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

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 장성환
redmk1025@gmail.com

오픈소스와 프리소스의 차이

프리소스는 영리적 활동에 상관이 없지만
오픈 소스는 문제가 생긴다.

모듈리식 커널 과 마이크로 커널

마이크로 커널 - 조립형식 구조 ? 붙이고 싶음 붙이고 떼고 싶음 떼는 구조.
모듈리식 커널 - 필수적 요소를 한번에 때려 박은 구조.

디바이스 드라이버 - 모듈리식 커널에 마이크로 커널의 개념을 탑재

시스템 콜 인터럽트 -유일한 소프트 웨어 콜
유저가 커널에 요청하면 커널이 처리하여 피드백.
여기서 유저가 커널에 요청하는 작업이 시스템 콜이다.

시스템 콜 = 소프트 웨어 인터럽트

open , read, write() 등이 시스템 콜이다.

커널이 인덱스만 주는 이유는 은닉 때문이다.

유저 → 운영체제 → 디바이스 드라이버 → 디스크 자기장 입력 등등 으로
동작이 된다.

Close (index) 파일 포인터에서 인덱스에 해당하는 값을 해제 (쓰지 않을 거니까)

```
 1 #include <stdio.h>
 2 #include <sys/types.h>
 3 #include <unistd.h>
 4 #include <fcntl.h>
 5
 6 int main(void){
 7     int filedes;
 8     off_t newpos;
 9
10     filedes = open("data1.txt", O_RDONLY);
11     newpos = lseek(filedes, (off_t)0, SEEK_END);
12
13     printf("file size : %d\n", newpos);
14
15     return 0;
16 }
~
```

off_t 0
시작을 0 부터 하겠다는 뜻

SEEK_END
파일의 마지막 까지 읽겠다.

Newpos 는 따라서 만든 파일의 사이즈를 나타나게 된다.

반드시 알아야 할 Linux 기본 명령어

Mv
cp
rm
ls
cd
mkdir
gcc
mkfifo
touch

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


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

    int i;
    int fsource, fdest;
    ssize_t nread;
    char buf[1024];

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

    fsource = open(argv[1],O_RDONLY);
    fdest = open(argv[2],O_WRONLY | O_CREAT | O_TRUNC, 0644);

    while((nread = read(fsource, buf,1024)) >0){
        if(write(fdest, buf, nread)<nread){

            close(fsource);
            close(fdest);
        }
    }

    close(fsource);
    close(fdest);
    return 0;
}
```

해당 파일은

main 함수에 들어오는 인자값(입력받은 문자열 갯수 및 문자열) 을 이용한다.

파일 실행명 다음에 들어오는 문자열을 버퍼에 저장 시킨다.

실행파일 – 문자열 1 - 문자열 2 이런 식으로 argv 에 저장이 된다.

리눅스에서 cp 소스파일 목적파일 기능과 똑같이 구현하였다.

```
#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;
}
```

Fopen - 시스템 콜은 아니다. “r” 은 O_RDONLY 랑 같은 뜻이다.

속도는 시스템 콜이 압도적으로 빠르다.

Fread - 1 바이트 씩 1024 번 읽어라.

Usleep 마이크로 세컨드 (1 초 후)

stdout 표준 출력 = 모니터

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

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

    int fd, ret;
    char buf[1024];
    if(argc != 2){
        printf("Usage : mycat2 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;
}
```

시스템 콜 버전

해당 파일은

main 함수에 들어오는 인자값(입력받은 문자열 갯수 및 문자열) 을 이용한다.

파일 실행명 다음에 들어오는 문자열을 버퍼에 저장 시킨뒤에
모니터에 출력하는 기능이다.

