

TI DSP, MCU 및 Xilinx Zynq FPGA 프로그램 전문가 과정

강사 - Innova Lee(이상훈)

gcccompil3r@gmail.com

학생 - 은태영

zero_bird@naver.com

파일

```
tewill@tewill-B85M-D3H: ~/Downloads
#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);
    }
    printf("fd = %d\n", filedес);
    close(filedes);

    return 0;
}
```

12,5 Bot

- ❖ open(파일 명, 조건, 권한)
- ❖ 리턴 값으로 파일의 인덱스 값을 리턴.
- ❖ O_CREAT : 파일 생성
- ❖ O_RDWR : 읽고 쓰기 형식
- ❖ O_EXCL : 파일 여부 파악
- ❖ close(파일 명) : 문법. open 후 닫아줘야 함
- ❖ temp.txt 파일을 읽고 쓰기 형식으로 만든다.
- ❖ 이때 중복파일이 있을 경우 error 조건이 참.
- ❖ 에러를 출력하고 종료.
- ❖ 아닐 경우, 인덱스 값 출력 후, 파일 닫기.

파일

```
tewill@tewill-B85M-D3H: ~/Downloads
#include <fcntl.h>

int main(void)
{
    int filesdes1, filesdes2;
    filesdes1 = open("data1.txt", O_WRONLY | O_CREAT | O_TRUNC, 0644);
    filesdes2 = creat("data2.txt", 0644);

    close(filesdes1);
    close(filesdes2);
    return 0;
}
```

1,5 All

- ❖ O_WRONLY : 쓰기 형식
- ❖ O_TRUNC : 파일 갱신
- ❖ creat(파일 명, 권한) : 파일 생성
- ❖ data1.txt 파일과 data2.txt 파일을 생성한다.
- ❖ 두 파일을 닫는다.

파일

```
tewill@tewill-B85M-D3H: ~/Downloads
#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);

    return 0;
}
```

1,5 All

- ❖ O_RDONLY : 읽기 형식
 - ❖ read(읽을 파일, 옮길 파일, 읽을 크기)
 - ❖ 리턴 값은 읽은 바이트 값
 - ❖ write(쓰기 파일, 옮길 파일, 쓰기 크기)
 - ❖ 리턴 값은 쓰기 한 바이트 값
-
- ❖ 읽기 형식과 쓰기 형식의 파일 두개를 연다.
 - ❖ read 를 통해 읽고, write 를 통해 쓴다.
 - ❖ 반복문을 통해 더 이상 읽을 데이터가 없을 때 까지 반복한다.
 - ❖ 오류 체크를 위해 write 리턴 값 비교를 한다.

파일

```
tewill@tewill-B85M-D3H: ~/Downloads
#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 : %ld\n", newpos);

    return 0;
}
```

1,5 All

- ❖ lseek(파일 명, 시작 위치, 목표 위치)
- ❖ 리턴 값은 지정한 길이의 값
- ❖ SEEK_END : 끝까지 진행
- ❖ data1.txt 파일을 열고, 처음부터 끝의 길이 리턴.
- ❖ 출력을 통해 길이 값을 표시한다.
- ❖ close(filedес); 가 빠졌다.

파일

```
tewill@tewill-B85M-D3H: ~/Downloads
#include<stdio.h>

int main(void)
{
    FILE *fp = fopen("mycat.c", "r");
    char buf[1024] = "\0";
    int ret;
    while(ret = fread(buf, 1, sizeof(buf), fp))
    {
        usleep(1000000);
        fwrite(buf, 1, ret, stdout);
    }
    fclose(fp);
    return 0;
}
~
~
1,9 All
```

- ❖ 해당 방식은 속도가 느리다.
- ❖ fopen(오픈 할 이름, 형식)
- ❖ r : 읽기 형식
- ❖ fread(웁길 파일, 웁길 크기, 범위, 읽을 파일)
- ❖ usleep(1000000) : ms = 10^{-6} = 1초
- ❖ fwrite(웁길 파일, 웁길 크기, 범위, 쓰기 파일)
- ❖ fclose() : 문법. fopen 한 파일을 닫아줌

