

TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com

학생 – GJ (박현우)
uc820@naver.com

1. 시스템 프로그래밍 - 1

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

#define ERROR -1

int main(void){

    int filedес;
    char pathname[] = "temp.txt";

    if( (filedes = open(pathname, O_CREAT | O_RDWR | O_EXCL, 0644)) == ERROR){

        printf("File Open Error\n");
        exit(1);
        //제 불손, 데니스 리치
        // 모든 것은 파일이다!!!
    }

    printf("fd = %d\n",filedes);

    close(filedes);
    return 0;

}
```

■ 코드 이해

Open()의 return값이 filedес로 들어가 숫자 3이 나온다.

숫자 3은 파일 인덱스 위치정보를 나타낸다.

0은 표준입력
1은 표준출력
2는 표준에러
숫자 3은 filedес가 가리키는 값이다.

■ open()함수

int open(const char *path, int flags, mode)

1번째 인자 : 열어야 하는 파일의 경로

2번째 인자 : 파일을 열 때 사용할 수 있는 플래그 값

ex) O_RDONLY : 읽기 전용(0)
O_WRONLY : 쓰기 전용(1)
O_RDWR : 읽기/쓰기 모두 가능(2)
O_APPEND : 쓰기 작업 수행 시 파일의 끝에 새로운 내용을 추가
O_CREAT : 파일이 없을 경우 파일을 생성
O_EXCL : 파일이 있는 경우에 에러를 발생시킨다.
O_TRUNC : 기존 파일내용이 있으면 지운다.
O_NONBLOCK : 년블로킹모드로 전환한다.
O_SYNC : 쓰기 연산마다 버퍼를 사용하지 않고 변경된 내용을 바로 디스크에 저장한다.

3번째 인자 : 생략 가능한 인자로 파일 생성시의 접근권한을 설정할 수 있다.

1. 시스템 프로그래밍 - 2

```
#include<fcntl.h>

int main(void){

    int filedes1, filedes2;

    filedes1 = open("data1.txt", O_WRONLY | O_CREAT | O_TRUNC, 0644);
    filedes2 = creat("data2.txt", 0644);
    // open 말고 creat로도 파일을 만들 수 있다.
    close(filedes1);
    close(filedes2);
    // 파일을 열었으니 close로 닫아야 한다.
    return 0;
}
```

1. 시스템 프로그래밍 - 3

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

int main(void){

    int fdin, fdout;
    ssize_t nread;
    char buf[1024];

    fdin = open("temp1.txt", O_RDONLY);
    fdout = open("temp2.txt", O_WRONLY | O_CREAT | O_TRUNC, 0644);

    while( (nread = read(fdin, buf, 1024)) > 0 ){

        if(write(fdout, buf, nread) < nread){

            close(fdin);
            close(fdout);

        }

        close(fdin);
        close(fdout);

    }

}
```

■ 코드 이해

Fdin에는 기존에 있는 temp1.txt파일을 읽기 형식으로 open()함수 값을 리턴하고
Fdout에는 temp2.txt파일을 쓰기용으로 새로 만들어 open()함수 값을 리턴한다.

While루프에서 read함수로 fdin에서 읽어온 값을 buf에 최대 1024바이트 읽어와 저장할 수 있다.

Read에서 리턴한 값이 nread에 들어가고 write함수로 buf에 있는 값을 fdout에 nread 크기 만큼 쓰여진다.
즉, cp의 기능을 소프트웨어 인터럽트 방식으로 구현한 것이다.

■ read()함수

ssize_t read(int fd, void *buf, size_t count)

1번째 인자 : 파일 식별자

2번째 인자 : 읽은 데이터를 저장하는 버퍼의 포인터형 변수

3번째 인자 : 한번에 읽을 수 있는 데이터의 최대 바이트 수

return값으로는 읽어들인 데이터의 바이트 수를 long int형으로 리턴함.

■ write()함수

ssize_t write(int fd, void *buf, size_t count)

1번째 인자 : 파일 식별자

2번째 인자 : 쓸 데이터가 있는 버퍼의 포인터형 변수

3번째 인자 : 한번에 읽을 수 있는 데이터의 최대 바이트 수

write함수의 실행이 성공 시,

return값으로는 쓰여진 데이터의 바이트 수를 long int형으로 리턴함.

