

운영체제 과제2

운영체제 월목B
정보컴퓨터공학과
201120883
김현재

1. 동작 시험 결과

스크린샷 및 설명

```
hyunjae@hyunjae-U2442: ~/바탕화면/052
hyunjae@hyunjae-U2442:~/바탕화면/052$ gcc -o counter_no_sem counter_no_sem.c -lpthread
hyunjae@hyunjae-U2442:~/바탕화면/052$ ./counter_no_sem
Thread 140458671666944 is Created.
Thread 140458663274240 is Created.
Thread 140458654881536 is Created.
Thread 140458671666944 count : 1 .
Thread 140458663274240 count : 2 .
Thread 140458654881536 count : 3 .
Thread 140458663274240 count : 4 .
Thread 140458671666944 count : 4 .
Thread 140458654881536 count : 5 .
Thread 140458663274240 count : 6 .
Thread 140458654881536 count : 8 .
Thread 140458671666944 count : 7 .
Thread 140458663274240 count : 9 .
Thread 140458654881536 count : 10 .
Thread 140458671666944 count : 11 .
Thread 140458663274240 count : 12 .
Thread 140458671666944 count : 14 .
Thread 140458654881536 count : 13 .
Thread 140458663274240 count : 15 .
Thread 140458671666944 count : 15 .
Thread 140458654881536 count : 16 .
^C
hyunjae@hyunjae-U2442:~/바탕화면/052$
```

세마포어를 사용하지 않은 카운트 프로그램이다.

3개의 pthread를 이용하여 카운트 하게 하였고 세마포어를 사용하지 않았기 때문에 중간에 실행값 중간에 4가 두번 카운트 되는 장면을 볼수 있고 아래에 15가 2번 카운트되는 장면 등을 볼 수 있다.

```
hyunjae@hyunjae-U2442: ~/바탕화면/052
hyunjae@hyunjae-U2442:~/바탕화면/052$ gcc -o counter_sem counter_sem.c -lpthread
hyunjae@hyunjae-U2442:~/바탕화면/052$ ./counter_sem
Thread 140220390729472 is Created.
Thread 140220399122176 is Created.
Thread 140220382336768 is Created.
Thread 140220390729472 count : 1 .
Thread 140220399122176 count : 2 .
Thread 140220382336768 count : 3 .
Thread 140220390729472 count : 4 .
Thread 140220399122176 count : 5 .
Thread 140220382336768 count : 6 .
Thread 140220390729472 count : 7 .
Thread 140220399122176 count : 8 .
Thread 140220382336768 count : 9 .
Thread 140220390729472 count : 10 .
Thread 140220399122176 count : 11 .
Thread 140220382336768 count : 12 .
Thread 140220390729472 count : 13 .
Thread 140220399122176 count : 14 .
Thread 140220382336768 count : 15 .
Thread 140220390729472 count : 16 .
Thread 140220399122176 count : 17 .
Thread 140220382336768 count : 18 .
Thread 140220390729472 count : 19 .
^C
hyunjae@hyunjae-U2442:~/바탕화면/052$
```

세마포어를 이용하여 동기화 기능을 추가한 프로그램이다.

세마포어를 사용하여 동기화 하였기 때문에 모든 쓰레드가 한번에 하나씩 임계영역으로 들어가서 처리하였기 때문에 결과 값이 정상적으로 나오게 되었다.

스크린샷 및 설명

```

x - □ hyunjae@hyunjae-U2442: ~/바탕화면/052
hyunjae@hyunjae-U2442:~/바탕화면/052$ gcc -o PC_no_sem PC_no_sem.c -lpthread
hyunjae@hyunjae-U2442:~/바탕화면/052$ ./PC_no_sem
< Consumer > counter : -1 buffer[0] : 0
< Producer > counter : 1 buffer[0] : 0
< Producer > counter : 1 buffer[0] : 0
< Consumer > counter : -1 buffer[0] : 0
< Consumer > counter : 0 buffer[2] : 0
< Consumer > counter : 0 buffer[2] : 1
< Producer > counter : 2 buffer[2] : 1
< Producer > counter : 3 buffer[2] : 1
< Producer > counter : 2 buffer[4] : 2
< Producer > counter : 3 buffer[4] : 2
< Consumer > counter : -1 buffer[4] : 0
< Consumer > counter : -1 buffer[4] : 2
< Producer > counter : 2 buffer[6] : 3
< Consumer > counter : 0 buffer[6] : 3
< Producer > counter : 3 buffer[6] : 3
< Consumer > counter : -1 buffer[6] : 3
< Producer > counter : 2 buffer[8] : 4
< Consumer > counter : -1 buffer[8] : 4
< Producer > counter : 2 buffer[9] : 4
< Consumer > counter : -1 buffer[9] : 4
< Producer > counter : 2 buffer[0] : 5
< Producer > counter : 2 buffer[1] : 5
< Consumer > counter : -1 buffer[0] : 5
< Consumer > counter : -1 buffer[0] : 5
< Producer > counter : 2 buffer[2] : 6
< Producer > counter : 3 buffer[3] : 6
< Consumer > counter : 0 buffer[2] : 6
< Consumer > counter : -1 buffer[3] : 6
^C
hyunjae@hyunjae-U2442:~/바탕화면/052$

```

세마포어를 사용하지 않은 생성자/소비자 프로그램이다.

