



Devices and Filesystems

ITSC 1402/KRF

1

Devices



- A device is a generic name for any physical or logical component the kernel has to interact with
 - ◆ Physical devices include things such as disk drives, CR-ROM drives, mice, printers, and sound cards
 - ◆ Logical devices include things such as pseudo-terminals, memory, the kernel itself, and network ports
- Device files are special types of files that allow programs to interact with devices via the kernel
 - ◆ These files don't really contain any data; they are simply a means for programs to communicate with the device or kernel

ITSC 1402/KRF

2



- Device drivers are the actual code that interacts with the device
 - ◆ They provide the low-level interface between the device and the kernel
 - ◆ Device drivers may be either compiled into the kernel or dynamically loaded as modules when needed
- Most device files are kept in `/dev`
- There are a multitude of types of device files in `/dev`
 - ◆ Standard devices. These include `mem` - access to physical memory; `kmem` - access to kernel virtual memory; `null` - null device; `port` - access to I/O ports

ITSC 1402/KRF

3

Loadable Device Drivers/Modules



- Linux offers support for dynamically loadable modules for versions later than 1.2
- To enable this feature, `kernel` support must be compiled into the kernel
- The idea behind `kernel` is simple: Whenever the kernel tries to access an unavailable resource, it notifies `kernel` instead of simply returning an error
- `kernel` then attempts to retrieve the resource by loading the appropriate driver
- Supporting files include `/etc/conf.modules` or `/etc/modules.conf` depending on the version

ITSC 1402/KRF

4

Terminals and Ports



- Virtual Terminals - These are the devices associated with the console and are called `ttyX`, where `X` can be from 0 through 63
 - ◆ They are accessed by `alt->F-key`
- Serial Devices - Serial ports and corresponding dialout devices. For device `ttySX`, there is also the device `cuaX` which is used to dial out with.
- Pseudo Terminals (Non-Physical terminals) - The pseudo-terminals are `pty[p-s][0-9af]`
- Standard parallel ports. The devices are `lp0`, `lp1`, and `lp2`
- Mice - The various bus mice devices

ITSC 1402/KRF

5

Floppy Disk Drives



- Disk Devices - Floppy disk devices. The device `fd[0-7]` is the device which autodetects the format, and the additional devices are fixed format. The other devices are named as `fd#LN`.
 - ◆ The single letter L identifies the type of floppy disk
 - (d = 5.25" DD, h = 5.25" HD, D = 3.5" DD, H = 3.5" HD, E = 3.5" ED)
 - The number N represents the capacity of that format in K. The standard formats are `fd#d360`, `fd#h1200`, `fd#D720`, `fd#H1440`, and `fd#E2880`
- Devices `fd0?` through `fd3?` are floppy disks on the first controller, and devices `fd4?` through `fd7?` are floppy disks on the second controller

ITSC 1402/KRF

6

Hard Disk Drives



- Hard disks. The device `hdx` provides access to the whole disk, with the partitions being `hdx[1-20]`
 - ◆ The four primary partitions are `hdx1` through `hdx4`, with the logical partitions being numbered from `hdx5` through `hdx20` (A primary partition can be made into an extended partition, which can hold 4 logical partitions).
- Drives `hda` and `hdb` are the master and slave on the first controller
 - ◆ If using the IDE driver then `hdc` and `hdd` are the master and slave on the secondary controller
 - ◆ These devices can also be used to access IDE CD-ROMs if using the IDE driver

ITSC 1402/KRF

7

SCSI and Loopback Devices



- SCSI hard disks - The partitions are similar to the IDE disks, but there is a limit of 11 logical partitions (`sdx5` through `sdx15`). Up to 8 SCSI disks can be used.
- Loopback disk devices - These allow you to use a regular file as a block device
 - ◆ This means that images of file systems can be mounted, and used as normal
 - ◆ There are 8 devices, `loop0` through `loop7`

ITSC 1402/KRF

8

Solaris Disk Naming



- In Linux, we referred to partitions on the mass storage devices (hard drives, CD-ROMs, Zip drives) as `hda3`, or `hdc2`, or `hdd5`
- In Solaris, there doesn't seem to be a corresponding name for these devices
- Instead, there are long, strange names like `c0t0d0s3` or `c1t0d1s0` or even `c0d1s7`

ITSC 1402/KRF

9

- For disks with bus controllers, such as SCSI, disk names take the form

`cwtxqlyszj`

- └─ Slice Number
- └─ Drive Number
- └─ Physical Bus Target Number
- └─ Logical Controller Number

