

TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)

gcccompil3r@gmail.com

학생 – 문한나

mhn97@naver.com

opendir()

헤더 : dirent.h

인수 : 열기 대상 디렉토리

반환값 : 디렉토리 정보 구조체인 DIR 포인터를 반환

예제 1)

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

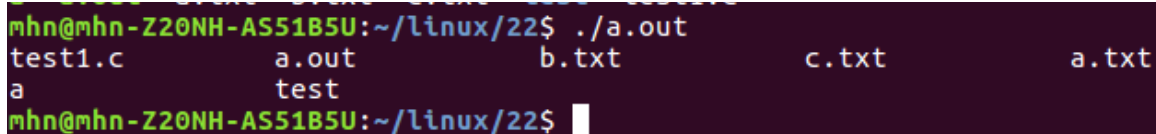
int main(void){

    DIR *dp; // 디렉토리 스트림에 대한 포인터를 반환한다. 포인터가 가리키는 디렉토리의 첫번째 항목
    int i=0;
    struct dirent *p; //디렉토리 안에 있는 리스트
    dp = opendir("."); //현재 디렉토리를 열꺼다 (.)
    while(p = readdir(dp)){ //디렉토리 읽음 디렉토리에 있는 리스트를 p 로 받음

        if(p->d_name[0] == '.') //맨 앞에 글자가 .이면 패스
            continue;
        printf("%-16s",p->d_name); //아니면 출력
        if((i+1) % 5 == 0) //5 번 넘으면
            printf("\n"); //엔터
        i++;

    }printf("\n");
    closedir(dp); //오픈을 했으니 닫아주자

    return 0;
//ls 를 구현했다!!!
}
```



```
mhn@mhn-Z20NH-AS51B5U:~/linux/22$ ./a.out
test1.c      a.out      b.txt      c.txt      a.txt
a            test
mhn@mhn-Z20NH-AS51B5U:~/linux/22$
```

getopt()

헤더 : unistd.h

인수 : 인수의 개수, 인수의 내용, 검색하려는 문자열

반환 : 찾은 문자열 / 지정이 안되어 있는 문자열이라면 ?, 옵션이 없으면 -1

예제 2)

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

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

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

```
int main(int argc, char **argv){
```

```
    int cmd;
```

```
    while((cmd = getopt(argc, argv, "ab")) > 0){ //옵션처리한다. 인자갯수, 인자문자열 처리할 수있는 옵션->ab
```

몇개인지 감지하려고 argc, argv 를 사용

