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

<리눅스 시스템 프로그래밍>
2018.03.22 - 21 일차

강사 - 이상훈
gcccompil3r@gmail.com

학생 - 안상재
sangjae2015@naver.com

- 명령어 , 시스템 콜 및 함수 등 정리

- DIR *[변수명] : 디렉토리 구조체를 가리키는 포인터
- touch [파일명] : 파일 만들기
- getopt(argc, argv, “옵션명”) : “ ” 안의 옵션명을 찾아서 해당 옵션명을 반환한다.
ex) ./a.out -a, ./a.out -ba

- ls -a: 숨긴파일을 모두보여줌.
- stat(argv[1], &buf) : argv[1] 파일의 상태를 보여줌.

- getopt(argc,argv,”a”) : argv 파일 중에 a 옵션을 찾고 a (옵션명)을 반환한다.

- DIR *opendir(const char *dirname) : 변수 dirname 에 지정된 디렉토리 스트림을 열어 처음을 가리킴.

ex) opendir(“.”) : 현재 디렉토리를 open 함.

- struct dirent *readdir(DIR *dirp) : dirp 가 가리키는 디렉토리 내 파일 목록의 주소를 반환함.

- int closedir(DIR *dp) : dp 가 가리키는 디렉토리 파일을 닫아준다.

- struct dirent{ // 디렉토리 구조체

long d_ino; // I- 노드번호(삭제된 파일은 I-노드번호가 0)

off_t d_off; // offset

unsigned short d_reclen; // 파일 이름 길이

char d_name[NAME+MAX+1] // 파일 이름

}

- struct stat{ // 파일 상태 구조체

dev_t st_dev; // 파일의 device 의 ID

ino_t st_ino; // inode number

mode_t st_mode; // 파일의 종류 및 접근권한

nlink_t st_nlink; // hardlink (복사본) 된 횟수

uid_t st_uid; // 파일의 owner

gid_t st_gid; // group ID of owner

dev_t st_rdev; // device ID (if special file)

off_t st_size; // 파일의 크기(bytes)

blksize_t st_blksize; // 파일 시스템 IO 의 blocksizes

blkcnt_t st_blocks; // 할당된 512B 블록의 수

time_t st_atime; // 마지막 접근의 시간

time_t st_mtime; // 마지막 수정의 시간

time_t st_ctime; // 마지막 상태 변화의 시간

}

* 파일의 종류를 체크하는 함수

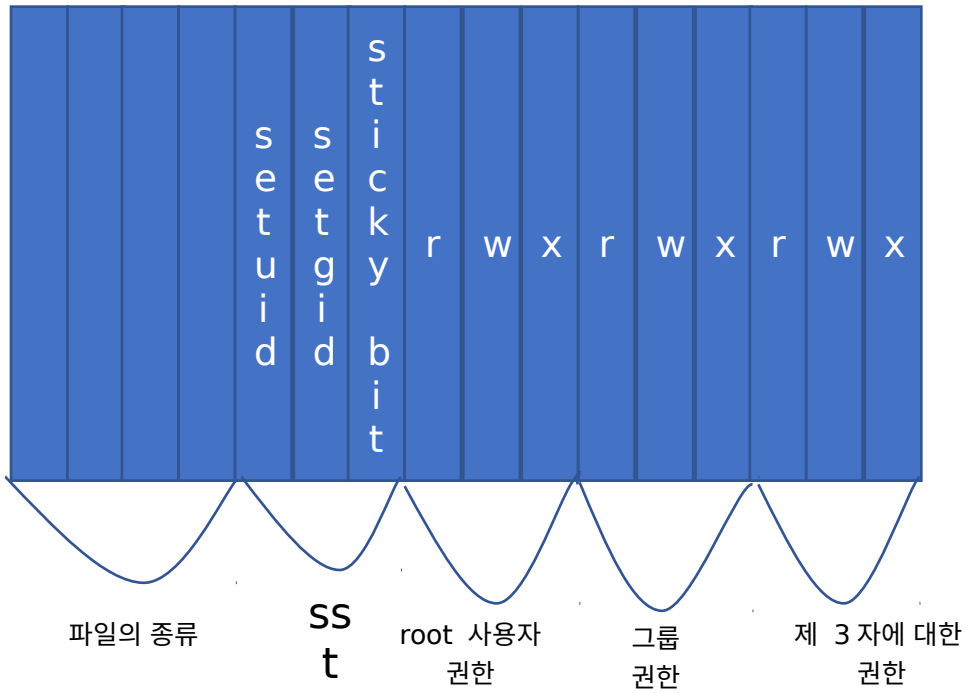
struct stat buf;

- S_ISREG(buf.st_mode) : regular file
- S_ISDIR(buf.st_mode) : directory file
- S_ISBLK(buf.st_mode) : block special file
- S_ISCHR(buf.st_mode) : character special file
- S_ISLNK(buf.st_mode) : link file
- S_ISFIFO(buf.st_mode) : pipe file
- S_ISSOCK(buf.st_mode) : socket file

* inode : 운영체제의 파일과 하드시스크의 원본파일을 연결해주는 연결통로이다. inode 에는 해당파일에 대한 정보들이 저장되어 있다.

