# Xilinx Zynq FPGA,TI DSP,
# MCU 기반의
# 프로그래밍 전문가 과정

날 짜 : 2018 . 3. 22

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 정한별
hanbulkr@gmail.com

# <ls_module>

-opendir : 인자로 들어오는 “~” 녀석의 주소를 열겠다. (return 값 : 인자의 주소)
-readdir : 인자로 받은 주소의 있는 것을 읽겠다.
-getopt(argc, argv, ‘~’): ‘~’ 안의 옵션을 argv 라는 주소에서 받겠다.
-closedir : 디렉토리 0pen 시 마지막에 닫아 주어야 한다.

### * 옵션을 효율적으로 관리하기 위해서, 이진 연산을 해준다.

ls -R : 현재 부터 하위 모두 출력
ls -l : 그냥 보이는 리스트만 (현재 디렉토리만) 출력.
ls -i : node 파일 출력
ls -a : 전체 출력

### * 파일의 종류 및 권한. [ r , w , x ] → 8 진수

**r** : 읽기
**w** : 쓰기
**x** : 실행(excutive)

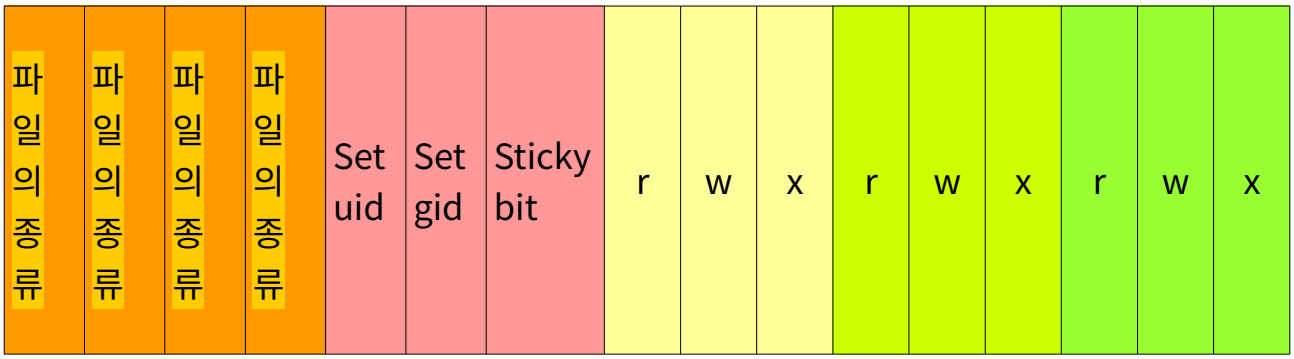
### *표현 되는 모양

__ (r w x)(r w x) (r w x)

종류 사용자 그룹 타인

0 6 4 4

(chmod) 터미널 명령으로 퍼미션 권한을 조정 할 수 있다.




uid 와 gid ,sticky 는 ls 했을 때 우리가 보는 위치에 안나오고 sst 소문자 대문자 형태로 나오는데
**소문자: 현재 bit 에 권한이 있을 경우 이다.**
**대문자: 현재 bit 에 권한이 없을 경우 이다.**

## <ls_module1.c>

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

int main(void)
{
	//DIR 이라는 구조체가 라이브러리에 포함되어 있음.
	DIR	*dp;
	int i = 0;
	//dirent 라는 구조체 변수를 새로 선언함.
	struct dirent *p;
	// opendir 은 '.' 이라는 경로(현재 디렉토리)를 열겠다 라는 뜻
	dp = opendir(".");
	// readdir 은 open 한 경로의 주소 를 받아  p 라는 구조체에 넣는다.  (현재 디렉토리 읽은것들 다 출력)
	while(p = readdir(dp))
	{
		//읽은 구조체 이름 단위중 첫번째가 .이면 출력하지 않음
		if(p -> d_name[0] == '.')
			continue;
		//16 개의 공간에서 왼쪽 정렬로 해서 구조체에 읽은 이름 단위 하나를 출력
		printf("%-16s",p->d_name);
		// 이름 단위 5 개 마다 한줄 개행
		if((i+1)%5 == 0)
			printf("\n");
		i++;
	}
	printf("\n");
	// open 을 했으니 닫아주어야 한다.
	closedir(dp);