```
        switch(cmd){ //걸리는 애가 cmd, 만약 a 면 cmd 가 a 가 된다
```

```
            case 'a':
```

```
                printf("a option\n");
```

```
                break;
```

```
            case 'b':
```

```
                printf("b option\n");
```

```
                break;
```

```
            default: //a 와 b 가 아니면
```

```
                printf("unknown option\n");
```

```
        }
```

```
    }
```

```
    return 0;
```

```
}
```

```
mhn@mhn-Z20NH-AS51B5U:~/linux/22$ ./a.out -a
a option
mhn@mhn-Z20NH-AS51B5U:~/linux/22$ ./a.out -b
b option
mhn@mhn-Z20NH-AS51B5U:~/linux/22$ ./a.out -ba
b option
a option
mhn@mhn-Z20NH-AS51B5U:~/linux/22$ ./a.out -ab
a option
b option
mhn@mhn-Z20NH-AS51B5U:~/linux/22$ vi test2.c
mhn@mhn-Z20NH-AS51B5U:~/linux/22$ ./a.out -abc
a option
b option
./a.out: invalid option -- 'c'
unknown option
mhn@mhn-Z20NH-AS51B5U:~/linux/22$
```

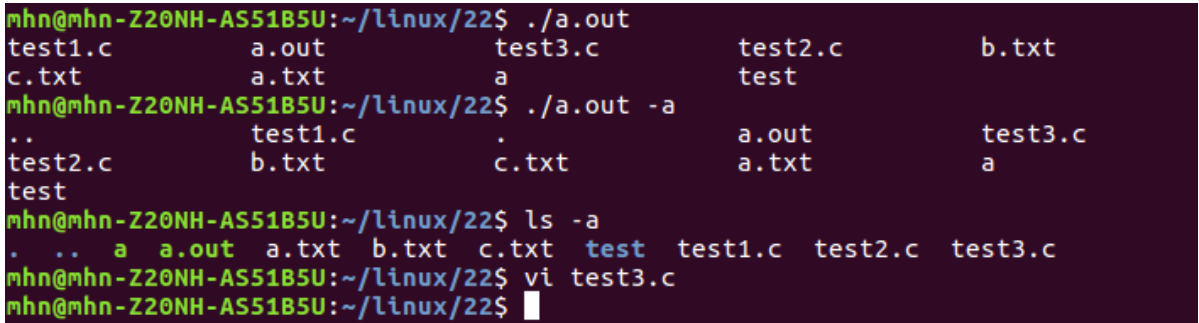
예제 3)

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

int main(int argc, char **argv){

    DIR *dp; //파일 구조체 포인터
    int i=0, cmd;
    struct dirent *p; //디렉토리 안에 있는 리스트
    cmd = getopt(argc, argv, "a");
    dp = opendir("."); //현재 디렉토리 포인터
    while(p = readdir(dp)){
        if(cmd != 'a'){ //a 없으면
            if(p->d_name[0] == '.')
                continue;
        }
        printf("%-16s", p->d_name); //디렉토리 안에 있는 이름을 출력한다
        if((i+1) % 5 == 0)
            printf("\n");
        i++;
    }printf("\n");
    closedir(dp);

    return 0;
}
```



```
mhn@mhn-Z20NH-AS51BSU:~/linux/22$ ./a.out
test1.c      a.out      test3.c      test2.c      b.txt
c.txt       a.txt      a            test
mhn@mhn-Z20NH-AS51BSU:~/linux/22$ ./a.out -a
..          test1.c      .            a.out      test3.c
test2.c     b.txt      c.txt       a.txt      a
test
mhn@mhn-Z20NH-AS51BSU:~/linux/22$ ls -a
.  ..  a  a.out  a.txt  b.txt  c.txt  test  test1.c  test2.c  test3.c
mhn@mhn-Z20NH-AS51BSU:~/linux/22$ vi test3.c
mhn@mhn-Z20NH-AS51BSU:~/linux/22$
```

flag

int 형 4 바이트이므로 32 개의 상태를 저장할 수 있다.

플래그로 사용할 변수에 |= 연산자와 숫자를 사용하여 특정 비트를 켜다.

예제 4)

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

int main(int argc,int **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; // 0 번째 비트 값을 저장했다 ... 0000 0001
                break;
            case 'b':
                flag |= 2; // 0010
                break;
            case 'R':
                flag |= 4; // 0100
                break;
            case 'i':
                flag |= 8; // 1000
                break;
        }
    }

    dp = opendir(".");
    while(p = readdir(dp)){
        if(!(flag & 1)){ //a 옵션이 있으면
            if(p->d_name[0] == '.')
                continue;
        }
        printf("%-16s",p->d_name); //a 옵션이 없으면
        if((i+1)%5==0)
            printf("\n");
        i++;
    }
    printf("\n");
    closedir(dp);
    return 0;
}
```

```

mhn@mhn-Z20NH-AS51B5U:~/linux/22$ ./a.out
test1.c      test4.c      a.out      test3.c      test2.c
b.txt        c.txt        a.txt      test5.c      a
test
mhn@mhn-Z20NH-AS51B5U:~/linux/22$ ./a.out -a
..           test1.c      test4.c      .           a.out
test3.c      test2.c      b.txt        c.txt       a.txt
test5.c      a           test
mhn@mhn-Z20NH-AS51B5U:~/linux/22$ █

```

stat()

파일의 상태를 알아올 수 있는 함수

헤더 : #include <sys/stat.h>

인자 : 디렉토리 이름, 버퍼 → 첫번째 인자로 주어진 이름의 상태를 얻어와서 두번째 인자인 버퍼에 저장한다.

함수가 성공적으로 수행될 경우 파일의 정보를 stat 구조체에 복사한다

stat 구조체)

