int pthread\_create(pthread\_t \*thread, const pthread\_attr\_t \*attr, void \*(\*start\_routine)(void \*), void \*arg);

첫번째 아규먼트인 thread 는 쓰레드가 성공적으로 생성되었을때 생성된 쓰레드를 식별하기 위해서 사용되는 쓰레드 식별자이다.

두번째 아규먼트인 attr 은 쓰레드 특성을 지정하기 위해서 사용하며, 기본 쓰레드 특성을 이용하고자 할경우에 NULL 을 사용한다.

3번째 아규먼트인 start\_routine는 분기시켜서 실행할 쓰레드 함수이며,

4번째 아규먼는인 arg는 쓰레드 함수의 인자이다.

int pthread\_join(pthread\_t th, void \*\*thread\_return);

첫번째 아규먼트 th는 기다릴(join)할 쓰레드 식별자이며

두번째 아규먼트 thread\_return은 쓰레드의 리턴(return) 값이다.

thread\_return 이 NULL 이 아닐경우 해다 포인터로 쓰레드 리턴 값을 받아올수 있다.

int pthread\_detach(pthread\_t th);

detach는 main 쓰레드에서 pthread\_create 를 이용해 생성된 쓰레드를 분리시킨다.

이 함수는 식별번호th인 쓰레드를 detach 시키는데

detach 되었을경우 해당(detach 된) 쓰레드가 종료될경우

pthread\_join 을 호출하지 않더라도 즉시 모든 자원이 해제(free) 된다

void pthread\_exit(void \*retval);

pthread\_exit 는 현재 실행중인 쓰레드를 종료시키고자 할때 사용한다

만약 pthread\_cleanup\_push 가 정의되어 있다면, pthread\_exit 가 호출될경우 cleanup handler 가 호출된다

보통 이 cleanup handler 은 메모리를 정리하는 등의 일을 하게 된다.

void pthread\_cleanup\_push(void (\*routine) (void \*), void \*arg);

이것은 cleanup handlers 를 인스톨하기 위해서 사용된다

pthread\_exit(3)가 호출되어서 쓰레드가 종료될때 pthread\_cleanup\_push 에 의해서 인스톨된 함수가 호출된다

routine이 쓰레드가 종료될때 호출되는 함수이다. arg는 아규먼트이다.

cleanup handlers 는 주로 자원을 되돌려주거나, mutex 잠금등의 해제를 위한 용도로 사용된다.

만약 mutex 영역에서 pthread\_exit 가 호출되어 버릴경우 다른쓰레드에서 영원히 block 될수 있기 때문이다.

또한 malloc 으로 할당받은 메모리, 열린 파일지정자를 닫기 위해서도 사용한다.

void pthread\_cleanup\_pop(int execute);

만약 execute 가 0 이라면, pthread\_cleanup\_push 에 의해 인스톨된 cleanup handler 를 (실행시키지 않고)삭제만 시킨다.

0 이 아닌 숫자라면 cleanup handler 을 실행시키고 삭제 된다.

그리고 pthread\_cleanup\_push 와 pthread\_cleanup\_pop 은 반드시 같은 함수내의 같은 레벨의 블럭에서 한쌍으로 사용해야 한다.

int pthread\_mutex\_init(pthread\_mutex\_t \* mutex, const pthread\_mutex\_attr \*attr);

mutex 는 여러개의 쓰레드가 공유하는 데이타를 보호하기 위해서 사용되는 도구로써

보호하고자 하는 데이타를 다루는 코드영역을 단지 한번에 하나의 쓰레드만 실행가능 하도록 하는 방법으로 공유되는 데이타를 보호한다.

이러한 코드영역(하나의 쓰레드만 점유가능한)을 critical section 이라고 하며, mutex 관련 API 를 이용해서 관리할수 있다.

pthread\_mutex\_init 는 mutex 객체를 초기화 시키기 위해서 사용한다.

첫번째 인자로 주어지는 mutex 객체 mutex를 초기화시키며, 두번째 인자인 attr 를 이용해서 mutex 특성을 변경할수 있다.