	return 0;
}
```

## <ls_module2.c>

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

// 매개변수를 받을 수 있게 한다.
int main(int argc, char **argv)
{
	int cmd;
	// getopt 를 이용해서 내가 받고 싶은 옵션범위와  매개변수의 주소들을 가져온다.( -a,-b ,-ab )
	//  '-' 를 기본적으로 포함하고 옵션을 받을수 있게 함수가 가지고 있다.
	while((cmd = getopt(argc, argv,"ab"))> 0)
	{
		//cmd 는 getopt 에 의해서 옵션이 들어온 순서대로 하나씩 읽는다.
		switch(cmd)
		{
			case'a':
				printf("a option\n");
				break;
			case'b':
				printf("b option\n");
				break;
			default:
				printf("unknown iption\n");
		}
	}
	return 0;
}
```

## <ls_module3.c>

```
int main(int argc,char **argv)
{
   DIR *dp;
   int i=0,cmd;
   struct dirent *p;
// a 라는 옵션인자를 받겠다 라는 뜻.
   cmd=getopt(argc,argv,"a");
// 현재 디렉토리 오픈
   dp=opendir(".");
//p 에 현재 디렉토리 오픈 한것을 읽어서 p 라는 구조체 에다가 반환함
   while(p=readdir(dp))
   {
          // a 가 옵션으로 들어 있지 않으면  '.' 이 들어있는 파일은 프린트 하지 않음.
          if(cmd!='a')
          {
            if(p->d_name[0]=='.')
                  continue;
          }
          // 16 개의 공간에 받은 이름 덩어리를 순서대로 왼쪽 정렬한다. 나머지 공간에 null 이 들어감 .
          printf("%-16s ",p→d_name);
          // 5 개 씩 받아서 개행한다.
          if((i+1)%5==0)
            printf("\n");
          i++;
   }
   printf("\n");
// 열었던 디렉토리를 닫는다.
   closedir(dp);
   return 0;
}
```

## <ls_module4.c>

```
int main(int argc,char **argv)
{
   DIR *dp;
   int i=0;
   int cmd;
   struct dirent *p;
   int flag=0;
   while((cmd=getopt(argc,argv,"alRi"))>0)
   {
          switch(cmd)
          {
             case 'a':
                    // flag 를 사용해서 현재 가지고 있는 옵션 정보를 저장한다.
                    // 받은 옵션이 총 4 개 이면 순서대로 1111 이진수로 들어감 . 각자리에 하나씩 있다고 표현한거임.
                    flag |=1;
                    break;
             case 'l':
                    flag |=2;
                    break;
             case 'R':
                    flag |=4;
                    break;
             case 'i':
                    flag |=8;
                    break;
          }
   }
//현재 디렉토리 오픈
   dp=opendir(".");
// 현재 디렉토리에 읽은 것을 구조체 p 에 저장
   while(p=readdir(dp))
   {
          if(!(flag&1))
          {
          // 구조체 p 에 저장된 이름중 '.' 이 있으면 무시한다.
             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;
}
```