파일

```
tewill@tewill-B85M-D3H: ~/Downloads
#include <stdio.h>
#include <fcntl.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char **argv)
{
    int nread, fdout;
    char buf[1024];

    if(argc != 2)
    {
        printf("1개의 인자만 받는다.\n");
        exit(-1);
    }

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

    nread = read(0, buf, sizeof(buf));

    // write(1, buf, nread);

    if(write(fdout, buf, nread) < nread)
    {
        printf("ERROR!\n");
    }
    close(fdout);

    return 0;
}
```

1,1 All

- ❖ int main(int argc, char **srgv)
- ❖ 사용 시, 인자의 정보를 받아온다.
- ❖ argc 는 인자의 숫자 값을 받는다.
- ❖ srgv 는 문자열을 받아온다.
- ❖ 인덱스의 0번은 표준 입력[키보드] 이다.
- ❖ 인덱스의 1번은 표준 출력[모니터] 이다.

- ❖ 키보드를 통해 입력 받는다.
- ❖ // 모니터에 출력한다.
- ❖ 쓰기파일에 옮긴다.

파일

```
tewill@tewill-B85M-D3H: ~/Downloads
#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: mywc 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;
            }
        }
        else
        {
            flag = 0;
        }
    }
    close(fd);
    printf("%d %d %d %s\n", line, word, cnt, argv[1]);
    return 0;
}
```

24,2-9 All

- ❖ 추가 인자로 1개를 더 받는다.
- ❖ fd 에 인자로 받은 파일을 읽기 형식으로 연다.
- ❖ 인자 수와 파일에 대한 에러 체크를 한다.
- ❖ 반복문을 통해 fd 를 1바이트로 ch 에 저장한다.
- ❖ 데이터 숫자를 cnt 에 저장한다.
- ❖ 엔터를 line 에 저장한다.
- ❖ 각 문단의 숫자를 word에 저장한다.
- ❖ 더 이상 읽을 데이터가 없을 때 반복문을 나온다.
- ❖ 파일을 닫고, 얻은 데이터의 정보를 출력한다.

파일

```
tewill@tewill-B85M-D3H: ~/Downloads
#include <stdio.h>
#include <fcntl.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char **argv)
{
    int fd, ret;
    char buf[1024];
    if(argc != 2)
    {
        printf("Usage: mycat filename\n");
        exit(-1);
    }
    fd = open(argv[1], O_RDONLY);
    while(ret = read(fd, buf, sizeof(buf)))
    {
        write(1, buf, ret);
    }
    close(fd);

    return 0;
}
```

1,1 All

- ❖ 추가 인자로 1개를 받아온다.
- ❖ 읽기 형식으로 받아온 파일을 연다.
- ❖ 반복문을 통해 파일을 읽어 온다.
- ❖ 쓰기를 통해 모니터에 받아온 정보를 출력한다.
- ❖ 더 이상 데이터가 없을 때 반복문을 나온다.
- ❖ 파일을 닫는다.

파일

```
tewill@tewill-B85M-D3H: ~/Downloads
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdlib.h>

int main(int argc, char **argv)
{
    int fdin, fdout;
    char buf[1024] = {0};
    ssize_t nread;
    int i;

    if(argc != 3)
    {
        printf("인자 입력 3개 하라고 ! \n");
        exit(-1);
    }

    fdin = open(argv[1], O_RDONLY);
    fdout = open(argv[2], O_WRONLY | O_CREAT, 0644);

    while((nread = read(fdin, buf, 1024)) > 0)
    {
        if((write(fdout, buf, nread)) < nread) {
            close(fdin);
            close(fdout);
            printf("비정상 종료\n");
        }
    }
    close(fdin);
    close(fdout);

    for(i = 0; i < argc; i++)
    {
        printf("당신이 입력한 인자는 = %s\n", argv[i]);
    }

    return 0;
}
```

