프로세스 동기화(도전과제)

20185309
황명원

소스코드:

```c
#include <pthread.h>
#include <semaphore.h>
#include <stdlib.h>
#include <stdio.h>

#define BufferSize 5 // Size of the buffer

sem_t semaphore;
pthread_mutex_t mutex;

int cnt;

void *func(void *cnt_num) {

    for (int i = 0; i < 100000; i++) {
        sem_wait(&semaphore);
        pthread_mutex_lock(&mutex);

        cnt++;
        printf("cnt%d :: %d\n", *((int *)cnt_num),cnt);

        pthread_mutex_unlock(&mutex);
        sem_post(&semaphore);

    }
}
int main()
{
    pthread_mutex_init(&mutex, NULL);
    sem_init(&semaphore,0,BufferSize);

    pthread_t thread[5];
    int iret[5]={1,2,3,4,5};

    for(int i=0 ; i<5 ; i++){
        pthread_create(&thread[i], NULL, (void *)func, (void *)&iret[i]);
    }
    for(int i=0 ; i<5 ; i++){
        pthread_join(thread[i], NULL);
    }

    printf("Thread error: %d\n",500000-cnt);

    exit(0);

    pthread_mutex_destroy(&mutex);
    sem_destroy(&semaphore);

    return 0;
}
```

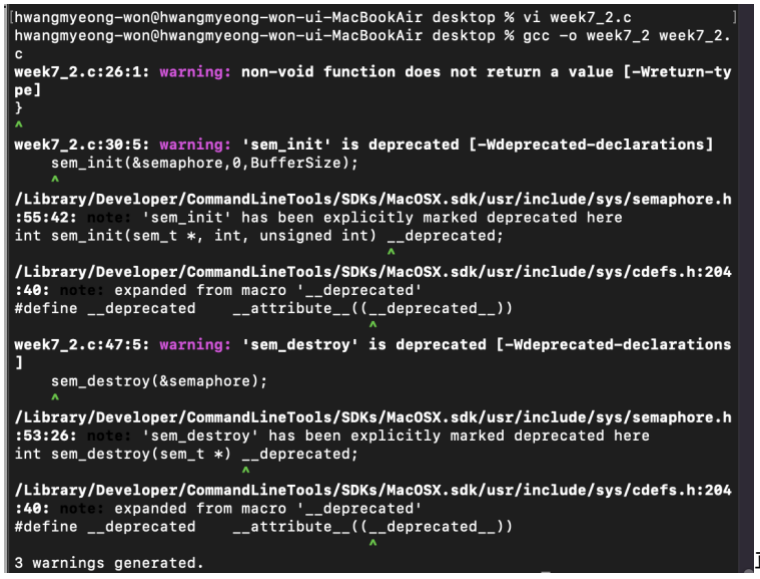
분석내용:

2개의 프로세스를 다룬다면 mutex 만 써도 되지만 2개이상의 프로세스를 다루는 상황으로는 세마포어를 사용하는것이 좋습니다.

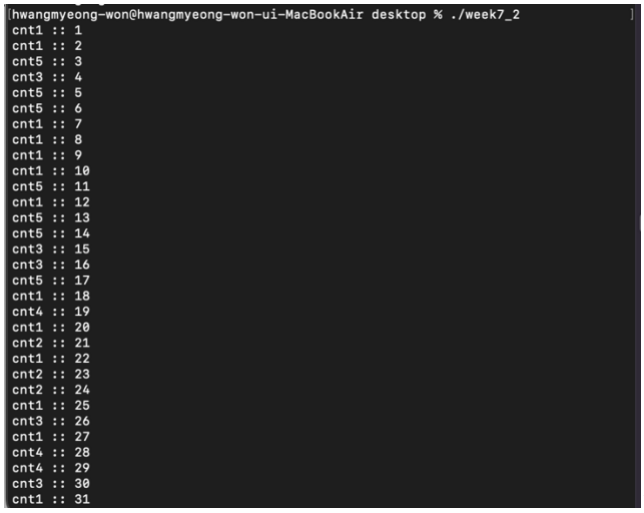
모든 프로세스는 mutex라는 세마포어를 공유하며, mutex가 1로 초기화 됩니다.
wait 연산을 제일 먼저 실행하는 프로세스는 mutex 값이 1이므로 이 값을 0으로 줄이고 임계 구역 내부로 들어갈 수 있습니다. 이 프로세스가 임계 지역을 빠져나오면서 signal 연산을 통해서 mutex값을 0에서 1로 증가시키면 다른 프로세스가 임계 영역 내부로 들어올 수 있습니다.

