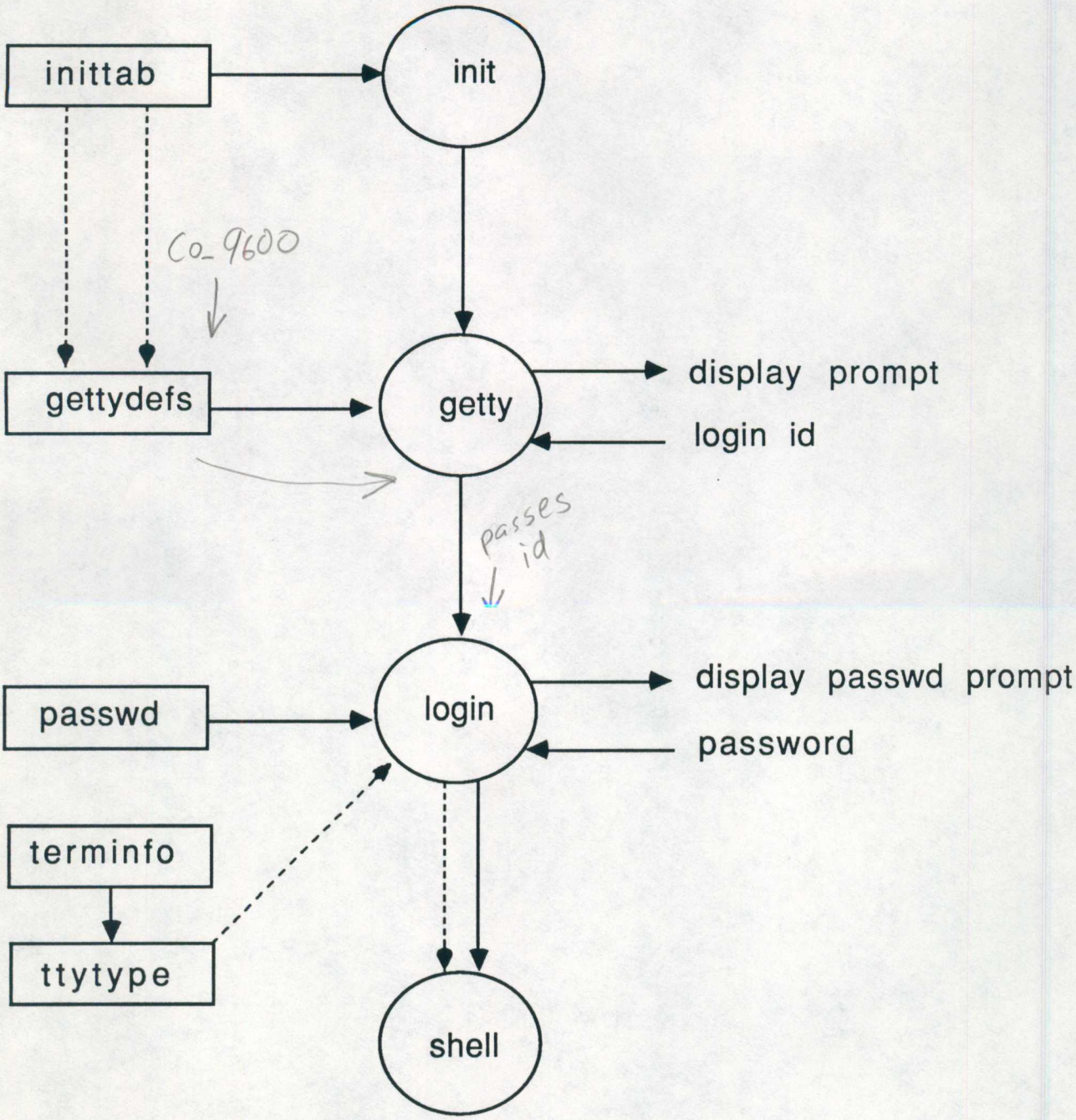
- login prompts for password
- user responds, login verifies with /etc/passwd
- login execs user login shell defined in /etc/passwd
- user put in home directory defined in /etc/passwd
- user's config files run, set up environment, terminal

Login Process Lab I

Objectives:

- examine /etc/inittab
- step through login process
- monitor processes

note - getty and gettydefs details covered in Terminal Support section


inittab
init
Co_9600
gettydefs
getty
display prompt
login id
passes id
passwd
login
display passwd prompt
password
terminfo
ttytype
shell

Terminal Support

Objectives:

- overview of terminal devices
- terminal setup procedure
- terminal communication

Terminal Device Overview

communication requires:

- physical connection
- proper cable configuration
- cpu interface
- software interface
- terminal device files

Terminal Device Files

- ttyd1-12 (RS 232 port devices)
- ttym1-12 (RS 232 port modem devices)
- ttyq1-99 (virtual terminals for ethernet)
- syscon (system console)
- systty (system tty)
- console (console)

note: syscon, systty, are identical!

Terminal Setup Procedure

- make physical connection
- modify */etc/inittab* to enable the port
- modify */etc/gettydefs* to specify baud rate, prompt and port configuration (optional)
- modify */etc/ttytype* to specify terminal type
- optionally, set user login file
- update init with *telinit q*

Make Physical Connection

- simplified RS-232 "null modem" cable
- pin configurations:

IRIS	Terminal	Signals
2	3	Transmit data
3	2	Receive data
7	7	Signal ground

Modify /etc/inittab

- su to root, cd to /etc
- vi inittab
- remove *x* from port entries
- modify gettydef pointer (if required)
- write and quit vi (:wq!)

```
co:234:respawn:/etc/getty console console none LDISC0# the console
t1:234:respawn:/etc/getty -s console ttyd1 co_9600 none LDISC0# alt cons
t2:x:respawn:/etc/getty ttyd2 co_9600 none LDISC0# port 2
t3:x:respawn:/etc/getty ttyd3 co_9600 none LDISC0# port 3
#
```

Make Physical Connection

- RS-232 "null modem" cable
- pin configurations:

IRIS	Terminal	Signals
1	1	Chassis ground
2	3	Transmit data
3	2	Receive data
4	8	Request to send, Clear to send
8	4,5	Carrier detect
6	20	Data set ready
20	6,22	Data terminal ready
7	7	Signal ground

Modify /etc/inittab

- su to root, cd to /etc
- vi inittab
- remove *x* from port entries
- modify gettydef pointer (if required)
- write and quit vi (:wq!)

```
co:234:respawn:/etc/getty console console none LDISC0# the console
t1:234:respawn:/etc/getty -s console ttyd1 co_9600 none LDISC0# alt cons
t2:x:respawn:/etc/getty ttyd2 co_9600 none LDISC0# port 2
t3:x:respawn:/etc/getty ttyd3 co_9600 none LDISC0# port 3
#
```

the getty

"get tty" settings

- sets terminal type, modes, speed, line discipline
- format:

/etc/getty line speed term-type linedisc

```
co:234:respawn:/etc/getty console console none LDISC0# the console
t1:234:respawn:/etc/getty -s console ttyd1 co_9600 none LDISC0# alt cons
t2:x:respawn:/etc/getty ttyd2 co_9600 none LDISC0# port 2
t3:x:respawn:/etc/getty ttyd3 co_9600 none LDISC0# port 3
#
```

- speed points to entry in */etc/gettydefs*

- term-type normally "none"

- line discipline always "LDISC0"

Modify /etc/gettydefs

- su to root, cd to /etc
- vi gettydefs
- set speed cycling sequence
- set system prompt
- write and quit (:wq!)

the /etc/gettydefs File

label : initial flags : final flags : prompt : next-label

```
console# B9600 # B9600 SANE TAB3 #\r\n\n$HOSTNAME console login: #console

co_9600# B9600 # B9600 SANE TAB3 #\r\n\n$HOSTNAME login: #co_4800

co_4800# B4800 # B4800 SANE TAB3 #\r\n\n$HOSTNAME login: #co_2400

co_2400# B2400 # B2400 SANE TAB3 #\r\n\n$HOSTNAME login: #co_1200

co_1200# B1200 # B1200 SANE TAB3 #\r\n\n$HOSTNAME login: #co_300

co_300# B300 # B300 SANE TAB3 #\r\n\n$HOSTNAME login: #co_9600

dx_9600# B9600 # B9600 SANE TAB3 HUPCL #\r\n\n$HOSTNAME login: #dx_9600

dx_4800# B4800 # B4800 SANE TAB3 HUPCL #\r\n\n$HOSTNAME login: #dx_4800

dx_2400# B2400 # B2400 SANE TAB3 HUPCL #\r\n\n$HOSTNAME login: #dx_2400

dx_1200# B1200 # B1200 SANE TAB3 HUPCL #\r\n\n$HOSTNAME login: #dx_1200

du_2400# B2400 # B2400 SANE TAB3 HUPCL #\r\n\n$HOSTNAME login: #du_1200

du_1200# B1200 # B1200 SANE TAB3 HUPCL #\r\n\n$HOSTNAME login: #du_300

du_300# B300 # B300 SANE TAB3 HUPCL #\r\n\n$HOSTNAME login: #du_2400
```

ascii terminals

modem

Hangup line when finished

Modify /etc/ttytype

- su to root, cd to /etc
- vi ttytype
- set terminal type for port
- save and quit (:wq!)

```
iris-ansi       console
iris-ansi       systty
vt100           ttyd1
?vt100          ttyd2
vt100           ttyd3
```

the terminfo Database

- directory of terminal type subdirectories
- files define terminal protocols
- alphanumeric organization (a-z, 1-99)
- terminal referenced by common name

to find a specific terminal:

```
ls -R /usr/lib/terminfo | grep vt100
```

also see:

terminfo(4), term(5), infocmp(1M)

Modify User Configuration Files

stty:

- sets terminal options (erase, kill, etc.)

  (-a option displays terminal settings)

tset:

- looks at TERM to get term type
- references terminfo to get terminal protocol
- initializes terminal

can be used to *set* TERM variable:

```
set tmp=(`tset -S -Q`)
setenv TERM $tmp
```

the Final Steps

- Inform init:

  telinit q

- Test terminal:

  <return> should display prompt

  if unexpected "garbage", press <break>

Terminal Troubleshooting

- check connections
- reset terminal (power off)
- stty sane (stty sane < /dev/ttyd*N*)
- tset
- set terminal settings
- kill terminal process
- may need to kill getty

Resources

command/file	function	manual
console	console interface	Sys Adm Reference (7)
getty	sets terminal settings, prompt	Sys Adm Reference (1M)
gettydefs	defines terminal setting	Progr Reference (4)
infocmp	display terminal protocol	Sys Adm Reference (1M)
stty	set terminal settings	User's Reference (1)
term	defines terminal names	Progr Reference (5)
term	format of compiled term file	Progr Reference (4)
terminfo	terminal protocol database	Progr Reference (4)
termio	terminal interface	Sys Adm Reference (7)
tty	display terminal port assignment	User's Reference (1)
ttytype	assigns terminal types to ports	System Admin Student Workbook

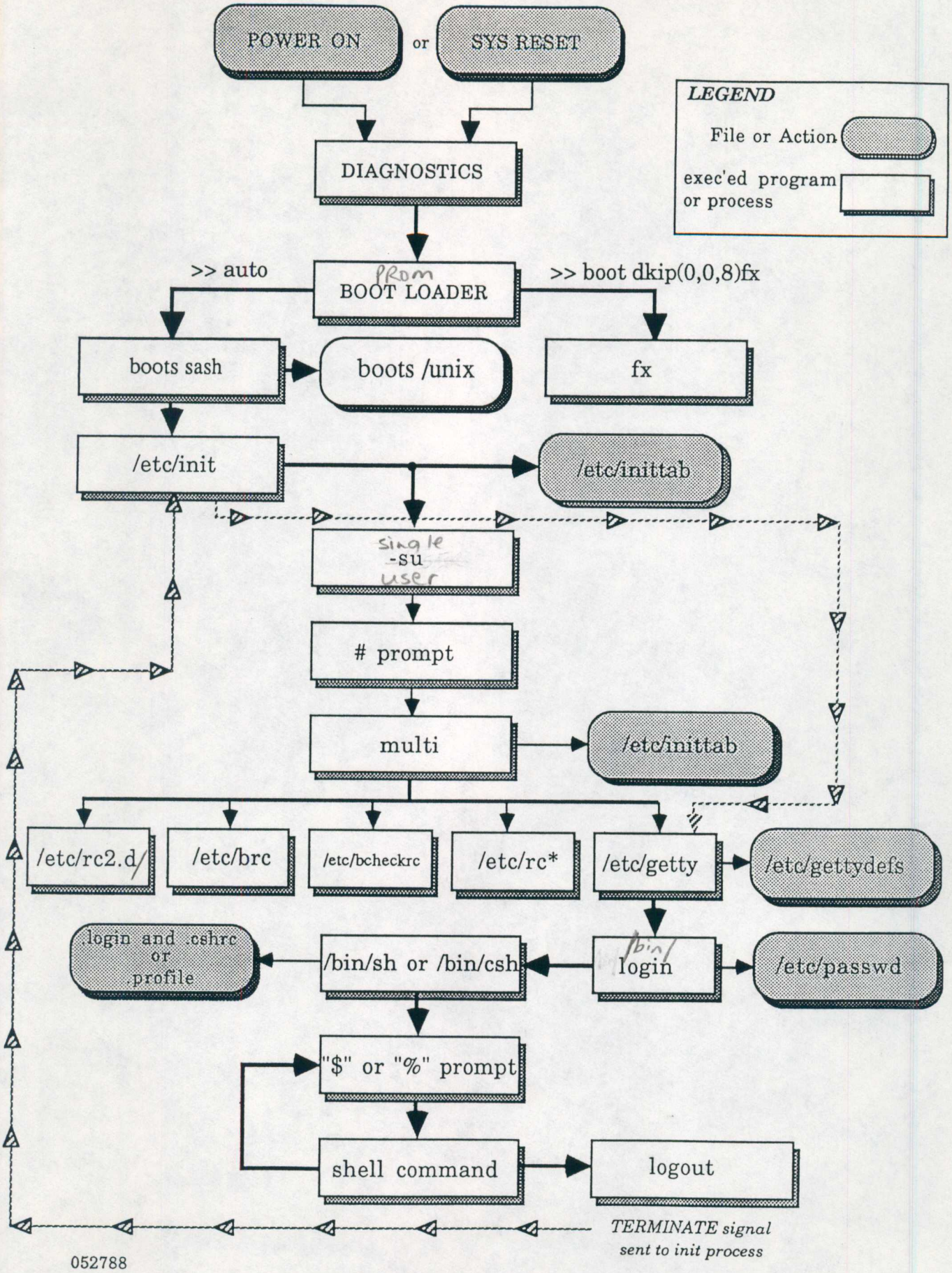
tset

Terminal Support Lab

- set up terminal
- experiment with stty

① edit inittab file
② ttytypefile file

UNIX STATE DIAGRAM



Printer Support

Objectives:

- printer setup procedure
- print spooler configuration

Setting Up a Printer

- change ownership of port device file
- modify inittab file to turn off prompt
- make physical connection
- configure spooler
- set port communication

Change Ownership of Port Device File

- become superuser
- ls -l /dev/ttyd*N*

  NOTE: do not use ttyd1

- chown lp /dev/ttyd*N*
- chgrp sys /dev/ttyd*N*
- chmod 660 /dev/ttyd*N*
- ls -l /dev/ttyd*N*

Modify Inittab to Turn Off Prompt

- su to root, cd to /etc
- vi inittab
- insert *x* in second field
- save and quit (:wq!)
- telinit q

```
t2:x:respawn:/etc/getty ttyd2 co_9600 none LDISC0# port 2
```

Connect Printer to Port

- connect to ports 2, 3 or 4
- simplified rs-232 "null modem" cable:

IRIS	Printer	Signals
2	3	Transmit data
3	2	Receive data
7	7	Signal ground

the Print Spooler

- allows many users to share printer(s)
- increases productivity
- flexible queueing of print requests
- ability to monitor, cancel print requests

Connect Printer to Port

- connect to ports 2, 3 or 4
- rs-232 "null modem" cable:

IRIS	Printer	Signals
1	1	Chassis ground
2	3	Transmit data
3	2	Receive data
4	8	Request to send, Clear to send
8	4,5	Carrier detect
6	20	Data set ready
20	6,22	Data terminal ready
7	7	Signal ground

the Print Spooler

- allows many users to share printer(s)
- increases productivity
- flexible queueing of print requests
- ability to monitor, cancel print requests

