파일 입출력

부산대학교 공과대학
정보컴퓨터공학부

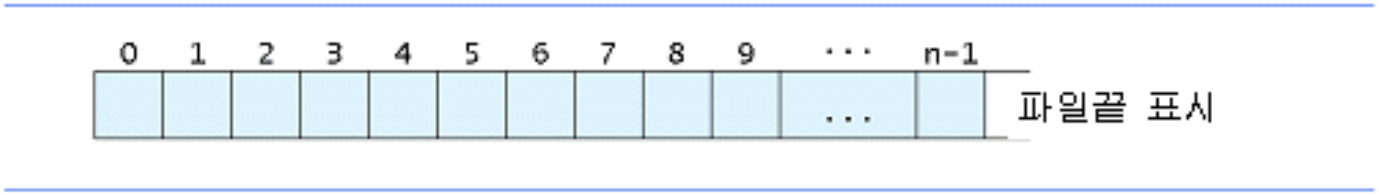
학습 목표

- 파일의 기본 개념과 특징을 이해할 수 있다.
- 파일 처리 과정을 이해할 수 있다.
- 형식을 지정한 파일 입출력 함수의 사용법을 알 수 있다.

파일과 파일 포인터

파일

❖ C의 파일은 모든 데이터를 연속된 바이트 형태로 저장한다.



C 언어의 파일 종류

❖ 텍스트 파일(text file)

- 사람들이 읽을 수 있는 문자들을 저장하고 있는 파일
- 텍스트 파일에서 “한 줄의 끝”을 나타내는 표현은 파일이 읽어 들여질 때, C 내부의 방식으로 변환된다.

❖ 이진 파일(binary file)

- 모든 데이터는 있는 그대로 바이트의 연속으로 저장
- 이진 파일을 이용하여 메모리에 저장된 변수 값 형태 그대로 파일에 저장할 수 있다.

파일 열기

파일을 사용하기 위해서는

❖ 반드시 먼저 파일 열기(fopen)를 해야 한다.

- 파일 열기를 하면 FILE (구조체에 대한) 포인터가 리턴된다.
- FILE 포인터는 열린 파일을 지정한다.

❖ FILE 구조체

- stdio.h에 정의되어 있음.
- 열려진 파일의 현재 상태를 나타내는 필드 변수들
- 특히 파일 입출력에 사용되는 버퍼 관련 변수들

표준 입출력

❖ stdin, stdout, stderr

- 각각 표준 입력, 표준 출력, 표준 오류를 나타내는 FILE 포인터
- C 프로그램이 실행되면 자동적으로 열리고 프로그램이 종료될 때 자동으로 닫힘
- 표준입출력 포인터

표준입출력 포인터	설명	가리키는 장치
stdin	표준입력에 대한 FILE 포인터	키보드
stdout	표준출력에 대한 FILE 포인터	모니터
stderr	표준오류에 대한 FILE 포인터	모니터

파일 열기

❖ 함수 fopen()

- FILE *fopen(const char *filename, const char *mode);
  - const char *filename: 파일명에 대한 포인터
  - const char *mode: 모드로 파일을 여는 형식

❖ 예

```
FILE *fp;
fp = fopen("c:\work\text.txt", "r");
if (fp == NULL) {
    printf("파일 열기 오류\n");
}
```

❖ 예

- fp = fopen("outdata.txt", "w");
- fp = fopen("outdata.txt", "a");

fopen 모드

❖ fopen 함수의 텍스트 파일 입출력 모드

모드	의미	파일이 없으면	파일이 있으면
"r"	읽기 전용(read)	NULL 반환	정상 동작
"w"	쓰기 전용(write)	새로 생성	기존 내용 삭제
"a"	추가 쓰기(append)	새로 생성	기존 내용 뒤에 추가
"r+"	읽기와 쓰기	NULL 반환	정상 동작
"w+"	읽기와 쓰기	새로 생성	기존 내용 삭제
"a+"	추가를 위한 읽기와 쓰기	새로 생성	기존 내용 뒤에 추가

파일 닫기

- 파일을 열어서 사용한 후에는 파일을 닫아야 한다.
  - int fclose(FILE *fp);
    - fp는 fopen 함수에서 받았던 포인터
    - 닫기에 성공하면 0, 오류일 때는 EOF(-1)를 리턴한다.

