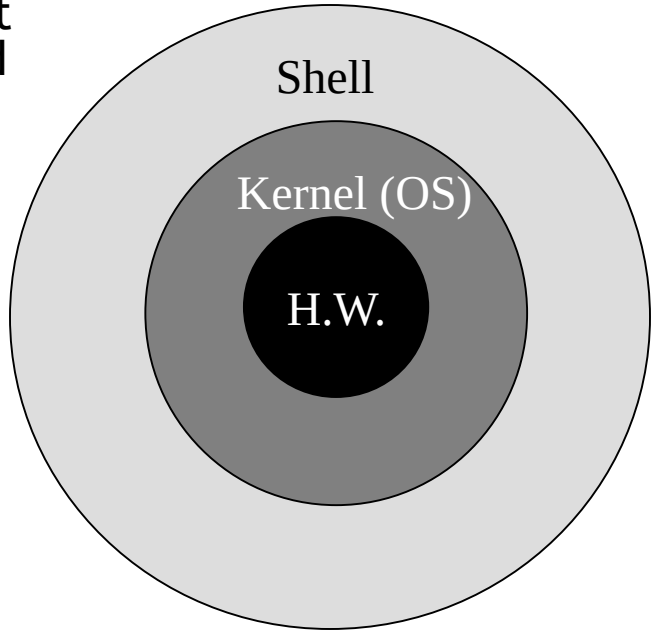
Unix Shell & File System

UNIX Operating System

- A multiuser, multitasking operation system
- The Kernel
  - The heart of the UNIX operating system. It is loaded into memory at boot-up time and manages the system
    - Create and control processes
    - Talk to the hardware and devices
    - Manages memory and storage (file system)
- The Shell
  - A utility program, a command interpreter
  - Interface between the user and the kernel
- The Application Programs
  - Compilers
  - Text editors
  - Mail utilities
  - Etc…



UNIX/Linux Shells

- The SHELL
  - Ritchie & Thompson's paper:
    - `The shell is a command line interpreter. It reads lines typed by the user and interprets them as requests to execute other programs.`
    - `A command line consists of the command name followed by arguments to the command, all separated by spaces`
  - A special program used as an interface b/w user and kernel (OS)
  - It starts up when you log on the system (not true for Linux desktops)
  - First significant, standard UNIX shell was introduced in 1979 (Bourne shell)
- UNIX shells
  - Thompson Shell (1971) ->Bourne shell (1977); C shell; Korn shell
- Linux shells (GNU shells)
  - Bash: GNU Bourne Again shell
  - TC shell, a popular extension of C shell
  - Z shell, a popular extension of Korn shell
  - Now "**`dash`**" from Ubuntu distribution
    - Debian Almquist SHell
    - A tale of two shells: bash or dash: https://lwn.net/Articles/343924/

SHELL Variables

- SHELL variables
  - **Environmental variables**
    - Available to all the shells (what does this mean?)
    - HOME, PATH, PWD, LD_LIBRARY_PATH, SHELL, etc. (by convention, they are all capitalized)
    - "env", command lists all the current defined environment variables
    - "printenv" does the same thing
    - To set an environment variable

      export VARNAME=VALUE # in bash
  - **Local variables**
    - For a specified shell only (what does this mean?)
    - To set a local variable: `name="Amy Lin"`
  - To clear a variable: "`unset VARNAME`"
- Prefix a "`$`" sign, `$varname` when referencing the variable

Display Values of Variables

- echo : display (print) a line of text to the screen
  - `echo "My home directory is $HOME"`
- The `echo` command and its options
  - `-n`: suppress newline at the end of a line output
    - This is useful when you want to continue to write on the same line
  - `-e`: enable backslash interpretation of the escape sequences, such as `\t (tab space)`, `\n` (EOL)

`echo "you are so \t nice "` ➔ `you are so \t nice`

`echo -e "you are so \t nice "` ➔ `you are so  nice`

- `echo ${varname}` ➔➔ `echo $varname`
  - Use curly bracket for string concatenation
  - `name=${variable}ABC`
- Learn more with command: `man echo`

The Three Types of Quotes

- Single quotes
  - no expansion or substitution, display string literally, including the special characters
  - `echo '$PATH'` ➔ `$PATH`
- Double quotes
  - allow variable and command substitution, also preserve white spaces

    `echo "here are 5 space:  .  "` (five space)

