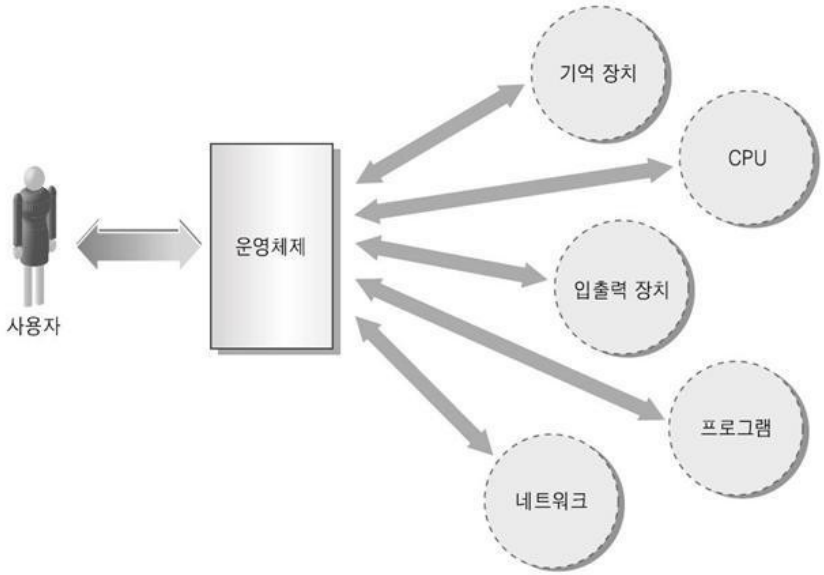
TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

학생 – 이우석
colre99@naver.com

[3/19(월) – 18 일차]

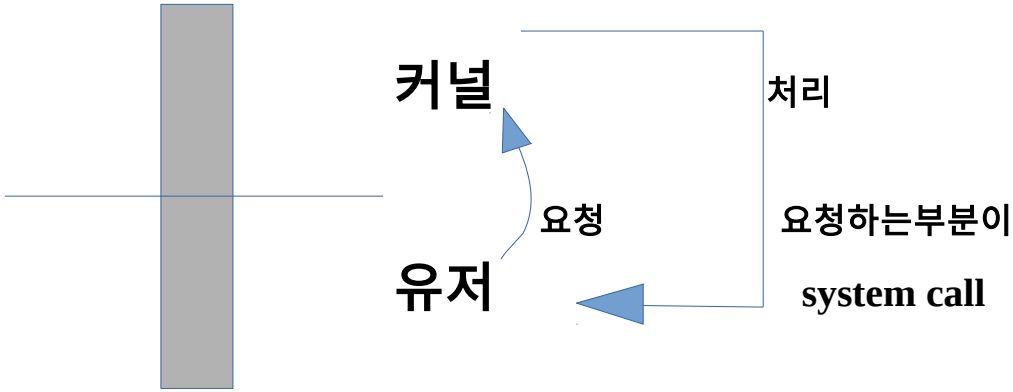
[리눅스 시스템 프로그래밍]

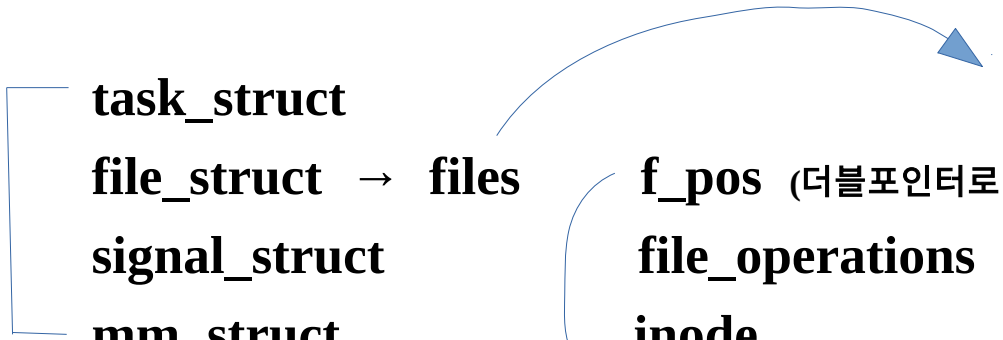


시스템 프로그래밍을 하는 이유는 하드웨어와 코드 사이의 긴밀한 관계를 이해하고, 좀더 최적화 시켜주기 위함 이다. 또한, 코딩작업시 간혹 버그가 일어나는 것을 파악하고 찾아내기 위함이다.

