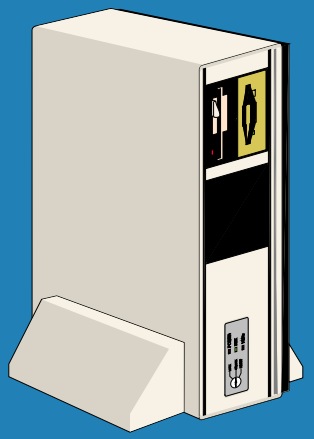
GNU/Linux & Unix



Lecture #1

Introduction

Lecture #2

Using the Bourne shell

Unix

- Developed as a multi-user OS for minicomputers
- Made available to US Universities with source code.

  Used for early Internet development

  More “open” than proprietary OSs (e.g. Windows)

  Some legal issues with ownership of Unix
- MacOS is a version of Unix

- Note: Unlike DOS or Windows, Unix is case-sensitive,

GNU/Linux

1987 Prof Andrew S. Tanenbaum, (http://cs.vu.nl/~ast) developed Minix, a limited Unix look-alike OS.

Source code was published in his OS book.

1983 Richard Stallman initiated the GNU project to promote open source development.

This project created many of the tools required to build a new OS.
(i.e. gcc GNU C compilier)
http://www.gnu.org/

Aug 25th 1991 Finnish student Linus Torvalds annouces his own free operating system on comp.os.minix.

Linux placed under General Public License (GPL)

1994 First stable release of Linux.

File System

All types of data are held in files.

Files are grouped in directories.

("folders" in Windows terminology)

A directory may contain sub-directories.

The parent directory of the system is called "root", labelled / .

Every user has a "home" directory which is normally identified by their username;

on login, the system places the user in their home directory (e.g. /home/bsc666).

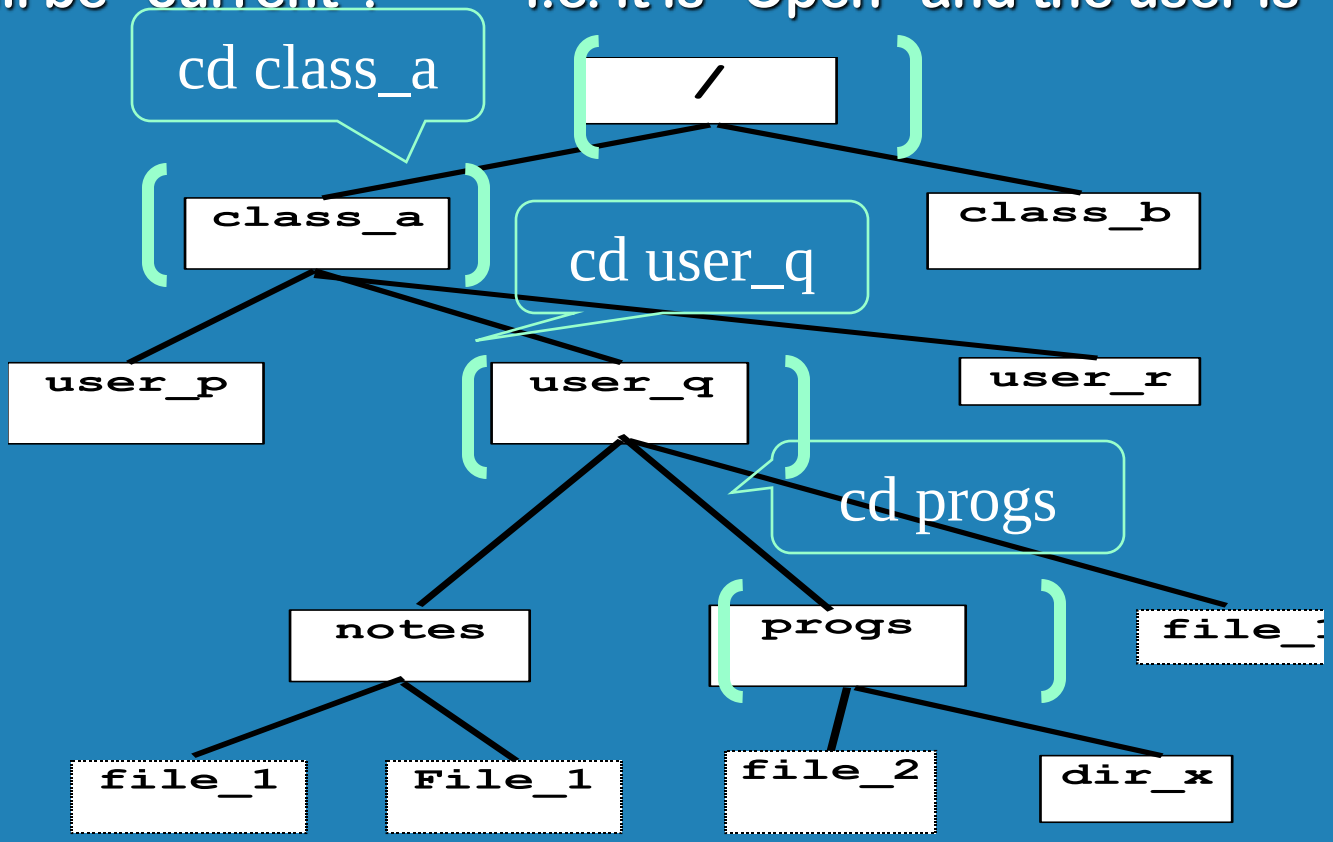
Much of the file system is standard across systems.