    `echo here are 5 space:  .` (only one space displayed!)
- Backslash (`\`)
  - Line continuity
  - Print some special characters, such as
    - $: `echo It costs me\$500`
    - \: `echo \\n` ➔ *\n*

Env. Variables: PATH & HOME

- `echo $PATH` gives:

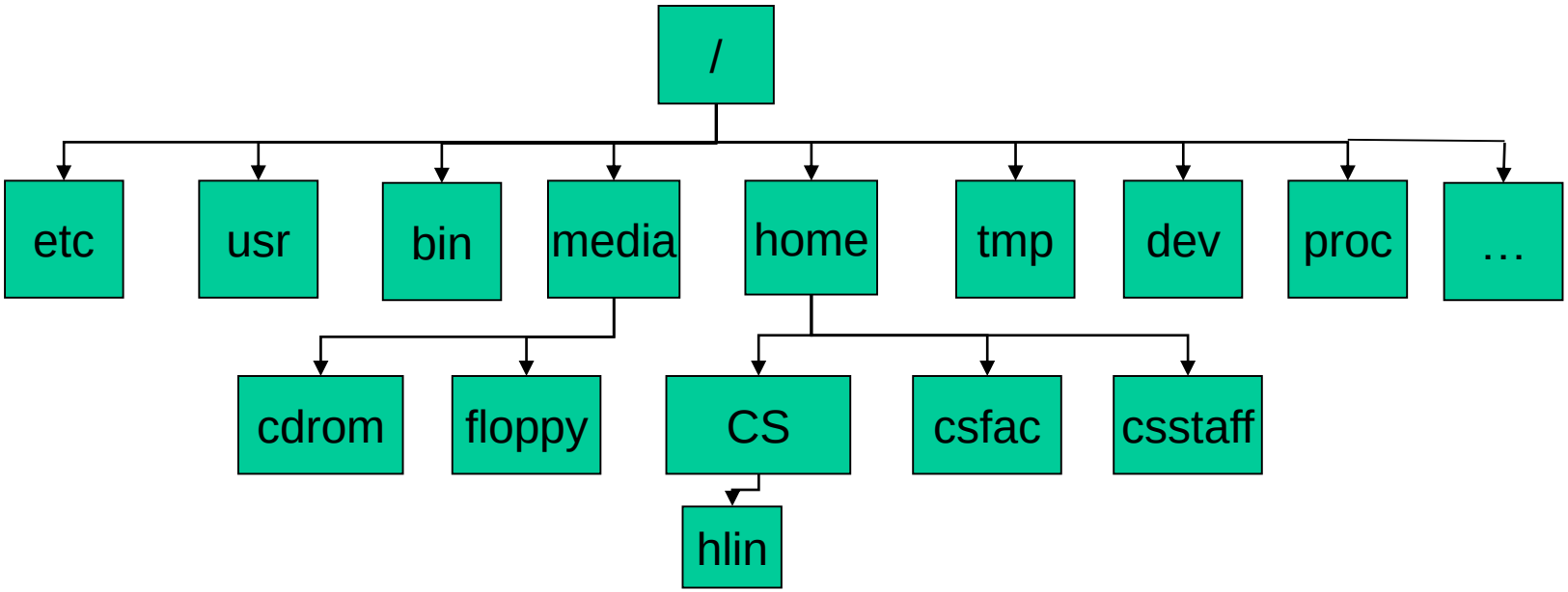
  /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games

- *`PATH` is a colon-separated list of directories used by the shell when searching for a program.* The order of the search is from left to right
  - The search will stop as soon as the "program" is found
  - If the "program" is at multiple places, the one found first (from left) will be used
  - In case the program is not found, the shell sends error message: program: not found
  - System defined PATH and user-defined PATH
    - The default PATH is system-dependent, and is set by the administrator who installed the Shell
    - The user-defined path is added to the system-defined path and normally is defined in a shell initialization file, such as .bashrc on a LINUX system
- **HOME** is where you are after you logon the system, always referred as "your home directory"
  - HOME can also be modified in your shell start-up file, such as `.bashrc` on a Linux system

UNIX File System

- **Everything is a file in UNIX/Linux**
- There are SEVEN types of FILE in three categories
  - Ordinary files (or regular files)
    - Regular Files
      - Binary program (executables)
      - Text files (source codes)
    - Symbolic links (similar to the shortcut on Windows), is a type of file that points to another (of any types)
  - Directory Files (folders): file that contains other files
  - Special Files
    - Character special file.
    - Block special file, such as hard drive
    - Pipe, also called FIFO: a type of file used for communication between processes running on the same system.
    - Socket: a type of file used for network communication between processes on different systems

