운영체제 과제0 보고서

컴퓨터공학부 202011632 김수비

1. 주석이 포함된 전체 코드

hw0 > C os0.c

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct{ // 프로세스 구조체
    int pid; // 프로세스 ID
    int arrival_time; // 도착시간
    int code_bytes; // 코드 길이
} process;

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

int main(int argc, char* argv[]) {
    process p; // process 구조체 p
    codes c; // codes 구조체 c

    while(fread(&p, sizeof(process), 1, stdin))
    {
        fprintf(stdout, "%d %d %d\n", p.pid, p.arrival_time, p.code_bytes);

       for(int i=0; i<p.code_bytes/2; i++){  // 코드길이/2 만큼 반복
            fread(&c, sizeof(codes), 1, stdin);
            fprintf(stdout, "%d %d\n", c.length, c.operation); // 동작 시간과 동작 종류 출력
        }
    }
    return 0;
}
```

2. 주요한 부분에 대한 설명

(1) 프로세스 정보 출력

```
ubuntu@202011632:~/hw0$ gcc -o os0 os0.c
ubuntu@202011632:~/hw0$ cat test\(1\).bin | ./os0
```

os0.c를 컴파일하여 만들어지는 출력 파일을 os0이라는 이름으로 설정합니다. 파이프(|)를 사용해 바이너리 파일 test(1).bin의 이진 데이터를 os0와 연결시켜 사용할 수 있도록 합니다.

(2) 프로세스 정보 출력

```
19      while(fread(&p, sizeof(process), 1, stdin))
20      {
21          fprintf(stdout, "%d %d %d\n", p.pid, p.arrival_time, p.code_bytes);
22
23          for(int i=0; i<p.code_bytes/2; i++){  // 코드길이/2 만큼 반복
24              fread(&c, sizeof(codes), 1, stdin);
25              fprintf(stdout, "%d %d\n", c.length, c.operation); // 동작 시간과 동작 종류 출력
26          }
27      }
```

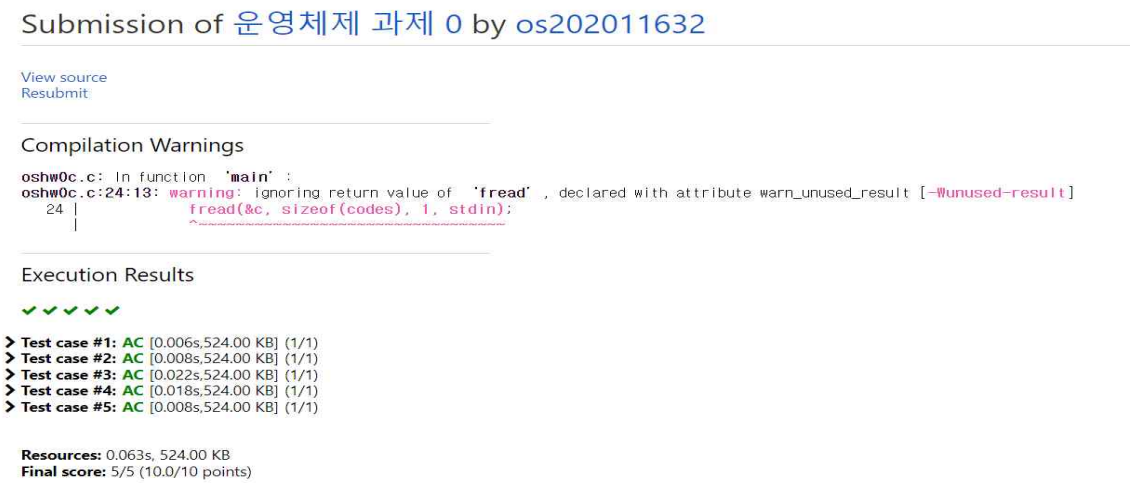
while 문 안에 fread함수를 넣어 프로세스의 정보를 출력하였습니다. fread함수는 process 구조체 p에서 process의 크기만큼의 데이터를 한번 읽어와 stdin으로 입력받습니다. 이진 데이터 전부를 읽을 때까지 while문은 반복됩니다. fprintf를 사용하여 구조체 p의 PID와, 도착 시간, 코드 길이를 표준 출력의 형태로 출력했습니다.

(2)코드 출력

```
23          for(int i=0; i<p.code_bytes/2; i++){  // 코드길이/2 만큼 반복
24              fread(&c, sizeof(codes), 1, stdin);
25              fprintf(stdout, "%d %d\n", c.length, c.operation); // 동작 시간과 동작 종류 출력
26          }
```

각 프로세스의 코드는 for문을 이용하여 반복 출력하였습니다. code tuple은 1바이트 크기의 길이와 동작 2개로 구성되어있기 때문에 프로세스 코드 길이의 반만큼의 코드 줄이 나옵니다. 그러므로 for문 이용해 p의 코드 길이의 반만큼 반복합니다. for문 안은 fread함수를 사용하여 c의 정보를 codes의 크기만큼 데이터를 한번 읽어와 stdin으로 입력시키는 동작을 하게 합니다. 또 fprint함수를 사용하여 c의 길이와 동작을 표준 출력시킵니다.

3. JOTA “운영체제과제 과제0” 제출 및 결과 캡처


Submission of 운영체제 과제 0 by os202011632

View source
Resubmit

Compilation Warnings

```
oshw0c.c: In function 'main':
oshw0c.c:24:13: warning: ignoring return value of 'fread', declared with attribute warn_unused_result [-Wunused-result]
   24 |             fread(&c, sizeof(codes), 1, stdin);
      |             ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Execution Results

✔ ✔ ✔ ✔ ✔

> **Test case #1:** AC [0.006s,524.00 KB] (1/1)
> **Test case #2:** AC [0.008s,524.00 KB] (1/1)
> **Test case #3:** AC [0.022s,524.00 KB] (1/1)
> **Test case #4:** AC [0.018s,524.00 KB] (1/1)
> **Test case #5:** AC [0.008s,524.00 KB] (1/1)

Resources: 0.063s, 524.00 KB
Final score: 5/5 (10.0/10 points)

4. 과제 수행 시 어려웠던 점 및 해결 방안

(1) 바이너리 파일 연결

주어진 test(1).bin 파일을 os0.c로 가져오는 과정에서 어려움을 느꼈습니다. 과제 ppt의 참고 문제를 보았는데도 이해가 되지 않아서 작년에 배운 유닉스프로그래밍 내용의 숙지가 부족하다 판단했습니다. 작년 수업 영상과 ppt를 통해 gcc, 파이프, fread함수, fprintf 함수의 동작 원리를 복습하였습니다. 또 구글링을 통해 추가적인 정보를 얻어 해결책을 찾을 수 있었습니다.

(2) 바이트 수

프로세스의 3가지 정보(PID, 도착시간, 코드 길이)는 4바이트이기에 int 자료형을 쓸 수 있었지만, 코드 출력에서는 그렇지 않았습니다. 코드는 1바이트 크기의 동작과 길이로 구성되어 있기 때문에 바이너리 파일을 1바이트 단위로 끊어줄 새로운 자료형이 필요했습니다. 그래서 1바이트 정수형 자료형을 찾아보았고 char형을 사용하게 되었습니다. 또 음수는 출력하지 않아도 되기 때문에 unsigned char 자료형은 사용해 length, operation을 선언할 수 있었습니다.