작성자 : 김기희 (cupid072@naver.com)

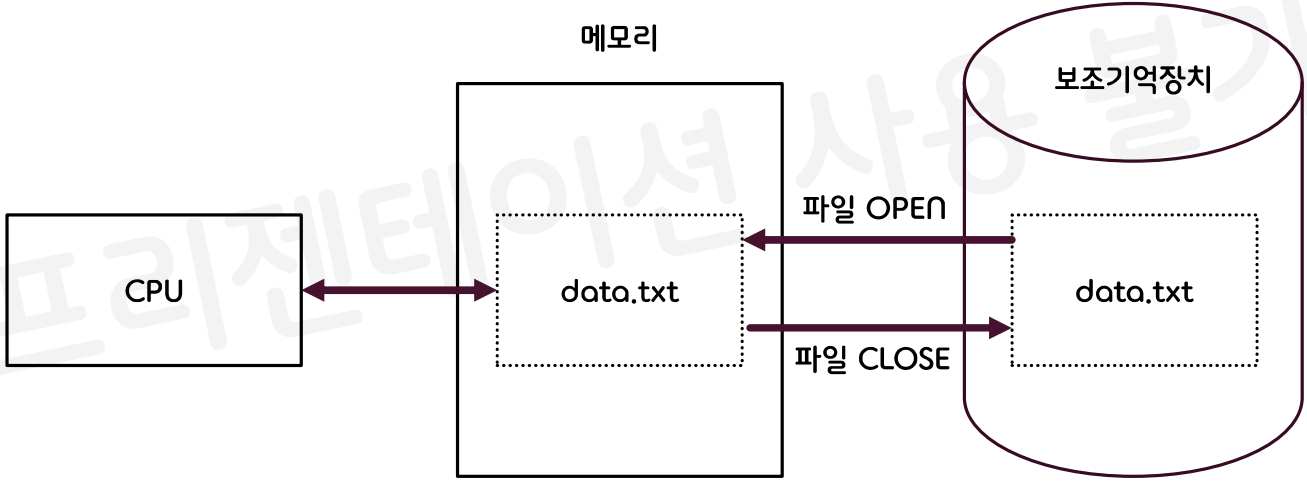
파일 입출력 (File I/O)

- 학습목표
  - 파일 개방의 의미와 파일 개방 시 수행되는 절차에 대해 설명할 수 있다.
  - 파일 폐쇄의 의미와 파일 폐쇄 시 수행되는 절차에 대해 설명할 수 있다.
  - 입출력 스트림과 입출력 버퍼에 대해 설명할 수 있다.
  - 파일의 개방 모드에 대해 설명할 수 있다.
  - 텍스트 모드와 바이너리 모드에 대해 설명할 수 있다.
  - 텍스트 파일로부터 데이터를 입출력 할 수 있다.
  - 바이너리 파일로부터 데이터를 입출력 할 수 있다.

프리젠테이션 사용 불가

파일 I/O 개념

- 보조 기억 장치내의 파일을 직접 조작하는 것은 불가능하다.
- 보조 기억 장치내의 파일 데이터를 읽어 들여 메모리에 적재(파일 OPEN)한 후 작업을 수행한다.
- 작업이 완료되면 메모리에 적재된 데이터와 파일의 연결을 종료(파일 CLOSE) 하여야 한다.



프리젠테이션 사용 불가

파일(File) 개방(open)과 폐쇄(close)

- 파일 OPEN 시 수행되는 작업
  - 운영체제로부터 파일을 식별할 수 있는 고유 번호(File Descriptor)를 부여 받는다.
  - 보조 기억 장치 내의 파일의 내용을 입출력(I/O) 할 수 있는 File stream이 생성된다.
  - 파일 구조체를 생성하여 파일의 모든 정보를 저장한 후 구조체 포인터를 반환한다.

- 파일 CLOSE 시 수행되는 작업
  - File stream의 내용을 파일에 저장한다.
  - 운영체제에 File Descriptor를 반환한다.
  - File stream을 삭제한다.
  - 파일 구조체를 메모리에서 삭제한다.

프리젠테이션 사용 불가

File Descriptor (파일 기술자)

- File Descriptor
  - 메모리에 적재된 파일 데이터를 구분하기 위한 고유 번호이다.
  - File Descriptor 테이블을 사용하여 관리되며 운영체제에 의해 관리된다.
  - 일부 File Descriptor는 시스템에 의해 예약되어 있다.