1,1 All

- ❖ 추가 인자를 2개 받아온다.
- ❖ 각각 읽기 파일과 쓰기 파일로 연다.
- ❖ 반복문을 통해 읽기 파일을 읽어온다.
- ❖ 반복문 안에서 쓰기 파일에 읽어온 정보를 넣는다.
- ❖ 이때, 리턴값을 비교하여 오류 체크를 한다.
- ❖ 데이터를 다 읽고 반복문을 나와 파일을 닫는다.
- ❖ 받아온 인자의 문자열을 출력한다.

파일

```
tewill@tewill-B85M-D3H: ~/Downloads
#include "my_scanf.h"

int my_scanf(char *buf, int size)
{
    int nr = read(0, buf, size);
    return nr;
}
~
1,1 All
```

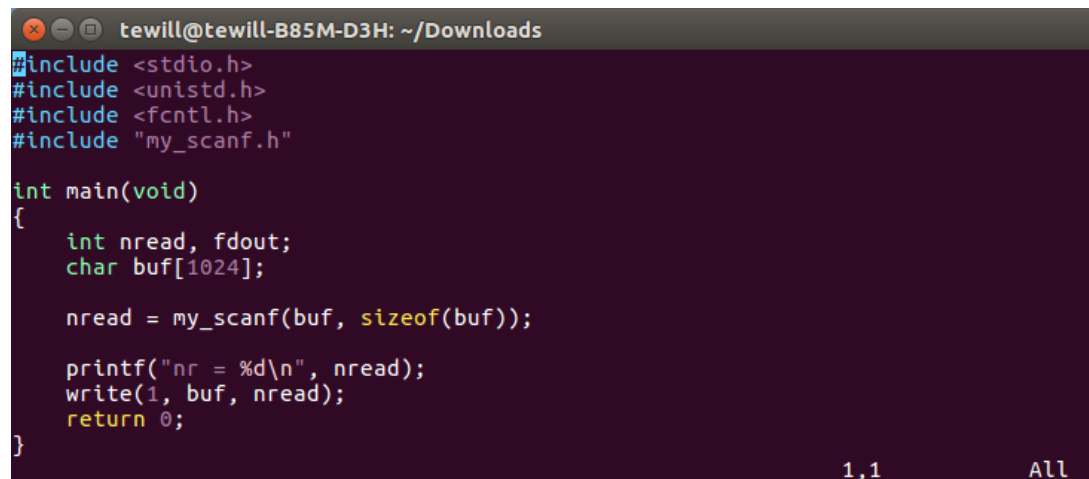
```
tewill@tewill-B85M-D3H: ~/Downloads
#ifndef _MY_SCANF_H_
#define _MY_SCANF_H_

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

int my_scanf(char *, int);
#endif
~
1,1 All
```

- ❖ ifndef / define / endif : 일종의 헤더파일 형식
 - ❖ ifndef 선언 여부를 확인한다
 - ❖ define 선언되어 있지 않을 경우 선언한다.
 - ❖ endif 종료 위치를 표시한다.
-
- ❖ 헤더 파일을 만들고 사용될 함수 형식을 입력한다.
 - ❖ 실제 함수를 만들고, 헤더파일을 include 한다.

파일



```
tewill@tewill-B85M-D3H: ~/Downloads
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include "my_scanf.h"

int main(void)
{
    int nread, fdout;
    char buf[1024];

    nread = my_scanf(buf, sizeof(buf));

    printf("nr = %d\n", nread);
    write(1, buf, nread);
    return 0;
}
```

1,1 All

- ❖ “ ” 사용자 지정 헤더파일
- ❖ 만들어 둔 my_scanf.h 를 include 한다.
- ❖ 사용자 지정 헤더파일에서 함수를 인식한다.
- ❖ 함수가 사용될 때, 해당 함수를 찾아간다.

