

## Operation System – COSE341

### 실습 2 과제 레포트 – IPC – Assignment 1\_2

고려대학교 컴퓨터학과 2017320108

고재영

개발 환경 : Oracle VM VirtualBox, Linux, Ubuntu 18.04.5, Visual Studio 2019

과제 만기일 : 2022/05/09 10:29 AM

유의 사항 : Visual studio 2019로 작성된 .c 파일의 produce와 consume 영역만 발췌한 스크린샷과 작성된 코드와 semaphore에 대한 설명이 수록되어 있다.

제출 날짜 : 2022.05.07

#### [0] - 과제 1\_2. Semaphore

● 목표: 세마포어 사용하기

assignment 1-1에서 구현한 코드의 일부를 복사하여 사용

1. 아래 기능을 구현할 것

assignment\_1\_2/message\_buffer\_semaphore.c

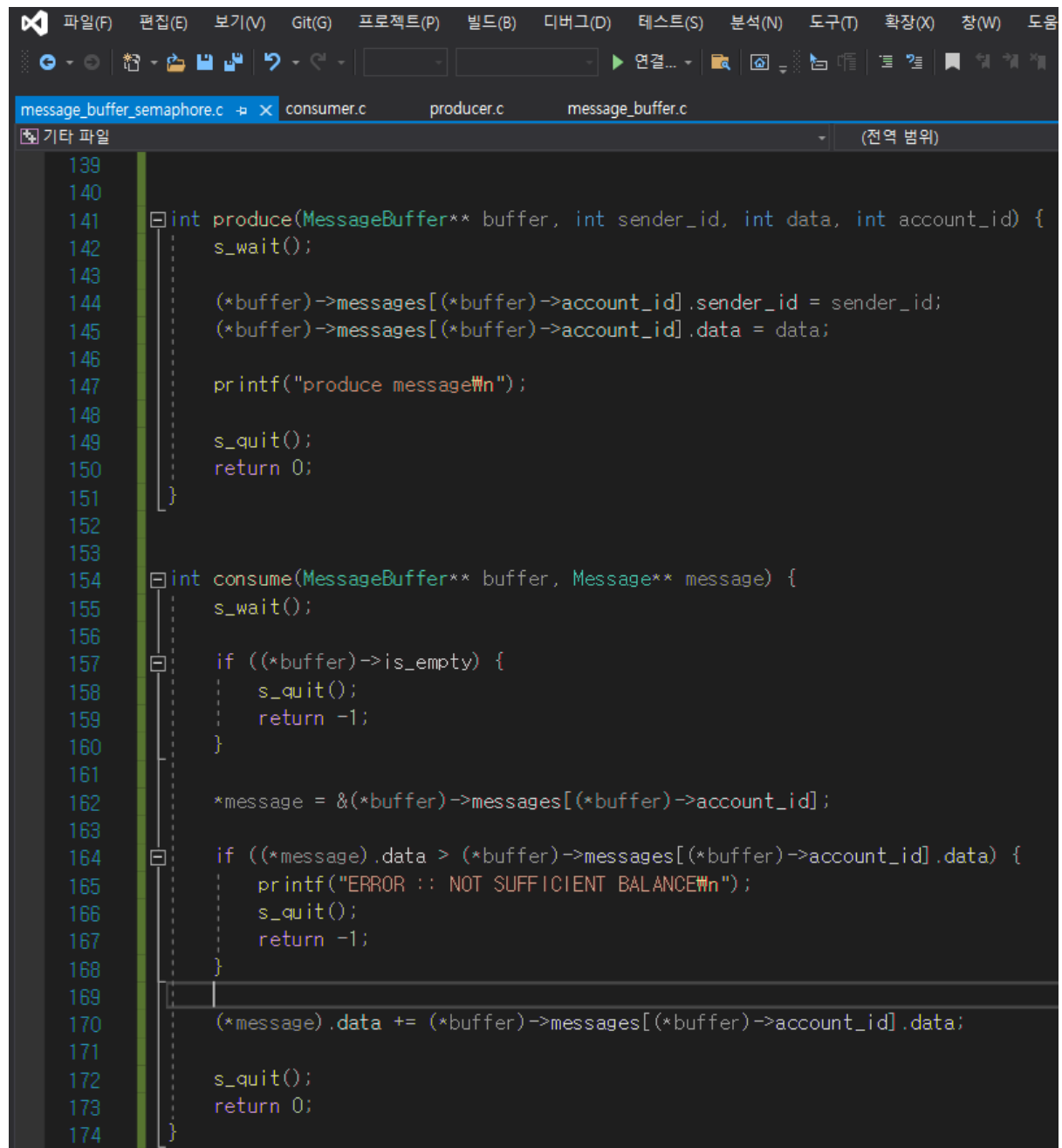
- init semaphore

- destroy semaphore

- use s\_quit() and s\_wait()

2. 특정 작업이 원자적으로 수행되도록 s\_quit(), s\_wait()를 사용하고, 사용되어야 하는 위치와 이유를 작성하여 PDF로 제출

## [1] message\_buffer\_semaphore.c 코드 스크린샷



```
139
140
141 int produce(MessageBuffer** buffer, int sender_id, int data, int account_id) {
142     s_wait();
143
144     (*buffer)->messages[(*buffer)->account_id].sender_id = sender_id;
145     (*buffer)->messages[(*buffer)->account_id].data = data;
146
147     printf("produce message#n");
148
149     s_quit();
150     return 0;
151 }
152
153
154 int consume(MessageBuffer** buffer, Message** message) {
155     s_wait();
156
157     if ((*buffer)->is_empty) {
158         s_quit();
159         return -1;
160     }
161
162     *message = &(*buffer)->messages[(*buffer)->account_id];
163
164     if ((*message).data > (*buffer)->messages[(*buffer)->account_id].data) {
165         printf("ERROR :: NOT SUFFICIENT BALANCE#n");
166         s_quit();
167         return -1;
168     }
169
170     (*message).data += (*buffer)->messages[(*buffer)->account_id].data;
171
172     s_quit();
173     return 0;
174 }
```

## [2] Semaphore 구현 - s\_wait()과 s\_quit()의 설명

Assignment 1\_2에서 s\_wait()과 s\_quit()를 사용하는 것은 두 개의 프로세스 producer와 consumer에 대해서 하나의 shared memory인 buffer에 공동으로 사용하는 데, 동시에 접근하게 되어 race condition이 발생하는 일을 방지하기 위함으로 semaphore를 사용하는 것이다. Race condition 발생하며 동시에 접근할 경우 여러 가지 상황의 consistency 문제를 야기하기 때문에 이를 막도록 일종의 permission을 주고 반납하게 하듯 semaphore를 sequence control의 일환으로 사용한다.

따라서, producer든 consumer든 코드 영역에서 buffer에 접근하기 전에 먼저 permission을 받기 위해 s\_wait()로 권한을 부여받는다. 그렇게 permission을 받은 후 buffer에 관하여 수행할 일이 모두 끝나면 s\_quit()을 통해 permission을 release한다. 특히나, consume 에서는 buffer가 empty이거나, 현재 account에 있는 금액(data)이 출금을 요청한 금액(data)보다 적어서 출금이 불가능하여 error를 리턴하는 경우도 프로세스 수행이 끝나기 때문에 return하기 전에 s\_quit()을 호출하여 permission release하는 것을 알 수 있다.