TI DSP, MCU 및
Xilinx Zynq FPGA
프로그래밍 전문가 과정

강사 : Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 : 황수정
sue100012@naver.com
21일차 (2018. 03. 22)

학습 내용 복습

* touch a.txt b.txt c.txt d.txt : a.txt b.txt c.txt d.txt 가 생성
* opendir( ) : ( )안의 디렉토리를 열고 그 디렉토리에 대한 포인터를 반환한다.
* readdir( ) : 디렉토리를 읽는다.

학습예제

```
#include <sys/types.h>
#include <dirent.h>                // 밑에 선언한 'struct dirent *p;' 를 쓰기 위해서 필요한 헤더파일
#include <stdio.h>

int main(void){
   DIR *dp;          //파일 포인터랑 비슷한 것!
   int i = 0;
   struct dirent *p;        //디렉토리 내에 있는 것을 선언한 것이다. 디렉토리 내부 파일들의 정보 보관
                            구조체이다.
   dp = opendir(".");     //디렉토리도 오픈해야 쓸 수 있다.
   while(p= readdir(dp))          //디렉토리를 읽는다. 즉, 디렉토리 리스트를 받아오는 것이다.
   {
      if(p->d_name[0] == '.')      //배열의 0번은 맨 앞 글자 .이면 패스
     continue;
      printf("%-16s", p->d_name);
      if((i + 1) % 5 ==0)     //다섯번 넘으면 개행하라
    printf("₩n");
      i++;
   }    //더 이상 돌 리스트 없으면 while이 끝남
   printf("₩n");
   closedir(dp);
   return 0;
}
```

> 결과는 터미널 창에서 ls 명령어를 넣었을 때와 동일하다. 리스트를 보여준다.

```
 a.out          2.c          c.txt          3.c          1.c          b.txt          a.txt          0_1.c
d.txt          00.txt
```

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

int main (int argc, char **argv)
{
   int cmd;
   while((cmd = getopt(argc, argv, "ab")) >0) //argc 인자 갯수 argv 문자열의 주소 ab는 처리할 수 있는
                                                옵션. 현재 옵션이 몇개인지 알기 위해
```

```
      {        switch(cmd)
       {      case 'a':
        printf(" a option₩n");
        break;
     case 'b':
        printf("b option₩n");
        break;
     default:
        printf("unknown option₩n");
        }
   }
   return 0;
}
```

> 결과는 실행을 어떻게 시키느냐에 따라 다른다.

./a.out -a 면 a option이 출력된다.

./a.out –ba 면 b option a option 이 출력되고

./a.out -abc 면 a option b option ./a.out: invalid option -- 'c' unkown option 이 출력된다.

c에서 unkown option만 출력 될 줄 알았는데, ./a.out: invalid option -- 'c'가 먼저 출력 된다.

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

int main (int argc, char **argv)
{
   DIR *dp;
   int i = 0, cmd;
   struct dirent *p;
   cmd = getopt(argc, argv, "a");
   dp = opendir(".");
   while (p = readdir(dp))   {
     if(cmd != 'a')                    //a가 살아있지 않으면, .을 패스
      {         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;
}
```
> 결과는 a가 없으면 ls와 같고 a를 넣어주면 '.' , '..'과 같이 숨김 파일도 보여준다.

* 옵션들을 편하게 관리하는 기법
 어떻게 하면 될까? 옵션이 많다는 것은 메모리 공간을 많이 차지한다는 소리이다.
그렇다면, 공간 효율성, 속도도 증가시키기 위한 좋은 방법은 무엇일까?
그 동안 배운 방법 중에 동적할당이 있는데 동적할당은 오히려 공간을 많이 먹는다. 이 방법은 아니다.
바로 **비트 연산**이다! int형에 32비트 사용 가능 인트형 하나로 32개 옵션을 쓸 수 있기 때문이다*/

 비트 연산을 이용하여 옵션 주기(선생님 예제)

```
#include <stdio.h>
 #include <unistd.h>
 #define A       (1 << 0)
 #define B       (1 << 1)
 #define C       (1 << 2)
 #define D       (1 << 3)
 #define E       (1 << 4)
 #define F       (1 << 5)
 #define G       (1 << 6)

 void check_flag(int flag) {
         int i, tmp = flag;

        for(i = 0; i < 7; i++) {
                switch(tmp & (1 << i))  //flag 에 들어온 인자 값에 따라 비트 수가 달라진다.
  //case 와 flag 가 같은 비트 수 에서 case 밑의 명령이 실행.

                {
                        case 1:
                                printf("A₩n");
                                break;
                        case 2:
                                printf("B₩n");
                                break;
                        case 4:
                                printf("C₩n");
                                break;
                       case 8:
                                printf("D₩n");
                                break;
```

```
                case 16:
                        printf("E₩n");
                        break;
                case 32:
                        printf("F₩n");
                        break;
                case 64:
                        printf("G₩n");
                        break;                  }
        } }
int main(int argc, char **argv) {
        int cmd;
        int flag;

       /* 7 개의 옵션: a, b, c, d, e, f, g */
        while((cmd = getopt(argc, argv, "abcdefg")) > 0)
 // 인자로 받은 것 중 a, b, c, d, e, f, g 만 옵션으로만 받을 것이다.
         {                 switch(cmd) {
                        case 'a':
                                flag |= 1 << 0;  //1
 // 1을 쉬프트 연산을 하고 flag 와 OR 연산을 한 뒤 flag 에 값을 넣는다.
                               printf("a option₩n");
                                break;


                case 'b':
                        flag |= 1 << 1; // 10
                        printf("b option₩n");
                        break;
                case 'c':
                        flag |= 1 << 2; //100
                        printf("c option₩n");
                        break;
                case 'd':
                        flag |= 1 << 3; //1000
                        printf("d option₩n");
                        break;
                case 'e':
                        flag |= 1 << 4; //10000
                        printf("e option₩n");
                        break;
                case 'f':
                        flag |= 1 << 5; //100000
```

```
                printf("f option₩n");
                break;
        case 'g':
                flag |= 1 << 6; //100000
                printf("g option₩n");
                break;
        default:
                printf("unknown option₩n");                }
}

check_flag(flag);
 return 0; }
```

시스템 프로그래밍 - 가상메모리