CS114: Lecture 4
IO, Redirection, Job control

Please pick up homework 2

Random tip of the day

- Searching in big manpages
  - Press / then type a string and hit ENTER
  - Press n to search again
  - Very helpful with shell manpages

Standard files

- Standard input
  - `stdin` in C, `System.in` in Java
  - Reads from the terminal (console) unless redirected
- Standard output
  - `stdout` in C, `System.out` in Java
  - Writes to terminal unless redirected
- Standard error
  - `stderr` in C, `System.err` in Java
  - Like standard output, used for error messages

Redirecting input and output

- To redirect standard input to come from a file, add `< filename` to the command
- To redirect standard output to go to a file, add `> filename` to the command
  - `ls ~ > files-in-my-home-dir`
- To append standard output to a file, add `>> filename` to the command

Some commands that use stdin/stdout

- `wc` – count the number of lines, words, and characters in stdin
- `head -37` – print the first 37 (or whatever) lines of stdin to stdout
- `tail -26` – print the last 26 (or whatever) lines of stdin to stdout
- `sort` – sort the lines of stdin, print to stdout. Many options

Examples of redirection

- Write the first 10 lines of the file `emails` to the file `10-emails`
  - `head -10 < emails > 10-emails`
- Count the number of lines in `/etc/passwd`
  - `wc < /etc/passwd`
- Use less to read the file `.cshrc`
  - `less < .cshrc`
  - Note: `less .cshrc` *also works* – but this is different behavior
- El cheapo text "editor": `cat > filename`

Chaining programs together: pipes

- Connect stdout of `prog1` to stdin of `prog2`
  - `prog1 | prog2`
- List the files in /bin
  - `ls /bin`
- List the files in /bin, sorted in reverse order
  - `ls /bin | sort -r`
- List the files in /bin, sorted in reverse order, and count the lines in the result
  - `ls /bin | sort -r | wc`
- ...

Job control

- UNIXes can run several commands at once
  - In the same window!
- Each pipeline running is a “job”
- A job can be
  - Running in the foreground
    - This is what usually happens
    - Can read from terminal
    - Can write to terminal
  - Running in the background
    - Can write to terminal
  - Suspended (stopped)

List the jobs running in this shell

- `jobs`

```
[ejb34@gala ~]% jobs
[1]  - Running
 yes > /dev/null
[2]  + Suspended
 vim .cshrc
```

- `+` = current job
- `-` = previous job

Moving jobs between states

- Foreground -> suspended
  - Press Ctrl-Z (usually)
- Starting as a background job
  - Append `&` to the command
- Switch job to foreground
  - `fg %`*`job-number`*
  - `fg` (switches current job)
- Switch job to background
  - `bg %`*`job-number`*
  - `bg` (switches current job)

Killing jobs

- Ctrl-C (kills the foreground job)
- `kill %`*`job-number`*
  - Tells job “please, I'd appreciate if you would stop”
- `kill -9 %`*`job-number`*
  - “Terminate with extreme prejudice”

Processes

- Every program that runs is a process
  - Job = 1 or more processes
- To see a list of processes:
  - `ps`
  - `top`
    - Press 'q' to quit
- `ps` by default prints only processes running in this shell, owned by you