기본 mutex 특성을 이용하기 원한다면 NULL 을 사용하면 된다.

mutex 특성(종류) 에는 "fast", "recurisev", "error checking" 의 종류가 있으며, 기본으로 "fast" 가 사용된다.

int pthread\_mutex\_destroy(pthread\_mutex\_t \*mutex);

인자로 주어진 뮤텍스 객체 mutex 를 제거하기 위해서 사용된다.

mutex 는 pthread\_mutex\_init()함수를 이용해서 생성된 뮤텍스 객체이다.

pthread\_mutex\_destroy 를 이용해서 제대로 mutex 를 삭제하려면 이 mutex 는 반드시 unlock 상태이여야 한다.

int pthread\_mutex\_lock(pthread\_mutex\_t \*mutex);

pthread\_mutex\_lock 는 critcal section 에 들어가기 위해서 mutex lock 을 요청한다.

만약 이미 다른 쓰레드에서 mutex lock 를 얻어서 사용하고 있다면 다른 쓰레드에서 mutex lock(뮤텍스 잠금) 을 해제할때까지(사용할수 있을때까지) 블럭 된다.

만약 다른 어떤 쓰레드에서도 mutex lock 을 사용하고 있지 않다면, 즉시 mutex lock 을 얻을수 있게 되고 critcal section 에 진입하게 된다.

critcal section 에서의 모든 작업을 마쳐서 사용하고 있는 mutex lock 이 더이상 필요 없다면 pthread\_mutex\_unlock 를 호출해서 mtuex lock 를 되돌려준다.

int pthread\_mutex\_unlock(pthread\_mutex\_t \*mutex);

critical section 에서의 모든 작업을 마치고 mutex lock 을 돌려주기 위해서 사용한다.

pthread\_mutex\_unlock 를 이용해서 mutex lock 를 되돌려주면 다른 쓰레드에서 mutex lock 를 얻을수 있는 상태가 된다.

int pthread\_cond\_init(pthread\_cond\_t \*cond, const pthread\_cond\_attr \*attr);

pthread\_cond\_init는 조견변수 (condition variable)cond를 초기화하기 위해서 사용한다.

attr 를 이용해서 조건변수의 특성을 변경할수 있으며, NULL 을 줄경우 기본특성으로 초기화된다.

조건변수 cond는 상수 PTHREAD\_COND\_INITIALIZER 을 이용해서도 초기화 할수 있다.

즉 다음과 같은 2가지 초기화 방법이 존재한다.

pthread\_cond\_t cond = PTHREAD\_COND\_INITIALIZER;

pthread\_cond\_init(&cond, NULL);

int pthread\_cond\_signal(pthread\_cond\_t \*cond);

조건변수 cond에 시그날을 보낸다. 시그날을 보낼경우 cond에서 기다리는(wait) 쓰레드가 있다면 쓰레드를 깨우게 된다(봉쇄가 풀림).

만약 조건변수 cond를 기다리는 쓰레드가 없다면, 아무런 일도 일어나지 않게되며, 여러개의 쓰레드가 기다리고 있다면 그중 하나의 쓰레드에게만 전달된다.

이때 어떤 쓰레드에게 신호가 전달될지는 알수 없다.

int pthread\_cond\_broadcast(pthread\_cond\_t \*cond);

조건변수 cond에서 기다리는(wait) 모든 쓰레드에게 신호를 보내서, 깨운다는 점을 제외하고는 pthread\_cond\_signal과 동일하게 작동한다.

int pthread\_cond\_wait(pthread\_cond\_t cond, pthread\_mutex\_t \*mutex);

조건변수 cond를 통해서 신호가 전달될때까지 블럭된다. 만약 신호가 전달되지 않는다면 영원히 블럭될수도 있다.

pthread\_cond\_wait는 블럭되기 전에 mutex 잠금을 자동으로 되돌려준다.

int pthread\_cond\_timedwait(pthread\_cont\_t \*cond, pthread\_mutex\_t \*mutex, const struct timespec \*abstime);

조건변수 cond를 통해서 신호가 전달될때까지 블럭되며 자동으로 mutex을 돌려주는 점에서는 pthread\_cond\_wait와 동일하다.

그러나 시간체크가 가능해서 abstime시간동안 신호가 도착하지 않는다면 error 를 발생하면서 리턴한다.

이때 리턴값은 ETIMEDOUT 이다. errno 가 세팅되는게 아닌, 리턴값으로 에러가 넘어오는것에 주의해야 한다.

또한 pthread\_cond\_timedwait함수는 다른 signal 에 의해서 interrupted 될수 있으며 이때 EINTR 을 리턴한다.

