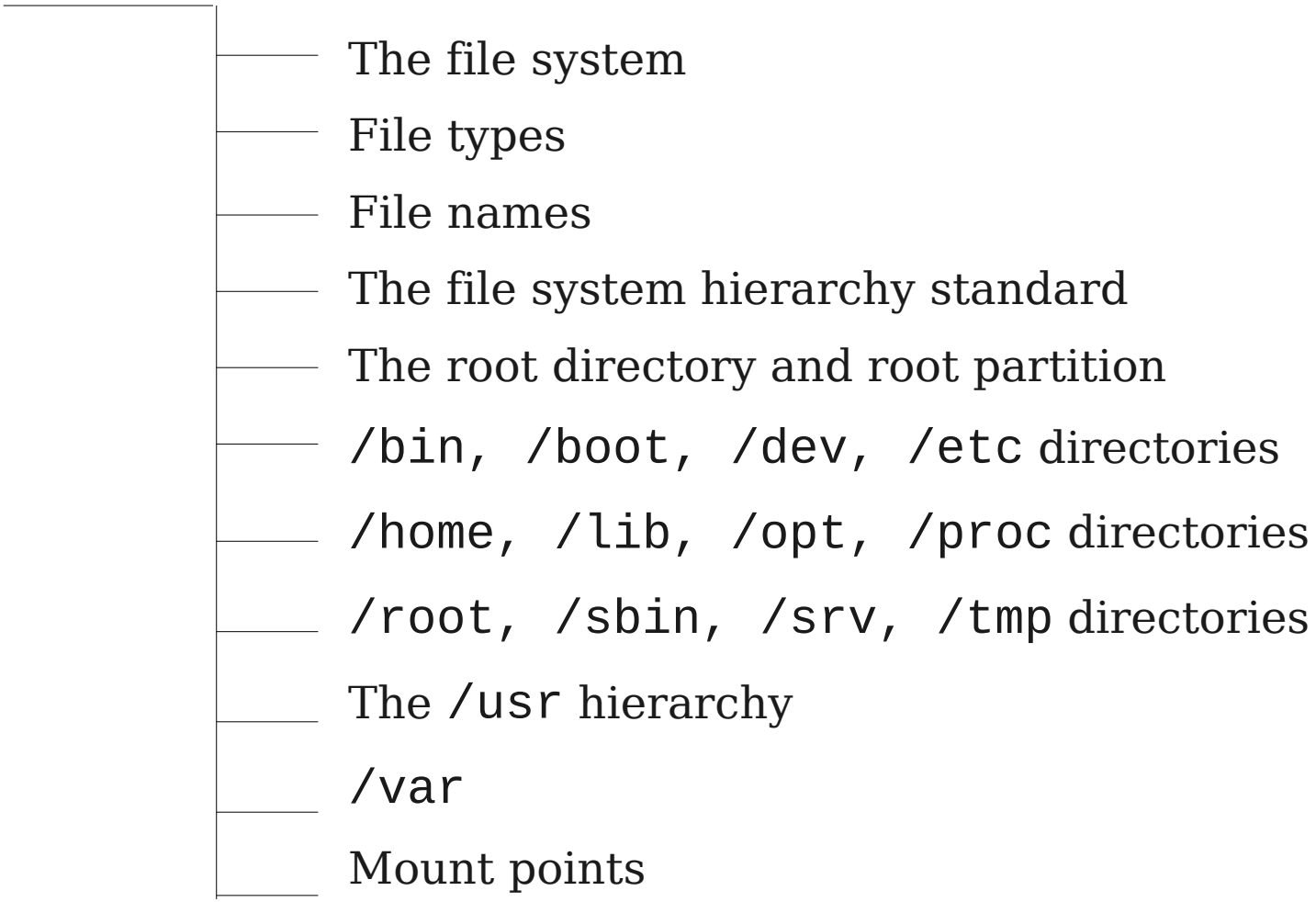
Chapter 4
File system management

Chapter 4 Outline

- In this chapter we will learn about:
  - ✓ The standard hierarchy of the file system
  - ✓ Local and network file system types
  - ✓ How to create and manage disk partitions
  - ✓ How to mount partitions automatically

The file system hierarchy

- The file system hierarchy



The file system

- The file system is organised in a hierarchy (tree)
  - The “root directory” (/) is at the top of the tree
- Pieces of the filesystem may exist on multiple disk partitions or on remote file servers on the network
  - The pieces are 'mounted' onto directories to make the file system appear as a single tree

File types

- Several types of object exist in the filesystem
- Normal files
  - A set of contiguous data identified by a name
  - Includes text files, graphics files, executable programs, etc;
  - the filesystem does not distinguish the type of data
  - '`.`' is not a special character in file names and the filesystem does not recognise separate names and extensions (e.g. `report.txt`), though many applications do use an extension to identify the type of data in the file
- Directories
  - Directories contain named 'links' to other files
  - They cannot be opened, read and written like ordinary files

File types (continued)

- Device Files
  - Devices (disks, tape drives, mice, etc) are identified by device file entries which are usually in the /dev directory
  - If access permissions allow, some device files may be opened, read and written like ordinary files (for example an archive may be written directly to a tape device)
- Symbolic Links
  - References to files located at other points in the file system
  - Allow a single file to be referenced using multiple names
  - Symbolic links can be opened like regular files, but the operation is automatically redirected to the file that the link points to
- Sockets and FIFOs (named pipes)
  - Named communication end points used for interprocess communication, not covered in this course

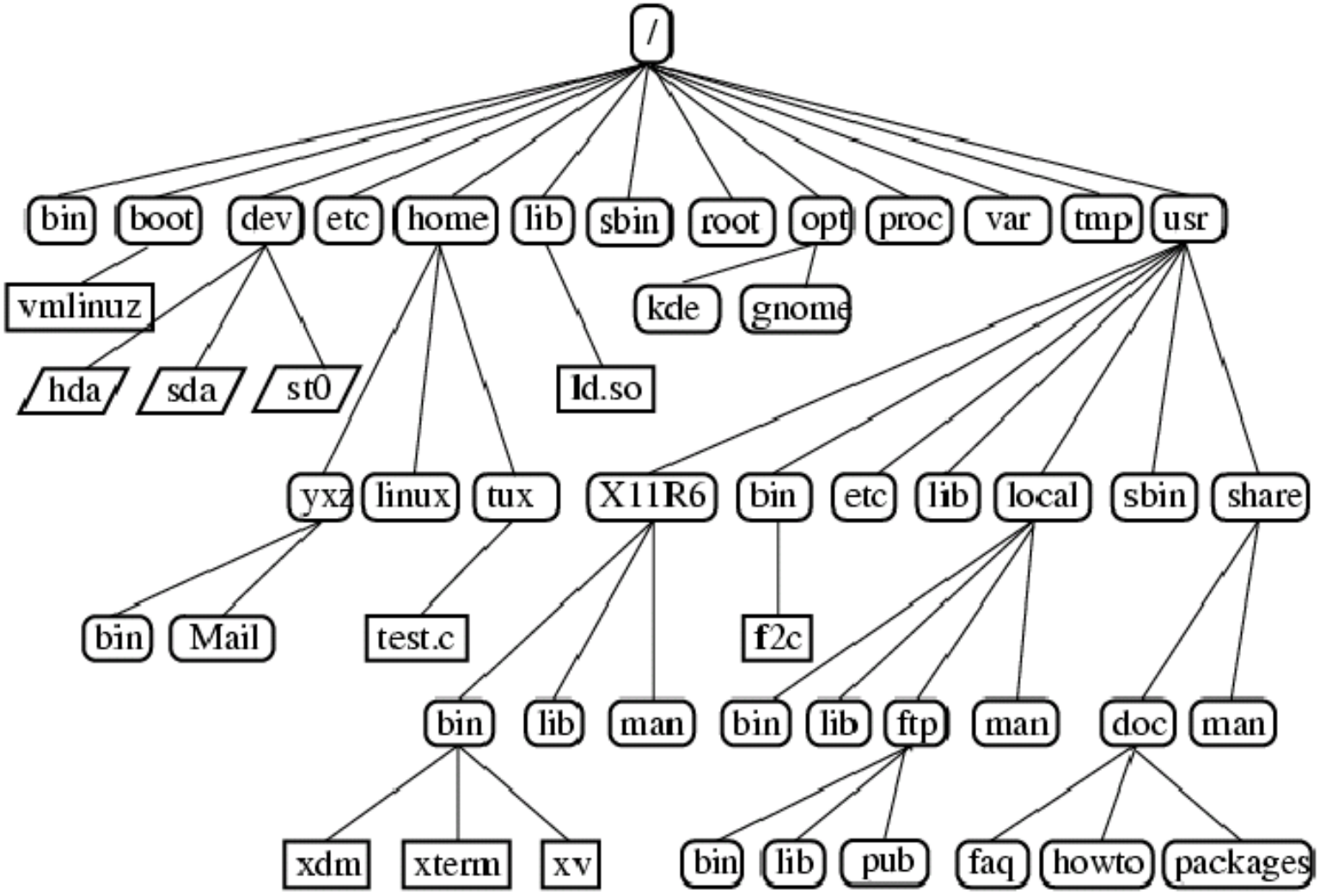
File names

- File names can be up to 255 characters long
  - Case sensitive
- All characters except '/' are legal in filenames
  - '/' is used as a separator in path names
- Some characters have special meaning to the shell. They are awkward to work with in file names and are best avoided:
  - `+, %, $, #, !, \, -, ~, =`, space, others ...
  - Recommend use only upper and lower case letters, digits, and '_'
- Maximum length of a path name is 4096 characters

