# Lecture 2b

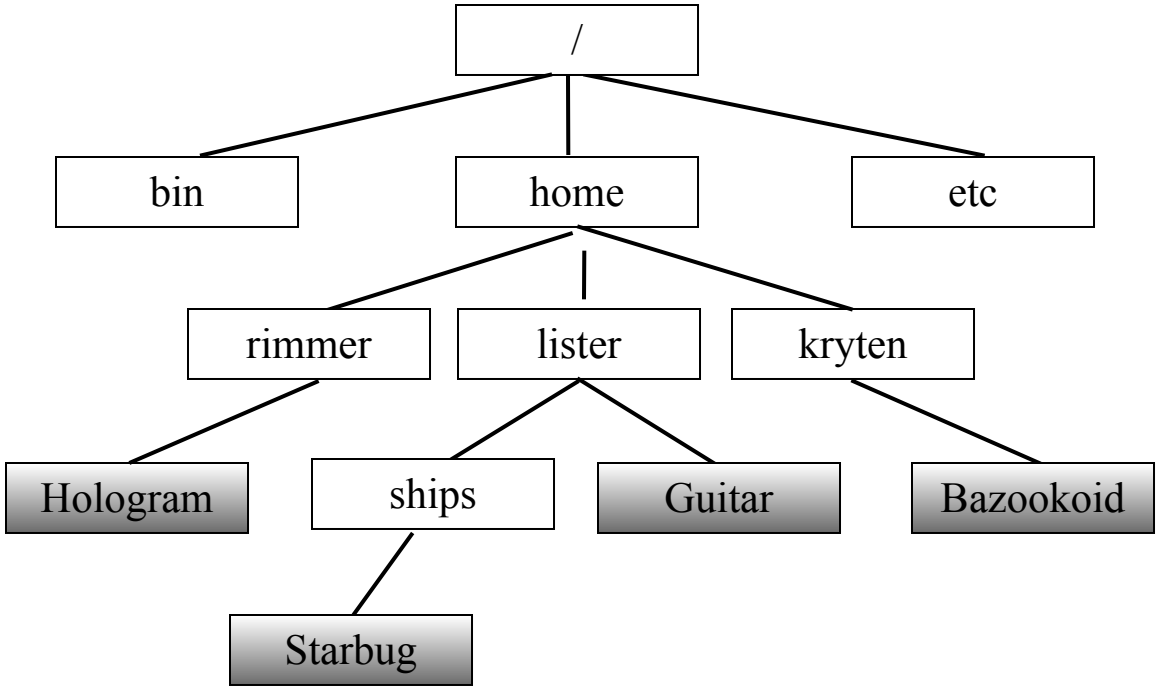
Pathnames, files, special characters in filenames, and file permissions.

COP 3353 Introduction to UNIX, FALL 2013

# Files

- Files
  - A well defined repository of information
    - Program or component of a program
    - Arbitrary text
    - An executable (binary text)
  - Special files called directories contain or point to other files
- Structure of Directories
  - Hierarchically structured like an inverted tree
  - / is the starting directory or “root”
- Pathnames
  - Locating a file by following a sequence of directories until you reach the file
  - / is also the separator between the directories and final file

# Example set of directories and files



# More on pathnames

- Absolute pathnames start at root
  - /home/lister/ships/Starbug
  - /bin
- Relative pathnames start at current directory
  - Suppose current directory is “home”, then:
  - rimmer/Hologram (relative) would refer to the same file as:
  - /home/rimmer/Hologram (absolute)
- Special symbols for current directory and parent
  - “..” refers to parent directory (the directory “above”)
  - “.” is current directory
- Referencing user directories
  - `~rimmer` is the absolute pathname to the user directory “rimmer” (in the directory “home” in this example)
  - `~/` is shorthand for the path to your own user directory

# More on pathnames

- NOTE:
  - Any pathname beginning with / starts from the ROOT and is also an ABSOLUTE pathname.
  - Any pathname beginning with ~ starts from the HOME DIRECTORY
  - The pathnames: ../.. and ../../ are the same thing. The ending slash is optional and both paths will refer to the same directory.

