Pthread 실습 01

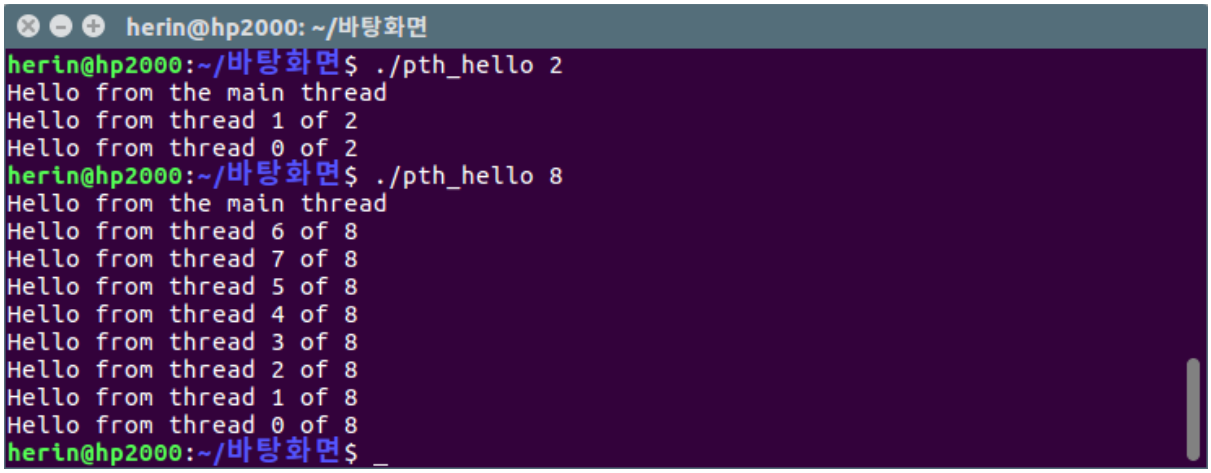
컴퓨터과학부 1515386 정혜린

실습 01 간단한 Pthread 프로그램

herin@hp2000: ~/바탕화면

```
herin@hp2000:~/바탕화면$ ls
KakaoTalk.desktop  pth_ave.c  pth_hello.c  sem_pi.c
pth_ave            pth_hello  sem_pi       timer.h
herin@hp2000:~/바탕화면$ cat pth_hello.c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

int thread_count;
void* Hello(void* rank);

int main(int argc, char* argv[]) {
        long thread;
        pthread_t* thread_handles;

        thread_count = strtol(argv[1], NULL, 10);

        thread_handles = malloc(thread_count * sizeof(pthread_t));

        for(thread = 0; thread < thread_count; thread++)
                pthread_create(&thread_handles[thread], NULL, Hello, (void*)thre
ad);

        printf("Hello from the main thread\n");

        for(thread = 0; thread < thread_count; thread++)
                pthread_join(thread_handles[thread], NULL);

        free(thread_handles);
        return 0;
}

void* Hello(void* rank) {
        long my_rank = (long) rank;
        printf("Hello from thread %ld of %d\n", my_rank, thread_count);
        return NULL;
}herin@hp2000:~/바탕화면$ _
```

입력 값으로 작업 프로세스 개수를 받아와 프로세스를 생성한다. 시작 문구를 출력하고 프로세스가 rank 를 출력하도록 한다.
pthread_join()을 사용해서 분할한 프로세스를 정리한다.



