OP SYS HW 3 Deliverable

How to:

* Open a terminal and change the directory to the folder with these files. Then type “make”
* The executable is called “restroomRun” and takes 4 args
  + Number of users: (1-100)
  + Avg arrival time: (1-100)
  + Avg stay time: (1-100)
  + Avg loop count: (1-100)

(Reason for this restraint is use of the usleep linux function which has a max value of 1000000 (and 1000\*100 = 1000000)

Control Module The restroom control module in this program only allows one thread at a time to access any restroom variables, stored in a restroom\_data\_t struct. This is through the use of pthread\_mutex\_t variables that block calling processes if another process already has the lock. The same lock is tested in each function of the restroom interface, so only one thread can be calling any function in the interface at a time. This setup attempts to duplicate the effect of a monitor.

When a thread wants to enter the restroom, but it is of a different gender, the thread blocks due to a condition variable. When the opposite gender has left the restroom, it uses pthread\_cond\_broadcast to alert all threads in the queue that the restroom is now vacant. Then, the opposite gender is free to enter the restroom and test the mutex to enter. Other threads of the same gender will block until it is their turn.

Only one gender is allowed in the restroom at a time. This is ensured through a restroom\_state variable within the restroom\_data\_t struct. This variable either holds the value -1, 0, or 1. The state -1 represents a vacant restroom, 0 represents the enumerated value for male, and 1 represents the enumerated value for female. Each time a thread attempts to enter the restroom, its gender is checked against the current restroom state.

Invariants When no thread is changing the state of the restroom, invariants are as follows:

• No variable within the restroom\_data\_t structure is being edited

• No thread is using any of the restroom functions described in restroom.h (with the exception of get\_elapsed\_ms)

Test Program The test program is what generates all of the threads with random arrival and staying times, a random loop count, and a random gender. The random distribution is based off of the Box-Muller transform, located in the function get\_distributed\_rand. This test program is also where a global restroom\_data\_t struct is created for use with the restroom interface. The Individual function is the function for all of the threads to use and collect data. I managed to prevent threads from printing at the same time by using a pthread\_mutex\_t variable and locking it during printing.

A safe exit occurs when the Individual function calls pthread\_exit(NULL) which is caught by the pthread\_join function in the main thread. When all of the threads are joined back to the main thread, the thread then calls the finalize function of the restroom and then cleanly exits.