FHS – Filesystem Hierarchy Standard

- The Filesystem Hierarchy Standard (FHS) is a vendor-independent guideline that specifies the layout of the upper levels of the file system tree
  - See `www.pathname.com/fhs`
  - Aims to provide consistency across UNIX versions
  - SuSE linux conforms quite closely to the FHS guidelines
- FHS distinguishes two major characteristics of files
  - Sharable (across multiple machines) vs non-sharable
  - Static (do not change without system admin intervention) vs dynamic
- Files that differ in either respect should be in different directories
- In the following slides we will tour some of the important directories specified by the FHS

Filesystem hierarchy



ict@innovation

FOSSFA

Root directory and root partition

- The *root directory* is the top level directory of the tree
  - Do not confuse this with the home directory of the superuser, which is usually `/root`
- The *root partition* is the partition containing the root directory
  - At boot time, the root partition is initially the only one mounted
  - Files needed at boot time must be in directories on the root partition
    - `/bin, /dev, /etc, /lib and /sbin`

Binary directory: /bin

- /bin contains important executable (binary) programs
  - Needed early in boot sequence
  - Needed for emergency maintenance if other partitions unavailable
  - Includes shells, filesystem maintenance commands, other tools
  - Contents include:

File	Description
/bin/bash	The bash shell
/bin/cat	Display / concatenate files
/bin/cp	Copy files
/bin/mv	Rename files
/bin/rm	Remove files
/bin/mount	Mount file systems
/bin/vi	Text editor
/bin/tar	Tape archiver

Boot directory: /boot

- /boot contains the files needed to get the linux kernel up and running
  - Second-stage files for the boot loader (GRUB or LILO)
  - Backup of Master Boot Record
  - The image of the linux kernel
- Sometimes /boot is on a separate partition
  - Early stages of booting on a PC rely on the PC's BIOS firmware to access the hard drive.
  - On early PCs the BIOS could not access cylinder numbers beyond 1023 so it was essential that the kernel image lay below this boundary
  - Putting /boot on a separate (small) partition guarentees this

Device files: /dev

- Device files give file names to hardware devices
  - Associates a name (e.g. /dev/hda1) with a major / minor device number
  - Identifies the device and the driver used to read/write data on the device

```
earth:~ # ls -l /dev/hda1
brw-rw----  1 root   disk   3,  1 2003-09-23 18:59 /dev/hda1
```

b = block device (disks)
c = character device
(printer, mouse, tape etc)

Minor device number
Major device number

- Usually all required device files are created automatically
  - Many reference hardware which is not actually present
  - The mknod command is used to create new device files

Device files: /dev (continued)

Device	Device File	Description
Terminals	/dev/console /dev/tty*	The system console Virtual terminals
Serial ports	/dev/ttyS0 /dev/ttyS*	ttyS0 is equivalent to DOS “COM1”
Parallel ports	/dev/lp0 /dev/lp*	lp0 is equivalent to DOS “LPT1”
Floppy disks	/dev/fd0	The first floppy disk drive
IDE hard drives	/dev/hda /dev/hdb /dev/hdc	Master drive on primary IDE controller Slave drive on primary IDE controller Master drive on secondary IDE controller
IDE CDROM drives	/dev/hd*	IDE CD Rom drives are named within the same scheme used for IDE hard drives
SCSI hard drives	/dev/sda /dev/sdb /dev/sda1	The first SCSI drive The second SCSI drive First partition on first SCSI drive
SCSI CDROM drives	/dev/scd0	First SCSI CDROM drive

Configuration files: /etc

- /etc contains system configuration files
  - mostly plain text; editable using any text editor

File	Description
/etc/SuSE-release	Version number of the SuSE product installed
/etc/inittab	Master configuration file for the init process
/etc/init.d/*	Scripts for starting services
/etc/grub.conf	Configuration file for GRUB bootstrap loader
/etc/modules.conf	Configuration file for the kernel modules
/etc/X11/XF86Config	Configuration file for the X server
/etc/fstab	Table of filesystems mounted automatically at boot time
/etc/profile	System-wide login script for the shell
/etc/passwd	Database of locally defined user accounts
/etc/shadow	Database of encrypted user passwords
/etc/group	Database of locally defined groups
/etc/hosts	Local mapping of machine names to IP addresses
/etc/motd	Message of the day: Message printed after login
/etc/sysconfig/*	Directory containing central system config files

User directories: /home

- /home contains the *home directories* of individual users
  - After login, a user's current directory is his home directory
- /home is often on a separate partition
  - or may be mounted from a file server – this arrangement means that users are not tied to specific machines
- A user's personal configuration files are stored in his home directory
  - “Hidden” files (name starts with '.')
  - Examples: `.bashrc, .profile, .bash_history, .xinitrc`
- The shell recognises '~' as a shorthand for your home directory
  - Example: `~/.bashrc`

Libraries: `/lib`

- A *library* is a collection of compiled binary files
  - Contains code for functions used by many programs
  - A program that needs access to library functions links to the required libraries at run time (dynamic linking)
  - UNIX refers to dynamically linked libraries as *shared objects* (`.so` files)
- `/lib` contains the essential system libraries needed to run the commands in the root filesystem i.e. `/bin` and `/sbin`
  - `/lib/libc.so.6` is the main C runtime library
  - Most other application libraries are in `/usr/lib`
- `/lib/modules` contains dynamically loaded kernel modules

Directory of applications /opt

- The /opt directory is used to store the 'static' files of additional applications such as Netscape, Gnome, KDE, or Open Office

```
$ ls /opt
cxoffice  gnome  kde3  mozilla  OpenOffice.org
```

- Note that the open source community does not always follow this guideline. When applications are installed by compiling from source code, the default installation directory is usually /usr/local
  - This can be changed using command line options when the application is installed

Process files /proc

- The files in /proc are a figment of the kernel's imagination
  - Make internal kernel information available via normal file read commands
  - There is a subdirectory for each process, named after the process ID
  - Other 'files' in /proc provide information about the system as a whole

File	Description
/proc/cpuinfo	Information about the processor
/proc/interrupts	Allocation of interrupt vectors
/proc/ioports	Allocation of I/O port addresses
/proc/modules	Active kernel modules
/proc/partitions	The disk partitions and I/O statistics
/proc/mounts	Mounted file systems
/proc/fileystems	Supported file system formats
/proc/version	The kernel version
/proc/pci	PCI bus devices
/proc/sys/*	System and kernel information

Directory of the administrator `/root`

- The super-user's home directory is `/root`
  - On the root partition
  - Allows root to login even if no additional partitions can be mounted – the home directories for ordinary accounts (under `/home`) may be on a separate partition

System administration commands: /sbin

- /sbin contains binaries essential for booting, restoring, recovering, configuring or repairing the system
  - Usually only root can run these programs to make changes to the system
  - /sbin lies in the root partition

File	Description
/sbin/SuSEconfig	This SuSE-specific tool performs overall system configuration, reading the files in /etc/sysconfig
/sbin/conf.d/*	The individual configuration scripts invoked by SuSEconfig
/sbin/yast2	The SuSE graphical system administration tool
/sbin/fdisk	Tool for creating and modifying disk partitions
/sbin/fsck	Tool for checking the consistency of the file system
/sbin/init	The first process created when the system boots
/sbin/ifconfig	Tool to configure network interface
/sbin/modprobe	Tool to manually add or remove kernel modules

Server directories and temporary area

- `/srv` contains site-specific data which is served by this system
  - Typically broken down into subdirectories based on the protocol used to serve the data, e.g. `/srv/www` and `/srv/ftp`
  - Note that many linux systems do not follow this FHS guideline
- `/tmp` contains temporary files
  - Anyone can create files in `/tmp`
  - Programs should not rely on files surviving in `/tmp` between one program invocation and the next
  - Some administrators set up the system to empty `/tmp` at boot time
  - Important to avoid name clashes in `/tmp` – some applications include their process ID within the file name

The /usr hierarchy

- The FHS standard defines a second level of directories under /usr
  - Sharable (across machines); “static” files that do not normally change
  - Often on a separate partition; may be mounted read-only
  - Subdirectories include:

