

Threading

Threads are “cheap processes” almost always share memory with other threads.

There are Kernel Threads and User Threads.

Kernel Threads – Created and managed by OS.

User Threads – Created and managed by user – usually with a threading package.

Threading models

Many-to-One	All threading Package
One-to-One	All OS
Many-to-Many	Mixed

Pthreads

<http://www.llnl.gov/computing/tutorials/pthreads/>

This is the best on-line resource for pthreads information

Creating threads:

```
pthread_create (thread,attr,start_routine,arg)
pthread_exit (status)
```

```
pthread_attr_init (attr)
pthread_attr_destroy (attr)
```

```
pthread_join();
```

Mutex Variables

```
pthread_mutex_t mymutex = PTHREAD_MUTEX_INITIALIZER;
```

```
pthread_mutex_init (mutex,attr)
pthread_mutex_destroy (mutex)
pthread_mutexattr_init (attr)
pthread_mutexattr_destroy (attr)
```

```
pthread_mutex_lock (mutex)
pthread_mutex_trylock (mutex)
pthread_mutex_unlock (mutex)
```

Condition Variables

```
pthread_cond_t myconvar = PTHREAD_COND_INITIALIZER;
```

```
pthread_cond_init (condition,attr)
```

```
pthread_cond_destroy (condition)
```

```
pthread_condattr_init (attr)
```

```
pthread_condattr_destroy (attr)
```

```
pthread_cond_wait (condition,mutex)
```

```
pthread_cond_signal (condition)
```

Linux Clone System Call

```
int clone(int (*fn)(void *), void *child_stack, int flags, void *arg);
```

```
syscall2(int, clone, int, flags, void *, child_stack)
```

Flags:

```
CLONE_PARENT
```

```
CLONE_FS
```

```
CLONE_FILES
```

```
CLONE_NEWNS
```

```
CLONE_SIGHAND
```

```
CLONE_PTRACE
```

```
CLONE_VFORK
```

```
CLONE_VM
```

```
CLONE_PID
```

```
CLONE_THREAD
```