TI DSP, MCU 및 Xilinx Zynq FPGA
프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 하성용
accept0108@naver.com

21 일차
- 리눅스 프로그래밍 4 일차

< ls module >

touch a.txt b.txt c.txt d.txt e.txt
mkdir fire getout gogo
ls_module1.c 구동시키면 이와 유사한 결과를 볼수있음

ls_module1.c

```
#include<stdio.h>
#include<sys/types.h>
#include<dirent.h>

int main(void)
{
	DIR *dp; //파일포인터와 비슷하지만 차이 있음
	int i = 0;
	struct dirent *p; //디렉토리내에있는 리스트라고보면됨
	dp = opendir("."); //디렉토리를 오픈
.은 현재디렉토리를 나타냄
	while(p=readdir(dp)) //dp 를 읽으면 리스트가 넘어옴 //데이터있으면 와일문 돔
	{
		if(p->d_name[0] == '.') //배열의 0 번은 맨앞의글자가 .이면
			continue; //계속
		printf("%-16s",p->d_name);
		if((i+1)%5==0) //5 번 넘으면
			printf("\n"); //개행출력
		i++;
	}
	printf("\n");
	closedir(dp); //닫음
	return 0;
}
```

//옵션을 사용하기위한 ls 모듈

ls_moudle2.c

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

int main(int argc, char **argv)
{
	int cmd;
	while((cmd = getopt(argc, argv, "ab")) > 0) //getopt 가 옵션처리해서
argc 인자갯수, argv 문자열, “ab”//처리할 옵션
현재 처리할 인자가 몇개가왔는지를 감지하기위해
만약 c 옵션을 감지할거면 c 를적으면됨
-a 옵션을 집어넣으면 a 에 걸림
	{
		switch(cmd) // 걸리면 cmd 에 a 가 걸림
		{
			case 'a':
				printf("a option"); //a 옵션걸리면 a option 출력
				break;
			case 'b':
```

```
                                        printf("b option");
                                default:
                                        printf("unknown option\n"); //c 는 없기때문에 디
폴트에 걸려서 출력
                        }
                }
                return 0;
}
```

옵션넣기 :

./a.out -a
결과 :
a option

./a.out -ba
결과 :
b optionunknown option
a option

./a.out -b
결과 :
b optionunknown option

./a.out -abc
결과 :
a optionb optionunknown option
./a.out: invalid option -- 'c'
unknown option

---

main 이 옵션을 받을수있게됨 전처럼 void 하면 안됨

ls_moudle3.c

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


int main(int argc, char **argv)
{
        DIR *dp;
        int i=0,cmd;
        struct dirent *p;
        cmd=getopt(argc, argv, "a");
        dp=opendir(".");
        while(p=readdir(dp))
        {
                if(cmd != 'a') //a 가 살아있지않으면
                {
                        if(p->d_name[0] == '.') //. 도 출력
                                continue;
                }
                printf("%-16s",p->d_name);
                if((i+1)%5==0)
                        printf("\n");
                i++;
        }
```

```
		printf("\n");
		closedir(dp);
		return 0;
}
```

. 의 의미
.. // 상위
. // 현재

 -a //숨김파일이든 뭐든 다보여주게됨, 메모리공간과 속도측면향상을 위해 필요한것

비트연산

예제 2 를 이용해서 옵션 일곱개 만들어서 옵션이 무엇이 사용됬는지 파악해야함

퀴즈

int tmp = 어던 세팅이되었는지확인

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

#define	A		(1 << 0)
#define	B		(1 << 1)
#define	C		(1 << 2)
#define	D		(1 << 3)
#define	E		(1 << 4)
#define	F		(1 << 5)
#define	G		(1 << 6)

void check_flag(int flag)
{
		int i, tmp = flag;

		for(i = 0; i < 7; i++)
		{
			switch(tmp & (1 << i))
			{
				case 1:
					printf("A\n");
					break;
				case 2:
					printf("B\n");
					break;
				case 4:
					printf("C\n");
					break;
				case 8:
					printf("D\n");
					break;
				case 16:
					printf("E\n");
					break;
				case 32:
					printf("F\n");
					break;
				case 64:
					printf("G\n");
```

```
                                        break;
                        }
                }
}

int main(int argc, char **argv)
{
        int cmd;
        int flag;

        /* 7 개의 옵션: a, b, c, d, e, f, g */
        while((cmd = getopt(argc, argv, "abcdefg")) > 0)
        {
                switch(cmd)
                {
                        case 'a':
                                flag |= 1 << 0;
                                printf("a option\n");
                                break;
                        case 'b':
                                flag |= 1 << 1;
                                printf("b option\n");
                                break;
                        case 'c':
                                flag |= 1 << 2;
                                printf("c option\n");
                                break;
                        case 'd':
                                flag |= 1 << 3;
                                printf("d option\n");
                                break;
                        case 'e':
                                flag |= 1 << 4;
                                printf("e option\n");
                                break;
                        case 'f':
                                flag |= 1 << 5;
                                printf("f option\n");
                                break;
                        case 'g':
                                flag |= 1 << 6;
                                printf("g option\n");
                                break;
                        default:
                                printf("unknown option\n");
                }
        }

