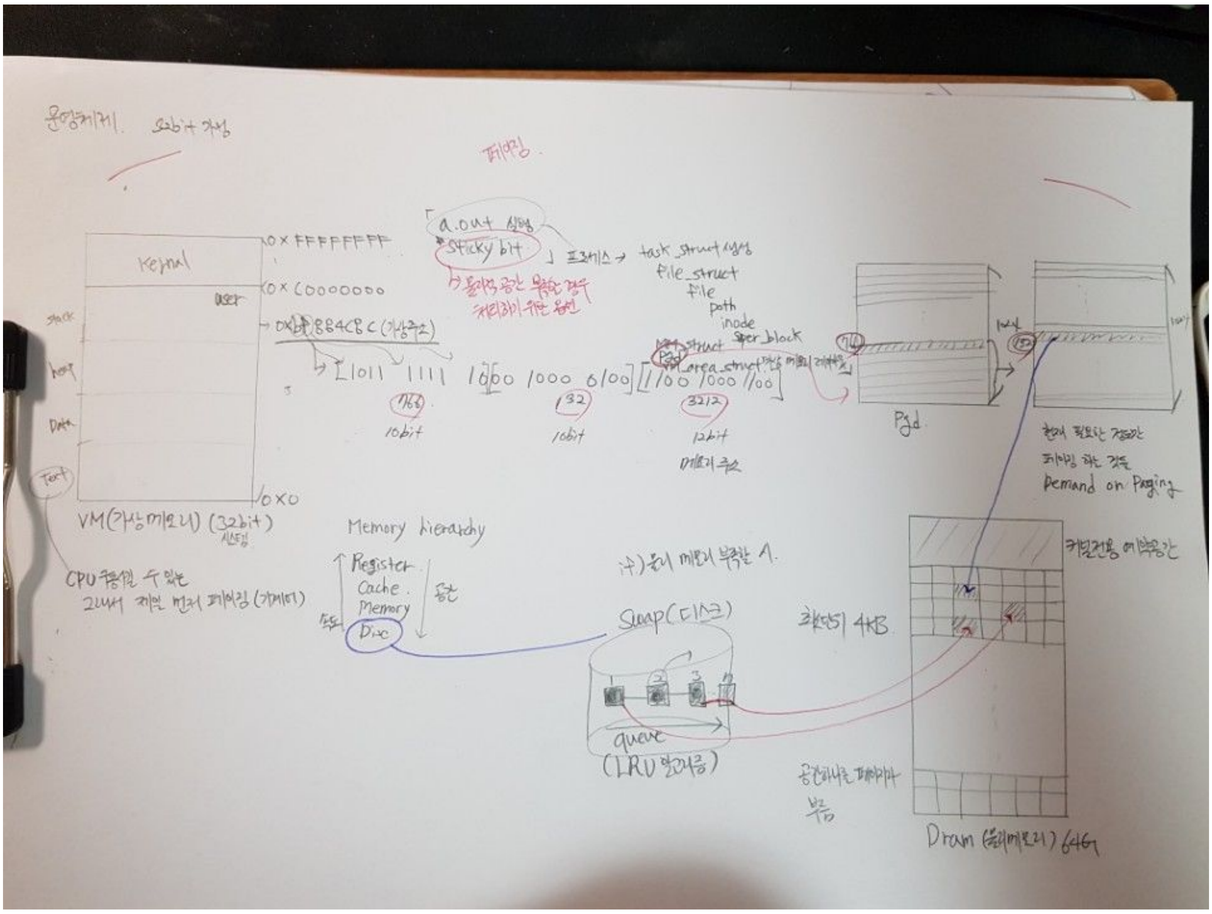
Xilinx Zynq FPGA, TI DSP,MCU기반의 프로그래밍 및 회로 설계 전문가 과정

