

TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

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

학생 - 하성용
accept0108@naver.com

18 일차

-리눅스 프로그래밍 1 일차

SW 3 대장

1. OS

리소스

-HW

-SW

2. Compiler

3. DB

Linux 는 우리가 무언가를 하고자 할 때 Resource 를 제공해준다

Windows 에서 마우스 움직이는 것, 키보드 치는것, 모니터가 보이는것
이러한 기능들을 전부 Linux OS 가 담당한다.

스톨만- Linux 의 개발자 이며 사회주의자

Linux 에 GNU 사상이 많이 들어감

GNU 란?

GNU is Not Unix 란말 = 나는 돈을 받지않겠다

모든 소프트웨어는 무료로 제공해야한다

Free 랑 Open 의 차이

Free SW 란 → Free 뜻은 자유, 책임 X

Open SW → 오픈소스를 사용할때는 책임이 있으니 주의

OS – Linus(제자 리누즈, 모놀리식 커널) → Mich (스승, 마이크로커널)

Linux 는 어떤 소프트웨어를 깔아도 재부팅 필요 X

ex) ?? 설치하면 재부팅하십시오 메시지 (윈도우도 후에 리눅스 따라함)

디바이스 드라이버(제자가 스승의 영향을 받음)

디바이스 드라이버란? 커널공간에서 응용프로그램과 하드웨어간의 데이터를 주고 받게 해주는 프로그램

프로그램을 추려서 넣는게 모놀리식 커널(운영체제 필요한것 몇가지)

마이크로 커널(ex. 레고블럭)은 붙이고싶으면 붙였다 떼고싶으면 뗄수있는 커널

System call 은 User 가 Kernul 에게 요청하는 작업을 의미한다

System call 은 소프트웨어에서 발생시키는 인터럽트이다

유저가 커널에게 요청 - > 커널은 유저에게 처리(여기서 요청하는게 System call 에 해당)

file_io2.c

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

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

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

```
#include<unistd.h> // warning 안나오게
```

```
#define ERROR
```

```

int main(void)
{
    int filedes;
    char pathname[] = "temp.txt"; //temp.txt 생성
    if((filedes = open(pathname, O_CREAT | O_RDWR, /%$%$O_EXCL, -//0644)) ==
ERROR)
/*
filedes 란 것에 무엇이 들어간다는것, Open 은 숫자가 들어간다는건데 어떤 리턴이 들어가는지
O_CREAT//쓰다만거같은느낌=음주코딩 E 빠뜨림, O_RDWR//read 랑 write
*/
    {
        printf("File Open Error!\n");
        exit(1);
    }
    close(filedes); //filedes 가 숫자를 나타낸다는것

    return 0;
}

```

open()의 Flags 부분에 O_EXCL 을 추가\

o_excl 은 어떤동일한 이름의 파일이 있을때 그것을 덮어쓸수없게만드는소스

파일실행하고 temp.c 가 있는지 확인할것

리누즈 토발즈는 스톨만과 과격함이 닮음

리눅스에서 그래픽에 대해서 불만들을 많이들음

엔비디아한테 대학강의중에 중지를 날리고 엔비디아는 다음날 업데이트를함(영향력상당)

printf("fd=%d\n",filedes) // 현재 사용하고있는 파일갯수

유닉스가 내세우는 핵심철학이란?

모든것은 파일이다

모니터도 파일이고 마우스도 파일이고~

OR 는 합집합 AND 는 교집합

AND 는 공통분모없이는 사용하면안됨

0644 앞에 | 추가하면 Error 뜸

temp.txt 를 삭제하고 실행하면 실행됨

file_io3.c

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

```
int main(void)
```

```
{
    int filedes1, filedes2;

    filedes1 = open("data1.txt",O_WRONLY|O_CREAT|O_TRUNC,0644);
    /* O_WRONLY 는 오직 쓰기만, O_TRUNC 는 수정하면 무조건 파일을 한번 밀어버린다는거
    ->무언가 임시적으로 저장할때 사용 ex)쿠키*/
    filedes2 = creat("data2.txt",0644);

    close(filedes1);
    close(filedes2);

    return 0;
}
```

//data1, data2 가 만들어짐

file_io4.c //컴파일만하고 실행하지말고 쓰고 기다릴것

fd 의 약자 = 파일 디스크립터 (파일 기술자)

숫자값이 나타내는것이 무엇을 의미?