```

struct stat {
    dev_t      st_dev;      /* device */
    ino_t      st_ino;      /* inode */
    mode_t     st_mode;     /* protection */
    nlink_t    st_nlink;    /* number of hard links */
    uid_t      st_uid;      /* user ID of owner */
    gid_t      st_gid;      /* group ID of owner */
    dev_t      st_rdev;     /* device type (if inode device) */
    off_t      st_size;     /* total size, in bytes */
    blksize_t  st_blksize;  /* blocksize for filesystem I/O */
    blkcnt_t   st_blocks;   /* number of blocks allocated */
    time_t     st_atime;    /* time of last access */
    time_t     st_mtime;    /* time of last modification */
    time_t     st_ctime;    /* time of last change */
};

```

예제 5)

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

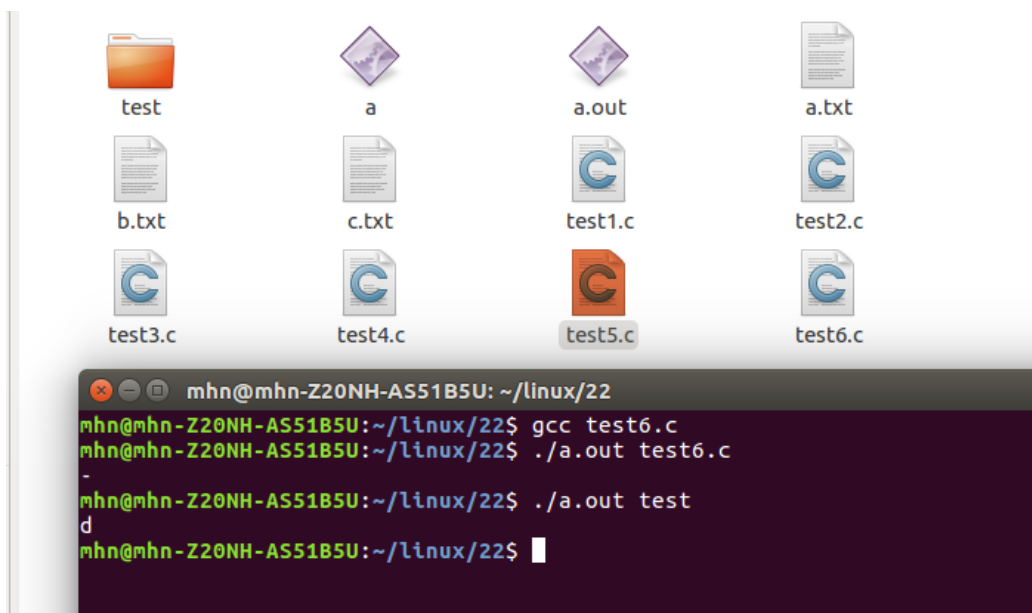
int main(int argc, char **argv){

    struct stat buf;
    char ch;
    stat(argv[1], &buf); //상태보기

    if(S_ISDIR(buf.st_mode)) //디렉토리
        ch = 'd';
    if(S_ISREG(buf.st_mode)) //파일
        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)) //캐릭터디바이스
        ch = 'c';
    if(S_ISBLK(buf.st_mode)) //블록디바이스
        ch = 'b';
    printf("%c\n", ch);

    return 0;

//두번째 인자로 받은 것이 무엇인지 판별
//파일인지 디렉토리인지 파이프인지 등등
}
```



예제 6)

```
#include <stdio.h>
#include <unistd.h>
#include <sys/stat.h>
#include <sys/types.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); //상태보기
    if(S_ISDIR(buf.st_mode)) //디렉토리
        perm[0] = 'd';
    if(S_ISREG(buf.st_mode)) //파일
        perm[0] = '-';
    if(S_ISFIFO(buf.st_mode)) //파이프
        perm[0] = 'p';
    if(S_ISLNK(buf.st_mode)) //바로가기파일
        perm[0] = 'l';
    if(S_ISSOCK(buf.st_mode)) //소켓
        perm[0] = 's';
    if(S_ISCHR(buf.st_mode)) //캐릭터디바이스
        perm[0] = 'c';
    if(S_ISBLK(buf.st_mode)) //블록디바이스
        perm[0] = 'b';

    for(i=0; i<9; i++)
        if((buf.st_mode >> (8-i)) & 1) // buf.st_mode 를 8 비트 민다.
            perm[i+1] = rwx[i%3]; // 첫번째 자리에 r 을 채운다

    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];
    }
}
```