- For disk with direct controllers, like IDE, disk names take the form

`cxqlysz`

- └─ Slice Number
- └─ Drive Number
- └─ Logical Controller Number

ITSC 1402/KRF

10

- Audio - These are the audio devices used by the sound driver
 - ◆ These include `mixer`, `sequencer`, `dsp`, and `audio`
- While the `/dev` directory contains the device files for many types of devices, only those devices that have device drivers present in the kernel can be used
 - ◆ For example, while your system may have a `/dev/sbpcd`, it doesn't mean that your kernel can support a Sound Blaster CD
 - ◆ To enable the support, the kernel will have to be recompiled with the Sound Blaster driver included

ITSC 1402/KRF

11

Characteristics of Device Files



- If you were to examine the output of the `ls -al` command on a device file, you'd see something like:

```
brw-rw---- 1 root disk 3, 1 May 5 1998 /dev/hda1
```

- ◆ This is the device file for the first partition of `hda`

ITSC 1402/KRF

12



- There are two major differences in the file listing of a device file from that of a "normal" file
 - ◆ The first difference is the first character of the "file permissions" grouping - the file type
 - ◆ On directories this is a "d", on "normal" files it will be blank but on devices it will be "c" or "b"
 - ◆ This character indicates c for character mode or b for block mode
 - ◆ This is the way in which the device interacts - either character by character or in blocks of characters



- For example, devices like the console output (and input) character by character
 - ◆ However, devices like hard disks read and write in blocks
- The second difference is the two numbers where the file size field is normally located
- These two numbers (delimited by a comma) are the major and minor device numbers

Device Numbers



- Major and minor device numbers are the way the kernel determines which device is being used and what device driver is required
- The kernel maintains a list of available device drivers, given by the major number of a device file
- When a device file is used the kernel runs the appropriate device driver, passing it the minor device number as an argument
- The device driver determines which physical device is being used by the minor device number



- For example:


```
$ ls -al /dev/hda1
brw-rw---- 1 root disk 3, 1 May 5 1998 /dev/hda1
$ ls -al /dev/hdb1
brw-rw---- 1 root disk 3, 65 May 5 1998 /dev/hdb1
```
- What this listing shows is that a device driver, major number 3, controls both hard drives *hda* and *hdb*
 - ◆ When those devices are used, the device driver will know which is which (physically) because *hda1* has a minor device number of 1 and *hdb1* has a minor device number of 65

Finding Devices on Your System



- The */proc* file system provides access to certain values within the kernel of the operating system
- This means you can't copy files into the */proc* directory, most of the directory structure is read only
- Instead you read the files under */proc* to find out information about the configuration of your system and in particular the kernel

/proc/cpuinfo



- The file */proc/cpuinfo* contains information about the CPU of your system.

```
$ cat /proc/cpuinfo
processor       : 0
vendor_id     : GenuineIntel
cpu family    : 6
model         : 1
model name    : Pentium Pro
stepping     : 7
cpu MHz       : 199.315375
cache size   : 256 KB
fddiv_bug    : no
hlt_bug      : no
sep_bug      : no
f00f_bug     : no
fpu          : yes
fpu_exception : yes
cpuid level  : 2
wp           : yes
flags        : fpu vme de pse tsc msr pae mce cx8 apic
              sep mtrr pge mca cmov
bogomips     : 198.66
```

/proc/devices



- Another file in the proc file system is called devices that contains a list of the devices for which your kernel has device drivers

```
$ cat /proc/devices
Character devices:
1 mem
2 pty
3 tty
4 ttyS
5 cua
7 vcs
10 misc
29 fb
36 netlink
128 ptm
136 pts
Block devices:
1 ramdisk
2 fd
3 ide0
9 md
22 ide1
```

ITSC 1402/KRF

19

procinfo



- A number of Unix commands use the /proc file system including *ps*, *top* and *uptime*
- The *procinfo* command is useful for displaying system status information from /proc

```
$ procinfo
Linux 2.2.5-15smp (root@porky.devel.redhat.com) (gcc egcs-
2.91.66) #1 [sneezy.]
Memory: Total Used Free Shared Buffers Cached
Mem: 128000 99236 28764 14592 26748 57608
Swap: 136040 0 136040
Bootup: Sun Jul 15 18:36:28 2001 Load average: 0.00
0.00 0.00 1/32 15909
```

ITSC 1402/KRF

20

Why Use Device Files?



