Math 4970: UNIX Basics

Dr. Bill Slough

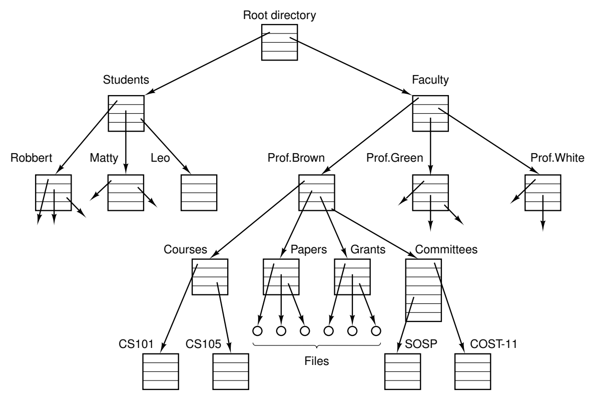
Spring 2008

The UNIX Shell

- The shell is a program — a command-line interpreter.
- The shell acts on commands we enter:
  - prompt for a command line
  - wait for a line of input
  - parse the line
  - create processes, pipes, as needed
  - wait for completion or use background process
- There are many different shells one can select
- A shell is assigned to each user when account is created

```
eiu% grep "William Slough" /etc/passwd
cfwas:x:6338:200:William Slough:/export/home/cfwas:/bin/csh

eiu% echo $SHELL
/bin/csh
```

Files and Directories: Conceptual View



Directories: Listing Contents

- **ls**
- **ls -l**
- **ls -a**
- **ls -l *.tex**
- **man ls**

```
eiu% ls -l *.tex
-rw-r--r--  1 cfwas  faculty  5171 Mar  9  2001 nancy-paper.tex
-rw-r--r--  1 cfwas  faculty 35120 Oct 22  1996 pmgraph.tex
-rw-r--r--  1 cfwas  faculty  4136 Oct 11  1996 programme.tex
```

Directories: Creation and Navigation

- **mkdir math4970**
- **cd ~**
- **cd**
- **cd math4970**
- **pwd**

Files: Creating and Viewing

- **vi README**
- **emacs README &**
- **cat README**
- **cat part1 part2 part3**
- **touch README**
- **more README**
- **less README**

Files and Directories: Removal

- `rm README`
- `mv README ~/.Trash`
- `rm *`
- `rm *.log`
- `rm -rf project3`
- `mv project3 ~/.Trash`

File Permissions

- Three categories: user, group, others → u, g, o
- Each category allows/prevents: read, write, execute

```
-rw-r--r-- 1  cfwas csfac   7059 2005-01-17  13:08 fred.tex
```

- Sample commands to change permissions:
  - `chmod u-r fred.tex`
  - `chmod g+w fred.tex`
  - `chmod o+rw fred.tex`
  - `chmod 644 fred.tex` (octal)

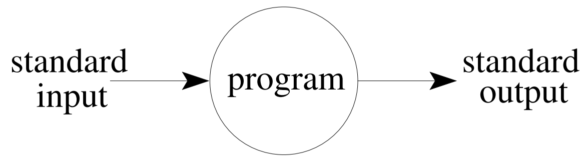
Standard Files

name	file descriptor	C++	redirection
standard input	0	`cin`	<
standard output	1	`cout`	> or >>
standard error	2	`cerr`	2>

Redirection

- `program < myinput > myoutput`
- `cat part1 part2 part3 > result`
- `cat part4 >> result`

Filters: Conceptual



- Read input
- Process the input
- Output the result

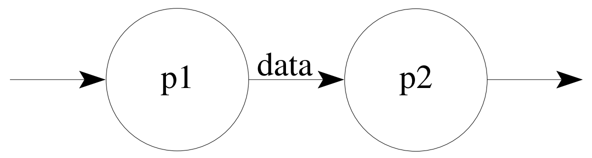
Such a simple idea; yet very powerful

Filters: Examples

- `tr 'a-z' 'A-Z'`
  Input from standard input; result on standard output
- `tr 'a-z' 'A-Z' < myfile`
  Input from `myfile`; result on standard output
- `tr 'a-z' 'A-Z' > myresult`
  Input from standard input; result in `myresult`
- `tr 'a-z' 'A-Z' < myfile > myresult`
  Input from standard input; result in `myresult`

All examples “translate” lower-case to upper-case.