Spooler Terminology

- spooler - print scheduler, controls flow of data to printer(s)
- device - physical printer, one per queue
- queue - logical pipeline for print requests
- class - group of printers or queues into one logical device

Spooler Commands

- lpadmin - configure the spooler
- lpsched - turn on line printer spoooler
- lpshut - turn off line printer spooler
- enable - enable printer for use by scheduler
- disable - disable printer from use
- accept - restart a queue
- reject - shut off queue
- lp - send a print file to the spooler
- cancel - remove a print request from the queue
- lpstat - display status of print request

Configure Spooler

- stop scheduler: /usr/lib/lpshut
- create the queue:

  /usr/lib/lpadmin -p*queue* -v*port* -m*filter*

- assign a default destination:

  /usr/lib/lpadmin -d*queuename*

- start the queue: /usr/lib/accept *queue*
- enable queue: /usr/lib/enable *queue*
- turn on scheduler: /usr/lib/lpsched

Set Port Communication

- stty 9600 ixon -onlcr < /dev/ttyd*N*

Adding Print Destinations

- stop scheduler: /usr/lib/lpshut
- create the queue:

  /usr/lib/lpadmin -p*queue* -v*port* -m*filter*

- start the queue: /usr/lib/accept *queue*
- enable the queue: /usr/lib/enable *queue*
- turn on scheduler: /usr/lib/lpsched

Deleting Print Destinations

- stop scheduler:

  /usr/lib/lpshut

- delete the queue:

  /usr/lib/lpadmin -x*queue*

- turn on scheduler:

  /usr/lib/lpsched

Defining Print Classes

- stop scheduler:

  /usr/lib/lpshut

- create class, assign queue(s):

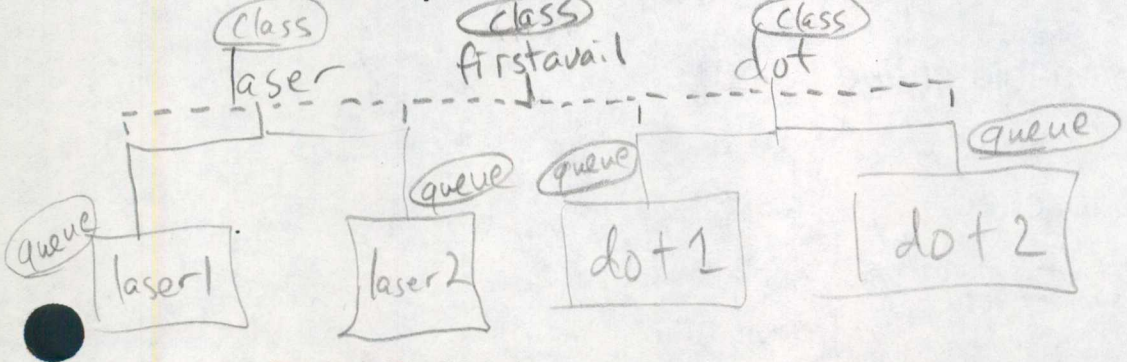
  /usr/lib/lpadmin -p*queue1* -c*class*

  /usr/lib/lpadmin -p*queue2* -c*class*

- turn on scheduler:

  /usr/lib/lpsched

note: a queue can be assigned to multiple classes



Defining Interface Programs

interface program examples in /usr/spool/lp/model

/usr/spool/lp/interface

- binary code or shell script
- "massage" data (print banner, troff, etc.)
- prepare output for different model printers
- assigned to a queue, which uses a particular device:

/usr/lib/lpshut

/usr/lib/lpadmin -p*queue1* -v*port* -i*lprogram*
-d model

/usr/lib/accept *queue*

/usr/~~lib~~bin/enable *queue*

/usr/lib/lpsched

the lpstat Command

options:

- -u *user* (status of output request for *user*)
- -d (lists default destination for *lp*)
- -v (lists destinations and associated devices)
- -t (all status information)

```
%lpstat -t
scheduler is running
system default destination: apple
members of class PostScript:
  apple
device for apple: /usr/spool/lp/transcript/apple-log
apple accepting requests since May 10 07:34
PostScript accepting requests since May 10 07:34
printer apple now printing PostScript-1740.  enabled since May
PostScript-1740 joe   13313   Jun 14 11:48 on apple
%
```

Printer Support Lab

Objectives:

- set up ascii terminal as printer
- configure print spooler

/usr/spool/lp/SCHEDLOCK

```
# #ident "@(#)lp:model/dumb     1.2"
#ident "$Header: /jake/att/usr/src/cmd/lp/model/RCS/dumb,v 1.2 86/10/07 10:53:14 mc
# lp interface for dumb line printer
#

x="XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX>
echo "\014\c"
echo "$x\n$x\n$x\n$x\n"
banner "$2"
echo "\n"
user=`grep "^$2:" /etc/passwd | line | cut -d: -f5`
if [ -n "$user" ]
then
        echo "User: $user\n"
else
        echo "\n"
fi
echo "Request id: $1    Printer: `basename $0`\n"
date
echo "\n"
if [ -n "$3" ]
then
        banner $3
fi
copies=$4
echo "\014\c"
shift; shift; shift; shift; shift
files="$*"
i=1
while [ $i -le $copies ]
do
        for file in $files
        do
                cat "$file" 2>&1
                echo "\014\c"
        done
        i=`expr $i + 1`
done
echo "$x\n$x\n$x\n$x\n$x\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n"
echo "\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n$x\n$x\n$x\n$x"
echo "$x\n$x\n$x\n$x\n$x\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n"
echo "\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n"
echo "$x\n$x\n$x\n$x\n$x\n$x\n$x\n$x\n$x\n"
exit 0
```

Field	Contents
$0	Printer Name
$1	Request or Sequence ID.
$2	User ID.
$3	Title
$4	# of copies
$5	(options) (Banner Page) etc.
$6-7	Filenames to be printed.

```
# System V line printer spooler model for PostScript/TranScript printer
# Copyright (c) 1985 Adobe Systems Incorporated
# PostScript and TranScript are trademarks of Adobe Systems Incorporated
# RCSID: $Header: /jake/sgi/usr/src/usr.bin/print/./lp.spool/lib/RCS/psinterface,v

# stty options are:
#       stty cs8 9600 cread -clocal -ignbrk brkint -parmrk \
#       inpck -istrip -inlcr -igncr -icrnl -iuclc ixon -ixany ixoff \
#       -opost -isig -icanon -xcase
#       -echo -echoe -echok -echonl min \^a time \^d
#
# on 3B2 Sys Vr2v2 this is "1412:0:4bd:0:7f:1c:23:40:1:4:0:0"

sttyopts="1416:1804:cbd:0:7f:1c:8:15:1:4:0:0"

# establish basics
# the log files is also our stdout and stderr file (set up with lpadmin)
#       argv[0] is interface/PRINTERNAME
#       cwd is the spooler directory (/usr/spool/lp)
#       our environment gives us fairly little info though lots is passed in

prog=$0
pr=`basename $prog`
printer=/dev/$pr
cwd=`pwd`
ptime=`date`
log=$cwd/transcript/${pr}-log
if [ ! -w "$log" ] ; then
        disable -r"can't access log file $log" $pr 1>/dev/console 2>&1
        exit 1
fi
# set these however you want, set psrv to null if you don't want reversal
PSLIBDIR=/usr/lib/ps
send=$PSLIBDIR/pscomm
isend=/usr/lib/print/pprint
banner=$PSLIBDIR/psbanner
psrv=$PSLIBDIR/psrv
format=$PSLIBDIR/pstext
BANNERPRO=$PSLIBDIR/banner.pro
BANNERFIRST=0
BANNERLAST=1
REVERSE=1
PSTEMPDIR=/usr/tmp
export BANNERPRO BANNERFIRST BANNERLAST PSTEMPDIR REVERSE PSLIBDIR

# printer-specific options file (can change any of the above)
test -r ./transcript/${pr}.opt && . ./transcript/${pr}.opt

if [ "$REVERSE" != "1" ] ; then
        psrv=
fi

# parse command line options (cannonical)
seqid=$1
name=$2
```

```
title="$3"
copies=$4
options="$5"
shift; shift; shift; shift; shift
files="$*"
if [ -z "$title" ] ; then
        title=`basename $1`
fi

for j in $options
do      case $j in
        r)      erase=$j;;
        erase)  erase=$j;;
        *)      ;;
        esac
done

# parse TranScript-specific user options with getopt
set -- `getopt hmr "$options"`
if [ $? != 0 ] ; then
        echo $pr: $seqid bad user options $options
        exit 2
fi
Hflag= Mflag= Rflag=
for i in $*
do      case $i in
        -h|h)   Hflag=$i; shift;;       # no banner page
        -m|m)   Mflag=$i; shift;;       # mail stream output if any
        -r|r)   Rflag=$i; shift;;       # never reverse
        --)     shift ;;
        esac
done

# set up to send the job
if [ ! -x $send ] ; then
        disable -r"can't execute $send filter" $pr
        exit 1
fi
if [ ! -x $isend ] ; then
        disable -r"can't execute $isend filter" $pr
        exit 1
fi
echo $pr: $seqid $name "$title" start - $ptime

# create banner page
if [ -z "$Hflag" -a \( "$BANNERFIRST" = "1" -o "$BANNERLAST" = "1" \) ] ; then
        if [ ! -x $banner ] ; then
            disable -r"can't exec $banner program" $pr
            exit 1
        fi
        bannerf=$PSTEMPDIR/b$seqid.$$
#       trap "rm -f $bannerf" 1 2 3 15
        if [ ! -r $BANNERPRO ] ; then
                disable -r"can't access banner prolog" $pr
                exit 1
        fi
```

```
        $banner $pr $seqid $name "$title" "$ptime" >$bannerf
fi

# print banner page first ?
if [ -z "$Hflag" -a "$BANNERFIRST" = "1" ] ; then
        (stty $sttyopts <&1
        $send <$bannerf
        stty < $printer > /dev/null
        ) 1>$printer 2>>$log 3<$printer
fi

# set up to mail job output if user flag set
if [ -n "$Mflag" ] ; then
        JOBOUTPUT=${PSTEMPDIR}/o$seqid.$$
        export JOBOUTPUT
fi

# now process the print files: first the sgi-style possibilities
flist=
curmap=
for f in $files
do
   srch='strings $f'
   set 'dd bs=1 count=4 skip=104 if=$f 2>/dev/null | od -b'
   case $5 in
        003) curmap=$f ;;
        *) flist="$flist $f" ;;
   esac
done

# now print any sgi files
for f in $flist
do

    echo $pr: $seqid 'basename $f' - 'date'
    cop=$copies
#   check for magic number: see if it is an image file
    set 'dd bs=1 count=2 < $f 2> /dev/null | od -h'
    if [ "$2" = 01DA -o "$2" = DA01 -o "$2" = 01da -o "$2" = da01 ] ; then
        while [ $cop -ge 1 ]
        do   (stty $sttyopts <&1
                $isend $f $curmap | $send
                stty < $printer > /dev/null
             ) 1>$printer 2>>$log 3<$printer
             cop='expr $cop - 1'
        done

#   Transcript files: check for magic number and format if plain text

    else
        magic='line <$f | cut -c1-11'
        rev= tfile=
        case "$magic" in
            %!PS-Adobe-) rev=1 ;;
            %!*) ;;
            *)  tfile=$PSTEMPDIR/t$seqid.$$ ;
```

```
                $format <$f >$tfile ;
                f=$tfile ; rev=1 ;;
        esac

        # if multiple copies, reverse only once
        if [ $copies -gt 1 -a -n "$rev" -a -n "$psrv" -a -z "$Rflag" ] ; then
            rfile=$PSTEMPDIR/r$seqid.$$
            $psrv <$f >$rfile
            f=$rfile ; rev=
        fi

        # print all the copies, reversing as necessary
        while [ $cop -ge 1 ]
        do  (stty $sttyopts <&1
             if [ -z "$rev" -o -z "$psrv" -o -n "$Rflag" ] ; then
                    $send <$f
                    stty < $printer > /dev/null
             else
                    $psrv <$f | $send
                    stty < $printer > /dev/null
             fi ) 1>$printer 2>>$log 3<$printer
            cop=`expr $cop - 1`
        done
        rm -f $rfile $tfile
    fi
done


if [ -n "$erase" ] ; then
    rm -f $files
fi


# print banner page last ?
if [ -z "$Hflag" -a "$BANNERLAST" = "1" ] ; then
        (stty $sttyopts <&1
        $send <$bannerf
        stty < $printer > /dev/null
        ) 1>$printer 2>>$log 3<$printer
fi

# mail user the job ouput if flag set
if [ -n "$Mflag" ] ; then
        if [ -s "$JOBOUTPUT" ] ; then
                (echo Subject: output from PostScript print job $seqid follows
                cat $JOBOUTPUT ) | mail $name
        fi
        rm -f $JOBOUTPUT
fi

echo $pr: $seqid end - `date`

# clean up
rm -f $bannerf
```

`cancel` *id number*

and it is used this way:

```
$ cancel epson-4857
request "epson-4857" cancelled
$
```

To fully illustrate the usefulness of the `cancel` command, let's say you accidentally sent an executable binary file to the printer rather than its documentation. If you don't cancel the request, the printer goes into a schizophrenic fit and prints a paper box full of blanks, odd characters, and graphics! `cancel` arms the average user with the necessary tool to stop any undesirable printing process. The system administrator needs more sophisticated tools, so let's take a look inside the `lp` system.

How the `lp` Printer Spooler Works

The `lp` system is relatively complex, and to understand it, some form of analogy is helpful. UNIX has often been compared to a set of plumbing devices, and it's an apt comparison, particularly because we are dealing with a character stream that flows throughout the entire system. Please refer to Fig. 11-1.

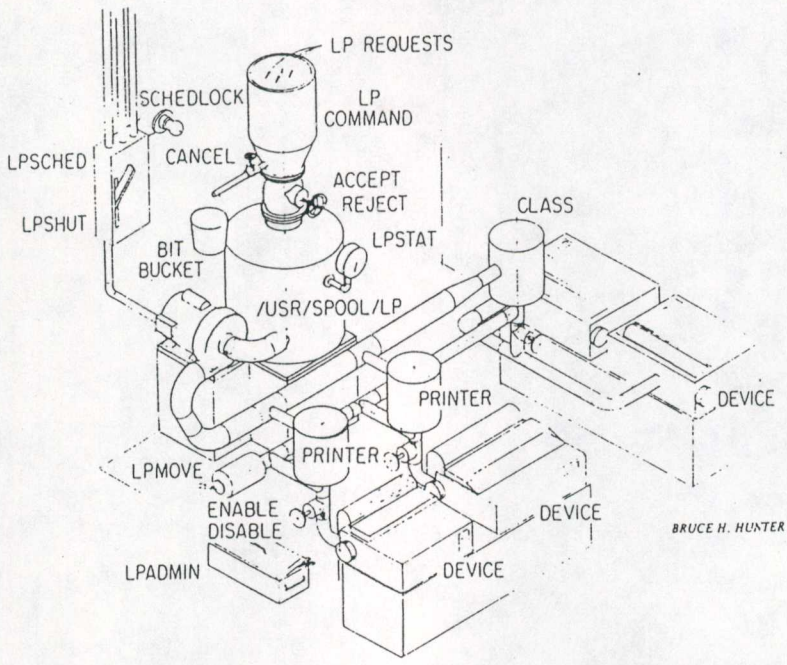


Fig. 11-1. The System V `lp` Spooler System

Disk Configuration and Maintenance

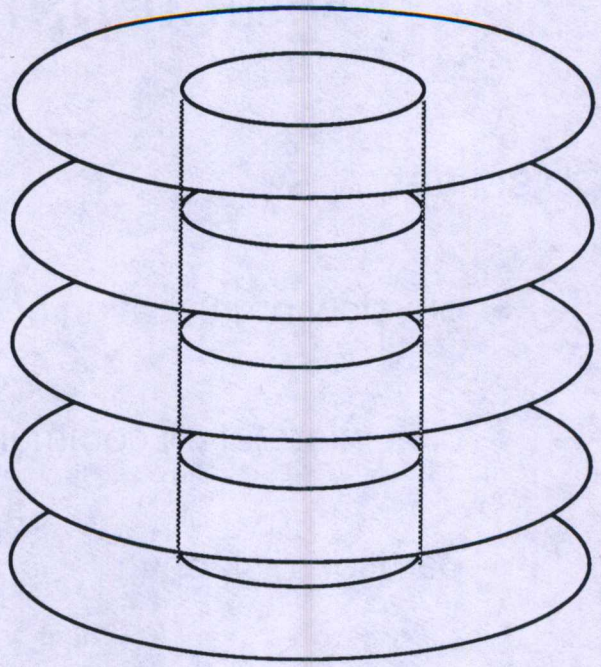
Objectives:

