The Shell, Revisited

POSIX

Portable Operating System Interface (X???)

How to interact in a POSIX way

- **Linux**: Already there.
- **Mac OS X**: Already mostly there.
- **Windows**: Need to add a program like...
  - Git Shell
  - Cygwin
  - ... and it is still not quite there, it behaves a little weirdly.

POSIX

- Processes
- Input and Output
  - Files
  - Pipes
  - Network
- Environment Variables
- Time
- Users and Permissions

What is a shell?

It's a program that's designed to let you interact with other programs, environment variables, and the filesystem.

The shell we'll be using is **bash**, the **B**ourne **A**gain **Sh**ell

It's actually kind of a utility knife of programming languages

Starting programs

- Just type the name of the program
- Followed by any arguments

Processes

A running program in POSIX is called a process. Each one has:

- A Process ID (PID)
- A set of input/output handles
  - Standard In
  - Standard Out
  - Standard Error
- An owner
- A parent process

Describing Processes

```
gobo:~ naomi$ ps
  PID TTY           TIME CMD
79157 ttys000    0:00.05 -bash
  840 ttys001    0:01.59 emacs assignment.txt
79169 ttys001    0:00.28 -bash
42262 ttys002    0:00.01 man xargs
42270 ttys002    0:00.04 /usr/bin/less -is
91394 ttys002    0:00.05 -bash
```

For more information try 'ps aux': more columns, and all processes you have permission to see, including ones that have no controlling terminal, and ones that are not yours

Manipulating Processes

```
gobo:~ naomi$ kill 840
gobo:~ naomi$ kill -9 840
gobo:~ naomi$ renice +10 91394
```

Return codes

```
gobo: naomi$ true && echo FOO
FOO
gobo: naomi$ false && echo FOO
gobo: naomi$ true || echo FOO
gobo: naomi$ false || echo FOO
FOO
gobo: naomi$ true && echo A || echo B
A
gobo: naomi$ false && echo A || echo B
B
```

Input and output

- By default, input comes from the keyboard
- By default, output goes to the terminal
- ... but you can change that.

```
gobo:~ naomi$ ls > files.txt
gobo:~ naomi$ cat < files.txt
Desktop
Documents
Downloads
...
```

Pipes

You can also make the output of one program go to the input of the next.

```
gobo:~ naomi$ ls | nl
     1 Desktop
     2 Documents
     3 Downloads
```

Environment Variables

Each process has an environment of some string variables. Some of these have special uses. You get the value of an environment variable by putting a dollar sign in front of it.

```
gobo naomi$ EDITOR=emacs git commit -a
gobo naomi$ export PATH=~/bin:$PATH
```

Users and Permissions

```
naomi$ ls -l
-rw-r--r--  1 naomi  staff  3208 Nov 15 20:16 assignment.txt
-rw-r--r--  1 naomi  staff     0 Nov 15 20:16 fooddatabase.py
-rw-r--r--  1 naomi  staff     0 Nov 15 20:16 guessmynumber.
py
-rw-r--r--  1 naomi  staff     0 Nov 15 20:16 listsorter.py
```

Users and Permissions

```
naomi$ chmod +x myprogram.py
naomi$ chmod 755 myprogram.py

naomi$ chown enne myprogram.py

naomi$ sudo cp myprogram.py
                /usr/local/bin/myprogram
```

Utility programs

- **nl** numbers lines (from stdin or a file)
- **cat** concatenates files to stdout
- **sort** sorts the lines in a file or stdin
- **uniq** de-duplicates adjacent identical lines
- **cut** selects specific fields from stdin to stdout
- **grep** searches the input (file or stdin) for the given regex
- **find** lists all matching files in a directory
- **xargs** takes each line of input, and runs a program on it.
- **head** & **tail** output the top or bottom of a file
- **echo** just prints its argument to stdout