시스템 프로그래밍

프로그래밍 과제 1

소프트웨어학과

2020301043 심보영

[1] 파일의 내용 출력하기 (show)

show.c 코드

class1@dilab: ~/2020301043/code

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>

int main(int argc, char* argv[]){
        int rfd, wfd, n;
        char buf[100];

        if( argc != 2 && argc != 3){
                printf("Error, write down 'file name'\n");
                exit(1);
        }
        rfd = open(argv[1],O_RDONLY);
        if(rfd == -1){
                perror("Fail to open");
                exit(1);
        }
        if( argc == 2) {
                while(read(rfd,buf,sizeof(buf)))
                        printf("파 일 출 력 : %s\n", buf);
                        close(rfd);
        }
        if( argc == 3) {
                wfd = open(argv[2],O_CREAT | O_WRONLY | O_TRUNC,0644);
                if( wfd == -1){
                        perror("Fail to open second file");
                        return 1;
                }
                while ((n = read(rfd, buf, sizeof(buf))) > 0){
                        write(wfd, buf, n);
                }
                close(rfd);
                close(wfd);
        }
        return 0;
}
~
~
```

show 결과 화면

```
class1@dilab:~/2020301043/code$ vi show.c
class1@dilab:~/2020301043/code$ gcc -o show show.c
class1@dilab:~/2020301043/code$ ./show
Error, write down 'file name'
class1@dilab:~/2020301043/code$ ./show data.txt
파 일 출 력 : This is a data.txt file
오 늘  날 씨 는  좋 다 .
안 하 세 요  ?
010-1234-5678
class1@dilab:~/2020301043/code$ ./show data.txt data.backup
class1@dilab:~/2020301043/code$ cat data.backup
This is a data.txt file
오 늘  날 씨 는  좋 다 .
안 하 세 요  ?
010-1234-5678 class1@dilab:~/2020301043/code$
```

<코드 설명>

- 헤더파일

표준 입출력, 입출력함수를 사용하기 위한 헤더파일 <stdio.h>

유닉스에서 사용하는 c 컴파일러 헤더파일 <unistd.h>

함수 조회와 설정을 할 수 있는 함수를 포함한 헤더파일 <fcntl.h>

문자열 변환 등 이용되지만 exit()함수를 사용하기 위해 포함한 헤더파일 <stdlib.h>

```
int main(int argc, char* argv[])
```
int argc : main()함수에 전달되는 데이터의 개수
char* argv[] : main()함수에 전달되는 실제적인 데이터로 char형 포인터 배열로 구성

```
int rfd, wfd, n;
```
읽기 전용 파일 디스크립터 변수를 표현하기 위한 변수 rfd
쓰기 전용 파일 디스크립터 변수를 표현하기 위한 변수 wfd
글자 수를 의미하는 변수 n

```
char buf[100];
```
버퍼의 크기를 100으로 지정

```
if (argc != 2 && argc != 3) {
                printf("Error, write down 'file name'\n");
                exit(1);
}
```
main함수에 전달되는 데이터의 개수가 2개나 3개가 아닐 경우 파일을 입력 받지 못한 것과 같으므로 에러문 출력 및 현재 c프로그램 완전히 종료시킴.

```
rfd = open(argv[1], O_RDONLY);
```
open() 함수를 이용하여 argv[1] 데이터 파일 열어 달라고 요청
O_RDONLY : 파일을 읽기 전용으로 연다.

```
if (rfd == -1) {
        perror("Fail to open");
        exit(1);
}
```
현재 열려 있는 파일을 구분하는 정수 값인 파일 기술자는 0,표준 입력부터 시작한다.

-1일 경우에는 파일이 제대로 읽혀 지지 않는다는 의미와 같으므로 -1일 경우 에러문 출력

`perror( )` : 존재하지 않는 파일을 읽기 모드로 열었을 경우 에러 메시지를 출력하는 함수

`exit( )` : 현재의 c프로그램 자체를 완전히 종료 시키는 기능을 가지는 함수

- ()안에 1을 넣는 이유는 에러가 발생하여 종료할 때 대체로 1를 반환

main 함수에 전달되는 데이터의 개수를 조건문을 통하여 조건을 만족할 경우 실행

- 데이터의 개수가 2개일 경우