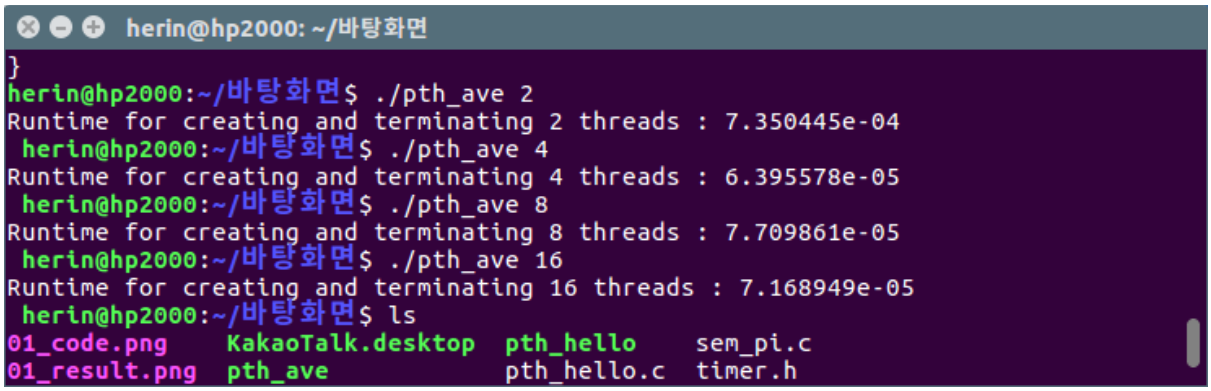
실습 02 스레드 생성, 종료 평균 시간

herin@hp2000: ~/바탕화면

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include "timer.h"

int thread_count;
void* generateFunc(void* rank);

int main(int argc, char* argv[]) {
        long thread;
        double start, end, elapsed;
        pthread_t* thread_handles;

        thread_count = strtol(argv[1], NULL, 10);

        thread_handles = malloc(thread_count * sizeof(pthread_t));

        //start dividing thread
        GET_TIME(start);

        for(thread = 0; thread < thread_count; thread++)
                pthread_create(&thread_handles[thread], NULL, generateF
unc, (void*)thread);

        for(thread = 0; thread < thread_count; thread++)
                pthread_join(thread_handles[thread], NULL);

        GET_TIME(end);

        elapsed = end - start;
        printf("Runtime for creating and terminating %d threads : %e\n
", thread_count, elapsed/thread_count);

        free(thread_handles);
        return 0;
}

void* generateFunc (void* rank) {
        return NULL;
}
herin@hp2000:~/바탕화면$ _
```

작업 프로세스 시작과 종료 시간의 차이 값을 작업 프로세스만큼 나누고 출력한다.

[결과]


herin@hp2000: ~/바탕화면
}
herin@hp2000:~/바탕화면$ ./pth_ave 2
Runtime for creating and terminating 2 threads : 7.350445e-04
herin@hp2000:~/바탕화면$ ./pth_ave 4
Runtime for creating and terminating 4 threads : 6.395578e-05
herin@hp2000:~/바탕화면$ ./pth_ave 8
Runtime for creating and terminating 8 threads : 7.709861e-05
herin@hp2000:~/바탕화면$ ./pth_ave 16
Runtime for creating and terminating 16 threads : 7.168949e-05
herin@hp2000:~/바탕화면$ ls
01_code.png KakaoTalk.desktop pth_hello sem_pi.c
01_result.png pth_ave pth_hello.c timer.h

실습 03 세마포어를 사용한 pi 계산

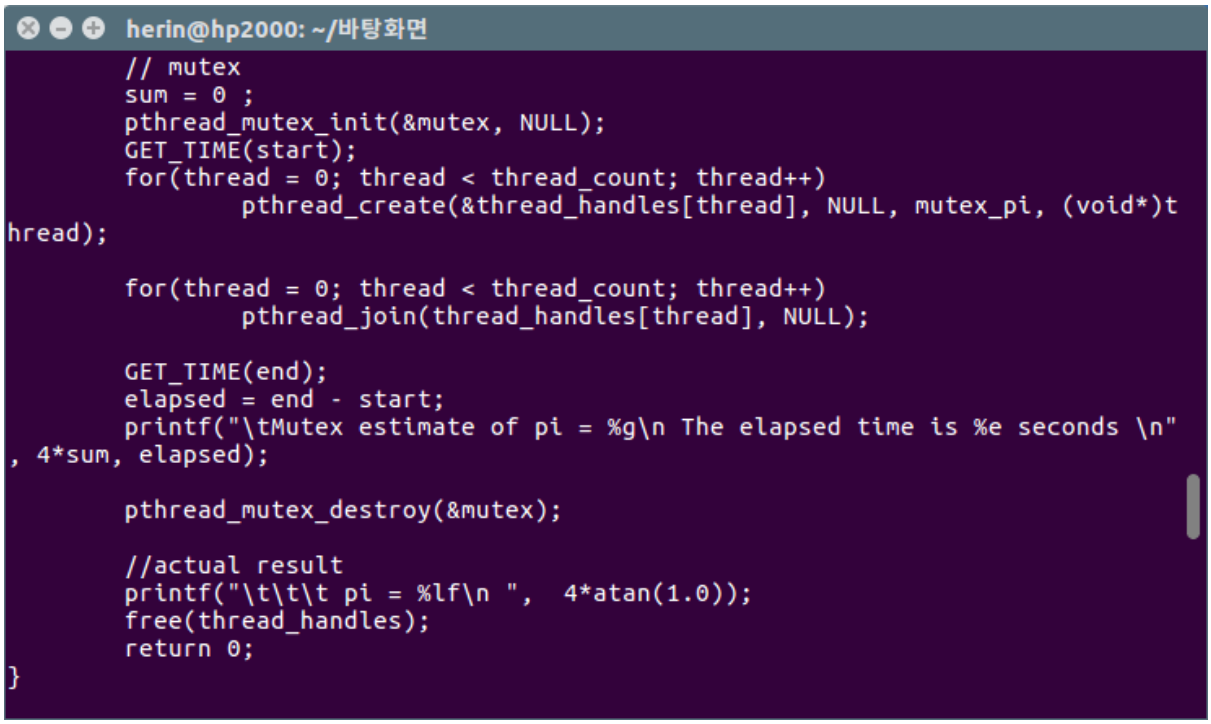
herin@hp2000: ~/바탕화면

