91.308 Operating Systems, Fall 2005
Assignment 1

Out: September 9, 2005
Due: September 16, 2005

1. The due date for this assignment is **Friday, September 16**.

2. Your submission must be made in class on the due date, and must have the course cover-sheet as the first page. The cover sheet is available on-line at the course home page, in the **Resources** menu at the top of the page.

3. You must submit **hardcopy** of your **source code**, **your program's output** and a **write-up** (including a **flow chart**) of your results as described below.

4. Indicate **clearly** on the course cover-sheet, how much of this assignment you feel you were able to do correctly, and what, if anything, you were not able to do. **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.

5. Your submitted code may be compiled and executed for testing purposes, so make sure you **include the path names to your files** and any build information (i.e. a makefile) if you are using multiple source files, in the space provided on **the course cover-sheet**. You should mark your **homework directories with drwx--x--x** which allows you full access but only allows anyone else with the ability to use your directory in a path name and **not the ability to do an "ls"** in such a directory. This way no one can see what's in your directory—including me—so it's critical for you to **give me the names of the objects** I need to test your project on the cover-sheet, and to make sure those objects have **-rw-r--r--** permissions, which allows them to be read or copied by anyone who knows their exact names.

6. The problem has been described in class and is formalized as:

This problem will require you to write a program on a UNIX platform that will:

- report a collection of process and thread information about itself to standard-out as described below
- create an 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
- finish

The child process that you spawn from your program must 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 tell the child to load a new program (this program is owned by me and has its SUID bit turned on)
- 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
- this new executable program (the "Professor program," which can be found on our PC-Linux servers such as `mercury` at `/usr/cs/fac4/fredm/www/courses/91.308-fall05/files/Assign1`) will then report a collection of process and thread information to standard-out in the same format previously used (and described below)

The `Assign1` program is compiled for x86-Linux, so in order for this last step to work, you must be running on one of our PC-based machines. So I recommend you do your work on `mercury`.

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 on-line at:

`http://www.cs.uml.edu/~fredm/courses/91.308-fall05/files/Assign1.c`

7. For this project, you should include a write-up of a paragraph or two describing your basic design and any problems or general observations you had during the development of your solution.

8. Also, in the write-up, include a visual flowchart **that illustrates flow of control because your program and the "professor program."** The flowchart should indicate which section or function of each program is active, how this activity corresponds (in time) between your program and the "professor program," and what events are causing changes to occur.

Credits: This assignment was developed by Prof. Moloney, UML CS Dept.