/bin /etc /usr

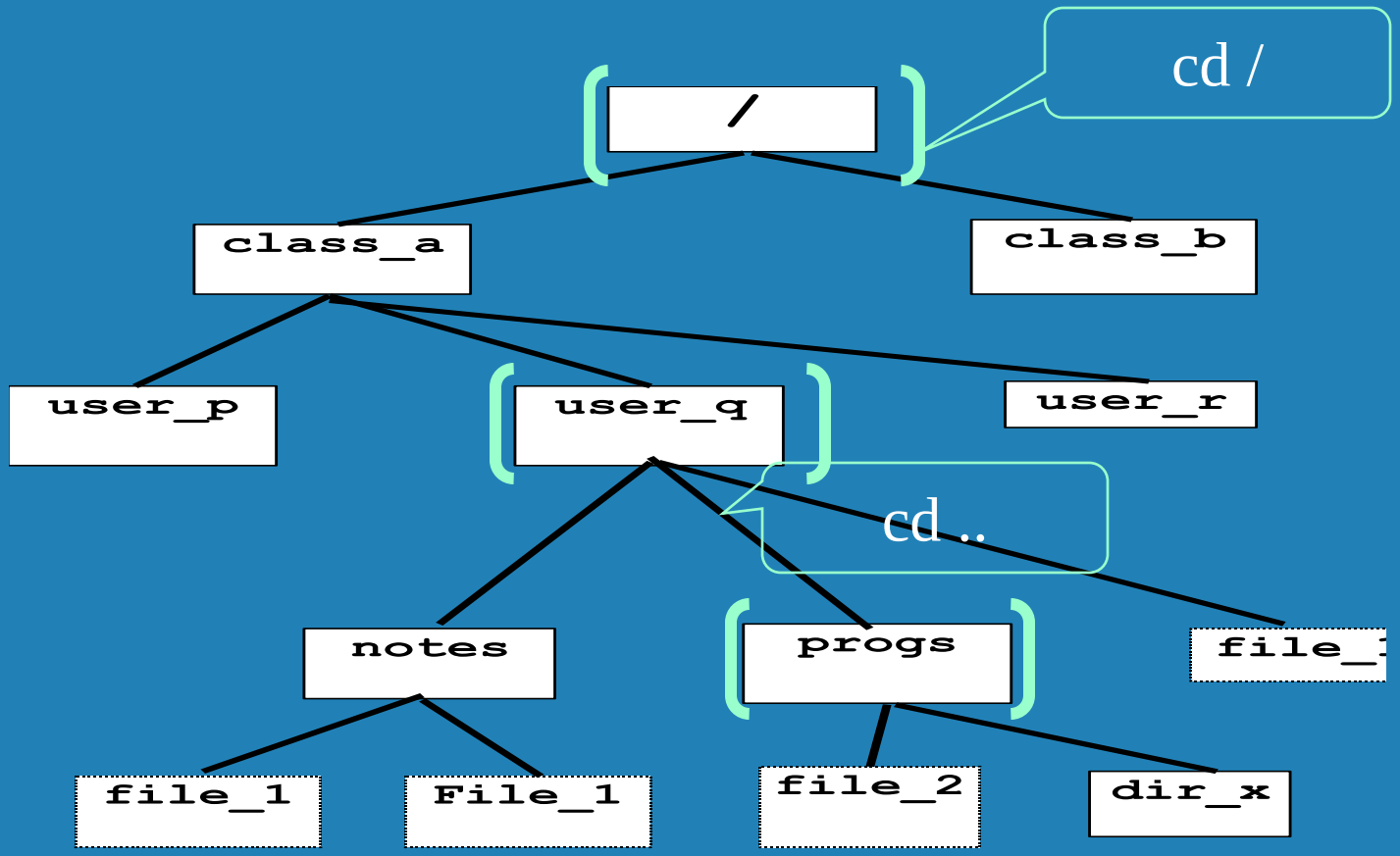
Users may create sub-directories and files in any directory they "own".