## <ls_module5.c>

```
int main(int argc,char **argv)
{
// stat 이라는 구조체 변수를 만듬.
   struct stat buf;
   char ch;
//현재 들어온 인자의 정보를 buf 에 저장 ( struct stat)
   stat(argv[1],&buf);
// st_mode 는 현재 읽으 파일의 종류와 권한 정보를 가지고 있다.
//  S_ISDIR 는 st_mode 가  directory 파일이면 1 을 반환한다.
   if(S_ISDIR(buf.st_mode))
           ch='d';
//  S_ISREG 는 st_mode 가  일반 파일이면 1 을 반환한다.
   if(S_ISREG(buf.st_mode))
           ch='-';
//  S_ISFIFO 는 st_mode 가  PIPE 파일이면 1 을 반환한다.
   if(S_ISFIFO(buf.st_mode))
           ch='p';
//  S_ISLNK( 는 st_mode 가 링크 파일이면 1 을 반환한다.
   if(S_ISLNK(buf.st_mode))
           ch='l';
//  S_ISSOCK 는 st_mode 가  소켓 파일이면 1 을 반환한다.
   if(S_ISSOCK(buf.st_mode))
           ch='s';
//  S_ISCHR 는 st_mode 가  캐릭터 파일이면 1 을 반환한다.
   if(S_ISCHR(buf.st_mode))
           ch='c';
//  S_ISBLK 는 st_mode 가  블럭 파일이면 1 을 반환한다.
   if(S_ISBLK(buf.st_mode))
           ch='b';
   printf("%c\n",ch);
   return 0;
}
```

## <ls_module9.c> - 앞에 한 나머지 파일이 겹치는 내용 이므로 마지막 꺼만 한다.

```
#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 ;
        // 10 자리의 퍼미션 자리를 표현한다. 마지막 자리는 null 문자가 온다.
        char perm[11]="----------";
        // 퍼미션 rwx 순으로 오기때문에 순서대로 받기 위한 문자열
        char rwx[4] ="rwx";
        // sst 를 통해서 root 권한을 표현 하기 위해서 쓴다.
        char sst[4] ="sst";
        int i;
        //stat 을 통해서 매개 변수로 받은 인자의 정보를 저장한다.
        stat(argv[1],&buf);

        // 인자의 첫 자리의 정보를 확인해서 파일의 종류를 표현해준다.
        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';
        // 8-i 은 첫 자리를  제외하고 0~8 까지의 9 개자리를 표현한 것.(비트연산)
        //Stat struct 의 권한값이 1 이면  그 권한 자리에 맞는 글자를 표현해준다.(rwx 순으로 들어오니’%3’함.)
        for(i= 0; i<9; i++)
                if((buf.st_mode >>(8-i))& 1)
                        perm[i+1] = rwx[i%3];
```

```
	// 11-i 는 set uid, set gid , sticky bit 자리 값에 따라 그자리의 퍼미션을 일시적 슈퍼권한으로 바꿈.
	for(i=0;i<3;i++)
		//이번에는 12 번째부터 3 가지를 봐야함 (set uid, setgid, sticky bit)
		if((buf.st_mode>>(11-i))&1)
			if(perm[(i+1)*3]=='-')
				// 권한이 없으면 대문자. Exclusive OR 로 대소문 차이인 32 를 더하고 뺌
				perm[(i+1)*3]=sst[i]^0x20;
			else
				// 권한이 있으면 소문자.
				perm[(i+1)*3]=sst[i];
	// 저장된 권한 값 출력.
	printf("%s ", perm);
	// 저장된 링크값 출력
	printf("%lu ",buf.st_nlink);
	// uid 값을 가져와 저장함
	pw = getpwuid(buf.st_uid);
	// uid 정보가 가리키는 이름 값 출력 ( user name 출력)
	printf("%s ",pw→pw_name);
	// gid 값을 가져와 저장함
	gr = getgrgid(buf.st_gid);
	// gid 정보가 가리키는 이름 값 출력 (group name 출력)
	printf("%s ",gr→gr_name);
	// 이 파일이 가지고 있는 크기를 출력
	printf("%lu ", buf.st_size);
	// 현재 시간 값을 가져와 저장함
	tm = localtime(&buf.st_mtime);
	// 현재 시간 값을 출력함.
	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;
}
```

## <sticky bit 권한이 있는 실행 파일>

- sticky bit 는 용량이 큰 것의 메모리 관리를 서포트 함 페이즈를 스왑에 넣는다.

## paging 에서 swap 까지의 동작 과정(Disk , swap 에서 동작은 용량이 클시 저장하는 방법) 32bit 운영체제에서 실행시 과정이다.