TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

학생 – 이우석
colre99@naver.com

[3/21(수) – 20 일차]

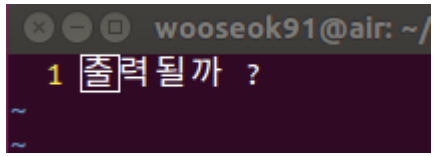
1 번.

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

int main(void)
{
    int fd;             //쓰기전용
    fd = open("a.txt", O_WRONLY | O_CREAT | O_TRUNC, 0644);
    close(1); //출력                          3번
    dup(fd);   //duplicate fd 가 1번의 역할.
    printf("출력될까 ?\n");
    return 0;
}
```

main 함수안에 변수 fd 를 보면 open 으로 하며 쓰기전용으로 a.txt 를 열어주며 생성과 초기화 하는 조건을 볼수있다.

결과값: (a.txt 에 다음과 같이 작성된걸 확인)



2 번.

```
 1 #include <stdio.h>
 2 #include <unistd.h>
 3 #include <fcntl.h>
 4
 5 int main(void)
 6 {
 7    int fd;
 8    char buff[1024];
 9    fd = open("a.txt", O_RDONLY);
10    close(0);  // 0 은 입력. 입력 닫힘.
    dup는
11    dup(fd); // 0번을 a.txt가  대체. gets 를 받음.
    최근에 닫힌걸 무조건 대체.
12    gets(buff); // 입력을 받는 함수. gets는 씹힘.
13    printf("출력될까 ?\n");
14    printf("%s", buff);
15    return 0;
16    }
17
18 // close 0,1,2 했으면 다시열기안된다.
19
20 //close 가 없으면 파일실행 안됨.
21 //꺽쇠를 프로그램 버젼으로 하면 이거.
~
                                1,1           All
```

결과값: (fd 변수에서 출력한걸 복사되어 똑같이 출력된걸 확인할 수 있다)

```
wooseok91@air:~/my_proj/Homework/
출력될까 ?
출력될까 ?wooseok91@air:~/my_proj
```

3 번.

```
 1 #include <sys/types.h>
 2 #include <unistd.h>
 3 #include <fcntl.h>
 4
 5 int main(int argc, char *argv[])
 6 {
 7     int i;
 8     char ch = 'a';      //쓰기전용
 9     int fd = open(argv[1], O_WRONLY | O_CREAT | O_TRUNC, 0644);
10     lseek(fd, 512 - 1, SEEK_SET); //512 번째로 강제시작.. <구성> > 파일이름, 파일위치, 사이즈
11     write(fd, &ch, 1);
12     close(fd);
13
14     return 0;
15 }
16              //lseek의 쓰는방법
~
~
~
~
~
~
~
                                                    1,1           All
```

4 번.

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

int main(void)
{
   int fd, ret;
   char buf[1024];
   mkfifo("myfifo"); // 함수 명령.
   fd = open("myfifo", O_RDWR);  //읽고쓰고를 리턴값주고.
   for(;;)
   {
      ret = read(0, buf, sizeof(buf));
      buf[ret -1] = 0; //작업이 완료 될때까지 기다려야 한다 = 블로킹.
      printf("keyboard Input : [%s]\n", buf);
      read(fd, buf, sizeof(buf)); //입력을 해줘야 블로킹 풀림.
      buf[ret - 1] = 0;
      printf("Pipe Input : [%s]\n", buf);
   }

   return 0;
}          //블로킹은 반드시 순서대로 해야할때 사용
           // 언블로킹은 다수가 동시에 사용할때 사용. ex) 사용예:  >
통신,네트워크

//instruction scheduling = compiler가 알아서 배치함.
~
                                              1,1           All
```

결과값:

```
wooseok91@air:~/my_proj/Homework/sanghoonlee/quiz$ ./a.out
one
keyboard Input : [one]
Pipe Input : [two]
three
keyboard Input : [three]
Pipe Input : [four
]

wooseok91@air:~/my_proj/Homework/sanghoonlee/quiz$ cat > myfifo
two
four
```

기존 터미널과 새로 연 터미널과의 주고받는 입력들을 확인할 수 있다.

5 번.

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

int main(void)
{
   int fd, ret;
   char buf[1024];
   fd = open("myfifo", O_RDWR);
   fcntl(0, F_SETFL, O_NONBLOCK);   //fctl 이 두줄 (논블록)로 입력없으면 다음으로 넘기기에.
   fcntl(fd, F_SETFL, O_NONBLOCK);
   for(;;)
   {
      if((ret = read(0, buf, sizeof(buf))) > 0)
      {
         buf[ret - 1] = 0;
         printf("keyboad Input : [%s\n]", buf);
      }
      if((ret = read(fd, buf, sizeof(buf))) > 0)
      {
         buf[ret - 1] = 0;
         printf("Pipe Input : [%s]\n", buf);
      }
   }
   close(fd);

   return 0;
}
```

결과값:

```
wooseok91@air:~/my_proj/Homework/sanghoonlee/quiz$ ./a.out
Pipe Input : [one]
Pipe Input : [two]
Pipe Input : [three]
Pipe Input : [four]
```

```
wooseok91@air: ~/my_proj/Homework/sanghoonlee/quiz
wooseok91@air:~/my_proj/Homework/sanghoonlee/quiz$ cat > myfifo
one
two
three
four
```