- On UNIX, the files are organized into a tree structure
- The root of the tree is named by the character `'/'`.
- The first few levels of the tree can look like this:



- Use program “tree” to list contents of directories in a tree like format
- The first level directories might not be on the same hard drive
- Never specify drive name like on Windows…

Directory

- On UNIX, "/" is the root of the entire file system
  - On Windows, C:\dir1\dir2\file1, D:\dir1\dir2\file2 which contain
    - name of the drive
    - use backward slash "\" for the nested path
  - On Unix, start with root "/": /home/CS/hlin/homework1.txt
    - **No hard drive name**
    - **Use forward slash, "/" for the nested path**
- The file system is "partitioned" into many "directory" files" for different purposes
  - `/bin, /usr/bin`: commonly used system and application programs
  - `/sbin, /usr/sbin`: not commonly used system programs
  - `/etc`: system configuration files, etc
  - `/lib, /usr/lib`: hold the libraries programs needed
  - `/proc`: holds files for system information and info for the running programs
  - `/dev`: hold all the device files
  - `/home`: user files
    - Each user has his/her own home directory, such as /home/CS/jsmith
  - `/tmp`: special temporary files, everyone can use it (create file and directories there), but it's very volatile, expect to be purged often

Navigate the File System

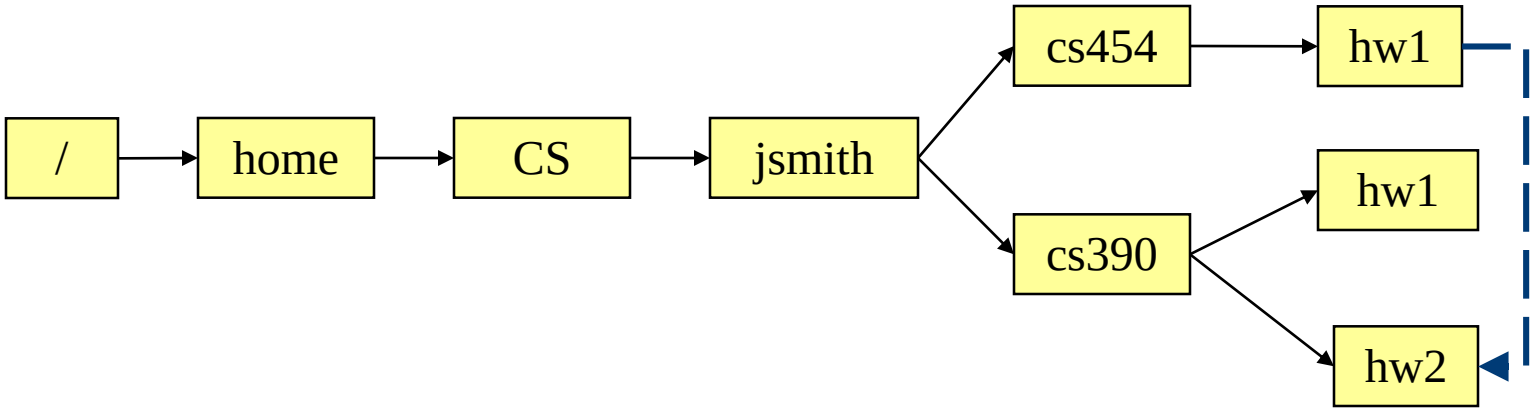
- Where are you when you log on a system?
  - You are always under your home directory by default, such as **`/home/CS/hlin`** for me
  - **`echo $HOME => ??`**
- Some useful shell commands when navigating the file system
  - pwd: Present Working Directory (env for variable PWD)
  - cd : Change Directory
    - "`cd /`" takes you to the "root" of the file system
    - "`cd`" without argument takes you back to the home directory
    - "`cd –`" (with an option of short dash) takes you to the previous directory
  - "`ls`": list the content of the current directory
    - "`ls filepath`" will list the content under path **`filepath`**
- absolute path or relative path
  - Absolute path: path starts from the root, the forward slash /

    ex: **`cd /home/CS/hlin/cs390`**
  - Relative path
    - Relative to the current working directory or some other directory

    ex: `cd cs390` (←→ `cd ./cs390_fall13`)

