**OS Project #2**

**Synchronization Problem에 대한**

**Readme 보고서**

**컴퓨터공학과**

**12193075 이유진**

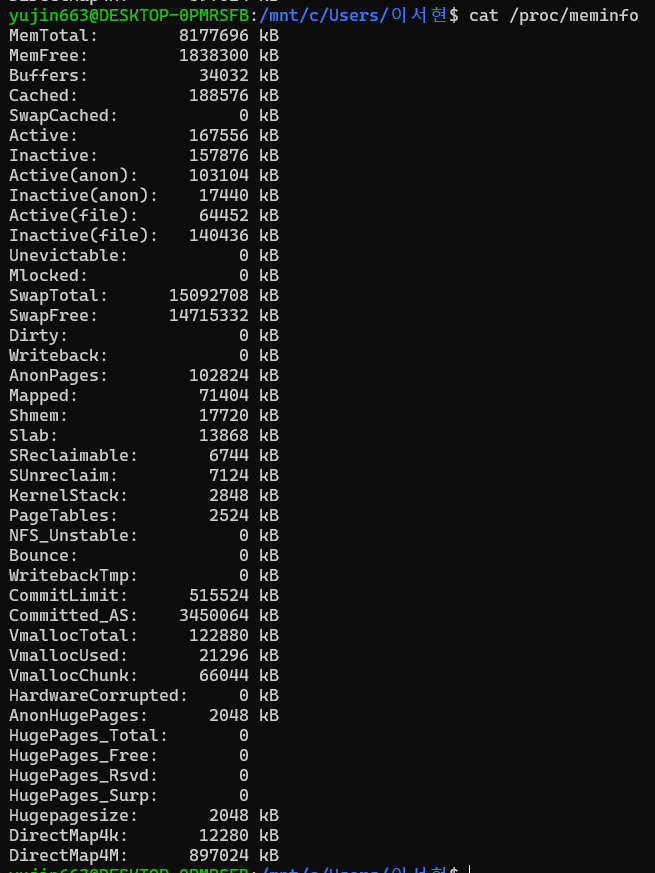
**2021.06.**

**1. 개발환경**

-CPU 정보

 …

-메모리 관련 정보



-운영체제명, 커널 버전(정보), gcc 컴파일 버전, 생성 날짜



**2. 각 코드 구성내용** – (주요 수정 내용을 위주로 설명하겠습니다.)

1)synch\_pthread.c

**-생산자**

void\* producer

(void\* arg) {

int i, value;

for(i=0; i <ITER; i++) {

**임계 영역 (Critical Section)**

:상호배제를 위해 pthread\_mutax\_lock / unlock을 사용하고 크기가 30인 버퍼가 가득 차서 더 이상 put할 수 없는 경우 따로 신호를 준다.

**pthread\_mutex\_lock(&m);**

value = count;

printf("%u: %d\n", (unsigned int)pthread\_self(), value);

while(count == MAX)

**pthread\_cond\_wait(&empty, &m);**

put(i);

**pthread\_cond\_signal(&fill);**

**pthread\_mutex\_unlock(&m);**

} }

void put(int value) {

버퍼에 value를 삽입하고 count는 1을 더한다.

buffer[fill\_cnt] = value;

fill\_cnt = (fill\_cnt + 1) % MAX;

count++; }

-**소비자**

void\* consumer(void\* arg) {

int value; int i;

for(i=0; i <ITER; i++) {

**pthread\_mutex\_lock(&m);**

**임계 영역 (Critical Section)**

:상호 배제를 위해 pthread\_mutax\_lock / unlock을 사용하고 크기가 30인 버퍼가 비어서 더 이상 get할 수 없는 경우 따로 신호를 준다.

value = count;

printf("%u: %d\n", (unsigned int)pthread\_self(), value);

while (count == 0)

**pthread\_cond\_wait(&fill, &m);**

value = get();

**pthread\_cond\_signal(&empty);**

**pthread\_mutex\_unlock(&m);**

}

int get() {

버퍼에 value를 빼내고 count는 1을 뺀다.

int tmp = buffer[use\_cnt];

use\_cnt = (use\_cnt + 1) % MAX;

count--;

return tmp; }

2) synch\_semaphore.c

-**생산자**

void\* producer(void\* arg) {

int i; int value;

for (i=0; i < ITER ; i++) {

**sem\_wait(&empty);**

**임계 영역 (Critical Section)**

:상호 배제를 위해 sem\_wait / post을 사용하고 크기가 30인 버퍼가 가득 차서 더 이상 put할 수 없는 경우 따로 신호를 준다.

