CIS 620 Project 2 Spring 2021

(Due: Mar. 18)

The purpose of this project is to familiarize yourself with the design and implementation issues of a user-level thread package on PC/Linux. You need to use the `tar` command with the options `xvfz` to uncompress and extract files from ~cis620s/pub/xt.tar.gz to your working directory.

Part I

The thread package which we discussed in the class is non-preemptive. That is, a thread can run to completion unless it yields the control to other threads. For the first part of this assignment, you need to make the threads preemptive. You can use `signal` and `ualarm` to provide a clock interrupt every 0.01 second. When the clock interrupt occurs, the interrupt handler suspends the current running thread and finds a runnable thread to run. On PC/Linux, you need `sigemptyset, sigaddset,` and `sigprocmask` to unblock the SIGALRM signal to allow the next SIGALRM delivered. Add more comments to the source code.

Part II

You also need to enhance the thread library with the message passing mechanism. The mailboxes are declared as variables with a data type `xthread_mbox_t`. Each message box can hold only one positive integer value. It also has a queue of threads waiting for a message.

You have to implement the following four thread functions:

- `int xthread_init_mbox(xthread_mbox_t *mptr);`
  The mailbox pointed by `mptr` is initialized when `xthread_init_mbox()` is invoked. That is, no message is in the mailbox and the waiting queue in the mailbox is empty.
- `int xthread_send(xthread_mbox_t *mptr, int msg);`
  The function `xthread_send` deposits a message `msg` to the mailbox pointed by `mptr`. If there is a thread waiting for a message, `xthread_send` changes the thread to the ready state and delivers the message to it. On success, `xthread_send` returns 0. On error, it returns -1 if the mailbox is full.
- `int xthread_broadcast(xthread_mbox_t *mptr, int msg);`
  The function `xthread_broadcast` operates much like the function `xthread_send` except that all of the threads waiting in the queue will be unblocked and get the message `msg`.
- `void xthread_recv(xthread_mbox_t *mptr, int *msgptr);`
  The function `xthread_recv` checks whether there is a message in the mailbox pointed by `mptr`. If yes, the message can be delivered to the location pointed by `msgptr`. Otherwise, the calling thread has to wait until a message arrives.

Note that if the SIGALRM occurs during the execution of thread creation/completion, exit/join, etc., the process table may lead to an inconsistent state (why? explain it in your report). To solve this problem, you can use

```
usec = ualarm(0,0);
...
ualarm(usec,0);
```

to disable the timer interrupt at the function entrance and restore it before the function returns.

Turnin

Each group (at most two students) has to submit your report and program electronically. You have to put your report file (i.e. `report.pdf`) under the `xt` dir. Before you submit, you need to use `make clean` to clean all of the object/executable files. Then, on grail, change the directory to the parent directory of the `xt` dir and use the following command to submit the whole `xt` dir:

```
turnin -c cis620s -p proj2 xt
```

Your report should include the description of your code, the thread state transition diagram, experiences in debugging and testing, etc. The cover page should contain your picture(s), name(s) and the login id you used to turnin the project. Start on time and good luck. If you have any questions, send e-mail to `j.sang@csuohio.edu`.