Lab Assignment 3 - Implementing Your Own `system` Function

EECS-213 - Spring 2007

Important Dates

Out: May 23th, 2007.

Due: June 1st, 11:59PM.

John Otto is the lead person and bomb squad chief for this lab.

Introduction

This assignment helps you develop a basic understanding of process creation and management in Unix systems. When you want to execute a program from within a C application, you typically use the `system` function. The system() function causes string to be given to the shell as input, as if string had been typed as a command at a terminal.

Your task is to write your own version of the Unix `system` function:

```
int mysystem(char *command);
```

The `mysystem` function executes `command` by calling `"/bin/sh -c command"`, and then returns after `command` exits normally (by calling the `exit` function or executing a `return` statement), then `mysystem` returns the `command` exit status. For example, if `command` terminates by calling `exit(8)`, then `system` returns the value 8. Otherwise, if `command` terminates abnormally, them `mysystem` returns the status returned by the shell.

Logistics

You must work on your own when solving the problems for this assignment. Any clarifications and revisions to the assignment will be posted on the course Web page.

Hand Out Instructions

Please download `mysystem-handout.tar` from the course website.

Start by copying `mysystem.tar` to a (protected) directory in which you plan to do your work. Then give the command "`tar xvf mysystem.tar`". This will cause a number of files to be unpacked in the skeleton directory:

`Makefile`: Compiles your program.

`mysystem-test.c`: Contains the `main` function. You need to modify this function to test your code.

`mysystem.c`: Your `mysystem` function, which you need to implement.

`extern.h`: Header file for the `mysystem` function.

Your assignment must be compiled on a Linux machine.

In the following instructions, we will assume that you have copied the three programs to a protected local directory, and that you are executing them in that local directory.

The `mysystem` Function

The implementation requires you to master the `fork` and `exec` system calls (and/or their variants), make the parent `wait` until the child returns and to check for return codes from the child process. Check the corresponding man pages (or Google) for usage and examples.

Deliverables

Use the class submission page to submit *only* your `mysystem.c` file. We will use our own main function to test your implementation.