File	Description
/usr/X11R6	Files for the X11 window system
/usr/bin	Most “regular” executable programs are here
/usr/lib	Most libraries are here
/usr/local	Locally installed tools and software may be placed here to avoid being removed or replaced during a system upgrade. Many open source packages install into /usr/local by default
/usr/sbin	Non-essential binaries used by the system administrator
/usr/share	Architecture-independent sharable data e.g. Man pages and other documentation

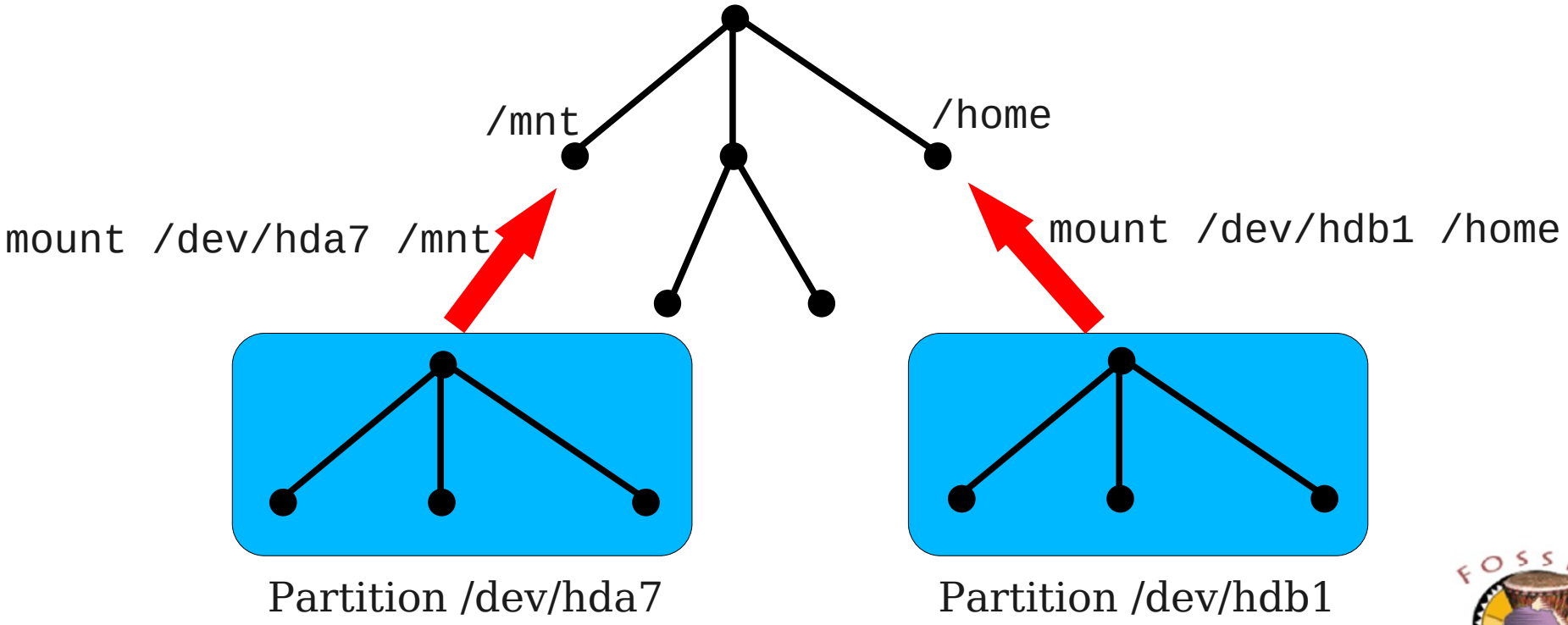
Changeable files: /var

- /var contains data files that change during normal system operation
  - Spool directories and files
  - Administrative and logging data
  - Transient and temporary files
  - Key subdirectories include:

File	Description
/var/lib	Long-term state information held by applications. For example, /var/lib/rpm contains the database of all installed packages
/var/log	Most log files live here
/var/run	Various files describing the state of the system since it was booted. For example, /var/run/cron.pid is a file containing the process ID of the cron daemon
/var/spool	Directory for spool queues (printer and mail susbsystems, etc)
/var/lock	Lock files that prevent simultaneous use of a device by more than one application

Mount points

- The file system is often split across several disk partitions
  - Each partition has a self-contained file system structure
  - Directories (“mount points”) are created on the root filesystem
  - The `mount` command attaches a partition's filesystem to a mount point
  - The `umount` command detaches a filesystem



Pre-defined mount points

- The (empty) directory `/mnt` is provided as a general-purpose mount point for temporary mounts
- Directories under `/media` are provided for mounting removable media
  - `floppy` Floppy disks
  - `cdrom` CDRoms
  - `dvd` DVDs
  - `cdrecorder` CD Recorders
  - `sda1` SCSI disk or devices emulated as SCSI
    e.g. USB memory sticks
- By default, only root can mount or unmount
  - But the system is usually configured to allow non-root users to mount removable media

File system hierarchy quiz

- Where would you expect to find:
  - The configuration file for the samba file server
  - Dilbert's home directory
  - The superuser's home directory
  - The “spool files” that hold incoming mail
  - The bootable linux kernel image
  - The executable for a regular command such as `less`
  - A file system mounted from a floppy disk
  - The device entries for the partitions on your hard drive
- True or false?
  - The `mount` command must be in the root partition
  - The FHS standard is specific to SuSE linux

File system types

- File system types
  - Native linux file system formats
  - File system formats of other systems
  - Network file system formats
  - The ext2 file system
  - The Reiser file system
  - The virtual file system

Native linux file system formats

- Linux supports many file system formats
- ext2
  - Probably the most common 'native' linux file system format
- ext3
  - Extension of ext2 to support journaling, backwards compatible
  - Much faster to check file system consistency after a crash
- Reiser File System
  - A more modern journaling file system, not compatible with ext2
  - Larger administrative overhead, do not use on very small partitions
- Others
  - XFS (from Silicon Graphics), JFS (from IBM)
- Can choose file system types at installation time

File system formats of other systems

- Linux also supports the native file systems of other operating systems
  - Useful on dual-boot systems
- FAT, VFAT
  - Used on floppies, and Windows 95/98
- NTFS
  - Used on Windows NT, 2000, XP
  - Under linux, only read access is supported reliably
- HPFS
  - Native file system of OS/2
- CDROM
  - ISO9660 with Joliet and Rockridge extensions

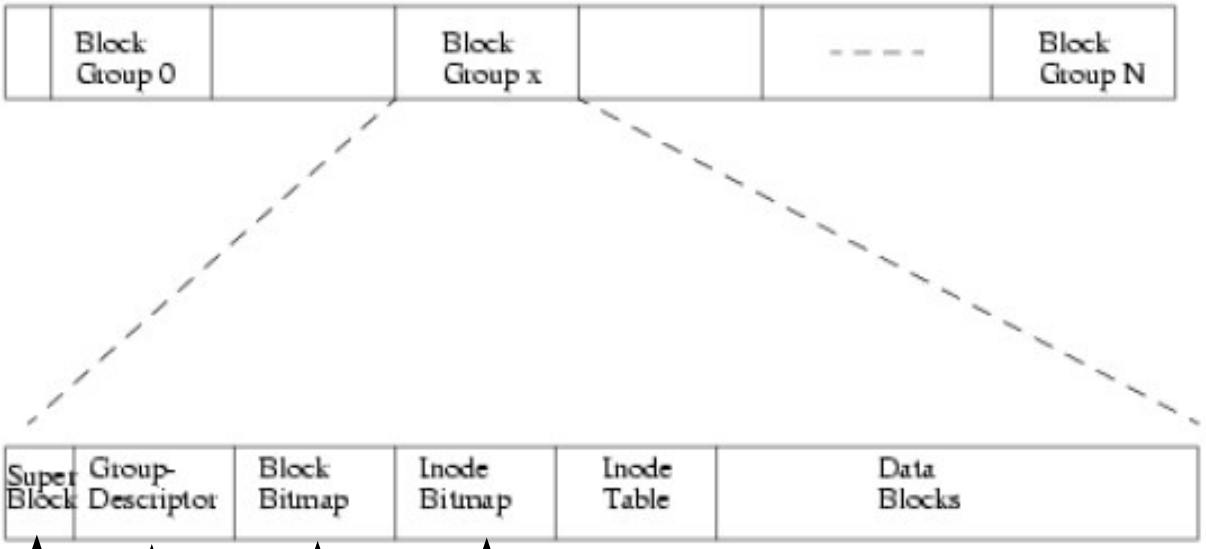
Network file system formats

- Linux can also mount file systems from remote file servers using a number of file sharing protocols
- NFS (Network File System)
  - The native UNIX file sharing protocol, originally from Sun Microsystems
- SMB (Server Message Block)
  - The native Windows file sharing protocol, supported on linux by the Samba package
- NCP (Network Core Protocol)
  - The native netware protocol