File System Navigation

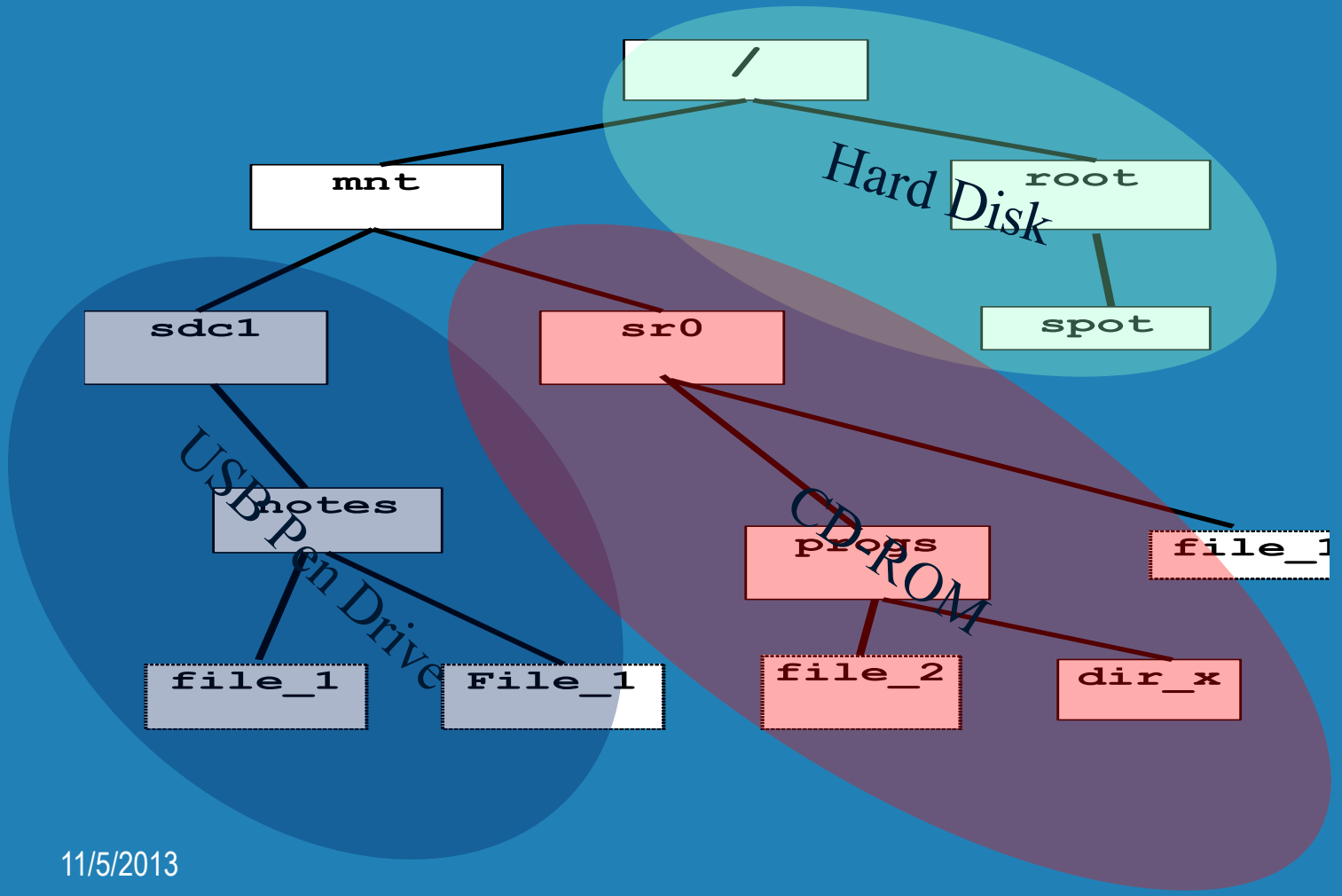
At any point in time, one of the directories will be "current". i.e. it is "Open" and the user is "in it".



