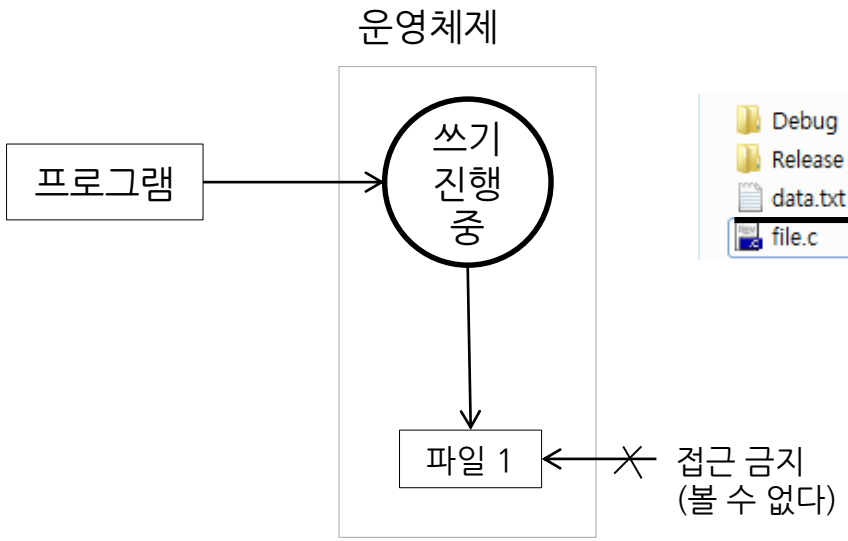
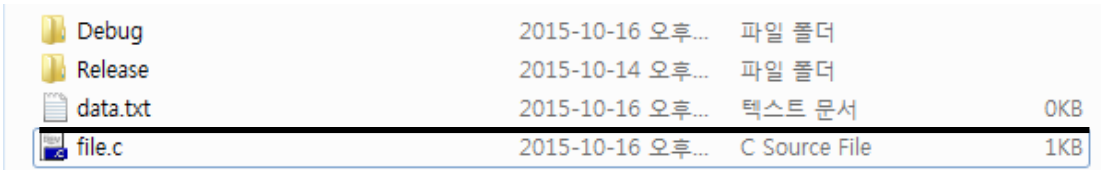
15차시 - 파일(3)

파일의 접근

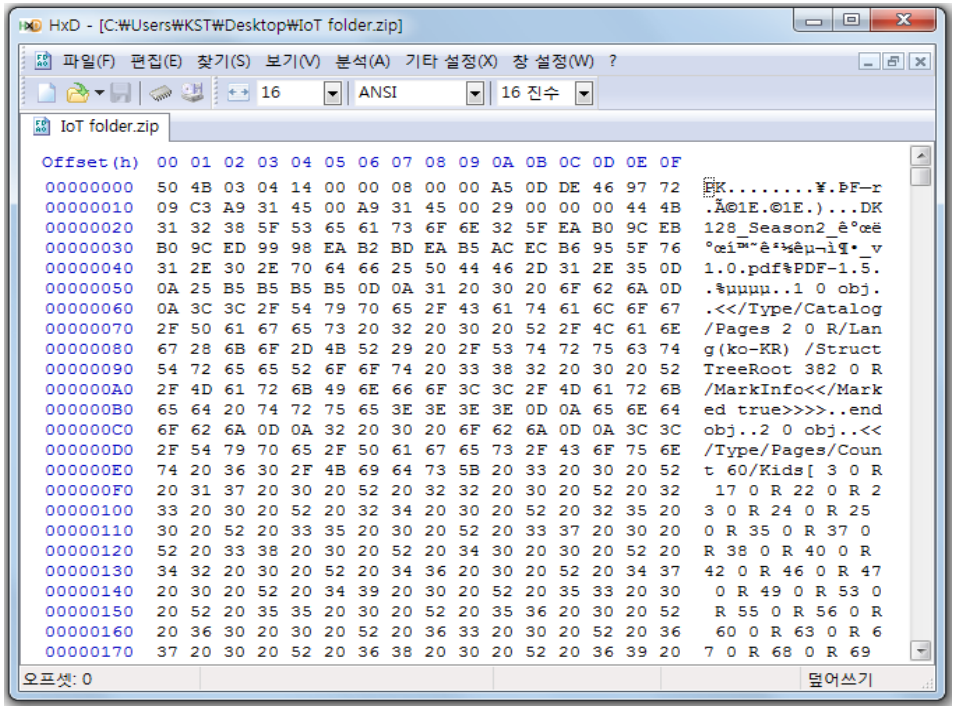
- **열려 있는 파일은**
  - 다른 프로세스에서 삭제할 수 없거나, (열려있는 파일)
  - 다른 프로세스에서 읽을 수 없거나, (쓰기 위해 열려있는 파일)
  - 다른 프로세스에서 읽을 내용이 없다. (쓰기 위해 열려 있는 파일)