The ext2 file system format

- The inode table is created when the file system is created
  - Cannot be extended later
  - *inode density* determines number of inodes; typically one per 4096 bytes of space on the partition
  - It's possible to run out of inodes before running out of data blocks
    - If partition is storing many small files
    - May need to choose a higher inode density
- Remaining space on partition is divided into blocks
  - Typically 4096 bytes
  - Unit of allocation of disk space
  - Large block size and many small files can waste lots of space
- Size limitations
  - Maximum file size 2Tb if using 4096-byte blocks
  - File names up to 255 characters

The ext2 file system format (continued)



The filesystem is divided into multiple *block groups*

Bitmap of used / free inodes
Bitmap of used / free data blocks
Specifies location of other components of this block group

The superblock is replicated in each block group.
Contains:
- number of free and occupied blocks and inodes
- Usage information (e.g. count of mounts since last fsck)
- other data relating to the filesystem as a whole

The “lost property office” `lost+found`

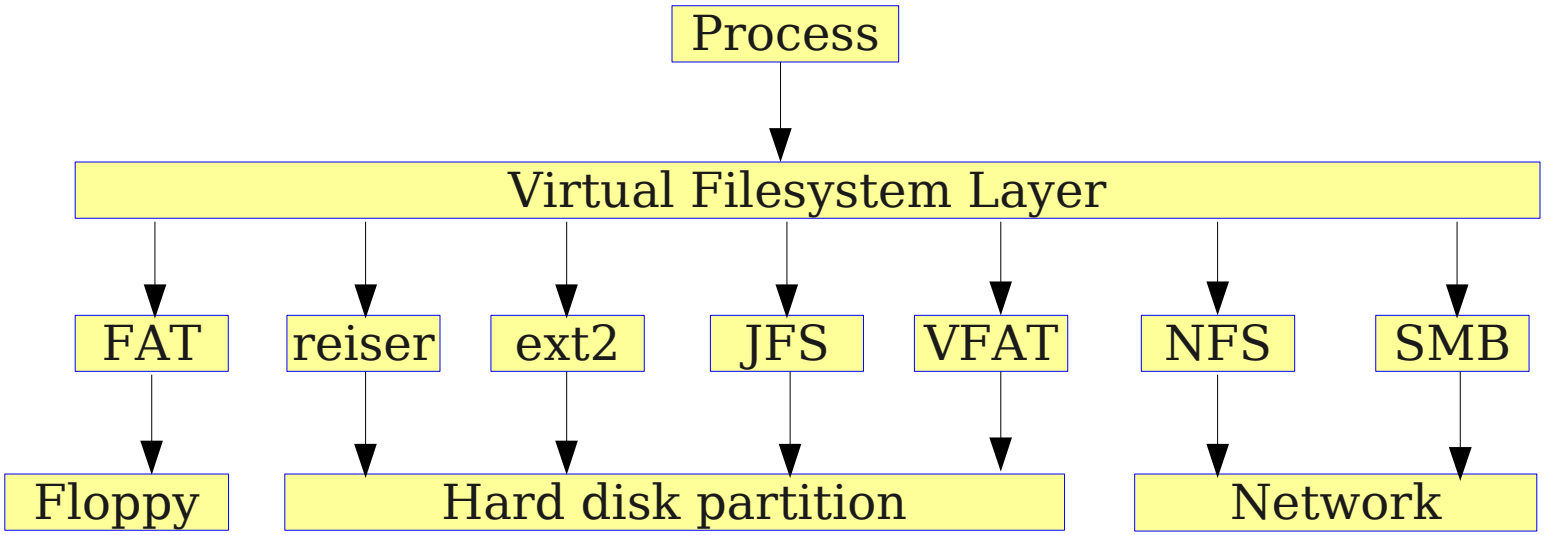
- The `lost+found` directory is a feature of the ext2 filesystem type
  - A directory of this name exists at the top level of every ext2 partition
- Used by the program `fsck` (file system check)
  - Typically run at boot time after an unclean shutdown (crash)
  - Files which appear to be intact but have no link in any directory are given entries in lost+found
  - Entry is named after the inode number of the file
  - System administrator can try to deduce the original name
  - This hardly ever happens!

The Reiser file system format

- Reiser file system developed by Hans Reiser offers several benefits:
  - Can store multiple file fragments in a single block (less space wasted)
  - Journaling
  - Logical block size of 4096 bytes
  - Fast access; data blocks organised using binary trees
  - File attributes are stored within the binary tree, there is no fixed-size inode table
  - Maximum partition size and file size is 16 Tb
  - File names up to 255 characters
- A disadvantage of Reiser is that it imposes a greater overhead on disk space
  - Not recommended on very small partitions
- Resier is the default file system type for a SuSE linux installation

The virtual file system

- The linux kernel provides a virtual filesystem layer which hides the differences between the physical filesystem layouts
  - Supports standard UNIX open/close/read/write operations
  - Provides illusion of UNIX filesystem semantics (e.g. rwx-style access permissions on FAT filesystems)

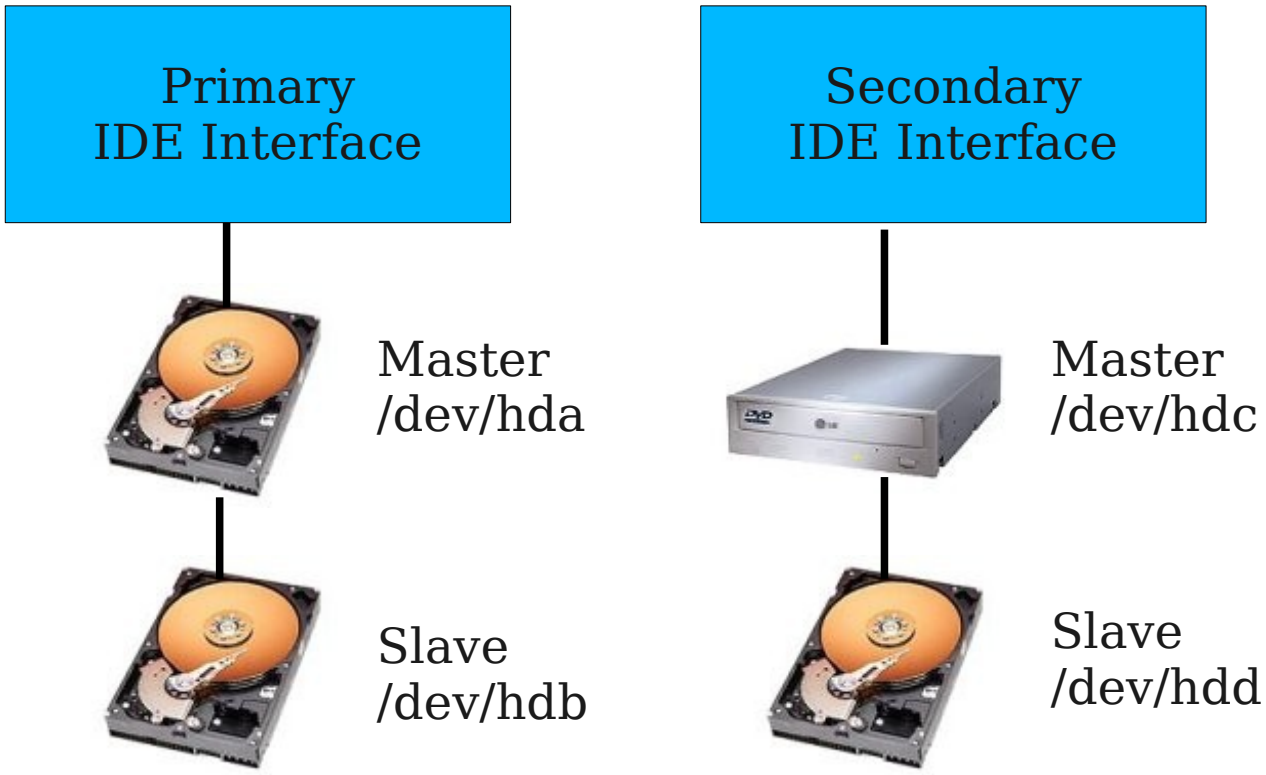


Partition management

- Partition management
  - Naming of IDE and SCSI drives
  - Naming of partitions
  - Swap partitions
  - Partitioning guidelines
  - Partitioning examples
  - Creating partitions with fdisk
  - Creating and mounting a file system
  - Creating partitions with YaST