표준 입력 : 0 (1 번)
표준 출력 : 1 (2 번)
표준 에러 : 2 (3 번)

System Call (SW interrupt)
Device Driver = Firmware





files 에 대한 fd 주소값 (files *fd[ ])

배열의 인덱스

파일포인터의 인덱스 호출

=> 은닉된 파일 읽기요청

close 도 system call.

하드웨어 적인 계산이 들어가면

다 system call

[코딩작업 도중 화면분할 하는방법]

코딩작업 도중 현 화면과 함께 보이는 동시에 다른 파일을 불러와 보고싶거나, 작성하고 싶다면 ‘편집모드’에서 ‘: vs’ or ‘:sp’ 를 입력하면 화면이 분할된다. 여기서 ‘:vs’ 와 ‘:sp’의 차이점은 vs 는 세로(열)로 화면이 분할되고, sp 는 가로(행)로 분할된다. 분할하면 본 화면과 같게 분할이 되는데 분할된 화면에서 다른파일을 불러오고 싶거나 새로 작성하고 싶다면 ‘편집모드’

에서 ‘:e 파일명’ 하면 된다. 분할된 화면을 끄고 싶다면 원래방식과 마찬가지로 ‘:q’로 입력하면 종료. 여기에서 화면분할 하며 다른파일을 만들어 주거나 새로 만들었다면, 컴파일시에 함께 해줘야 한다. ex) gcc 원본파일 다른파일
덧붙여, 화면 분할시 나뉘어진, 분할된 다음 화면으로 넘어가려면 Ctrl + W W 입력.

[헤더파일 구별]
#include < > = 시스템 헤더파일
#include “ ” = 사용자 정의 헤더파일

[커널소스 받기]

kernel 디렉토리를 만들어준 후에, 소스를 받기위해 아래주소를 설정하고, 압축풀기.
1. wget https://mirrors.edge.kernel.org/pub/linux/kernel/v4.x/linux-4.4.tar.gz
2. tar zxvf linux-4.4.tar.gz

[c 코딩시에 찾고자하는 문자에 하이라이트 주기방법]
편집 모드에서 ‘:set hlsearch’ 입력하고 ‘/찾고자하는 문자’.

ex) /task_struct 해주면, 해당 문자들이 하이라이트로 표시되며 찾을수 있다.

* 헤더파일을 많이 선언하면, 오류가 나기도 하고 성능이 저하된다.
* c 코딩시 통함수, 통 파일 하면 cache 가 깨지고, 속도도 느려지고, 간혹 컴파일안됨.

[오늘 학습한 내용]
1 번:

```c
#include <stdlib.h>
#include <fcntl.h>
#include <stdio.h>
#include <unistd.h>
#define ERROR   -1

int main(void)
{

    int filedes;
    char pathname[] = "temp.txt";           //filedes 는 숫자리턴.
    if((filedes = open(pathname, O_CREAT | O_RDWR | O_EXCL, 0644)) == ERROR)
    {                                 //생성    읽고,쓰고   //0644는 권한번호.
       printf("File Open Error!\n");
       exit(1); //강제종료
    }
                                      //filedes는 숫자를 가진다.
    close(filedes);

    printf("fd = %d\n", filedes);
    // 출력값 3이 나오는데 파일 3개 만들어짐을 의미
    return 0;
}
```

결과값:

```
wooseok91@air:~/linux$ ./a.out
File Open Error!
```

2 번:

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

int main(void)
{

    int filedes1, filedes2;

    filedes1 = open("data1.txt", O_WRONLY | O_CREAT | O_TRUNC, 0644);
    filedes2 = creat("data2.txt", 0644); // TRUNC는 무조건 한번 파일을 민다
                //  즉, 새로 갱신한다는말.=처음) 임시데이터 저장할때.
    close(filedes1);
    close(filedes2);

    return 0;
}
```

결과값: (결과값을 보면 data1.txt 와 data2.txt 가 생성된걸 확인할 수 있다.)