```
if (argc == 2) {
		while (read(rfd, buf, sizeof(buf)))
			printf("파일 출력: %s\n", buf);
		close(rfd);
}
```

반복문 while() (조건)을 만족하는 ~동안

rfd파일을 버퍼의 크기만큼 버퍼에 읽어 들일 동안

read() : open 함수로 열기로 한 파일의 내용을 읽어주는 함수

buf : 파일을 읽어 들일 버퍼

sizeof(buf) : 버퍼의 크기 sizeof()함수는 괄호 안 자료형의 크기를 출력하는 함수

printf(" 파일 출력: %s\n " ,buf)

즉, rfd 파일을 버퍼의 크기만큼 버퍼에 읽어 들일 동안에, 저장된 버퍼의 내용을 출력한다.

모든 버퍼가 읽어 들여지면, close(rfd)를 통해 rfd 파일을 닫아준다.

- 데이터의 개수가 3 개일 경우

```
if (argc == 3) {
		wfd = open(argv[2], O_CREAT | O_WRONLY | O_TRUNC, 0644);
		if (wfd == -1) {
			perror("Fail to open second file");
			return 1;
		}
		while ((n = read(rfd, buf, sizeof(buf))) > 0) {
			write(wfd, buf, n);
		}
		close(rfd);
		close(wfd);
	}
```

```
wfd = open(argv[2], O_CREAT | O_WRONLY | O_TRUNC, 0644);
```

open()함수를 이용하여 argv[2] 파일을 열어준다.
oflag와 숫자를 이용하여 파일의 접근 권한을 지정
O_CREAT : 파일이 없으면 생성
O_WRONLY : 파일을 쓰기 전용으로 연다.
O_TRUNC : 파일을 생성할 때 이미 있는 파일이고, 쓰기 옵션으로 열었을 경우 내용을 모두 지우고 파일의 길이를 0으로 변경
0644 : 파일에 대한 소유자는 읽기,쓰기,그룹과 기타 사용자는 읽기 권한이 있음을 의미

```
if (wfd == -1) {
                perror("Fail to open second file");
                return 1;
        }
```

조건문 if(조건) { 실행문 } : wfd 파일이 열리지 않을 경우

-1일 경우에는 파일이 제대로 읽혀 지지 않는 의미로 이것의 부정은 파일이 읽힌다는 의미기에 조건문을 통해 파일 열림의 유무를 판단한다.

perror() : 오류 상황을 표준 오류 스트림에 출력하기 위한 함수 -> 에러문 출력

return 1; : 1 의 값을 반환함으로써 해당 함수 종료

```
while ((n = read(rfd, buf, sizeof(buf))) > 0) {
                        write(wfd, buf, n);
}
```

반복문 while() (조건)을 만족하는 ~동안
읽어 들인 버퍼의 수가 0보다 큰 동안에 write()함수를 통해 파일의 내용을 wfd 파일에 작성

read( ) : open()함수로 열기로 한 파일의 내용을 읽는 함수

rfd : 읽을 파일의 파일 디스크립터

buf : 파일을 읽어 들일 버퍼

sizeof(buf) : 버퍼의 크기 sizeof() 함수는 괄호 안에 자료형의 크기를 출력하는 함수

즉, rfd 파일을 버퍼의 크기만큼 버퍼에 읽어 들인다.

write() : 파일 디스크립터를 이용하여 참조한 파일에 데이터를 쓴다.

(wfd, buf, n) : buf가 가리키는 메모리에서 n으로 지정한 크기만큼 wfd 파일에 기록

```
close(rfd);
```

rfd 파일을 닫아준다.

```
close(wfd);
```

wfd 파일을 닫아준다.

```
return 0;
```

main 함수를 끝낸 후 벗어나기 위하여 0을 return한다.

[2] 파일에서 검색하기 (ff)

ff.c 코드

class1@dilab: ~/2020301043/code

```
#include <stdio.h>
#include <fcntl.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>

int main(void){
        int rfd,n;
        char buf[256];
        off_t start, cur;
        int count_n = 0;

        rfd = open("data2.txt",O_RDONLY);
        if(rfd == -1){
                perror("Open data2.txt");
                exit(1);
        }
        start = lseek(rfd, 0, SEEK_SET);
        n = read(rfd,buf, sizeof(buf));
        for(n = 0; n <strlen(buf); n++){
                if(buf[n]== 'a'){
                        count_n++;
                }
        }
        buf[n] = '\0';
        printf("count word =%d\n",count_n);
        close(rfd);
        return 0;
}
```

ff 실행 결과 화면