1. 시스템 프로그래밍 - 4

```
#include<sys/types.h>
#include<unistd.h>
#include<fcntl.h>
#include<stdio.h>

int main(void) {

    int filedес;
    off_t newpos;

    filedес = open("data1.txt", O_RDONLY);

    newpos = lseek( filedес, (off_t)0, SEEK_END);

    printf("file size = %d\n", newpos);

    return 0;
}
```

▪ 코드 이해

Data1.txt 파일을 읽기 전용으로 열고 리턴값을 filedес에 저장한다.

Lseek함수에서 filedес 식별자를 통해서 파일의 처음부터 끝까지 회전해서 파일의 사이즈를 newpos에 저장한다.

▪ lseek()함수

off_T read(int fd, off_t offset, int whence)

1번째 인자 : 파일 식별자

2번째 인자 : 기준 위치로부터 상대적인 변위를 나타낸다.

3번째 인자 : 기준위치

Ex) SEEK_SET : 파일의 맨처음
SEEK_CUR : 현재의 위치
SEEK_END : 파일의 맨 끝

1. 시스템 프로그래밍 - 5

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

int main(int argc, char **argv){

    int fdout, ret;

    char buf[1024];
    if(argc != 2){
        printf("Usage : mvcat filename\n");
        exit(-1);
    }

    fdout = open(argv[1], O_WRONLY | O_CREAT);

    ret = read(0, buf, sizeof(buf));
    write(fdout ,buf,ret);

    close(fdout);
    return 0;
}
```

■ 코드 이해

Argv[1]에 입력된 이름으로 쓰기가 가능한 파일을 만들고

Read에서 식별자를 0번으로 해서 키보드에서 입력 받은 글자를

Buf에 저장하고 write로 buf에 저장된 값을 ret 크기 만큼 fdout에 저장하는

코드이다.

즉, scanf에서 받지 못했던 띄어쓰기 입력을 받을 수 있게 된다.

1. 시스템 프로그래밍 - 6

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

int main(int argc, char **argv){

    int fd = open(argv[1], O_RDONLY);
    int line = 0;
    int word = 0;
    int flag = 0;
    int cnt = 0;
    char ch;
    if(argc != 2){

        printf("You need 1 more parameter\n");
        printf("Usage: mvwc filename\n");
        exit(-1);
    }

    if( (fd = open(argv[1], O_RDONLY)) < 0 ){

        perror("open() ");
        exit(-1);
    }
    while(read(fd, &ch,1)){
        cnt++;
        if(ch == '\n')
            line++;
        if(ch != '\n' && ch != '\t' && ch != ' '){
            if(flag == 0){
                word++;
                flag = 1;
            }//if
        }//if
        else{
            flag = 0;
        }
    }//while

    close(fd);
    printf("%d %d %d %s\n", line, word, cnt, argv[1]);

    return 0;
}
```