- 예
  - fclose(fp);

파일 입출력 함수

파일 입출력 함수

❖ 표준 입출력 함수와 표준 파일입출력 함수

표준 입출력함수	표준파일 입출력함수	기능
getchar()	fgetc(), getc()	문자 단위로 입력하는 함수
putchar()	fputc(), putc()	문자 단위로 출력하는 함수
gets()	fgets()	문자열을 입력하는 함수
puts()	fputs()	문자열을 출력하는 함수
scanf()	fscanf()	자료형에 따라 자료를 입력하는 함수
printf()	fprintf()	자료형에 따라 자료를 출력하는 함수

문자 단위 입출력

- fgetc() 함수와 fputc() 함수
  - 파일에 문자 단위 입출력을 할 수 있다.

- int fgetc(FILE *fp);
  - getc 함수는 fp가 지정한 파일에서 한 문자를 읽어서 리턴한다.
  - 파일 끝에 도달했을 경우에는 EOF(-1)를 리턴한다.

- int fputc(int c, FILE *fp);
  - putc 함수는 파일에 한 문자씩 출력하는 함수
  - 리턴값으로 출력하는 문자 리턴
  - 출력시 오류가 발생하면 EOF(-1) 리턴

- 예
  - [프로그램 14-1] 키보드에서 문자를 입력 받아 모니터에 출력 (copy1.c)
  - [프로그램 14-2] 키보드에서 문자를 입력 받아 파일에 출력 (copy2.c)
  - [프로그램 14-3] 파일 복사 프로그램 (copy3.c)
    - 학습 point: 표준 파일입출력 함수인 fputc를 이용한 표준 입출력 동작 이해

프로그램 11-1

```c
#include <stdio.h>

int main()
{
  int c;

  c = fgetc(stdin);   // 키보드로부터 입력 받은 문자의 ASCII 코드
    while (c != EOF) {  // file 끝이 아니면.
    fputc(c, stdout);   // fp가 가리키는 파일에 문자 c 출력
      c = fgetc(stdin);   // 키보드로부터 문자를 읽어 c에 저장
  }

  return 0;
}
```

프로그램 11-2

```c
#include <stdio.h>

int main(int argc, char *argv[])
{
  FILE *fp;
  int c;

  fp = fopen(argv[1],"w");  // 쓰기 전용으로
  c = getc(stdin);          // 키보드로부터 입력받은 문자의 ASCII코드
      while (c != EOF) {    // file의 끝이 아니면.
    putc(c, fp);    // fp가 가리키는 파일에 문자 c저장
      c = getc(stdin);  // 키보드로부터 문자를 읽어 c에 저장
  }
  fclose(fp);
  printf("%s 파일에 저장 완료.\n", argv[1]);

  return 0;
}
```

프로그램 11-3

```c
#include <stdio.h>

int main(int argc, char *argv[])
{
  char c;
  FILE *fp1, *fp2;

  fp1 = fopen(argv[1], "r");
  if (fp1 == NULL) {
    printf("파일 %s 열기 오류!\n", argv[1]);
    exit(1);
  }

  fp2 = fopen(argv[2], "w");
  while ((c = fgetc(fp1)) != EOF)
    fputc(c, fp2);

  fclose(fp1);
  fclose(fp2);

  return 0;
}
```

기타 파일 관련 함수

- int feof(FILE *fp)
  - 파일 포인터 fp가 파일의 끝을 탐지하면 0이 아닌 값을 리턴하고 파일 끝이면 0을 리턴 한다.

- int ungetc(int c, FILE *p)
  - c에 저장된 문자를 입력 스트림에 반납한다. 마치 문자를 읽지 않은 것처럼 파일 위치 지정자를 1 감소시킨다.

- int fflush(FILE *fp)
  - 아직 기록되지 않고 버퍼에 남아 있는 데이터를 fp가 가리키는 출력 파일에 보낸다. 버퍼 비우기 기능을 수행하는 함수이다.

줄 단위 입출력

- fgets() 함수와 fputs() 함수
  - 텍스트 파일에서 한 줄씩 읽거나 쓸 수 있다.