강사 - Innov (이상훈)

gcccompil3r@gmail.com

학생 - 이유성

dbtjd1102@naver.com



운영체제. 32bit 가정
페이징.
a.out 실행
sticky bit
프로세스 → task_struct 생성
file_struct
file
path
inode
super_block
mm_struct
pgd
area_struct
0×FFFFFFFF
kernel
0×C0000000
user
0×BF884CBC (가상주소)
1011 1111 1000 1000 0100 1100 1000 1100
766
32
3212
10bit
10bit
12bit
Stack
heap
Data
Text
0×0
VM(가상메모리) (32bit)
Pgd.
Demand on Paging
Memory hierarchy
Register
Cache
Memory
Disc
Swap(디스크)
queue
(LRU 알고리즘)
4KB
Dram (물리메모리) 64G

file1.c

```
#include<sys/types.h>
#include<dirent.h>  //구조체 DIR *dp struct dirent *p 내부 함수 opendir() readdir()closedir()
#include<stdio.h>

int main(void)
{
        DIR *dp ;  // DIR 디렉토리의 많은 정보를 저장할 수 있는 필드              구조체를 가리키는 포인터라고
생각하면 됨
        int i= 0;
        struct dirent *p;//디렉토리에 있는 내용물 리스트 저장 구조체
        dp = opendir(".");//opendir()함수는 열고자 하는 디렉토리 경로를 입력으로 받고  DIR 구조체에 대한
포인터를 반환 만약 해당 디렉토리 경로가
 존재하지 않거나 퍼미션등의 이유로 실패한다면 NULL 반환
        //현재 디렉토리 심볼 .
        while(p = readdir(dp))//*dp가 가르키는 곳       리스트가 있으면 루프 돌아
        {
        if(p->d_name[0] =='.')  //맨앞이 . 이면 패스 (숨김 파일)
                continue;
        printf("%-16s",p->d_name);
        if((i+1)%5 ==0)
        printf("\n");
        i++;
        }
        printf("\n");
        closedir(dp);
        return 0;
}

//ls 구현

// 디렉토리 - 모든것은 파일이다.

#include<sys/stat.h>
내부 구조체

struct stat ->파일의 상태를 담은 구조체
struct stat{
    _dev_t st_dev;
    _ino_t st_ino;
    unsigned short st_mode;
    short st_nlink;
    short st_uid;
    shjor st_gid;

    _off_t st_size;
    time_t st_atime;
    time_t st_mtime;
    time_t st_ctime;
    };
```

file2.c

```
#include<stdio.h>
#include<unistd.h>
#include<fcntl.h>
                        //option 넣기전에 터미널에 ls -al쳐봐
int main(int argc, char **argv)
{
        int cmd; // (command)
        while((cmd = getopt(argc,argv,"ab")) >0)  //getopt 인자개수 , 인자의주소(문자열) , 처리할 수 있는 옵션

        {                       //옵션을 감지 하기위해 적어,
        switch(cmd)             // 하나씩 받아와서 c,d같은경우는 ?로 반환
        {
                case 'a':
                        printf("a option\n");
                        break;
                case 'b':
                        printf("b option\n");
                        break;
                default:
                        printf("unkonwn option\n");
        }
        }
        return 0;
}

// ./a.out -a ./a.out -b ./a.out -ab ./a.out -abc

// ba 였을때 b  ,  a 옵션 나오는
```

file3.c

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

// .. 이전(상위) .현재

int main(int argc, char **argv)
{
        DIR *dp;  //디렉토리를 가르키는 포인터
        int i =0 , cmd;
        struct dirent *p;//디렉토리 안에있는 파일
        cmd = getopt(argc,argv,"a");
        dp = opendir(".");//현재 디렉토리를 열겠다. 반환값 : 오픈한 디렉토리의 주소값."위치"위치를
오픈.
        while(p = readdir(dp))
        {
        if(cmd != 'a')
        {
                if(p->d_name[0] =='.') // .을 숨김 파일로..
                        continue;
        }
        printf("%-16s",p->d_name);
        if((i+1) % 5 ==0)
                printf("\n");
        i++;
        }
        printf("\n");
        closedir(dp);
        return 0;
}


//  ./a.out          ./a.out -a
```

file4.c

```
#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,"alRi"))>0)
	{
		switch(cmd)
		{
			case 'a':
				flag |=1;
				break;
			case 'l':
				flag |=2;
				break;
			case 'R':
				flag |=4;
				break;
			case 'i':
				flag |=8;
				break;
		}

