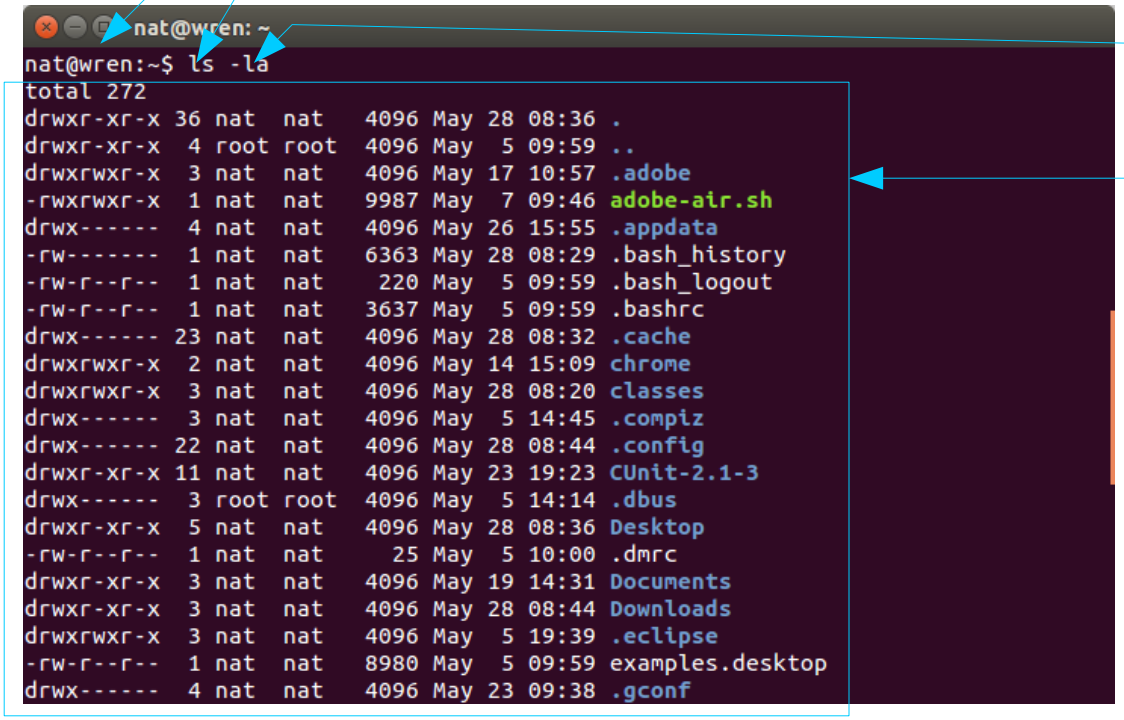
Week 2 Lecture 3

Unix

Terminal and Shell

Terminal

- Prompt
- Command
- Argument
- Result



Shell Intro

- A system program that allows a user to execute:
  - shell functions (e.g., ls -la)
  - other programs (e.g., eclipse)
  - shell scripts
- Linux/UNIX has a bunch of shell programs
  - We will use bash

Typing in bash

- Left and right arrows move left and right
- Backspace and delete characters
- Ctrl-a: beginning of line; Ctrl-e: end of line
- <Tab> completes file names
  - Always use <Tab> when typing file or command names to minimize misspelling

Command Format

- Format: command and 0 or more arguments:
  `% commandname [arg1] ... [argN]`
- % sign represents prompt here and hereafter.
- Arguments can be
  - options (switches to the command to indicate a mode of operation) ; usually prefixed with a hyphen (-) or two (--) in GNU style
  - non-options, or operands, basically the data to work with (actual data, or a file name)

Command Types

- Shell functions: The shell executes the commands when the enter key is hit, and prints results onto the terminal.
- Other programs: The shell executes the compiled program.
- Shell scripts: Programs consisting of shell functions, other programs and shell scripts.

exit

- Exit from your terminal session.
  - `% exit`
  - Ctrl-d
  - Close the window by clicking on the 'x'