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

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

    int fd,ret;
    int fo;

    char buf[1024] = "\0";

    fd = open("input.txt",O_WRONLY | O_CREAT | O_TRUNC,0644);


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


    close(fd);

    return 0;
}
```

Ret 에 read()함수에 엔터가 들어가면 입력 된 버퍼 수를 리턴;

read() 함수 고찰

read(index, buffer, size)
return 값은 읽은 buffer 의 수 +1
size 이상의 크기일 경우 size 만큼 읽는다.
Index 는 표준입출력 및 어떠한 파일을 가리킬 수 있다.
(표준 입력 : 0, 표준 출력 :1, 파일들 : 3, 4, 5 .....)

:sp :vs :e 만들고 싶은 파일명 ctrl + ww wget	위아래로 나누기 왼오른쪽으로 나누기 새롭게 파일 생성 윈도우 이동 인터넷에서 소스를 다운받겠다.

*파일 분할 하여 컴파일 할때, 해당 c 파일 모두 컴파일 해야 동작된다.
 	
<my_cat3.c> main
 	
#include <stdio.h>
#include <fcntl.h>
#include <stdlib.h>
#include <unistd.h>
#include "my_scanf.h"
 	
int main(int argc, char **argv){
 	
  int dest,ret;
 	
  char buf[1024]={0};
 	
  dest = open("input.txt",O_WRONLY | O_CREAT | O_TRUNC,0644);
 	
  ret = my_scanf(buf,sizeof(buf));
  write(dest,buf,ret);
 	
  close(dest);
 	
  return 0;
}
	ex) gcc my_scanf.c mycat.c 이런 식으로 컴파일 한다.
	
	메인함수
	
	해당 헤더 파일을 include 시킨다.
	
	만든 헤더 파일은 “ 헤더파일.h” 으로 해놓는다. (커스텀 헤더 파일)
	
	시스템에 있는 헤더 파일은 <헤더파일.h> 으로 한다.
	
	메인함수에는 없지만 해당 헤더파일을 넣어 두었기 때문에
	
	해당 헤더파일이 가리키는 함수를 바로 쓸 수 있다.

<my_scanf.h>

```
#ifndef __MY_SCANF_H__
#define __MY_SCANF_H__

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


int my_scanf(char *buf,int size);

#endif
```

헤더 파일

쓰이는 함수를 미리 선언시켜 놓는다.

(프로토 타입을 기술하여 놓는다.)

<my_scanf.c>

```
#include "my_scanf.h"


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

}
```

분할 c 코드 파일

실제 함수의 동작이 코딩되어 있다.

큰 파일 시스템을 만드는 경우에 이렇게 분할해 놓아야

알아보기도 편하고 기능의 분리가 되어 고치기도 편하다.

wget https://mirrors.edge.kernel.org/pub/linux/kernel/v4.x/linux-4.4.tar.gz tar zxvf linux-4.4.tar.gz set hlsearch include/linux/sched.h	리눅스 커널 다운받기 커널 압축풀기 하이라이트 검색 엔터 치고 /검색어 쓰면 된다. 해당 커널의 파일 오픈

* wc_implementation
 	
#include <stdio.h>
#include <fcntl.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <unistd.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){
	해당 파일을 실행파일과 같이 입력 받아서
	
	그 파일을 열고 파일의 단어의 갯수와 줄 수를 나타내 주는 알고리즘이다.
	
	탭이나 공백 개행을 기준으로 flag 가 0 인 상태일 때, 단어의 갯수가 오르게 된다.
	
	즉, 문자나 숫자일 경우 flag 가 0 인지 계속 확인하는 것이고
	
	그게 아니라면 flag 를 0 으로 맞춰준다.
	
	탭 공백 개행 이후에 단어가 있다는 것을 알고리즘으로 구현한 것이다.

```
        perror("open() ");
        exit(-1);
    }// 기법 오류를 세팅!

    while(read(fd,&ch,1)){
        cnt ++;
        if(ch == '\n')
            line++;

        if(ch != '\n' && ch != '\t' && ch != ' '){ //개행 탭 공백

            if(flag ==0){
                word++;
                flag =1;
            }

        }

        else{
            flag =0;
        }
    }

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

    return 0;
}
```