Naming of IDE drives

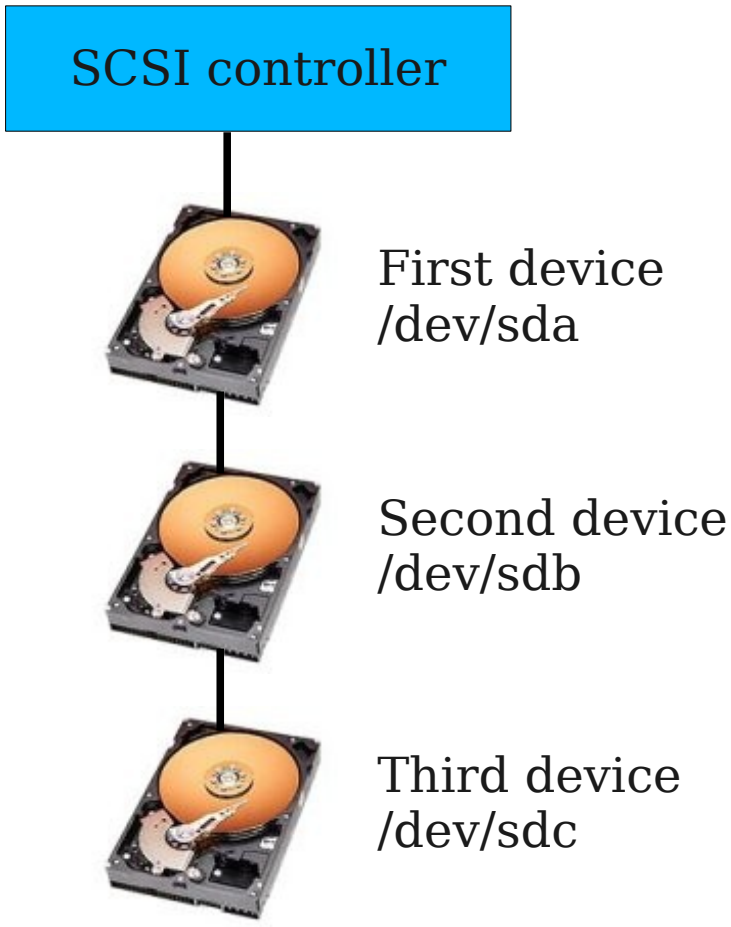
- Disk drives and partitions have names in the /dev directory
- Modern PCs are able to connect up to 4 IDE drives:



On a typical PC with one hard drive and one CD or DVD drive, the CD/DVD may be connected as hdb or hdc

Naming of SCSI drives

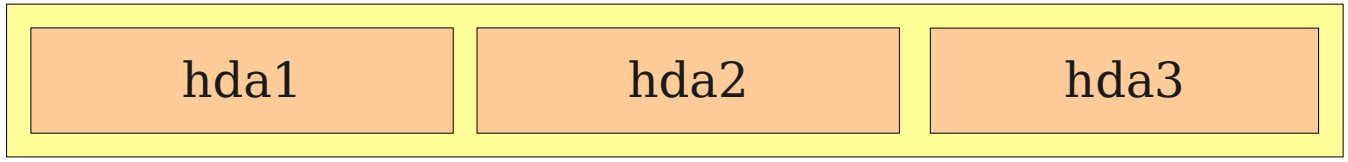
- SCSI controllers can handle multiple drives
  - They are simply named in order



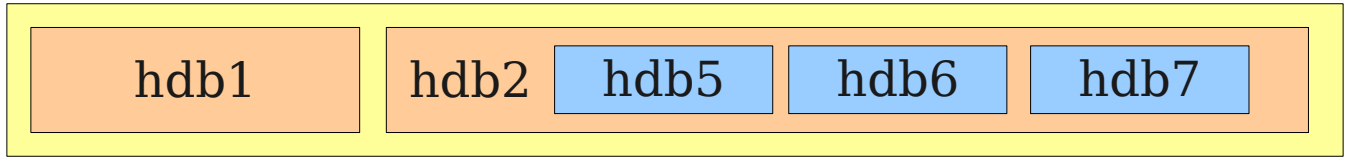
Note: Linux sometimes emulates other devices as SCSI devices; e.g. USB memory sticks or digital cameras

Naming of partitions

- Originally, PCs allowed a maximum of four partitions on a hard drive
  - To allow more, one partition can be designated an *extended* partition
  - Multiple *logical partitions* can be placed within the extended partition
- Linux numbers the primary partitions 1, 2, 3 and 4, and the logical partitions are numbered starting at 5 (even if there are less than 4 primary partitions)
- These examples are for a machine with IDE drives:



FIrst drives has three primary partitions



Second drive has one primary, one extended, and three logical partitions

Swap partitions

- It is normal to allocate one or more partitions as *swap partitions*
  - Swap partitions do not contain a filesystem
  - They are used to increase the available virtual memory space on the machine beyond the amount of RAM (random access memory)
- How big should the swap partition be?
  - Hard to give generic advice
  - A common rule of thumb is to make swap the same size as RAM
  - Performance degrades significantly if the system does a lot of swapping
  - Since memory is relatively cheap, a better guideline might be to put enough RAM in the machine so that it never swaps at all
  - We will see how to monitor memory and swap usage in chapter 10

Partitioning guidelines

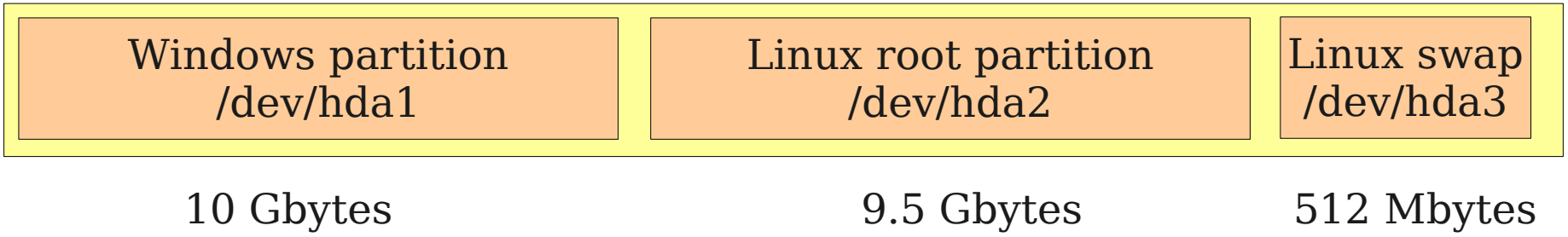
- The simplest partitioning scheme is to put the entire file system into the root partition
  - Easy, no need to make any up-front decisions about partition sizes
- However, there are good reasons for using more partitions
  - Keeping the root partition small minimises the amount of file system that must be intact and available for the system to boot successfully
  - Partitions provide a crude way to impose disk space quotas on pieces of the file system
  - Partitions are the “unit of administration” of the file system (e.g. Repair, backup and restore of the file system is on a per-partition basis)
  - Partitions for “static” pieces of the filesystem (e.g. `/usr`) can be mounted read-only, improving security
  - If your file system spans multiple hard drives, you necessarily have multiple partitions

Partitioning guidelines (continued)

- Directories essential for booting must be on the root partition
  - `/bin, /sbin, /etc, /lib`, …
- Other directories are candidates for being on separate partitions
  - `/usr` May be mounted readonly. Suggested minimum size 2 Gbytes
  - `/var`
  - `/boot` On early PCs this partition needs to be within the first 1024 cylinders. Suggested minimum size 50 Mbyte.
  - `/home` On a server, this partition may be exported to client machines This is likely to be the largest partition
  - `/tmp` Hard to give a specific guideline on the size but 1 Gbyte is probably more than enough
  - `/opt`

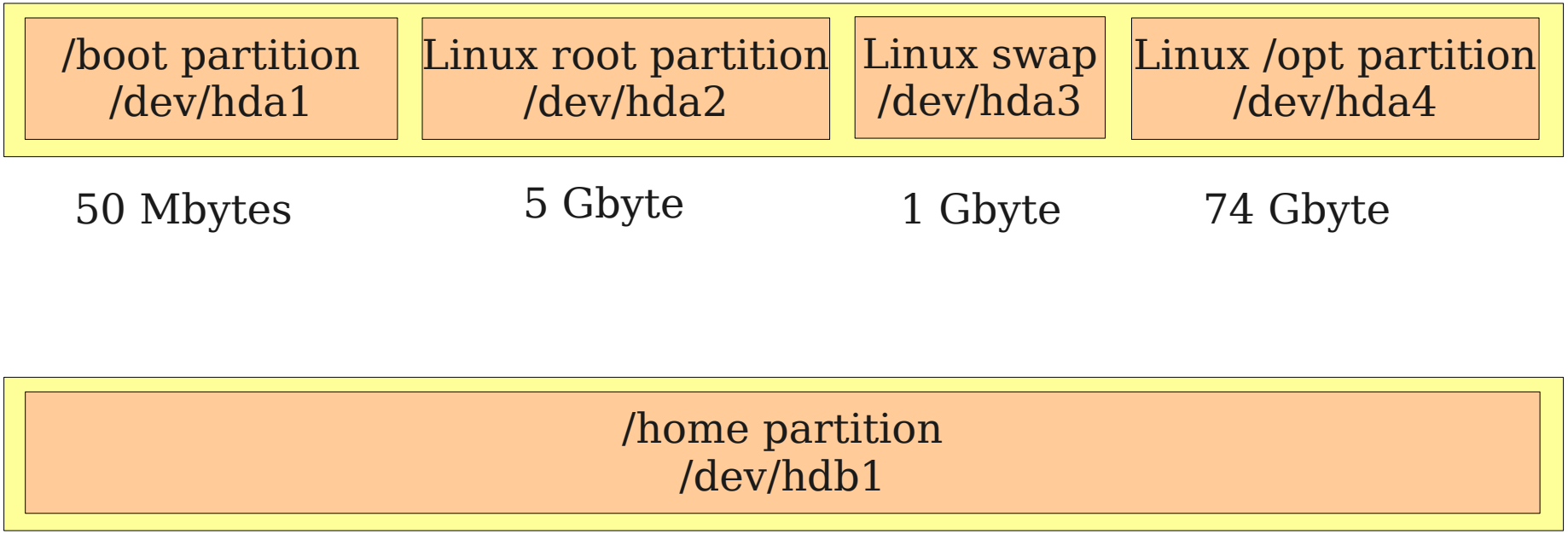
Partitioning example: dual-boot desktop machine