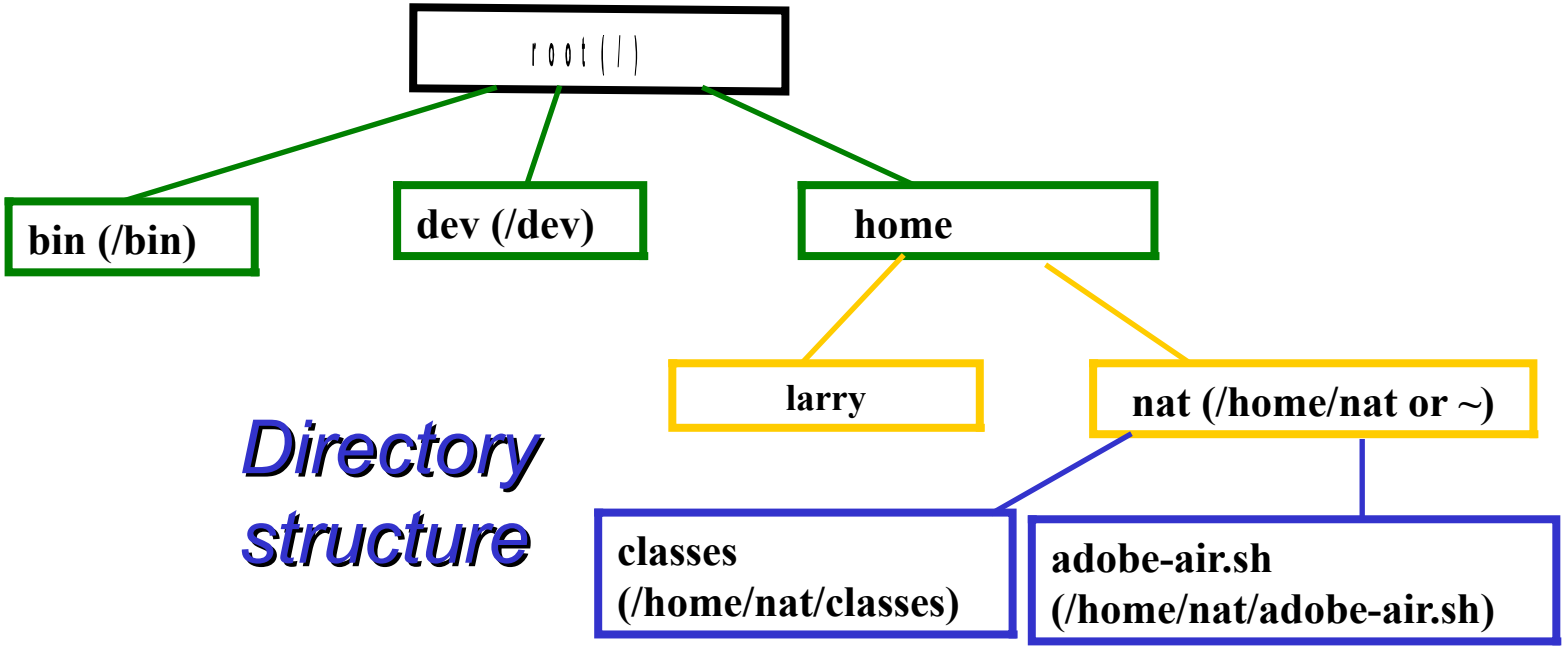
Files

Files and directories

- Unix provides files and directories.
- A directory contains files and other directories.
- The directories and files are called the file system.
- The directory that contains all other directories is called the “root” directory, whose name is written “/”.

Example

- Every file has a name (root's name is written /
- Every directory has two paths
  - Absolute paths start with the root
  - Relative paths start from the current directory



Permissions

- Every file and directory have a set of permission for:
  - Self: what the file owner can do
  - Group: what the file's group can do
  - All: what everyone can do
- Each set of permission has three settings
  - Read: the person or group can read the file
  - Write: the person or group can write the file
  - Execute: the person or group can execute the file

Permission example

drwxrwxr-x 17 nat nat 4096 May 28 09:29 sp

- Directory
- Self: read write execute permissions
- Group: read write execute permissions
- All: read execute permissions
- Owner is “nat”
- Group is “nat”
- Last touched on May 28 9:29 AM
- Directory name is “sp”

Wild Cards

- You can refer to files and directories using the wild cards: * and ?.
  - * matches any string of characters
    - a*z matches abz, abbz, abcdez, and azzz but not abcza
    - az* matches azzz, aza but not abz
  - ? matches a single character
    - a?z matches abz, and acz, but not abbz, or abza

Execute Permission

- **Files**: execute means the file is a command
- **Directories**: execute means the user can see the contents