File System Navigation



File System Media



Pathnames

- **The full location identifier of a directory or file is called its "pathname".**
- **There are two kinds:**

  relative to the "current" directory,

  If the "current" directory is **/class_a/user_q** a user can address **file_2** as **progs/file_2**

  absolute (measured from the root "/")

  /class_a/user_q/progs/file_2

- **files in the current directory can be addressed simply by name (i.e. no pathname needed)**

  when in user_q, file_1 refers to the file on the same level as notes and progs.

Special Directory Names

~ (tilde) means the user's home directory.
(when used at the start of a pathname).

. means the current directory.

.. means the parent directory of the current one.

/ means the root directory
(when used at the start of a pathname).

File Access Privileges - 1

- All users belong to one of three categories with respect to any file:

  user - the owner

  group - members of same group.

  other - all other users (aka “world”).

  The group a user belongs to is set by the system administrator.

  Permissions may be given for each category to have certain privileges on a file or directory:-

  r read

  w write

  x execute

File Access Privileges - 2

- The `ls -l` command shows the current privileges on a file:

  `rwx r-- ---` shows that the owner has full rights and members of the owner's group may read the file.

  to browse the content of a directory the execute permission must be set:

  `rwx --x --x` allows anyone to pass through a directory.

  Privileges can be shown as octal numbers –

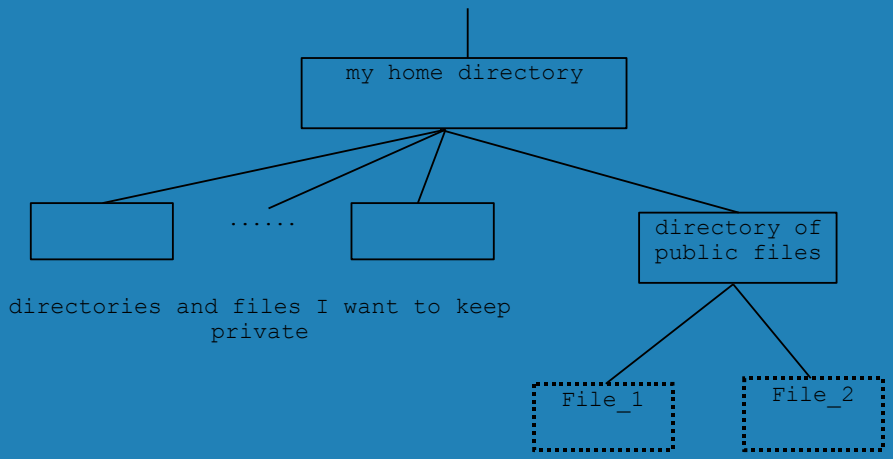
  `7 4 0` is

  `(111 100 000)`

  equivalent to

  `rwx r-- ---`

File Access Privileges - 3



Home directory - 751

would allow group to read and execute directory.

directories and files to be kept private - 700

only the owner can access these files and directories

directory of public files - 711

would allow group and world to execute the directory

File_1 and File_2 - 744

would allow group and world to read the contents of these files

Use

chmod nnn *pathname*

to set privileges.

Command Shell

- **The "shell" is the command interpreter, through which we operate the system.**
  - **Several shells exist, the most simple is the Bourne shell (/bin/sh).**
    - Bourne Again Shell (/bin/bash) is open source alternative
  - **The "tc" shell, or tcsh is common on Sun Solaris machines and "bash" on Linux.**
    - The Korn shell (ksh) is proprietry to Sun.
  - **Most commands behave the same way regardless of the shell used to invoke them.**
  - **However different shells have different characteristics as regards script syntax.**

Common File Commands

- The following commands are are used to explore the file system:

pwd - prints the name of the current working directory

ls - lists the names of all files in the current directory.

cd *dirname* - changes the current directory to that named.

cat *filename* - displays the contents of the file named.

cp *file1 file2* - copies the contents of the file1 to file2.

rm *pathname* - deletes the file named.

mkdir *pathname* - creates a subdirectory

rmdir *pathname* - deletes the empty directory named.

grep - the Pattern Matcher

- displays all lines in a file which contain a specified pattern or string.

```
grep string filename
```

If the search string contains spaces it must be in double quotes:

```
grep "any text you want" filename
```

To omit lines containing a string use -v:

```
grep -v "text you do not want" filename
```

To list lines beginning with a character:

```
grep ^The filename
```

lists lines beginning with "T"

Regular Expressions

. (dot)	matches any character
[abc]	matches any single character a ,b or c
[A-Z]	matches any single uppercase character
[0-9]	matches any single digit
Mon|Tue	matches strings “Mon” OR “Tue”
*	matches the literal asterisk character itself, the backslash is used as the escape character.
^	matches the start point of a line
$	matches the end point of a line

*	matches any number (including none) of the previous character
+	matches one or more of the previous character
?	matches none or one of the previous character

Wild Cards

differ from regular expressions

- These can specify multiple pathnames in commands.