```
                        pi = 3.141593
herin@hp2000:~/바탕화면$ cat sem_pi.c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include "timer.h"
#include <semaphore.h>
#include <math.h>

int thread_count;
int counter=0;
void* mutex_pi (void* rank);
void* semaphore_pi (void* rank);
double sum=0;
int n;
pthread_mutex_t mutex;
sem_t sem;

int main(int argc, char* argv[]) {
        long thread;
        double start, end, elapsed;
        pthread_t* thread_handles;

        thread_count = strtol(argv[1], NULL, 10);
```

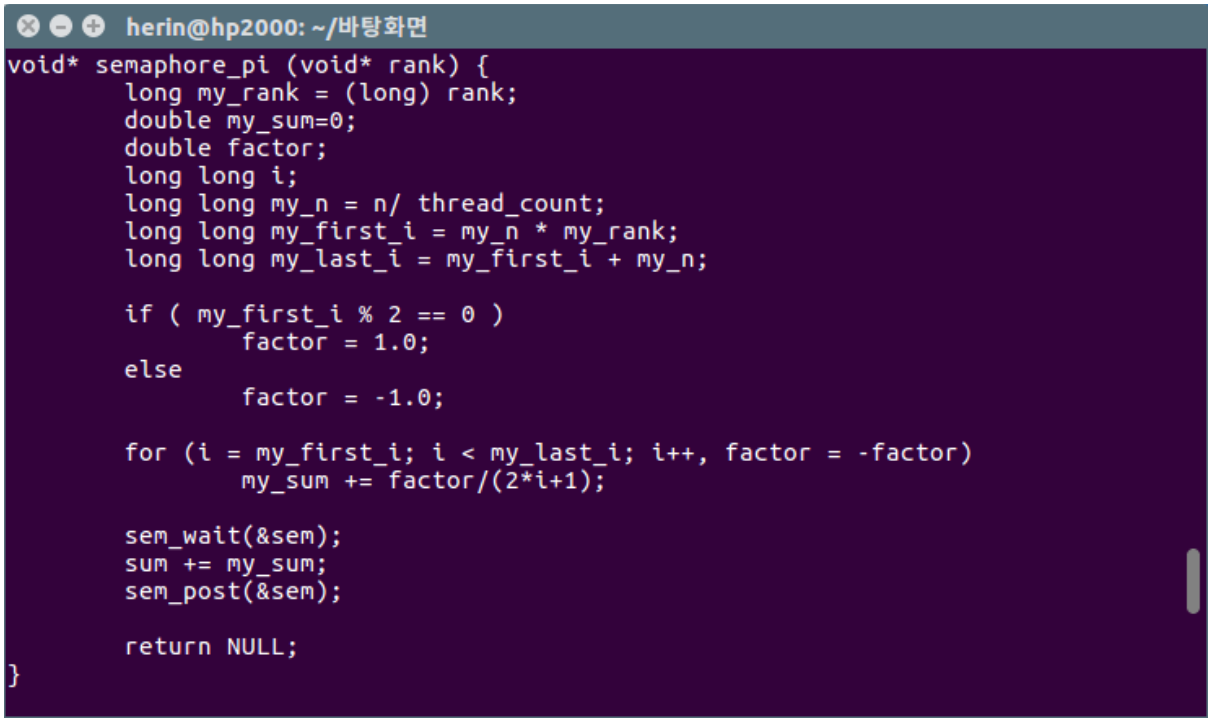
herin@hp2000: ~/바탕화면