- char* fgets(char *s, int n, FILE *fp);
  - 파일로부터 한 줄을 읽어서 문자열 포인터 s에 저장하고 s를 리턴
  - 개행문자('\n')나 EOF를 만날 때까지 파일로부터 최대 n-1 개의 문자를 읽고 읽어온 데이터의 끝에는 NULL 문자를 붙여준다.
  - 파일을 읽는 중 파일 끝 혹은 오류가 발생하면 NULL 포인터 리턴.

- int fputs(const char *s, FILE *fp);
  - 문자열 s를 파일 포인터 fp가 가리키는 파일에 출력
  - 성공적으로 출력한 경우에는 출력한 바이트 수를 리턴
  - 출력할 때 오류가 발생하면 EOF 값을 리턴

- 예
  - [프로그램 14-4] 줄 번호와 함께 출력하는 프로그램 (line.c)
    - 학습 point: fgets를 이용한 파일의 줄단위 읽기

프로그램 11-4

```
6   #include <stdio.h>
7   #define MAXLINE 80
8
9   int main(int argc, char *argv[])
10  {
11      FILE *fp;
12      int line = 0;
13      char buffer[MAXLINE];
14
15      if (argc != 2) {
16          fprintf(stderr, "사용법:line 파일이름\n");
17          return 1;
18      }
19
20      if ( (fp = fopen(argv[1],"r")) == NULL) {
21          fprintf(stderr, "파일 열기 오류");
22          return 2;
23      }
24
25      while (fgets(buffer, MAXLINE, fp) != NULL)   // 한 줄씩 읽기
26      {
27            line++;
28          printf("%3d %s", line, buffer);     // 줄번호와 함께 프린트
29      }
30      return 0;
31  }
```

포맷 입출력

- fprintf() 함수
  - printf() 함수와 같은 방법으로 파일에 데이터를 출력할 수 있다.
- fscanf() 함수
  - scanf() 함수와 같은 방법으로 파일로부터 데이터를 읽어 들일 수 있다.

- int fprintf(FILE *fp, const char *format, ...);
  - fprintf 함수의 첫 번째 인수 fp는 출력할 파일에 대한 FILE 포인터
  - 두 번째부터의 인수는 printf 함수와 동일

- int fscanf(FILE *fp, const char *format, ...);
  - fscanf 함수의 첫 번째 인수 fp는 입력받을 파일에 대한 FILE 포인터
  - 두 번째부터의 인수는 scanf 함수와 동일

- 예
  - [프로그램 14-5] 학생 정보를 읽어 텍스트 파일에 저장하는 프로그램 (fprint.c)
  - [프로그램 14-6] 텍스트 파일에서 학생 정보를 읽어 출력한다. (fscan.c)
    - 학습 point: fgets를 이용한 파일의 줄단위 읽기

프로그램 11-5

```c
#include <stdio.h>
#include "student.h"

int main(int argc, char* argv[])
{
    struct student record;
    FILE *fp;

    if (argc != 2) {
        fprintf(stderr, "사용법: %s 파일이름\n", argv[0]);
        return 1;
    }

    fp = fopen(argv[1], "w");
    printf("%s %7s %6s\n", "학번",  "이름",  "점수");

    while (scanf("%d %s %d", &record.id, record.name, &record.score) == 3)
        fprintf(fp, "%d %s %d ", record.id, record.name, record.score);

    fclose(fp);
    return 0;
}
```

student.h

```c
student struct {
    int id;
    char name[20];
    short score;
}
```

프로그램 11-6

```c
#include <stdio.h>
#include "student.h"

int main(int argc, char* argv[])
{
    struct student record;
    FILE *fp;

    if (argc != 2) {
        fprintf(stderr, "사용법: %s 파일이름\n", argv[0]);
        return 1;
    }

    fp = fopen(argv[1], "r");
    printf("%s %7s %6s\n", "학번",  "이름",  "점수");

    while (fscanf(fp,"%d %s %d", &record.id, record.name, &record.score)==3)
        printf("%d %s %d\n", record.id, record.name, record.score);

    fclose(fp);
    return 0;
}
```

이진 파일

블록 입출력 (1/2)