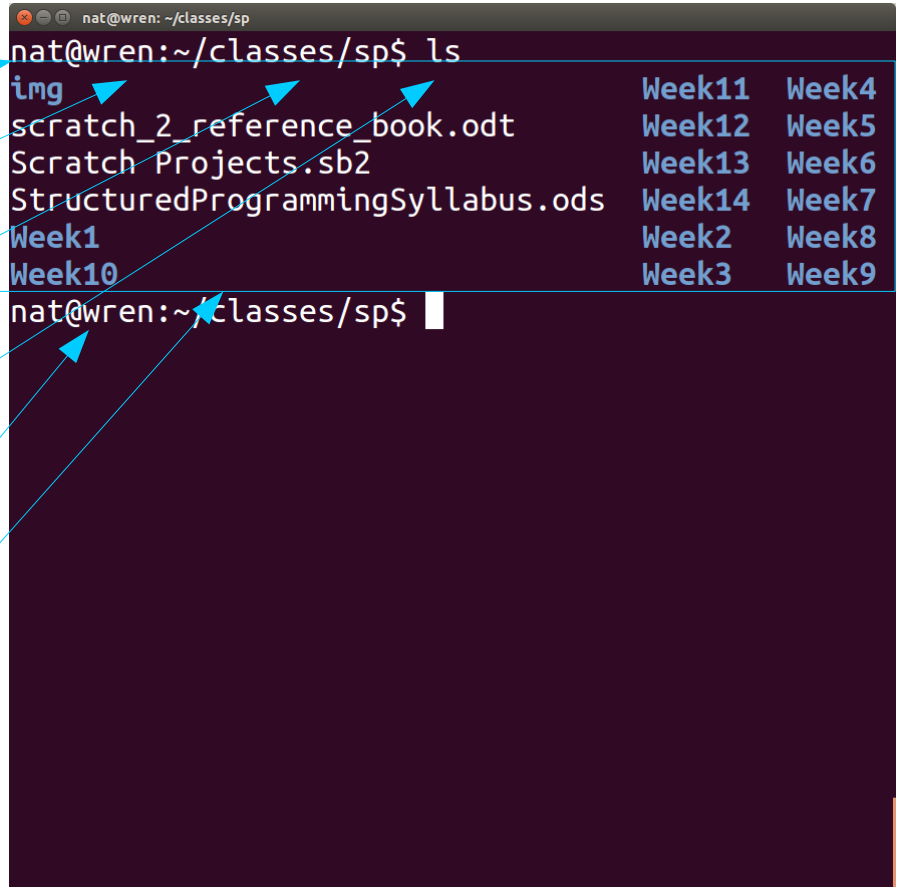
File Commands

Print Working Directory: pwd

- Prompt
  - Me
  - Computer
  - Directory
- Command: pwd
- Current directory
- Next Prompt

Directory contents: ls

- Prompt
  - Me
  - Computer
  - Directory
- Command: ls
- Directory contents
- Next Prompt



ls

- List directory contents
- Has whole bunch of options, see man ls for details.
- `% ls`
  - all files except those starting with a “.”
- `% ls -a`
  - all
- `% ls -A`
  - all without “.” and “..”
- `% ls -F`
  - append “/” to dirs and “*” to executables
- `% ls -l`
  - long format
- `% ls -al`
- `% ls -lt`
  - sort by modification time (latest - earliest)
- `% ls -ltr`
  - reverse

Finding Permission: ls -l

- The -l parameter displays permissions

```
nat@wren: ~/classes/sp
nat@wren:~/classes/sp$ ls -l
total 92
drwxrwxr-x 2 nat nat  4096 May 14 15:43 img
-rw-rw-r-- 1 nat nat 25040 May 23 10:05 Schedule.ods
drwxrwxr-x 2 nat nat  4096 May 28 10:06 Scratch
drwxrwxr-x 3 nat nat  4096 May 28 08:22 Week1
drwxrwxr-x 2 nat nat  4096 May 23 09:56 Week10
drwxrwxr-x 2 nat nat  4096 May 23 10:19 Week11
drwxrwxr-x 2 nat nat  4096 May 23 10:21 Week12
drwxrwxr-x 2 nat nat  4096 May 23 10:57 Week13
drwxrwxr-x 2 nat nat  4096 May 23 11:44 Week14
drwxrwxr-x 3 nat nat  4096 May 28 10:02 Week2
drwxrwxr-x 3 nat nat  4096 May 28 09:27 Week3
drwxrwxr-x 2 nat nat  4096 May 22 09:29 Week4
drwxrwxr-x 2 nat nat  4096 May 22 14:19 Week5
drwxrwxr-x 2 nat nat  4096 May 22 16:21 Week6
drwxrwxr-x 2 nat nat  4096 May 23 14:19 Week7
drwxrwxr-x 2 nat nat  4096 May 23 14:18 Week8
drwxrwxr-x 2 nat nat  4096 May 22 22:05 Week9
nat@wren:~/classes/sp$
```

