Pthreads API

Gaurav Somani

Reference : *Bil Lewis and Daniel J. Berg*, PThreads Primer - A Guide to Multithreaded Programming

Pthreads Library

Creating Threads

Terminating Thread Execution

Passing Arguments To Threads

Thread Identifiers

Joining Threads

Detaching / Undetaching Threads

The pthreads API

The subroutines which comprise the Pthreads API can be informally grouped into three major classes:

Thread management: threads - creating, detaching, joining, etc. They include functions to set/query thread attributes (joinable, scheduling etc.)

Mutexes: Mutex functions provide for creating, destroying, locking and unlocking mutexes ("**mutual exclusion**").

Condition variables: This class includes functions to create, destroy, wait and signal based upon specified variable values. Functions to set/query condition variable attributes are also included.

Naming conventions: All identifiers in the threads library begin with **pthread_**

The pthreads API

int **pthread_create**(pthread t *th, pthread attr t *attr, void *(*start)(void *), void *arg);

void **pthread_exit**(void *retval);

int **pthread_join**(pthread_t threadhandle, void **returnvalue);

int pthread_cancel(pthread_t thread);

Thread Initialization :

Include the pthread.h library :

```
#include <pthread.h>
```

Declare a variable of type pthread_t :

```
pthread_t    the_thread
```

When you compile, add -lpthread to the linker flags :

```
cc or gcc   threads.c   -o   threads  -lpthread
```

Initially, threads are created from within a process. Once created, threads are peers, and may create other threads. Note that an "initial thread" exists by default and is the thread which runs main.

Thread Identifiers :

pthread_self ( )

Returns the unique thread ID of the calling thread.

The returned data object is opaque can not be easily inspected.

pthread_equal (thread1, thread2)

Compares two thread IDs:

If the two IDs are different 0 is returned, otherwise a non-zero value is returned.

Because thread IDs are opaque objects, the C language equivalence operator == should not be used to compare two thread IDs.

Thanks