Pipelines: Conceptual



```
p1 | p2
```

Generalizes to more than two processes:

```
p1 | p2 | p3
p1 | p2 | p3 | p4
...
```

Pipelines: Examples with tr

- `cat myfile | tr 'a-z' 'A-Z'`
- `cat myfile | tr ' ' '*' | more`
- `cat myfile | tr -s ' ' '*' | more`

-
  ```
  cat myfile |
  tr -s -c 'a-zA-Z' '*' |
  more
  ```

-
  ```
  cat myfile |
  tr -s -c 'a-zA-Z' '\n' |
  more
  ```

Pipelines: Example - Top Five

```
cat myfile |
tr 'A-Z' 'a-z' |
tr -s -c 'a-z' '\n' |
sort |
uniq -c |
sort -n |
tail -5
```

A few utilities can work together in interesting ways!

Poor Man's Spelling Checker

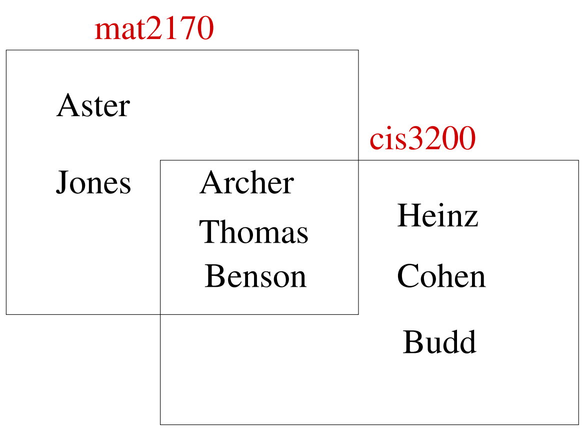
Exercise: Design a pipeline that will spell-check a text file.

- What is a word?
- Compare words against `/usr/share/dict/words`
- What UNIX utilities will be helpful?
- Can it be done with a single pipeline?

Viewing Files as Sets: comm

```
NAME
     comm - compare two sorted files line by line

SYNOPSIS
     comm [OPTION]... LEFT_FILE RIGHT_FILE

DESCRIPTION
     Compare sorted files LEFT_FILE and RIGHT_FILE line by line.

     -1     suppress lines unique to left file

     -2     suppress lines unique to right file

     -3     suppress lines that appear in both files

     --help display this help and exit

     --version
            output version information and exit
```

Viewing Files as Sets



Filters and Pipelines: *Another Example*

How many users are named William?

```
cut -d : -f 5 < /etc/passwd |
grep "^William" |
wc -l
```

One line from /etc/passwd:
cfwas:x:6338:200:William Slough:/export/home/cfwas:/bin/csh

`-d :`	fields are delimited with colon
`-f 5`	extract the fifth field

A Laundry List of Filters

awk	cat	comm	cut
expand	compress	fold	grep
head	nl	pr	sed
sh	sort	split	strings
tail	tee	tr	uniq
wc			

Environment Variables

- Many predefined variables
  `PATH`
  `HOSTNAME`
  `USERNAME`
  ...
- Prepending $ yields the value
  `echo $PATH`
  `/usr/local/bin:/usr/bin:/bin:/usr/bin/X11`
- Changing values and adding new variables
  `PATH=$PATH:/home/cfwas/bin`
  `NAME='William Slough'`
  `course=4970`
- Syntax unusually fussy
  `course = 4970` Illegal!

The Art of Quoting

Single quote	'	Use in pairs; quoted portion treated as one quantity
Double quote	"	Like single quote, but evaluate $
Back quote	`	Perform indicated action
Escape	\	Use only one; single quotes the following symbol convenient way to break multiple lines

Using Quotes: Examples

Which are legal? What will they do?

- `grep William Slough /etc/passwd`
- `grep 'William Slough' /etc/passwd`
- `grep "William Slough" /etc/passwd`
- `grep $NAME /etc/passwd`
- `grep '$NAME' /etc/passwd`
- `grep "$NAME" /etc/passwd`

```
NAME
   grep, egrep, fgrep, rgrep - print lines matching
                               a pattern

SYNOPSIS
   grep [options] PATTERN [FILE...]
   grep [options] [-e PATTERN | -f FILE] [FILE...]
```

Using Quotes: *More Examples*

- `FILENAME=temp-`date +%s``
- `echo Today\'s date is \`
  `` `date | cut -d ' ' -f 2-3` ``
- `echo "4970 : `date` : $USERNAME"`