- Device files are a Unix abstraction that greatly simplifies dealing with devices
- Other operating systems require you to deal directly with the device's driver
 - ◆ This means the programmer must know the various calls and required parameters for each call that are provided by each driver
- By using device files, Unix allows your program to use the standard file access functions, such as *read*, *write* and *seek*
 - ◆ It also allows the SysAdmin to set permissions on the device files to provide another level of security

ITSC 1402/KRF

21

Creating Device Files



- There are two ways to create device files - the easy way and the hard way
- The easy way involves using the Linux command *MAKEDEV*
 - ◆ This is actually a script that can be found in the /dev directory
 - ◆ *MAKEDEV* accepts a number of parameters (check the man page)
- In general, *MAKEDEV* is run as:
/dev/MAKEDEV device
where *device* is the name of a device file

ITSC 1402/KRF

22

- Suppose you accidentally erased or corrupted your console device file (*/dev/console*)

- ◆ You'd recreate it by issuing the command:

```
# /dev/MAKEDEV console
```

- NOTE: This must be done as the root user

- What if your /dev directory had been corrupted and you lost the *MAKEDEV* script?
- In this case you'd have to manually use the *mknod* command.
 - ◆ With the *mknod* command you must know the major and minor device number as well as the type of device (character or block)

ITSC 1402/KRF

23

- To create a device file using *mknod*, you issue the command:

```
# mknod device_file_name device_type
major_number minor_number
```

- For example, to create the device file for */dev/tty0* you'd issue the command:

```
# mknod /dev/tty0 c 4 240
```
- How do you know what type a device file is and what major and minor number it has so you can re-create it?
 - ◆ The solution to most SA problems, be prepared, comes into play

ITSC 1402/KRF

24



- Being a good system administrator, you'd have a listing of every device file stored in a file kept safely on a disk
- You'd issue the command:


```
$ ls -al /dev > /mnt/device_file_listing
```

 before you lost your `/dev` directory in a cataclysmic disaster, so you could read the file and recreate the `/dev` structure
 - ◆ It might also be smart to copy the `MAKEDEV` script onto this same disk just to make your life easier



- `MAKEDEV` is only found on some Linux systems
 - ◆ Primarily Red Hat and Debian
- It relies on the fact that the major and minor devices numbers for the system are hard-coded into the script
- Running `MAKEDEV` on a non-Linux system won't work because:
 - ◆ The device names are different
 - ◆ The major and minor numbers of similar devices are different
- Solaris uses `drvconfig` or `devfsadm` (preferred)

Adding Hard Disks



- A common SysAdmin activity is adding additional storage space, usually in the form of additional hard disks
- Install the new controller (if needed) and ensure you have kernel support for it
- Attach the new drive including signal cables and power
 - ◆ Make sure they are securely attached. Flaky cable attachment is one of the major reasons for reopening the case
- Determine (or create) the special files associated with the new partitions



- Partition the new disk
- Create the appropriate filesystems on the partitions
- Check the new filesystems for consistency
- Mount the new filesystems
- Specify the new filesystems' boot-time behavior and other operations in `/etc/fstab` (`/etc/vfstab` in Solaris)

```
$ ls -l /dev/hda*
```

```
brw-rw---- 1 root disk 3, 0 May 5 1998 /dev/hda
brw-rw---- 1 root disk 3, 1 May 5 1998 /dev/hda1
brw-rw---- 1 root disk 3, 10 May 5 1998 /dev/hda10
brw-rw---- 1 root disk 3, 11 May 5 1998 /dev/hda11
brw-rw---- 1 root disk 3, 12 May 5 1998 /dev/hda12
brw-rw---- 1 root disk 3, 13 May 5 1998 /dev/hda13
brw-rw---- 1 root disk 3, 14 May 5 1998 /dev/hda14
brw-rw---- 1 root disk 3, 15 May 5 1998 /dev/hda15
brw-rw---- 1 root disk 3, 16 May 5 1998 /dev/hda16
brw-rw---- 1 root disk 3, 2 May 5 1998 /dev/hda2
brw-rw---- 1 root disk 3, 3 May 5 1998 /dev/hda3
brw-rw---- 1 root disk 3, 4 May 5 1998 /dev/hda4
brw-rw---- 1 root disk 3, 5 May 5 1998 /dev/hda5
brw-rw---- 1 root disk 3, 6 May 5 1998 /dev/hda6
brw-rw---- 1 root disk 3, 7 May 5 1998 /dev/hda7
brw-rw---- 1 root disk 3, 8 May 5 1998 /dev/hda8
brw-rw---- 1 root disk 3, 9 May 5 1998 /dev/hda9
```