- Small desktop machine with 20 Gbyte hard drive



Partitioning example: server with two hard drives

- Server machine with two 80 Gbyte hard drives



Creating partitions with `fdisk`

- The command-line tool `fdisk` allows manipulation of the partition table on a hard drive
  - Has a rather clunky user interface
  - The following slides show a typical dialog
- Can be used non-interactively to list the partition table:

```
# fdisk -l /dev/hda
Disk /dev/hda: 255 heads, 63 sectors, 3648 cylinders
Units = cylinders of 16065 * 512 bytes

 Device    Boot    Start       End    Blocks   Id  System
/dev/hda1   *          1       784   6297448+  83  Linux
/dev/hda2            785       850    530145   82  Linux swap
/dev/hda4            851      3648  22474935    5  Extended
/dev/hda5            851       914    514048+  83  Linux
```

- The tool `cfdisk` provides a slightly better interface

Creating partitions with `fdisk`

```
# fdisk /dev/hda

Command (m for help): p

Disk /dev/hda: 255 heads, 63 sectors, 3648 cylinders
Units = cylinders of 16065 * 512 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1   *           1         784     6297448+  83  Linux
/dev/hda2             785         850      530145   82  Linux swap

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
e
Partition number (1-4): 4
First cylinder (851-3648, default 851):
Using default value 851
Last cylinder or +size or +sizeM or +sizeK (851-3648, default 3648):
Using default value 3648
```

Display the existing partition table

Create an extended partition (hda4) spanning the whole of the rest of the disk

ict@innovation

Creating partitions with `fdisk` (continued)

```
Command (m for help): n
Command action
   l   logical (5 or over)
   p   primary partition (1-4)
l
First cylinder (851-3648, default 851):
Using default value 851
Last cylinder or +size or +sizeM or +sizeK (851-3648, default 3648): +500M

Command (m for help): p

Disk /dev/hda: 255 heads, 63 sectors, 3648 cylinders
Units = cylinders of 16065 * 512 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1   *           1         784     6297448+  83  Linux
/dev/hda2             785         850      530145   82  Linux swap
/dev/hda4             851        3648    22474935    5  Extended
/dev/hda5             851         914      514048+  83  Linux
```

Create a logical partition (hda5) of 500 Mbytes

Redisplay the partition table to verify

ict@innovation

FOSSFA

Creating partitions with fdisk (continued)

```
Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: Re-reading the partition table failed with error 16:
         Device or resource busy.
The kernel still uses the old table.
The new table will be used at the next reboot.
Syncing disks.
#
```

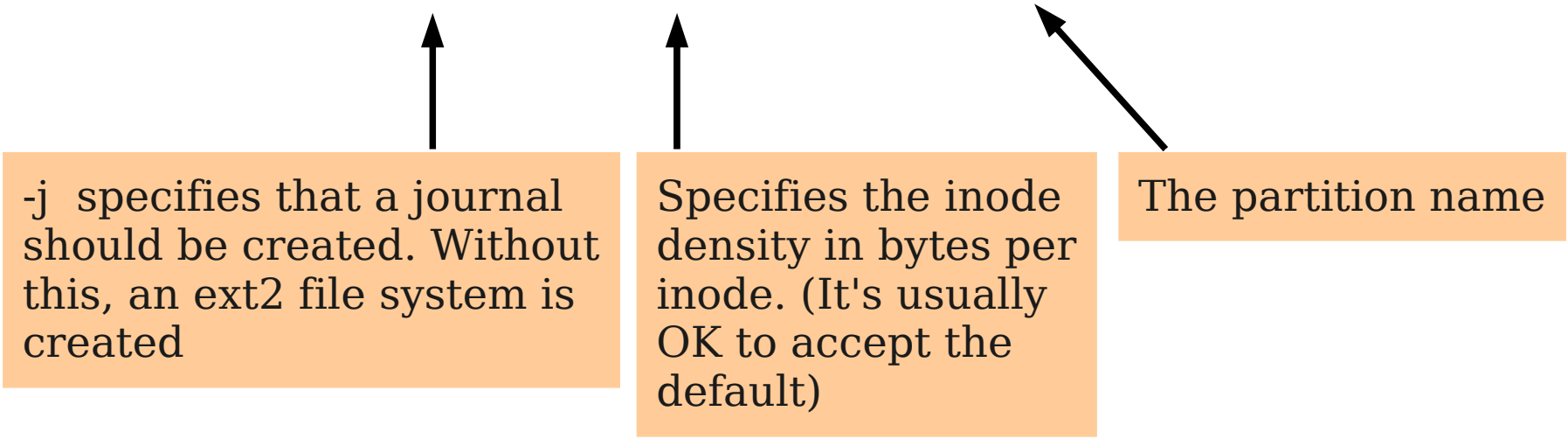
Finally, the new partition table is written to the disk and `fdisk` exits

It is apparently necessary to reboot to force the kernel to use the new table

Creating a file system

- After creating a partition you need to build a filesystem on it
- For example, to build an ext3 filesystem on `/dev/hda5`:

```
# mke2fs  -j -i 8192 /dev/hda5
```



- To create a reiser file system on `/dev/hda6`:

```
# mkreiserfs  /dev/hda6
```

Mounting a file system

- Our new partition must be mounted before it can be accessed
- If necessary, create a mount point first:

```
# mkdir /new1
```

- Now attach the partition to the mount point:

```
# mount /dev/hda5 /new1
```

- Finally, we can copy some files onto the new partition, for example:

```
# cp -r /usr/share/man /new1
```

- To unmount the partition:

```
# umount /dev/hda5
```

  - Or ...