-ls -I : inode 번호 출력
-ls -li : inode 리스트 출력
-ls -a : 숨김파일을 포함한 모든 파일을 출력함.

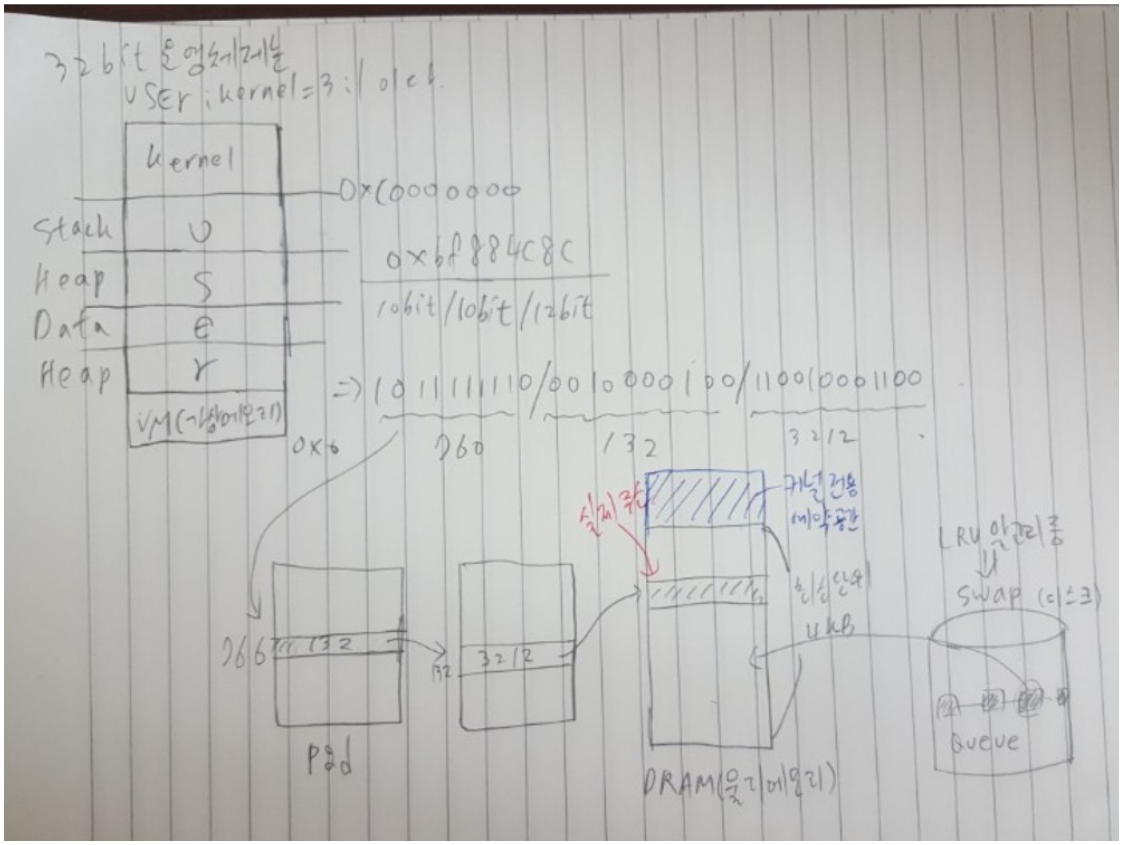
* st_mode (struct stat 안의 멤버)



- r : 읽기
- w : 쓰기
- x (execution) : 실행
- set gid : 해당 권한이 설정되어 있는 파일을 실행할 경우, 이 파일의 그룹 권한을 가진다.
- set uid : 해당 권한이 설정되어 있는 파일을 실행할 경우, 이 파일의 소유자 권한을 가진다.
- sticky bit : 해당 권한이 설정되어 있는 디렉토리는 모든 유저가 공유를 한다.(삭제, 생성 가능)
  swap 으로 사용할 경우에도 붙인다.

- 운영체제의 페이징

-> 운영체제의 가상메모리를 실제 물리메모리에 맵핑 시키는 과정.
-> 스와핑은 디스크 속의 프로그램을 주메모리(DRAM)에 올려서 실행시키기 위해, 페이징 단위로 나누고 Queue 형태로 자료구조를 만들어서, 일정한 규칙에 의해 주메모리에 올리는 과정을 말한다. 이렇게 함으로써 디스크 속의 프로그램의 용량이 주 메모리 용량보다 훨씬 크더라도 장시간의 loading 시간 없이 실행을 할 수 있게 되는 것이다. LRU 알고리즘을 사용함.
→ 현재 필요한 정보만을 페이징 하는 것을 Demand on Paging 이라고 함.