Linux Partition Names



Device	Name
First floppy drive (A: in MS-DOS)	<code>/dev/fd0</code>
Second floppy drive (B: in MS-DOS)	<code>/dev/fd1</code>
First IDE drive (entire disk)	<code>/dev/hda</code>
First IDE drive, primary partition 1	<code>/dev/hda1</code>
First IDE drive, primary partition 2	<code>/dev/hda2</code>
First IDE drive, primary partition 3	<code>/dev/hda3</code>
First IDE drive, primary partition 4	<code>/dev/hda4</code>
First IDE drive, logical partition 1	<code>/dev/hda5</code>
First IDE drive, logical partition 2	<code>/dev/hda6</code>
etc	
Second IDE drive (entire disk)	<code>/dev/hdb</code>
Second IDE drive, primary partition 1	<code>/dev/hdb1</code>
etc	
First SCSI drive (entire disk)	<code>/dev/sda</code>
First SCSI drive, primary partition 1	<code>/dev/sda1</code>
etc	
Second SCSI drive (entire disk)	<code>/dev/sdb</code>
Second SCSI drive, primary partition 1	<code>/dev/sdb1</code>
etc	

Locating Disk Device Files



- One easy way is to examine the output from `dmesg`

```
$ dmesg | egrep '^([s|h]d)'
```

hda: WDC AC35100L, ATA DISK drive
hdc: WDC AC2540F, ATA DISK drive
hdd: ATAPI CD ROM DRIVE 50X MAX, ATAPI CDROM drive
hda: WDC AC35100L, 4924MB w/256kB Cache,
CHS=10672/15/63
hdc: WDC AC2540F, 515MB w/64kB Cache, CHS=1048/16/63

ITSC 1402/KRF

31

`/proc/partitions`



- Another method is to look at `/proc/partitions`, which contains a list of all the drives and partitions the system recognizes

```
$ cat /proc/partitions
```

major	minor	#blocks	name
3	0	5042520	hda
3	1	22648	hda1
3	2	1	hda2
3	5	1542208	hda5
3	6	514048	hda6
3	7	136048	hda7
3	8	2827408	hda8
22	0	528160	hdc
22	1	527152	hdc1
22	64	1073741823	hdd

ITSC 1402/KRF

32

- `/etc/mtab` holds information on which filesystems are mounted
 - ◆ Used by the `mount` command when run without options to display the currently mounted filesystems
- Solaris uses `/etc/mnttab`



ITSC 1402/KRF

33

Creating Partitions



- Partitions are created using `fdisk` or:
 - ◆ `cfdisk` – a curses-based partition editor
 - ◆ `sfdisk` – another command-line partition editor
 - ◆ Disk Druid – Red Hat's GUI partition editor
 - ◆ `parted` – the GNU partition editor
 - ◆ Disk Drake - Mandrake's partition editor
 - ◆ Partition Magic – a commercial partition editor
- `fdisk` lets you add, delete, display, and verify disk partitions
- With `fdisk`, you can easily destroy the contents of your entire disk if used carelessly

ITSC 1402/KRF

34

`fdisk`



```
Command (m for help): m
```

Command	action
a	toggle a bootable flag
d	delete a partition
l	list known partition types
m	print this menu
n	add a new partition
p	print the partition table
q	quit without saving changes
t	change a partition's system id
u	change display/entry units
v	verify the partition table
w	write table to disk and exit
x	extra functionality (experts only)

ITSC 1402/KRF

35

- First thing to do is to (p)rint your current partition table and write the information down for reference

```
Command (m for help): p
```

Disk /dev/hda: 16 heads, 38 sectors, 683 cylinders
Units = cylinders of 608 * 512 bytes

Device	Boot	Begin	Start	End	Blocks	ID	System
/dev/hda1	*	1	1	203	61693	6	Dos 16-bit >=32M

- This shows that the disk has a single DOS partition of 61693 blocks, or about 60 MB

ITSC 1402/KRF

36



- ❑ To create a new partition, use the (n)ew command

Command (m for help) : n

Command action

```
e      extended
p      primary partition (1-4)
```

p

- ❑ Here we've opted to create a new primary partition
- ❑ Disks can have up to four primary partitions
 - ◆ If you need more than four partitions, at least one must be an extended partition, which then allows you to create one or more logical partitions



- ❑ *fdisk* will then ask for the partition number to create

Partition number (1-4) : 2

- ❑ Next, enter the starting cylinder number of the new partition

- ◆ Since cylinders 204 through 683 are unused, we'll start the new partition on cylinder 204 (There is NO reason to leave space between partitions)

First cylinder (204-683) : 204



- ❑ Next *fdisk* asks for the size of the partition

- ◆ This can be given by ending cylinder number, bytes, kilobytes or megabytes

Last cylinder or +size or +sizeM or +sizeK (204-683): +80M

- ❑ Now, create any additional partitions you desire using the same steps



- ❑ When you're done, (p)rint the partition table and record the information displayed
 - ◆ You'll want the block sizes later when you create the filesystems