■ 코드 이해

read함수로 fd에서 char형 1바이트씩 읽어올 때 마다 cnt(총 글자 수)를 증가시키고

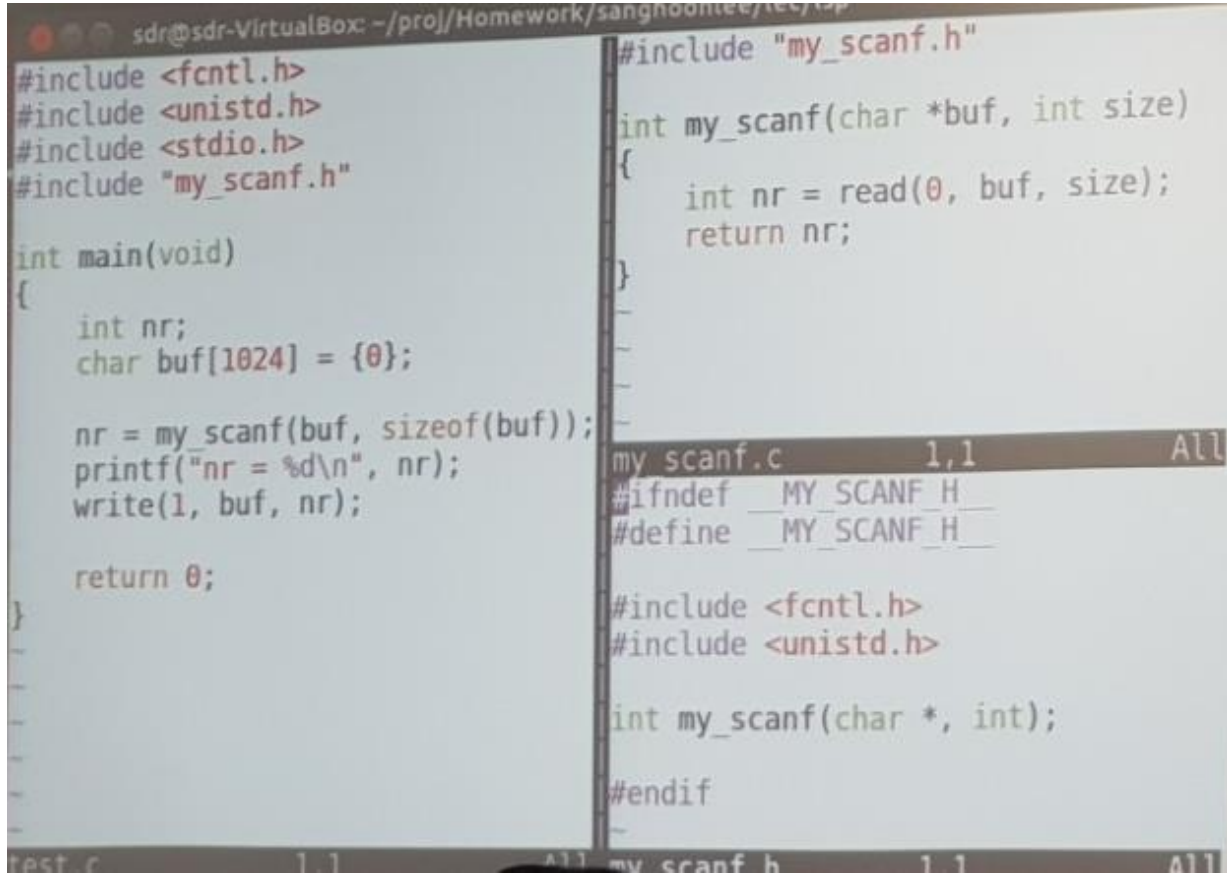
ch가 문단 바꿈이면 line수를 증가하고

ch가 \t도 문단 바꿈도 띄어쓰기도 아니면 if문으로 들어가고

if문에서 다시 flag가 0이면 word(단어 수)가 증가하고 flag를 1로 만든다.

띄어쓰기가 나오면 else로 가서 flag를 0으로 초기화.

1. 시스템 프로그래밍 - 7



```
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>
#include "my_scanf.h"

int main(void)
{
    int nr;
    char buf[1024] = {0};

    nr = my_scanf(buf, sizeof(buf));
    printf("nr = %d\n", nr);
    write(1, buf, nr);

    return 0;
}

#include "my_scanf.h"

int my_scanf(char *buf, int size)
{
    int nr = read(0, buf, size);
    return nr;
}

my_scanf.c 1,1 All
#include <fcntl.h>
#include <unistd.h>

int my_scanf(char *, int);

#endif
```

■ 사용자 정의 헤더파일 만들기

1) my_scanf.h를 만들고

#ifndef #endif를 쌍으로 헤더파일 중복 사용을 방지한다.

#defin으로 사용할 함수를 정의하고
프로토타입을 작성하면 끝.

2) my_scanf.c를 만들고

my_scanf.h에서 만들었던 프로토타입을 제대로 된 함수로 작성한다.

3) 사용할 c파일을 열어서 #include "my_scanf.h"로 선언을 하면
사용자 정의함수를 사용할 수 있다.