Seeing hidden files: ls -a

- The -a parameter display files that start with a '.'
- '.' is the name of the current directory
- '..' is the name of the parent directory

```
nat@wren: ~/classes/sp
nat@wren:~/classes/sp$ ls -la
total 100
drwxrwxr-x 18 nat nat  4096 May 28 10:06 .
drwxrwxr-x  3 nat nat  4096 May 28 09:56 ..
drwxrwxr-x  2 nat nat  4096 May 14 15:43 img
-rw-rw-r--  1 nat nat 25040 May 23 10:05 Schedule.ods
drwxrwxr-x  2 nat nat  4096 May 28 10:06 Scratch
drwxrwxr-x  3 nat nat  4096 May 28 08:22 Week1
drwxrwxr-x  2 nat nat  4096 May 23 09:56 Week10
drwxrwxr-x  2 nat nat  4096 May 23 10:19 Week11
drwxrwxr-x  2 nat nat  4096 May 23 10:21 Week12
drwxrwxr-x  2 nat nat  4096 May 23 10:57 Week13
drwxrwxr-x  2 nat nat  4096 May 23 11:44 Week14
drwxrwxr-x  3 nat nat  4096 May 28 10:02 Week2
drwxrwxr-x  3 nat nat  4096 May 28 09:27 Week3
drwxrwxr-x  2 nat nat  4096 May 22 09:29 Week4
drwxrwxr-x  2 nat nat  4096 May 22 14:19 Week5
drwxrwxr-x  2 nat nat  4096 May 22 16:21 Week6
drwxrwxr-x  2 nat nat  4096 May 23 14:19 Week7
drwxrwxr-x  2 nat nat  4096 May 23 14:18 Week8
drwxrwxr-x  2 nat nat  4096 May 22 22:05 Week9
nat@wren:~/classes/sp$
```

cat

- Display and concatenate files.
- `% cat`
  - Will read from STDIN and print to STDOT every line you enter.
- `% cat file1 [file2] ...`
  - Will concatenate all files in one and print them to STDOUT
- `% cat > filename`
  - Will take whatever you type from STDIN and will put it into the file `filename`
- To exit `cat` or `cat > filename` type Ctrl+D to indicate EOF (End of File).

more / less

- Pagers to display contents of large files page by page or scroll line by line up and down.
- Have a lot of viewing options and search capability.
- Interactive. To exit: ‘q’

less

- `less` ("less is more") smarter than the `more` command
- to display contents of a file:
  - `% less filename`
- To display line numbers:
  - `% less -N filename`
- To display a prompt:
  - `% less -P"Press 'q' to quit" filename`
- Combine the two:
  - `% less -NP"Blah-blah-blah" filename`
- For more information:
  - `% man less`

touch

- By *touching* a file you either create it if it did not exists (with 0 length).
- Or you update it's last modification and access times.
- There are options to override the default behavior.
- `% touch file`
- `% man touch`

cp

- Copies files / directories.
- `% cp [options] <source> <destination>`
- `% cp file1 file2`
- `% cp file1 [file2] … /directory`
- Useful option:
  - `-i` to prevent overwriting existing files and prompt the user to confirm.
  - `-r` to copy a directory and all of its contents

mv

- Moves or renames files/directories.
- `% mv <source> <destination>`
  - The <source> gets removed
- `% mv file1 dir/`
- `% mv file1 file2`
  - rename
- `% mv file1 file2 dir/`
- `% mv dir1 dir2`

rm

- Removes file(s) and/or directories.
- `% rm file1 [file2] ...`
- `% rm -r dir1 [dir2] ...`
- `% rm -r file1 dir1 dir2 file4 …`
  - `-r` option removes directory and all of its contents

mkdir

- Creates a directory.
- `% mkdir newdir`
- Often people make an alias of `md` for it.

cd

- Changes your current directory to a new one.
- `% cd /some/other/dir`
  - Absolute path
- `% cd subdir`
  - Assuming `subdir` is in the current directory.
- `% cd`
  - Returns you to your home directory.

rmdir

- Removes a directory.
- `% rmdir dirname`
- Almost equivalent:
  - `% rm -r dirname`
    - `rmdir` will complain if the file isn't empty
    - rm -r will not.

ln

- Symbolic link or a “shortcut”.
- `% ln -s <real-name> <fake-name>`

chmod

- Changes file permissions
- Possible invocations
  - `% chmod 600 filename`
    - -rw------- 1 user group 2785 Feb 8 14:18 filename
  - `% chmod u+rw filename`
    - the same thing, more readable
  - For the assignment:
    - `% chmod u+x myshellscript`
      (mysshellscript is now executable)
    - -rwx------ 1 user group 2785 Feb 8 14:18 myshellscript

Why does 600 = rw-?

- Permissions are represented as an octal number (i.e., 3 bits)
  - r: 4
  - w: 2
  - x: 1
- 600 is self rw or 4+2, 0, 0
- 755 is rwxr-xr-x or 4+2+1, 4+0+1, 4+0+1
  - Common directory permissions
- 644 is rw-r—r-- or 4+2, 4, 4
  - Common file permissions