4주차 실습 보고서

실습 주제

- mailbox를 이용한 동기화

성공 실패 여부

- 성공

원인 분석

- mailbox생성

```
OS_EVENT* mbox_to_random[4];
OS_EVENT* mbox_to_decision[4];

for(i = 0; i < 4; i++) {
    mbox_to_random[i] = OSMboxCreate((void *)0);
    mbox_to_decision[i] = OSMboxCreate((void *)0);
}
```

각 OS_EVENT * []에 OSMboxCreate()을 통해 mailbox를 생성.

랜덤숫자를 보낼 mailbox 4개, 결과 문자 값을 보낼 mailbox 4개.

- decision task에서 랜덤 숫자 pending

```
int get_number[4];
for (i = 0;  i < N_TASK - 1; i++) {
    get_number[i] = *(int *)OSMboxPend(mbox_to_random[i], 0, &
}
```

반복문을 task의 개수만큼 돌면서 OSMBoxPend()을 통해, i번 째의 task의 random 값을 받을 때 까지 decision task 를 wait상태로 전환

이때, get_number[] 은pointer 가 아닌 int primitive type 자료형이므로 주소값이 아닌 실제 값을 저장해야 하기에, OSMboxPend의 return값인 (void *)을 (int *)을 형 변환 한 후, 추가적으로 *를 통해 역참조 해 주었다.

- decision task에서 결과 문자 post

```
char push_letter;
for (i = 0; i < N_TASK - 1; i++) {
        if (i == min_task) {
            push_letter = 'W';
        }
        else {
            push_letter = 'L';
        }
        if (select == 1) {
            OSMboxPost(mbox_to_decision[i], (void *)&push_let
        }
        else if (select == 2) {

        }
    }
```

가장 작은 random number을 보낸 task에 대해 push_letter을 설정하고, OSMboxPost()을 통해 mbox_to_decision[i] mailbox각각에 push_letter을 전송.

- random task에서 랜덤 숫자 pending, post

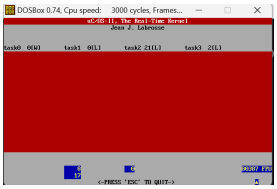
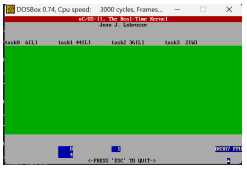
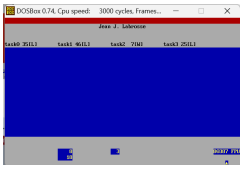
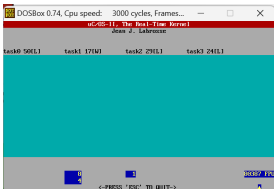
```
push_number = random(64);
char get_letter;
OSMboxPost(mbox_to_random[task_number], (void *)&push_number)
get_letter = *(char *)OSMboxPend(mbox_to_decision[task_number
```

OSMboxPost()을 통해 random task에서 decision task에 보낼 mailbox에 랜덤으로 생성한 숫자를 전달.

이후 OSMboxPend()을 통해 decision task에서 mbox_to_decision을 통해 보낸 값을 전달 받기 전까지 waiting 상태로 대기. 마찬가지로 get_letter은 type이 char이기 때문에

(void *) 을 (char *)로 형 변환 한 후 *를 통해 주소값을 역참조 하였다.

결과

5. 결과 분석

랜덤으로 돌아가며 각 task를 나타내는 빨 , 초, 파, 하늘이 출력되는 것을 확인 할 수 있다.