```
wooseok91@air:~/linux$ ls
a.out     cat1.c      cptest.c   data2.txt  li3.c       my_scanf.h  temp.txt
cat1-1.c  cat.c       data1.c    li1.c      li4.c       temp1.txt   wc.c
cat1-2.c  cat_copy.c  data1.txt  li2.c      my_scanf.c  temp2.txt
```

3 번:

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

int main(void)                  //fd = file descriptor? 파일 식별자.
{

   int fdin, fdout;
   ssize_t nread;
   char buf[1024];  //읽고 1024을 집어 넣겠다!! 위치어디로 가는지.

   fdin = open("temp1.txt", O_RDONLY);  //파일안에 아무것도 없으면 안열림.
   fdout = open("temp2.txt", O_WRONLY | O_CREAT | O_TRUNC, 0644);

   while((nread = read(fdin, buf, 1024)) > 0) //fdin은 3을 받음(empty)
   {    //=number read     //buf = 옮기는 역할(임시역할)
      if(write(fdout, buf, nread) < nread)  //fdout은 4를 받음.
      {
         close(fdin);                    //read(fd, buf, 읽을크기) = byte
         close(fdout);                   //write(fd, buf, 쓸 크기)
      }
   }          //파일읽고 buf에 넣는다.
                                  /*읽는크기가 1024 넘어가면 1024
                                   크기만큼만 복사되고 나머진 짤림*/
   close(fdin);
   close(fdout);

   return 0;
}
```

염 사진처럼 작성시,
temp1.txt 와
temp2.txt 가 생김.

temp1 에 내용적고,
./a.out 실행해주면,
temp1 의 내용은
다시 리셋이 된다.

temp2 는 그대로
복사되며 리셋.

결과값:

```
wooseok91@air:~/linux$ ls
a.out      cat1.c      cptest.c   data2.txt  li3.c       my_scanf.h  temp.txt
cat1-1.c   cat.c       data1.c    li1.c      li4.c       temp1.txt   wc.c
cat1-2.c   cat_copy.c  data1.txt  li2.c      my_scanf.c  temp2.txt
```

4 번:

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

int main(void)
{
    FILE *fp = fopen("mycat.c", "r"); //open으로 치면 r 은 read이다
    char buf[1024] = "\0";          //속도면에서 system call이 더 빠름.
    int ret;

    while(ret = fread(buf, 1, sizeof(buf), fp))
    {
       usleep(1000000); //micro second = usleep = 10^-6 = 1초
       fwrite(buf, 1, ret, stdout);  // 표준출력(모니터)
             // 1바이트씩 읽어서 buf를
    }
    fclose(fp);
    return 0;

}
```

결과값:

```
wooseok91@air:~/linux$ ./a.out
Segmentation fault (core dumped)
```

5 번:

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


int main(int argc, char **argv)
{     //적을 개수 , 문자열 배열.[0],[1],[2]
    int fd, ret;
    char buf[1024];
    if(argc != 2) //2개를 써주는건데 원했던게 아닐경우  if문 실행됨.
    {
        printf("Usage: mycat filename\n");
        exit(-1); // 강제종료
    }
                                  // ./a.out실행후 두번째 파일을 열여서 읽음.
    fd = open(argv[1], O_RDONLY);
              //argv 들어가서 배열 1번 을 읽는다.
    while(ret = read(fd, buf, sizeof(buf)))
    {
        write(1, buf, ret);
    }     //1번은 표준출력.

    close(fd);

    return 0;
}
```

결과값:

```
wooseok91@air:~/linux$ ./a.out
Usage: mycat filename
```

6 번:

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

        //적을 개수를 숫자로 나타냄.
int main(int argc, char **argv)
{                   //문자열 배열.[0],[1],[2].
   int fd, ret;
   char buf[1024];


         //0은 표준입력
   ret = read(0, buf, sizeof(buf));  //size는 buf[1024] 만큼 지정.
      write(1, buf, ret);    //size는 ret에 지정된 만큼.
         //1은 표준출력.


   close(fd);

   return 0;
}
```

```
wooseok91@air:~/linux$ ./a.out
hello again
hello again
◆◆◆◆[][][] ◆[][][]']◆◆P(][]U[][][][]◆◆[][][](][]◆◆X[][][][]◆◆[][][])[][][][][][][][]P[][][][] ◆[][][]◆◆[][][]K&◆[][][]◆◆[][][]P[][][][](][]◆
◆(][]◆◆[]◆[][][]O◆◆[][]0[][]◆◆(][]◆◆.N=◆8◆◆[][][][])[][]◆◆◆[][][] ◆[][][](][]◆◆)[]U[][][][][][][][][][][][]◆◆[][][]*][]◆PU[][][][]◆[
[][]*][][]◆[][][]PU[][][][][]◆[][][]◆be◆◆◆[][][][]◆◆)[][][]◆◆)[][][])[][].N=◆◆[][]◆◆◆◆◆◆◆◆[][][][]◆[][][]P[][][][]*][]◆◆◆[][][]][
◆`+][]◆◆[][][]{+][][]◆[][][]◆+][]◆8[][]0@ ,][]◆◆z◆[][][][][]◆◆[][][]+][]◆#◆◆[][][]/////////////////◆ttime__vdso_get
```

위에 쓰레기 값은 15 행 size 를 buf[1024]로 지정했을경우 나타난다.

