Lab #9: Working with Sigsetjmp, Timers, and IPC

1. Build the executables:
   (a) The Makefile makes use of two environment variables, `$ILIST` and `$LLIST`. From the shell, export the two environment variables given below. May I suggest that you place these commands in your `$HOME/.bash_profile`, so that every time you start up a new bash shell, these two environment variables will be set and exported.
   (b) `export ILIST=-I(path to xtdio.h)` (don't include xtdio.h in this path, just the path to the directory)
   (c) `export LLIST=(path to xtdio.a)`. This path includes the full path to xtdio.a, including xtdio.a itself
   (d) `make`
2. The work below will have you making "Script.xyz" files.
   In addition, it will ask you questions.
   Put answers to these questions in files such as Q*item*; for example, Q3c.
   When done with the lab: `tar -cf MyLab09.tar Script.* Q*`.
   Compress the tarball: `gzip MyLab09.tar`.
   Then email `MyLab09.tar.gz` to me.
3. Work with `sigsetjmp` as shown. Then, record a session of you working with it by using `script Script.sigsetjmp`. `exit` when done recording.

   (a) Follow the program's directions to send it a SIGUSR1 signal.
   (b) Follow the program's directions to send it a SIGUSR2 signal.
   (c) How did the program "know" what its pid was?
   (d) Modify the program to take a SIGPIPE (you need not record your doing this)
   (e) Run `sigsetjmp` and send it a SIGPIPE
4. Work with `timer` as shown. Then, record a session of you working with it by using `script Script.timer`. `exit` when done recording.

   (a) Run `timer`; what happens?
   (b) Modify the program so that it waits 2 seconds, instead. Compile and run it.
5. Work with the message queue programs as shown. Then, record a session of you working with them by using `script Script.mq`. `exit` when done recording.

   (a) Use `msgget` to get a message queue.

(b) Use `msgop` to send the message: `this is my message`

(c) Use `msgop` to receive the message

(d) Use `msgctl` to remove the message queue

(note: using the shell, the command "`ipcs -q`" will show you your message queues)

6. Again, work with the following programs and then record your work with `script Script.sem`, and use `exit` when done recording.

   (a) Use `semget` to get a semaphore set with 5 semaphores.

   (b) Use `semctl` to set the 0^{th} semaphore to 1

   (c) Use `semop` to *test&set* the 0^{th} semaphore towards zero.

   (d) Remove the semaphore set you made using the `semctl` program.

   (note: using the shell, the command "`ipcs -s`" will show you your semaphore sets)

7. Work with the following programs and then record your work in `Script.shm`.

   (a) Use `shmget` to get a shared memory segment of 1000 bytes.

   (b) Pick an appropriate number for the flags.

   (c) Use `shmop` to put the string "This is a test" into your shared memory segment. Read up on `man shmop` to answer the question about "shared memory address shmaddr".

   (d) Use `shmop` again; this time, use "?" to have it query for the contents of the shared memory segment.

   (e) Use `shmctl` and its `IPC_STAT` option.

   (f) Use `shmctl` to lock the shared memory

   (g) Use `shmop` again; this time, use "?" to have it query for the contents of the shared memory segment. Note that `SHM_LOCK` did not prevent `shmop` from querying the shared memory. Why? What does `SHM_LOCK` do?

   (h) Use `shmctl` to remove the shared memory segment

   (note: using the shell, the command "`ipcs -m`" will show you your message queues)

8. helloworld: This program will exercise your use of efence and gdb

   (a) Use the "script hw8.script" command to record your actions (use exit when done)

   (b) Compile helloworld (get source code from website)

   (c) Run: helloworld abcdefghijkl . Did this work?

(d) Re-compile helloworld, but this time with the efence library (and with the `-g` flag)

(e) You may need to use `ulimit -c unlimited` to enable core dumps on your system.
Put that into your `.profile` for permanent enabling of core dumps

(f) Ubuntu users: you'll want to
sudo apt-get update
sudo apt-get install electric-fence

(g) Ubuntu and Mac users: alternative method to get efence:
https://ubuntu.pkgs.org/14.04/ubuntu-universe-amd64/electricfence_2.2.2_amd64.deb.html
Direct link to file:
http://archive_ubuntu.com/ubuntu/pool/universe/e/electricfence/electric_fence_2.2.4_amd64.deb

(h) Run: `helloworld abcdefghijkl` . Did this work?

(i) Use `gdb` to single step through helloworld

(j) Use `b main` to set up a breakpoint at the first line in `main`

(k) Use `s` to single step through the program

(l) Except: use `n` instead of `s` to skip over the internals of working with efence's `calloc`

(m) What happened? Why?