        check_flag(flag);

        return 0;
}
```

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

```
int main(int argc, char **argv)
{
	DIR *dp;
	int i=0, cmd;
	struct dirent *p;
	int flag = 0;
	while((cmd=getopt(argc, argv, "abRi")) > 0)
	{
		switch(cmd)
		{
			case 'a':
				flag |= 1;
				break;
			case 'b':
				flag |= 2;
				break;
			case 'R':
				flag |= 4;
				break;
			case 'i':
				flag |= 8;
				break;
		}
	}
	dp=opendir(".");
	while(p=readdir(dp))
	{
		if(!(flag&1)) // a 옵션 있으면 거짓 .을제끼지않음
		{
			if(p->d_name[0]=='.')
				continue;
		}
		printf("%-16s",p->d_name);
		if((i+1)%5==0)
			printf("\n");
		i++;
	}
	printf("\n");
	closedir(dp);
	return 0;
}
```

첫번째옵션이 있으면 . 출력
없으면 . 없애기

./a.out -a

옵션

ls -R 은 디렉토리 가장 밑에서 하는게 좋음
디렉토리를 순회하면서 쭉돔

ls -i 하면 숫자가나옴

ls -li 하면
또 숫자가 나오는데 i node 번호 이다

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

int main(int argc, char **argv)
{
	struct stat buf; //stat = 상태(정보) 보는것
	char ch;
	stat(argv[1],&buf); //1 번째로 받는 인자를 받아서 버퍼에 저장하겠다
	if(S_ISDIR(buf.st_mode)) //디렉토리인지(버퍼 에스티모드에 들어있음)
		ch='d';
	if(S_ISREG(buf.st_mode)) //레귤러.일반파일
		ch='-';
	if(S_ISFIFO(buf.st_mode)) //파이프
		ch='p';
	if(S_ISLNK(buf.st_mode)) //바로가기파일
		ch='l';
	if(S_ISSOCK(buf.st_mode)) //소켓
		ch='s';
	if(S_ISCHR(buf.st_mode)) //캐릭터 디바이스
		ch='c';
	if(S_ISBLK(buf.st_mode)) //블록디바이스
		ch='b';

	printf("%c\n",ch); //최종적으로 프린트에프로 들어온 파일에 대한 정보출력
	return 0;
}
```

---

ls_moudle6.c

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

int main(int argc, char **argv)
{
	struct stat buf;
	char ch;
	char perm[11] = "----------";
	char rwx[4] = "rwx";
	int i;
	stat(argv[1],&buf);

	if(S_ISDIR(buf.st_mode))
		perm[0]='d';
	if(S_ISREG(buf.st_mode))
		perm[0]='-';
	if(S_ISFIFO(buf.st_mode))
		perm[0]='p';
	if(S_ISSOCK(buf.st_mode))
		perm[0]='l';
	if(S_ISCHR(buf.st_mode))
```

```
                perm[0]='s';
        if(S_ISBLK(buf.st_mode))
                perm[0]='b';
        for(i=0; i<9; i++) //i 가 0 부터 9 까지 채워넣을거임
                if((buf.st_mode>>(8-i))&1) //0
                        perm[i+1]=rwx[i%3];
        printf("%s\n",perm);
        return 0;
}
```

8 진수 쓰이는곳 : 파일의 권한지정
rwx
3 비트 ->8 개를 나타낼수있음

맨앞에 파일의 종류

( )	rwx	( )	( )
파일의	root	그룹	제 3 자
종류	및 사용자 권한	실행권한	권한지정
	read		
	write		

655 // 사용자는 읽고쓰기가 가능하고 그룹은 읽는것만 가능하다
755 // 읽고쓰고 실행까지 가능, 나외 다르사람들은 쓰기권한 X

- struct stat

				set uid	set gid	Sticky bit	r	w	x	r	w	x	r	w	x

맨 앞의 4bit 는 파일의 종류를 나타낸다

ls_moudle7.c

```
#include<sys/types.h>
#include<sys/stat.h>
#include<unistd.h>
#include<stdio.h>
#include<pwd.h>
#include<grp.h>

int main(int argc, char **argv)
{
        struct stat buf;
        struct passwd *pw;
        struct group *gr;
        char ch;
        char perm[11]="----------";
        char rwx[4]="rwx";
        int i;
        stat(argv[1],&buf);

        if(S_ISDIR(buf.st_mode))
                ch='d';
        if(S_ISREG(buf.st_mode))
                ch='-';
        if(S_ISFIFO(buf.st_mode))
                ch='p';
        if(S_ISLNK(buf.st_mode))
                ch='l';
```

```
		if(S_ISSOCK(buf.st_mode))
			ch='s';
		if(S_ISCHR(buf.st_mode))
			ch='c';
		if(S_ISBLK(buf.st_mode))
			ch='b';
		for(i=0; i<9; i++)
			if((buf.st_mode >> (8-i))&1)
				perm[i+1]=rwx[i%3];
		printf("%s",perm);
		printf("%lu",buf.st_nlink); //파일갯수라고도 볼수있는데
관리포인트가 분할되면 숫자가 정확히 안맞을수있음
		pw=getpwuid(buf.st_uid);
		printf("%s",pw->pw_name);
		gr=getgrgid(buf.st_gid);
		printf("%s",gr->gr_name);
		return 0;
}