	}

	dp = opendir(".");
	while( p = readdir(dp))
	{
		if(!(flag & 1))	//  첫번째 옵션이 있는지 없는지		조건에 따라.
		{
			if(p->d_name[0] =='.')//a옵션없으면 .파일 빼.
				continue;
		}
		printf("%-16s",p->d_name);
		if((i +1) % 5 ==0)
			printf("\n");
		i++;
	}
	printf("\n");
	closedir(dp);
	return 0;
}
```

파일의 상태를 확인하고 stat구조체 안의 st_mode를 통해 파일의 종류를 알아내고 출력.

file5.c

```
#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 보는것 해당 파일의 정보를 얻어옴,
	char ch;
	stat(argv[1], &buf);
	if(S_ISDIR(buf.st_mode))
	ch ='d';		//디렉토리(도 파일)  st_mode 에서 판단,
	if(S_ISREG(buf.st_mode))
	ch = '-';		//파일명		(일반)
	if(S_ISFIFO(buf.st_mode))
	ch ='P';		//파이프		(파이프)
	if(S_ISLNK(buf.st_mode))
	ch ='l';		//링크		(바로가기)
	if(S_ISSOCK(buf.st_mode))
	ch = 's';		//소켓		(소켓)
	if(S_ISCHR(buf.st_mode))
	ch = 'c';		//캐릭터		(캐릭터디바이스)
	if(S_ISBLK(buf.st_mode))
	ch = 'b';		//블록		(블록디바이스)
	printf("%c\n",ch);

	return 0;
}
```

file7.c

```
#include<sys/types.h>
#include<sys/stat.h>
#include<unistd.h>
#include<pwd.h>
#include<grp.h>
#include<stdio.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))
        perm[0] = '-';
        if(S_ISFIFO(buf.st_mode))
        perm[0] = 'p';
        if(S_ISLNK(buf.st_mode))
        perm[0] = 'l';
        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];
        printf("%s",perm);
        printf("%lu",buf.st_nlink);
        pw = getpwuid(buf.st_uid);
        printf("%s",pw->pw_name);
        gr = getgrgid(buf.st_gid);
        printf("%s",gr->gr_name);
        return 0;
}
```

// ./a.out . 자기 자신의 디렉토리에 있는 파일 수(잘 안맞음) ./a.out 파일 = 자기 자신이니까 1

file6.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";  //파일 권한 같은거 표현할때 8진수 사용        read write
x(execute)실행권한 755 타인에게 쓸권한을 주지 않겠다.
        int i;                  // [파일종류 ,rwx,rwx,rwx] [파일종류 ,root/사용자 권한,그룹권한,타인(제3자)]
        .st_mode16비트 앞에 4칸 이름, ~~
        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_ISLNK(buf.st_mode))
        perm[0] = 'l';
        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)       //          st_mode 16칸 앞부분 종류 ,  rwxrwxrwx 다1
                perm[i+1] = rwx[i % 3];
        printf("%s\n",perm);                //  i 0 = perm[1] =rwx[0]


        return 0;
}
```

file7.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 passwd *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 ++)			//st_mode에 rwx가 있으면 perm배열에 rwx저장
		if((buf.st_mode >>( 8 -i)) & 1)			// | | | | | | | | |
			perm[i + 1 ] = rwx[i % 3];		//		|1|&
	for(i =0 ; i<3 ; i++)
		if((buf.st_mode >>(11-i)) & 1)
			if(perm[(i+1)*3] =='-')// 비어있다면,
				perm[(i + 1) *3 ] =sst[i]^ 0x20;//rwx없으면 대문자
			else
				perm[(i + 1) *3 ] =sst[i];//rwx있으면 소문자


	printf("%s",perm);
```

```
        printf("%lu",buf.st_nlink);   //hardlink된 횟수 출력
        pw = getpwuid(buf.st_uid);  //유저
        printf("%s",pw->pw_name);
        gr = getgrgid(buf.st_gid);
        printf("%s",gr->gr_name);
        printf("%lu",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);//마지막 수정 시간
        printf("\n");
        return 0;
}
```