실습 #1

아래 프로그램 thr1.c을 컴파일하고 실행하시오.

컴파일 방법

```
gcc  -o thr1  thr1.c  -lpthread
```

실행

```
./thr1 &
```

1. 몇 개의 프로세스가 생성되었는가?

   ps

2. 몇 개의 스레드가 수행되는가?

   스레드 확인 방법: ps -L

   이때 LWP는 무엇을 의미하는가? PID와의 차이를 설명하시오.

3. 생성된 스레드(자식)는 어떤 일을 하는가?
   또 main 스레드(부모)는 어떤 일을 하는가?

4. pthread_join() 역할과 pthread_exit()의 역할을 설명하시오.

[thr1.c]

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

void * func(void *i);
int main(void)
{
        pthread_t       thr;

        printf("\n");
        pthread_create(&thr, NULL, func, NULL);
        printf("Main thread ...\n");
        pthread_join(thr, NULL);
        printf("Joined ...\n");
}

void * func(void *arg)
{
       printf("Child thread ...\n");
       sleep(60);
       pthread_exit(NULL);
}
```

실습 #2

아래 프로그램 fork5.c을 컴파일하고 실행하시오.

컴파일 방법

```
gcc  -o fork5  fork5.c
```

실행

```
./fork5 &
```

1. 실행 결과는?
2. 부모와 자식 프로세스는 변수 global 값을 공유하는가?

[fork5.c]

```
#include <unistd.h>
#include <sys/wait.h>
#include <stdio.h>
#include <stdlib.h>

void  X(int  i);
int  global = 0;
int  main(void)
{
       int    status;
       if (fork() != 0) {
          X(1);
          waitpid(-1, &status, 0);
       } else {
          X(2);
          exit(0);
       }
       printf("global = %d after child process exits\n", global);
}
void X(int  i)
{
       global = i;
       printf("global = %d\n", global);
}
```

실습 #3

아래 프로그램 thr2.c을 컴파일하고 실행하시오.

컴파일 방법

```
gcc  -o thr2  thr2.c  -lpthread
```

실행

```
./thr2
```

종료

Ctrl-C로 종료

1. 실행 결과는?
2. main thread와 child thread는 변수 global을 공유한는가?

[thr2.c]

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

void * X(void *p);
void   Y(int j);

int  global = 0;
int main()
{
   pthread_t   t1;
   void              *status;

   printf("\n");
   pthread_create(&t1, NULL, X, NULL);
   Y(1);
   pthread_join(t1, &status);
   printf("global=%d  in  main  thread  after  the  child  thread  exiting\n",
global);
}
void * X(void *p)
{
   int i;
   do {
      global = 2;
      for (i = 0; i < 100000; i++) ;
      printf("Child thread:            global=%d\n", global);
   } while (1);
   pthread_exit((void *)NULL);
}
void Y(int j)
{
   int i;
   do {
      global = j;
      for (i = 0; i < 100000; i++) ;
      printf("Main thread: global=%d\n", global);
   } while (1);
}
```