이진 파일 보기

- **바이너리 편집기를 이용한다.**
  - 이진 데이터를 보더라도 이해하기 어렵다.
  - 그나마 쉽게 볼 수 있도록 해주는 도구



텍스트 파일의 이진 파일 보기

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00000000	23	64	65	66	69	6E	65	20	5F	43	52	54	5F	53	45	43
00000010	55	52	45	5F	4E	4F	5F	57	41	52	4E	49	4E	47	53	0D
00000020	0A	23	69	6E	63	6C	75	64	65	20	3C	73	74	64	69	6F
00000030	2E	68	3E	20	0D	0A	0D	0A	69	6E	74	20	6D	61	69	6E
00000040	28	29	0D	0A	7B	0D	0A	09	46	49	4C	45	20	2A	20	70
00000050	46	69	6C	65	3B	0D	0A	09	69	6E	74	20	63	3D	31	30
00000060	30	30	3B	0D	0A	09	70	46	69	6C	65	20	3D	20	66	6F
00000070	70	65	6E	28	22	69	6E	74	2E	64	61	74	22	2C	20	22
00000080	77	62	22	29	3B	0D	0A	0D	0A	09	69	66	20	28	70	46
00000090	69	6C	65	20	3D	3D	20	4E	55	4C	4C	29	20	70	65	72
000000A0	72	6F	72	28	22	45	72	72	6F	72	20	6F	70	65	6E	69
000000B0	6E	67	20	66	69	6C	65	22	29	3B	0D	0A	09	65	6C	73
000000C0	65	0D	0A	09	7B	0D	0A	09	09	66	77	72	69	74	65	28
000000D0	26	63	2C	20	34	2C	20	31	2C	20	70	46	69	6C	65	29
000000E0	3B	0D	0A	09	09	66	63	6C	6F	73	65	28	70	46	69	6C
000000F0	65	29	3B	0D	0A	09	09	72	65	74	75	72	6E	20	30	3B
00000100	0D	0A	09	7D	0D	0A	7D									

```
#define _CRT_SEC
URE_NO_WARNINGS.
.#include <stdio
.h> ....int main
()..{...FILE * p
File;...int c=10
00;...pFile = fo
pen("int.dat", "
wb");.....if (pF
ile == NULL) per
ror("Error openi
ng file");...els
e...{....fwrite(
&c, 4, 1, pFile)
;....fclose(pFil
e);....return 0;
...}..}
```

이진 값으로 표시하면

텍스트로 표시하면

특이점을 찾아보자.