- physical organization of disk
- disk information commands
- partition disk drive
- add a second disk

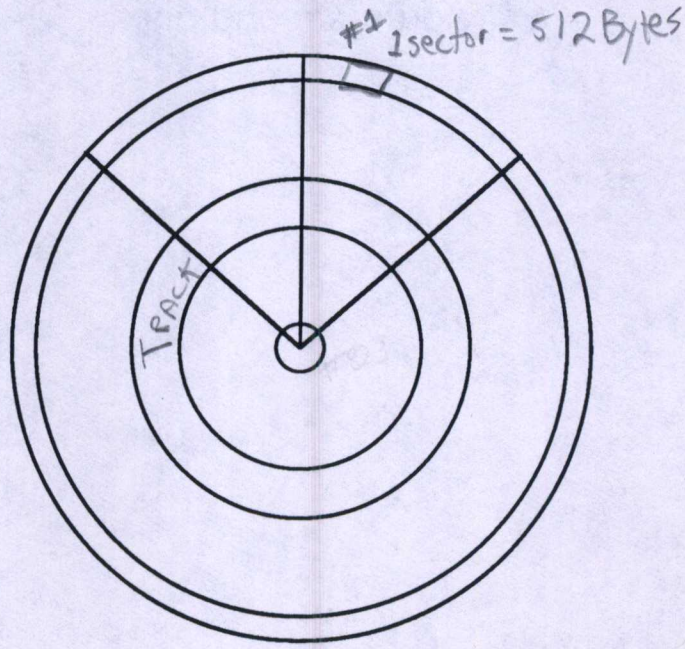
Disk Structure

- platters
- tracks
  823 per disk
- cylinders = same track on all 10 surfaces
- sectors
- block

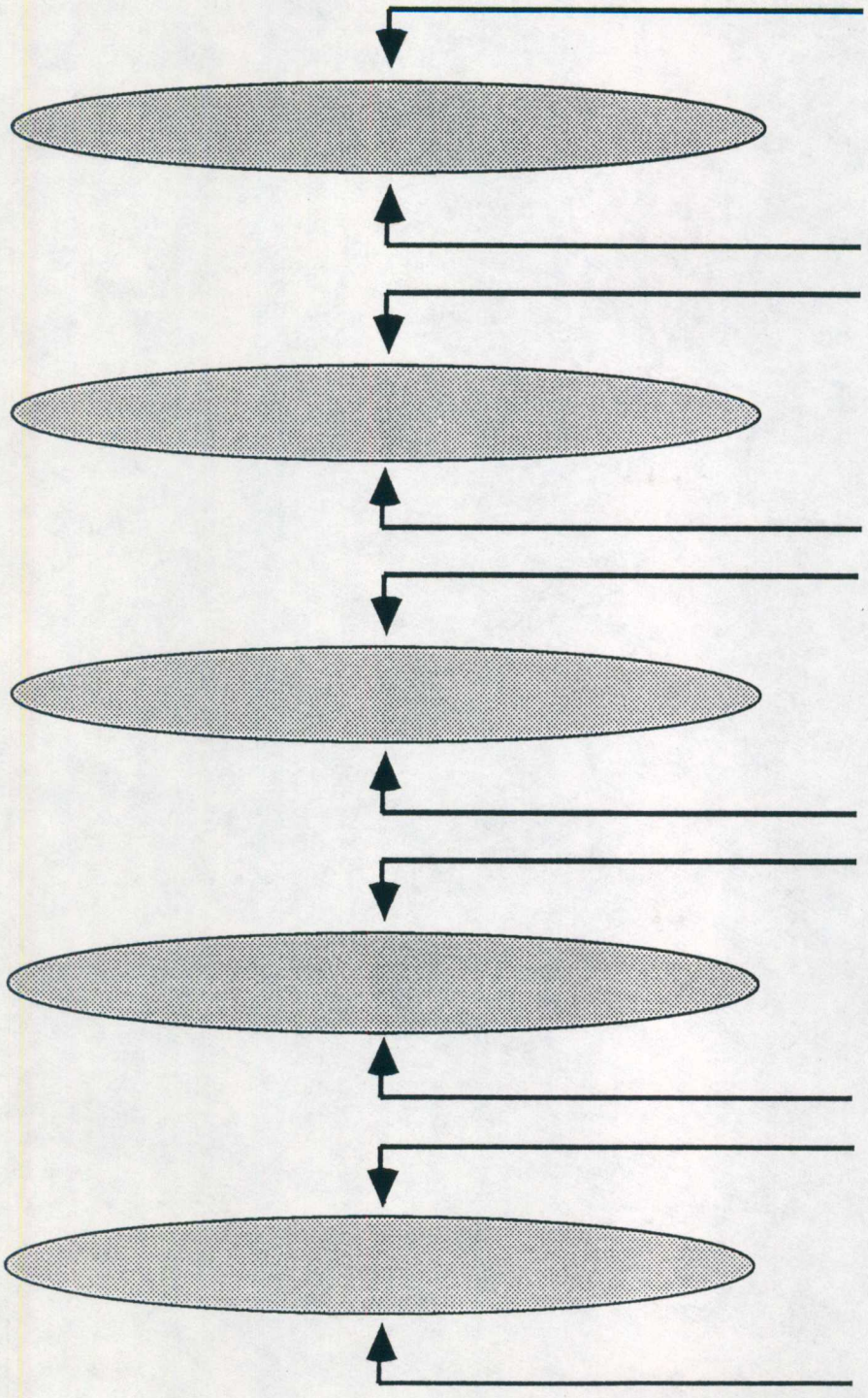
5 12 Byte.



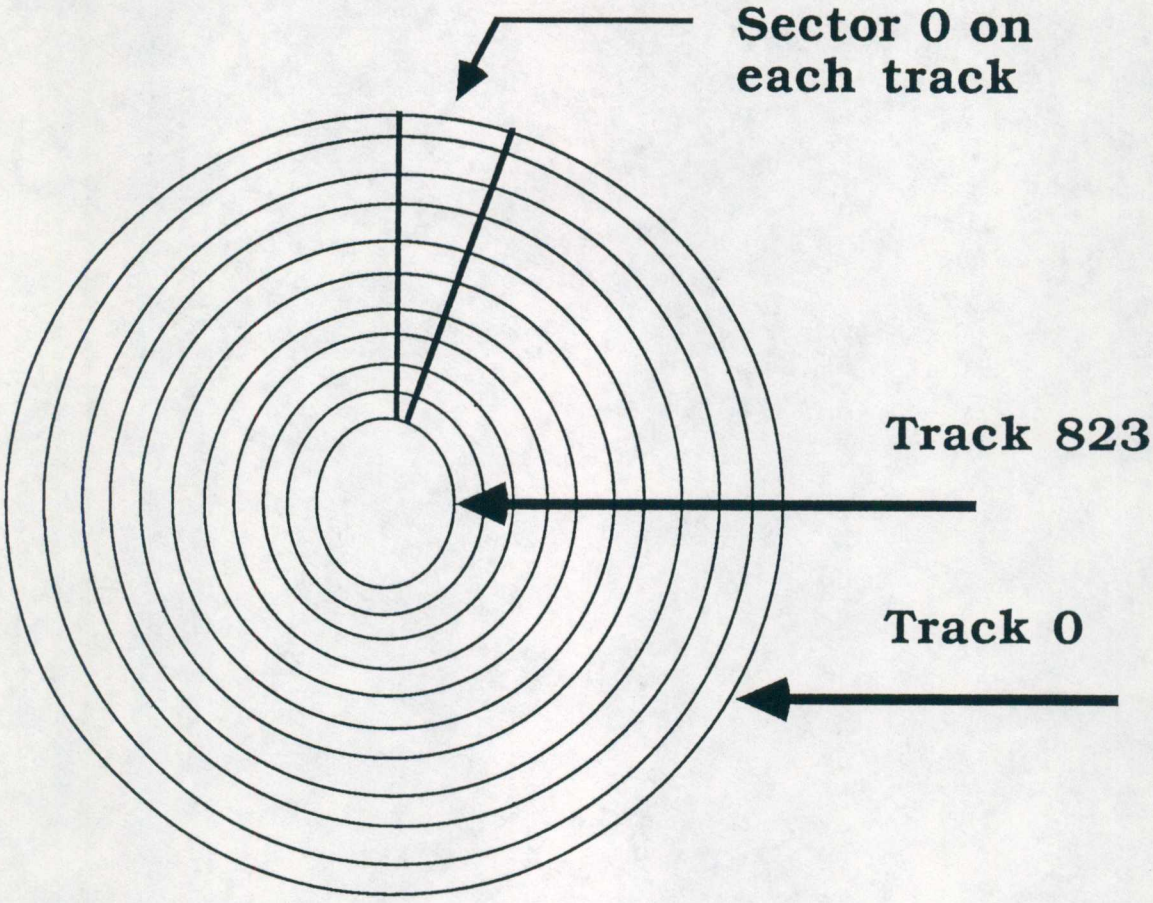
- partitions
  group of cylinders



Platter Diagram



Platter Diagram



Block/sector = 512 bytes

32 sectors = 1 track

823 tracks = 1 platter side

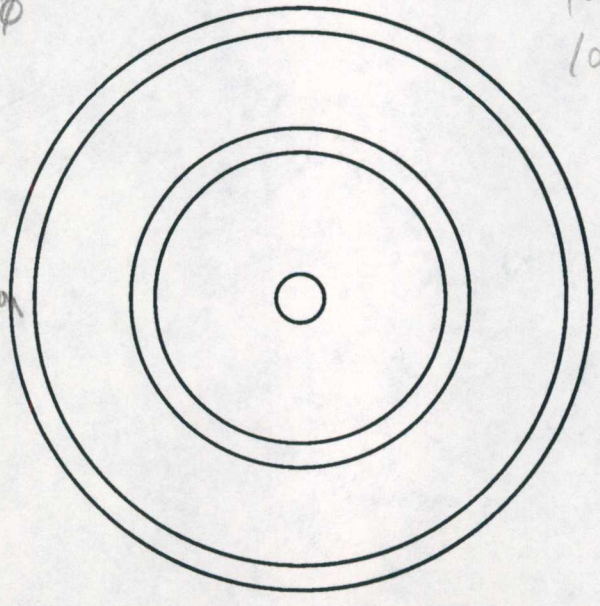
Disk Partitions

- group of cylinders
- assigned specific functions
- separate file structure data from raw data
- organization and security of data
- accessed via separate device files

ESDI → /dev/dsk/ip s ø d ø s ø

SCSI → /dev/dsk/d k s ø d 1 s ø (drive, controller, partition)

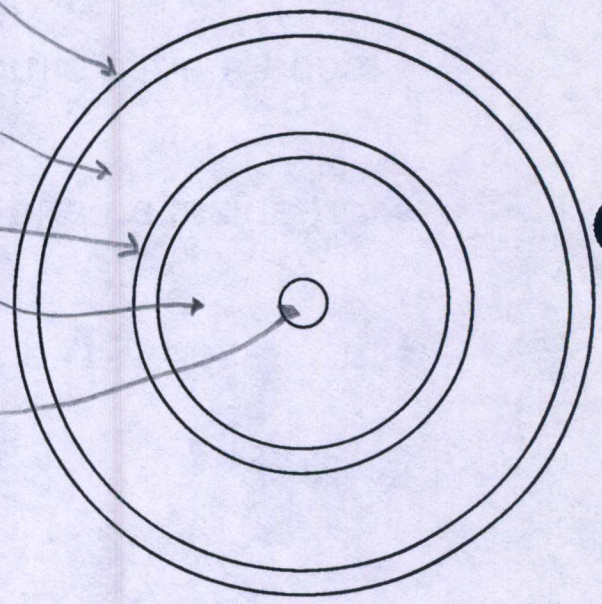
/dev/xyø9
/dev/xyød
/dev/xyøg etc



Available Disk Partitions

(different device file for each)

- vh (volume header) is s8, but not refered to as such
- s0 (root file system)
- s1 (swap space)
- s2, s5, s6 (usr file system)
- s9 (alternate tracks - ESDI)
- s7 (second disk)
- s10 (entire disk)

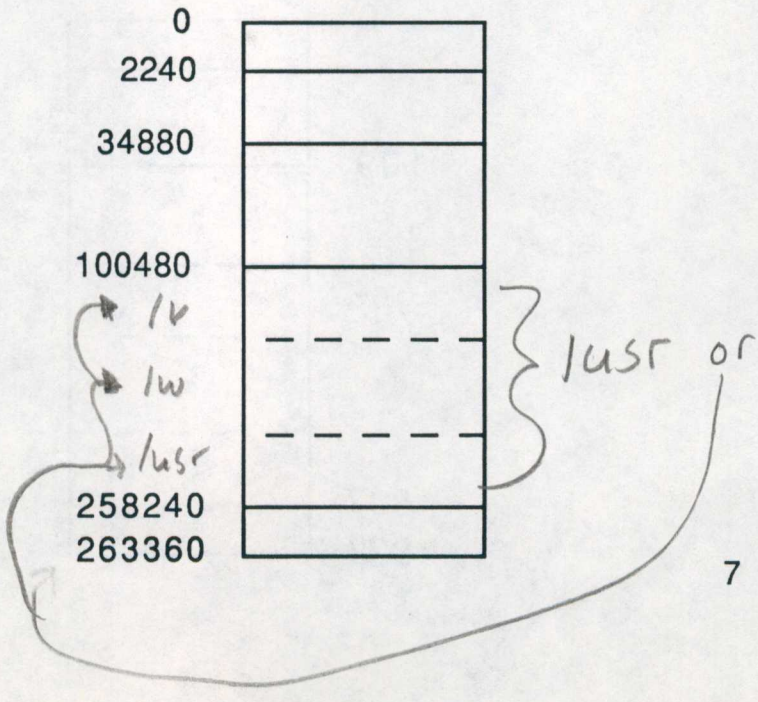


Partition Summary

by sector

Partition	Name	Type	What	Starting Sector	Sector Count	Bytes
i	vh	volhdr	volhdr	0	2240	
a	s0	sysv	root	2240	32640	16711680
b	s1	rdata	swap	34880	65600	
c	s2	sysv	usr	100480	157760	80773632
f	s5	sysv	usr	100480	157760	80773632
g	s6	sysv	usr	100480	157760	80773632
j*	s9	trkrepl	trkrepl	258240	5129	
h	s7	sysv	disk	2240	25600	13107200
k	s10	alldisk	alldisk	0	263360	134840320