- fread()와 fwrite()
  - 한번에 일정한 크기의 데이터를 파일에 읽거나 쓰기 위한 입출력 함수

- int fread(void *buf, int size, int n, FILE *fp);
  - fp가 가리키는 파일에서 size 크기의 블록(연속된 바이트)을 n개 읽어서 버퍼 포인터 buf가 가리키는 곳에 저장
  - 읽어온 블록의 개수를 리턴

- int fwrite(const void *buf, int size, int n, FILE *fp);
  - 파일 포인터 fp가 지정한 파일에 버퍼 buf에 저장되어 있는 size 크기의 블록(연속된 바이트)을 n개 기록
  - 성공적으로 출력한 블록 개수를 리턴

블록 입출력 (2/2)

❖ 기본 아이디어

- 어떤 자료형의 데이터이던지 그 데이터를 연속된 바이트로 해석해서 파일에 저장
- 파일에 저장된 데이터를 연속된 바이트 형태로 읽어서 원래 자료형 변수에 순서대로 저장하여 원래 데이터를 그대로 복원

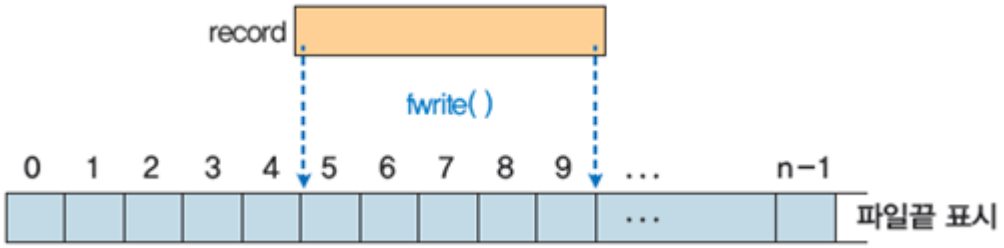
❖ 예: record 저장/읽기

```
struct student record;
FILE *fp = fopen("intfile", "wb+");
```

fwrite(), fread()

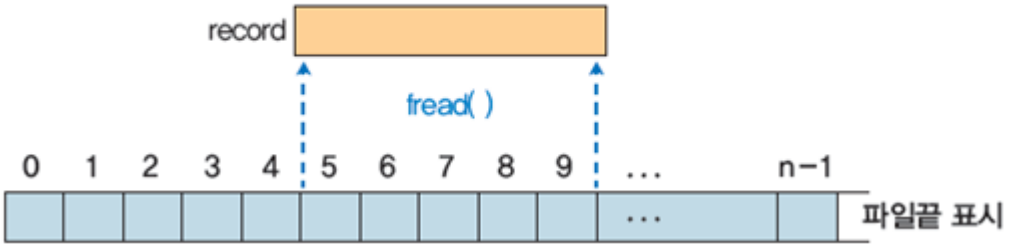
❖ fread(&record, sizeof(record), 1, fp);

- 파일에 record 구조체 쓰기



❖ fread(&record, sizeof(record), 1, fp);

- 파일로부터 record 구조체 읽기



이진 파일 입출력 모드

❖ fopen의 이진 파일 입출력 모드

모드	의미	파일이 없으면	파일이 있으면
"rb"	읽기 전용(read)	NULL 반환	정상 동작
"wb"	쓰기 전용(write)	새로 생성	기존 내용 삭제
"ab"	추가 쓰기(append)	새로 생성	기존 내용 뒤에 추가
"rb+"	읽기와 쓰기	NULL 반환	정상 동작
"wb+"	읽기와 쓰기	새로 생성	기존 내용 삭제
"ab+"	추가를 위한 읽기와 쓰기	새로 생성	기존 내용 뒤에 추가

❖ 예

- [프로그램 14-7] 구조체를 이용하여 학생 정보를 파일에 저장 (stcreate1.c)
  - 학습 point: 이진 파일 출력 함수인 fwrite 함수를 활용하여 학생 정보를 파일에 저장할 수 있다.
- [프로그램 14-8] 파일에 저장된 모든 학생 정보를 읽어서 출력(stprint.c)
  - 학습 point: 이진 파일 입력 함수인 fread 함수를 활용하여 학생 정보를 파일로 부터 읽어서 출력할 수 있다.

