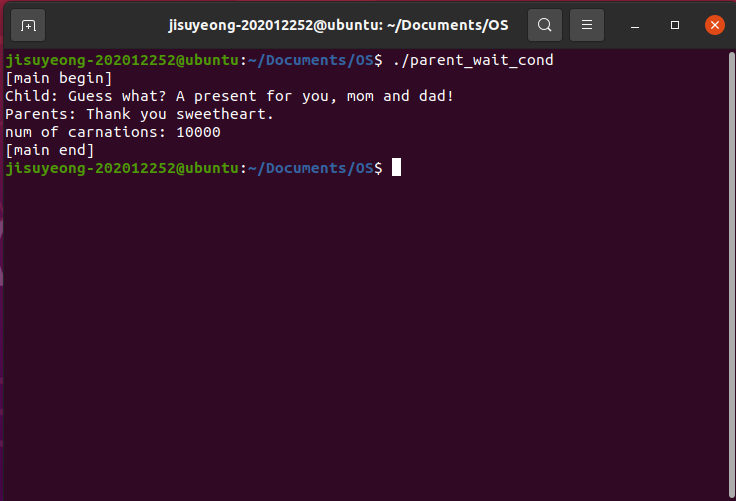
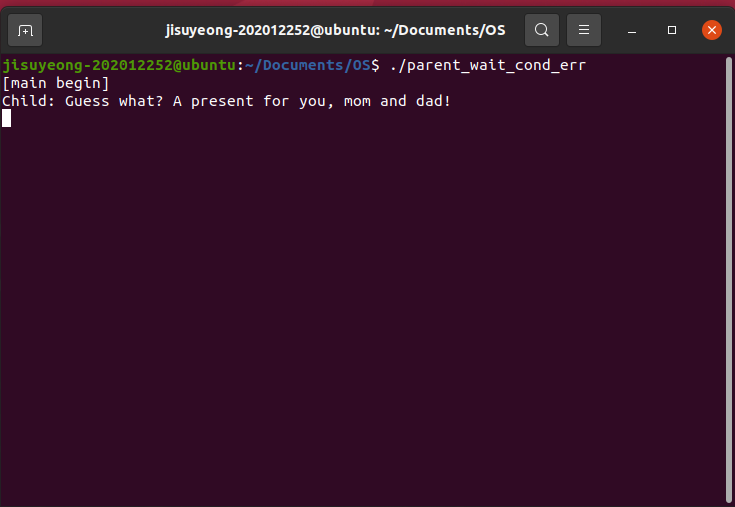
**문제 1.**

**[parent\_wait\_cond.c 실행결과]**



**[parent\_wait\_cond\_err.c 실행결과]**

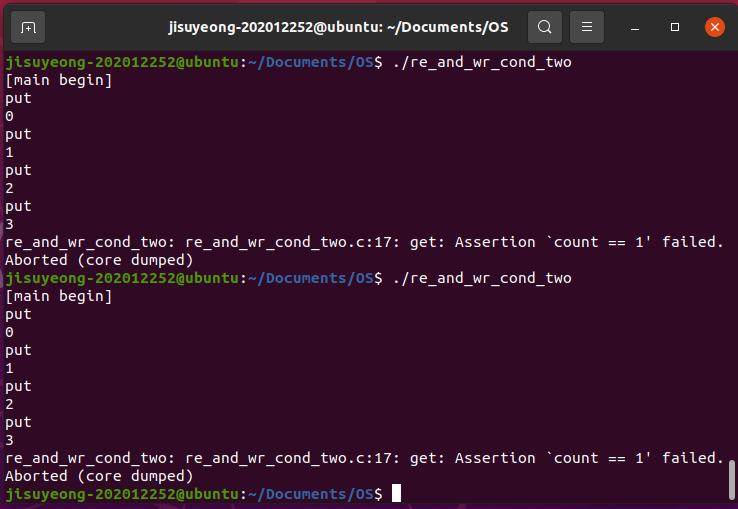


**[문제가 발생하는 이유]**

자식 스레드가 바로 실행되어 thr\_exit()를 호출하는 경우 자식 스레드에서 signal을 보내도 lock 상태의 스레드가 존재하지 않아 그냥 끝나게 된다. 그 뒤에 부모 스레드가 실행되어 wait를 호출했을 때 sleep 상태로 넘어가면 아무도 부모 스레드를 깨울 수 없게 된다.

**문제2.**

[re\_and\_wr\_cond\_two.c 실행 결과]



[문제가 발생하는 이유]

버퍼가 비어있는 경우에 소비자가 get()을 호출해 데이터를 사용하려 했기 때문에 assert에 의해 실행이 중단되었다..

**문제3.**

**[re\_and\_wr\_cond\_final.c 소스코드]**

#include<stdio.h>

#include<assert.h>

#include<pthread.h>

int buffer;

int count = 0;

pthread\_mutex\_t mutex = PTHREAD\_MUTEX\_INITIALIZER;

pthread\_cond\_t empty = PTHREAD\_COND\_INITIALIZER;

pthread\_cond\_t fill = PTHREAD\_COND\_INITIALIZER;

void put(int value){

assert(count == 0);

count = 1;

buffer = value;

}

int get(){

assert(count == 1);

count = 0;

return buffer;

}

void \*producer(void \*arg){

int i;

int loops = (int) arg;

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

pthread\_mutex\_lock(&mutex);

while(count == 1){

pthread\_cond\_wait(&empty, &mutex);

}

printf("put \n");

put(i);

pthread\_cond\_signal(&fill);

pthread\_mutex\_unlock(&mutex);

}

}

void \*consumer(void \*arg){

int i;

int loops = (int)arg;

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

pthread\_mutex\_lock(&mutex);

while(count == 0){

pthread\_cond\_wait(&fill, &mutex);

}

int tmp = get();

pthread\_cond\_signal(&empty);

pthread\_mutex\_unlock(&mutex);

printf("%d \n", tmp);

}

}

int main(int argc, char \* argv[]){

pthread\_t p, c1, c2;

printf("[main begin] \n");

pthread\_create(&p, NULL, producer, 10);

pthread\_create(&c1, NULL, consumer, 5);

pthread\_create(&c2, NULL, consumer, 5);

pthread\_join(p, NULL);

pthread\_join(c1, NULL);

pthread\_join(c2, NULL);

printf("[main end] \n");

return 0;

}

**[re\_and\_wr\_cond\_final.c 실행 결과]**