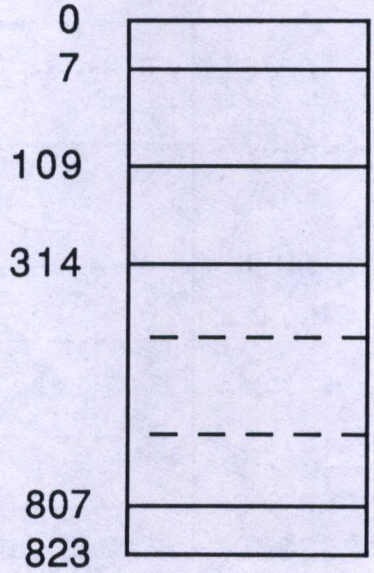
* ESDI only



Partition Summary

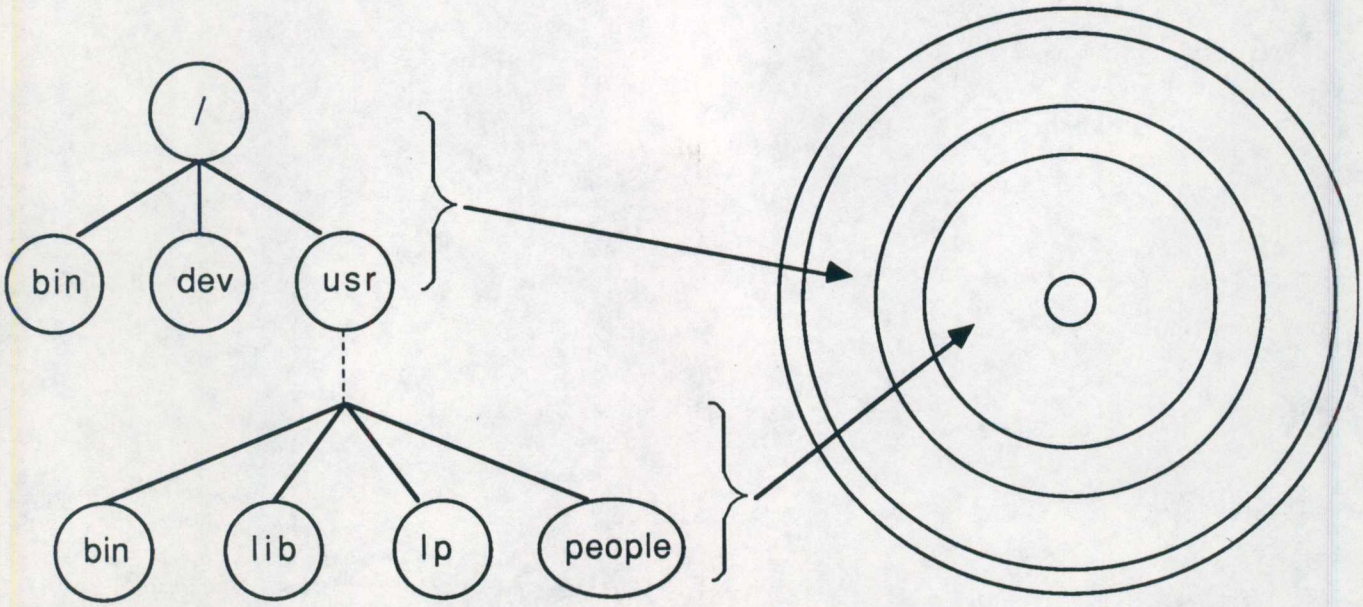
by cylinder used by ex

Partition	Name	Type	What	Starting Cylinder	Cylinder Count	Bytes
i	vh	volhdr	volhdr	0	7	
a	s0	sysv	root	7	102	16711680
b	s1	rdata	swap	109	205	
c	s2	sysv	usr	314	493	80773632
f	s5	sysv	usr	314	493	80773632
g	s6	sysv	usr	314	493	80773632
j*	s9	trkrepl	trkrepl	807	16	
h	s7	sysv	disk	7	800	13107200
k	s10	alldisk	alldisk	0	823	134840320



Disk Partitions and File Systems

- root and usr file systems physically separate
- s0 (root file system)
- s2, s5, s6 (usr file system)
- usr "mounted" at /usr



the Volume Header

- special partition on each disk
- contains *sash*, *sgi label*, *fx*
- contains bad block table
- contains *partition* information

Reading the Volume TOC

- prtvtoc /dev/dsk/dks0d1vh (SCSI)

"Print Volume Table Of Contents"

- prtvtoc /dev/dsk/ips0d0vh (ESDI)

```
* /dev/dsk/dks0d1vh (bootfile "/unix") partition map
*
* Dimensions:
*     512 bytes/sector
*      35 sectors/track
*      10 tracks/cylinder
*     825 cylinders
*     819 accessible cylinders
* Unallocated space:
*  Start      Size
*      288750      -286650
*      288750-182000
*      288750   -182000
*
*   Partition  Tag  Flags  First Sector  Sector Count   Mount Directory
        0      5     0        2100          32550
        1      3     0       34650          72100
        2      5     0      106750         182000
        5      5     0      106750         182000
        6      5     0      106750         182000
        7      5     0        2100         286650
        8      0     0           0           2100
       10      6     0           0         288750
```

Reading the Volume Header with dvhtool

options:

- vd (list volume header files, start blocks, length)
- pt (list partitions, start blocks, length and type)
- dp (list controller information)
- q (quit and return to shell)

su
dvhtool

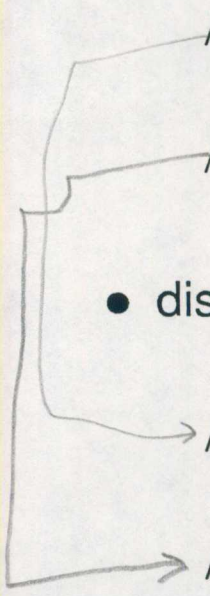
```
Command? (read, vd, pt, dp, write, bootfile, or quit): r
Volume? /dev/dsk/ips0d0vh

Command? (read, vd, pt, dp, write, bootfile, or quit): pt
```

<cr> to return to main prompt
q to quit

Device Assignments

- file system accessed via device file



- disk partition accessed via device file

- file system device and partition device linked

Partition and File System Device Files

- character and block device files for root and usr
- links to partition character and block device files:

  /dev/usr --------> /dev/dsk/dks0d1s6

  /dev/rusr --------> /dev/rdsk/dks0d1s6

  /dev/root --------> /dev/dsk/dks0d1s0

  /dev/rroot --------> /dev/rdsk/dks0d1s0

- confirm device assignment by comparing *inodes*

  ls -i /dev/usr gives inode # → or do ls -l

  ncheck -i*number* ls -i /dev/dks/* | grep (inode #)

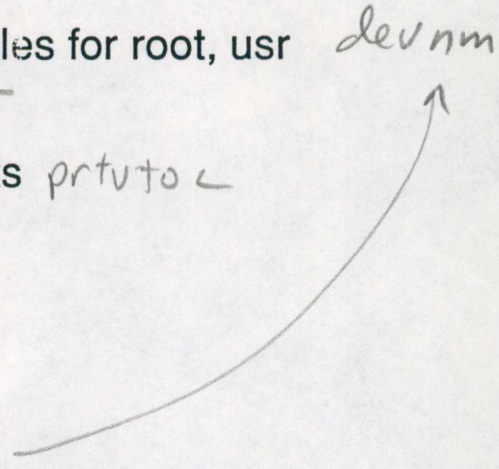
  or use:

  /etc/devnm /usr

Disk Information Lab

Objectives:

- identify system disk device files for root, usr
- read volume table of contents
- read volume header
- confirm device assignments

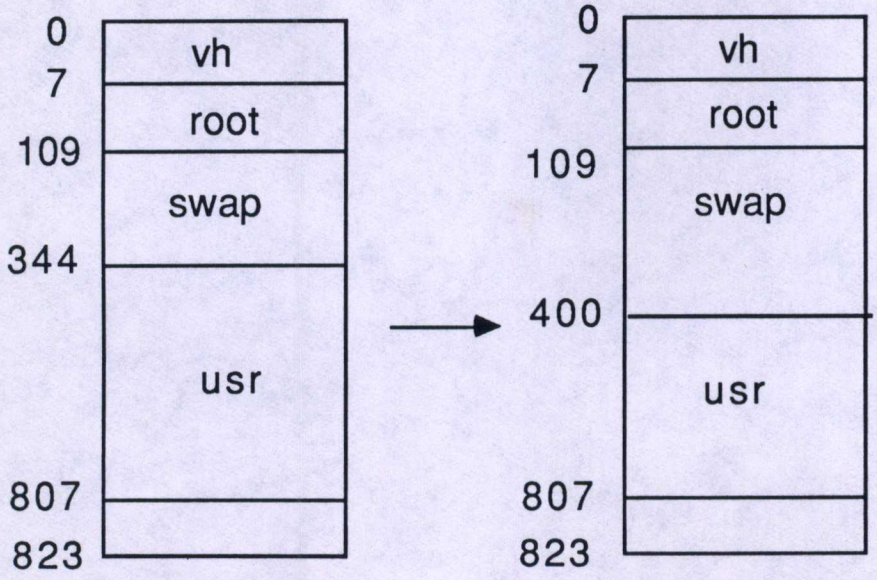


Changing Disk Partitions

- system runs out of swap space more room for virtual memory
- need more room in *root*
- need more room in *usr*
- segment *usr* into multiple file systems
- prepare second disk drive

Disk Partition Rules and Recommendations

- increasing one partition reduces another
- plan partition changes carefully
- always back up entire file system first
- after repartitioning, restore file system
- repartitioning root requires special recovery



Disk Partition Procedure

Overview:

- gather current partition info
- plan future partitions
- back-up file systems (if existing)
- boot *fx*
- use *fx* to verify, reconfigure and verify again.
- update system files (fstab)
- remake, reload file systems
- update system logbook

fx Overview

- formatting disk (erases all data, including label)
- exercise (find bad spots)
- restore (after format, to create root file system)
- badblock (recreate bad block table)
  – ESDI only –
- label:

  create - initialize section

  read - read in section from disk to memory

  set - change section

  show - print section image from memory

Booting fx

- from distribution tape:

  boot -f tpsc(0,7,0)fx.IP4 (SCSI tape)
  boot -f tqic()fx.R2300 (MBC with VME-QIC tape)
  boot -f tqic()fx.IP4 (SBC with VME-QIC tape)

  tpqic

- from disk:

  boot -f dksc(0,1,8)fx (SCSI disk)
  boot -f dkip(0,0,8)fx (ESDI disk)

- over network:

  boot -f bootp()*host*:/stand/fx

- from remote tape:

  boot -f bootp()*host*:/dev/tape(fx[.IP4])

fx prompts

dkip ?
dksc ?

- select the controller:

  fx: ctrl# = (0) <cr> for default, or select

- select the drive:

  fx: drive# = (0) <cr> for default, or select

- select drive type:

  fx: drive-type = (Hitachi 512-17) <cr> for default...

fx Directory Structure

fx

auto (label creation - SGI only)
badblock/ (maintain bad block table)
debug/ (read, write, dump data)
exercise/ (exercise the disk)
format (format the disk)
label/ (change partitions)
restore (create initial fs)
exit (return to *prom*)

label:

sync (updates label in cyl 0)
create/ (create SGI label)
readin/ (read label into memory)
set/ (change disk label)
show/ (display label image)

? for command description

Partitioning Procedures

multi-user to fx

- become superuser
- list current partitions
- confirm device assignments
- follow shutdown sequence
- shutdown to *prom*
- boot *fx*
- select appropriate controller, drive
- at *fx* menu, select *label*

Partitioning Procedures

calculate new partitions

- select *show*
- select *partitions*

  note: fx uses letters for partition names

- note current partition configuration
- calculate new base and size

  base and size are in cylinders

  one cylinder = 10 tracks = 320 sectors = 163840 bytes

- return to *label* (../..)

Sample Partition Configuration

Add together -1

Partition	Name	Type	Base+size	(sectors)
a	s0	sysv	7+102	(32640)
b	s1	rdata	109+205	(65600)
c	s2	sysv	314+493	(157760)
f	s5	sysv	314+493	(157760)
g	s6	sysv	314+493	(157760)
h	s7	sysv	7+800	(256000)
i	vh	volhdr	0+7	(2240)
j*	s9	trkrepl	807+16	(5120)
k	s10	alldisk	0+823	(263360)

2nd Drive →

* ESDI only

Setting Partitions

- select *set*, select *partitions*
- fx displays partition entries, one at a time:

  partition letter (prints current information)
  type (do *not* change)
  base cylinder
  size (in cylinders)
  (prints new information)

- press <cr> for default, or enter new setting
- complete one partition, cycle to next
- cycle through all partitions until back to *set*
- back up to *label*, select *sync*,
- back up to *fx*, *exit*

Disk Partitioning Lab

Objectives:

- boot fx, select drive #2
- display current partition settings show
- calculate new settings →
- repartition disk

increase swap by 50 cyl.
decrease /usr by 50 cyl.

fx Partitioning Worksheet

Partition	Name	Type	Current Base+size	New Base+size	(sectors)
a	s0	sysv	7 + 102	7 + 102	
b	s1	rdata	109 + 205	109 # 255	
c	s2	sysv	314 + 493	364 + 443	
f	s5	sysv	" "	364 + 443	
g	s6	sysv	" "	364 + 443	
h	s7	sysv			
i	vh	volhdr			
j*	s9	trkrepl			
k	s10	alldisk			

DEFAULT DISK LAYOUT
DRIVE 1

CYLS	CNTL/DRIVE 1	PARTITION	MAJ/MIN	FILENAME	XFER
0	Volume Header — SASH — SGI Label — Drive param — Partition tbl	8	(4,24)	ips0d1vh	block
6					
7	UNIX User File System (/d)	7	(4,23)	ips0d1s7	block
806					
807	ALTERNATE TRACKS	9	(4,25)	ips0d1s9	block
823					

DEFAULT DISK LAYOUT
DRIVE 0

CYLS	CNTL/DRIVE 0	PARTITION	MAJ/MIN	FILENAME	XFER
0	Volume Header — SASH — SGI Label — Drive param — Partition tbl	8	(4,8)	ips0d0vh	block
6					
7	UNIX ROOT File System (/)	0	(4,0)	ips0d0s0	block
108					
109	UNIX SWAP SPACE	1	(4,1)	ips0d0s1	block
313					
314	UNIX USER FILE SYSTEM (/usr)	6	(4,6)	ips0d0s6	block
806					
807	ALTERNATE TRACKS	9	(4,9)	ips0d0s9	block
823					

Adding a Second Disk

- boot *fx*, select drive #1 (ESDI) or #2 (SCSI)
- select partition(s), or reconfigure
- boot *unix*
- create file system directory(s)
- link partition device file(s)
- create new file system(s)
- label and mount file system(s)
- update */etc/fstab*
- *sync* information to disk

Second Disk Options

- one large *usr* space, plus *vh*, [*trkrepl*]
- multiple *usr* file systems
- duplicate of drive #1

Create Directory(s) and Link Device Files

- need "mount-point directory" in "/":

  mkdir /usr1 or /d1

- create links from device files to */dev/usr1*

  ln /dev/dsk/dks0d2s7 /dev/usr1 (block device)

  ln /dev/rdsk/dks0d2s7 /dev/rusr1 (character device)

dks = SCSI
ips = ESDI

Create, Label, and Mount File Systems

- make file system using character device:

  mkfs /dev/rusr1

- label file system using character device:

  labelit /dev/rusr1 usr1 sgi ("sgi" is volume name)

- mount file system using block device:

  mount /dev/usr1 /usr1

Final Steps

- update */etc/fstab*:

  one line for each file system

  entry specifies both raw and block device:

  ```
  /dev/usr1   /usr1   efs   rw,raw=/dev/rusr1   0   0
  ```

  linked file mount point type read/write

- flush new information to disk:

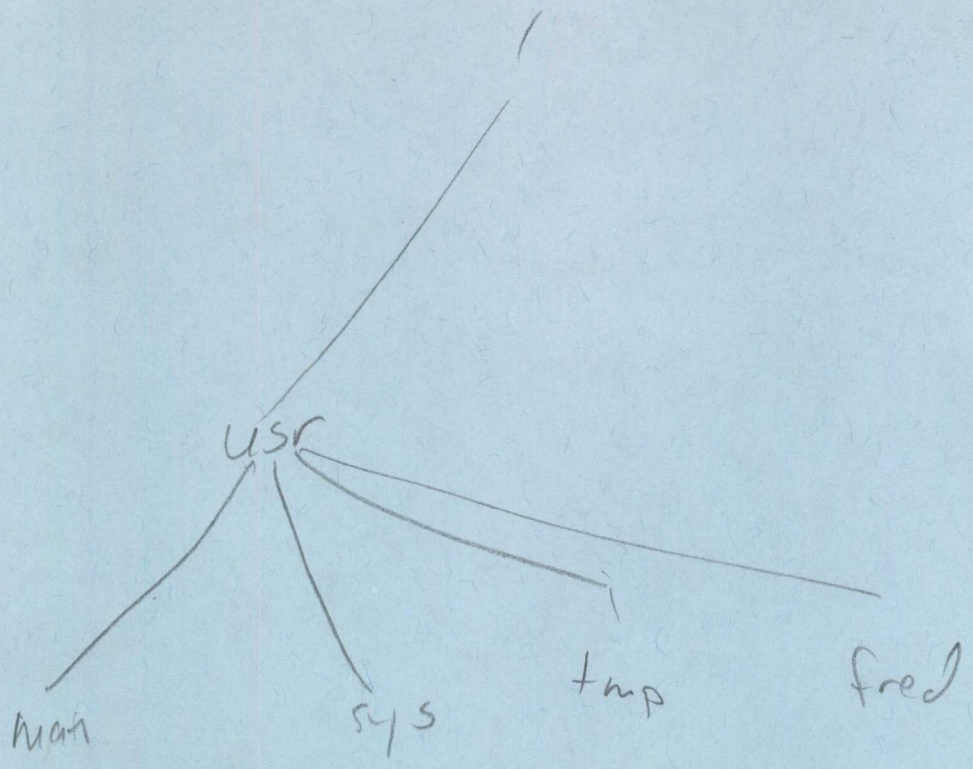
  sync

- update system logbook

Disk Addition Lab

Objectives:

- create file system(s)
- mount and label type df 1st
- update */etc/fstab*
- reboot and verify


/
usr
man
sys
tmp
fred

Installing Software

Objectives:

- describe software update tools
- learn software update procedure
- use *versions* utility
- network installation

Software Update Tapes

software distributed on tape which:

- "automates" and simplifies the update process
- contains self-updating installation tools
- permit specialized tools for specific release or option

Distribution Tape Organization

cpio format:

dd format

Standalone Tools	Swap area mini root	Product Descriptor	Image	Image	Image
fx.sash	inst		Subsystem	Subsystem	Subsystem

- fx, sash
- inst ("miniroot")
- descriptor (index to tape)
- subsystems (image files of update)

programfiles & man pages

Overview of Update Process

- back up system
- shut down to prom
- [set console to ascii terminal]
- boot sash from distribution tape
- copy miniroot into swap partition
- boot the miniroot (runs installation tool)
- select items to install
- reboot
- backup system

Update Details

- boot sash from distribution tape:

```
boot -f tpqic()sash.R2300    (MBC with VME-QIC)
boot -f tpqic()sash.IP4      (SBC with VME-QIC)
boot -f tpsc(0,7,0)sash.IP4  (SBC with SCSI)
```

- copy miniroot into swap partition:

```
cp -b 4k tpqic(,,1) dkip(0,0,1)     (VME tape, ESDI disk)
cp -b 4k tpqic(,,1) dksc(0,1,1)     (VME tape, SCSI disk)
cp -b 4k tpsc(0,7,1) dkip(0,0,1)    (SCSI tape, ESDI disk)
cp -b 4k tpsc(0,7,1) dksc(0,1,1)    (SCSI tape, SCSI disk)
```

- boot the miniroot:

```
boot -f dkip(0,0,1)unix.R2300    (MBC, ESDI disk)
boot -f dkip(0,0,1)unix.IP4      (SBC, ESDI disk)
boot -f dksc(0,1,1)unix.IP4      (SBC, SCSI disk)
```

Installation Script

- prompts for network:

```
Are you going to use the network? (y or n)
```

→ for remote install?