파일의 선택 (텍스트 / 이진)

- `i = 1000;`
- **이 값을 저장하는 두 가지 방법**
  - ① 텍스트 파일로 오픈한 후, `fputs`로 "1000" 저장 (4바이트)
    - 읽을 때, `fgets`로 읽은 후, `atoi`로 변환하거나, `fscanf`로 읽는다.
  - ② 이진 파일로 오픈한 후, `fwrite`로 1000 저장 (4바이트)
    - 읽을 때 `fread`로 읽는다.
- **상황에 따라 맞는 방식으로 저장한다.**

파일의 선택 (텍스트 / 이진)

```
FILE *pFile;
int  i = 123456, j;

pFile = fopen("data.txt", "w");
fprintf(pFile, "%d", i);
fclose(pFile);

pFile = fopen("data.txt", "r");
fscanf(pFile, "%d", &j );
fclose(pFile);
```

"123456" (6바이트)

```
FILE *pFile;
int  i = 123456, j;

pFile = fopen("data.dat", "wb");
fwrite(&i, 4, 1, pFile);
fclose(pFile);

pFile = fopen("data.dat", "rb");
fread(&j, 4, 1, pFile);
fclose(pFile);
```

40 E2 01 00 (4바이트)

파일의 선택 (텍스트 / 이진)

```
struct class {
    char  name[30];       // 이름
    char  address[50];    // 주소
    char  telephone[15];  // 전화번호
}  myclass;
```

```
struct class myclass = {"Kim", "Seoul",
"010-1111-2222"}, yourclass;

FILE *fp;
fp = fopen("data.txt", "w");
fprintf(fp, "%s\n", myclass.name);
fprintf(fp, "%s\n", myclass.address);
fprintf(fp, "%s\n", myclass.telephone);
fclose(fp);

fp = fopen("data.txt", "r");
fgets(yourclass.name, 30, fp);
fgets(yourclass.address, 50, fp);
fgets(yourclass.telephone, 15, fp);
fclose(fp);
```

27 bytes

```
struct class myclass = {"Kim", "Seoul", "010-
1111-2222" }, yourclass;

FILE *fp;
fp = fopen("data.dat", "wb");
fwrite(&myclass, sizeof(struct class), 1, fp);
fclose(fp);



fp = fopen("data.dat", "rb");
fread(&yourclass,  sizeof(struct  class),  1,
fp);
fclose(fp);
```

95 bytes

- 파일 저장 방식은 선택의 문제이다.

파일의 선택

- **전반적으로**
  - 파일 크기가 효율적인 경우
  - 처리가 단순한 경우
- **텍스트 파일이 좋은 경우**
- **이진 파일이 좋은 경우**

파일 처리의 성능

- **파일 저장 장치(보조 기억장치)는 느리다.**
  - 운영체제는 나름 노력한다. 그래도 느리다.
  - 개발자가 빠르게 사용하기 위한 노력을 해야 한다.
- **성능 향상 방법**
  - 파일 **입출력은 비교적 크게** 한다.
  - 파일 입출력은 **한꺼번에, 순차적으로** 한다.

파일의 설계

텍스트 파일의 설계

방식 1

```
Kim
80
80
90
```

순서가 반드시 유지되어야 한다.

순서를 알아야 읽기가 가능하다.

데이터 파일을 수정할 때
조심해야 한다.

간단한 프로그램을 작성할 때
용이하다.

