POSIX Threads

W4118 Operating Systems I

https://cs4118.github.io/www/2024-1/

Credits to Jae

Processes vs. Threads

- Processes DO NOT share virtual memory address space
- Threads DO share virtual memory address space AND file descriptors
  - but each thread has its own stack

See `bank0.c`

POSIX Mutex API

```
#include <pthread.h>
int pthread_mutex_init(pthread_mutex_t *restrict mutex,
                       const pthread_mutexattr_t *restrict attr);
int pthread_mutex_destroy(pthread_mutex_t *mutex);
        // Both return: 0 if OK, error number on failure
int pthread_mutex_lock(pthread_mutex_t *mutex);
int pthread_mutex_trylock(pthread_mutex_t *mutex);
int pthread_mutex_unlock(pthread_mutex_t *mutex);
        // All return: 0 if OK, error number on failure
#include <pthread.h>
#include <time.h>
int pthread_mutex_timedlock(pthread_mutex_t *restrict mutex,
                            const struct timespec *restrict tsptr);
        // Returns: 0 if OK, error number on failure
```

Deadlock

Deadlock

- Thread tries to lock the same mutex twice
- A thread holds mutex A and tries to lock mutex B, and another thread holds mutex B and tries to lock mutex A

- Strict lock ordering can avoid deadlock
- See APUE Figures 11.11 and 11.12 for an example

Condition Variables

See `condition.c`

More Synchronization Primitives

- Spin locks
- Barriers
- more…