- prompts for update vs build:

```
Is this an update or build?  [u]
```

- prompts for distribution source:

```
Distribution source? [/dev/nrtape]
```

linked to UNE

if SCSI, use SCSI default below

note - pre 3.0 links /dev/nrtape to /dev/tpqic device!

tps=SCSI /dev/mt/tps0d7nr
ts=UNE /dev/mt/ts0d7nr

Installation Hierarchy

product - image - subsystem

image installation prompts:

- yes - install the entire set
- no - skip to next section
- pick - individually select subsystems

each subsystem consists of:

- *option*.sw.*subsystem*
- *option*.man.*subsystem*

Software Update Lab

Objectives:

- install NFS Software
- follow Section 9.1.2 of Owner's Guide
- substitute correct disk, tape devices

Identifying Software versions

version:

- reports version number
- lists files according to installation status

format:

versions [options] [operator] [selectors]

options:

m - operate on files modified since installation

u - operate on files unmodified since installation

c - operate on configuration files

s - operate on system files

v - verbose listing

versions Operators

- list (list file names):

  versions -m list

- long (list in long format):

  versions -c long

- remove (removes the files)

  versions -v remove std.sw.uucp

  versions -v remove std.man.uucp

- config (displays status of config files)

Updates and Config Files

- new version = old:

  do not install

- new version, non-critical information:

  new version installed as *file.N*

- new version contains critical features:

  install new, rename old as *file.O*

never replace a new config file with an older one!

Owner's Guide Section 9-5 9.1.3

Updates Over the Network

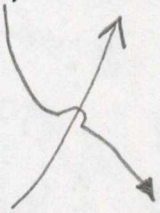
- identify local host name, address
- identify remote host name, address
- set *netaddr* prom variable
- run *bootp* daemon on remote machine
- load sash from remote tape
- copy miniroot into local memory
- boot miniroot
- select images to install

Remote Sash, Miniroot

- boot sash from remote tape:

boot -f bootp()*server* :/dev/tape(sash.IP4) - MBC

or



boot -f bootp()*server* :/dev/tape(sash.R2300) - SBC

- copy miniroot into local memory:

1. cp -b 4k bootp()*server* :/dev/tape null()
2. cp -b 4k bootp()*server* :/dev/nrtape null()
3. cp -b 4k bootp()*server* :/dev/nrtape dkip(0,0,1) - ESDI

do all

or

1. cp -b 4k bootp()*server* :/dev/tape null()
2. cp -b 4k bootp()*server* :/dev/nrtape null()
3. cp -b 4k bootp()*server* :/dev/nrtape dksc(0,1,1) - SCSI

do all

or

File System Backups

Objectives:

- need for file system backups
- tape care and storage
- tape backup utilities
- complete file system backup
- incremental file system backup
- file system restores

Why Backup Your System?

- avoid loss of data should system crash
- archive data for long-term storage
- offload files to create disk space
- prepare for file system updates
- after completing file system updates

Backup Options

simple file, directory dumps

- move files to another system
- create user's own copy

complete file system backups

- create root tape for recovery of root file system
- create usr tapes for recovery of usr file system

Types of Backups

full backup

- entire system
- fairly involved process
- requires multiple tapes

incremental backup

- files modified since last full
- easy to automate

Backup Locations

to tape

- easy to manage
- secure
- inexpensive

to disk

- fast
- limited security
- expensive

to remote system (disk or tape)

- tapeless local systems
- easiest "recovery"

Frequency of Backups

users:

- as needed

administrator:

- incremental nightly
- full backup weekly, biweekly
- others as needed (.e.g., updates, maintenance)

Backup Tips

- document!
- automate incremental backups
- verify backup contents
- rotate tapes with care

the Rotating "Grandfather"

- create full backup (tapes A1-n)
- create incremental backups (tapes B1-n)
- create next full backup (tapes C1-n)
- create incremental backups (reuse tapes B1-n)
- create full backup (reuse tapes A1-n)
- archive one full backup monthly

Cartridge Tape Maintenance

- keep clean, free of static & moisture
- store on edge
- don't touch tape surface
- avoid magnetic surfaces, xrays
- retention before using:

  mt ret

- label properly
- clean tape heads frequently

Tape Labels

- name of system, drive, file system
- date of backup
- full or incremental
- number in sequence
- starting directory
- tape backup command (complete format)
- Name & ph # of person.

Tape Cleaning

- clean heads and capstans every two hours of operation
- use supplied swabs, solvent (Freon TF)
- don't dip used swab in solvent
- procedure:

  wet swab with solvent

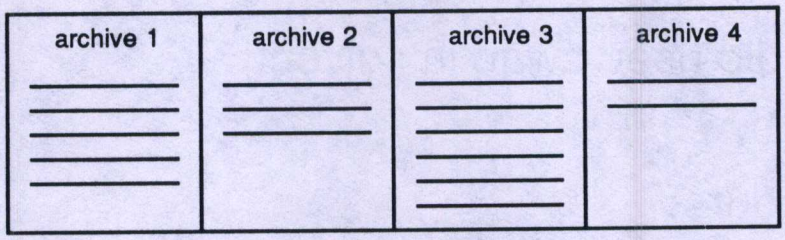
  gently rub head surface

  clean roller guides with fresh swab and solvent

  dry with fresh swab

What is an Archive?

- multiple files stored as one large file
- headers separate files for access



- norewind driver allows multiple archives on one tape

  /dev/nrtape -------> /dev/mt/tps0d7nr (SCSI)

  /dev/nrtape -------> /dev/mt/ts0d0nr (VME-QIC)

- rewind driver requires serial read through entire tape

  /dev/tape ------> /dev/mt/tps0d7 (SCSI)

  /dev/tape ------> /dev/mt/ts0d0 (VME-QIC)

Tape Archive Utilities

tar (tape archive) - for user backups

- simple to use
- slower performance
- supports "/" stripping

cpio (copy in/out) - for system backups (dump)

- harder to use
- many options

dd (convert and copy)

- backup to remote tape drive

tar, cpio, and dd are incompatible!

Archiving with tar

format:

tar [key] [name ...]

keys:

c - create new tape

x - extract from tape

t - list names on tape

---

v - verbose

f - use alternate device (must supply)

w - prompt for user confirmation

Archiving with tar

rewind examples:

- tar cv .
- tar t
- tar xv
- tar xv file1

no rewind example:

- mt rewind

  mt fsf 2

  tar xvf /dev/nrtape file1

Creating List of Files for tar

- use find command to select files
- append minus (-) after tar key

examples:

```
ls file* | tar c -

find . -mtime -7 -print | tar c -

find /usr/people -user bill -print | tar c -
```

Copying with cpio

format:

find [find options] | cpio key[options] > device

keys:

o - copy out

i - copy in

options:

v - verbose

t - print table of contents

only

-it lists files only

Copying with cpio

rewind examples:

- find . -print | cpio -oBH > /dev/tape
- cpio -itvB < /dev/tape
- cpio -ivBmud < /dev/tape
- cpio -ivBmud file1 < /dev/tape single file

no rewind example:

mt rewind

mt fsf 3

cpio -ivBmud file < /dev/nrtape

Remote Copying with dd

- *tar* or *cpio* on local machine
- pipe to *dd* on remote machine
- specify remote tape device file
- use *dd* options shown

Objective	tar/cpio Commands
create	tar cbf 10 - *dir* | rsh *host* dd ibs=10k obs=60k of=*device*
create	find *dir* -print | cpio -oBH | rsh *host* dd ibs=10k obs=60k of=*device*
read	rsh *host* dd ibs=60k obs=10k if=*device* | tar tvbf 10 -
read	rsh *host* dd ibs=60k obs=10k if=*device* | cpio -itvB
restore	rsh *host* dd ibs=60k obs=10k if=*device* | tar xvbf 10 - [*files*]
restore	rsh *host* dd ibs=60k obs=10k if=*device* | cpio -ivBmud [*files*]

Making Backups

- number of archives per tape

  (to rewind or to not rewind)

- starting directory, addressing scheme

  (relative, absolute, absolute-relative)

- size of directories

  (segmentation of file system directories)

Multiple or Single Archives

multiple:

- cheapest
- quickest access to files
- requires careful tape management
- write errors limit use

single:

- most expensive
- slowest access to files
- easiest to manage

Directories and Addressing Schemes

relative:

- *cd* to the directory
- back up current directory, all subdirectories
- to restore, return to same directory

absolute:

- specify any directory with absolute path
- to restore, ~~must be in root directory~~ can be in any

"absolute-relative":

- specify any directory with relative path
- to restore, must be in same directory as store

Addressing Scheme Examples

on tape label:

- specify starting directory
- specify tape copy command (complete)

Size of Directory

- cd to directory
- check directory size:

  du -s du -s *
- if too big to fit on 39 mbyte tape (76k blocks), cd to lower directory
- use tar or cpio to copy directory to tape

Examining Disk Usage with du

- du - recursively list directory size (blocks)
- du -s - total all subdirectories

```
%du
24./familyroom
20./kitchen/backporch/backyard/barn
22./kitchen/backporch/backyard
25./kitchen/backporch
1  ./kitchen/oven
5  ./kitchen/cabinet
34./kitchen
3  ./livingroom
62.
```

Blocks if less than 96,000 Blocks

Complete System Backups

Process Overview:

- become super user, cd to root
- follow complete shutdown sequence, to init 1
- back up *root* file system (no absolute addresses!)
  tar cv.
- run *fsck* on *usr* file system
- mount *usr*
- cd to *usr*
- back up *usr* (no absolute addresses!)
  tar cv.
- verify tapes and document

note: multi-volume backups not supported

Backing Up *usr*

- run *fsck* to ensure integrity:

  fsck /dev/rusr

  file system must *not* be mounted

- mount *usr* file system: mount /dev/usr /usr
- cd to /usr
- check file system size with *df* reports in Bytes
- check *usr* directory sizes with *du -s* reports in Blocks
- prioritize and group directories
- create separate archive for each group

/usr Directory Priorities

- put directories that change most on same tape
- total directory size must fit on tape
- "smart" script can automate directory grouping

Tape Width	Tape Length	Capacity
Quarter-inch	450 ft.	40MB
Quarter-inch	600 ft.	55MB
Half-inch	2400 ft.	44.6MB

Hints for Incremental Backups

- use *find* options to get specific files

```
find . -mtime -7 | tar cv -

find . -mtime -7 | cpio -oBH > /dev/tape
```

- employ shell script to help out

```
filename=‘date | cut -d" " -f1-3 | tr -d " "‘.ilog
date > $filename
echo >> $filename
find / -mtime -7 | cpio -oBH > /dev/tape 2>> $filename
echo >> $filename
cpio -ivtB < /dev/tape >> $filename
pr $filename | lp
```

loose!

- automate with entry in *cron* directory

```
0 0 * * 1-5 /bin/sh /usr/admin/backup.i
```

Restoring Files from Backup

- use same utility as store
- *cd* to same directory as store
- use same addressing scheme as store
- restoring *root* requires system recovery using *miniroot*

4D System Administration

Backup Strategy Example

1) su; cd /

2) execute S.A. shutdown procedures (e.g., ps, who, whodo, write, wall)

3) init 1

4) df

5) tar cv .

6) fsck /dev/rusr

7) mount /dev/usr /usr

8) cd /usr

9) du -s * | sort -nr > backup.size

10) cat backup.size | cut -f2 > backup.master

11) du -s /usr > backup.total

168067 /usr

12) expr 39000000 \/ 512 > backup.blocks

76171

13) compare backup.size backup.pri backup.master and create
backup.one backup.two backup.three.

Backup.size		Backup.master	Backup.pri
68263	people	people	1 people
24242	lib	lib	3 lib
15397	bin	bin	3 bin
13583	demos	demos	1 demos
9128	diag	diag	3 diag
8243	sbin	sbin	4 sbin
8067	catman	catman	4 catman
6003	etc	etc	4 etc
4886	tmp	tmp	1 tmp
2779	sysgen	sysgen	5 sysgen
2719	include	include	6 include
1719	bsd	bsd	4 bsd
727	adm	adm	5 adm
682	dict	dict	6 dict
635	lbin	lbin	5 lbin
312	spool	spool	2 spool
261	tutor	tutor	7 tutor

```
224     admin           admin           4 admin
166     lost+found      lost+found      1 lost+found
19      pub             pub             8 pub
9       local           local           7 local
3       news            news            2 news
2       mail            mail            2 mail
1       src             src             7 src
1       preserve        preserve        8 preserve
1       games           games           3 games
```

```
Backup.one      Backup.two      Backup.three
===========================================

people          lib             catman
tmp             bin             etc
spool           demos           include
lost+found      diag
pub             sbin
local           sysgen
news            bsd
mail            adm
src             dict
preserve        lbin
games           tutor
                admin
```