```
wooseok91@air:~/linux$ ./a.out
hello again
hello again
```

= 왼쪽 결과값은 15 행 사이즈를 ret 으로 지정한 경우.

7 번:

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


int main(int agrc, char argv)
{

    int fb;
    char buf[2][1024];
          //1024 공간을 2행으로 배열을 나타냄.

            //표준입력을 하고 buf에 저장되며 사이즈는 배열에 지정된 만큼.
    fb = read(0, buf, sizeof(buf));
             write(1, buf, fb);
              //표준 입력을 하고  사이즈는 fb에 저장된 만큼.
return 0;

}
```

결과값:

```
wooseok91@air:~/linux$ ./a.out
hellooooooooooo
hellooooooooooo
```

첫째줄에 사용자가 작성한 내용이
둘째줄에 복사되어 출력되었다.

8 번:

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

int main(void)
{

    int nr;
    char buf[1024] = {0};

                // 사이즈는buf[1024] 배열에 지정된만큼. 0으로 초기화.
    nr = my_scanf(buf, sizeof(buf));
    printf("nr = %d\n", nr);
    write(1, buf, nr);
     //표준입력 1을 해주고 사이즈는 nr에 지정된만큼.
return 0;

}
```

결과값: (오류가 있었는지 결과값 이상함. 추후 수정예정.)

```
wooseok91@air:~/linux$ gcc cat_copy.c
/tmp/cc4jwzgX.o: In function `main':
cat_copy.c:(.text+0x41): undefined reference to `my_scanf'
collect2: error: ld returned 1 exit status
```

9 번:

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


int main(int argc, char **argv)
{     //문장의 개수, 문장자체를 저장. 배열순으로 0, 1, 2 = 3문장
    int i;
    int in, out;       // ./a.out 실행동시에 './a.out' 도 0번째에 포함.
    char buf[1024] = {0};
    int nread;

    if(argc != 3)
    {
       printf("인자 입력 3개 하라고!\n");
       exit(-1); // 이건 강제종료 라는 뜻.
    }

    for(i = 0; i < argc; i++)  //문장 3개 되었을때 실행되며.
    {
       printf("당신이 입력한 인자는 =  %s\n", argv[i]);
    }

    in = open(argv[1], O_RDONLY); //argv의 배열공간은 두번째꺼.
    out = open(argv[2], O_CREAT | O_WRONLY | O_TRUNC); //배열공간 3번째꺼.

    while((nread = read(in, buf, 1204)) > 0)
    {        //read가 함수명령어고 'in'에 들어가서 글자를 읽고 buf에 넣어줌.
       if(write(out, buf, nread) < nread)
       {
          close(in);
          close(out);
          printf("비정상 종료\n");
          exit(-1);
       }
    }

    close(in);
    close(out);
    printf("정상 종료\n");

    return 0;
}
```

결과값:

```
wooseok91@air:~/linux$ ./a.out
인자 입력 3개 하라고!
```

10 번:

```c
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>

int main(int argc, char **argv)
{
   int fd = open(argv[1], O_RDONLY); // 인자에 파일이 와야해서.
   int line = 0;
   int word = 0;
   int flag = 0;
   int cnt = 0;
   char ch;

   if(argc != 2) // 사용법 처리구간.
   {
      printf("You need 1 more parameter\n");
      printf("Usage: mywc filename\n");
      exit(-1);
   }

   if((fd = open(argv[1], O_RDONLY)) < 0)  // 0은 표준입력, 1은 표준출력,  2는 표준>
에러.
   {
      perror("open()");              // 0644 이런건 권한 번호.
      exit(-1);
   }

   while(read(fd, &ch,1))
   {
      cnt++; // 문자가 몇개 사용 됐는지 알아보기위해.

      if(ch == '\n')   // 개행
         line++;

      if(ch != '\n' && ch != '\t' && ch != ' ')
               // 개행(\n) , 탭(\t) , 공백(' ') 이게 다 아니라면. -> = 숫자거나 문자
.
      {


         if(flag == 0)     // 다시 while로 간다.
         {
            word++;  // 단어개수 증가
            flag = 1;    //flag = 1이라고 지정.
         }
      }

      else
      {
         flag = 0;
      }
   }

   close(fd);
   printf("%d %d %d %s\n", line, word, cnt, argv[1]);
   return 0;

```