운영체제 실습2 / 2018320212 김상엽

1-2. semaphore

1. init_buffer / attach_buffer에 init_sem()을 추가 -> semaphore를 사용할 수 있게끔.

```
75  int init_buffer(MessageBuffer **buffer) {
76      /*---------------------------------------*/
77      /* TODO 1 : init buffer                  */
78
79      if((shmid = shmget(SHM_KEY, sizeof(MessageBuffer), IPC_CREAT|0666)) == -1) return -1;
80      if((memory_segment = shmat(shmid, NULL, 0)) == (void*)-1) return -1;
81      (*buffer) = (MessageBuffer*) memory_segment;
82      (*buffer)->is_empty = 1;
83      init_sem();
84
85      /* TODO 1 : END                          */
86      /*---------------------------------------*/
87
88      printf("init buffer\n");
89      return 0;
90  }
92  int attach_buffer(MessageBuffer **buffer) {
93      /*---------------------------------------*/
94      /* TODO 2 : attach buffer                */
95      /* do not consider "no buffer situation" */
96
97      if((shmid = shmget(SHM_KEY, sizeof(MessageBuffer), IPC_CREAT|0666)) == -1) return -1;
98      if((memory_segment = shmat(shmid, NULL, 0)) == (void*)-1) return -1;
99      (*buffer) = (MessageBuffer*) memory_segment;
100     init_sem();
101
102     /* TODO 2 : END                          */
103     /*---------------------------------------*/
104
105     printf("attach buffer\n");
106     printf("\n");
107     return 0;
108 }
```

2. destroy_buffer에 destroy_sem() 추가, shared memory 제거 시 semaphore도 제거 할 수 있도록 함.

```
120 int destroy_buffer() {
121     if(shmctl(shmid, IPC_RMID, NULL) == -1) {
122         printf("shmctl error!\n\n");
123         return -1;
124     }
125     destroy_sem();
126     printf("destroy shared_memory\n\n");
127     return 0;
128 }
```

3. produce에서도 semaphore를 이용할 수 있도록 semaphore의 id를 얻는 것과 s_wait()를 먼저 수행하도록 추가하였음. 이후 작업이 다 될 때, 즉 return을 하기 직전에 s_quit()을 실행하도록 해, produce라는 작업이 원자적으로 수행되도록 함.

```c
int produce(MessageBuffer **buffer, int sender_id, int data, int account_id) {

    /*----------------------------------------*/
    /* TODO 3 : produce message               */
    if ((semid = semget(SEM_KEY, 1, IPC_CREAT|IPC_EXCL|0666)) == -1) {
        if ((semid = semget(SEM_KEY, 0, 0)) == -1) return -1;
    }
    s_wait();

    if((*buffer)->messages[account_id].data + data < 0){
        printf("not sufficient\n");
        s_quit();
        return -1;
    }

    (*buffer)->messages[account_id].sender_id = sender_id;
    (*buffer)->messages[account_id].data += data;
    (*buffer)->is_empty = 0;
    (*buffer)->account_id = account_id;

    /* TODO 3 : END                           */
    /*----------------------------------------*/
    printf("produce message\n");
    s_quit();
    return 0;
}
```

4. consume도 마찬가지로, consume이 실행되자마자 semid를 얻고 s_wait, 작업을 마치고 return 되기 전에 s_quit()을 하여 작업의 원자성을 보장하였음.

```c
157  int consume(MessageBuffer **buffer, Message **message) {
158
159      if ((semid = semget(SEM_KEY, 1, IPC_CREAT|IPC_EXCL|0666)) == -1) {
160          if ((semid = semget(SEM_KEY, 0, 0)) == -1) return -1;
161      }
162      s_wait();
163      if((*buffer)->is_empty){
164          s_quit();
165          return -1;
166      }
167      /*----------------------------------------*/
168      /* TODO 4 : consume message               */
169      *message = &(*buffer)->messages[(*buffer)->account_id];
170      (*buffer)->is_empty = 1;
171      /* TODO 4 : END                           */
172      /*----------------------------------------*/
173      s_quit();
174      return 0;
175  }
176
```