1. Difference Between threads and processes  
다음은 스레드와 프로세스 간의 주요 차이점 중 일부입니다.  
프로세스는 동일한 프로세스에서 실행되는 스레드가 주소 공간을 공유하는 동안 프로세스는 자신의 주소 공간을 공유하지 않습니다.  
above 위의 점에서 프로세스는 서로 독립적으로 실행되며 프로세스 간의 동기화는 커널에 의해서만 처리됩니다. 반면에 스레드 동기화는 스레드가 실행되는 프로세스에주의를 기울여야합니다  
threads 프로세스 간 컨텍스트 스위칭에 비해 스레드 간 컨텍스트 스위칭이 빠름  
two 두 프로세스 간의 상호 작용은 표준 프로세스 간 통신을 통해서만 달성되는 반면 동일한 프로세스에서 실행되는 스레드는 메모리, 텍스트 세그먼트 등과 같은 대부분의 리소스를 공유하므로 쉽게 통신 할 수 있습니다.

2. User threads Vs Kernel Threads

사용자 공간 스레드는 사용자 공간 스레드 라이브러리를 사용하여 생성, 제어 및 소멸됩니다. 이 스레드는 커널에 알려지지 않으므로 커널은 아무 것도 처리 과정에 관여하지 않습니다. 이 쓰레드는 협업 멀티 태스킹을 따르며, 쓰레드에서 자신의 의도대로 CPU를 해제합니다. 즉, 스케쥴러가 쓰레드를 선점 할 수 없습니다. 사용자 공간 스레드의 장점은 두 스레드 간의 전환이 많은 오버 헤드를 필요로하지 않으며 일반적으로 매우 빠르다는 점입니다. 이러한 스레드는 협업 멀티 태스킹을 따르므로 한 스레드가 블록되면 전체 프로세스가 차단됩니다.

커널 공간 스레드는 커널에 의해 생성, 제어 및 파괴됩니다. 사용자 공간에있는 모든 스레드에 대해 해당 커널 스레드가 있습니다. 이러한 스레드는 커널에 의해 관리되므로 스케줄러가 실행 준비가 된 우선 순위가 높은 스레드로 실행중인 스레드를 선점 할 수있는 선점 형 멀티 태스킹을 수행합니다. 커널 스레드의 주요 이점은 스레드 중 하나가 차단 된 경우에도 커널 스레드가 선점 형 스케줄링을 수행 할 때 전체 프로세스가 차단되지 않는 반면 부정적인면에서는 컨텍스트 전환이 사용자 공간 스레드와 비교할 때 매우 빠르지 않다는 것입니다.

3. Thread Identification

프로세스 ID를 통해 프로세스가 식별되는 것처럼 스레드는 스레드 ID로 식별됩니다. 그러나 흥미롭게도이 두 가지의 유사점은 여기에서 끝납니다.

프로세스 ID는 한 시스템에서 고유하지만 스레드 ID가 단일 프로세스의 컨텍스트에서만 고유합니다.

process 프로세스 ID는 정수 값이지만 스레드 ID는 정수 값일 필요는 없습니다. 그것은 구조 일 수있다.

thread 스레드 ID가 인쇄하기 쉽지 않으면 서 프로세스 ID를 매우 쉽게 인쇄 할 수 있습니다.

스레드 ID는 'pthread\_t'유형으로 표시됩니다. 이미 대부분의 경우에이 유형은 구조체이므로 두 스레드 ID를 비교할 수있는 함수 여야합니다.

#include <pthread.h>

int pthread\_equal(pthread\_t tid1, pthread\_t tid2);

4. Thread Termination

각 프로그램은 main () 함수가 실행되는 스레드 인 적어도 하나의 스레드로 시작됩니다. 따라서 프로그램에서 실행되는 모든 스레드의 최대 수명은 주 스레드의 최대 수명입니다. 그래서 우리가 주 스레드가 다른 모든 스레드가 끝날 때까지 기다려야한다면 pthread\_join () 함수가있다

#include <pthread.h>

int pthread\_join(pthread\_t thread, void \*\*rval\_ptr);

위의 함수는 부모 스레드가 완료 될 때까지 종료되지 않도록합니다. 이 함수는 부모 스레드 내에서 호출되며 첫 번째 인수는 대기 할 스레드의 스레드 ID이고 두 번째 인수는 부모 스레드가 기다려야하는 스레드의 반환 값입니다. 반환 값에 관심이 없다면이 포인터를 NULL로 설정할 수 있습니다.

스레드는 세 가지 방법으로 종료 될 수 있습니다.  
스레드가 시작 루틴에서 돌아 오는 경우.  
2. 다른 스레드에 의해 취소 된 경우. 여기서 사용되는 함수는 pthread\_cancel ()입니다.  
3. pthread\_exit () 함수가 자체에서 호출되면.  
  
여기서 초점은 pthread\_exit ()에있을 것이다. 프로토 타입은 다음과 같습니다.

#include <pthread.h>

void pthread\_exit(void \*rval\_ptr);

따라서이 함수는이 함수를 호출하는 스레드에서 반환되는 하나의 인수 만 받아들입니다. 이 반환 값은이 스레드가 종료되기를 기다리고있는 부모 스레드에 의해 액세스됩니다. pthread\_exit () 함수에 의해 종료 된 쓰레드의 반환 값은 위에서 설명한 pthread\_join의 두 번째 인자에서 접근 할 수있다.