장치명	장치	고유번호	구조체 포인터 명	선언
표준입력	console	0	stdin	#define stdin (__acrt_iob_func(0))
표준출력	console	1	stdout	#define stdout (__acrt_iob_func(1))
에러출력	console	2	stderr	#define stderr (__acrt_iob_func(2))

입출력 스트림 (I/O stream)

- 입출력 스트림은 장치로부터 데이터를 읽어 들이거나 장치로 데이터를 내보내기 위해 사용되는 데이터 입출력 버스이다.

- 입출력 스트림은 단방향 이며 입력 스트림과 출력 스트림으로 구분된다.
  - 입력 스트림 : 장치로부터 데이터를 읽어 들이기 위해 사용되는 스트림
  - 출력 스트림 : 장치로 데이터를 내보내기 위해 사용되는 스트림

- 선입선출(FIFO) 구조이다.

- 입출력 스트림 자체는 데이터를 저장하는 기능이 없으나 입출력 버퍼(buffer)를 사용하면 스트림에 데이터를 저장할 수 있다.

프리젠테이션 사용 불가

스트림 버퍼(Stream Buffer)

- 스트림 버퍼(Stream Buffer)는 입출력 스트림의 데이터를 일시적으로 저장할 수 있는 기능을 제공한다.

- 스트림 버퍼(Stream Buffer)를 사용하면 I/O 성능을 높일 수 있다.
  - 입력시 : 데이터를 미리 읽어 들여 버퍼에 저장한 후 버퍼로부터 읽어 들인다.
  - 출력시 : 데이터를 버퍼에 저장해 두었다가 한 번에 버퍼의 데이터를 내보낸다.

- C에서 사용 가능한 버퍼의 종류는 다음과 같다. (운영체제가 지원하는 경우)

버퍼종류	상수 값	설명
FULL BUFFER	int __IOFBF	버퍼가 가득 찼을 경우 flush
LINE BUFFER	int __IOLBF	라인 넘김 입력 시, 버퍼가 가득 찼을 경우 flush
NO BUFFER	int __IONBF	버퍼를 사용하지 않음

인터넷 사용 불가

파일 개방 (File open)

- fopen() 함수

  FILE * fopen (const char *filename, const char *opentype)

  - filename : 개방할 파일 경로
  - opentype : 개방모드

opentype	모드	기존파일		커서위치	커서제어
		있을 시	없을 시		
r	읽기 전용	개방	에러	선두	불가
w	쓰기 전용	생성	삭제 후 생성	선두	불가
a	추가 전용	생성	개방	마지막	불가
r+	읽기/쓰기	개방	에러	선두	가능
w+	읽기/쓰기	생성	삭제 후 생성	선두	가능
a+	읽기/쓰기	생성	개방	마지막	가능
t	문자 입출력 모드로 개방				
b	이진 데이터 입출력 모드로 개방				

파일 개방 (File open)

- 파일 입출력을 위해 파일을 개방한다.

  FILE * freopen (const char *filename, const char *opentype, FILE *stream)

  - filename : 개방할 파일 경로
  - opentype : 개방모드
  - stream : 다시 개방할 파일 구조체 포인터

- freopen() 함수는 현재 stream과 연관된 파일을 닫고 filename에서 지정된 파일에 stream을 재지정한다.

- freopen() 함수를 사용하면 표준 입력을 파일로 연결할 수 도 있다.

```
freopen("개방파일경로", "r", stdin);      // stdin 파일 구조체를 파일로 연결
scanf("%s", line);                        // 파일로 부터 라인을 읽어들인다.
```

프리젠테이션 사용 불가

파일 폐쇄(File Close)

- 개방된 파일을 닫는다.

  int fclose(FILE* stream)

  - stream : 닫을 파일의 파일 구조체 포인터
  - 반환 값 : 성공 시 0, 실패 시 : EOR

- stream과 연관된 파일을 닫는다.
- 스트림을 닫기 전 스트림에 지정된 버퍼를 모두 삭제한 후 버퍼를 해제한다.

프리젠테이션 사용 불가

fgets()

- 파일로부터 문자열을 입력받는다.
  char* fgets (char *s, int count, FILE *stream)

- 파라미터
  s : 읽어 들인 문자 저장 주소
  count : 읽어 들일 문자의 수
  stream: 데이터를 읽어 들일 파일 구조체 포인터

- 반환값
  성공 : 읽어 들인 문자가 저장된 주소
  실패 : NULL

- fgets() 함수는 fp의 현재 커서로부터 다음 개행 문자('\n')까지 읽어 들여 s가 가리키는 곳에 저장한다.
- 만약 현재 커서로부터 다음 개행 문자까지의 길이가 count – 1 을 초과하는 경우 count – 1 만큼의 문자를 읽어 들여 s가 가리키는 곳에 저장한다.
- count – 1 값 만큼을 읽어 들이는 이유는 마지막에 NULL 문자('\0')를 추가하기 때문이다.
- fp 내의 커서가 파일의 끝이거나 파일을 읽어 들이는 도중 에러가 발생하는 경우 NULL을 반환한다.