일단은 파일 식별자

temp1 텍스트는 읽기전용

temp2 는 ~

와일문이 열리고

read(fd, buf, 읽을크기)

buf // 먼가 읽을거를 저장하는

temp2 텍스트에 무언가를 읽어서 buf 에 저장하겠다~

nread 는 내가 몇바이트를 읽었나 를 받게됨

1024 를 적었지만 읽을거리가 더적다면 와일루프 빠져나감

1024 를 넘어도 다받아짐

1024 지우면 다못받아짐

write(fd,buf,쓸 크기)

buf 에 있는값이 fd 에 써진다는거

거기서 써질크기는 읽은 바이트로

nread 보다 더적다는건 무언가 문제가 있다는것으로 받아들이고 close

더 할작업이 없다면 0 으로 종료함

이소스가 에러가 안뜨려면 temp1 에 무언가 내용이 입력되어야함
temp1 을 직접만들고 내용을 입력하고 소스 실행하면
temp2 에 temp1 의 내용이 카피됨
카피명령어를 소스로 나타낸거

```
#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;
}
```

```
-----

#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;
}

```

read(fd, buf, 읽을크기

task_struct 라는 구조체를 배우게될건데
구조체 안을 들여다보면

task_struct
files_struct → //라는 구조체를 따라가보면 // f_pos ← **fd 더블포인터로 관리
재해석하면 이 주소값을 저장할수있는 files *fd[] // 배열의 인덱스를 의미

표준출력

1 번이 모니터, 0 번은 표준입력 키보드, 표준에러(error) 는 에러메시지 뿜앗는게 2 번

배열의 인덱스를 유저한테 넘겨줌

, file_operations
inode
path
super_block

system call (SW Interrupt)

커널 ↓
↑
유저 ←

사용자가 내용을 보지못하도록 정보를 어느정도 은닉하는게 좋음(가독성,보안)

파일포인터의 인덱스를 넘겨주는것

인덱스에 해당하는 내용을 읽고싶어 혹은 쓰고싶어 라는걸 유저가 커널에게 요청할수있음

위 파일은 내가 읽은 fdout 이라는 파일 인덱스의 내용을 buf 에 써주라는 요청

close 는 방금 만든거 해제해줄것을 요청

close 도 마찬가지로 system call

무언가 하드웨어적인 연산이 가미되는거 전부 system call

file_io5.c

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

```
int main(void)
{
    int filedес;
    off_t newpos;

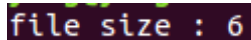
    filedес = open("data1.txt", O_RDONLY);

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

    printf("file size : %d\n",newpos);
}
```

// %ld 가 붙으면 워닝이 안뜰거임

// data1 에 Hello 를 입력하고 실행해보면 6 이나올거임

A terminal window snippet showing the output of the program: "file size : 6". The text is white on a dark background.

0 부터 맨끝까지 순회하면 파일의 사이즈가 나옴

newpos 가 의미하는건 파일의 사이즈를 의미

리눅스 명령어

Linux Command Line Collection

Linux 를 사용함에 있어서 최소한 알아야할 명령어

1. mv
2. cp
3. rm
4. ls
5. cd
6. mkdir
7. gcc
8. mkfifo
9. touch
10. cat
11. tar

cp test.txt a.txt // test.txt 란 파일을 a.txt 란 이름으로 만들기

실습 1

my cp

arc arv 쓸것

포맷 힌트

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

```
int main(int argc, char **arav)
{
    int i;
    if(argc != 3)
    {
        printf("인자 입력 3 개 하라고!\n");
        exit(-1);
    }
```

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

//아까 while 문참고

mycp test.txt a.txt

a.out

답 :

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

```
int main(int argc, char **arav)
{
    int i;
    char buf[1024] = {0};
    int in, out, nread;
    if(argc != 3)
    {
        printf("인자 입력 3 개 하라고!\n");
        exit(-1);
    }

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



```

in = open(argv[1], O_RDONLY);
out = open(argv[2], O_CREAT | O_WRONLY | O_TRUNC);

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

return 0;
}

```

실행방법 :

```

a.out 1
a.out 2
./a.out test.txt a.txt
cat lmpementation
cat 만들어보기

```

```

cat1.c // 완성아님
#include<stdio.h>
#include<unistd.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;
}

```

system call 로 바꾸게하면