이 함수를 쓸때는 interrupted 상황에 대한 처리를 해주어야 한다.

int pthread\_cond\_destroy(pthread\_cond\_t \*cond);

pthread\_cond\_init를 통해서 생성한 조건변수cond에 대한 자원을 해제한다.

destroy 함수를 호출하기 전에 어떤 쓰레드도 cond에서의 시그널을 기다리지 않는걸 확인해야 한다.

만약 cond 시그널을 기다리는 쓰레드가 존재한다면 이 함수는 실패하고 EBUSY 를 리턴한다.

int pthread\_attr\_init(pthread\_attr\_t \*attr);

pthread\_attr\_init는 thread attribute 객체인 attr을 디폴트 값으로 초기화 시킨다.

성공할경우 0을 돌려주고 실패할경우 -1 을 되돌려준다.

int pthread\_attr\_destroy(pthread\_attr\_t \*attr);

pthread\_attr\_init에 의해 생성된 thread attribute 객체인 attr을 제거한다.

제거된 attr 을 다시 사용하기 위해서는 pthread\_attr\_init를 이용해서 다시 초기화 해주어야 한다.

int pthread\_attr\_getscope(const pthread\_attr\_t \*attr, int \*scope);

쓰레드가 어떤 영역(scope)에서 다루어지고 있는지를 얻어오기 위해서 사용된다.

PTHREAD\_SCOPE\_SYSTEM과 PTHREAD\_SCOPE\_PROCESS 의 2가지 영역중에 선택할수 있다.

SYSTEM 영역 쓰레드는 user 모드 쓰레드라고 불리우며, PROCESS 쓰레드는 커널모드 쓰레드라고 불리운다.

리눅스의 경우 유저모드 쓰레드인데, 즉 커널에서 쓰레드를 스케쥴링하는 방식이 아닌 쓰레드 라이브러리를 통해서 쓰레드를 스케쥴링 하는 방식을 사용한다.

int pthread\_attr\_setscope(pthread\_attr\_t \*attr, int scope);

쓰레드가 어떤 영역(scope)에서 작동하게 할것인지 결정하기 위해서 사용한다.

리눅스의 경우 Kernel mode 쓰레드를 지원하지 않음으로 오직 PTHREAD\_SCOPE\_SYSTEM 만을 선택할수 있다.

반면 솔라리스는 유저모드와 커널모드중 선택이 가능하다.

int pthread\_attr\_getdetachstate(pthread\_attr\_t \*attr, int detachstate);

쓰레드가 join 가능한 상태(PTHREAD\_CREATE\_JOINABLE) 인지 detached 상태인지 (PTHREAD\_CREATE\_DETACHED) 인지를 알아낸다.

알아낸 값은 아규먼트 detachstate에 저장된다.

기본은 PTHREAD\_CREATE\_JOINABLE 이며, pthread\_detach를 이용해서 생성된 쓰레드를 detach 상태로 만들었을경우

또는 pthread\_attr\_setdetachstate함수를 이용해서 쓰레드를 detache 상태로 변경시켰을경우 PTHREAD\_CREATE\_DETACHED 상태가 된다.

int pthread\_attr\_setdetachstate(pthread\_attr\_t \*attr, int detachstate);

쓰레드의 상태를 PTHREAD\_CREATE\_JOINABLE 혹은 PTHREAD\_CREATE\_DETACHED 상태로 변경시키기 위해서 사용된다.

int pthread\_sigmask(int how, const sigset\_t \*newmask, sigset\_t \*oldmask);

쓰레드에서 시그널은 서로 공유된다. 그런이유로 만약 프로세스에 시그널이 전달되면 프로세스가 생성된 모든 쓰레드로 시그널이 전달된다.

그러나 특정 쓰레드만 시그널을 받도록 하고 싶을 때가 있을 것이다. 이경우 이 함수를 이용하면 된다.

int pthread\_kill(pthread\_t thread, int signo);

쓰레드 식별번호 thread로 signo번호의 시그널을 전달한다.

int sigwait(const sigset\_t \*set, int \*sig);

시그널 전달을 동기적으로 기다린다.

쓰레드 취소

int pthread\_cancel(pthread\_t thread);

int pthread\_setcancelstate(int state, int \*oldstate);

int pthread\_setcancelstate(int state, int \*oldstate);

int pthread\_setcanceltype(int type, int \*oldtype);

void pthread\_testcancel(void);