```
class1@dilab:~/2020301043/code$ vi ff.c
class1@dilab:~/2020301043/code$ gcc -o ff ff.c
class1@dilab:~/2020301043/code$ ./ff
count word =4
class1@dilab:~/2020301043/code$ cat data2.txt
abcdefg~hijklmnop!
가 나 다 라 마 바 사 아 자 차 카 타 파 하
123456789
This is a data2.txt file.
Welcome~! class1@dilab:~/2020301043/code$
```

<코드 설명>

- 헤더파일

표준 입출력, 입출력함수를 사용하기 위한 헤더파일 <stdio.h>

유닉스에서 사용하는 c 컴파일러 헤더파일 <unistd.h>

함수 조회와 설정을 할 수 있는 함수를 포함한 헤더파일 <fcntl.h>

문자열 변환 등 이용되지만 exit()함수를 사용하기 위해 포함한 헤더파일 <stdlib.h>

c형식 문자열을 다룰 수 있는 함수를 포함한 헤더파일로 strlen()함수를 위해 포함 <string.h>

```
int main(void)
```

```
int rfd, n;
```

읽기전용 파일 디스크립터 변수를 표현하기 위한 변수 rfd

글자 수를 의미하는 변수 n

```
char buf[256];
```
버퍼의 크기를 256으로 지정

```
off_t start, cur;
```
off_t를 이용하여 파일 오프셋을 이동시킬 기준 값을 지정

```
int count_n =0;
```
찾은 검색어의 횟수를 나타내기 위한 count_n 변수로, 0으로 초기화 시켜준다.

```
rfd = open("data2.txt", O_RDONLY);
```

`open( )` 함수를 이용하여 txt파일을 열어 달라고 요청

`O_RDONLY` 파일을 읽기 전용으로 연다.

```
if (rfd == -1) {
        perror("Open data2.txt");
        exit(1);
}
```

현재 열려 있는 파일을 구분하는 정수 값인 파일 기술자는 0 ‘ 표준 입력 ’ 부터 시작한다.

-1일 경우에는 파일이 제대로 읽혀 지지 않는다는 의미와 같으므로 -1일 경우 에러문 출력

`perror( )` : 존재하지 않는 파일을 읽기 모드로 열었을 경우 에러 메시지를 출력하는 함수이다.
`exit( )` : 현재의 c프로그램 자체를 완전히 종료 시키는 기능을 가지는 함수

- ( )안에 1을 넣는 이유는 에러가 발생하여 종료할 때 대체로 1를 반환한다.

```
start = lseek(rfd, 0, SEEK_SET);
```

`lseek( )` : 함수의 seek pointer 커서를 조정하는 함수로,

rfd = 조정할 파일의 파일 디스크립터

0 = off_t offset으로 기준점으로부터 이동할 거리

SEEK_SET = 파일의 맨 앞으로 기준점을 지정해준다

즉 rfd 파일의 맨 앞으로 커서를 조정한다.

```
n = read(rfd, buf, sizeof(buf));
```

read( ) : open()함수로 열기로 한 파일의 내용을 읽는 함수

rfd = 읽을 파일의 파일 디스크립터

buf = 파일을 읽어 들일 버퍼

sizeof(buf) = 버퍼의 크기 sizeof( ) 함수는 괄호 안에 자료형의 크기를 출력하는 함수

즉, rfd 파일을 버퍼의 크기만큼 버퍼에 읽어 들인다.

```
for (n = 0; n < strlen(buf); n++) {
        if (buf[n] == 'a') {
                count_n++;
        }
 }
```

반복문 for (범위) { 실행문 }

: n이 문자열의 길이만큼 반복문을 도는 동안 만약 ‘ a ’ 라는 검색어가 나올 경우

strlen( ) : c언어 스타일의 문자열을 받아 그것의 길이를 반환하는 함수

증감연산자 ++ : 값을 1 증가시킨 후 연산을 진행시키는 연산자

즉, n이 문자열의 길이만큼 반복문을 도는 동안 만약 ‘ a ’ 라는 검색어 나올 경우 count_n 증감.

```
buf[n] = '\0';
```

문자열 제일 끝을 표현해주기 위해 ‘ \0 ’ 을 넣어서 표현한다.

```
printf("find word = % d\n ”, count_n);
```

반복문을 통해 찾은 검색어의 수 출력.

```
close(rfd);
```

열 수 있는 파일의 개수는 제한 되어있으므로 파일을 닫아준다.

```
return 0;
```

main함수를 끝낸 후 벗어나기 위한 return