2주차 3차시 임시 파일 입출력

【학습목표】

1. 임시 파일을 이용한 파일 입출력을 설명할 수 있다.
2. 파일 디스크립터/파일 포인터를 활용하여 파일 입출력을 할 수 있다.

학습내용1 : 파일 디스크립터/파일 포인터

1. 파일 기술자와 파일 포인터 간 변환

* 상호 변환

기능	함수원형
파일 기술자 → 파일 포인터	FILE *fdopen(int fildes, const char *mode);
파일 포인터 → 파일 기술자	int fileno(FILE *stream);

- 파일 기술자에서 파일 포인터 생성 : fdopen(3) 함수 사용
- 파일포인터에서 파일 기술자 정보 추출 : fileno(3) 함수 사용

* 파일 기술자에서 파일 포인터 생성
- fdopen(3)
성공 시 : 파일 포인터 리턴
실패 시 : NULL 리턴

```
#include <stdio.h>
FILE *fdopen(int fildes, const char *mode);
```

fildes : 파일 기술자
mode : 열기 모드

fdopen() 함수 사용

```
01  #include <fcntl.h>
02  #include <stdlib.h>
03  #include <stdio.h>
04
05  int main(void) {
06      FILE *fp;
07      int fd;
08      char str[BUFSIZ];
09
10      fd = open("unix.txt", O_RDONLY);
11      if (fd == -1) {
12          perror("open");
13          exit(1);
14      }
15
16      fp = fdopen(fd, "r");
17
18      fgets(str, BUFSIZ, fp);
19      printf("Read : %s\n", str);
20
21      fclose(fp);
22
23      return 0;
24  }
```

저수준 파일입출력 함수로 파일 오픈

파일 포인터 생성

고수준 파일읽기 함수로 읽기

```
# ex2_18.out
Read : Unix System Programming
```

- fileno(3)

```
#include <stdio.h>
int fileno(FILE *stream);
```

stream : 파일 포인터

fileno()함수 사용하기

```
01  #include <unistd.h>
02  #include <fcntl.h>
03  #include <stdlib.h>
04  #include <stdio.h>
05
06  int main(void) {
07      FILE *fp;
08      int fd, n;
09      char str[BUFSIZ];
10
11      fp = fopen("unix.txt", "r");
12      if (fp == NULL) {
13          perror("fopen");
14          exit(1);
15      }
16
17      fd = fileno(fp);
18      printf("fd : %d\n", fd);
19
20      n = read(fd, str, BUFSIZ);
21      str[n] = '\0';
22      printf("Read : %s\n", str);22
23
24      close(fd);
25
26      return 0;
27  }
```

고수준 파일입출력 함수로 파일 오픈

파일 기술자 리턴

저수준 파일읽기 함수로 읽기

```
# ex2_19.out
fd : 3
Read : Unix System Programming
```

학습내용2 : 임시 파일 입출력

1. 임시 파일명 생성

임시 파일명이 중복되지 않도록 임시 파일명 생성
tmpnam(3)
임시 파일명을 시스템이 알아서 생성
인자 있을 경우 해당 인자가 가르키는 곳에 임시 파일명 저장
인자가 NULL일 경우 임시파일명을 리턴

```
#include <stdio.h>
char *tmpnam(char *s);
```

s : 파일명을 저장할 버퍼의 시작 주소

tempnam(3)
임시 파일명에 사용할 디렉터리와 접두어 지정하여 임시파일명 리턴
접두어는 5글자만 허용

```
#include <stdio.h>
char *tempnam(const char *dir, const char *pfx);
```

dir : 임시 파일명의 디렉터리
pfx : 임시 파일명의 접두어

mktemp(3)
인자로 임시 파일의 텟플릿을 받아 이를 임시 파일명으로 리턴
텟플릿은 대문자 'X'6개로 마쳐야 함.

```
#include <stdlib.h>
char *mktemp(char *template);
```

template : 임시 파일명의 텟플릿

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

int main(void) {
    char *fname;
    char fntmp[BUFSIZ];
    char template[32];

    fname = tmpnam(NULL);
    printf("1. TMP File Name(tmpnam) : %s\n", fname);

    tmpnam(fntmp);
    printf("2. TMP File Name(tmpnam) : %s\n", fntmp);

    fname = tempnam("/tmp", "hanbit");
    printf("3. TMP File Name(tempnam) : %s\n", fname);

    strcpy(template, "/tmp/hanbitXXXXXX");
    fname = mktemp(template);
    printf("4. TMP File Name(mktemp) : %s\n", fname);

    return 0;
}
```

```
# ex2_20.out
1. TMP File Name(tmpnam) : /var/tmp/aaaFUaGOe
2. TMP File Name(tmpnam) : /var/tmp/baaGUaGOe
3. TMP File Name(tempnam) : /tmp/hanbiAAAHUaGOe
4. TMP File Name(mktemp) : /tmp/hanbitIUaGOe
```

2. 임시 파일의 파일 포인터 생성

tmpfile(3)
자동으로 w+ 모드로 열린 파일 포인터를 리턴
tmpnam(), tempnam(), mktemp() 함수들을 임시파일만 생성함, 파일을 열어야 하는데 파일명을 알 필요없고 파일 포인터만 알면 됨.

```c
#include <stdio.h>
FILE *tmpfile();
```

```c
#include <stdio.h>

int main(void) {
    FILE *fp;

    fp = tmpfile();

    fputs("unix system", fp);

    fclose(fp);

    return 0;
}
```

임시 파일에 출력

【학습정리】

1. 파일은 관련 있는 데이터들의 집합으로 하드디스크 같은 저장장치에 일정한 형태로 저장된다.

2. 유닉스에서 파일은 데이터를 저장하기 위해서 뿐만 아니라 데이터를 전송하거나 장치에 접근하기 위해서도 사용한다.

3. 저수준 파일 입출력
- 파일 지시자 : int fd(파일 기술자)
- 특징 : 더 빠르고, 바이트 단위로 읽고 쓰며, 특수 파일에 대한 접근이 가능함

4. 고수준 파일 입출력
- 파일 지시자 : FILE *fp; (파일 포인터)
- 특징 : 사용하기 쉽고, 버퍼 단위로 읽고 쓰며, 데이터의 입출력 동기화가 쉽고, 여러 가지 형식을 지원함