Programming Project #1
Command Shell

CS-3013 Operating Systems
Hugh C. Lauer

(Slides include materials from
Modern Operating Systems, 3rd ed., by Andrew Tanenbaum
and from Operating System Concepts, 7th ed., by Silbershatz, Galvin, & Gagne)
Purpose

- To practice using the `fork` and `exec` facilities of Unix/Linux
- To learn how to find and use on-line documentation about system calls and system facilities.
  - `man` pages
- To become accustomed to speaking directly with the operating system
Programming Assignment

Build a command shell capable of running commands in the background:—

1. Write a command called `doit` to execute another command and print out statistics

2. Adapt `doit` to become a basic `shell` that prompts for and executes commands

3. Modify your `shell` to become `shell2` to execute commands in the background
Part 1: doit

- **doit** command:–
  - Take another command line as argument
  - Execute that command
  - Print out statistics about command
- Must compile and execute on your virtual machine
  - Likely to be compatible with most modern Linux systems
- Useful system calls and functions:–
  - `fork()`, `execvp()`, `wait()`
  - `getrusage()`, `gettimeofday()`
  - May not use `system()` system call

Use `execvp()`, not `execve()`
Part 1 (sample output)

% ./doit cat /etc/motd
/* print the current message of the day */
/* statistics about the cat command */

% ./doit ls -l /usr/src
/* list contents of the /usr/src directory */
/* statistics of the ls command */

% ./doit cp -al /usr/src/linux-2.6.27.25-0.1 kernelSrc
/* output of the cp command (if any) */
/* statistics of the cp command */
doit Implementation Hint

- Command line is already tokenized in `argv`
- Strip off the $0^{th}$ element of `argv` vector
- Pass the remainder of `argv` to `exec()` for arguments of command
Part 2: shell

- Repeatedly prompt for command lines
- Execute each command and print out statistics, as in `doit`
- Implement special “built-in” commands
  - `exit`
  - `cd` `dir`
- Special conditions
  - Input from `stdin`
  - Exit on end of file
  - Complain about illegal commands
  - Check for line length, number of arguments
% shell
==>cat /etc/motd
    /* print the current message of the day */
    /* statistics about the cat command */
==>cd dir
    /* current directory is changed to dir */
==>ls -l /boot
    /* listing of files in /boot directory */
    /* statistics about this ls command */
==>exit
% /* back to your previous Linux prompt */

% shell < filename.txt
    /* reads, echoes, and executes each command*/
    /* statistics about each command */
    /* exits after last command is completed */
Implementation Hints

• Input command line
  • May limit to 128 characters, 32 arguments
  • Must tokenize — use `strtok()` function

• Built-in commands are *not* handled by `fork` and `exec`
  • Must be executed “in-line” in shell

• `getrusage()` returns cumulative statistics
  • For all children of this process
  • Must remember and subtract previous values to get statistics for most process most recently waited for
Part 3: Background Execution (shell2)

- Modify **shell** to allow commands to execute in background, indicated by ‘&’ character
  - Causes shell to prompt for next command before previous command is complete
  - **jobs** built-in command lists background tasks in progress
- Modified **wait()** call
  - Returns immediately if nothing has terminated
  - Returns PID if any process has terminated
- Report completion of command at next opportunity
  - *i.e.*, the next time user types a command, or whenever **wait()** returns PID of the command
Part 3 output

% shell2
==>make -j4 O=~/kernelDst &
[1] 12345 /* indicate background task #1, process id */
==>jobs
[1] 12345 make
    /* print process id and command name for tasks */
==>ls
    /* listing of files in the current directory */
    /* statistics about this ls command */
==>cat /etc/motd
[1] 12345 make Completed
    /* statistics about background job */
    /* print the current message of the day */
    /* statistics about this cat command */
==>exit
% /* back to your previous Linux prompt */
shell2 Implementation Hints

• Use `wait3()` to
  • Wait for any process
  • Wait without blocking – `WNOHANG` option
  • Get usage information directly

• After `fork()`, `shell2` does
  
  ```c
  while (pid = wait3(WNOHANG)) {
      print msg & statistics of pid
      update list of outstanding tasks
  }
  if (latest not background && latest not done)
      wait3(latest_pid);
  else get next command;
  ```
Work environment

- Program in C
  - Even though it could easily be done in C++
- You may work on any Linux system
  - E.g., your own or corporate environment
- We will grade it on openSUSE 11.4
  - I.e., your virtual machine environment
- Note:
  - Linux systems are often compatible with each other
  - However, not all C/C++ compilers and Linux kernels are alike
  - Some may produce or ignore compiler warnings
  - Your virtual machine is the one that counts
Project submission

- Due Tuesday, Sept 6 @ 11:59 PM
- Submit via web-based *Turnin*
  - Project name = *Project 1*
- *Put your name at top of every file!*
  (You would be surprised how many students forget!)
Project Submission (continued)

- Code of your solution
- A single Makefile
  - make clean
  - make all
  - make doit
  - make shell
  - make shell2
- Test cases for each part
- Write-up explaining everything
  - Especially how to keep track of outstanding processes in shell2
- Zip everything together
- **DO NOT**
  - Put different parts of project in separate folders
  - Use different makefiles for different parts of project
Grading

- Ten point project
  - Three points for each part
- Point allocation per part
  - Correct compilation and build; no warnings! – 1 pt.
  - Correct execution of graders’ test cases – 1 pt.
  - Submission and correct execution of own test cases – 1 pt
- Write-up explaining what you did and your results
  - 1 pt
  - MS Word or PDF
- No credit for a part if it fails to build
Not a team project!

- Each student must submit own solution
- Encouraged to discuss general approach, meaning of system calls, etc., with friends
- Encouraged to seek help if puzzled by some concept, system call, etc.
- Distill all discussions into your own work!
Questions?