EECS 111 - System Software

Spring 2010

Assignment 1

Posted: Monday, April 5, 2010

Due: Tuesday, April 13, 2010, 12pm (noon)

A. Discussion: OS Concepts, I/O Methods, System Call

The goal of these exercises is to review and clarify the understanding of essential aspects covered in recent lectures.

- Operating System Concepts: Exercise 1.1
- Input/Output Methods: Exercises 1.22 and 1.23
- System Calls: Exercises 2.1 and 2.13

These topics will be discussed at the beginning of this week's discussion session.

B. Project: C Programming Environment, Processes

The goal of this first project assignment is to setup a working C programming environment on the Solaris operating system (EECS servers) that we will be using for this and the following assignments. At the same time, we will review and confirm some basic understanding of running processes.

Step 1: Setup your C programming environment

- We will use the EECS department servers as work platform, namely **`malibu.eecs.uci.edu`** (or, as alternatives, **`vivian.eecs.uci.edu`** and **`newport.eecs.uci.edu`**). If you do not have an account on these machines, send an email to the course instructor and an account will be created for you.
- To work on the server, you will need to connect to one of the machines through a remote shell client. The secure-shell (SSH) protocol is necessary for this. Secure shell clients for all major operating systems exist for free, please refer to the resources page on the course web pages for details: http://eee.uci.edu/10s/18050/resources.html.
- Hint: If you are not really familiar with C programming in the EECS environment, you may want to review the lecture slides of the introductory course on C programming: http://eee.uci.edu/09f/18010/schedule.html.

Create a new directory for this class (e.g. "**eecs111**") and separate subdirectories for this and each following assignment (e.g. "**eecs111/hw1/**"). Place all files associated with an assignment in the corresponding directory. Ensure that the directories and all files have proper access permissions (e.g. 640 for source files, 750 for directories and executables).

Step 2: Create, compile, and test a simple C program

As an initial starting point, create, compile and test (and debug, if necessary!) a simple C program that computes the n-th Fibonacci-number by use of *recursion*.

Your source file should be named "**fibo.c**" and the corresponding executable should be called "**fibo**". The program should read its input, an integer **n**, directly from the command line and print all reports to the **stdout** stream. Make sure your code includes proper error checking, reporting, and handling.

Your compilation and execution log should look as follows:

```
malibu.eecs.uci.edu % vi fibo.c
malibu.eecs.uci.edu % gcc -O2 -Wall fibo.c -o fibo
malibu.eecs.uci.edu % fibo 10
fibo: computing fibonacci(10)...
fibo: fibonacci(10) = 55
malibu.eecs.uci.edu %
```

Step 3: Measure the execution time of your implementation

The Solaris operating system that we are using includes accounting support for measuring CPU time. We will use the system program **/usr/bin/time** to print the execution time of our programs. Lookup the Solaris manual page for the **time** command (it is in section 1 of the manual) and use it to determine the system, user, and elapsed execution times for computing the 40^{th}, 41^{st}, 42^{nd}, and 43^{rd} Fibonacci numbers with your **fibo** program.

List your measured times in a table (note the server name(s) you are using) and briefly explain why the times come out this way. Do the times match your expectation? Why?

Step 4: Investigate the memory layout of a process

In the text book, Figure 3.1 (page 102) shows the conceptual organization of a process in main memory. Extend (instrument) your Fibonacci program (name this one **fibo_mem.c**) so that it prints the memory location of the functions and variables in different memory segments. Remember, you can use the address-of operator (**&**) to determine the (virtual) addresses of functions and variables. Print these addresses as 8-digit hexadecimal numbers for easier interpretation.

Your execution log should look similar to this (with **X**'s replaced):

```
malibu.eecs.uci.edu % fibo_mem 2
fibo_mem: computing fibonacci(2)...
fibo_mem: address of XXXXXX is hex XXXXXXXX (in text segment)
fibo_mem: address of XXXXXX is hex XXXXXXXX (in data segment)
fibo_mem: address of XXXXXX is hex XXXXXXXX (in heap segment)
fibo_mem: address of XXXXXX is hex XXXXXXXX (in stack segment)
...
fibo_mem: fibonacci(2) = 1
malibu.eecs.uci.edu %
```

Compare your addresses with Figure 3.1 and briefly explain whether or not they fit the expected memory layout.

Deliverables:

1. Statement: "I have read the Section on Academic Honesty in the UCI Catalogue of Classes (available online at http://www.editor.uci.edu/catalogue/appx/appx.2.htm#gen0) and submit this work accordingly."
2. Source file: **`fibo.c`**, execution log **`fibo.log`**, table with execution times, and brief interpretation of the results (1-2 paragraphs) [50 points].
3. Source file: **`fibo_mem.c`**, execution log **`fibo_mem.log`**, and brief interpretation of the results (1 paragraph) [50 points].

Submission instructions:

To submit your homework, send the deliverables in an email with subject "EECS111 HW1" to the course instructor at doemer@uci.edu.

To ensure proper credit, be sure to send your email before the deadline: Tuesday, April 13, 2010, at 12:00pm (noon).

--

Rainer Doemer (EH 3217, x4-9007, doemer@uci.edu)