Some properties of directories

- directories are actually ordinary files
- information contained in a directory file simply has a special format
- every directory contains two special directory entries
  - ".." refers to parent directory in hierarchy
  - "." refers to the current directory (itself)
- '~' is used to denote a **home directory**
  - `% cd /home/user` ≈ `cd ~user`
    `% cd` ≈ `cd ~`

Directory commands

- Example:
  - `% cd /home`
- listing directories
  - `% ls`
    `se265mz bgates`
  - `% ls bgates`
    `w2k.src`
- relative pathnames
  - `% cd /home`
    `% lpr bgates/w2k.src`
    `% lpr ./bgates/w2k.src`
    `% lpr ./bgates/../bgates/w2k.src`

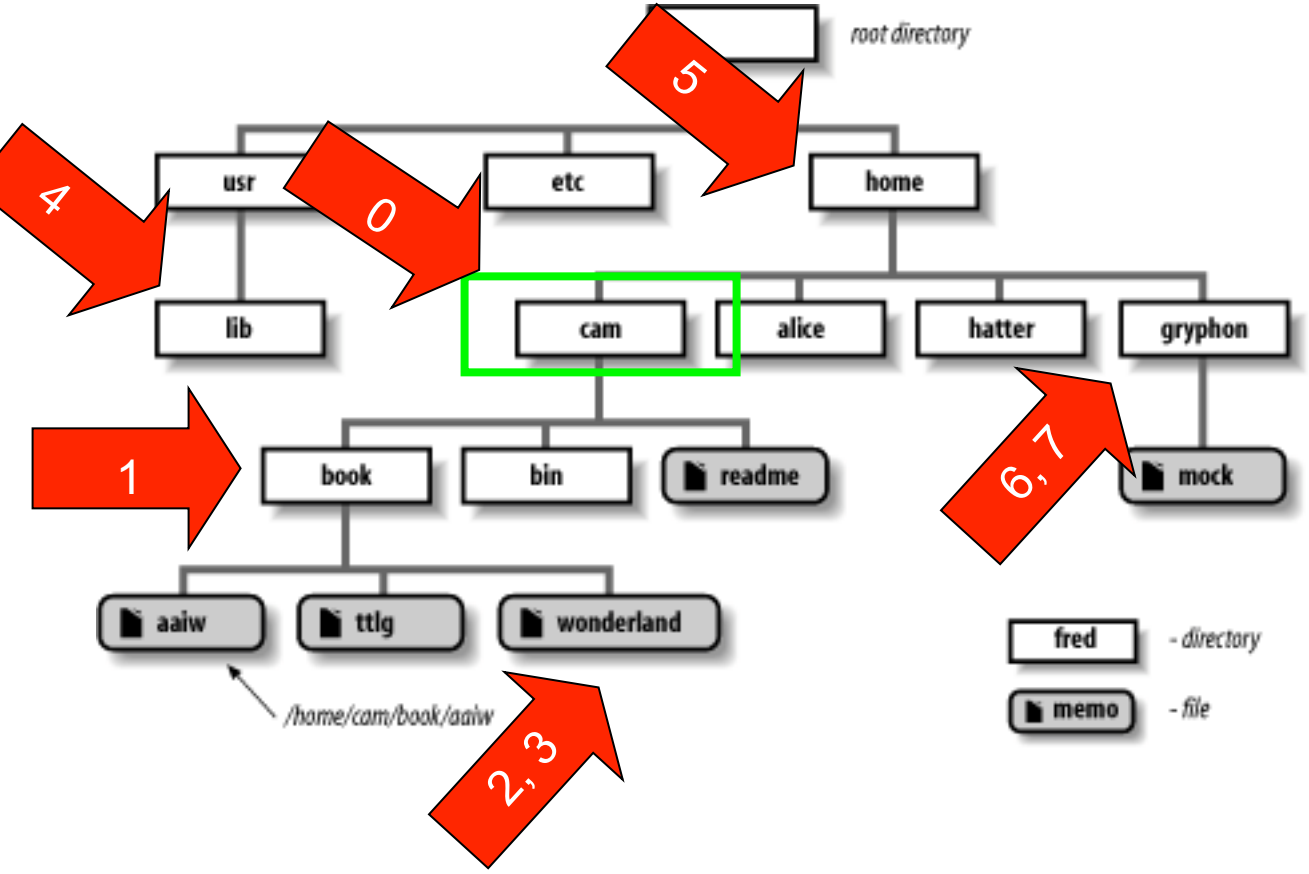
Directory commands (2)

- traversing directories
  - ```
    % cd /usr
    % ls
    ucb bin lib
    ```
- display the **current working directory**
  - ```
    % pwd
    /usr
    ```
- creating a **symbolic reference** to a file (i.e., like an alias"
  - ```
    % cd ~se265mz
    ```
  - ```
    % ln -s read.txt error.log
    % ls
    read.txt  error.log
    ```

“working” vs. “home” directory

- “Working” directory is the directory you are “in” at any given time.
  - Eliminates the need to continuously specify full pathnames for files and directories
  - “Relative pathnames” are locations worked out relative to the current working directory.
- “Home” directory is (usually) made to be your working directory upon logging into the system
  - Sometimes called the “login” directory
  - `/home/zastre` & `/home/seng265` are typical home directories

working directories




```
# "cam" is the logged-in user
#
# Each of the following commands
# assumes Cam's current directory
# is /home/cam (i.e., every item
# below assumes we reference
# from at red-arrow 0).
% cd book                    #1
% vi book/wonderland         #2
% vi ~/book/wonderland       #3
% cd /usr/lib                #4
% cd ..                      #5
% cd ../gryphon              #6
% cd ~gryphon                #7
% cd alice                   # ??
```

File attributes

- every plain file and directory has a set of **attributes**, including:
  - user name (owner of file)
  - group name (for sharing)
  - file size (bytes)
  - creation time, modification time
  - file type (file, directory, device, link)
  - permissions

```
% ls -l unix.tex test
-rwxr-xr-x  1  joe  users     200  Aug 29  14:39  test
-rw-r--r--  1  dmg  users   21009  Aug 29  14:39  unix.tex
```

Who has permission?