프리젠테이션 사용 불가

fgets() 예제

```
#include <stdio.h>
#include <string.h>
#define BUFFER_SIZE 1024                    // 파일을 읽어 들이기 위한 문자 배열의 크기

int main() {
  FILE* fp = fopen("data.txt", "r");        // 현재 디렉토리의 data.txt 파일을 읽기 전용으로 open
  char buffer[BUFFER_SIZE];                 // 파일로 부터 읽어 들인 문자열을 저장하기 위한 배열
  if (fp != NULL) {
        while (1) {
              memset(buffer, 0, sizeof(buffer));      // 문자열을 저장하기 위한 배열의 모든 원소의 값을 0으로
              char *pos = fgets(buffer, sizeof(buffer), fp);  // fp로부터 라인을 읽어 들여 문자 배열에 저장한다.
              if (pos == NULL) break;                 // fp로부터 읽어 들인 데이터가 있는가를 평가한다.
              printf("%s", buffer);                   // 읽어 들인 문자열을 출력한다.
        }
        fclose(fp);
  }
  return 0;
}
```

프리젠테이션 사용 불가

fputs()

- 문자열을 파일로 출력한다.

  int fputs (const char *s, FILE *stream)

- 파라미터

  s　　: 출력 문자열
  stream　: 데이터를 출력할 파일 구조체 포인터

- 반환값

  성공시　: 음수가 아닌 값
  실패시　: EOF

- fputs() 함수는 인수 s가 가리키는 문자열을 fp가 가리키는 파일로 출력한다.
- fputs() 함수는 NULL 문자는 내보내지 않는다.
- fputs() 함수는 개행문자('\n')를 추가하지 않는다.

프리젠테이션 사용 불가

fputs() 예제

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

int main() {
	char user[][20] = { "홍길동", "이순신", "강감찬", "프로그래머" };
	FILE* fp = fopen("data.txt", "w");                    // 파일을 쓰기 전용으로 open
	if (fp != NULL) {                                     // 파일 open 검사
		for (int i=0 ; i<sizeof(user)/sizeof(user[0]) ; i++) {
			int out = fputs(user[i], fp);                 // 배열 원소의 값을 파일로 내보내기
		}
		fclose(fp);                                       // 파일 close
	}
	return 0;
}
```

프리젠테이션 사용 불가

fprintf()

- 출력 서식을 사용하여 형식화된 포멧으로 데이터를 출력한다.

  int fprintf (FILE *stream, const char *template, ...)

- 파라미터

  stream : 문자열을 내보낼 FILE 구조체 포인터
  template : 출력 서식 문자열
  ..... : 출력 서식 대체 값

- 반환값

  성공시 : 출력 문자의 수
  실패시 : 음의 값

- fprintf() 함수는 일련의 문자와 값을 서식화 하여 파일로 출력한다.

프리젠테이션 사용 불가

fprintf() 예제

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

int main() {
	char user[][20] = { "홍길동", "이순신", "강감찬", "프로그래머", "ABC" };
	FILE* fp = fopen("data.txt", "w");                    // 파일을 쓰기 전용으로 개방

	if (fp != NULL) {                                     // 파일 open 검사
		for (int i=0 ; i<sizeof(user)/sizeof(user[0]) ; i++) {
			int count = fprintf(fp, "회원이름 : %s\n", user[i]);   // 파일로 이름 출력
			printf("출력 문자 수 : %d\n", count);
		}
		fclose(fp);
	}
	return 0;
}
```

프리젠테이션 사용 불가

fscanf()

- 형식화된 입력 서식을 사용하여 파일로 부터 데이터를 읽어 들인다.

  int fscanf (FILE *stream, const char *template, ...)