  ? - represents any single character

  ***** - represents zero or more characters

  `ls -l tunix?` will list tunix1, tunix2, ..., tunixX

  `ls -l p*` will list prog1, p-X and program_Z.

- Use wildcards with care in rm commands

  `rm *.*`

  will delete all files in a directory.

Redirecting input & output

- **Redirection -**

  a process in takes its input from an input channel and sends output to an output channel.

- **Normally, the channels are:**

  Standard Input - the keyboard

  Standard Output - the screen, or window, from which the commands were issued.

Redirecting input & output

- **Output Redirection**
  - `> filename` send output to "filename"
  - `>> filename` append output to "filename"

  e.g. `cat file_1 >> file_2`
- **Input Redirection**
  - `< filename` take input from "filename"
- **Piping**
  - `|` Output used as input to next command.

```
$ ps -ef | grep root
$ ls | more
```

pipes the `ls` output through `more` causing the data to be displayed a screenful at a time.

Combining Commands

- **The result of one command can be `embedded` in another.**

  E.g. pwd prints the working directory

```
$ grep `pwd` archive.list
```

- **same as**

```
$ grep /home/root/mydir archive.list
```

Asynchronous and Sequential Statements (; &)

- Multiple commands can be submitted on the shell command line.
- These can be run one after the other (as a Sequential list).

```
$ cd scripts; mv testscr backup.txt; cat backup.txt
```

- Or independently (an Asynchronous list)

```
$ date >> log.txt & ps >> status.log
```

Note placing a "&" after a single command effectively runs it in the background

Some Other Useful Commands

- `cut`

  This command can parse fields (columns) from text files. Useful for "piping".

  ```
  $ cut -f2 -d":"
  ```

  Cut the second field from each line of text, delimit fields on the colon: character. Note that the default delimiter is the TAB character.

- `sort`

  This command can sort a text file line by line. By default, sorts on the first character in a line but can also sort on specified columns/fields. Useful for "piping".

  ```
  $ sort jumbledfile.txt > sortedfile.txt
  ```

- `echo`

  As in DOS this merely echoes strings of values to the standard output (screen). Useful for script messages, debugging etc.

  ```
  $ echo Hello there
  Hello there
  ```

Some Other Useful Commands

- `dirname`

  This command will parse a pathname by returning only the directory part.

  ```
  $ dirname  /root/mydir/subdir/foobar
     /root/mydir/subdir/
  ```

- `basename`

  This command will parse a pathname by removing the directory part.

  ```
  $ basename  /root/mydir/subdir/foobar
     foobar
  ```

- `readlink -f`

  This will provide an absolute pathname, given any valid (relative or absolute) pathname.

  ```
  $ readlink -f ~
    /root

  $ readlink -f ../dave/spot/myfile.txt
    /root/spot/myfile.txt
  ```

Checking File Content

- **md5sum**

  Reads a file contents and computes a 128bit "message digest" or "hash".

```
$ md5sum PuppyWary.5.3.7z
a6dc1f13574c40eb4286d4752daa48e4 PuppyWary.5.3.7z
```

  The chances of two different files having the same md5sum digest is practically nil.

  The 128bit digest can be considered a unique signature for the file.

  E.g. http://distrowatch.com/?newsid=07736

Checking File Content

- **diff**

  Reads and compares two text files

```
$ diff ip.addresses ip.addresses.orig
< 146.176.14.42
---
> 146.176.14.48
```

  In this case two files are identical except for one line.

  The ip.addresses file has 146.176.14.42 where ip.addresses.orig has 146.176.14.48

find – powerful search and execute

- find will search a whole branch of the file system for any file (or directory) with the desired attributes.

  Attributes:

  - Date of creation, last access or modification
  - Size of file, +bigger than -smaller than
  - Specific file permissions (e.g. rwxrwxrwx)
  - Name
  - The owner (user) or group a file belongs to

find -exec

- As well as searching the file system, find can also execute a command on every file the search turns up.

- Example

- `find /usr -name *.htm -perm 700 -exec chmod 744 {} \;`