프로그램 11-7

```c
#include <stdio.h>
#include "student.h"

int main(int argc, char* argv[])
{
    struct student record;
    FILE *fp;

    if (argc != 2) {
        fprintf(stderr, "사용법: %s 파일이름\n", argv[0]);
        exit(1);
    }

    fp = fopen(argv[1], "wb");
    printf("%s %7s %6s\n", "학번",  "이름",  "점수");
    while (scanf("%d %s %d", &record.id, record.name, &record.score) == 3)
        fwrite(&record, sizeof(record), 1, fp);

    fclose(fp);
    exit(0);
}
```

프로그램 11-8

```c
#include <stdio.h>
#include "student.h"

int main(int argc, char* argv[])
{
   struct student record;
   FILE *fp;

   if (argc != 2) {
      fprintf(stderr, "사용법: %s 파일이름\n", argv[0]);
      return 1;
   }

   if ((fp = fopen(argv[1], "rb")) == NULL ) {
      fprintf(stderr, "파일 열기 오류\n");
      return 2;
   }

   printf("----------------------------------\n");
   printf("%s %7s %6s\n", "학번", "이름", "점수");
   printf("----------------------------------\n");

   while (fread(&record, sizeof(record), 1, fp) > 0)
      if (record.id != 0)
         printf("%d %s %d\n", record.id, record.name, record.score);

   printf("----------------------------------\n");
   fclose(fp);
   return 0;
}
```

임의 접근 파일 처리

파일 내 위치

- 파일 위치(file position)
  - 열린 파일에서 다음 읽거나 기록할 파일 내 위치
- 파일 위치 지정자(file position indicator)
  - 시스템 내에 그 파일의 파일 위치를 저장하고 있다.

파일 위치 관련 함수 1

- fseek(FILE *fp, long offset, int mode)
  - 파일 위치 지정자를 임의로 설정할 수 있는 함수이다.
- rewind(FILE *fp)
  - 파일 위치를 파일 시작점에 위치시켜 처음부터 다시 읽을 수 있도록
- ftell(FILE *fp)
  - 파일의 현재 파일 위치를 나타내는 파일 위치 지정자 값 리턴

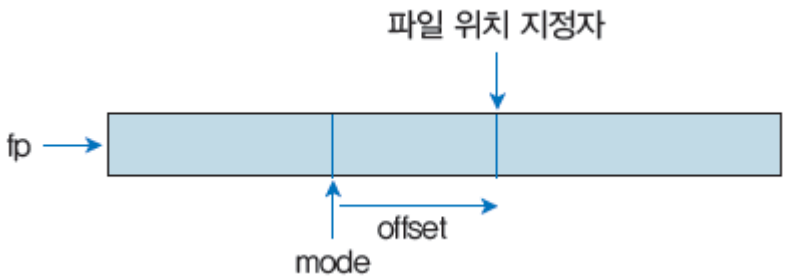
fseek()

- fseek(FILE *fp, long offset, int mode)
  - FILE 포인터 fp가 가리키고 파일의 파일 위치를
  - 모드(mode) 기준으로 오프셋(offset)만큼 옮긴다.

- fseek 함수의 모드

기호	값	의미
SEEK_SET	0	파일시작
SEEK_CUR	1	현재 위치
SEEK_END	2	파일끝

- fseek 함수의 역할



fseek() 사용 예 (1/2)

❖ 예1

- fseek(fp, 0L, SEEK_SET) 파일처음으로 이동
- fseek(fp, 100L, SEEK_CUR) 현재 위치에서 100 바이트 우로 이동
- fseek(fp, 0L, SEEK_END) 파일 끝으로 이동

❖ 예2

- fseek(fp, n * sizeof(record), SEEK_SET)

  n+1번째 레코드 시작위치로 이동
- fseek(fp, sizeof(record), SEEK_CUR)

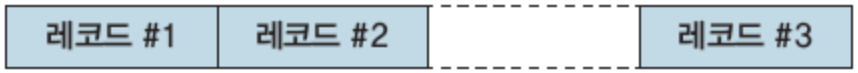
  다음 레코드 시작위치로 이동
- fseek(fp, –sizeof(record), SEEK_CUR)

  전 레코드 시작위치로 이동

fseek() 사용 예 (2/2)