- 파라미터

  stream : 데이터를 읽어 들일 파일 구조체 포인터
  template : 입력 서식 문자열
  ... : 입력 값을 저장할 주소

- 반환값

  성공시 : 성공적으로 입력한 항목의 수
  실패시 : EOF

프리젠테이션 사용 불가

fscanf() 예제

```
#include <stdio.h>

int main() {
	char name[20];                                          // 문자열을 입력 받기 위한 문자배열
	int age;                                                // 정수 값을 입력 받기 위한 변수

	FILE* fp = fopen("data.txt", "r");                      // 파일을 읽기 전용으로 open
	if (fp != NULL) {
		while (1) {
			int readCount = fscanf(fp, "%s %d", name, &age);    // 파일로 부터 문자열과 정수 값을 읽어 들인다.
			if (readCount != 2) break;
			printf("이름 : %s, 연령 : %d\n", name, age);        // 읽어 들인 값 출력
		}
		fclose(fp);                                         // 파일 close
	}
	return 0;
}
```

프리젠테이션 사용 불가

fread()

- fread() 함수는 파일로부터 데이터를 배열로 읽어 들인다.

  size_t fread (void *data, size_t size, size_t count, FILE *stream)

- 파라미터

  data : 읽어 들인 내용을 저장할 위치를 가리키는 포인터
  size : 데이터 블록 하나의 크기
  count : 데이터 블록의 수
  stream : 데이터를 읽어 들일 파일 구조체 포인터

- 반환값

  성공시 : 읽어 들인 데이터 블록의 수

프리젠테이션 사용 불가

fread() 예제

```
#include <stdio.h>
#include <stdlib.h>
extern int filesize(char*);

int main() {
	int size = filesize("data.txt");
	char* buffer = (char*)calloc(sizeof(char), size);
	if (buffer != NULL) {
		FILE* fp = fopen("data.txt", "r");
		if (fp != NULL) {
			fread(buffer, sizeof(char), size, fp);
			printf("%s", buffer);
			fclose(fp);
		}
		free(buffer);
	}
	return 0;
}
```

```
#include <sys/stat.h>

int filesize(const char* file) {
	struct stat file_info;
	if (0 > stat(file, &file_info)) {
		return -1;
	}
	return file_info.st_size;
}
```

프리젠테이션 사용불가

fwrite()

- fwrite() 함수는 배열 형태의 데이터를 파일로 출력한다.

  size_t fwrite (const void *data, size_t size, size_t count, FILE *stream)

- 파라미터

  data : 출력할 내용을 가리키는 포인터
  size : 블록 하나의 크기
  count : 출력할 블록의 수
  stream : 출력할 파일 구조체 포인터

- 반환값

  정상적으로 출력한 데이터 블록의 수

프레젠테이션 사용 불가

fwrite() 예제

```
#include <stdio.h>

int main() {
    int data[] = { 100, 200, 300, 400 };
    FILE* fp = fopen("data.txt", "w");
    if (fp != NULL) {
        size_t count = fwrite(data, sizeof(data)/sizeof(data[0]), sizeof(int), fp);
        printf("%I64u\n", count);
        fclose(fp);
    }
}
```

☞ 위 예제에 의해 출력된 파일로 부터 데이터를 배열로 읽어 들이는 코드를 작성해 보세요.

프리젠테이션 사용 불가

fseek()

- 파일 개방 후 파일 내부의 커서 위치를 제어한다.

  int fseek (FILE *stream, long int offset, int whence)

- 파라미터

  stream : 커서를 제어할 파일 구조체 포인터
  offset : 상대거리
  whence : 기준위치 (SEEK_SET, SEEK_CUR, SEEK_END 중의 하나)

- 반환값

  성공시 : 0
  실패시 : 0 이 외의 값

- 파일로부터 데이터를 읽거나 쓰기 또는 fseek() 함수를 사용하여 커서를 제어하기 이전에는 커서의 위치를 구할 수 없다.

프리젠테이션 사용 불가

rewind()

- 파일 내부의 커서를 파일의 선두로 이동시킨다.

  void rewind (FILE *stream)

- 파라미터

  stream : 커서를 제어할 파일 구조체 포인터

프리젠테이션 사용 불가

ftell()

- 파일 내부의 현재 커서의 위치를 조사한다.

  long int ftell (FILE *stream)

- 파라미터

  stream : 커서 위치를 조사할 파일 구조체 포인터

- 반환값

  성공시 : 파일 선두로부터의 커서 위치 값

  실패시 : -1L

- 파일 개방 직후에는 파일 내부 커서의 위치를 알 수 없으며 내부 커서를 이동 시키는 작업을 수행하고 난 후이어야 커서의 위치를 조사할 수 있다.
- 내부 커서를 이동시키는 작업은 파일 데이터의 입/출력, fseek(), rewind() 함수 호출에 의한 커서 제어 등이 있다.

프리젠테이션 사용 불가