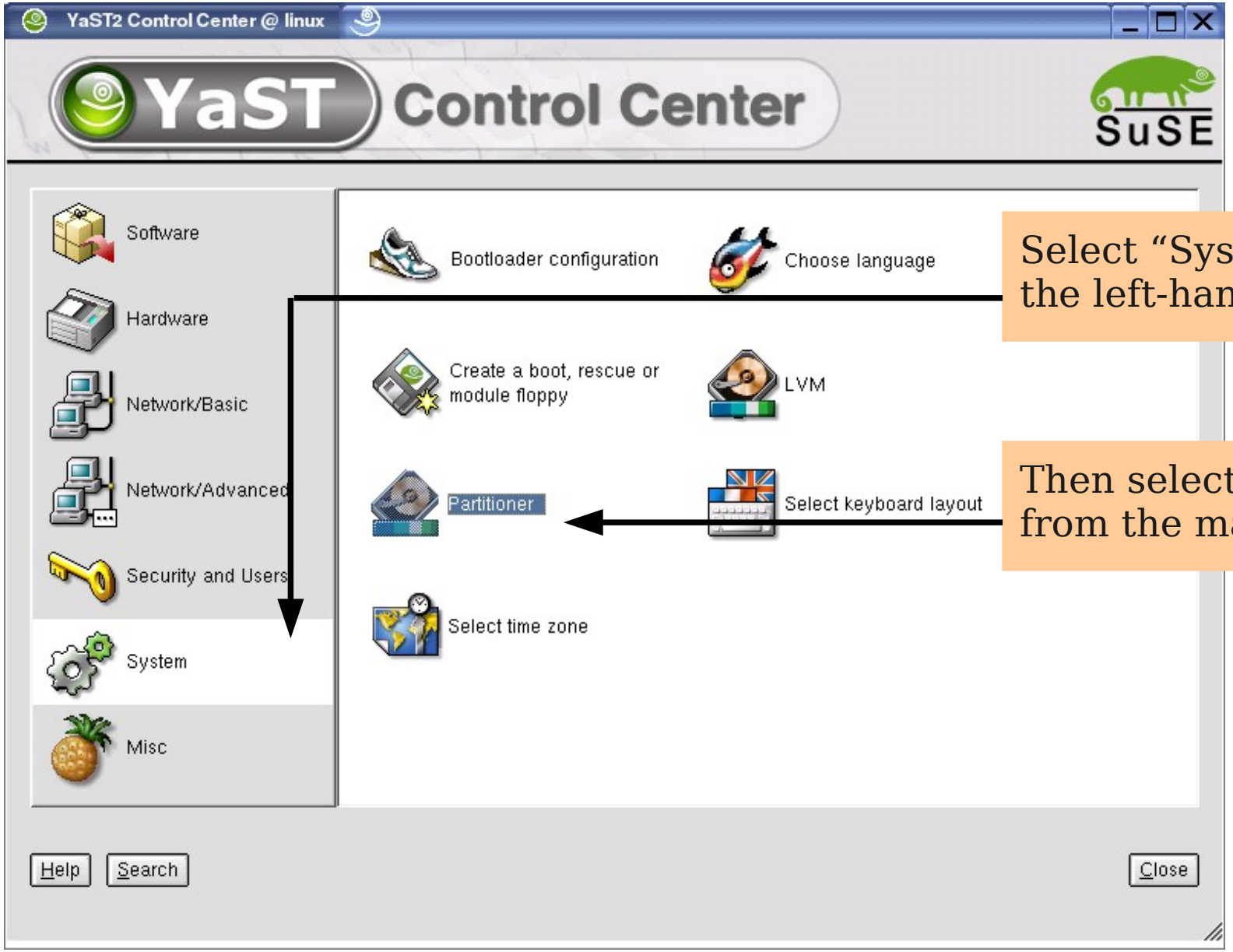
```
# umount /new
```

Note the command is `umount` not `unmount`

Managing partitions with YaST

- YaST can also be used to create partitions. The tool will automatically:
  - Create the partition and update the partition table
  - Format the partition (i.e. Create an empty file system)
  - Create the directory for the mount point
  - Mount the partition
  - Create an entry in `/etc/fstab` (to be discussed later) so that the partition can be mounted automatically at boot time
- YaST is invoked via `Main Menu --> System --> Yast`
  - You will be prompted for the root password

YaST main screen



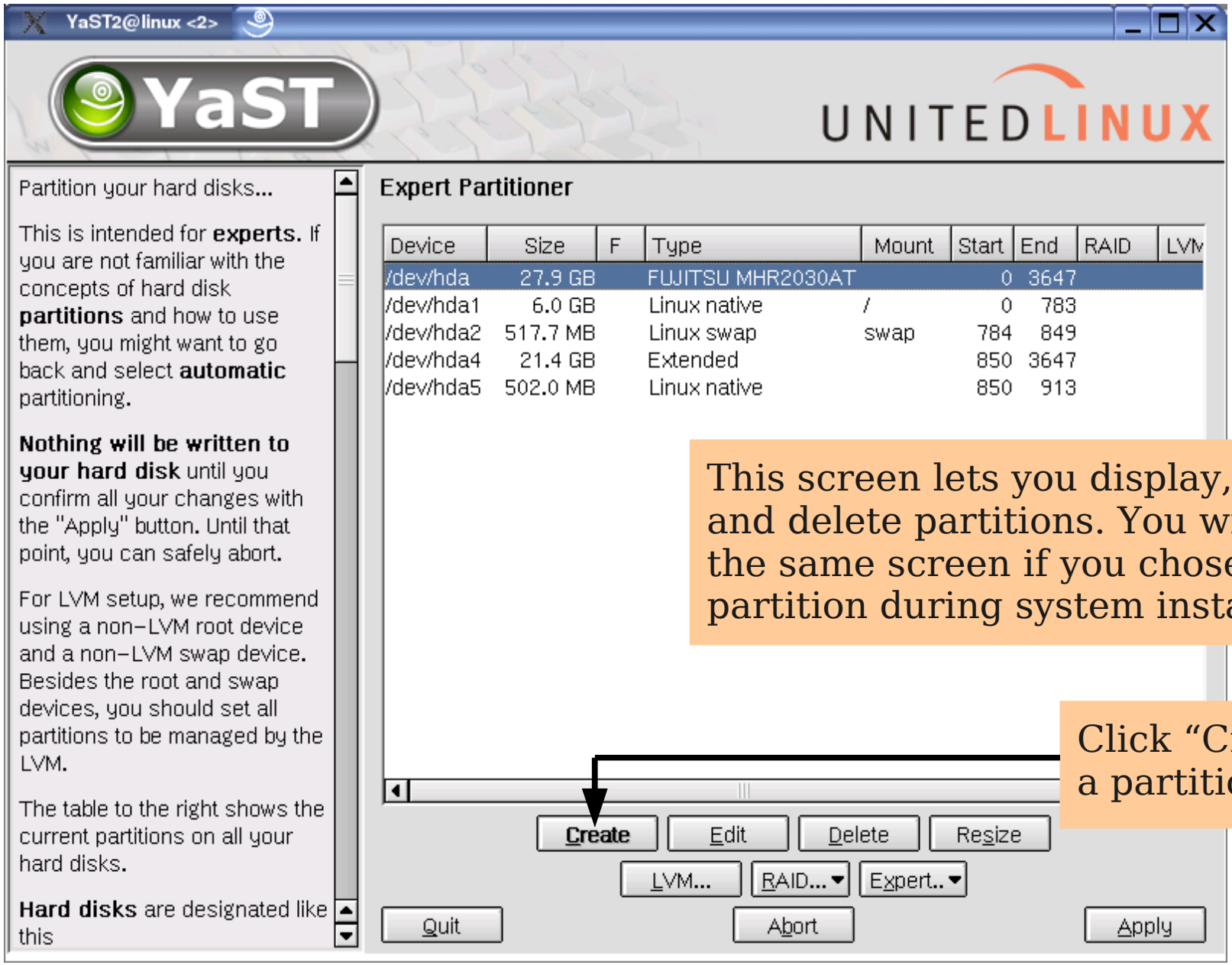
Select “System” from the left-hand pane

Then select “Partitioner” from the main window

ict@innovation

FOSSFA

YaST “Expert Partitioner” screen

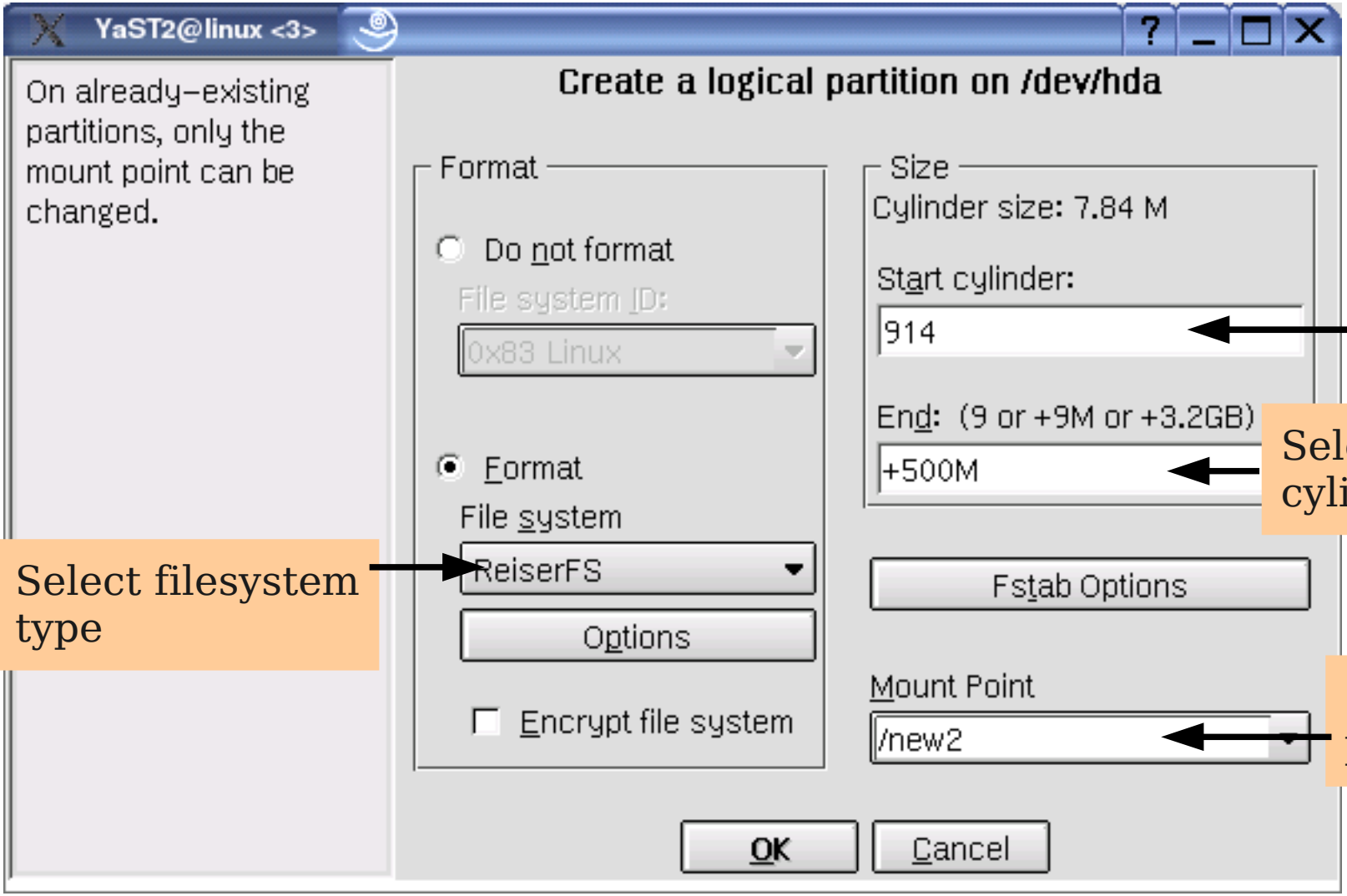


This screen lets you display, create, edit and delete partitions. You will also see the same screen if you chose to manually partition during system installation