```
        pthread_t* thread_handles;

        thread_count = strtol(argv[1], NULL, 10);
        n = strtol(argv[2], NULL, 10);
        printf("With n = %d terms, \n", n);
        thread_handles = malloc(thread_count * sizeof(pthread_t));

        //semaphore
        sem_init(&sem, 0, 1);
        GET_TIME(start);
        for(thread = 0; thread < thread_count; thread++)
                pthread_create(&thread_handles[thread], NULL, semaphore_pi, (voi
d*)thread);

        for(thread = 0; thread < thread_count; thread++)
                pthread_join(thread_handles[thread], NULL);
        GET_TIME(end);

        elapsed = end - start;

        printf("\tSemaphore estimate of pi = %g\n, The elapsed time is %e second
s \n", 4*sum, elapsed);
        sem_destroy(&sem);
```

세마포어를 활용한 예제로, 세마포어를 초기화하고 작업 완료 후 세마포어를 정리한다.



뮤텍스를 사용한 코드로, 뮤텍스를 초기화하고 작업 완료 후 할당환 자원을 정리한다.



sem_wait 을 사용해서 임계영역에 대한 접근을 막는다.

sem_post 를 사용해서 임계영역에 대한 접근을 허용한다.

herin@hp2000: ~/바탕화면

```
void* mutex_pi (void* rank) {
        long my_rank = (long) rank;
        double factor;
        long long i;
        long long my_n = n/ thread_count;
        long long my_first_i = my_n * my_rank;
        long long my_last_i = my_first_i + my_n;
        double my_sum=0;

        if ( my_first_i % 2 == 0 )
                factor = 1.0;
        else
                factor = -1.0;

        for (i = my_first_i; i < my_last_i; i++, factor = -factor) {
                my_sum += factor/(double)(2*i+1);
        }

        pthread_mutex_lock(&mutex);
        sum += my_sum;
        pthread_mutex_unlock(&mutex);

        return NULL;
}
```

pthread_mutex_lock 을 사용해서 임계영역에 대한 접근을 잠근다.
pthread_mutex_unlock 을 사용해서 임계영역에 대한 접근을 허용한다.

[결과]

herin@hp2000: ~/바탕화면

```
herin@hp2000:~/바탕화면$ gcc -g -Wall -o sem_pi sem_pi.c -lm -pthread
herin@hp2000:~/바탕화면$ ./sem_pi 4 2048
With n = 2048 terms,
        Semaphore estimate of pi = 3.1411
, The elapsed time is 3.879070e-04 seconds
        Mutex estimate of pi = 3.1411
 The elapsed time is 2.429485e-04 seconds
                        pi = 3.141593
 herin@hp2000:~/바탕화면$ ./sem_pi 4 8192
With n = 8192 terms,
        Semaphore estimate of pi = 3.14147
, The elapsed time is 4.298687e-04 seconds
        Mutex estimate of pi = 3.14147
 The elapsed time is 3.471375e-04 seconds
                        pi = 3.141593
 herin@hp2000:~/바탕화면$ ./sem_pi 8 8192
With n = 8192 terms,
        Semaphore estimate of pi = 3.14147
, The elapsed time is 6.220341e-04 seconds
        Mutex estimate of pi = 3.14147
 The elapsed time is 4.858971e-04 seconds
                        pi = 3.141593
 herin@hp2000:~/바탕화면$ _
```