Lecture 14:
Advanced Unix

November “4”, 2018  Hank Childs, University of Oregon
Return values in shells

C02LN00GFD58:330 hank$ ./a.out copy.c copy2.c
Copying 697 bytes from copy.c to copy2.c
C02LN00GFD58:330 hank$ echo $?
0
C02LN00GFD58:330 hank$ ./a.out copy.c
Usage: ./a.out <file1> <file2>
C02LN00GFD58:330 hank$ echo $?
1

$? is the return value of the last executed command
Printing to terminal and reading from terminal

• In Unix, printing to terminal and reading from terminal is done with file I/O

• Keyboard and screen are files in the file system!
  – (at least they were ...)

Standard Streams

• Wikipedia: “preconnected input and output channels between a computer program and its environment (typically a text terminal) when it begins execution”

• Three standard streams:
  – stdin (standard input)
  – stdout (standard output)
  – stderr (standard error)

What mechanisms in C allow you to access standard streams?
printf

• Print to stdout
  – printf(“hello world\n”);
  – printf(“Integers are like this %d\n”, 6);
  – printf(“Two floats: %f, %f”, 3.5, 7.0);
fprintf

• Just like printf, but to streams
  • fprintf(stdout, “helloworld\n”);
    – → same as printf
  • fprintf(stderr, “helloworld\n”);
    – prints to “standard error”
  • fprintf(f_out, “helloworld\n”);
    – prints to the file pointed to by FILE *f_out.
Streams in Unix
Unix shells allows you to manipulate standard streams.

- “>” redirect output of program to a file
- Example:
  - `ls > output`
  - `echo "this is a file" > output2`
  - `cat file1 file2 > file3`
Unix shells allows you to manipulate standard streams.

• “<” redirect file to input of program
• Example:
  – python < myscript.py
    • Note: python quits when it reads a special character called EOF (End of File)
    • You can type this character by typing Ctrl-D
    • This is why Python quits when you type Ctrl-D
      – (many other programs too)
Unix shells allows you to manipulate standard streams.

- “>>” concatenate output of program to end of existing file
  - (or create file if it doesn’t exist)
- Example:
  - echo “I am starting the file” > file1
  - echo “I am adding to the file” >> file1
  - cat file1
    - I am starting the file
    - I am adding to the file
What’s happening here?

C02LN00GFD58:330 hank$ mkdir tmp
C02LN00GFD58:330 hank$ cd tmp
C02LN00GFD58:tmp hank$ touch f1
C02LN00GFD58:tmp hank$ ls f1 f2 > out
ls: f2: No such file or directory
C02LN00GFD58:tmp hank$ cat out
f1

ls is outputting its error messages to stderr
Redirecting stderr in a shell

C02LN00GFD58:Documents hank$ cd ~/330
C02LN00GFD58:330 hank$ mkdir tmp
C02LN00GFD58:330 hank$ cd tmp
C02LN00GFD58:tmp hank$ touch f1
C02LN00GFD58:tmp hank$ ls f1 f2 > out
  ls: f2: No such file or directory
C02LN00GFD58:tmp hank$ cat out
  f1
C02LN00GFD58:tmp hank$ ls f1 f2 > out 2>out_error
C02LN00GFD58:tmp hank$ cat out_error
  ls: f2: No such file or directory
Redirecting stderr to stdout

Convenient when you want both to go to the same stream

```bash
C02LN00GFD58:330 hank$ mkdir tmp
C02LN00GFD58:330 hank$ cd tmp
C02LN00GFD58:tmp hank$ touch f1
C02LN00GFD58:tmp hank$ ls f1 f2 > out
ls: f2: No such file or directory
C02LN00GFD58:tmp hank$ cat out f1
C02LN00GFD58:tmp hank$ ls f1 f2 > out 2>out_error
C02LN00GFD58:tmp hank$ cat out_error
ls: f2: No such file or directory
C02LN00GFD58:tmp hank$ ls f1 f2 > out 2>&1
C02LN00GFD58:tmp hank$ cat out
ls: f2: No such file or directory
f1
```
c functions: fork and pipe

• fork: duplicates current program into a separate instance
  – Two running programs!
  – Only differentiated by return value of fork (which is original and which is new)

• pipe: mechanism for connecting file descriptors between two forked programs

Through fork and pipe, you can connect two running programs. One writes to a file descriptor, and the other reads the output from its file descriptor

Only used on special occasions. (And one of those occasions is with the shell.)
pipes in Unix shells

- represented with “|”
- output of one program becomes input to another program
Very useful programs

