펭권 스터디 내용18

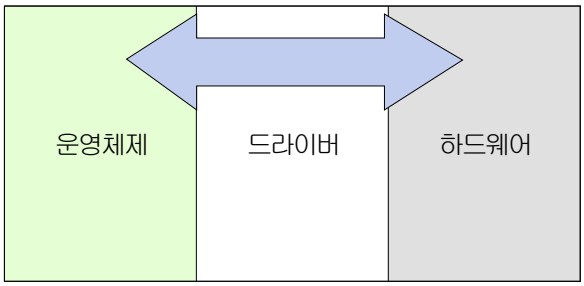
작성자 : 류성수
작성일자 : 24.07.05

url : https://www.youtube.com/watch?v=W7JZ3nkJtUI&list=PLz—ENLG_8TMdMJIwyqDlpcEOysvNoonf&index=20

● 파일입출력의 이해

테이프부터 SSD까지 저장매체들은 저장하는 법이 다 다르다.
임베디드 단에선 메모리에 읽고, 기록하기위해, 저장매체의 data sheet를 다 읽어와야 함.
어떻게 기록하고 읽어와야 하는지는 해당 sheet에 존재해서 그 순서대로 FW를 작성한다.

저장매체 HW의 파일구조는 HW에 어떻게 저장되는지, 몇블록 단위로 저장되는지 다 다르다
이러한 저장매체마다 다 다르게 개발자가 저장하는 방법을 다르게 하는 것은 비효율적이기에, 계층을 나눠놓았다.



드라이버는 OS와 HW의 다리역할을 한다.
그래서 앱 개발자들은 드라이버에 대해 신경쓰지 않고, 하나의 함수로만 사용하면, 운영체제가 드라이버를 통해서 HW로 data를 가져오거나 전달함.

누군가(드라이버개발자) 운영체제의 드라이버를 구현해놓았다.

1. stream

HW는 대체적으로 한번에 64KB씩 저장한다. 만약 문자 하나하나당 64KB의 저장공간을 갖게 되면 너무나도 비효율적이다.
그래서 스트림이란 구조가 운영체제에 논리적으로 적용된다.

이 스트림은 사용자의 입력을 다른곳에 저장하고, HW마다 쓰는 단위 이상의 data가 스트림에 저장될 경우, 해당 단위를 드라이버를 통해서 HW에 작성한다.

혹시라도 이를 원치 않으면, fflush 함수를 사용하여 그냥 HW에 작성하는 방법도 있따.

● 실습

배울 함수는 fopen , fread, fwrite 함수들에 대해서 배운다.

먼저, fopen으로 스트림을 열고,
fread로 스트림을 통해서 읽는다. 혹은 fwrite로 스트림을 통해서 쓴다.
꼭, fclose 함수로 스트림을 닫아줘야 한다.

```c
#include <stdio.h>

void main()
{
	FILE* fp;
	char ch;

	fp = fopen("text.txt","a");

	if(fp == NULL)
		printf("file open error\n");

	fprintf(fp, "hello hello~ 한글도 되나요? \n");

	fclose(fp);

	fp = fopen("text.txt","r");

	if(fp == NULL)
		printf("file open error\n");

	while(1)
	{
		ch = fgetc(fp);

		if(ch == EOF)
			break;

		putchar(ch);
	}


	fclose(fp);
```

다음과 같이 파일포인터를 선언하고, 쓰기모드(추가)로 파일을 열고, 쓴다음, 다시 읽기모드로 열어서 읽어서 내용을 출력하였다.

```
hello linux
my name is sungsu ryu
thank you
hello hello~ 한글도 되나요?
```

결과를 확인하였다.

● 파일 디스크립터

운영체제에서 파일이나 입출력 자원을 나타내기 위해 사용하는 추상적인 핸들이다.
주로 유닉스와 리눅스계열의 운영체제에서 사용된다.

1. 특징

파일디스크립터는 정수로 표현된다. 해당 정수는 특정 파일이나 입출력 리소스를 가리킨다

정수값	디스크립터
-1	스트림 열기 실패 (NULL)
0	표준 입력 (stdin)
1	표준 출력 (stdout)
2	표준 오류 (stderr)
3~4	특정 파일

2. 함수

리눅스, 유닉스 계열에서 파일디스크립터를 사용하기 위해선 다음과 같은 함수들을 사용할 수 있다.

함수	내용
open()	파일을 열고 파일 디스크립터를 반환
read()	파일 디스크립터로부터 데이터를 읽음
write()	파일 디스크럽터에 데이터를 씀
close()	파일 디스크립터를 닫음
dup()	기존 파일 디스크립터를 복제함

어떻게 보면, 리눅스의 fopen, fwrite, fread 함수들의 원형은 이러한 함수들로 이루어져 있다고 볼 수 있다.

```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>

int fd;

void main()
{
        fd = open("test.txt",O_WRONLY|O_APPEND);
        // fopen의 원형
        printf("fd : %d\n",fd);
        // fd = 0 표준입력 (키보드)
        // fd = 1 표준출력 (콘솔, 혹은 설정한대로)
        // fd = 2 에러출력 (콘솔) -> 3번과는 다른 스트림
        // fd = 3,4 (파일)

        char test[] = "this is for test\n";
        write(fd,test,sizeof(test));
        // fwrite의 원형

        close(fd);
}
```

fopen() mode	open() flags
r	O_RDONLY
w	O_WRONLY | O_CREAT | O_TRUNC
a	O_WRONLY | O_CREAT | O_APPEND
r+	O_RDWR
w+	O_RDWR | O_CREAT | O_TRUNC
a+	O_RDWR | O_CREAT | O_APPEND

파일을 열고서 디스크립터를 확인 해 보니, 디스크립터가 3이 나왔다.

```
root@LAPTOP-BQ4AK11N:/project/24-Cstudy18# ./FILE_original.out
fd : 3
```

3. 파일구조체는 디스크립터로 만들어진건가?

```c
struct _iobuf {
        char *_ptr;
        int   _cnt;
        char *_base;
        int   _flag;
        int   _file;
        int   _charbuf;
        int   _bufsiz;
        char *_tmpfname;
        };
typedef struct _iobuf FILE;
```

파일구조체에는 다음과 같은 멤버들이 존재하고, 이들 중 파일 디스크립터를 다른이름으로 저장한 멤버가 있다.

```c
#include <stdio.h>

FILE*fp;
FILE*fp2;

void main()
{
        fp = fopen("test.txt","a");
        fp2 = fopen("test2.txt","a+");

        printf("fp1's fd : %d\n", fp->_fileno);
        printf("fp2's fd : %d\n", fp2->_fileno);

        fclose(fp);
        fclose(fp2);
}
```

여기서 구조체의 _fileno 멤버를 열면, 디스크립터가 얼마로 설정된건지를 확인이 가능하다.

```
root@LAPTOP-BQ4AK11N:/project/24-Cstudy18# ./FILE_original2.out
fp1's fd : 3
fp2's fd : 4
```