Command (m for help): p

Disk /dev/hda: 16 heads, 38 sectors, 683 cylinders

Units = cylinders of 608 * 512 bytes

```
Device Boot Begin Start End Blocks ID System
/dev/hda1 *    1    1  203  61693  6  Dos 16-bit >=32M
/dev/hda2          204  204  473  82080  83 Linux native
```

- ❑ Now use the 't' command to set the type of each of your partitions (if other than Linux native) and the 'a' command to make your base partition (probably /dev/hda1) bootable

Linux Partition Types



0 Empty	16 Hidden FAT16	61 SpeedStor	a6 OpenBSD
1 FAT12	17 Hidden HPFS/NTFS	63 GNU HURD or Sys	a7 NeXTSTEP
2 XENIX root	18 AST Windows swa	64 Novell Netware	b7 BSDI fs
3 XENIX usr	24 NEC DOS	65 Novell Netware	b8 BSDI swap
4 FAT16 <32M	3c PartitionMagic	70 DiskSecure Mult	c1 DRDOS/sec (FAT-
5 Extended	40 Venix 80286	75 PC/IX	c4 DRDOS/sec (FAT-
6 FAT16	41 PPC PreP Boot	80 Old Minix	c6 DRDOS/sec (FAT-
7 HPFS/NTFS	42 SFS	81 Minix / old Lin	c7 Syrnix
8 AIX	4d QNX4.x	82 Linux swap	db CP/M / CTOS / .
9 AIX bootable	4e QNX4.x 2nd part	83 Linux	e1 DOS access
a OS/2 Boot Manag	4f QNX4.x 3rd part	84 OS/2 hidden C:	e3 DOS R/O
b Win95 FAT32	50 OnTrack DM	85 Linux extended	e4 SpeedStor
c Win95 FAT32 (LB	51 OnTrack DM6 Aux	86 NTFS volume set	eb BeOS fs
e Win95 FAT16 (LB	52 CP/M	87 NTFS volume set	f1 SpeedStor
f Win95 Ext'd (LB	53 OnTrack DM6 Aux	93 Amoeba	f4 SpeedStor
10 OPUS	54 OnTrackDM6	94 Amoeba BBT	f2 DOS secondary
11 Hidden FAT12	55 EZ-Drive	a0 IBM Thinkpad hi	fe LANtsep
12 Compaq diagnost	56 Golden Bow	a5 BSD/386	ff BBT
14 Hidden FAT16 <3	5c Priam Edisk		



- ❑ Finally, use the (w)rite command to write the changes in the partition table back to disk

- ❑ REMEMBER: none of your changes will take effect until you write them to disk with the w command

Solaris *format* Command



- ❑ Solaris uses the *format* command to provide the functionality of Linux's *fdisk*
- ❑ Syntax is:
 - ◆ *format /dev/rdisk/c##t##d##s#* or
 - ◆ *format /dev/rdisk/c##d##s#*

ITSC 1402/KRF

43

Which FS's Need Their Own Partition?



- ❑ Depends upon how the system will be used
 - ◆ */var* contains spool directories and the error log directory. If the machine develops a chronic error, these messages can fill up the partition so it ought to be in a different partition than */*
 - ◆ */usr* is where most executables are located
 - ◆ */tmp* can also grow rapidly so placing it somewhere besides */* is usually a good idea
 - ◆ */home* is where the users home directories go. Having it as a separate partition can make maintenance easier as well as protecting the root directory from getting filled up
 - ◆ */boot* used to be needed since LILO couldn't see past cylinder 1024 but this has been fixed in later versions

ITSC 1402/KRF

44

Example: 10 Gbyte Dual Boot Box



Partition	Mount Point	Size
/dev/hda1	/boot	15 Mbytes
/dev/hda2	Windows 98 partition	2 Gbytes
/dev/hda5	swap space	64 Mbytes
/dev/hda6	/tmp	50 Mbytes
/dev/hda7	/	150 Mbytes
/dev/hda8	/usr	1.5 Gbytes
/dev/hda9	/home	rest of drive (~6.75 Gbytes)

ITSC 1402/KRF

45

Creating Filesystems



- ❑ Before you can use a partition, you must create a filesystem on it
- ❑ Creating a filesystem is analogous to formatting a partition under MS-DOS
- ❑ There are several filesystem types available under Linux
 - ◆ Each has its own set of characteristics, such as filename length, maximum file size, etc.

