펭귄 스터디 내용13

작성자 : 류성수
작성일자 : 24.07.04

url : https://www.youtube.com/watch?v=T_JhyBV3gTI&list=PLz—ENLG_8TMdMJIwyqDIpcEOysvNoonf&index=15

● 구조체

1. 구조체는 효율성을 위해 패딩된다.

구조체는 데이터를 읽을 때, 속도의 효율성을 위해, 전체 크기를 4의배수로 올림연산을 한다.
그러나, pragma 명령어를 이용해, 전처리기 명령어를 내려주면, 원하는 크기로 패딩을 할 수 있다.
이를 확인하기 위해 실험해봤다.

```c
#include<stdio.h>

typedef struct
{
        int ID;
        char Name[21];
        char phoneNum[31];
        char sex;
}DATA;

#pragma pack(push,1)

typedef struct
{
        int ID;
        char Name[21];
        char phoneNum[31];
        char sex;
}DATA2;

#pragma pack(pop)

void main()
{
        DATA data1;
        DATA2 data2;

        printf("non_pragma struct size : %ld\n",sizeof(data1));
        printf("pragma struct size : %ld\n", sizeof(data2));

}
```

DATA 구조체는 4+21+31+1 = 57 이기에, 총 60바이트의 크기가 나온다
DATA2 구조체는 패딩자체를 안줬기에, 57바이트의 크기가 나와야 한다.

```
root@LAPTOP-BQ4AK11N:/project/24-Cstudy13# ./struct_prac.out
non_pragma struct size : 60
pragma struct size : 57
```

예상대로 나온 것을 확인할 수 있다.

이와 같은 명령어는 특히, 이미지 파일의 헤더나 음성 파일의 헤더를 운영체제가 정확히 읽어야 할 때, 매우 중요한 명령이다.
패딩이 된 헤더를 운영체제가 잘못읽으면, 구조체 헤더의 멤버를 잘못읽어서 아예 읽을 수 없는 파일이라고 뜰 수 있기 때문이다.

2. 구조체 인스턴스의 시작주소

구조체의 인스턴스의 시작주소는 무엇인지를 확인하였다.

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

typedef struct
{
    int id;
    char name[20];
} DAT;

void struct_init(DAT* data)
{
    data->id = 10;
    strcpy(data->name, "sungsu");
}

int main()
{
    DAT data1;

    struct_init(&data1);

    printf("data : %d, %s\n", data1.id, data1.name);
    printf("just data1 : %p\n", data1);
    printf("addr data.id : %p\n", &data1.id);
    printf("addr data1 : %p\n", &data1);

    // 인스턴스의 첫번째 주소값은 인스턴스 첫멤버의 주소값과 같다.
    // 배열처럼 작동
    return 0;
}
```

```
root@LAPTOP-BQ4AK11N:/project/24-Cstudy13# ./struct_prac2.out
data : 10, sungsu
just data1 : 0x560b66ff42a0
addr data.id : 0x7ffcf1c51620
addr data1 : 0x7ffcf1c51620
```

결과를 통해, 구조체 인스턴스의 시작주소는 구조체의 첫 멤버의 시작주소와 같음을 확인하였다.