Systems Programming 1 – 300167

Tutorial and Lab Practice Seven (follows Lecture Seven).

This work **will** be marked and is due in your scheduled lab class the week commencing Monday 2nd of May, and is worth **4%** of your assessment. You should hand in a hard copy of your answers and demonstrate your programs to you Tutor during your scheduled lab session.
Email submission is not accepted.

Tutorial

1. Read Chapter seven of the textbook.
2. Review the terminology introduced and concepts taught in Lecture Seven.
3. *pause* waits for any signal to arrive, including signals generated from the keyboard, such as *SIGINT*.
   (a) Run *sleep1* and press *Ctrl-C*. What happens? Why?
   (b) Modify *sleep1.c* to handle *SIGINT*.
   (c) Now run the program again and press *Ctrl-C*. What happens? Why?
4. Review `sleep()`, `alarm()` and `pause()` functions.
5. What is a signal? How is it handled? Where do signals come from? How multiple signals are handled? (**1%**)
6. How does a process send a signal to another process? (**1%**)

Lab Practice

1. Modify *hello1.c* to display a blinking message in the center of the screen. If the user supplies a message on the command line, your program should display that message, otherwise the default message is displayed. Use the *sleep* function to pause the program between printing the message and then erasing it.
2. Modify *sigdemo3.c* by combing the two signal handlers into one signal handler that checks its argument to determine which signal invoked the handler. (**2%**)

Optional Work **(For those students who wish to practise more.)**

1. Write a program that measures how quickly a user can response. The program waits for a random interval of time and then prints a single digit on the screen. The user has to type that digit as quickly as possible. The program records how long it takes the user to respond. You program should perform 10 such tests and report the minimum, maximum and average response time. (refer to the manual page for gettimeofday).

2. *sigactdomo.c* does not reset *Ctrl-C* handler, two *Ctrl-C* will kill the program. Modify it so it can handle multiple *Ctrl-C*s.