14) tar cv - < backup.one or cat backup.one | tar cv -

15) repeat step 13 for each backup.X file

```
#! /bin/sh

# Backup script


clear
backup=/usr/backup

echo "Checking user ID\n\n"
if [ -x /usr/bin/id ]
then
  eval `id  |  sed 's/[^a-z0-9=].*//'`
  if [ "${uid:=0}" -ne 0 ]
  then
    echo "$0:  Only root can run this install script."
    exit 2
  fi
fi

echo
echo
echo "Removing temporary files \
: backup.size,master,total,temp1,temp2,temp3"

test -f $backup/backup.size && rm $backup/backup.size
test -f $backup/backup.master && rm $backup/backup.master
test -f $backup/backup.total && rm $backup/backup.total
test -f $backup/backup.temp1 && rm $backup/backup.temp1
test -f $backup/backup.temp2 && rm $backup/backup.temp2
test -f $backup/backup.temp3 && rm $backup/backup.temp3
test -f $backup/backup.temp4 && rm $backup/backup.temp4

cd /usr

# create file with directory sizes

echo
echo
echo Checking File System size

du -s * | sort -nr > /usr/backup/backup.size


# create file with just directory names in same order as backup.size

cat /usr/backup/backup.size | cut -f2 > /usr/backup/backup.master


# create file with usr's block size

du -s /usr | cut -f1 > /usr/backup/backup.total


# create variable with usr's block size

blk_count=`cat /usr/backup/backup.total`
```

```
echo
echo
echo The usr file system is $blk_count blocks


# Create variable that converts 39mg to blocks, max size per tape

blk_max=`expr 39000000 \/ 512`


# using backup.pri to build files to be used later for creating
# backup1, backup2, backup3, and so on ...

paste /usr/backup/backup.size /usr/backup/backup.pri > \
/usr/backup/backup.temp1

cat /usr/backup/backup.temp1 | tr -s '  ' ' ' | tr -s ' ' ':' > \
/usr/backup/backup.temp2

cat /usr/backup/backup.temp2 | sort -t: +2n +0nr | cut -d: -f1,2 > \
/usr/backup/backup.temp3


# Checking to ensure that size and pri are same size in length

chk1=`wc -l $backup/backup.size | tr -s ' ' ' ' | tr -s ' ' ':' | \
        cut -d: -f2`
chk2=`wc -l $backup/backup.pri | tr -s ' ' ' ' | tr -s ' ' ':' | \
        cut -d: -f2`

if [ "$chk1" -ne "$chk2" ]
then
        clear
        echo "backup.size and backup.pri are not same size"
        exit 2
fi


# start building backup_temp files

export blk_max
export blk_count
/usr/backup/backup.script2
```

```
#! /bin/sh

blk_count_temp=$blk_count
blk_max_temp=$blk_max

tapenum=0

file="/usr/backup/backup.temp3"
backup="/usr/backup"

month=`date | tr -s ' ' ':' | cut -d: -f2`
day=`date | tr -s ' ' ':' | cut -d: -f3`
date_stamp="$month $day"
date_stamp=`echo $date_stamp | tr -d ' '`

echo
echo
echo Starting to create backup files

file_len=`wc -l $file | tr -s ' ' ' ' | tr -s ' ' ':' | cut -d: -f2`
while [ "$file_len" -ne "0" ]
do

        tapenum=`expr $tapenum + 1`
        touch $backup/backup_$tapenum.$date_stamp
        line_num=1
        blk_max_temp=$blk_max

        until [ "$line_num" -gt "$file_len" ]
        do

        read_value=`sed -n "$line_num p" $file`
        dir_name=`echo $read_value | cut -d: -f2`
        blk_size=`echo $read_value | cut -d: -f1`
        blk_count_chk=`expr $blk_max_temp - $blk_size`
        if [ "$blk_count_chk" -gt "0" ]
        then
                sed "$line_num d" $file > $backup/backup.temp4
                file_len=`expr $file_len - 1`
                mv $backup/backup.temp4 $backup/backup.temp3
                echo $dir_name >> $backup/backup_$tapenum.$date_stamp
                blk_max_temp=`expr $blk_max_temp - $blk_size`
                blk_count_temp=`expr $blk_count_temp - $blk_size`
        else
                line_num=`expr $line_num + "1"`
        fi

        done
done

# Instead of until loops, the following could be used

# for i in /usr/backup/temp3
# do
#       read_value=$i
#       ...
```

File System Backup Lab

Objectives:

- backup root
- backup usr
- label tapes

Commands to Store

Tape Type	tar Command
Cartridge/ rewind	tar c *dir*
Cartridge/ norewind	tar cf *dir* /dev/nrtape
Half-inch/ rewind	tar cf *dir* /dev/mt/xmt0d0.*density*
Half-inch/ norewind	tar cf *dir* /dev/mt/xmt0d0nr.*density*

Tape Type	cpio Command
Cartridge/ rewind	find *dir* -print | cpio -oBH > dev/tape
Cartridge/ norewind	find *dir* -print | cpio -oBH > dev/nrtape
Half-inch/ rewind	find *dir* -print | cpio -oBH > /dev/mt/xmt0d0.*density*
Half-inch/ norewind	find *dir* -print | cpio -oBH > /dev/mt/xmt0d0nr.*density*

Commands to List

Tape Type	tar Command
Cartridge/ rewind	tar tv
Cartridge/ norewind	tar tvf /dev/nrtape
Half-inch/ rewind	tar tvf /dev/mt/xmt0d0.*density*
Half-inch/ norewind	tar tvf /dev/mt/xmt0d0nr.*density*

Tape Type	cpio Command
Cartridge/ rewind	cpio -itvB < /dev/tape
Cartridge/ norewind	cpio -itvB < /dev/nrtape
Half-inch/ rewind	cpio -itvB < /dev/mt/xmt0d0.*density*
Half-inch/ norewind	cpio -itvB < /dev/mt/xmt0d0nr.*density*

Commands for Restore

Tape Type	tar Command
Cartridge/ rewind	tar xv [*files*]
Cartridge/ norewind	tar xvf /dev/nrtape [*files*]
Half-inch/ rewind	tar xvf /dev/mt/xmt0d0.*density* [*files*]
Half-inch/ norewind	tar xvf /dev/mt/xmt0d0nr.*density* [*files*]

check links

Tape Type	cpio Command
Cartridge/ rewind	cpio -ivBmud [*files*] < /dev/tape
Cartridge/ norewind	cpio -ivBmud [*files*] < /dev/nrtape
Half-inch/ rewind	cpio -ivBmud [*files*] < /dev/mt/xmt0d0.*density*
Half-inch/ norewind	cpio -ivBmud [*files*] < /dev/mt/xmt0d0nr.*density*

File System Recovery

Objectives:

- when to restore
- miniroot recovery tool
- full system recovery
- partial system recovery

When to Restore a File System

- partitions moved and file system destroyed
- disk drive physically damaged
- *fsck* shows major problems
- major files or kernel destroyed or removed

Things to Check Before Restoring

- eliminate physical problems:
  - disk drive cabling
  - console cabling
  - reseat controller boards
- is root file system bad?
  - boot from second disk
  - boot from remote system
  - boot *sash*, cat specific files
  - boot *fx*, read disk label

Recovery Tool: Miniroot

- "bare-bones" operating system
- simple file system
- on every distribution tape
- lives in disk "swap" partition during recovery

Types of Recovery

complete system recovery:

- all data on disk is lost
- root, usr file systems rebuilt
- root, usr file systems restored from backup

partial system recovery:

- root good, usr bad (restore usr)
- usr good, root bad (restore root)
- retrieve usr files, then do full recovery

Miniroot Recovery Process

- boot sash from distribution tape
- copy miniroot into swap partition
- boot miniroot
- update or build?

  build - full system recovery

  update - partial recovery, automatic file system mounts

  quit - partial recovery, manual file system mounts

- reload root, usr from backup tapes
- reboot system

Full System Recovery

- boot sash from distribution tape:

Source	boot Command
MBC w/VME-QIC	boot -f tpqic()sash.R2300
SBC w/VME-QIC	boot -f tpqic()sash.IP4
SBC w/SCSI	boot -f tpsc(0,7,0)sash.IP4

only 1st time - when 2nd, don't use .IP4 or .R2300 suffix

- copy miniroot into swap partition:

Source, destination	copy Command
VME tape, ESDI disk	cp -b 4k tpqic(,,1) dkip(0,0,1)
VME tape, SCSI disk	cp -b 4k tpqic(,,1) dksc(0,1,1)
SCSI tape, ESDI disk	cp -b 4k tpsc(0,7,1) dkip(0,0,1)
SCSI tape, SCSI disk	cp -b 4k tpsc(0,7,1) dksc(0,1,1)

Full System Recovery

- boot miniroot:

Source	boot Command
MBC, ESDI disk	boot -f dkip(0,0,1)unix.R2300
SBC, ESDI disk	boot -f dkip(0,0,1)unix.IP4
SBC, SCSI disk	boot -f dksc(0,1,1)unix.IP4

- miniroot prompts:

```
Are you going to use the network? (y or n)  n

Is this an update or a build? [u]  b

Do you want to make file systems [y]  y

Distribution source? [/dev/nrtape]  quit
```

Full System Recovery:

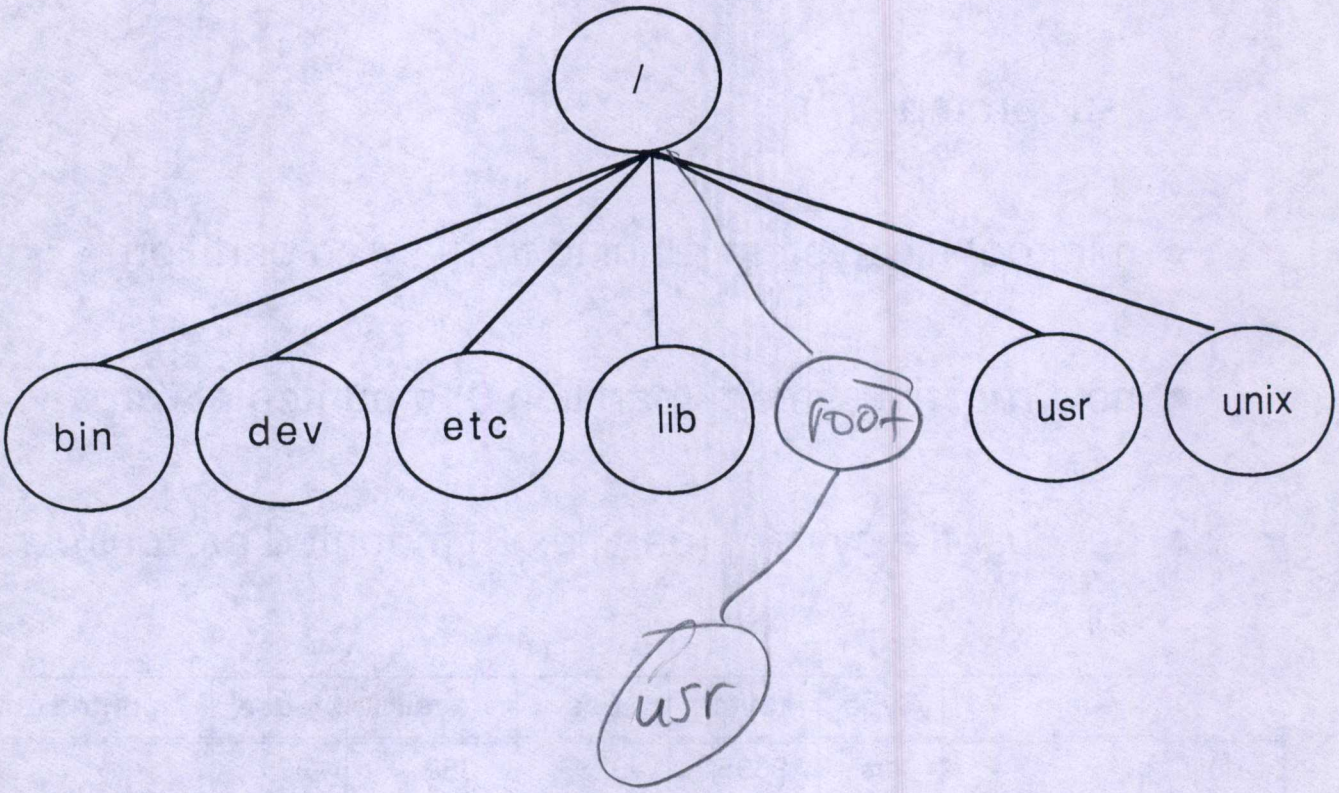
the Miniroot Environment

- "su" prompt (#)
- miniroot file system running from swap partition
- new *root* file system (partition 0) mounted at */root*
- new *usr* file system (partition 6) mounted at */root/usr*

Filesystem	Type	kbytes	use	avail	%use	Mounted on
/dev/root	efs	9695	9236	369	96%	/
/dev/dsk/ips0d0s0	efs	15695	10	15685	0%	/root
or						
/dev/dsk/dks0d1s0	efs	15695	10	15685	0%	/root
/dev/dsk/ips0d0s6	efs	76159	11	76148	0%	/root/usr
or						
/dev/dsk/dks0d1s6	efs	76159	11	76148	0%	/root/usr

use: /etc/devnm to find mount points.

the Miniroot Filesystem



Full System Recovery:

Restore the root File System

- cd to */root*
- use *tar* or *cpio* to restore *root*

  warning! watch for absolute addresses!

  watch for miniroot tape link to wrong device

Full System Recovery:

Restore the usr File System

- reboot system and cd to *usr*

  or

- stay in miniroot and cd to */root/usr*

- use *tar* or *cpio* to restore *root*

  warning! watch for absolute addresses!

  watch for miniroot tape link to wrong device

- reboot system

Partial System Recovery

- boot sash from distribution tape:

Source	boot Command
MBC w/VME-QIC	boot -f tpqic()sash.R2300
SBC w/VME-QIC	boot -f tpqic()sash.IP4
SBC w/SCSI	boot -f tpsc(0,7,0)sash.IP4

- copy miniroot into swap partition:

Source, destination	copy Command
VME tape, ESDI disk	cp -b 4k tpqic(,,1) dkip(0,0,1)
VME tape, SCSI disk	cp -b 4k tpqic(,,1) dksc(0,1,1)
SCSI tape, ESDI disk	cp -b 4k tpsc(0,7,1) dkip(0,0,1)
SCSI tape, SCSI disk	cp -b 4k tpsc(0,7,1) dksc(0,1,1)

Partial System Recovery

- boot miniroot:

Source	boot Command
MBC, ESDI disk	boot -f dkip(0,0,1)unix.R2300
SBC, ESDI disk	boot -f dkip(0,0,1)unix.IP4
SBC, SCSI disk	boot -f dksc(0,1,1)unix.IP4

- miniroot prompts:

```
Are you going to use the network? (y or n)  n

Is this an update or a build? [u]  quit
```

leaves you with miniroot prompt. #

Partial System Recovery:

the Miniroot Environment

- "su" prompt (#)
- miniroot file system running from swap partition
- old file systems are intact, but not mounted
- mkfs, label and mount root to restore *root* only

  or

  mount old root, then
  mkfs, label and mount usr to restore *usr* only

Partial System Recovery:

Restore root File System

- if no "root" directory in mini-root, then
- mkdir root
- ls -C or
- ESDI systems:

```
mkfs  /dev/rdsk/ips0d0s0
labelit /dev/rdsk/ips0d0s0  root  SGI
mount  /dev/dsk/ips0d0s0  /root
```

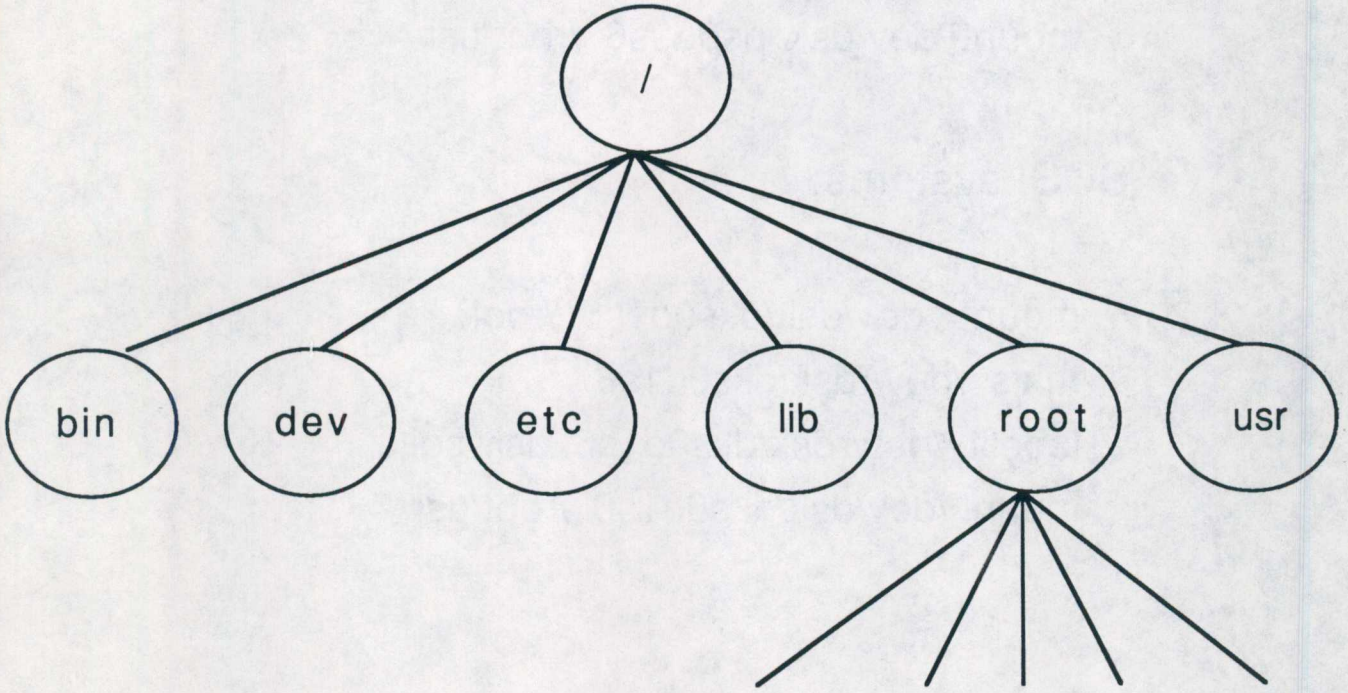
- SCSI systems:

```
mkfs  /dev/rdsk/dks0d1s0
labelit /dev/rdsk/dks0d1s0  root  SGI
mount  /dev/dsk/dks0d1s0  /root
```