```

#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"); //파일명
        exit(-1);
    }
    fd=open(argv[1], O_RDONLY);
    while(ret=read(fd,buf,sizeof(buf)))
    {
        write(1, buf, ret);
    }
    close(fd);
    return 0;
}

```

실행어 : ./a.out cat2.c

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

속도는 시스템콜이 압도적으로 빠름

해당파일에 대한 디스크래터를 얻었으니 버퍼에 저장

1 바이트씩 1024 번 읽어라 어디에서? fp 에서

1024 읽어서 어디에 넣어라? 버퍼에

읽은 버퍼는 어디로? Ret

usleep 은? 마이크로세컨드

마이크로세컨드 단위는 10 의 -6 승 = 1 초

1 초후에 모니터에 써라는 뜻이됨

스텐다드아웃은? 표준출력

모니터는? 출력장치

모니터에 뭐하래?

읽은내용을 출력

아니면? 종료

if 는? argv1 번째리라고있음

1 번은? readonly 로

파일디스크래터화 되고 리드하고 콰이트

argc =

mycp test.txt a.txt

1	2	3
---	---	---

argv = 배열

mycp test.txt a.txt

[0]	[1]	[2]
-----	-----	-----

argv 다음에 오는게 표준 들

scan1.c ←

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

int main(int argc, char **argv)
{
    int fd, ret;
    char buf[1024] = {0};

    nr = read(0,buf,sizeof(buf));
    write("nr=%d\n",nr); //차이를 알고싶을때
    write(1, buf, nr);

    return 0;
}
```

:sp = 나누기

:e 파일명

test.c
my_scanf.c
my_scanf.h

test.c
#include "my_scanf.h"

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

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

    return 0;
}
```

my_scanf.c
#include "my_scanf.h"

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

```

        return 0;
    }
}
my_scanf.h
#ifndef __MY_SCANF_H__ //함수이름이 중복되지않는걸 방지
어딘가에서 my_scanf 가 중복해서 선언되면 중복선언으
#define __MY_SCANF_H__ //디파인 없으면 0 으로 정의해줌

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

int my_scanf(char *, int); //헤더파일 중복해서 받으면 오류남

#endif

```

실행시 : gcc test.c my_scanf.c

헤더파일은 컴파일하는 경로상에 위치해있어야함
 내가만든건 땡땡 :
 시스템에 있다면 꺾쇠 ^

wc Implementation

```

#include<fcntl.h>
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<errno.h>
#include<unistd.h> //warning 만나게

int main(int argc, char **argv)
{
    int fd = open(argv[1], O_RDONLY); //argv[1]을 리드온리로 열고있음 즉 인자가 하나와야함
    파일을 오게해야지 오픈을 할수있음
    int line = 0; //라인숫자 0
    int word = 0; //단어갯수 0
    int flag = 0; //플래그 0
    int cnt = 0; //문자카운트갯수 0
    char ch;
    if(argc != 2)
    {
        printf("You need 1 more parameter\n"); //이상하니까 1 하나만 넣어줘
        printf("Usage : mywc filename\n");
        exit(-1);
    }
    if((fd = open(argv[1], O_RDONLY)) < 0 ) //0 보다 작으면 에러
    {

```

```

        perror("open() ");
        exit(-1); //-1 은 프로세스의 상태값을 표시하는거고 표준 에러와는 관계없음 //표준 에러
    }
    while(read(fd, &ch, 1)) //배열은 주소, 일반적인 ch 형은 주솟값을 전달하기위해서 &이 필요함
    {
        cnt++; //
        if(ch == '\n')
            line++; //라인 추가하고
        if(ch != '\n' && ch != '\t' && ch != ' ') //계행도 아니고 =이아니고아니면 숫자또는 문자밖에
없음
        {
            if(flag == 0)
            {
                word++;
                flag=1; //플래그를 1 로설정 위로가서 리드
            }
        }
        else //읽을게 더없으면 와일루프가 끝나고 0
        {
            flag=0;
        }
    }
    close(fd);
    printf("%d %d %d %s\n",line, word, cnt, argv[1]);
    return 0;
}

```

커널소스(운영체제) 다운받는법

mkdir kernel

cd kernel/

wget <http://mirrors.edge.kernel.org/pub/linux/kernel/v4.x/linux-4.4.tar.gz>

tar zxvf linux-4.4.tar.gz

소스 코드 받는법

토발즈가 만든 소스들

30 개가 아니면 다운 다시받아야함

cd kernel/linux-4.4/include/linux/sched.h

:sp

:set hlsearch

내일 커널소스들을 알아볼거

오답노트 깃허브 x

다음주 월요일까지 → 강사님 g 메일로 보내기[gmail.com]

오늘 학습내용 정리