

CS 537

Section 1: The Shell Project

Michael Swift

© 2005-2006, Stanford University, All rights reserved.
http://cs537.svn.sourceforge.net/viewvc.cgi

1

Project Goal

- Get used to programming in C and Unix
- Learn how the Unix shell works
- Understand process control functions
 - create
 - wait
- Understand string processing
 - strtok()

© 2005-2006, Stanford University, All rights reserved.
http://cs537.svn.sourceforge.net/viewvc.cgi

2

What is a shell?

- **Command line interpreter**
 - You type "ls /etc"
 - The shell invokes the first parameter as a command, with the remainder as the parameters
 - eg: exec(0, "ls", "/etc")
- **Built-in commands**
 - Most commands are separate executable programs
 - ls, mv, mv, make, gcc
 - Some commands are interpreted by the shell
 - cd, exit
- **In your shell, the only built-in command is "quit"**

© 2005-2006, Stanford University, All rights reserved.
http://cs537.svn.sourceforge.net/viewvc.cgi

3

Interactive vs Batch

- **Interactive**
 - User types commands in, hits return to invoke them
- **Batch**
 - shell reads from an input file
- **What is the difference?**
 - where the commands come from
- **How do you code this?**
 - Change which file you read from

© 2005-2006, Stanford University, All rights reserved.
http://cs537.svn.sourceforge.net/viewvc.cgi

4

File I/O in C

- Standard commands for reading from a file:
 - `struct FILE * f = fopen(filename, mode)`
 - mode = "r", "w" plus some others
 - `size_t bytes = fread(buffer, rec_size, rec_num, f)`
 - read `rec_size * rec_num` bytes from `f`
 - returns number of bytes read
 - returns zero if no data left
 - ignores line numbers
- `char * s = fgets(string_buf, string_len, f)`
 - reads up to `string_len-1` characters
 - null terminated string
 - include newline
 - return NULL if end-of-file

© 2005 Pearson Education, Inc. All rights reserved.
http://www.pearson.com

3

End of file

- What happens if the input file ends without a "quit"?
 - You must detect it
 - This can occur if a user enters CTRL-D
- How?
 - Error return from input routines
 - `feof()` tells you explicitly

© 2005 Pearson Education, Inc. All rights reserved.
http://www.pearson.com

4

Printing errors

- C has 3 standard files prepared for you
 - `stdin` = input
 - `stdout` = output
 - `stderr` = error output
- `printf("foo") == fprintf(stdout, "foo")`
- `scanf("%s", str) == fscanf(stdin, "%s", str)`
- `fprintf(stderr, "Panic!")` prints an error message separately
 - Why?

© 2005 Pearson Education, Inc. All rights reserved.
http://www.pearson.com

5

Process Control

- Your shell should execute the next command line after the previous one terminates
 - you must wait for any programs that you launch to finish
- You can launch multiple simultaneous commands with ";" separating them (not needed at the end)
 - `ls -l ; cat file`
 - You need to wait for `ls` and `cat` to finish here

© 2005 Pearson Education, Inc. All rights reserved.
http://www.pearson.com

6

Hints

- A shell is a loop
 - read input
 - execute program
 - wait program
 - repeat
- Useful routines
 - `isdigit()` for string parsing
 - `strchr()` for parsing
- Executing commands
 - `fork()` creates a new process
 - `execvp()` runs a new program and does path processing
 - `wait()`, `waitpid()` waits for a child process to terminate

© 2005 Pearson Education, Inc. All rights reserved.
http://www.pearson.com

9

Managing your code

- Be sure to keep old versions in case you delete something useful
 - Easy technique: a backup (or more) directories
 - Better technique: revision control
 - can concurrent version control
 - stores history of programs
 - can commit changes when you have working code, or as a checkpoint of your work.

© 2005 Pearson Education, Inc. All rights reserved.
http://www.pearson.com

10