# Try these examples:

- Suppose:
  - your username is “lister”
  - your current directory is “ships”
- Write the absolute pathnames for:
  - bin
  - Guitar
- Write the relative pathnames for:
  - Starbug
  - home
  - Guitar
  - rimmer
- Where could you use directory references? (~)

# Characters in filenames

- File names can contain any characters except “/”, but it is recommended that you use upper or lower case letters, numbers, and the characters “-” “.”
- For example although a file name could contain a space or spaces:

```
confusing name
```

commands using this would not work correctly unless you tell the shell to not break an argument at the spaces by quoting the filename.

```
rm “confusing name”
```

# Wildcards

- an asterisk “*” matches any number of characters in a filename
  - con* will match con, condor, constant.exe
  - *.c will match all files that end in .c
  - rm * will remove all the files in a directory
- a “?” matches any single character in a filename
  - b?t will match bit, bot, bat. It will not match bt or boot
- square brackets “[]” will match any **one** of the characters in the brackets. A hyphen “-” can be used to match any of a range of consecutive characters.
  - [bhr]at will match bat, hat and rat
  - chap[5-8].c will match chap5.c, chap6.c, chap7.c and chap8.c

Wildcard examples (in class)

# File Permissions

- 3 types of processes can access a file
  - *u*ser or owner: process spawned by user who created file
  - *g*roup: process spawned by members of the same group
  - *o*ther: process spawned by anyone else
- Permission types
  - *r*ead: access file or list directory
  - *w*rite: write to / remove file (directory)
  - e*x*ecute: run file as a program or enter directory

# Example Output

- Current permissions can be viewed using ls -l
  - First line is the number of disk blocks ( 1 block is 512 bytes) taken up by all the files

```
[sudhir@www scop3344]$ ls -al
total 596
drwxr-xr-x   3 sudhir fac   4096 Jan 22 17:38 .
drwxr-xr-x  11 sudhir fac   4096 Jan  3 18:30 ..
-rw-r--r--   1 sudhir fac   4631 Jan 18 16:10 Assignment1.txt
drwxr-xr-x   3 sudhir fac   4096 Jan  9 17:07 index_files
-rw-r--r--   1 sudhir fac  51693 Jan 22 17:35 index.html
-rw-r--r--   1 sudhir fac 247017 Jan 18 10:51 Lecture1.pdf
-rw-r--r--   1 sudhir fac  92123 Jan 16 09:05 Lecture2.pdf
-rw-r--r--   1 sudhir fac 175410 Jan 22 17:24 Lecture3.pdf
[sudhir@www scop3344]$
```

# Columns in the Display

- First entry in a line is the mode
  - The first character is *d* for directory, else - for a normal file
  - The remain 9 characters in groups of 3 are r, w, x permissions for user, group and other respectively (- indicates not having that permission)
- Second entry indicates number of links to the file (usually 1)
- Third entry is the user id of owner and the fourth entry is the group id
- Fifth entry is the number of bytes of the file
- Sixth entry is the date the file was last modified

# Changing Permissions

- Using the chmod command to set permissions
  - Numeric (using octal)
    - Directly set the permissions for u, g, o using each 3 bit value as an octal value
    - `chmod 754 lecture1.pdf`
      will set to 111 101 100 or rwx r-x r--
    - `chmod 700 lecture1.pdf`
      will set to 111 000 000 or rwx --- ---
    - `chmod 644 lecture1.pdf`
      will set to 110 100 100 or rw- r-- r--

# Changing Permissions (cont)

- Symbolic
  - Format: chmod [who] [operation] [permissions] <filename>
  - who is one or more of u, g, o
  - operation is + (add), - (remove), = (set)
  - Permissions are one or more of r, w, x

  - Examples

```
chmod go-rwx myfile.doc
chmod g+w myfile.doc
chmod u=rwx,g=rx,o=r myfile.doc
```