ITSC 1402/KRF

46

❑ The native Linux filesystem is the Third Extended Filesystem (ext3fs)



- ◆ It is one of the most efficient and flexible of the supported filesystems
- ◆ It allows file names up to 256 characters and filesystems of up to 4 terabytes and is a journaling filesystem
- ❑ To create an ext3fs filesystem, enter
 - ◆ *mkfs.ext3 -c* partition size
 - # *mkfs.ext3 -c /dev/hda2 82080*
- ❑ Read the *mkfs.ext3*, *mkfs*, and *mke2fs* man pages

ITSC 1402/KRF

47

❑ If you are using multiple partitions, you will have to run *mkfs* or *mkfs.ext3* on each one individually



- ◆ Except for the swap partition which has its own filesystem creation command
- ❑ If you are using different types of filesystems, you will have to run the appropriate "*mk fs*" command for each one

ITSC 1402/KRF

48

Major Linux Filesystem Types



Filesystem	Type Name	Description
Third Extended Filesystem	ext3	Journaling filesystem
Second Extended Filesystem	ext2	Native Linux filesystem
Reiser Filesystem	rfs	Balanced tree journaling filesystem
xfs	xfs	SGI's Journaling filesystem
Journalized Filesystem	jfs	IBM's journaling filesystem
Extended Filesystem	Ext	Superseded by ext2
Minix Filesystem	minix	
UMSDOS Filesystem	umsdos	Used to install Linux on an MS-DOS partition
MS-DOS Filesystem	msdos	Used to access MS-DOS files
/proc Filesystem	proc	Provides process information for ps, an "in-memory" filesystem
ISO 9660 Filesystem	iso9660	Used for most CD-ROMs
System V Filesystem	sysv	Used to access System V variants

ITSC 1402/KRF

49

Checking/Repairing Filesystems



- ❑ `fsck` is used to check and repair filesystems
 - ◆ Must be run on an unmounted partition
 - ◆ `mount mount_point`
 - # `mount /usr`
- ❑ Syntax: `fsck options device`
 - # `fsck -f -y /dev/hdb1`

Option	Meaning
-f	Force a check even if the FS doesn't appear to need one
-p	Preen the FS, automatically fixing all problems that can be safely corrected without data loss
-y	Answer "yes" to all prompts; may cause data loss if the FS has errors
-b #	Use the specified backup superblock

ITSC 1402/KRF

50

Mounting Filesystems



- ❑ Once you've created your filesystem, you must mount it before it can be used
- ❑ This is done with the `mount` command
 - ◆ `mount -t fstype -o options device mount_point`
 - # `mount -t ext2 /dev/hda2 /usr`
 - ◆ options control such things as whether the filesystem should be read only, when it should be checked for consistency, and whether execution of binaries is allowed from the partition – see the man page for `mount`

ITSC 1402/KRF

51

Mounting Filesystems at Boot



- ❑ Entries in `/etc/fstab` (`/etc/vfstab` for Solaris) control how filesystems are mounted at boot time
- ❑ Example `/etc/fstab` file:

```
# device mount point FS type mount options dump fsck
# device mount point FS type options freq pass
/dev/hda8 / ext2 defaults 1 1
/dev/hda1 /boot ext2 defaults 1 2
/dev/hda6 /home ext2 defaults 1 2
/dev/hdc1 /jpeg ext2 defaults 1 2
/dev/hda5 /usr/local ext2 defaults 1 2
/dev/hda7 swap swap defaults 0 0
/dev/fd0 /mnt/floppy ext2 noauto 0 0
/dev/cdrom /mnt/cdrom iso9660 noauto,ro 0 0
none /proc proc defaults 0 0
none /dev/pts devpts mode=0622 0 0
```

ITSC 1402/KRF

52

Default Mount Options



Option	Meaning (opposite)
rw	Filesystem is read-write (<code>ro</code>)
auto	Include this FS when using <code>mount -a</code> (<code>noauto</code>)
nouser	Don't allow ordinary users to mount (<code>user</code>)
noexec	Respect SETUID/SETGID settings (<code>exec</code>)
noexec	Allow execution of binaries (<code>exec</code>)
nogrpuid	Use SysV group ownership rules unless the directory's SETGID bit is set (<code>grpuid</code>)
dev	Interpret corresponding device files (<code>nodev</code>)
async	Perform I/O asynchronously (<code>sync</code>)
atime	Update file access times (<code>noatime</code>)

ITSC 1402/KRF

53

Swap



- ❑ Swap space is a generic term for disk storage used to increase the apparent amount of available memory
- ❑ Under Linux, swap is used to implement paging, a process where memory is written to disk, a page at a time, when physical memory is low, and read back from disk as needed
- ❑ A major limitation of swap is speed
 - ◆ Disk access is much, much slower than physical memory access