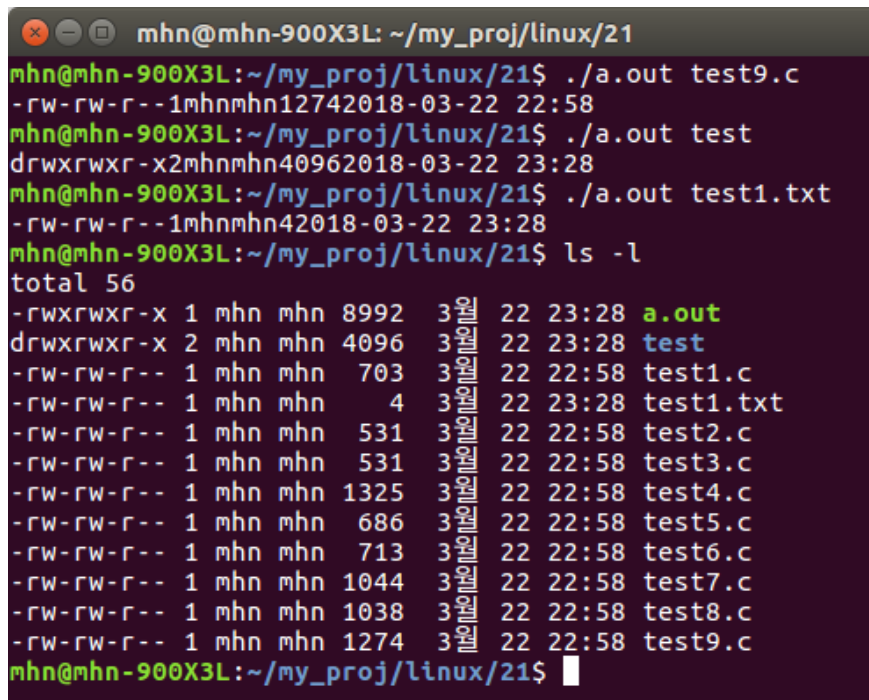


```

printf("%s",perm);
printf("%d", buf.st_nlink); //umber of hard links
pw = getpwuid(buf.st_uid); // user ID
printf("%s",pw->pw_name);
gr = getgrgid(buf.st_gid); //group ID
printf("%s",gr->gr_name);
printf("%d",buf.st_size); // total size, in bytes
tm = localtime(&buf.st_mtime); // time of last modification
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;
}

```



A terminal window titled 'mhn@mhn-900X3L: ~/my_proj/linux/21' showing the execution of a program. The program takes a filename as input and prints its permissions, owner, group, size, number of links, and modification time. The files listed are test9.c, test, test1.txt, test2.c, test3.c, test4.c, test5.c, test6.c, test7.c, test8.c, and test9.c. The output for test9.c is: -rw-rw-r-- 1 mhn mhn 1274 2018-03-22 22:58. The output for test is: drwxrwxr-x 2 mhn mhn 4096 2018-03-22 23:28. The output for test1.txt is: -rw-rw-r-- 1 mhn mhn 4 2018-03-22 23:28. The output for test2.c is: -rw-rw-r-- 1 mhn mhn 531 2018-03-22 22:58. The output for test3.c is: -rw-rw-r-- 1 mhn mhn 531 2018-03-22 22:58. The output for test4.c is: -rw-rw-r-- 1 mhn mhn 1325 2018-03-22 22:58. The output for test5.c is: -rw-rw-r-- 1 mhn mhn 686 2018-03-22 22:58. The output for test6.c is: -rw-rw-r-- 1 mhn mhn 713 2018-03-22 22:58. The output for test7.c is: -rw-rw-r-- 1 mhn mhn 1044 2018-03-22 22:58. The output for test8.c is: -rw-rw-r-- 1 mhn mhn 1038 2018-03-22 22:58. The output for test9.c is: -rw-rw-r-- 1 mhn mhn 1274 2018-03-22 22:58.

```

mhn@mhn-900X3L: ~/my_proj/linux/21
mhn@mhn-900X3L:~/my_proj/linux/21$ ./a.out test9.c
-rw-rw-r-- 1 mhn mhn 1274 2018-03-22 22:58
mhn@mhn-900X3L:~/my_proj/linux/21$ ./a.out test
drwxrwxr-x 2 mhn mhn 4096 2018-03-22 23:28
mhn@mhn-900X3L:~/my_proj/linux/21$ ./a.out test1.txt
-rw-rw-r-- 1 mhn mhn 4 2018-03-22 23:28
mhn@mhn-900X3L:~/my_proj/linux/21$ ls -l
total 56
-rwxrwxr-x 1 mhn mhn 8992 3 22 23:28 a.out
drwxrwxr-x 2 mhn mhn 4096 3 22 23:28 test
-rw-rw-r-- 1 mhn mhn 703 3 22 22:58 test1.c
-rw-rw-r-- 1 mhn mhn 4 3 22 23:28 test1.txt
-rw-rw-r-- 1 mhn mhn 531 3 22 22:58 test2.c
-rw-rw-r-- 1 mhn mhn 531 3 22 22:58 test3.c
-rw-rw-r-- 1 mhn mhn 1325 3 22 22:58 test4.c
-rw-rw-r-- 1 mhn mhn 686 3 22 22:58 test5.c
-rw-rw-r-- 1 mhn mhn 713 3 22 22:58 test6.c
-rw-rw-r-- 1 mhn mhn 1044 3 22 22:58 test7.c
-rw-rw-r-- 1 mhn mhn 1038 3 22 22:58 test8.c
-rw-rw-r-- 1 mhn mhn 1274 3 22 22:58 test9.c
mhn@mhn-900X3L:~/my_proj/linux/21$