**sem\_wait(&s);**

value = count;

printf("%u: %d\n",(unsigned int) pthread\_self(), value);

put(value);

**sem\_post(&s);**

**sem\_post(&fill);**

}

pthread\_exit(NULL);

}

void put(int value) {

buffer[fill\_cnt] = value;

fill\_cnt = (fill\_cnt + 1) % MAX;

count++; }

**-소비자**

void\* consumer(void\* arg) {

int i; int value;

for (i=0; i<ITER; i++){

**sem\_wait(&fill);**

**임계 영역 (Critical Section)**

:상호 배제를 위해 sem\_wait / post을 사용하고 버퍼가 비어서 더 이상 get할 수 없는 경우 따로 신호를 준다.

**sem\_wait(&s);**

value = count;

printf("%u: %d\n", (unsigned int) pthread\_self(), value);

get();

**sem\_post(&s);**

**sem\_post(&empty);**

}

pthread\_exit(NULL); }

int get() {

int tmp = buffer[use\_cnt];

use\_cnt = (use\_cnt + 1) %MAX;

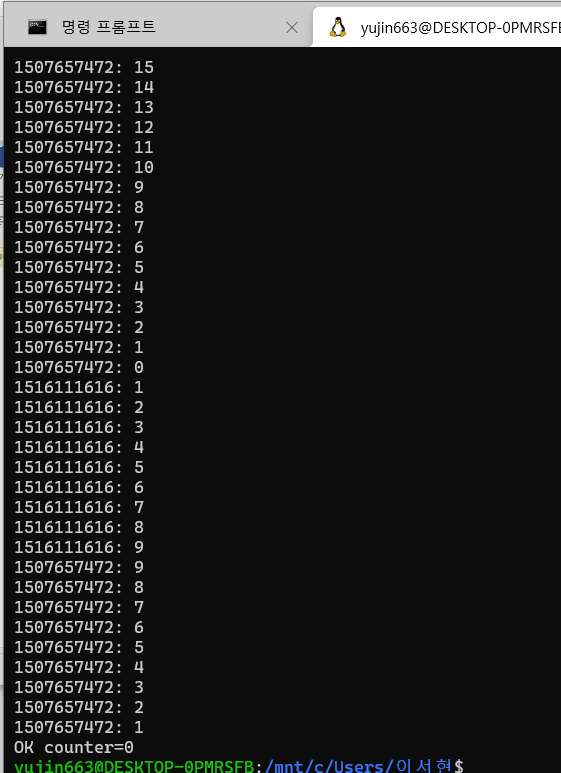
count--;

return tmp;

}

**3. 결과**

1) synch\_pthread.c

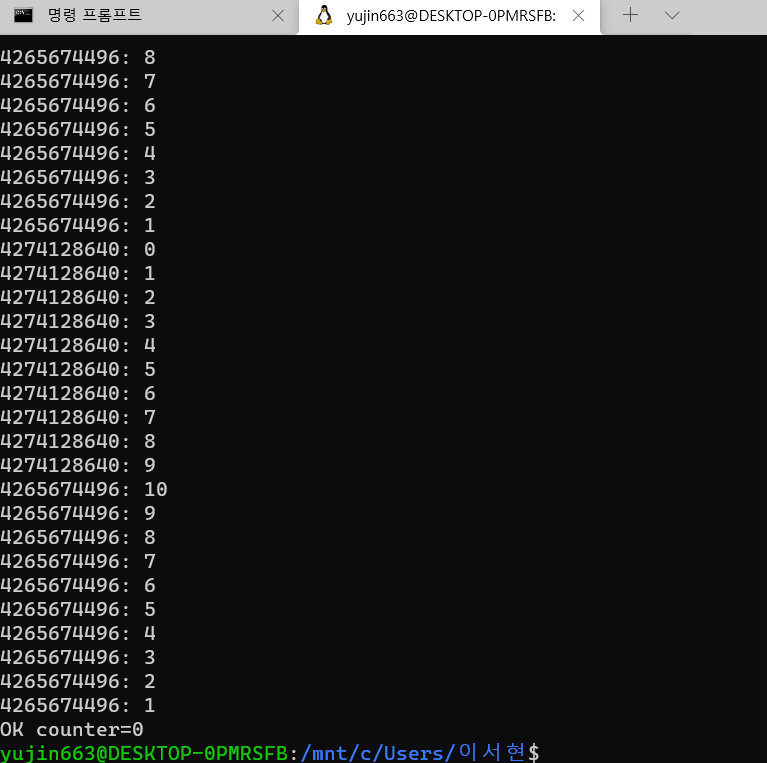


임계영역에서 상호배제를 했기 때문에

중간에 interleaving되지 않고 Correct result를 얻었다.

버퍼의 크기가 30이므로 0~30의 수가 순차적으로 출력되고 총 반복 횟수가 1000번이므로 마지막 경우에는 10번만 출력된다.

2)synch\_semaphore.c



마찬가지로, 임계영역에서 상호배제를 했기 때문에 Correct result를 얻었다.

버퍼의 크기가 30이므로 0~30의 수가 순차적으로 출력되고 총 반복 횟수가 1000번이므로 마지막 경우에는 10번만 출력된다.

**4. makefile**

