Class 3: Unix and You

Overview

1. Announcements

2. Review

3. Q&A

4. Exercises

5. Basic assignment

Announcements

- Basic/Adv - Git 1 due September 28

- Basic/Adv - Unix due October 5

- Git 1 survey closing tonight
  - Class surveys close in a week!

Review

- Programs vs processes

- Processes have extra info
  - PID

  - File descriptor table

- Signals are another communication mechanism
  - ^C: SIGINT

  - ^Z: SIGTSTP

  - `kill`

Review

- Everything is a file
  - Files are a read/write interface

  - Become a general communication mechanism

  - Can be used to represent anything that'd work with such an interface

  - Terminals are files as well!

- Files have metadata including permissions
  - Represented in decimal/octal, with three bits

  - `rwx`

    ```
    user    group   other
    rwx     rwx     rwx
    110     100     100
    6       4       4
    ```

Shell operation

1. Receive a command from a file or terminal input
   - `ls -l $HOME > some_file`

2. Splits it into tokens separated by **white-space**
   - Takes into account *"quoting"* rules
   - `ls`, `-l`, **`$HOME`**, `>`, `some_file`

3. Expands/substitutes special tokens
   - `ls`, `-l`, **`/home/brandon`**, `>`, `some_file`

4. Perform file redirections (and making sure they don't end up as command args)
   - `ls`, `-l`, `/home/brandon`; (set standard output to **`some_file`**)

5. Execute command (remember our friend `exec()`?)
   - `argc` = 3
   - `argv` = [`"ls"`, `"-l"`, `"/home/brandon"`]
   - Standard output redirected to `some_file`
   - First "normal" token is the command/utility to run

Shell operation

- File redirection
  - `<, >, >>`

- Variable expansion
  - Text substitution before execution
  - `echo $HOME`
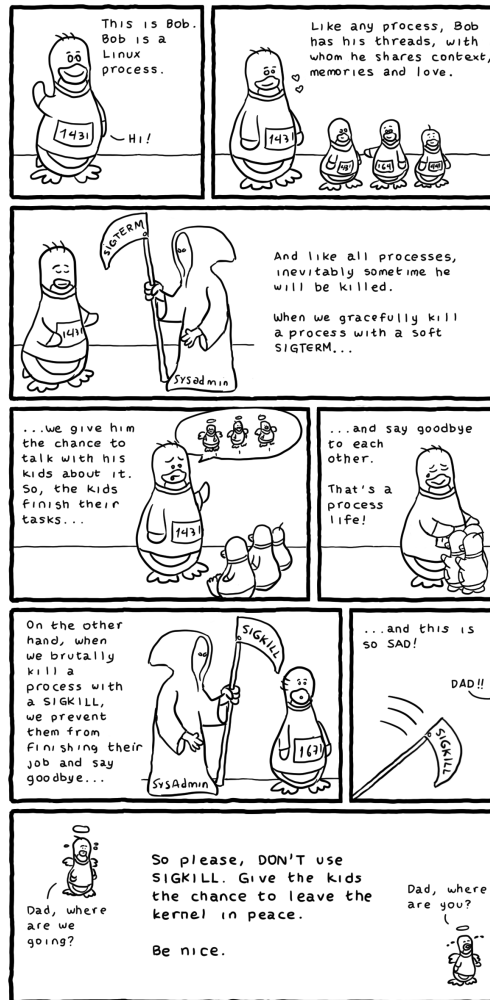 - `$` indicates variable expansion

Shell scripts

- Just files with shell commands in them

- Same stuff that you'd type at terminal

- Shebang to specify interpreter when you execute the file
  - `#!/bin/bash`

  - Applicable to other interpreted languages e.g. Python

- Special

Q&A

Exercises

1. Write a command that saves the output of `ls` to a file `listing`

2. The command `rev` reverses a line of text and `sort` takes lines of input and outputs them in a sorted manner
   - Write a command that takes the output of `ls`, reverses the name of each file, sorts these reversed names and saves it to a file called `gnitsil`

3. Write a command that runs `git status` and saves the standard output to `out.txt` and standard error to `err.txt`

4. The command `date` outputs a timestamp
   - Write a command that appends the current timestamp to a file called `timestamps.log`

Exercises

- Write a shell script that appends an ISO 8601 format timestamp, then appends the first argument to a file named `log`
    - `date -Isec` can get this timestamp for you
    - Make sure to give it a shebang
    - Make sure to `chmod` it so it's executable
    - Run it with an argument e.g. `$ ./myscript this-is-an-argument`

Basic assignment

Addenda