

CS 537 Section 2: The Shell Project

Michael Swift

© 2005 Michael Swift. All rights reserved.
http://www.cs.cmu.edu/~mswift

1

Questions on Homework 1

- You can skip 1.8 about the "device-status table"

© 2005 Michael Swift. All rights reserved.
http://www.cs.cmu.edu/~mswift

2

Questions on Project 0

- Due 9 pm tonight in the "handin" directory
- Any problems?

© 2005 Michael Swift. All rights reserved.
http://www.cs.cmu.edu/~mswift

3

Project 1 Goal

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

© 2005 Michael Swift. All rights reserved.
http://www.cs.cmu.edu/~mswift

4

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("ls", "/etc")
- Built-in commands
 - Most commands are separate executable programs
 - ls, rm, mv, make, gcc
 - Some commands are interpreted by the shell
 - cd, exit
- In your shell, the built-in commands are "exit", "echo", "pwd", and "cd"

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

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

4

Process Control

- Your shell should execute the next command line after the previous one terminates (unless ...)
 - 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
- Sequences of commands ending in "&" can launch in parallel:
 - echo "this" ; echo "that" & echo "hello" ; echo "goodbye"
 - hello / goodbye don't wait for (this / that)

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

5

Hints

- A shell is a loop
 - read input
 - execute program
 - wait program
 - repeat
- Useful routines
 - fgets() for string parsing
 - strtok() 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

6

parsing with strtok

```
#include <string.h>
char * strtok(char str, char * sep);
```

- The `strtok()` function tokenizes a string into words
 - `str` is the string to tokenize
 - `sep` are the characters that separate tokens, e.g., space, tab, new line
 - `strtok` remembers the string after the first call

- **Example:**

```
tmp = strtok(buffer, " \t\n");
while (tmp) {
    cmd[num_cmds] = tmp;
    num_cmds++;
    tmp = strtok(NULL, " \t\n");
}
```

© 2005 Pearson Education, Inc. All rights reserved. <http://www.pearsoned.com>

9

strtok Things to Remember

- `strtok()` modifies the string
 - It replaces the separator with a null character
 - `strtok()` returns `NULL` when you get the last token of a string
 - As long as the buffer you parse remains allocated, you can store the pointers returned from `strtok()`
- You can use `strtok()` again on the strings returned from `strtok` to parse with different separators
 - e.g., separate a string into commands, and then a command into arguments

© 2005 Pearson Education, Inc. All rights reserved. <http://www.pearsoned.com>

10

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
 - auto 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.pearsoned.com>

11