Partial System Recovery:

Restore root File System

- cd to /root
- restore root files from backup tape
  tar xvf /dev/t
- reboot



Partial System Recovery:

Restore usr File System

- if no "root" directory,) else
- mkdir root
- ESDI systems:

```
mount  /dev/dsk/ips0d0s0  /root
mkfs  /dev/rdsk/ips0d0s6
labelit  /dev/rdsk/ips0d0s6  usr  sgi
mount /dev/dsk/ips0d0s6  /root/usr
```

- SCSI systems:

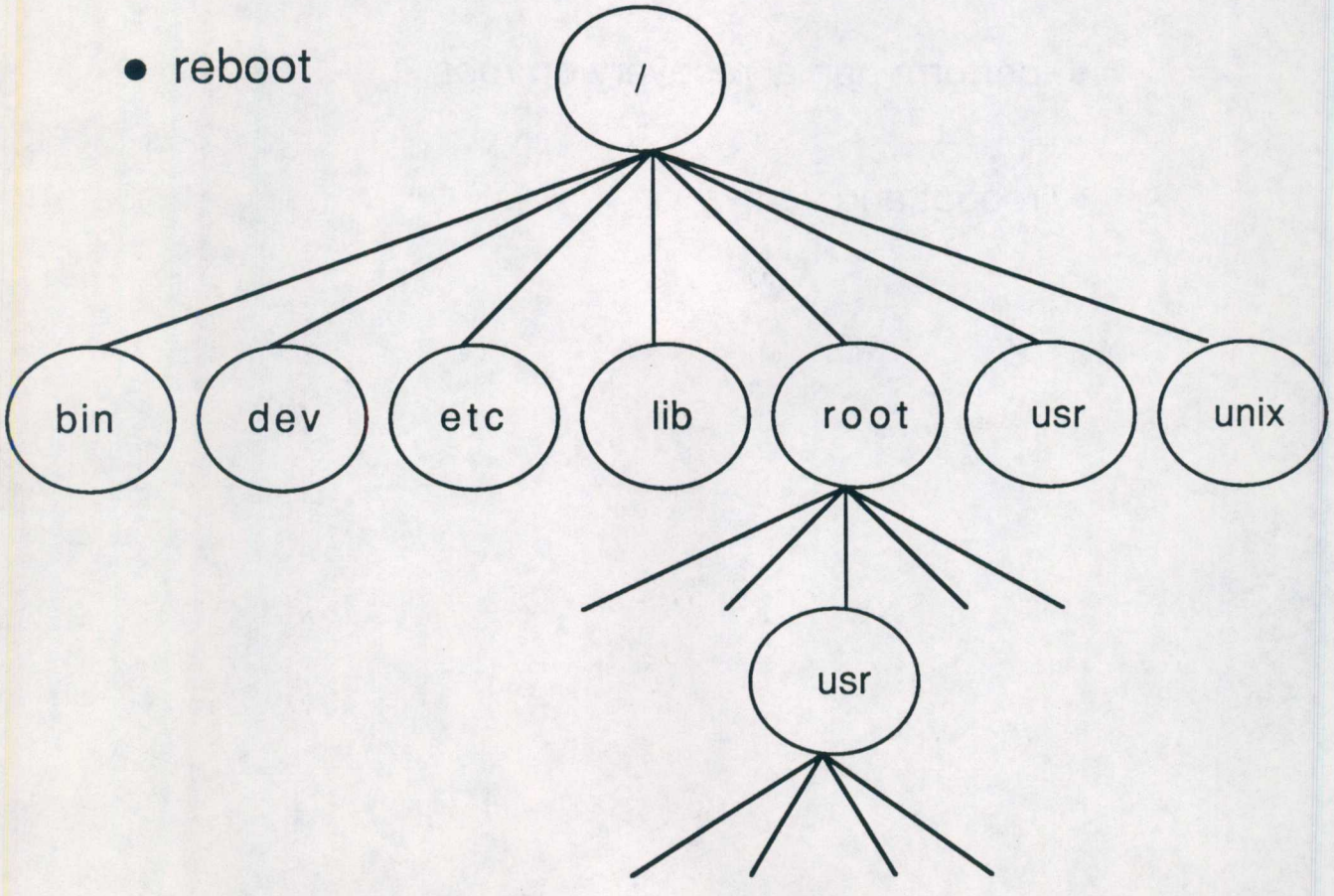
```
mount  /dev/dsk/dks0d1s0  /root
mkfs  /dev/rdsk/dks0d1s6
labelit  /dev/rdsk/dks0d1s6  usr  sgi
mount /dev/dsk/dks0d1s6  /root/usr
```

Partial System Recovery:

Restore usr File System

Don't reboot yet!!

- cd to /root/usr
- restore usr files from backup tape
- reboot



File System Recovery Lab

Objectives:

- boot sash
- copy miniroot
- boot miniroot
- perform partial recovery on *root*
- reboot and verify

4D System Administration

Figuring out how to answer the miniroot questions

question #1 : Are you going to use the network (y or n)

question #2 : Is this an update or build? [u]

question #3 : Distribution source? [/dev/nrtape]

Note : * means /dev/mt/tps0d7nr

	Builds Prod tapes	Updates Prod tapes	Full Sys Rec Backups	Partial Sys Rec Backups Auto mounts	Partial Sys Rec Backups Manually
q1	y or n	y or n	y or n	y or n	y or no
q2	b	u or <ret>	b	u or <ret>	quit
q3	*	*	quit	quit	

Builds from Product Distribution Tapes create new root & usr file Systems

Updates will add software to root and usr file systems

Full system recoveries create new root and usr file systems, be judicious in use

Partial sys rec with automatic mounts will mount /root & /root/usr

Partial sys rec without auto mounts need to mount /root & /root/usr manually

Warning !!! Always backup file systems before attempting to build or update your systems.

Note : even in system crashes, it may be wise to perform partial recoveries at first, backup recently modified or created data, then perform full system recovery procedures.

4D System Administration

Troubleshooting Miniroot Tricks

1) You can reboot miniroot from Sash as many times as needed as long as swap space has not been written over.

2) You can exit miniroot back to 'Update or Build' question by typing 'exit'.

3) You can get to prom monitor by typing 'reboot' or 'shutdown'. Do not hit the reset button or on/off switch because of FSCK problems.

4) To fix unix related problems over the network :

```
Are you installing over the network [y or n] "y"

        your system name : "two"
        your address     : "192.0.0.2"
        server name      : "five"
        server address   : "192.0.0.5"

Update or build [u or b] <return>
Distr source [ /dev/tape ]  "quit"

# rcp five:/unix /root/unix
        or
# rcp -r five:/ /root

# reboot

>> auto
```

owner's guide Page 8-12

Don't copy /dev files across the net → rcp /dev/MAKEDEV then run MAKEDEV.

5) To load software updates over the network :

```
>> setenv netaddr 192.0.0.2

>> boot

Sash: cp -b 4k bootp()server:/dev/tape null()
Sash: cp -b 4k bootp()server:/dev/nrtape null()
Sash: cp -b 4k bootp()server:/dev/nrtape dkip(0,0,1)

Sash: boot -f dkip(0,0,1)unix.IP4

Are you installing over the network [y or n] "y"

        your system name : "two"
        your address     : "192.0.0.2"
        server name      : "five"
        server address   : "192.0.0.5"

Update or build [u or b]    <return>
Distr source [ /dev/tape ]  "/dev/mt/tps0d7nr"

Answer "yes", "no", or "pick":

Install image? [y]
```

Linked to VME-QIC.

6) If a copy of unix is on the "usr" partition or file system

```
mount /dev/dsk/ips0d0s6 /usr
cp /usr/unix /root/unix
```

7) If a copy of unix is on the "/d" partition or file system

```
mount /dev/dsk/ips0d1s0 /d
cp /d/unix /root/unix
```

8) If the tape drive entries are not linked properly

A)
```
ln /dev/mt/tps0d7nr /dev/nrtape
ln /dev/mt/tps0d7 /dev/tape
cd /root
tar xv
```

B)
```
cd /root
tar -xvf /dev/tape
```
drivername

C)
```
cd /root
use dd .... /dev/nrtape
```

Note: if you use the ln command backwards, you have to reload miniroot and start over

9) Boot unix from remote system while in prom monitor, assuming that only the copy of the kernel is bad and that the file system is alright

```
boot -f bootp()five:/unix
```

10) If you make a major mistake or you want to recover minimally at this point and you do not have a local tape drive.

Boot miniroot and copy miniroot to /root : rcp -r / /root

But beware!!!!

You will still need to boot /unix carefully, and then restore root file system from backup tapes while in standard unix.

11) If all else fails, your root backup is bad, or you just plain giveup

Build new root and usr filesystems, reboot, restore root and usr from

Miniroot unix environment

1) Miniroot with the real UNIX root file system mounted :

```
Filesystem              Type kbytes    use  avail %use  Mounted on
/dev/root               efs    9695   9326    369  96%  /
/dev/dsk/ips0d0s0       efs   15695   7721   7974  49%  /root
/dev/dsk/ips0d0s6       efs   76159  40555  35604  53%  /root/usr
```

2) Standard UNIX with usr file system mounted

```
Filesystem              Type kbytes    use  avail %use  Mounted on
/dev/root               efs   15695   7721   7974  49%  /
/dev/usr                efs   76159  40555  35604  53%  /usr
```

3) Mount the UNIX /usr file system as /usr while in miniroot

```
Filesystem              Type kbytes    use  avail %use  Mounted on
/dev/root               efs    9695   9172    523  95%  /
/dev/dsk/ips0d0s0       efs   15695   7721   7974  49%  /root
/dev/dsk/ips0d0s6       efs   76159  40555  35604  53%  /usr
```

Note: you could have mounted usr as /dev/usr

4) Miniroot directories have fewer directories and files :

```
bin     dev     etc     lib     root    usr     unix
```

5) The UNIX root file system contents :

```
bin         d           etc         lib         stand       unix
core        dev         install     lost+found  tmp         usr
```

6) Miniroot's /usr directory is not a file system!!! Is it mounted???

```
adm   bin   bsd   etc   lib   sbin  src   tmp
```

7) The UNIX /usr file system contents :

```
adm         catman      lbin        mail        spool       tutor
admin       demos       lib         people      src
bin         etc         local       preserve    sysgen
bsd         include     lost+found  sbin        tmp
```

8) The Miniroot file system is linked to swap and the correct dev device file

```
/dev/root: (ESDI)
415     /dev/root
415     /dev/swap
415     /dev/dsk/ips0d0s1
```

9) The UNIX root file system is not linked to any other dev device file while in miniroot. Normally this file would be linked to the /dev/root device file

```
/dev/root: (ESDI)
411     /dev/dsk/ips0d0s0
```

10) In standard UNIX, this is how /dev/root is linked.

```
/dev/root: (ESDI)
411     /dev/root
411     /dev/dsk/ips0d0s0
```

11) Remember that Miniroot does not have a miniroot /usr file system

12) The UNIX /usr file system is linked to the correct dev device file

```
/dev/root:
435     /dev/usr
435     /dev/dsk/ips0d0s6
```

File System Maintenance

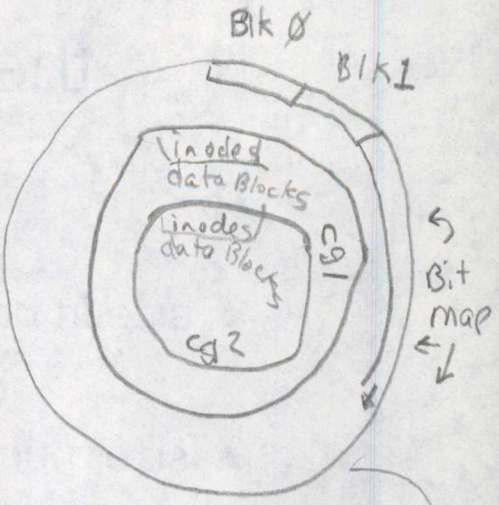
Objectives:

- Extent File System structure
- file system checking with *fsck*
- files system monitoring

File System Components

- the basic block (bb) = 512 bytes
- boot block (unused)
- superblock
- bitmap
- inodes
- data blocks
- cylinder groups

Extent File System Structure



Block 0 the Boot Block
Block 1 the Super Block
Block 2 the Bitmap Block n
inodes data blocks
inodes data blocks

cg 0

cylinder group 0:

- boot block (unused)
- superblock
- bitmap

1 bit per block

cg 1

cylinder groups 1 - n:

- inodes
- data blocks

for each partition

See pg. 7 →

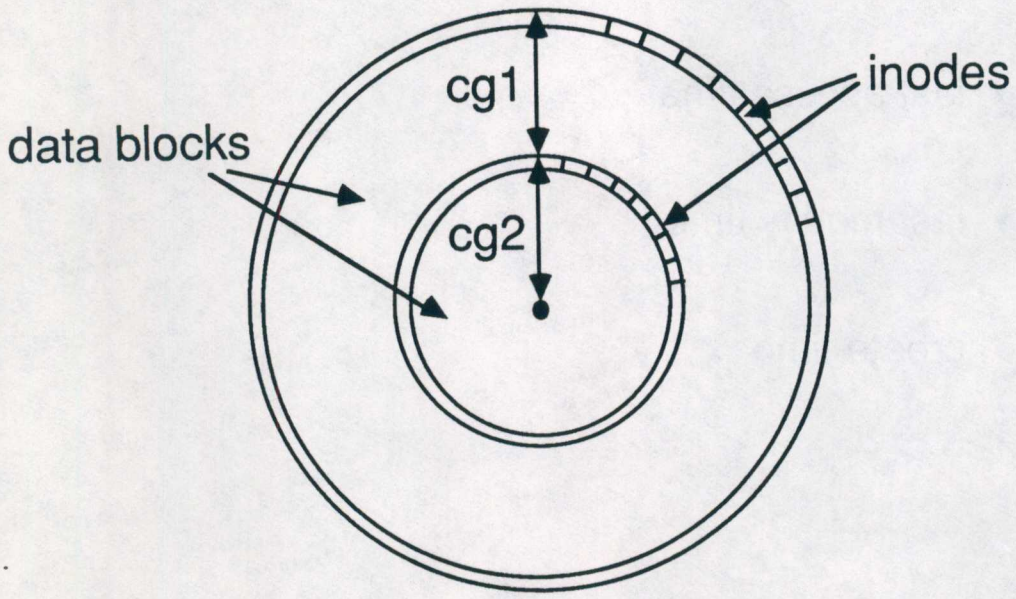
the efs SuperBlock

- size in basic blocks
- size, number of cylinder groups
- file system name
- fs_dirty bit tells fsck wether to check or not.
- size of bitmap
- number of free inodes, data blocks

see fs(4) man pages for details

the efs Cylinder Group

- inode section (constant size)
- data block section
- enables efficient access to files
- no alignment restrictions on cylinder groups



the efs Inode

128 bytes

file description:

- file mode/protection code
- number of links
- owner id
- group id
- bytes
- last access time
- last modify time
- create time

extents:

- locations
- addresses
- references.