Special Directories

- Three special directories
  - ".": the current directory
  - "..": the parent directory (one level up in file tree)
  - "~": refer to the home directory, such as /home/csuser/jsmith
- What do the following commands do?
  - `cd ~`
  - `cd ..`
  - `cd ../..`
  - `cd`
  - `cd $HOME`
  - `cd $HOME/cs390` ⇔ `cd ~/cs390`



- How to access/change to directory "`hw2`" as shown when jsmith is in `/home/CS/jsmith/cs454/hw1`?
  - Using absolute path:

```
cd /home/csuser/jsmith/cs390/hw2
cd $HOME/cs390/hw2
cd ~/cs390/hw2
```

  - Using relative path: `cd ../../cs390/hw2`

Operations for Directories

- Create directory with "`mkdir`"
  - `mkdir cs390`
    - gives error message if directory exists (use -p option?)
  - **`mkdir -p cs390/hw1`**
    - Create directory and parent directories if they do not exist
      - Create all the non-exist directories on the path
    - Silence if the directory exists
- Remove directories/files
  - Remove empty directory: **`rmdir dirname`**
  - Remove non-empty directory: `rm -r dirName`
    - `-f` option, the files/directories will be removed w/o asking
    - `-i` option, it always ask you to confirm any deletion

**Special Note:

* There is no "deleted" folder or trash bin or recycle bin on UNIX, once a file or directory is removed, it is gone!

* If you are not sure what option to use, read rm's man page: `man rm`

Operations for Regular Files

- Commands to display content of file to the screen

  syntax: `command <filename>`

  - `cat:` Display the whole content of file to the screen
  - `more` and `less` commands
    - Display a text file one page at a time (hit spacebar for next page)
    - "less" can work on gzip (compressed) files
    - To quit before reaching the end of file, hit either :ctrl-C, or 'q'
  - `head -<n>` and `tail -<n>` commands
    - `head -10 file.txt` ⇔ `head -n 10 file.txt`
    - List the first `n` or last `n` lines of the file
    - Without -n option, default is to display 10 lines
- Rename (move) files from one location to another
  - `mv srcfile newfile`
- Delete regular files with command `rm`
  - `rm srcfile newfile …`

Copy Files

`cp src dest`

- “copy” regular files: **`cp src_file dest_file`**
- “copy” directory: **`cp -r src_dir  dest_dir`**
  - recursively copy the content of directory `src_dir` to a new directory dest_dir
- All the files contain “path”:
  - **`cp -r ~/cs390/hw1 ~/cs390/hw2`**

scp -- Remote File Copy

- Transfer local file(s) to a remote system

```
scp local_file1 local_file2 linh@zeta.itsc.uah.edu:
scp -r local_dir linh@zeta.itsc.uah.edu:
```

- Transfer files from remote system to local

```
scp linh@pearl.itsc.uah.edu:remove_file .
( the dot refers to the current local place)
scp -r
  linh@pearl.itsc.uah.edu:/path/to/remote_dir
  .
```

Note: (you will be prompted for password)

Link Files

- Hard link files

```
ln fileA fileB
```

A hard link file `fileB` of `fileA` is created

If we run "ls", it will appear that a new file fileB has been created as a copy of fileA (as if `cp fileA fileB`, but NOT the same!!)

- Symbolic files

```
ln -s fileA fileC
```

Input / Output

- File Descriptors
  - A small unsigned integer, an index into a file descriptor table maintained by the kernel and used by the kernel to reference open files and I/O streams.
- The first three descriptors are reserved for the standard I/O (with terminal)
  - 0 – stdin: read input from terminal (keyboard)
  - 1 – stdout: print to terminal
  - 2 – stderr: print error to terminal by default
    - Note: csh/tcsh does not have stderr

Standard I/O Redirection

When a file descriptor is assigned to something other than a terminal, it is called Standard I/O redirection

- "<" is used for STDIN redirection
  - Meaning: program will read input from a file, rather than the terminal.
- ">" and ">>" used for STDOUT redirection
  - Program will write the output to a file rather than dumping them to the screen
  - >: create a new file, or overwrite the existing one
  - >>: create if not exist, otherwise append the content to the existing one.