❖ 예3

- fwrite(&record1, sizeof(record), 1, fp);
- fwrite(&record2, sizeof(record), 1, fp);
- fseek(fp, sizeof(record), SEEK_END);
- fwrite(&record3, sizeof(record), 1, fp);



파일 위치 관련 함수 2

- rewind(FILE *fp)
  - 파일 위치를 파일 시작점에 위치시켜 처음부터 다시 읽을 수 있도록 한다.
- ftell(FILE *fp)
  - 파일의 현재 파일 위치를 나타내는 파일 위치 지정자 값 리턴

예제 프로그램

- [프로그램 14-9] 구조체를 이용하여 학생 정보를 파일에 저장 (stcreate2.c)
  - 학습 point: 이진 파일 출력 함수인 fwrite 함수와 임의 접근 파일 함수인 fseek을 활용하여 학생 정보를 파일에 저장할 수 있다.
- [프로그램 14-10] 파일에 저장된 특정 학생의 정보를 검색해서 출력 (stquery.c)
  - 학습 point: 이진 파일 입력 함수인 fread 함수와 임의 접근 파일 함수인 fseek을 활용하여 학생 정보를 파일를 검색할 수 있다.

프로그램 11-9

```c
#include <stdio.h>
#include "student.h"
#define START_ID 1401001

int main(int argc, char* argv[])
{
    struct student record;
    FILE *fp;

    if (argc != 2) {
        fprintf(stderr, "사용법: %s 파일이름\n", argv[0]);
        exit(1);
    }

    fp = fopen(argv[1], "wb");

    printf("%s %7s %6s\n", "학번", "  ", "점수");
    while (scanf("%d %s %d", &record.id, record.name, &record.score) == 3) {
        fseek(fp, (record.id - STARTID)*sizeof(record), SEEK_SET);
        fwrite(&record, sizeof(record), 1, fp);
    }

    fclose(fp);
    exit(0);
}
```

fseak과 fwrite 함수를 이용한 record field의 파일 출력

프로그램 11-10

```c
#include <stdio.h>
#include "student.h"
#define START_ID 1301001

int main(int argc, char *argv[])
{
    struct student record;
    char c;
    int id;
    FILE *fp;

    if (argc != 2) {
        fprintf(stderr, "사용법: %s 파일이름\n", argv[0]);
        return 1;
    }

    if ((fp = fopen(argv[1], "rb")) == NULL ) {
        fprintf(stderr, "파일 열기 오류\n");
        return 2;
    }

    do {
        printf("검색할 학생의 학번 입력: ");
        if (scanf("%d", &id) == 1) {
            fseek(fp, (id - START_ID) * sizeof(record), SEEK_SET);
            if ((fread(&record, sizeof(record), 1, fp) > 0) &&
                (record.id != 0))
                printf("학번: %8d 이름: %4s 점수: %4d\n",
                        record.id, record.name, record.score);
            else printf("레코드 %d 없음\n", id);
        }
        else printf("입력 오류");

        printf("계속하겠습니까?(Y/N)");
        scanf(" %c", &c);
    } while (c == 'Y');

    fclose(fp);
    return 0;
}
```

fseak과 fread 함수를 이용한 record field 에 대한 임의 접근과 검색

프로그램 실습

❖ 전화번호 관리 프로그램에서 전화번호를 phonebook.txt 파일에 저장하도록 수정하시오.

- 프로그램 Tip
  - 프로그램이 시작될 때 맨 먼저 book 배열을phonebook.txt 파일에서 읽은 정보로 초기화하고 실행을 끝내기 전에 phonebook.txt 파일을 book 배열의 내용으로 새로 쓰는 작업을 추가해야 함
  - 그러나 이 방법은 등록이나 제거 작업을 한 후 프로그램이 정상적으로 종료되지 않고 갑자기 중단되는 경우 등록과 제거 작업이 phonebook.txt 파일에 전혀 반영되지 않음
  - 이를 해결하려면 1. 등록 기능을 실행할 때 마다 phonebook.txt 파일을 추가모드로 열어서 새로 등록한 정보를 파일 끝에 추가하며, 4. 제거 기능을 실행할 때마다 book 배열에서 정보를 제거한 후 phonebook.txt 파일을 쓰기모드로 열어서 book 배열의 내용으로 새로 기록해야 함


Q&A