COSE341 2018320182 김상언

```
int produce(MessageBuffer **buffer, int sender_id, int data, int ac

    if((account_id < 0) || (account_id > BUFFER_SIZE))
    {
        return -1;
    }
    else if((*buffer)->messages[account_id].data + data < 0)
    {
        return -1;
    }
    else
    {
    s_wait();

        (*buffer)->account_id = account_id;
        (*buffer)->messages[account_id].data = data;
        (*buffer)->messages[account_id].sender_id = sender_id;

        (*buffer)->is_empty--;

    s_quit();
```

```
int consume(MessageBuffer **buffer, Message **message) {
    if((*buffer)->is_empty)
    return -1;

    s_wait();

    (*message)->data = (*buffer)->messages[(*buffer)->account_id].data;
    (*message)->sender_id = (*buffer)->messages[(*buffer)->account_id].sender_id;
    (*buffer)->is_empty++;

    s_quit();
    return 0;
```

Produce와 Consume 함수에서 공통적으로 접근하는 변수는 버퍼 변수이다. 한쪽에서 값을 넣어주면 한쪽에서 그 값을 이용하는 방식이다. Race condition을 없애기 위해서 세마포어를 사용하는데, 버퍼와 같이 공유되는 변수의 경우 동시에 접근이 이루어지면 어떤 프로세스에서 수정된 데이터가 언제 변경되었는지 모르게 된다. 이를 막기 위해 wait와 quit을 버퍼를 이용하는 부분에 두어 이런 현상을 막는다.