- 현재 디렉토리에서 숨긴파일을 제외한 모든 파일을 출력함. (ls 명령어에 해당함.)

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

int main(void)
{
    DIR *dp;
    int i=0;
    struct dirent *p;  // 디레고리 구조체의 포인터변수 선언
    dp = opendir("."); // 현재 디렉토리의 주소를 받음.
    while(p = readdir(dp))  // 현재 디렉토리를 read 해서 처음 파일의 주소를 받음.
    {
        if(p->d_name[0] == '.') // 숨긴파일
            continue;         // 숨긴파일은 출력하지 않음.
        printf("%-16s ", p→d_name);   // 숨긴파일이 아니면 파일이름 출력
        if((i+1)%5 == 0)
            printf("\n");
        i++;
    }
    printf("\n");
    closedir(dp);
    return 0;
}
```

- main 의 인자로 들어온 파일 중에 a,b 를 옵션으로 하는 파일을 찾아서 출력함.

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

int main(int argc, char **argv)
{
    int cmd;
    while((cmd = getopt(argc,argv,"ab")) > 0) a,b 옵션이 있는지 검사하고 cmd 에 찾은 옵션값을 반환함.
    {
        switch(cmd)
        {
            case 'a':
                printf("a option\n");
                break;
            case 'b':
                printf("b option\n");
                break;
            default:      // a,b 옵션을 찾지 못했을 때
                printf("unknown option\n");
        }
    }
    return 0;
}
```

- 파일의 옵션에 따라 flag 라는 변수에 값을 다르게 셋팅하고, flag 값에 따라 숨김파일을 보여줄지 안보여줄지 결정함.
옵션이 많아지면 메모리 공간을 많이 차지 하기 때문에, 공간효율성, 속도를 증가시키기 위해 비트연산을 활용함.

```
#include <stdio.h>
#include <fcntl.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)    // 파일의 옵션이 alRi 중에 있는지 검사함.
    {
        switch(cmd)
        {
```

```
            case 'a':
                flag |= 1;
                break;
            case 'l':
                flag |= 2;
                break;
            case 'R':
                flag |=4;
                break;
            case 'i':
                flag |=8;
                break;
        }
    }
    dp = opendir(".");              // 현재 디텍토리를 open 함.
    while(p = readdir(dp))          // dp 디렉토리를 read 해서 파일의 포인터를 반환함.
    {
        if(!(flag & 1))         // 옵션이 a 가 아니면 숨김파일 안 보여줌, a 이면 숨김파일 보여줌.
        {
            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;
}
```

- 파일의 상태를 확인하고, stat 구조체 안의 st_mode 멤버를 통해 파일의 종류를 알아내고 출력함.

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

int main(int argc, char **argv)
{
    struct stat buf;
    char ch;

    stat(argv[1], &buf);

    if(S_ISDIR(buf.st_mode))        // 디렉토리
        ch = 'd';
    if(S_ISREG(buf.st_mode))      // 레귤러
        ch = '-';
    if(S_ISFIFO(buf.st_mode))     // 파이프
        ch = 'p';
    if(S_ISLNK(buf.st_mode))      // 링크
        ch = 'l';
    if(S_ISSOCK(buf.st_mode))   // 소켓
        ch = 's';
    if(S_ISCHR(buf.st_mode))      // 캐릭터
        ch = 'c';
    if(S_ISBLK(buf.st_mode))      // 블록
        ch = 'b';
```

```
    printf("%c\n",ch);

    return 0;
}
```

- 파일의 종류를 확인하고, 사용자 권한, 그룹권한, 제 3 자의 권한의 r,w,x 유무를 체크함. 또한 setuid, setgid, sticky bit 의 유무를 체크해서 sst 의 대문자 또는 소문자를 출력함.

```
#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++)
        if((buf.st_mode >> (8-i)) & 1)      // st_mode 에 r,w,x 가 있으면, perm 배열에 r,w,x 를 저장함.
            perm[i+1] = rwx[i%3];
    for(i=0;i<3;i++)
        if((buf.st_mode >> (11-i)) & 1)   // setuid, setgid, sticky bit 이 있으면
            if(perm[(i+1)*3] == '-')        // r,w,x 가 없는지 확인함
                perm[(i+1)*3] = sst[i] ^ 0x20;     // r,w,x 가 없으면 대문자 S,S,T 를 저장함.
            else
                perm[(i+1)*3] = sst[i];     // r,w,x 가 있으면 소문자 s,s,t 를 저장함.

    printf("%s", perm);                     // perm 출력
    printf("%lu", buf.st_nlink);            // hardlink 된 횟수 출력
    pw = getpwuid(buf.st_uid);              // 파일 소유자의 id 출력
    printf("%s", pw→pw_name);               // 파일 이름 출력
    gr = getgrgid(buf.st_gid);              // 그룹의 id 출력
    printf("%lu", buf.st_size);             // 파일의 크기 출력
    tm = localtime(&buf.st_mtime);          // 파일의 마지막 수정 시간 출력
```

```
    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;
}
```