CS 537
Section 2: The Shell Project

Michael Swift

Questions on Project 0

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

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()

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”

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

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
- A commands ending in '&' can launch in parallel:
  - echo "that" & echo "hello" &
  - (that and hello can print in either order)
- The shell does not wait for commands ending in '&'
  - echo "that" & echo "hello"
  - shell waits for "hello" but not "that"

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

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 strong after the first call
- Example:

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

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 strong into commands, and then a command into arguments

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
    - cvs: concurrent version control
    - stores history of programs
    - can commit changes when you have working code, or as a checkpoint of your work.