2개의 생산자 pthread와 2개의 소비자 pthread를 이용하여 카운트동작과 버퍼를 채우고 소비시키는 동작을 구현하였고 이 동작은 카운터 차례대로 내려갔다가 올라가야지만 동기화가 맞다는 것을 볼 수 있는데 위 스크린샷에 보면 버퍼를 채우는 동작과 소비하는 동작 그리고 카운터를 하는 동작 모두가 동기화되지 않았다는 것을 볼 수 있다.

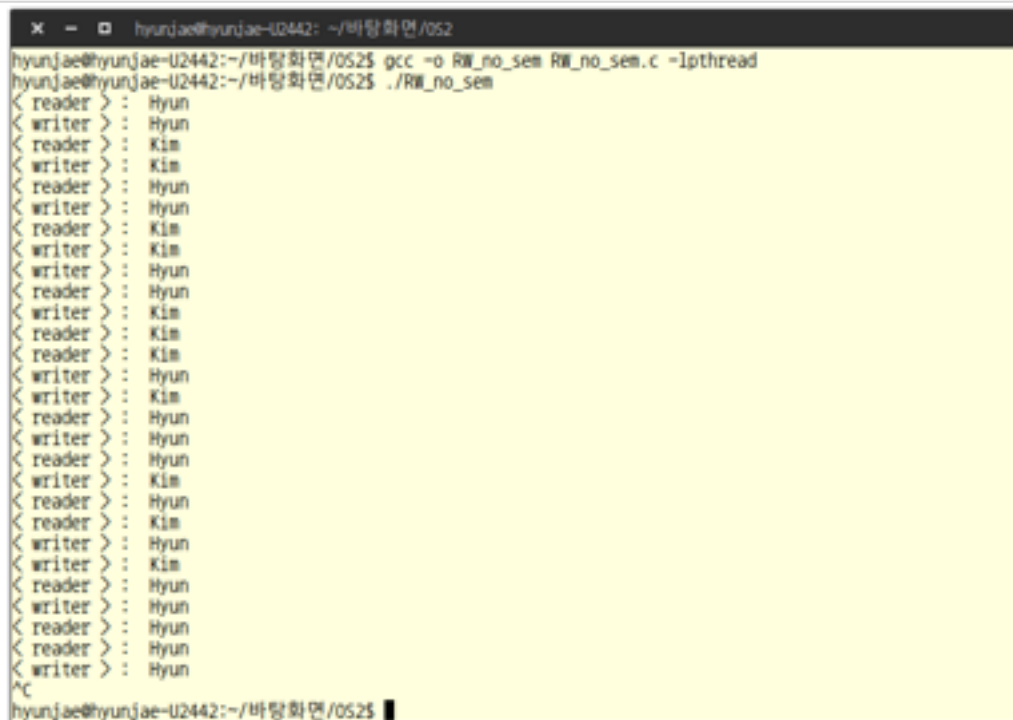
```

x - □ hyunjae@hyunjae-U2442: ~/바탕화면/052
hyunjae@hyunjae-U2442:~/바탕화면/052$ gcc -o PC_sem PC_sem.c -lpthread
hyunjae@hyunjae-U2442:~/바탕화면/052$ ./PC_sem
< Producer > counter : 0 buffer[0] : 0
< Producer > counter : 1 buffer[1] : 1
< Producer > counter : 2 buffer[2] : 2
< Producer > counter : 3 buffer[3] : 3
< Producer > counter : 4 buffer[4] : 4
< Producer > counter : 5 buffer[5] : 5
< Producer > counter : 6 buffer[6] : 6
< Producer > counter : 7 buffer[7] : 7
< Producer > counter : 8 buffer[8] : 8
< Producer > counter : 9 buffer[9] : 9
< Consumer > counter : 10 buffer[0] : 0
< Consumer > counter : 9 buffer[1] : 1
< Consumer > counter : 8 buffer[2] : 2
< Consumer > counter : 7 buffer[3] : 3
< Consumer > counter : 6 buffer[4] : 4
< Consumer > counter : 5 buffer[5] : 5
< Consumer > counter : 4 buffer[6] : 6
< Consumer > counter : 3 buffer[7] : 7
< Consumer > counter : 2 buffer[8] : 8
< Consumer > counter : 1 buffer[9] : 9
< Producer > counter : 0 buffer[0] : 9
< Producer > counter : 1 buffer[1] : 10
< Producer > counter : 2 buffer[2] : 11
< Producer > counter : 3 buffer[3] : 12
< Producer > counter : 4 buffer[4] : 13
< Producer > counter : 5 buffer[5] : 14
< Producer > counter : 6 buffer[6] : 15
< Producer > counter : 7 buffer[7] : 16
^C
hyunjae@hyunjae-U2442:~/바탕화면/052$

