운영체제 과제1 보고서

컴퓨터공학부 202011632 김수비

1. 자율 진도표

단계	완료 여부
List 자료구조 파악: 예제 수행	o
code, process 구조체 생성	o
포인터 생성 후 동적할당	o
list_add_tail()을 이용해 연결리스트 구현	o
list_for_each_entry_reverse()를 이용해 연결 리스트 탐색 후 출력	o
과제 1-1 완료	o

2. 주요 코드 서술

(1)

```
typedef struct{ // 코드 구조체
    unsigned char operation; // 코드 동작 시간
    unsigned char length; // 동작
} code;

typedef struct{ // 프로세스 구조체
    int pid; // 프로세스 pid
    int arrival_time; // 도착시간
    int code_bytes; // 코드 길이
    code *operations; // code tuples가 저장된 위치(여기에 operation, length가 들어가있음)
    struct list_head job, ready, wait; // job, ready, wait라는 리스트의 노드 생성
} process;
```

code와 process 구조체를 생성합니다. process 구조체 안에 operations라는 code형 포인터와 list_head형의 job, read, wait 리스트 멤버 변수를 만듭니다.

(2)

```
while(fread(&p, sizeof(int)*3, 1, stdin)) // pid, arrival_time, code_bytes를 읽어옴(int 3개만 짤라서)
{
    cur = malloc(sizeof(process)); // cur을 생성(포인터가 구조체 자체)
    cur->pid = p.pid;// cur의 pid는 p의 pid
    cur->arrival_time = p.arrival_time;
    cur->code_bytes = p.code_bytes;
    INIT_LIST_HEAD(&cur->job); // cur의 job리스트 초기화
```

fread() 함수를 사용해 test1.bin에 있는 process의 pid, arrival_time, code_bytes를

읽어옵니다. 이때 int형 3개 크기 만큼의 정보를 읽어올 수 있게 sizeof(int)*3을 해줍니다. 그리고 앞에서 선언한 cur포인터를 process의 크기만큼 동적할당 시켜줍니다. cur의 pid, arrival_time, code_bytes는 p의 것과 같게 해줍니다. 그리고 INIT_LIST_HEAD()함수를 사용하여 cur의 job리스트를 초기화 시켜줍니다.

(3)

```
cur->operations = malloc(sizeof(cur->code_bytes)/2); // 사이즈가 가변적이기 때문에 동적할당

// operation과 length를 읽어옴
for(int i=0; i<p.code_bytes/2; i++){  // 코드길이/2 만큼 반복
    fread(&c, sizeof(char)*2, 1, stdin); // char의 크기의 2배만큼 크기 설정
    cur->operations[i].operation = c.operation;
    cur->operations[i].length = c.length;
}
```

cur의 operations를 code_bytes의 반만큼의 사이즈로 동적할당 시켜줍니다. 그리고 for문을 사용해 그 크기만큼 fread()함수를 이용하여 operation과 length를 읽어옵니다. 이때 읽어온 c의 operation을 cur의 operations배열의 operation과, c의 length를 cur의 operations배열의 length와 짝지어서 같게 해줍니다.

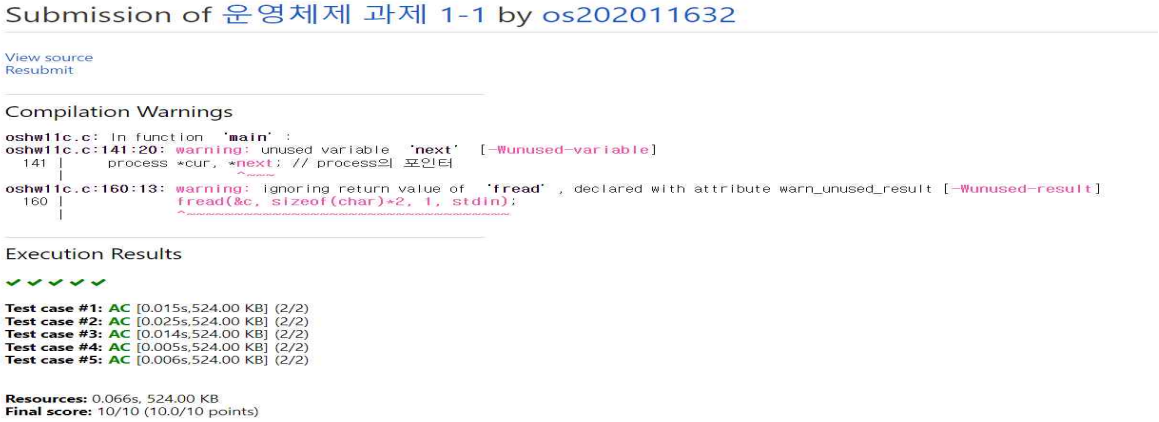
(4)

```
//list를 따라서 코드를 하나씩 출력
list_for_each_entry_reverse(cur, &job_q, job) // job_q 라는 리스트를 거꾸로 순회(cur의 job을 따라서)
{
    //pid, arrival_time, code_bytes를 출력
    fprintf(stdout, "PID: %03d\tARRIVAL: %03d\tCODESIZE: %03d\n", cur->pid, cur->arrival_time, cur->code_bytes);

    // operation과 length를 출력
    for(int i=0; i<cur->code_bytes/2; i++){  // 코드길이/2 만큼 반복
        fprintf(stdout, "%d %d\n", cur->operations[i].operation, cur->operations[i].length);
    }
```

list_for_each_entry_reverse()함수를 사용하여 리스트를 거꾸로 순회하고, 이때 cur의 pid, arrival_time, code_bytes를 순서대로 출력해줍니다. 그리고 for문을 사용하여 cur의 operation배열의 operation과 length를 차례로 출력해줍니다.

3. 1-1 제출 결과 캡처



4. 과제 수행 시 어려웠던 점 및 해결 방안

(1) sizeof(int)*3

```
while(fread(&p, sizeof(int)*3, 1, stdin))
```

test1.bin에서 각 process의 pid, arrival_time, code_bytes를 불러올 때의 사이즈 설정 문제에서 어려움을 겪었습니다. 과제0에서는 크기를 prrocess 구조체 그 자체로 설정하면 됐지만, 과제1에서는 code 포인터와 각 리스트들이 추가됐기 때문에 다른 방법이 필요했습니다. int형 변수 3개만 끊어서 불러올 수 있는 방법에 대해 생각했고 여러 가지 자료를 인터넷에서 찾아본 결과 sizeof(int)*3을 사용하면 int형 3개의 크기만큼 자료가 끊어져서 p의 각 정보에 대입된다는 사실을 알게 되었습니다.

(2) operations []

```
cur->operations[i].operation = c.operation;
cur->operations[i].length = c.length;
```

cur의 각 operations의 operation과 length 접근 방법에 대해 어려움을 겪었습니다. 포인터에 대한 이해가 부족하여 생긴 문제같아 포인터 자료를 찾아보고 공부를 다시 한 결과 포인터도 배열처럼 사용할 수 있다는 사실을 알게 되었습니다. 그래서 code형 포인터 operations를 배열처럼 i를 이용해 각 인덱스에 접근하여 operation과 length를 대입해줄 수 있었습니다.