Xilinx Zynq FPGA, TI DSP, MCU기반의 프로그래밍 및 회로 설계 전문가 과정

강사 - Innov (이상훈)
gcccompil3r@gmail.com
학생 - 이유성
dbtjd1102@naver.com

file1.c

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



#define ERROR -1

int main(void)
{
	int filedes;
	char pathname[] ="temp.txt";

	if((filedes = open(pathname, O_CREAT | O_RDWR|O_EXCL , 0644)) ==ERROR)
//중복파일이 있을 경우error참


{
		printf("File Open ERROR!\n");
		exit(-1);
	}
	printf("fd = %d\n",filedes);

	close(filedes);

	return 0;
}
```

```
  // pathname은 개방될 화일의 경로이름(주소)을 갖고 있는 문자열에 대한 포인터이다.
    //화일은 open이 호출되기전에 반드시 존재해야 한다. 그렇지 않으면 오류가 발생.
	//  열고 만들고 읽고 쓰고 중복이 안되는 용으로,,
	//O_EXCL <<원래있던 파일에 덮어쓰면 애러 중복 x
	//rdwr읽고쓰기,                배열의 이름 :주소
	//open은 숫자를 리턴,, ,  0664 - 변환때 다시얘기,
                        //open을 통해 temp.txt파일을 만들었다,,
```

fcntl.h 에서 정의되는 상수
- O_RDONLY 읽기만 허용하는 개방 화일
- O_WRONLY 쓰기만 허용하는 개방 화일
- O_RDWR 읽고 쓰는 것을 허용하는 개방 화일

nread = read(filedes, bufptr, n);

read 의 첫번째 인수 filedes 는 앞서 호출한 open 이나 creat 에 의해서 만들어진 화일 기술어이다. 두번째 인수 bufptr 은 자료가 복사되어질 문자 배열에 대한 포인터이다. 새번째 인수는 화일로부터 읽어질 바이트의 수를 가지는 양의 정수값이다. read 로부터의 복귀값 nread 는 실제로 읽힌 바이트의 수를 나타낸다. read 의 경우, 시스템은 각 호출 후에 읽혀진 바이트의 수만큼 읽기-쓰기 포인터를 전진 시킨다. read 호출에 의해 요청된 문자의 수가 화일에 남아있는 문자의 수보다 클 경우 시스템은 남아있는 문자들만 옮기고, 복귀값을 적적하게 고친다. 이제 이후에 호출되는 read 의 복귀값은 0 이 된다. 즉, 남아있는 문자가 하나도 없게 된다. 사실, read 로부터 복귀값이 0 인가를 조사하는 것이 프로그램 안에서 화일의 끝을 조사하는 보통의 방법이다

file2.c

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

int main(void)
{
	int filedes1,filedes2;

	filedes1 = open("data1.txt",O_WRONLY | O_CREAT | O_TRUNC, 0644);
	//열어라 data1.txt를 쓰기 전용 만들어 리프레쉬하고 그게 filedes1이다

	//write only , trunc ->어떤 작업을 하게 되면은 파일을 리프레쉬(다 지워..기존내용 x,,기록할 필요가 없는 정보들,,),.
	//.(임시적으로 데이터 저장할때 사용)
	filedes2 = creat("data2.txt",0644);

	printf("%d\n",filedes1);
	printf("%d\n",filedes2);

	close(filedes1);
```

```
	close(filedes2);



	return 0;
}
```

// creat면 만드는데 원래 data2.txt파일에 새로 덮어씌움 . 조건이 안붙으면 왜인지 creat가 안됨.

file3.c

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

int main(void)
{
	int fdin,fdout;	//fd filedes (파일 디스크립터) 의미하는 바 : 파일 식별자 파일을 오픈하고 파일에 대한 숫자값
	ssize_t nread;	//읽은 사이즈만큼  long int..
	char buf[1024];	//읽은 결과를 저장하는 버퍼

	fdin = open("temp1.txt",O_RDONLY);



	fdout = open("temp2.txt", O_WRONLY | O_CREAT | O_TRUNC, 0644);
                                                        //리프레쉬


	while((nread = read(fdin,buf,1024)) >0) //buf 바가지 1024 크기 fdin을 읽어라 1024씩 그리고 그 글 수를 nread에 대입
                                                //read(fd, buf , 읽을크기)  fdin ->buf  만약 5000글이면 5번퍼고,1024이하 한번
	{
		if(write(fdout,buf,nread) < nread)//buf에 있는 nread크기를  fdout에 적어
		{                                       //write(fd buf 쓸 크기) , buf->fdout
			close(fdin);
			close(fdout);
```

```
                }
        }
        printf("%lu\n" , nread);  //글을 모두 옮겼으니 0
        printf("%d\n",fdin);          //첫번째 실행 파일 3
        printf("%d\n",fdout);         //두번째 실행 파일 4  파일 고유 숫자
                                      //(파일을 관리 하기 위해서 모든 파일에 파일 디스크립터를 할당
해준다)
        return 0;
}