Inode File Protection Code

2 bytes = 16 bits

bit	meaning
0	Execute for Others (x) (t if sticky bit on)
1	Write for Others (w)
2	Read for Others (r)
3	Execute for Group (x) (S or s if setgid bit on)
4	Write for Group (w)
5	Read for Group (r)
6	Execute for User (x) (S or s if setuid bit on)
7	Write for User (w)
8	Read for User (r)
9	Sticky bit
10	Setgid bit
11	Setuid bit
12	FIFO or Pipe file
13	Character file (or block if both 13 and 14)
14	Directory file (or block if both 13 and 14)
15	Regular file

1

forces it to stay in memory — Never swapped out!

the efs Extent

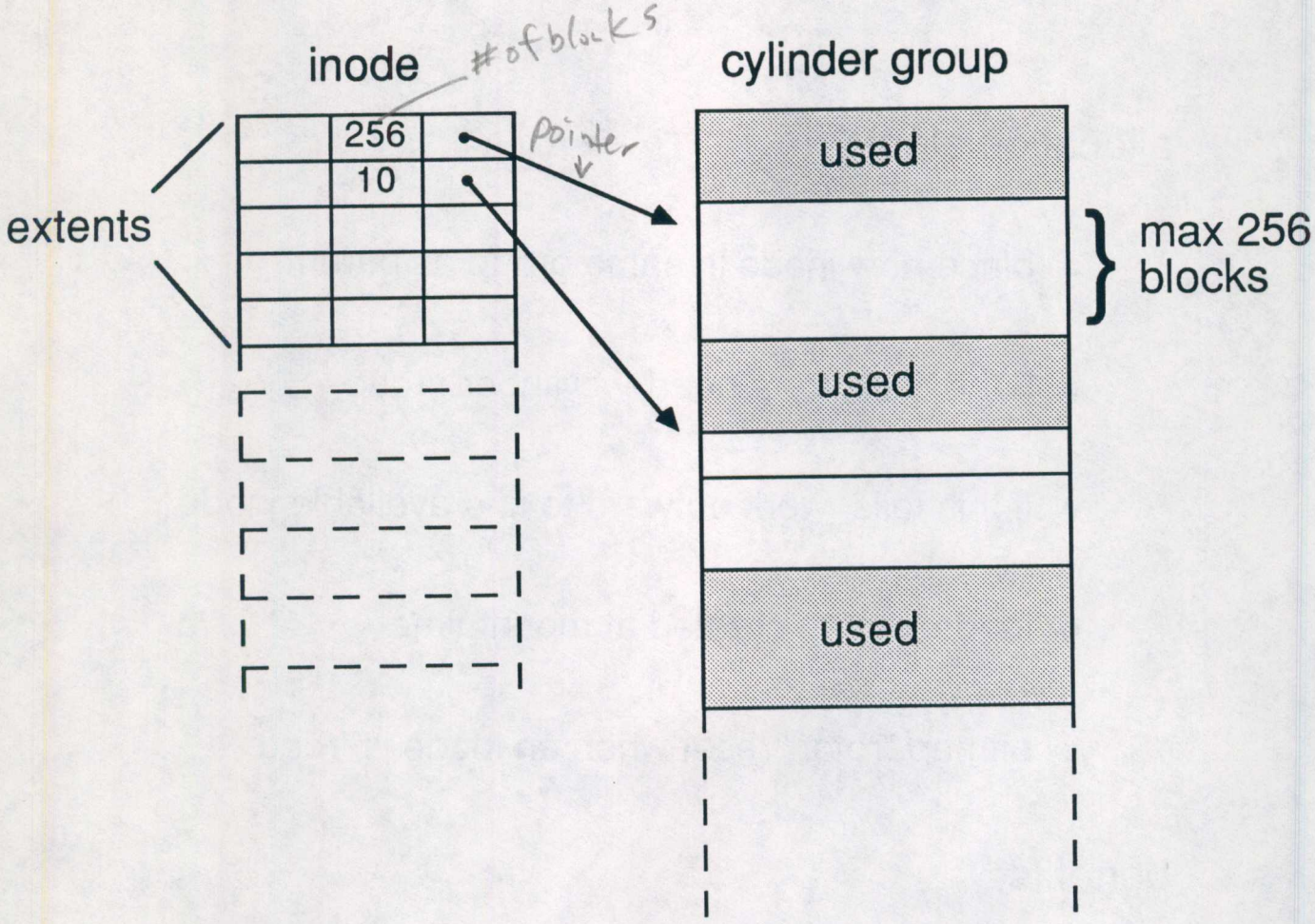
- address of file data blocks
- references *contiguous* blocks (max 256 bb's)
- up to 12 direct extents per file
- each extent contains:

  address of first physical block

  length of extent in basic blocks

  extent offset into file

efs Data Addressing



the Cylinder Group and Inode Allocation

allocation algorithm:

- place new inode in same group as parent
- if this fails, try a nearby cylinder group
- if this fails, work outward to find available inode
- inode bitmap created at mount time
- bitmap "rotor" reset when an inode is freed

pointer

benefits:

- reduce seek time to related inode
- improve packing of inodes

the Extent and Data Allocation

to "grow" a file:

- first, try to extend the extent
- if this fails, allocate a new extent:

  1) in same cylinder group as inode

  2) if this fails, search outward for free block

benefits:

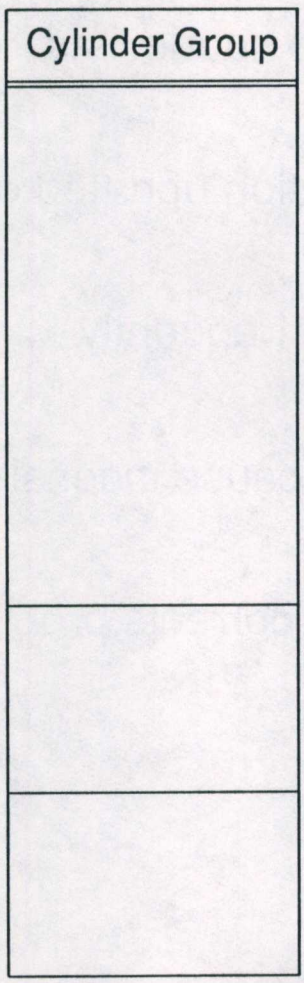
- file data blocks localized on disk
- direct address to 12 * 256 blocks of data = 1.2 meg. = 1 BL = 1 Buffload
- reduced number of seeks to access file

EFS and the System Administrator

disk fragmentation can occur:

- allocate a large file
- allocate a smaller file
- delete large file
- "hole" left that may be relatively useless
- over time, system performance reduced

Disk Fragmentation



File System Corruption

- superblock, bitmap, inode lists in memory
- in-core representation updated constantly
- disk copy updated frequently
- system crash can cause inconsistencies
- *fsck* identifies and corrects problems

Causes of Corruption

hardware failures:

- brownouts and power failures
- environment

software failures:

- if last output to disk was not *sync*-ed
- mounting a (previously) corrupted file system
- setting incorrect partition boundaries
- improper use of *mkfs* command
- running programs that (incorrectly) modify disk blocks

When to Run fsck

- automatically, if *fs_dirty* bit is set
- when booting the system
- when *fsstat(1M)* indicates a problem
- before creating backups
- after installing a new disk
- after restore or update from tape
- after using disk-level utilities (dvhtool, fx)

Running fsck on /usr

- enter single-user mode init 1
- make sure file system (usr) is not mounted df
- issue *fsck* command:

rdsk
fsck /dev/dsk/ips0d0s6 (ESDI)

rdsk
fsck /dev/dsk/dks0d1s6 (SCSI)

fsck Phases

- initialization
- Phase 1 - Inode Blocks and sizes
- Phase 2 - Pathnames
- Phase 3 - Connectivity
- Phase 4 - Reference Counts
- Phase 5 - Free List

Phase 1: Check Blocks and Sizes

Checks Inodes:

- valid inode types
- full zero-link count table
- bad or duplicate inode block numbers
- valid inode size
- valid inode format

Phase 2: Check Pathnames

Reports errors:

- root inode unallocated (serious condition!)
- directory entry points to bad inode

Prints inode info (optionally removes):

- unallocated inodes
- duplicate/bad files
- duplicate/bad directories

Phase 3: Check Connectivity

- looks for unreferenced directories

  unconnected directories put in *lost+found*

- reports if *lost+found* full or missing

Phase 4: Check Reference Counts

Reports (and optionally corrects) errors:

- unreferenced files

  (unconnected files put in *lost+found*)

- incorrect link counts

- bad or duplicate blocks

- full or missing *lost+found*

Phase 5: Check Free List

Reports errors:

- bad blocks (out of file system range)
- duplicate blocks
- missing blocks

Reports (and corrects) errors:

- bad free list
- free block count in superblock wrong

Running fsck on /

- enter single-user mode
- root file system is always mounted
- issue *fsck* command:

  fsck /dev/dsk/ips0d0s0 (ESDI)

  fsck /dev/dsk/dks0d1s0 (SCSI)

note: fsck on root always produces errors!

Tips for fsck

- do not run with *-y* option first time
- note the inode numbers reported
- manually check files (copy?)
- on second pass: -y (yes) or manually respond

  SALVAGE? FIX? CONTINUE?
  RECONNECT? ADJUST? - respond yes

  REMOVE? CLEAR? - yes with care!

- check *lost+found* for orphaned files/directories

Miscellaneous Maintenance

- monitor disk usage with *du -s, df -i*
- remove files from /tmp, /usr/tmp
- monitor files, directories that grow:

  /etc/utmp, /etc/wtmp, /usr/adm/sulog, /etc/SYSLOG

- remove inactive files (find -mtime +*numdays*)
- identify large space users (du /*homedir*)

File System Maintenance Lab

- run fsck on */usr*
- look for old files
- explore /usr/adm log files

Modem Support

Objectives:

- modem setup procedure
- overview of communication software

Modem Capabilities and Characteristics

- remote access from one system to another
- remote access from a terminal to a computer
- various communications protocols
- multiple line speeds
- dial in, dial out capabilities
- autodial, or manual dial

Modem Setup

Overview:

- physically connect to RS-232 port
- disable port in */etc/inittab*
- define port characteristics in */usr/lib/uucp/Devices*
- inform *init* with *telinit q*
- open access to port
- test port

Modem Setup:

Make Physical Connection

- RS-232 "null modem" cable
- pin configurations:

modem

IRIS	Terminal	Signals
1	1	Chassis ground
2	2	Transmit data
3	3	Receive data
4	4	Request to send, Clear to send
8	8	Carrier detect
6	20	Data set ready
20	6,22	Data terminal ready
7	7	Signal ground

Modem Setup

Modify Inittab

```
t2:x:respawn:/etc/getty ttyd2 dx_9600        # dial out port *
t3::respawn:/etc/getty ttyd3 dx_9600         # dial in port *
t4::respawn:/usr/lib/uugetty ttyd4 dx_9600   # bidirectional *
```

dial out modems:

- disable tty device *getty* process

dial in modems:

- enable tty device *getty* process

bidirectional modems:

- enable tty device *getty* process
- run */usr/lib/uugetty* instead of */etc/getty*

the /etc/gettydefs File

label : initial flags : final flags : prompt : next-label

```
console# B9600 # B9600 SANE TAB3 #\r\n\n$HOSTNAME console login: #console

co_9600# B9600 # B9600 SANE TAB3 #\r\n\n$HOSTNAME login: #co_4800

co_4800# B4800 # B4800 SANE TAB3 #\r\n\n$HOSTNAME login: #co_2400

co_2400# B2400 # B2400 SANE TAB3 #\r\n\n$HOSTNAME login: #co_1200

co_1200# B1200 # B1200 SANE TAB3 #\r\n\n$HOSTNAME login: #co_300

co_300# B300 # B300 SANE TAB3 #\r\n\n$HOSTNAME login: #co_9600

dx_9600# B9600 # B9600 SANE TAB3 HUPCL #\r\n\n$HOSTNAME login: #dx_9600

dx_4800# B4800 # B4800 SANE TAB3 HUPCL #\r\n\n$HOSTNAME login: #dx_4800

dx_2400# B2400 # B2400 SANE TAB3 HUPCL #\r\n\n$HOSTNAME login: #dx_2400

dx_1200# B1200 # B1200 SANE TAB3 HUPCL #\r\n\n$HOSTNAME login: #dx_1200

du_2400# B2400 # B2400 SANE TAB3 HUPCL #\r\n\n$HOSTNAME login: #du_1200

du_1200# B1200 # B1200 SANE TAB3 HUPCL #\r\n\n$HOSTNAME login: #du_300

du_300# B300 # B300 SANE TAB3 HUPCL #\r\n\n$HOSTNAME login: #du_2400
```

Modem Setup:

Modify /usr/lib/uucp/Devices

- defines location and line speed of modems
- one entry per modem port
- entries coordinated with *Dialers, Systems* and *Dialcodes* files
- format:

Type Line Line2 Class Dialer-Token-Pairs

- examples:

Direct ttyd3 - 9600 direct (null modem)

ACU ttyd4 - Any direct (modem)

Modem Setup:

- inform */etc/init*:

  telinit q

- set up port device:

  chmod 666 /dev/ttyd*

  chown root /dev/ttyd*

  chgrp sys /dev/ttyd*

- test serial line with *cu*:

  cu -s*baudrate* -l*device* *phonenumber*

Troubleshooting Tips

- be sure port device mode is set to 666.
- use *cu* from window manager - can kill if it hangs
- check /usr/spool/uucp for lock file (LCK-something)
- put most used ports first in /usr/lib/uucp/Devices
- *cu* use for file transfer is risky
- *kermit* recommended (has error checking)

Networking Software for Modems

cu ("call unix"):

- connect to remote computer
- transfer files, execute commands locally or remote

kermit

- like cu, but has data checking for file transfer
- terminal emulation
- readily available on a variety of hosts (pc, mini, etc.)

uucp, uuto, uupick, uux, uustat

Going Further

- configuration files:

  /usr/lib/uucp/Devices (port options for *cu* and *uucp*)

  /usr/lib/uucp/Systems (system-system connects)

  /usr/lib/uucp/Dialers (dialing with ACUs)

- resources:

  Iris System Administrator's Guide - section 8

  Owner's Guide - section 7

  man pages for *cu*, *uucp*

  Silicon Graphics Network Administration Course

Modem Setup Lab

Objectives:

- set up ports
- configure /usr/lib/uucp/Devices file
- verify functionality

TCP/IP Setup

Objectives:

- overview of TCP/IP
- procedure for configuring TCP/IP
- network security

TCP/IP Overview

- transmission control protocol/internet protocol
- generic ethernet "standard"
- supports mail, remote command execution
- remote login (virtual terminal)
- file transfer

TCP/IP Installation

- need an address
- need a local system name
- need a list of remote system names, addresses
- for more advanced stuff:

  gateways (hardware)

  networking tools (yp, sendmail)

Installing TCP/IP

- get address from "authorities"
- boot to single user mode
- name the system
- add name and address to other systems
- set access privileges
- edit configuration files:

  /etc/hosts

  /etc/hosts.equiv

  (optionally) .rhosts

a reboot

Network Documentation

- TCP/IP User's Guide
- IRIS-4D Programmer's Reference Manual
- Internet Protocol Workbook:

DDN Network Information Center
SRI International
Menlo Park, CA 94025
ph: (415) 326-6200 or (800) 235 3155

Network Addresses

- physical address:

  on ethernet board, transparent to user

- logical (internet) address

  entered in configuration files

  assigned by administrating agency

  Internet:

  Network Information Center (NIC)
  SRI International
  Menlo Park, CA 94025
  ph: (415) 326-6200 or (800) 235 3155

  ARPANET:

  Jon Postel (213) 822-1511
  University of Southern California
  Information Sciences Institute, 4676
  Adminalty Way, Marinal del Rey, CA 90291

Class 'C' Address

- four bytes
- first three bytes are "net" address

  byte #1 range: 192 to 223

  bytes #2, #3 range: 0 to 255

- fourth byte is "host" (range 0-255)

Byte 1	Byte 2	Byte 3	Byte 4
Net			Host

- example:

  193.26.31.1

192.0.0.10 *

Network Configuration Files

/etc/hosts

- maps system names and aliases to internet address
- one line for local host (testing)
- one line for each remote host

```
127.1           localhost
192.23.50.2     topgun.rob.eng   topgun
192.23.50.3     olympia.rob.doc  olympia
192.23.50.4     elvin.sgi.com    elvin
192.23.50.5     kenmall.com.pbs  kenny   ken
192.23.50.6     vienna.lib.com   vienna
```

Network Configuration Files

/etc/hosts.equiv

- list of trusted remote systems (and users)
- shares account (user) names
- enables remote users to bypass passwords
- user name must be on *both* systems
- format:

  host [user-id]

```
localhost     #for testing only
topgun
olympia
elvin  paul
kenny  ann
vienna
```

Network Configuration Files

.rhosts

- located in user's home directory
- overrides */etc/hosts.equiv*
- enables remote user to login as local user
- gives remote user same permissions as local user
- format:

  host [user-id]

```
localhost     #for testing only
elvin  paul
kenny  ann
```

Network Security

warning!

- *hosts.equiv* allows access without passwords
- *.rhosts* files override hosts.equiv
- use */.rhosts* only on physically secure systems

Network Setup Lab

- configure TCP/IP files
- test communication between systems
- secure systems

hosts
hosts.equiv
/.rhosts

SGI Hotline Support

Objectives:

- role of hotline group
- support process
- correct problem reporting
- fix distribution

Fix Distribution

- maintenance release for critical defects
- other defects in scheduled update
- release notes with each fix
- some fixes available on demand