세마포어를 통해 여러개 프로세스를 다루고 한개의 프로세스가 들어와 실행되는동안 다른 프로세스가 실행 못하게 합니다. 한개의 프로세스가 끝나면 다른 프로세스가 들어와서 실행하고 또 나머지 프로세스는 대기합니다. 이과정을 cnt가 500000이 될때까지 반복합니다.

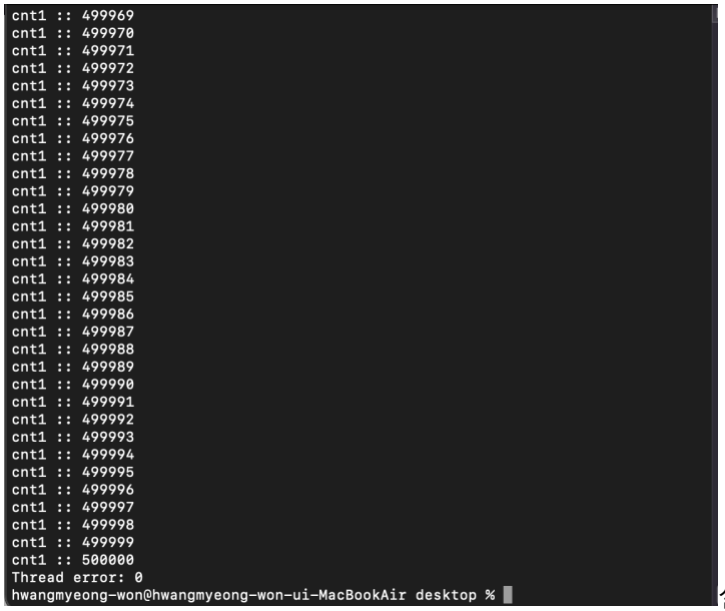
실행결과:



파일 생성


hwangmyeong-won@hwangmyeong-won-ui-MacBookAir desktop % ./week7_2
cnt1 :: 1
cnt1 :: 2
cnt5 :: 3
cnt3 :: 4
cnt5 :: 5
cnt5 :: 6
cnt1 :: 7
cnt1 :: 8
cnt1 :: 9
cnt1 :: 10
cnt5 :: 11
cnt1 :: 12
cnt5 :: 13
cnt5 :: 14
cnt3 :: 15
cnt3 :: 16
cnt5 :: 17
cnt1 :: 18
cnt4 :: 19
cnt1 :: 20
cnt2 :: 21
cnt1 :: 22
cnt2 :: 23
cnt2 :: 24
cnt1 :: 25
cnt3 :: 26
cnt1 :: 27
cnt4 :: 28
cnt4 :: 29
cnt3 :: 30
cnt1 :: 31
실행(1)

(중략)


cnt1 :: 499969
cnt1 :: 499970
cnt1 :: 499971
cnt1 :: 499972
cnt1 :: 499973
cnt1 :: 499974
cnt1 :: 499975
cnt1 :: 499976
cnt1 :: 499977
cnt1 :: 499978
cnt1 :: 499979
cnt1 :: 499980
cnt1 :: 499981
cnt1 :: 499982
cnt1 :: 499983
cnt1 :: 499984
cnt1 :: 499985
cnt1 :: 499986
cnt1 :: 499987
cnt1 :: 499988
cnt1 :: 499989
cnt1 :: 499990
cnt1 :: 499991
cnt1 :: 499992
cnt1 :: 499993
cnt1 :: 499994
cnt1 :: 499995
cnt1 :: 499996
cnt1 :: 499997
cnt1 :: 499998
cnt1 :: 499999
cnt1 :: 500000
Thread error: 0
hwangmyeong-won@hwangmyeong-won-ui-MacBookAir desktop %
실행(2)