Project 2: Stacks & Basics Report

Operating Systems Design 01:198:416

Tanish Kumar (tak139) & Tanay Kale (tvk20)

1. Structures used:

a. TCB: a struct (declared in thread_worker_types.h) called tcb is used to track a given TCB's threadID, threadContext, status, retVal (return value), and a list of other TCB's that may be waiting to join it called joiners.

b. waitingList: another struct by the name of waitingList (also declared in thread_worker_types.h) is used to keep track of lists of TCBs that are used for the scheduler queue, mutex queue, and every TCB's join queue.

c. worker_mutex_t: a struct (declared in mutex_types.h) called worker_mutex_t is used to track the current owner of a mutex (tcb *current), a waitlist of TCBs waiting to use the mutex (waitingList (wait_list), and the mutex's ID (id).

2. Mutex API:

Note: sig commands were used instead of atomic instructions to lock and unlock these mutexes. The SIG_BLOCK and SIG_UNBLOCK signals passed to sigprocmask allow for this.

a. worker_mutex_init: Assigns a new ID to the given mutex and sets its current owner and waitlist to NULL.

b. worker_mutex_lock: Returns -1 if the mutex provided is NULL or invalid. Otherwise, the scheduler timer is paused (so that the function can complete its work uninterrupted), and the provided mutex's current owner is set to the currently executing TCB (currentTCB). Next, the currentTCB's status is set to BLOCKED and a waitlist is created for the mutex that currently only contains currentTCB. Control is surrendered to the scheduler, which resumes the scheduler timer.

c. worker_mutex_unlock: Returns -1 if the mutex provided is NULL or invalid. Otherwise, the scheduler timer is paused and all the TCBs in the mutex's waitlist are set to READY, after which the mutex's waitlist is freed. The scheduler timer is restarted and control is surrendered to the scheduler.

d. worker_mutex_destroy: Returns -1 if the mutex procided is NULL or invalid. Otherwise, the TCB's in the mutex's waitlist are all set to READY and the waitlist is freed. The mutex's current owner is set to NULL again.

3. Scheduler Logic:

a. We implemented a Round-Robin scheduler for our library. It is initialized by initializeSched() that sets up a main thread and keeps track of the scheduler's context. The main thread remains a part of the scheduler's running queue (which is a closed doubly linked list referred to as "first") at all times so that it is executed periodically. The scheduler's context (schedContext) is set to the schedule() function when initialized.
b. Every thread's status and context are tracked in the TCB struct. Every time a thread is switched, the status is changed (between READY, SCHEDULED, BLOCKED, TERMINATED, and JOINED) and the context is saved.
c. The choice of thread is based on the closest SCHEDULED thread to the currently executing thread.
d. JOINED threads are cleaned up by the nextTCBToRun() function every time it iterates through the running queue.
e. Threads are marked as TERMINATED when they call the exit function.

4. <u>Collaboration and References:</u>

a. This project was completed by Tanay Kale (tvk20) & Tanish Kumar (tak139) collaboratively.
b. We extensively consulted our TAs: Abhilash Nambissan and Eric Jin to help with our planning and debugging stages.
c. We referred to the following links provided in the assignment writeup:
    i. http://man7.org/linux/man-pages/man3/makecontext.3.html
    ii. https://en.wikipedia.org/wiki/Setcontext
    iii. https://linux.die.net/man/3/makecontext
    iv. https://linux.die.net/man/3/swapcontext
    v. https://linux.die.net/man/2/setitimer
    vi. http://www.informit.com/articles/article.aspx?p=23618&seqNum=14
    vii. https://linux.die.net/man/2/sigaction
    viii. https://www.usna.edu/Users/cs/aviv/classes/ic221/s16/lec/20/lec.html
    ix. http://man7.org/linux/man-pages/man3/swapcontext.3.html
    x. https://linux.die.net/man/2/setitimer
d. We extensively referred to and drew from the provided sample code.