- permissions can be set for
  - user ("u") `[-rwx------]`: the file owner
  - group ("g") `[----rwx---]`: group for sharing
  - other ("o") `[-------rwx]`: any other
  - all ("a"): user + group + other
- **user**: the owner of the file or directory; owner has full control over permissions
- **group**: a group of users can be given shared access to a file
- **other**: any user who is not the owner and does not belong to the group sharing a file

What kind of permissions?

- files:
  - **read (r)** : allows file to be read
  - **write (w)**: allows file to be modified (edit,delete)
  - **execute (x)**: tells UNIX the file is executable
  - **dash (-)** **:** owner/group/other have no permissions
- directories:
  - **read (r)**: allows directory contents to be read (listed)
  - **write (w)**: allows directory contents to be modified (create, delete)
  - **execute (x)**: allows users to navigate into that directory (e.g, with the **cd** command)
  - **dash (-)** : owner/group/other have no permissions

chmod: set file permissions

- there are several ways to use "chmod"
  - use letter symbols to represent "who" and "what"

```
% chmod o+rx ~/.www/ppt    # other can read and cd "ppt"
% chmod u+x run.pl         # script "run.pl" executable
% chmod go-rwx ~/private   # removing access group & other
% chmod u=rwx,g=rx,o=x foobar.txt  # all permissions
```

  - can also use "octal" (base 8) notation, representing each three-bit field with an octal digit; $r \in \{0,4\}$, $w \in \{0,2\}$, $x \in \{0,1\}$

```
% chmod 751 foobar.txt      # specify all permissions
```

  - the following are different ways of setting "read-only" permission for a file

```
% chmod =r file
% chmod 444 file
% chmod a-wx,a+r file
```

Various & Sundry

- UNIX file names are case-sensitive
  - e.g., `myFile` and `myfile` are two different names, and the `logout` command cannot be typed as `Logout`
- commands are available to change the **owner** and/or **group** of a file; e.g. `chown, chgrp`
- **pager** is a command (`less`, `more`) used to display a text file one page at a time

```
% less unix.txt
```

- to quickly create a file

```
% touch unix.txt
% ls -l uxix.txt
-rw-r--r-- 1 zastre users  0  Aug 29 14:39  unix.tex
```

Introduction to UNIX (contd)

- The shell
- Basic command syntax
- Command types
- Getting help on commands
- I/O streams
- Redirection and pipelining
- Command sequences
- Console

the shell (again)

- the shell is the intermediary between you and the UNIX OS kernel
- it interprets your typed commands in order to do what you want
  - the shell reads the next input line
  - it interprets the line (expands arguments, substitutes aliases, etc.)
  - performs action
- there are two families of shells:
  - “sh” based shells, and “csh” based shells
  - they vary slightly in their syntax and functionality
  - we’ll use “bash”, the Bourne Again SHell (derivative of “sh”, known as the “Bourne shell”)
  - tip: you can find out what shell you are using by typing:
    ```
    echo $SHELL
    ```

basic command syntax

```
% cmd [options] [arguments]
```

- `cmd` is a builtin-shell or UNIX command
- `[options] = option*`
- `[arguments] = argument*`

option	*example*
`opt`	`a`
`-opt`	`-v`
`--optname`	`--verbose`
`-opt arg`	`-s 5`
`--optname arg`	`--size 5`

basic command syntax (2)

- `opt` is a character in `{a..zA..Z0..9}`
- `optname` is an option name; e.g., `--size, --keep`
- `argument, arg` is one of the following:
  - file name
  - directory name
  - device name, e.g., `/dev/hdb2`
  - number, e.g., `10, 010, 0x1af, …`
  - string, e.g., `"*.c", "Initial release", …`
  - `…`

command types

- commands can be:
  - built into the shell (e.g., `cd, alias, bg, set,`…)
  - aliases created by the user or on behalf of the user (e.g., `rm='rm -i', cp='cp -i', vi='vim'`)
  - an executable file
    - binary (compiled from source code)
    - script (system-parsed text file)
- Use the `type` command to determine if a command is builtin, an alias, or an executable.

```
% type rm
rm is aliased to ‘rm -I’
```

some simple commands

`% cat [file1 file2 ...]`

- (catenate) copy the files to stdout, in order listed

`% less [filename]`

- browse through a file, one screenful at a time

`% date`

- displays current date and time

`% wc [filename]`

- (word count) counts the number of lines, words and characters in the input

`% clear`

getting help on commands

- You can ask for help in several ways.
- Display a description of a shell command (builtin)

```
% help times
```

- Display a long description of a command (from section *n* of manual)

```
% man [n] chmod
```

- Display a one line description of a command

```
% whatis gcc
gcc   gcc (1) - GNU project C and C++ compiler
```

- Display a list of commands related by a keyword

```
% apropos cdrom
autorun (1) - automatically mounts/unmounts CDROMs...
xplaycd (1) - X based audio cd player for cdrom drives
```

- Many commands provide their own help

```
% somecmd -h
% somecmd --help
```