ITSC 1402/KRF

54

Types of Swap Space



- Linux supports two forms of swap areas
 - ◆ A separate disk partition
 - ◆ A file on an existing partition
- You can have up to 16 swap areas, with each swap area being a file or partition up to 128 MB in size
- Using a dedicated swap partition will generally provide improved performance
 - ◆ All the disk blocks are guaranteed to be contiguous
 - ◆ Linux can manage the raw partition
- Swap files are generally used when someone needs additional temporary swap space

ITSC 1402/KRF

55

The *free* Command



- *free* will tell you how much swap space you have

```
sneezy% free
```

```
      total    used    free   shared    buffers
Mem: 19308  17672   1636     8012    10820
Swap:16408         0   16408
```

- All the numbers are reported in 1024 (1K) byte blocks
- Note that the system has more physical memory than reported
 - ◆ *free* does not report on the memory used by the kernel

ITSC 1402/KRF

56

- The *shared* entry shows the physical memory that is shared between multiple processes
- The *buffers* entry shows the amount of physical memory in use by the kernel buffer cache
 - ◆ The buffer cache is used to speed up disk accesses by caching data in physical memory
 - ◆ The buffer cache will increase and decrease in size as memory use changes
 - The buffer cache memory is reclaimed if the physical memory is needed by applications

ITSC 1402/KRF

57

- The *swap* entry shows the total number of 1K byte blocks in swap space
- The *used* entry shows that none of the swap space is currently being used
 - ◆ The system has enough physical memory available to supply all the currently executing processes needs
 - ◆ If additional physical memory was needed, the kernel would first reduce the amount of buffer cache in use
 - ◆ Swap space is generally used as a last resort to meet memory needs because of its negative performance impact

ITSC 1402/KRF

58

- Note that the amount of swap space reported by *free* is less than the total size of your swap files and partitions
 - ◆ Several blocks of each swap space are used to store a map of how each page in the swap area is being used

ITSC 1402/KRF

59

Creating Swap Space



- First step is to create a file or partition to hold the new swap space
- To create a swap file, you must first create a file of the desired size
- An easy way to do this is with the *dd* command
 - ◆ To create an 8 MB swap file enter:

```
dd if=/dev/zero of=/swap bs=1024 count=8192
```

- ◆ After creating the file, use the *sync* command to synchronize the filesystems

ITSC 1402/KRF

60



- Once the file has been created, use the *mkswap* command to “format” the swap area
 - ◆ `mkswap -c device size`
 - ◆ In the case of our example file:


```
# mkswap -c /swap 8192
```
 - ◆ The '-c' is optional; it causes *mkswap* to check for bad blocks as it formats
- After using *mkswap* on a swap file, you should again issue the *sync* command to synchronize the file systems

Swap Partition



- To create a swap partition, you use *fdisk* just like we did earlier, only this time you set the partition (t)ype to swap
- After the updated partition table has been (w)ritten back to disk, use the *mkswap* command to “format” the swap partition
 - ◆ `mkswap -c device size`
 - ◆ For example, to format the first extended partition of the first IDE drive as a 128 MB swap partition enter:


```
# mkswap -c /dev/hda5 131072
```

Enabling/Disabling Swap Space



- Once the swap space has been created and formatted with *mkswap*, it must be enabled
- This is done with the *swapon* command
 - ◆ `swapon <device or file>`
 - ◆

```
# swapon /dev/hda5
```

 or

```
# swapon /swap
```
 - ◆ `swapon -s` will show current swap areas
- To disable swap space, the *swapoff* command is used
 - ◆ `swapoff <device or file>`
 - ◆

```
# swapoff /dev/hda5
```

 or

```
# swapoff /swap
```

Mounting Swap Space on Boot



- To have the system automatically mount the swap area when the system boots, an entry must be made in */etc/fstab* just like for any other filesystem
- If */etc/fstab* contains the following entries

```
#devices  directory  type options
/dev/hda5  none      swap sw
/swap     none      swap sw
```

then two swap areas, */dev/hda5* and */swap*, will be enabled at boot time

Where Are My Files?




