COMP 3080 Operating Systems - Assignment #1
Thursday September 15, 2016

1. This assignment must be submitted electronically and is due no later than **midnight** (11:59:59 PM) of **Thursday September 15.**

2. **All** of your submissions must include a minimum of **four** separate files:
   - **File 1:** A short **write-up** that first specifies what you think your **degree of success** with a project is (**from 0% to 100%**), followed by a brief discussion of your approach to the project along with a **detailed description** of any problems that you were **not** able to resolve for this project. **Failure to specifically provide this information will result in a 0 grade** on your assignment. If you do **not disclose** problems in your write-up and problems are detected when your program is tested, you will receive a grade of 0.
   - **File(s) 2(a, b, c, …)**: Your **complete source code**, in one or more **.c** and/or **.h** files
   - **File 3**: A **make file** to build your assignment. This file must be named **Makefile**.
   - **File 4:** A file that includes your **resulting output** run(s) from your project. This is a simple text file that shows your output, but make sure that you annotate it so that it is self descriptive and that all detailed output is well identified.

3. The files described above should be the only files placed in one of your subdirectories, and this subdirectory should be the target of your **submit** command (see the on-line file **Assignment_Submit_Details.pdf** for specific directions … this file will be posted when the class Teaching Assistant (TA) corrector is determined).

4. The problem you must solve as described in class, is formalized as follows:

This problem will require you to write a program on a UNIX platform (most any will do, including LINUX systems) which will:

- report a collection of process and thread information about itself to standard-out as described below
- create a unnamed pipe
- spawn a child process which will inherit access to the unnamed pipe
- read from the input channel of the pipe (waiting for the child to indicate that it has entered an endless loop of computation)
- send the child a SIGTERM signal to break it out of its endless loop
- wait for the child to terminate and report the child's exit status

- finally, complete its own execution

The child process that you spawn from you program will perform the following tasks:

- report a collection of process and thread information about itself to standard-out as described below (same as parent)
- set up a signal handler to handle the **SIGTERM** signal the parent will send
    - the handler must force the child to load a new program (this program is owned by me and will adjust its priority when it runs)
- write a message into the pipe inherited from its parent (any message will do, this is just a synchronization step to allow the child to get established before the parent tries to send it a signal)
- enter an endless loop in its code path (what you do here is not important)
- when the child enters its signal handler, it must use a member of the **exec** system call family to start a new program running in its process
- the new executable program (which is my program) can be found on any of our systems (like mercury) at **~bill/cs308/Assign1**, will then report a collection of process and thread information to standard-out in the same format previously used (and described below)

The reports generated from all three sources should look like this:

- Process ID is: PID
- Process parent ID is: PPID
- Real UID is: RUID
- Real GID is: RGID
- Effective UID is: EUID
- Effective GID is: EGID
- Process initial thread priority: IT_PRIOR

All of your output must be prefixed by either PARENT: ……. or CHILD:…… , so that it can be readily identified as to its source. The output from my program will also be prefixed uniquely. The source code for my program (which contains much of the functionality you'll need for your program) can be found at **~bill/cs308/Assign1.c** and also on-line at: www.cs.uml.edu/~bill/cs308/Assign1.c

An assignment help file can also be found at: www.cs.uml.edu/~bill/cs308/assignment_1_help.pdf