Click “Create” to add a partition

ict@innovation

YaST “Create Partition” dialog



Select start cylinder

Select last cylinder or size

Select filesystem type

Select mount point

ict@innovation

Exercise: Creating new partitions

In this exercise we will create two new disk partitions, each of 500 Mbytes, on the free space on the hard drive. We will format one of these partitions as an ext2 file system and the other as a Reiser file system. Create the first partition using fdisk:

1. Log in as `root`

2. Enter the command:

```
# fdisk /dev/hda
```

3. At the `fdisk` command prompt, enter the command 'p' to show the current partition table.

How many partitions are currently defined? _____

What are the device names of these partitions? _____

What is the highest cylinder number on the hard drive? _____

What is the highest cylinder number currently in use in a partition? _____

continued ...

Exercise (continued)

4. Following the example in the notes, use `fdisk` to create an extended partition (hda4) spanning the whole of the remaining free space on the disk

5. Following the example in the notes, create a logical partition (it will automatically be numbered hda5) starting at the first available cylinder and of size 500 Mbytes

6. Print the partition table to verify the result. If in doubt, ask your instructor!

   Write down the device name of your new partition: _____

7. Write out the partition table and exit from `fdisk`

8. Back at the shell prompt, enter the command `'reboot'` to reboot the machine.

9. After the machine has rebooted, log back in as root.

10. Following the example in the notes, create an ext2 filesystem with a journal on the new partition.

    (Be *very careful* not to re-format any of the other partitions. If you are in any doubt what the device name of the new partition is, ask your instructor.)

Exercise (continued)

Write down the command you used: _____

11. Create a mount point called "/new1" for the new partition:

    Write down the command you used: _____

12. Mount the new filesystem onto the new mount point.

    Write down the command you used: _____

13. Run the command

```
# ls -a /new1
```

    What directory exists on an empty ext2 file system? _____

Exercise (continued)

The second partition will be created using YaST:

14. Using the screenshots in the notes as a guide, use YaST to create an additional partition as follows:

    Device name: hda6
    File system type: ResierFS
    Size: 500 Mbytes
    Mount Point: /new2

    (YaST will automatically create the mount point and mount the partition for you)

15. Run the command `df` which shows used and free disk space on each partition

    How much space does an empty file system occupy for ext2? _____

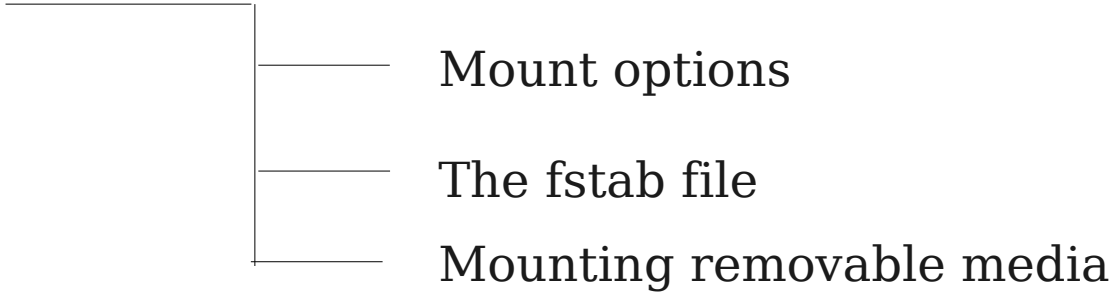
    How much space does an empty file system occupy for reiser? _____

16. Try copying some files onto the new partitions to verify they are accessible

End of exercise

Mount options and the `fstab` file

- Mount options and the fstab file



Mount options

- Mount has many options. Here are a few:

Option	Meaning
`-r`	Mount the partition read-only
`-t type`	Mount a file system of the specified type (for example, ext2, ext3, iso9660, msdos, nfs, reiserfs, smbfs) Normally not necessary as mount will figure out the file system type automatically
`-a`	Mount all the file systems listed in /etc/fstab
`-o noexec`	Do not allow files on this file system to be executed
`-o nodev`	Do not allow device files to be recognised on this filesystem
`-o nosuid`	Do not allow programs on this file system to run “set user ID”
`-o ro`	Mount the partition read-only (same as -r)
`-o remount`	Remount the partition (e.g. To change from read-only to read-write)

- Options following -o can be combined in a comma-separated list, e.g.

```
# mount -t ext2  -o ro,nodev,noexec  /dev/sda5  /data1
```

The /etc/fstab file

- The /etc/fstab file helps automate the mounting of file systems
- Entries in the file serve two purposes
  - They specify file systems to be mounted automatically at boot time
  - They associate a set of mount options and a mount point with a file system allowing it to be mounted using only a single argument to mount

```
/dev/hda1       /                 reiserfs  defaults              1 1
/dev/hda2       swap              swap      pri=42                0 0
devpts          /dev/pts          devpts    mode=0620,gid=5       0 0
proc            /proc             proc      defaults              0 0
usbdevfs        /proc/bus/usb     usbdevfs  noauto                0 0
/dev/cdrecorder /media/cdrecorder auto      ro,noauto,user,exec   0 0
/dev/fd0        /media/floppy     auto      noauto,user,sync      0 0
/dev/hda5       /new1             ext2      defaults              1 2
/dev/hda6       /new2             reiserfs  defaults              1 2
```

Partition name

Mount point

Filesystem type

Mount options

Dump and fsck parameters

Mount options in the `fstab` file

- The fourth field in `fstab` supplies options to use with the mount command.
- The `noauto` option
  - Specifies that the file system is not to be mounted at boot time. The purpose of such entries is to simplify the mount command. For example with `fstab` as shown, either of the commands

```
$ mount /dev/fd0
```

```
$ mount /media/floppy
```

  - Would be taken to mean:

```
$ mount -o noauto,user,sync /dev/fd0 /media/floppy
```

Mounting removable media

- By default, only root can mount and unmount file systems
  - There is a common requirement to allow non-root users to mount and unmount removable media such as floppies, CDs and DVDs
  - The `user` option allows ordinary users to mount and unmount this filesystem. (Once mounted, only the user who mounted it can unmount it)
- Removable media allow the introduction of 'uncontrolled' files and may carry security risks; e.g.
  - A shell owned by root with the setuid bit on
  - A device file entry for `/dev/hda1` with mode `rw-rw-rw`
  - Any malicious executable
- There are mount options to eliminate these risks:
  - `nosuid, nodev, noexec`
  - The `user` option implies `nosuid`, `nodev` and `noexec`

Exercise: Modifying `/etc/fstab`

1. Modify your `fstab` so that the two partitions you previously created are automatically mounted onto `/new1` and `/new2` when the system is booted.
2. Reboot the system to verify.

Quiz

- What is the device name for the slave IDE drive attached to the primary IDE controller?
- What is the device name for the first logical partition on the master IDE drive attached to the secondary IDE controller?
- In a long directory listing (i.e. The output of `ls -l`) what does it mean if the first character on the line is a 'b'?
  - In which directory might you find such an entry?
- Give one advantage and one disadvantage of the Reiser file system compared to ext3
- Give two reasons why you might prefer not to put the entire file system of a linux system into the root partition
- Assuming this line in `/etc/fstab`:

  ```
  /dev/fd0  /media/floppy  auto  noauto,user  0 0
  ```

  What do the options `noauto,user` mean?
-

True or False?

- Linux can read the NTFS file system format as used by Microsoft Windows
- Microsoft windows can read the ext2 file system as used by Linux
- On an ext2 file system the inode table is expanded on demand
- Only root can mount removable media into the file system
- YaST can be used to create partitions
- All the file systems listed in `/etc/fstab` are automatically mounted at boot time