```

스크린샷 및 설명

세마포어를 사용한 생산자 / 소비자 프로그램이다. 세마포어를 사용했기 때문에 동기화가 맞아 떨어지므로 카운터가 차례대로 증가(생산자가 생산한것을 저장할시 증가한다.) 그리고 카운터가 차례대로 감소(소비자가 소비할 때 카운터가 감소하게 된다.) 하고 버퍼안에 값이 생성이되어야만 소비자가 접근을 하여 소비한다는 것을 볼 수 있다.



```
hyunjae@hyunjae-U2442: ~/바탕화면/052
hyunjae@hyunjae-U2442:~/바탕화면/052$ gcc -o RW_no_sem RW_no_sem.c -lpthread
hyunjae@hyunjae-U2442:~/바탕화면/052$ ./RW_no_sem
< reader > : Hyun
< writer > : Hyun
< reader > : Kim
< writer > : Kim
< reader > : Hyun
< writer > : Hyun
< reader > : Kim
< writer > : Kim
< writer > : Hyun
< reader > : Hyun
< writer > : Kim
< reader > : Kim
< reader > : Kim
< writer > : Hyun
< writer > : Kim
< reader > : Hyun
< writer > : Hyun
< reader > : Hyun
< writer > : Kim
< reader > : Hyun
< reader > : Kim
< writer > : Hyun
< writer > : Kim
< reader > : Hyun
< reader > : Hyun
< reader > : Hyun
< writer > : Hyun
^C
hyunjae@hyunjae-U2442:~/바탕화면/052$
```

세마포어를 사용하지 않은 Writer/Reader 프로그램이다. 세마포어를 사용하지 않았기 때문에 동기화가 맞지 않고 그렇기 때문에 Writer가 Hyun이라는 단어를 쓰기도 전에 읽어오게 되고 Writer가 Kim을 썼는데도 그 전에 있던 Hyun을 읽어오게 되는 문제점이 발생하게 된다.

```

x - □ hyunjae@hyunjae-U2442: ~/다운로드
hyunjae@hyunjae-U2442:~/다운로드$ ./rw
< writer 2 > : Hyun
< writer 1 > : Kim
< reader > : Kim
< reader > : Kim
< writer 1 > : Kim
< writer 2 > : Hyun
< reader > : Hyun
< reader > : Hyun
< writer 1 > : Kim
< writer 2 > : Hyun
< reader > : Hyun
< reader > : Hyun
< writer 2 > : Hyun
< writer 1 > : Kim
< reader > : Kim
< reader > : Kim
< writer 1 > : Kim
< reader > : Kim
< reader > : Kim
< writer 2 > : Hyun
< writer 1 > : Kim
< reader > : Kim
< reader > : Kim
< writer 2 > : Hyun
^C
hyunjae@hyunjae-U2442:~/다운로드$ █

```

세마포어를 사용한 Writer / Reader 프로그램이다. 세마포어를 사용했기 때문에 Writer가 글씨를 쓰고 나서 Reader가 그 글을 읽으며 최신으로 쓴 글을 읽게 된다. 두개의 리더와 라이트 pthread가 있으며 세마포어를 활용하였기 때문에 동기화가 이루어 지게 된다.

2. 토의 및 느낀점

안타깝게도 4번 식사하는 철학자들의 문제는 풀지 못하였다. 요번 프로그램을 짜게 되면서 새롭게 pthread를 활용하는 방법에 대해서 배울 수 있었다. 특히 요번에는 세마포어를 이용하는 방법에 대해서 배울 수 있었다. 코딩을 하면서 가장 어려웠던 점은 임계영역을 어디다가 두냐가 아니고 얼마나 스레드를 쉬게 해줘서 원하는 결과가 나오게 하는 것 이었다. 그리고 gcc를 이용할 때 -lpthread 라는 명령어를 몰라서 더 해맸던 시간이 컸다. 이번 과제는 동기화를 활용하는 방법 이었는데 이론으로 배웠던 것을 구현해보니 어렵게 배웠던 개념을 다시 정리할수 있는 기회가 되었던거 같다.

3. 실행환경

Ubuntu Linux 14.04