TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – GJ (박현우)
uc820@naver.com

1. 시스템 프로그래밍 – 4 파일입출력

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

int main(void){

    DIR *dp;
    int i = 0;
    struct dirent *p;
    dp = opendir(".");
    while(p = readdir(dp)){

        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;
}
```

opendir함수로 현 위치의 디렉토리를 모두 연다.

Readdir함수로 모든 디렉토리를 읽어

디렉토리 0번 이름이 '.' 이면 패스하고

아니면 이름을 출력한다.

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

#define A (1 << 0)
#define B (1 << 1)
#define C (1 << 2)

void check_flag(int flag){

    switch(flag | A | B | C){

        case 1:
            printf("A\n");
            break;
        case 2:
            printf("B\n");
            break;
        case 4:
            printf("C\n");
            break;

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

    int cmd;
    int flag;

    while( (cmd = getopt(argc, argv, "abcdefg")) > 0){

        switch(cmd){

            case 'a' :
                flag |= 1 << 0;   // flag를  0000001로 만든다.
                printf("a option\n");
                break;
            case 'b' :
                flag |= 1 << 1;   // flag를  0000011로 만든다.
                printf("a option\n");
                break;
            case 'c' :
                flag |= 1 << 2;   // flag를  0000111로 만든다.
                printf("a option\n");
                break;
            default:
                printf("unknown option\n");
                break;

        }
    }

    check_flag(flag);  //  입력된 플래그를 확인 후 해당하는 것을 출력
```

1. 시스템 프로그래밍 – 4 파일입출력

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

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

    DIR *dp;
    int i = 0, cmd;
    struct dirent *p;
    cmd = getopt(argc, argv, "a");  // getopt로 옵션 설정
    dp = opendir(".");
    while(p = readdir(dp)){

        if(cmd != 'a'){  //  cmd가 a이면 전체 출력 아니면 .은 제외하고 출력

            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;
}
```

```c
#include<stdio.h>
#include<sys/types.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 'b':
                flag |= 2;
                break;
            case 'R':
                flag |= 4;
                break;
            case 'i':
                flag |= 8;
                break;

        }

    }

    dp = opendir(".");
    while(p = readdir(dp)){

        if(!(flag & 1)){  // 플래그의 첫번째 비트가 1이면 즉 a이면 모두 출력

            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;
}
```

1. 시스템 프로그래밍 – 4 파일입출력

```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;
    stat(argv[1], &buf);
    if(S_ISDIR(buf.st_mode)) // 디렉토리
        ch = 'd';
    if(S_ISREG(buf.st_mode)) // 레귤러 일반 파일
        ch = '-';
    if(S_ISFIFO(buf.st_mode))  // 파이프
        ch = 'p';
    if(S_ISLNK(buf.st_mode))  // 링크 바로가기
        ch = 'l';
    if(S_ISSOCK(buf.st_mode)) // 소켓
        ch = 's';
    if(S_ISCHR(buf.st_mode)) //케릭터 디바이스
        ch = 'c';
    if(S_ISBLK(buf.st_mode)) // 블락 디바이스
        ch = 'b';

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

    return 0;
}
```

```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)) // 디렉토리
        perm[0] = 'd';
    if(S_ISREG(buf.st_mode)) // 레귤러 일반 파일
        perm[0] = '-';
    if(S_ISFIFO(buf.st_mode))  // 파이프
        perm[0] = 'p';
    if(S_ISLNK(buf.st_mode))  // 링크 바로가기
        perm[0] = 'l';
    if(S_ISSOCK(buf.st_mode)) // 소켓
        perm[0] = 's';
    if(S_ISCHR(buf.st_mode)) //케릭터 디바이스
        perm[0] = 'c';
    if(S_ISBLK(buf.st_mode)) // 블락 디바이스
        perm[0] = 'b';

    for(i = 0; i<9; i++){

        if( (buf.st_mode >> ( 8 - i) & 1))  // buf.st_mode의 8번째 비트부터
            perm[i + 1] = rwx[i%3];      //1비트씩 우측 쉬프트로 검사해서
                                         //조건에 맞으면
                                         // perm[1] 부터 rwx차례로 삽입
    }
    printf("%s\n",perm);


    return 0;
}
```

1. 시스템 프로그래밍 – 4 파일입출력

```c
#include<sys/types.h>
#include<sys/stat.h>
#include<unistd.h>
#include<stdio.h>
#include<grp.h>
#include<pwd.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)) // 디렉토리
        perm[0] = 'd';
    if(S_ISREG(buf.st_mode)) // 레귤러 일반 파일
        perm[0]  = '-';
    if(S_ISFIFO(buf.st_mode))  // 파이프
        perm[0]  = 'p';
    if(S_ISLNK(buf.st_mode))  // 링크 바로가기
        perm[0]  = 'l';
    if(S_ISSOCK(buf.st_mode)) // 소켓
        perm[0]  = 's';
    if(S_ISCHR(buf.st_mode)) //케릭터 디바이스
        perm[0]  = 'c';
    if(S_ISBLK(buf.st_mode)) // 블락 디바이스
        perm[0]  = 'b';

    for(i = 0; i<9; i++){

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

    }
    printf("%s\n",perm);
    printf("%d ", buf.st_nlink); // 파일 갯수

    pw = getpwuid(buf.st_uid);  //user id
    printf("%s ", pw->pw_name);
    gr = getgrgid(buf.st_gid); // group id
    printf("%s", gr->gr_name);

    return 0;
}
```

```c
for(i = 0; i<3; i++){

    if(buf.st_mode >> (11 - i) & 1){  // 11번째 비트부터 1비트 씩 검사 후
        if(perm[ (i+1) * 3 == '-'])   // 파일이면 perm[3] 부터
            perm[ (i + 1) *3 ] = sst[i] ^ 0x20;  // sst[0]의 대문자 삽입
        else
            perm[ (i + 1) * 3] = sst[i];   // 아니면 소문자 삽입
    }
}

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);  // 시간
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");
```

- - 000 000 000
  파일의 종류 root 그룹 제3자

  4 2 1
- r w x
- x = execute (실행 권한)

- 0 6 4 4 (팔진수 표현할 때 0을 표현)

- d --- --t --- 이면 공유 파일
- - --- --- --t 이면 높은 용량을 처리하기 위한 옵션
- Sticky bit 소문자는 권한 있음, 대문자는 권한 없음

1. 시스템 프로그래밍 – 4 페이징