7주차 3차시 스레딩 생성/소멸

【학습목표】

1. 리눅스 스레드 구현에 대해 설명할 수 있다.
2. Pthread 생성 및 종료 관련 함수를 사용할 수 있다.

학습내용1 : 리눅스 스레드 구현

1. 스레드 생성

* 특징
스레드의 집합과 자원의 집합
수행 유닛은 스레드임
프로세스는 스레드의 수행 환경을 제공
Thread ID를 가지고 있음

2. 프로세스와 스레드 함수 비교

프로세스 함수	스레드함수	설명
fork	pthread_create	제어의 새로운 흐름을 만듦
exit	pthread_exit	존재하는 제어의 흐름으로부터 탈출
waitpid	pthread_join	제어의 흐름으로부터 탈출 상태를 획득
getpid	pthread_self	제어의 흐름을 위한 ID 획득
abort	pthread_cancel	제어 흐름의 비정상적인 종료를 요청

학습내용2 : Pthread 생성 및 종료

1. Pthread_self() 함수

자신의 스레드 ID를 얻음
성공 : 호출한 스레드의 스레드ID를 리턴
실패 : 정의 되어 있지 않음

```
#include <pthread.h>

Pthread_t pthread_self(void);
```

2. Pthread_create() 함수

스레드를 생성함
성공 시 : 0을 리턴
실패 시 : 0이 아닌 에러코드 리턴(EAGAIN,EINVAL, EPERM)

```
#include <pthread.h>

int pthread_create(pthread_t *restrict thread, const pthread_attr_t *restrict attr,
                   void *(*start_routine)(void*), void *restrict arg);
```

thread : 스레드 ID를 가리키는 포인트
attr : 스레드와 관련된 특성을 지정하기 위한 용도로 사용
마지막 두 개의 인자는 실행을 시작할 함수와 이 함수에 전달할 인자를 스레드에게 알려줌

예

```
1  #include <pthread.h>
2  #include <stdio.h>
3  #include <unistd.h>
4  #include <sys/types.h>
5  #include <stdlib.h>
6
7  pthread_t ntid;
8
9  void printids(const char *s)
10 {
11        pid_t pid;
12        pthread_t tid;
13        pid = getpid();
14        tid = pthread_self();
15        printf("%s pid %u tid %u (0x%x) ₩n", s, (unsigned int)pid,
                  (unsigned int)tid, (unsigned int)tid);
16 }
17
18 void *thr_fn(void *arg)
19 {
20        printids("new thread: ");
21        return ((void*)0);
22 }
23
24 int main(void)
25 {
26        int err;
27        err = pthread_create(&ntid, NULL, thr_fn, NULL);
28        if(err != 0)
29                  exit(1);
30        printids("main thread: ");
31        sleep(1);
32        exit(0);
33 }
```

```
# gcc -lpthread exThr_1.c -o exThr_1.o
#./exThr_1.o
New thread: pid 11310 tid 3086039952 (0xb7f13b90)
Main thread: pid 11310 tid 3086042816 (0xb7f146c0)
```

3. Pthread_exit() 함수

프로세스를 종료시키지 않고, 호출한 스레드만 종료
성공 시 : 0을 리턴
실패 시 : 에러 정의되어 있지 않음

```
#include <pthread.h>

void pthread_exit(void *value_ptr);
```

Value_ptr : 스레드가 참조하는 데이터를 가르키는 포인터
(pthread_join 함수 호출이 성공했을 때 사용)

4. Pthread_join() 함수 (=waitpid)

명시된 스레드가 종료할 때까지 호출한 스레드를 대기시킴
성공 : 0을 리턴
실패 : 0이 아닌 에러코드 리턴

```
#include <pthread.h>

void pthread_join(pthread_t thread, void **value_ptr);
```

thread : 종료를 기다리는 스레드 (thread가 종료되어야 현재 스레드는 join에서 벗어남)
Value_ptr : 타겟 스레드의 리턴 상태를 가리키는 포인터를 저장하기 위한 위치, 리턴 상태는 타겟 스레드가 pthread_exit 함수나 return 문으로 넘겨준 값

5. Pthread_cancel() 함수

다른 스레드를 제거
성공 : 0을 리턴
실패 : 0이 아닌 에러코드 리턴

```
#include <pthread.h>

int pthread_cancel(pthread_t thread);
```

thread : 취소하고자 하는 스레드 ID

6. Pthread_cleanup_push() 함수

스레드 cleanup handlers 를 스택 메모리에 push 함
성공 : 0을 리턴
실패 : 0이 아닌 에러코드 리턴

```
#include <pthread.h>

int pthread_cleanup_push(void (*routine)(void*), void *arg);
```

7. arg : 스레드 핸들러 루틴 argument

Pthread_cleanup_pop() 함수
스택에 push된 cleanup handler를 제거함

```
#include <pthread.h>

int pthread_cleanup_pop(int execute);
```

execute : 0 또는 0 이 아닌 값

【학습정리】

1. 프로세스와 스레드 함수 비교
(프로세스 함수 / 스레드 함수 : 설명)
- fork / pthread_create : 제어의 새로운 흐름을 만듦
- exit / pthread_exit : 존재하는 제어의 흐름으로부터 탈출
- waitpid / pthread_join : 제어의 흐름으로부터 탈출 상태를 획득
- getpid / pthread_self : 제어의 흐름을 위한 ID 획득
- abort / pthread_cancel : 제어 흐름의 비정상적인 종료를 요청

2. Pthread_self() 함수
- 자신의 스레드 ID를 얻음
- 성공 : 호출한 스레드의 스레드ID를 리턴
- 실패 : 정의 되어 있지 않음

* 스택 영역
- 함수 호출 시 되돌아갈 주소 값 및 지역변수(함수 안에서 선언한)를 저장하기 위한 메모리 공간
- 함수 호출 시 필요한 메모리 영역

* 코드 영역
- 코드영역의 main(), add(), subtract(), Devide()등을 다수의 스레드가 공유하면서 호출 가능

3. Pthread_create() 함수
- 스레드를 생성함
- 성공 시 : 0을 리턴
- 실패 시 : 0이 아닌 에러코드 리턴(EAGAIN,EINVAL, EPERM)

4. Pthread_exit() 함수
- 프로세스를 종료시키지 않고, 호출한 스레드만 종료
- 성공 시 : 0을 리턴
- 실패 시 : 에러 정의되어 있지 않음

5. Pthread_join() 함수 (=waitpid)
- 명시된 스레드가 종료할 때까지 호출한 스레드를 대기시킴
- 성공 : 0을 리턴
- 실패 : 0이 아닌 에러코드 리턴

6. Pthread_cancel() 함수
- 다른 스레드를 제거
- 성공 : 0을 리턴
- 실패 : 0이 아닌 에러코드 리턴

7. Pthread_cleanup_push() 함수
- 스레드 cleanup handlers 를 스택 메모리에 push 함
- 성공 : 0을 리턴
- 실패 : 0이 아닌 에러코드 리턴

8. Pthread_cleanup_pop() 함수
- 스택에 push된 cleanup handler를 제거함