- Redirect the standard output (stdout) to a file
  - `ls > ls.txt`
    - The output of "ls" command is redirected from the terminal (stdin) to file who.txt
  - `who > who.txt`
    - The output of command `who` is redirected from the terminal (stdout) to file `who.txt`
    - In case of error, the error message will be dumped to the screen
  - `date > date.txt  2> error.txt`
    - The output of command `date` is redirected from the terminal to file `date.txt`, any error will be redirected to file `error.txt`
  - `cat file2 file3 >> file1.txt`
    - concatenate file2 and file3 to file1.txt
- Use "<" to read input from a file instead of stdin
  - Using mail utility non-interactively:

    `mail -s "datafile" linh@uah.edu < datafile.txt`
  - `cat <file.txt` (normally, the < is dropped for simplicity)

Redirect to/from `pipe`...

- pipe
  - The output of one process is sent as the input of another process
  - It is the oldest form of UNIX Inter-Process Communication (IPC)
  - Allows processes to communicate with each other on the same system

- Syntax of `pipe` command
  - `who | wc`
    - Count the number of people currently logon the system

File Archive & Compression

- File Archiving
  - Store a group of files in one file (or on tape in the old days)
  - Easy for file backup and file transfer
- Archiving files with "**tar**" (tape archive), no data compression

  tar cvf file.tar filedir1 filedir2 file1 …
  - "c": create
  - "v": verbose
  - "f": create a tar file

  **Note: The files to be archived can be directories and regular files
- "**tar**" with compression option (`man tar`)
  - "z" for "gzip" compression

    `tar czvf file.tar.gz file1 file2 dir1 …`
  - "j" for "bzip2" compression

    `tar cjvf file.tar.bz2 file1 file2 dir1 …`
  - "J" for "xz" compression

    `tar cJvf file.tar.xz file1 file2 dir1 …`

Operations on Archive Files

- Examine the content of a compressed archive file
  - `tar tzvf file.tar.gz`
  - `tar tjvf file.tar.bz2`
- Extract files from archive file
  - `tar xvf file.tar`
- Decompress archive files: replacing “c” with “x”
  - `tar xzvf file.tar.gz  < -C destdir >`
  - `tar xjvf file.tar.bz2`
  - `tar xzvf /path/file.tar.gz`
- To decompress/extract files from a Window zip file
  - `unzip file.zip`
- How to open gz, bz2 files on Window?
  - 7-Zip: a free file archiver for MS Windows Users

"gzip"

- gzip: GNU zip
  - Works on regular files ONLY
    - By default, it will create a gzip file replacing the original file(s)

    `gzip foo.txt` ➔ `foo.txt` will be replaced by `foo.txt.gz`
    - To keep the original file(s)

    `gzip -c foo.txt >foo.gz` (-c option: write output on standard out)
  - Compress multiple files -- using pipe

    ```
    cat file1 file2 | gzip -c >file.gz
    ls | gzip -c >ls.gz
    ```
- decompress gz files
  - `gunzip foo.gz` ⇔ `gzip -d foo.gz`
    - The gz file will be replaced by the decompressed file
  - `gunzip -c foo.gz >foo.txt`
  - `zcat file.gz` ⇔ `gunzip -c file.gz`
- RAR (WinRAR)
  - Support UTF
  - On Ubuntu, package: unrar to extrat files from rar archives

openssl

- "**openssl**" to encrypt a file

```
openssl rc2 -in hw1.tar.gz -out hw1.tar.gz.rc2 -k 123456
```

- "**openssl**" to decrypt a file

```
openssl rc2 -d -in hw1.tar.gz.rc2 -out hw1.tar.gz -k 123456
```

Utilities for "print"

- In lab N328, **`lpstat -a`** #list all the available printers
  - **`lpstat -d`** – shows the current default destination
- "lp", "lpr"
  - `lpr -P laser329a filename`
  - `lp -d laser329a filename` (-d: destination)
- "a2ps" (anything to postscript)
  - format files for printing on a postscript printer
  - **`a2ps -P laser329a (--printer=laser329a) filename`**
    - By default, print two columns (pages) per page
    - "-R": print in portrait
    - --columns=1": will print one column (page) per page
    - More other options…
- Get more info with "`man lp`", "`man lpr`", "`man a2ps`", "`man lpstat`", etc and the "see also…" at the end of the man pages

NOTE: For all your assignments, if hardcopy is required, you MUST use **`a2ps`** to print out your work (two pages per sheet). Other formats are not accepted!