- Historically, the location of certain files has not been standard
 - ◆ So, where is *ifconfig* on **this** system?
- Unfortunately, this is still somewhat the case between different flavors of Unix
- To combat this, at least within the Linux community, a file standard was developed
 - ◆ Distros that conform to the file standard become easier to administer and to develop software for

The Directories in /




Directory	Contents
/bin	Required boot-time binaries
/boot	Boot configuration files for the loader and the kernel image
/dev	Device files
/etc	System configuration files and startup/shutdown scripts
/home	User directories
/lib	Main location for for shared OS libraries and kernel modules
/lost+found	Storage for "recovered" files after bad things have happened to your filesystem



Directory	Contents
/mnt	Temporary point to connect devices, such as the floppy or CD-ROM
/proc	Pseudo directory structure containing info about the kernel, currently running processes, and resource allocations
/root	Linux (non-standard) home directory for the root user. An alternative is /
/sbin	System administration binaries and tools
/tmp	Location for temporary files
/usr	Contains almost everything else – local binaries, libraries, applications, X Windows
/var	Variable data such as spool directories

ITSC 1402/KRF 67


home for root



- ❑ `/root` is the home for the root user
 - ◆ If the `/root` directory doesn't exist, the root user will be logged into /
- ❑ It is debatable if root should have a separate home directory directly off /
 - ◆ Most other Unix flavors do not do this
 - ◆ It does encourage root to set up their `.profile`
 - ◆ It also provides a place for programs like `elm`, `pine`, `tin`, and `netscape`
 - They all require a home directory in which to store configuration files

ITSC 1402/KRF 68


`/usr` and `/var`



- ❑ These directories often contain similar subdirectories
 - ◆ There are also a number of links that refer back and forth between the two directory structures
- ❑ This is primarily historical and the links are in place to reflect the "proper" location of many files
- ❑ `/var` is for variable data and files
 - ◆ News, mailer and print spool files, log files
- ❑ `/usr` is for user accessible data, programs and libraries

ITSC 1402/KRF 69


`/usr/local`



- ❑ This is where all add-on software packages should be installed, with the binaries in `/usr/local/bin`
- ❑ By keeping all add-on software in a central location, it makes backups much simpler
 - ◆ The entire `/usr/local` directory tree can be backed up and restored much more easily than getting `/usr/bin`, `/usr/src`, `/usr/lib`, etc individually
- ❑ Of course, Sun has deprecated the use of `/usr/local` and prefers `/opt` instead

ITSC 1402/KRF 70


`lib`, `src`, and `include`



- ❑ Linux is a very popular development platform
 - ◆ `/usr/local/lib`, `/usr/local/src`, and `/usr/local/include` are very important for that
 - ◆ `lib` holds most static libraries as well as subdirectories containing libraries for other languages, such as `perl` and `Tcl`
 - ◆ `include` contains the C/C++ header files
 - ◆ `src` contains source files for most installed packages

ITSC 1402/KRF 71

`/var/spool`



- ❑ This directory can cause a lot of grief since it is used to store (potentially) large volumes of temporary data associated with email and printing
- ❑ If a printer is malfunctioning, or someone has received a large quantity of mail, disk space can be rapidly consumed
 - ◆ This is the primary reason for giving `/var` its own partition

ITSC 1402/KRF 72

Binaries



- There are four major directories for storing binaries and they should not be used for other purposes
 - ◆ `/bin`
 - ◆ `/sbin`
 - ◆ `/usr/bin`
 - ◆ `/usr/local/bin`
- `/bin` contains utilities used during boot and must be present or the system will not boot

ITSC 1402/KRF

73

- `/sbin` contains the essential system administration utilities and scripts
 - ◆ As a general rule, if a regular user needs to access a script or utility, it shouldn't be kept in `/sbin`
- `/usr/sbin` is used for non-essential system administration tools
- `/usr/local/sbin` is used to store locally installed system admin tools
- `/usr/bin` contains binaries for most of the programs normal users will run
- `/usr/local/bin` contains binaries for add-on software that's not part of the OS package

ITSC 1402/KRF

74

Configuration Files and Logs



- The `/etc` directory structure is a place where the system administrator will spend a lot of time
- Not only does the `/etc` directory branch contain the password and shadow files but it also contains just about every configuration file for the system
- It also contains:
 - ◆ The startup and shutdown scripts
 - ◆ The skeleton user files, such as `.profile`, `.cshrc`, etc. that are placed in a new user's account (`/etc/skel`)
 - ◆ Configuration files for X Windows (`/etc/X11`)

ITSC 1402/KRF

75

- Linux keeps system logs in `/var/log`
 - ◆ This generally contains logs for `cron`, the `dmesg` file, mail logs, login information logs, and any other logs that are being maintained
- `/proc` is a pseudo directory that contains pseudo files associated with the executing kernel
 - ◆ These "files" contain info about the state of the system
 - Resource usage (how much memory, swap, and CPU time is being used)
 - Information about each process
 - Other useful kernel-level information
- `/dev` is the directory that contains device files

ITSC 1402/KRF

76