- grep: keep lines that match pattern, discard lines that don’t match pattern

```bash
C02LN00GFD58:Documents hank$ ls -l | grep ppt
-rw-r--r--@ 1 hank staff 3278589 Apr 5 11:40 CIS330_Lec2.pptx
-rw-r--r--@ 1 hank staff 2220104 Apr 8 20:57 CIS330_Lec3.pptx
-rw-r--r--@ 1 hank staff 3899863 Jan 21 09:26 CIS610_lec2.pptx
-rw-r--r--@ 1 hank staff 4629257 Jan 30 10:24 CIS610_lec3.pptx
-rw-r--r--@ 1 hank staff 21382185 Mar 25 12:40 CIS_colloquium2013.pptx
-rw-r--r--@ 1 hank staff 21382185 Jan 7 12:21 CIS_colloquium_2013.pptx
-rw-r--r--@ 1 hank staff 2172179 Dec 20 15:24 ICS_results.pptx
-rw-r--r--@ 1 hank staff 4841050 Nov 13 10:10 MBTI.pptx
-rw-r--r--@ 1 hank staff 2031749 Apr 5 16:20 SC14_flow.pptx
-rw-r--r--@ 1 hank staff 17972476 Mar 25 12:43 VMV_2013.pptx
-rw-r--r--@ 1 hank staff 98149068 Apr 1 10:25 aachen.pptx
-rw-r--r--@ 1 hank staff 9815146 Feb 24 07:00 childs_poster_SDAV_AHM_2014.pptx
-rw-r--r--@ 1 hank staff 592243 Feb 26 04:09 childs_sda slides.pptx
-rw-r--r--@ 1 hank staff 15765504 Feb 13 14:57 cig_exascale.ppt
-rw-r--r--@ 1 hank staff 16699392 Jan 7 12:14 cis610_Lec1.ppt
-rw-r--r--@ 1 hank staff 3159872 Jan 7 11:15 egpgv_cfg.pptx
-rw-r--r--@ 1 hank staff 15767552 Mar 23 02:48 eu_regional_school.ppt
-rw-r--r--@ 1 hank staff 35099136 Mar 25 09:42 eu_regional_school_part1.ppt
-rw-r--r--@ 1 hank staff 10775552 Mar 25 04:49 eu_regional_school_part1B.ppt
-rw-r--r--@ 1 hank staff 72966144 Mar 26 08:43 eu_regional_school_part2.ppt
-rw-r--r--@ 1 hank staff 7571317 Mar 25 12:53 ilm_booth_talk.pptx
```
Very useful programs

- **sed**: replace pattern 1 with pattern 2
  - `sed s/pattern1/pattern2/g`
    - `s` means substitute
    - `g` means “global” ... every instance on the line

- `sed` is also available in “vi”
  - `:%s/pattern1/pattern2/g` (% means all lines)
  - `:103,133s/p1/p2/g` (lines 103-133)
Wildcards

• ‘*’ is a wildcard with unix shells

```
fawcett:tmp child$s$ ls
Abe	Chavarria	Hebb	Macy	Smith
Alajaji	Chen	Jia	Maguire	Steelhammer
Alamoudi	Clark	Kine	Michlanski	Szczepanski
Anastas	Collier	Lee	Moreno	Totten
Andrade	Costello	Legge	Olson	Vega-Fujioka
Ballarche	Donnelly	Li	Owen	Wang
Brennan	Ezel	Lin	Pogrebinsky	Whiteley
Brockway	Friedrich	Liu	Qin	Woodruff
Brogan	Garvin	Lopes	Rhodes	Xu
Brooks	Gonzales	Luo	Roberts	Yaconelli
Bruce	Guo	Lynch	Rodriguez	Young
Carlton	Hampton	Lyon	Roush	Zhang
Chalmers	Harris	Machado	Rozenboim
def
```

`fawcett:tmp child$s$ ls C*`

Carleton	Chavarria	Clark	Costello
Chalmers	Chen	Collier

```
fawcett:tmp child$s$ ls *z
Rodriguez
```

```
fawcett:tmp child$s$ ls *ee*
Lee	Steelhammer
```

```
fawcett:tmp child$s$ ls *e*e*
Lee	Legge	Steelhammer	Whiteley
```

‘?’ is a wildcard that matches exactly one character
Other useful shell things

- ‘tab’: auto-complete
- esc=: show options for auto-complete
- Ctrl-A: go to beginning of line
- Ctrl-E: go to end of line
- Ctrl-R: search through history for command