        //temp1 내용이 temp2에 저장,, temp1.txt이라는 것을 내가 만들고,, 입력
```

file4.c

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

int main(void)
{

        int filedes;
        off_t newpos;    //0부터 시작??

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

        newpos = lseek(filedes,(off_t)0,SEEK_END);
        //0부터 시작 해서 끝까지  ( 처음부터 끝까지 순회) = > 파일의 사이즈..  newposition 방금 내가
만든 파일의 사이즈
        printf("file size : %lu\n " , newpos);

        return 0 ;

}

// \0  파일의 끝이라는 요고 때문에 문자 +1의 값이 나옴

// filesed 배열 같은 느낌으로 쓰면 됨.

data1.txt 에 hello hello 라는 텍스트 쳐놓음.

./a.out data1.txt
file size 12
```

file5.c

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


int main(int argc, char **argv)          //argc는 인자의 숫자 값을 받는다. srgv는 문자열을 받아온다)
{
        int i;
        char buf[1024] = {0};
        int in, out, nread;

        if(argc !=3)
        {
                printf("인자 입력 3개 하라고!\n");
                exit(-1);
        }                       //./a.out  => argv[0]

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

        in =open(argv[1],O_RDONLY);//복사 원본
        out = open(argv[2], O_CREAT | O_WRONLY | O_TRUNC); //복사할 곳


        printf("%d\n",in);
        printf("%d\n",out);

        while((nread = read(in,buf,1024)) >0)
        {
                if(write(out,buf,nread) >nread)
                {
                        close(in);
                        close(out);
                        printf("비정상 종료\n");
                        exit(-1);
                }
                close(in);
                close(out);
                printf("정상 종료\n");
        }
```

```
	return 0;
}


//./a.out ,2 , 3 인자 3개.,

//./a.out 자기가 만든걸 실행 ./a.out test.txt a.txt   cp test.txt a.txt
// ./a.out 가 cp 역할


//%lu unsigned long int , %ld signed long int


yoosunglee@yoosunglee-Z20NH-AS51B5U:~/Homework/yoosunglee/3st$ ./a.out text.txt a.txt
당신이 입력한 인자는 = ./a.out
당신이 입력한 인자는 = text.txt
당신이 입력한 인자는 = a.txt
3
4
정상 종료

text.txt에만 hello 적음 - > a.txt 에도 hello 복사됨.




file6.c

#include<stdio.h>



int main(void)
{

	FILE *fp = fopen("mycat.c","r");  //r => read only 와 같은 역할..[w..a] fopen("파일명","r",w,a)
	char buf[1024] = "\0";
	int ret;
	while(ret = fread(buf,1,sizeof(buf),fp)) //읽어서 버퍼 read에 저장  fwrite(쓸 문자열 주소,문자열
사이즈,반복횟수,쓸 파일 포인터)
	{
		usleep(1000000); //us=10^-6 1초 후 cat역할을 함


		fwrite(buf,1,ret,stdout);		fread(저장할 변수,사이즈,반복횟수,읽을 파일 포인터)
	}
	fclose(fp);    //해제(닫을 파일 포인터)
	return 0;
}
mycat.c 에 hello 적으면
```

```
yoosunglee@yoosunglee-Z20NH-AS51B5U:~/Homework/yoosunglee/3st$ ./a.out
hello
```

file7.c

```
#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)  //argc 가 2개가 아니면x
	{
		printf("Usage : mycat filename\n");
		exit(-1);
	}
	fd = open(argv[1], O_RDONLY); //argv문자열 배열 *argv[]   fd = 3

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

	while(ret = read(fd,buf,sizeof(buf)))
	{
		write(1,buf,ret); //출력   //만약 3을 넣으면 자기 자신에 내용을 넣는다는 얘기

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

//cat역할

```
3
hello hi
```

file8.c
헤더파일 추가하고 컴파일 할때 //
gcc -o 실행 파일 이름 file1.c file2.c ..

:vs 수직 분리
:sp 수평 분리
:e 파일명 ->파일명으로 들어가야 내용 달리 쓸 수 있다.
Ctrl + w , w 다음 창으로 넘어가기
/찾을단어
:set hlsearch ->블록처리

.c파일
```
#include "my_scanf.h"

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

h파일
```
#ifndef __Y_SCANF_H_  // ~이름이 정의되지 않았으면 밑에 실행
#define __Y_SCANF_H_ // ~이름이 정의되었으면 종료

#include<fctl.h>
#include<unistd.h>

int my_scanf(char *,int);

#endif   // 종료 위치를 표시
```

.c파일
```
#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;
}
```

file10.c

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

//wc

int main(int argc, char **argv)			//argc,argc프로그램으로 인자 전달
{
	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() ");//오류 상세히
		exit(-1);//프로세스의 상태,,
	}

	while(read(fd,&ch,1))// 배열은 주소 ..
	{
		cnt++;//문자 몇개
		if(ch =='\n')
			line++;

		if(ch != '\n' && ch != '\t' &&ch !=' ')//숫자거나 문자,,
		{
			if(flag ==0)
			{
				word ++;
				flag =1;
			}
		}
		else
		{
			flag =0;
		}
		close(fd);
		printf("%d %d %d %s\n",line,word,cnt,argv[1]);
	}
	return 0;
}
```