방식 2

```
name = Kim
Kscore = 80
Escore = 80
Mscore = 90
```

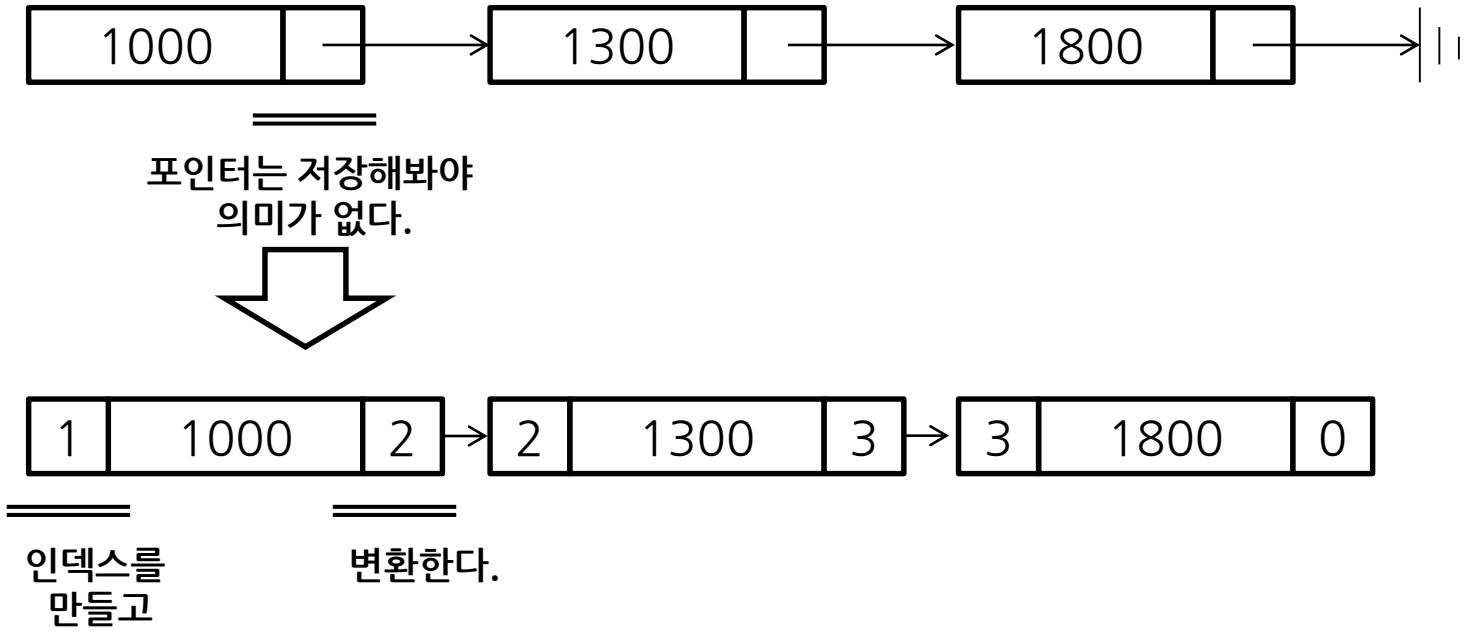
데이터의 수정이 용이하다.
순서가 유지될 필요도 없다.

사용자의 편의를 위해 개발자가 많은
일을 해야 한다.

범용적이므로 이 방식을 많이 쓴다.

파일의 설계

포인터의 저장

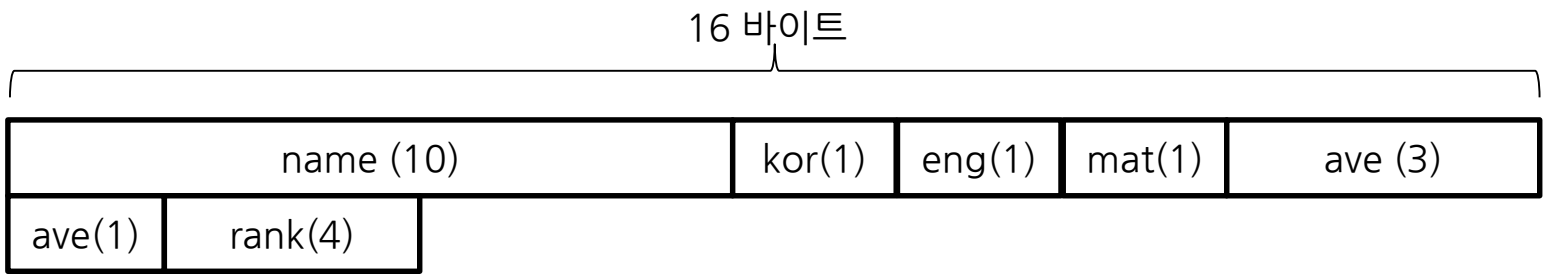


파일의 설계

- **이진 파일**
  - 데이터 파일은 대부분 이진 파일이다.
  - 파일의 저장 순서대로 파일을 읽어야 한다.
  - 파일의 저장 구조를 알려주지 않으면 파일의 내용을 알 수 없다.
  - 다른 프로그램과 공유하려면 파일의 구조를 잘 설명해 주어야 한다.
    - mp3, bmp, png, jpg, wav 등

파일의 설계

순서	이름	크기(자료형)	용도
1	name	10 (char)	학생의 이름
2	kor	1 (char)	국어 점수
3	eng	1 (char)	영어 점수
4	mat	1 (char)	수학 점수
5	ave	4 (float)	평균
6	rank	4 (int)	순위



16 바이트 단위로 설계 내용을 보여준다.

