Multi-threading Exercise

A good source is the book by Nichols, Buttlar, and Farrell ''Pthreads Programming'' O’Rielly & Associates, 1996

You must learn how to use mutex and condition variables correctly

`pthread_mutex_lock()/pthread_mutex_unlock()`

`pthread_cond_wait()/pthread_cond_signal()/pthread_cond_broadcast()`

You must learn how to handle UNIX signals

`pthread_sigmask()/sigwait()`

`pthread_setcancelstate()/pthread_setcanceltype()`

`pthread_testcancel()`

```c
#include <pthread.h>
/* #include <thread.h> */
thread_t user_threadID;
sigset_t new;

void *handler(), interrupt();

main( int argc, char *argv[] )  {
    sigemptyset(&new);
    sigaddset(&new, SIGINT);
    pthread_sigmask(SIG_BLOCK, &new, NULL);
    pthread_create(&user_threadID, NULL, handler, argv[1]);
    pthread_join(user_threadID, NULL);
    printf("thread handler, %d exited
",user_threadID);
    sleep(2);
    printf("main thread, %d is done
", thr_self());
} /* end main */
```

Look at the man pages of `pthread_sigmask()` on nunki and try to understand the example there

Designate child thread to handler SIGINT

Parent thread blocks SIGINT

```c
struct sigaction act;

void *
handler(char argv1[]) {
    act.sa_handler = interrupt;
    sigaction(SIGINT, &act, NULL);
    pthread_sigmask(SIG_UNBLOCK, &new, NULL);
    printf(" Press CTRL-C to deliver SIGINT
");
    sleep(8);  /* give user time to hit CTRL-C */
}
```

```c
void
interrupt(int sig) {
    printf("thread %d caught signal %d
", thr_self(), sig);
}
```

Child thread example

Child thread unblocks SIGINT

Child thread is designated to handle SIGINT, no other thread will get SIGINT

Ex:

Queueing Abstraction

Arrivals & Departures

$\mu$ : arrival rate

$\lambda$ : service time

$\mu$ : departure rate

$S_1$ : service time

$S_2$ : service time

$Q_1$ : queueing time

$d_i$ : departure time

$r_i$ : response (system) time

$q_i$ : queueing time

$C_1$ : arrival time

$C_2$ : arrival time

$d_1$ : departure time

$d_2$ : departure time

http://merlot.usc.edu/cs551-f12

Bill Cheng

Warm-up Project #2

CS551
Every active object is a thread.

To execute a job for an active object, the thread sleeps for a certain amount of time, called the "next event.

Every active object is awakened by the event queue.

Events are inserted into the event queue according to their time of occurrence. Each event is inserted into a sorted list of events according to their time of occurrence.

To "execute" an event means to notify the corresponding object so it can act accordingly.

Events are inserted into the event queue according to the time of occurrence of the event.

Objects: active objects, passive objects.

Initial events are inserted into the event queue.

Event handling: each object has a "next event"

Event handling (Cont...):

Initialization and random number generation:

Event processing:

Event processing (Cont...):

Arrivals & Departures (Cont...):

Event Handling (Cont...):

Time Driven Simulation:

Event Driven Simulation:

Arrivals & Departures:

Event Handling:

Initialization and random number generation:

Event processing:

Event processing (Cont...):

Arrivals & Departures (Cont...):

Time Driven Simulation:

Event Driven Simulation:

Arrivals & Departures:

Event Handling:

Initial events are inserted into the event queue.

Events are inserted into the event queue according to the time of occurrence of the event.

Every active object is an active object.

Every passive object is a passive object.

Every active object is awakened by the event queue.

Every passive object is a passive object.

Every active object has a "next event"

Every passive object has a "next event"

An event is executed if it is the next event, according to the order of insertion.
Computer Communications - CSCI 551

Time Driven Simulation (Cont.)

Time Driver Simulation

---

Coin Flipping (Cont.)

...
Exponential distribution

Note: inter-arrival time of a Poisson process is Exponentially distributed

\[ m = \frac{1}{x} \]

\[ w = r \times w \]

Calculating Statistics

**Mean and Standard Deviation**

For \( n \) samples, add up all the time and divide by \( n \).

\[ \text{Time in Q} = \text{time between begin service and leave server}. \]

\[ \text{Average time} = \frac{\text{overhead?} \times \text{time in Q}}{\text{amount of time in server}}. \]

\[ \text{Time in server} = \text{amount of time in select()}. \]

\[ \text{Average number of customer at Q1} = \text{average number of customer at a server} \times \text{for n samples, add up all the time and divide by n}. \]

\[ \text{Average number of customer at Q2} = \text{average number of customer at a server} \times \text{for n samples, add up all the time and divide by n}. \]

\[ \text{Average number of customer at Q3} = \text{average number of customer at a server} \times \text{for n samples, add up all the time and divide by n}. \]