Pthreads

This handout is designed to get you started with using the `pthreads` package. Read /usr/include/pthread.h and appropriate man pages for a complete set of pthread commands available. There are some example programs available on the class web page that demonstrate the use of various pthread commands.

`#include` <pthread.h>

Thread Creation

- `int` pthread_create(`pthread_t` **th*, `pthread_attr_t` **attr*, `void` *(**start*)(`void` *), `void` **arg*);

  Creates a thread with attributes specified by *attr*, starting execution at the function named *start* with the argument value *arg*. A null value of *attr* creates the thread with default attributes.

- `int` pthread_join(`pthread_t` *th*, `void` ***ret*);

  The calling thread waits for the thread identified by *th* to terminate.

Thread Creation Attributes

A thread attributes object is a collection of values that specify how a thread is created. Changes to the attribute values of the object do not affect threads already created using the object. Attributes include size and location of the stack, detached state, and scheduling attributes.

- `int` pthread_attr_init(`pthread_attr_t` **attr*);
- `int` pthread_attr_destroy(`pthread_attr_t` **attr*);
- `int` pthread_attr_setstacksize(`pthread_attr_t` **attr*, `size_t` *sz*);
- `int` pthread_attr_getstacksize(`const pthread_attr_t` **attr*, `size_t` **sz*);
- `int` pthread_attr_setstack(`pthread_attr_t` **attr*, `void` **addr*, `size_t` *stacksize*);
- `int` pthread_attr_getstack(`const pthread_attr_t` **attr*, `void` ***addr*, `size_t` **stacksize*);

Thread Scheduling Attributes

- `int` pthread_attr_setscope(`pthread_attr_t` *$attr$, `int` *scope*);

  Possible values for *scope* are PTHREAD_SCOPE_PROCESS and PTHREAD_SCOPE_SYSTEM. POSIX.1-2001 only requires that an implementation support one of these contention scopes, but permits both to be supported. Linux supports PTHREAD_SCOPE_SYSTEM, but not PTHREAD_SCOPE_PROCESS

- `int` pthread_attr_getscope(`const pthread_attr_t` *$attr$, `int` **scope*);

- `int` pthread_attr_setinheritsched(`pthread_attr_t` *$attr$, `int` *inherit*);

  Attributes of a newly created thread may be inherited from the parent thread or explicitly stated. Value for *inherit* must be PTHREAD_INHERIT_SCHED or PTHREAD_EXPLICIT_SCHED (default).

- `int` pthread_attr_getinheritsched(`const pthread_attr_t` *$attr$, `int` **inherit*);

- `int` pthread_attr_setschedpolicy(`pthread_attr_t` *$attr$, `int` *policy*);

  policy can be SCHED_FIFO, SCHED_RR, or SCHED_OTHER(default, same as SCHED_RR).

- `int` pthread_attr_getschedpolicy(`const pthread_attr_t` *$attr$, `int` **policy*);

Mutex

Mutual exclusion locks (mutexes) are used to serialize the execution of threads through critical sections of code which access shared data. A successful call for a mutex lock via `pthread_mutex_lock()` or `pthread_mutex_trylock()` will cause another thread that tries to acquire the same mutex via `pthread_mutex_lock()` to block until the owning thread calls `pthread_mutex_unlock()`.

- `int` pthread_mutex_init(`pthread_mutex_t` **mutex*, `const pthread_mutexattr_t` *$attr$);

  Initializes a mutex structure. A null value of *attr* initializes mutex with default attributes.

- `int` pthread_mutex_destroy(`pthread_mutex_t` **mutex*);

- `int` pthread_mutex_lock(`pthread_mutex_t` **mutex*);

- `int` pthread_mutex_trylock(`pthread_mutex_t` **mutex*);

- `int` pthread_mutex_unlock(`pthread_mutex_t` **mutex*);

Mutex Attributes

There are several attributes associated with a mutex structure. Read man pages of `pthread_mutexattr_init`, `pthread_mutexattr_setprotocol`, and `pthread_mutexattr_setprioceiling`.

Condition Variables

Condition variables provide high performance synchronization primitives to wait for or wake up threads waiting for certain conditions to be satisfied. Functions are provided to wait on a condition variable and to wake up (signal) threads that are waiting on the condition variable.

- `int` pthread_cond_init(`pthread_cond_t` **cond*, `pthread_condattr_t` **attr*);
- `int` pthread_cond_destroy(`pthread_cond_t` **cond*);
- `int` pthread_cond_signal(`pthread_cond_t` **cond*);
- `int` pthread_cond_broadcast(`pthread_cond_t` **cond*);
- `int` pthread_cond_wait(`pthread_cond_t` **cond*, `pthread_mutex_t` **mutex*);
- `int` pthread_cond_timedwait(`pthread_cond_t` **cond*, `pthread_mutex_t` **mutex*, `const struct timespec` **abstime*);

  Functions pthread_cond_wait() and pthread_cond_timedwait() are used to block on a condition variable. They must be called with mutex locked by the calling thread. The functions atomically release mutex and block the calling thread on the condition variable cond. Before return, the mutex is reacquired for the calling thread. The function pthread_cond_timedwait() is the same as pthread_cond_wait() except that an error is returned if the absolute time specified by abstime passes before the waiting thread is signalled. If a time-out occurs, pthread_cond_timedwait() still reacquires mutex before returning to the caller.