파일의 설계

- **별도의 파일 입출력 구조체를 이용한다.**
  - 저장/읽기를 할 데이터들을 구조체에 모아둔다.
  - 구조체 단위로 입출력을 한다.
  - 파일 구조를 유지시키고, 실수를 예방한다.

```
// 저장하기
FileBlock.year = thisyear;
FileBlock.month = thismonth;
FileBlock.day = today;
FileBlock.price = todayprice;

fwrite(&FileBlock,
       sizeof(struct _FILEB), 1,
       pFile);
```

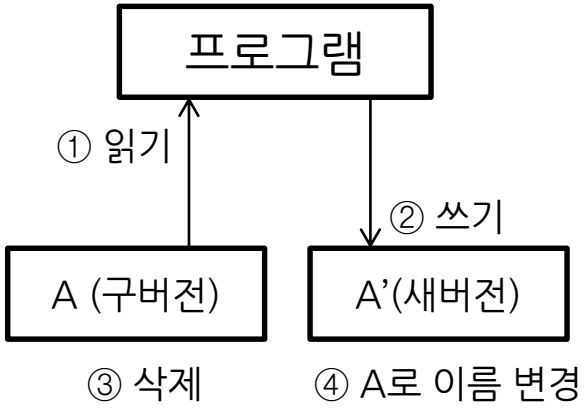
```
// 불러오기
fread(&FileBlock,
      sizeof(struct _FILEB), 1,
      pFile);

thisyear = FileBlock.year;
thismonth = FileBlock.month;
today = FileBlock.day;
todayptice = FileBlock.price;
```

파일 사용시 주의할 점

- 원본 데이터는 잘 보관하라.
- 하나의 파일에 파일 읽기와 쓰기를 하지 마라.
- 파일 입출력 부분은 별도의 함수로 구분하라
- 프로그램이 비정상 종료될 수 있다.