```

```
for(i=0;i<9;i++)
    if((buf.st_mode >> (8-i)) &1)
        perm[i+1] = rwx[i%3];
```

struct stat(16bit)

				s	s	t	r	w	x	r	w	x	r	w	x
--	--	--	--	---	---	---	---	---	---	---	---	---	---	---	---

-----파일종류-----

perm[11]

파일종류	r	w	x	r	w	x	r	w	x
------	---	---	---	---	---	---	---	---	---

perm[0] perm[1]

처음 0 이 들어가고 struct stat 을 8 비트만큼 민다

다음 perm[1]자리에 rwx[0]값인 r 을 배치한다.

이런 방법으로 9 번 반복한다

```
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];
}
```

struct stat(16bit)

				s	s	t	r	w	x	r	w	x	r	w	x
--	--	--	--	---	---	---	---	---	---	---	---	---	---	---	---

-----파일종류-----

perm[11]

파일종류	r	w	S / s	r	w	S / s	r	w	T/t
------	---	---	-------	---	---	-------	---	---	-----

perm[0] perm[1]

처음 0 이 들어가고 struct stat 을 11 비트만큼 민다

만약에 perm[3] 즉, x 자리가 비어있다면 그 자리에 sst[0]값인 s 가 대문자로 들어간다

비어있지 않다면 소문자로 들어갈 것이다.

다음 1 이 들어가면 struct stat 을 10 비트만큼 민다

만약 perm[6] 즉, x 자리가 비어있다면 그 자리에 sst[1]값인 s 가 대문자로 들어간다
비어있지 않다면 소문자로 들어갈 것이다.

마지막으로 2 가 들어가면 struct stat 을 9 비트만큼 민다

만약 perm[9] 즉, x 자리가 비어있다면 그 자리에 sst[2]값인 t 가 대문자로 들어간다
비어있지 않다면 소문자로 들어갈 것이다.

가상 메모리에서 물리 메모리로 접근하는 것을 페이징이라 한다.

예를 들어서 32bit 시스템 가상메모리에 0xbf884c8c 라고 하는 주소가 있다고 가정하자
이 주소를 10bit 10bit 12bit 로 쪼개어 찾아가면 물리 메모리까지 접근이 가능하다
위에 주소로 생각해보면 766 / 132 / 3212 로 쪼개지는데 여기에는 페이지 정보가 담겨있다.
이것을 페이지 테이블이라 한다.

mm_struct 구조체안에 가상의 배열에서 766 을 찾고 또 다음 배열에서 132 번을 찾으면 물리 메모리
에 접근할 수 있는 진짜 주소를 알 수 있다.

만약 실행한 파일이 용량이 너무 커서 물리 메모리가 부족하다면 swap 메커니즘을 활용할 수 있다.
물리 메모리가 부족할 때 자주쓰진 않지만 필요한 정보들을 디스크에 queue 형태로 저장한다. 그리고
이것을 필요할 때 빼서 쓰면 메모리 부족 문제를 해결할 수 있다.