./a.out .
./a.out ..

ls -a ..

getgrgid

ls -l 치면 각각 몇개 있는지 나오고 뒤에 계정명이 나옴
맨앞에있는게 유저아이디
sdr 뒤에 sdr 이 또있는데 그게 그룹아이디
```

---

ls_moudle8.c

```
#include<sys/types.h>
#include<sys/stat.h>
#include<unistd.h>
#include<stdio.h>
#include<pwd.h>
#include<grp.h>
#include<time.h>

int main(int argc, char **argv)
{
		struct stat buf;
		struct dirent *p;
		struct password *pw;
		struct group *gr;
		struct tm *tm;
		char ch;
		char perm[11]="----------"
			char rwx[4]="rwx";
		char sst[4]="sst";
		int i;
		stat(argv[1],&buf);

		if(S_ISDIR(buf.st_mode))
			perm[0]='d';
		if(S_ISREG(buf.st_mode))
```

```
                perm[0]='-';

        if(S_ISFIFO(buf.st_mode))
                perm[0]='p';
        if(S_ISSOCK(buf.st_mode))
                perm[0]='s';
        if(S_ISCHR(buf.st_mode))
                perm[0]='c';
        if(S_ISBLK(buf.st_mode))
                perm[0]='b';
        for(i=0; i<9; i++)
                if((buf.st_mode>>(8-i))&1)
                        perm[i+1]=rwx[i%3];
        for(i=0;i<3;i++)
                if((buf.st_mode>>(11-i))&1)
                        if(perm[(i+1)*3]=='-')
                                perm[(i+1)*3]=sst[i]^0x20;
                        else
                                perm[(i+1)*3]=sst[i];
        printf("%s",perm);
        printf("%d",buf.st_nlink);
        pw=getpwuid(buf.st_uid);
        printf("%s",pw->pw_name);
        gr=getgrgid(buf.st_gid);
        printf("%s",gr->gr_name);
        printf("%d",buf.st_size);
        tm=localtime(&buf.st_mtime);
        printf("%d-%02d-%02d %02d:%02d",tm->tm_year+1900,tm-
>tm_mon+1,tm->tm_mday,tm->tm_hour,tm->tm_min);
        printf("\n");
        return 0;
}
```

sst 관련된걸 볼수없을것
강제적으로 sst 를줄수있도록 작업을 해줘야하는데
ls -l a.txt 를 해보면
S 하고 T 가보임
ls -l root 하면

chmod 4644 a.txt
빨강색됨
실행권한이 없으면
대문자 S 로

chmod 4755 a.out
실행권한이 있으면
소문자 S 됨

sst 를 작업하는자리
rwx 를 끝낸자리

11-i 에 &

^ =
익스쿠션 권한이 있으면 0x20
없으면 (else
rwx 가 rws 로 바뀌는거
setuid 루트의권한을 부여하겠다

setgid? 그룹의 권한을 부여하겠다
맨뒤에꺼나 익스쿠션자리
t 는 디렉토리에 붙으면 공유폴더

chmod 1755 a.out
a.out 에 t 가 생겼는데 여기의 t 는
스티키비트?
나는 이곳에 올린걸 모두 캐시할거야?
하드디스크에 올라간

리눅스커널 내부구조
122 페이지
하드디스크에 대신 넣는거를 스왑이라고함
가상메모리를 물리메모리로 바꿔주는 과정을 페이징이라 함

122 는 32 비트 페이징
123 은 64 비트 페이징

122
a.out 실행
skiki bit

프로세스가됨

task_struct
	files_struct
		file
		path
		inode
		super_block
	mm_struct //허상, 경계값만 있음.
	mm_area_struct
	[가상 메모리 레이아웃]

가상메모리 - > 물리메로리 변환
0xbf884c8c

---

10 비트 10 비트 12 비트 :10 비트씩 나누기
b 를 1011
f 는 1111
8 은 1000
8 은 1000
4 는 0100
c 는 1100
8 은 1000
c 는 1100
[1011 111110][00 1000 0100] [1100 1000 1100]
    10bit   10bit   12bit

1023   128   2048 →2 의 11 승
 257   4   1024
 766   132   128
                                      12
                                    3212

최종적으로 옵셋이 피지컬메모리를 가리키고있음
물리메모리를 찾는작업

DRAM(물리메모리)

memory hierachy

속도↑ Register 공간↓
cache
memory
disk

물리메모리가 부족해지면 스왑의 힘을 빌리게됨

swap 매커니즘 : 범인은 반드시 왔던자리에 돌아온다
사용했던 물리메모리를 재활용할수있다

swqp 이 사용하는 알고리즘이 LRU 알고리즘

페이징은
용량이 큰걸 빨리 처리할수있도록 소프트캐시에 붙이기위한 옵션

공간활용보다는 응답속도를 위한[1]것

1