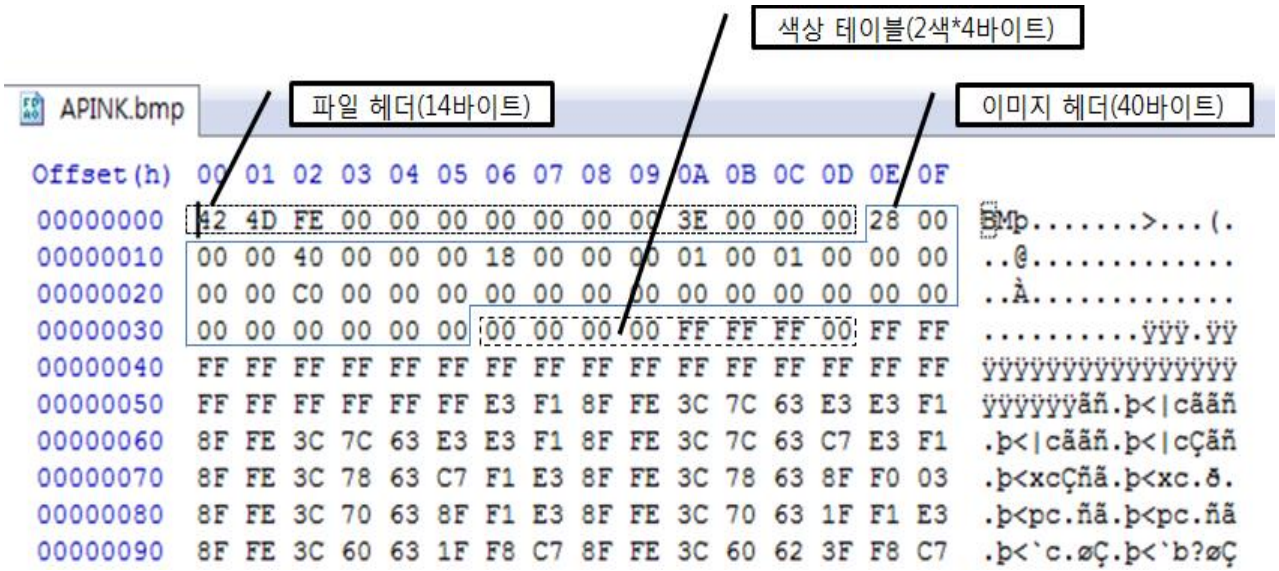
안전하게 저장하기

- **일반적인 프로그램**
  - 파일을 읽기
  - 데이터 처리
  - 파일에 기록하기 <= 이 순간 프로그램이 죽는다면?
- **안전한 저장 방법**



bmp 파일의 사용

- **bmp 파일**
  - 이미지 파일이면서 복잡한 압축 알고리즘이 없다.
  - 보다 쉽게 접근하기 위해 흑백 bmp 파일을 사용
  - 앞부분에 그림 파일의 정보는 분석해야 한다. (검색)
  - 이미지 부분만 읽어오면 그림을 표시할 수 있다.

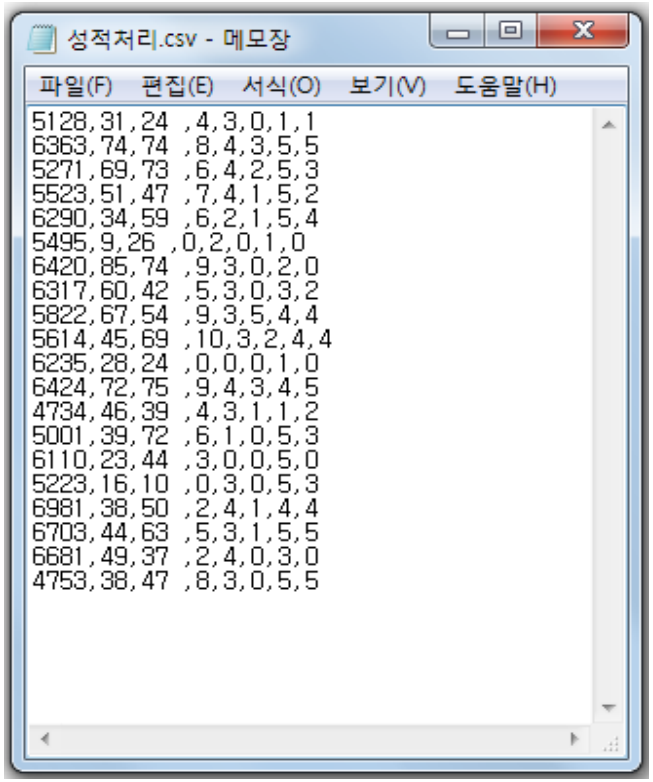


사전 파일 읽기

- **영어 퍼즐 게임**
  - 많은 단어가 포함된 **사전**이 필요
  - 사전 파일(올바른 단어가 많이 있는)을 사용
  - 사전 파일의 양식대로 단어를 읽어온다.
  - 텍스트 파일로 구성되어 있으면 더 편리하다.

엑셀 데이터를 읽어오기

- CSV
  - comma separated value (데이터를 ‘,’ 로 구분함)
  - 엑셀 데이터를 텍스트 파일로 저장하는 방식
- **CSV 파일에서 데이터를 추출**
  - 한 행을 읽은 후(`fgets`), ‘,’로 구분된 데이터 부분을 추출한다. (`strtok`)



ZIP 파일

- **압축 파일**
  - 압축 알고리즘을 이용해 작은 크기의 파일로 만든 것
  - “압축하지 않음”을 선택할 수도 있다.
  - 파일 목록과 내용이 하나의 파일에 들어있다.
- **과제**
  - “압축하지 않음”을 선택한 ZIP 파일에 포함된 파일을 원래 상태로 풀어놓는다.