**COP 4600-001 Operating System**

**Project #1 (Shared Memory)**

**January 22, 2014**

**Project Objectives:**

The purpose of this project is to introduce students to the concept of shared memory and the problems that can occur if shared memory is not protected adequately.

**Total points Available:** 100

**Due:** February 1st, 2015, 11:59 pm

**Project Description:**

In this assignment, you are to modify the base program ass1.c (Canvas under “Assignments” section) to create **3 processes**. Each of these processes will share a variable called "total". Each will increment the variable “total” by one to 100,000, 200,000 and 300,000 respectively. Make sure that only the newly created child calls the function “process#()”

After **all the children have finished**, the parent process should release the shared memory and terminate. Use the **"wait"** function so that the parent knows precisely when each of the children finishes. The parent should print the process id of each child as the child finishes execution. Then it should release shared memory and print "End of Program".

You need to run program several times and analyze your observations (write report).

**Submitting your assignment**

* Submission via Canvas Assignment.
  + It is your responsibility to submit these assignments in a timely fashion.
* All files should be zipped together.
* There should be a readme file explaining in detail the exact steps to be taken to compile and execute the code files and the title page
* Testing of this work should be done only on the CS lab machines. Please make sure these machines are not locked up due to your code. The execution for grading purposes will be done on the lab machines.
* In case of any code errors, partial credit may be offered based on the code and documentation.
* A report that presents the performance evaluation of your solution.
  + The report should be properly formatted (an academic format style, such as ACM or IEEE being preferred) and contain quantitative data along with you analysis of these data.

**Late Submission Policy**

* Late work will be not accepted.

**Grading Criteria:**

* Minus 90% if code does not compile. Minus 70% if it compiles but does not run.
* If the code compiles and runs, further deductions will be made for the following:
  + Minus 40% if 3 children are not created.
  + Minus 30% if the children fail to modify the shared variable.
  + Minus 20% if parent ends without waiting for all children to exit.
  + Minus 10% if parent does not release shared memory before ending.
  + Minus 10% if the report is not written
  + Minus 10% if children do not print out their results.
  + Minus 10% if parent does not print each time a child finishes.
  + Minus 5% if no comments.
  + Minus 3% if your name and username is not included in comments on the top of your source code

**Development Environment**

You may write your program using any available editor Nano, Pico, Emacs, Vi or whatever editor you are most comfortable with, BUT, it must compile with gcc and be executable on one of the CS machines:

PLEASE DO NOT USE ANY OTHER CSE MACHINE FOR THIS PROJECT!

To login to these machines remotely, download PUTTY (for Windows, Linux users skip this step) by going to: http://the.earth.li/~sgtatham/putty/latest/x86/putty-0.58-installer.exe

Then after the download, execute PUTTY and enter one of the lab machines for the Host Name. Then enter your password for your UNIX account (see your instructor if you do not have a UNIX account or aren't sure if you have one).

**Hints:**

Build your project in an incremental fashion. Attempt to meet each objective before moving on to the next.

**Some useful UNIX commands:**

<http://uw714doc.sco.com/en/SDK_sysprog/_Getting_Shared_Memory_Segments.html> and <http://uw714doc.sco.com/en/SDK_sysprog/_Getting_Shared_Memory_Segments.html#ipc_iC>

Find more information about these command and options used by them by using UNIX manual or by simply using man command.

**System Cleanup:**

How to kill a process:

*ps* [-f][-a] command shows you processes that are run by you, get the pid of the process and use kill [-9] pid command.

**Releasing shared memory:**

*ipcs* command gives you shared memory id of the shared memory unreleased by you, if you have any.

Type command *ipcrm* -m id to remove shared memory,

or *ipcrm* –s id to remove semaphores.

**Login Information:**

*who* - who is on the system

*whoami* - display the effective current username

*w* - display information about currently logged-in users

*whodo* - who is doing what

**Hardware and Machine Information:**

*version* - display version identification of object file or binary, The version command displays the version of the named file(s).

*machid*, *sun*, *iAPX286*, *i286*, *i386*, *i486*, *i860*, *pdp11*, *sparc*, *u3b*, *u3b2*, *u3b5*, *u3b15*, *vax*, *u370* - get processor type truth value

*uname* - print name of current system

*fpversion* - print information about the system CPU and FPU

*arch* - display the architecture of the current host

**Remote Information:**

*ssh* - secure shell client (remote login program)

*rusers* - who is logged in on remote machines

**Process Control Commands:**

*kill* - send a signal to a process

*suspend* - shell built-in function to halt the current shell

*jobs*, *fg*, *bg*, *stop*, *notify* - control process execution

*halt*, *poweroff* - stop the processor

**Other Commands:**

*limit*, *ulimit*, *unlimit* - set or get limitations on the sys-

*ping* - send ICMP ECHO\_REQUEST packets to network hosts

*man* - find and display reference manual pages