TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 최대성
c3d4s19@naver.com

2018.03.21. – 020일차 수업

---

* 리눅스 커널 함수

man [함수]

-> 리눅스 커널창에서 함수 Manual 보는 명령어

ps -ef

-> 현재 돌아가고있는 프로세스 정보

(PID -> 프로세스의 고유번호)

ps -ef|grep bash

-> 'bash'를 찾기

ps -ef|grep bash|grep -v grep

-> 'bash'를 찾는 프로세스는 제외하고 찾기

ps -ef|grep bash|grep -v grep|awk '{print $2}'

-> PID(Process ID) 번호 출력

tail -c [char개수] [파일명]

-> 맨 뒤에서 부터 [char개수]만큼 보여줌

tail -n [line개수] [파일명]

-> 맨 뒤에서 부터 [line개수]만큼 보여줌

mkfifo [파일명]

-> 파이프 파일 생성(일단은 데이터 값을 연결할 수 있는 통로 같은 것으로 알아두기)

cat >[파일명]

-> 커널창에서 파일 작성(원본 내용 삭제됨)

cp [복사파일 경로] [목적지 경로]

-> [복사파일 경로]에서 [목적지 경로]로 파일을 복사함

cp -r [복사폴더 경로] [목적지 경로]

-> [복사폴더 경로]에서 [목적지 경로]로 폴더를 복사함

ls -al

-> 모든 파일 표시

ls –al 명령어 내용 의미

-rw 일반파일

drw 디렉토리

prw 파이프

crw 캐릭터 디바이스(순서가 지켜져야 하는 것들) ex) 키보드, 그래픽, ...

brw 블록 디바이스(특정 단위로 움직임) ex)하드디스크, RAM, ...

//물리 메모리의 최소단위 = 4kb = 4096byte)

---

* dup() 함수

close([fd index 1]);

dup([fd index 2]);

-> 최근 종료 파일[fd index 1]을 복사해서 [fd index 2]로 그 기능을 대체함

(fd = file descriptor)

예제 1

```
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
int main() {
        int fd;
        fd = open("a.txt", O_WRONLY | O_CREAT |
O_TRUNC, 0644);
        close(1);
        dup(fd);
        printf("출력확인₩n");
        return 0;
}
```

예제 2

```
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
int main() {
        int fd;
        char buff[1024];
        fd = open("a.txt", O_RDONLY);
        close(0);
        dup(fd);
        gets(buff);
        printf(buff);
        return 0;
}
```

---

* Blocking 함수

read() -> blocking 함수(입력을 할 때까지 제어권을 넘겨주지 않고 계속 붙잡고 있음)

1. read() 함수를 blocking 상태로 사용하여 테스트

```
#include <stdio.h>
#include <string.h>
#include <fcntl.h>
int main() {

        int fd, ret;
        char buf[1024];
        mkfifo("myfifo");
        fd = open("myfifo", O_RDWR);
        for (;;) {
                ret = read(0, buf, sizeof(buf));
                buf[ret - 1] = 0;
                // buf[읽은값 - 1] 위치에 NULL
넣음 (원래 '₩n'값 들어있는데 덮어씀)
                printf("Keyboard Input : [%s]₩n",
buf);
                read(fd, buf, sizeof(buf));
                buf[ret - 1] = 0;
                printf("Pipe Input : [%s]₩n",
buf);
        }
        return 0;
}
```

새 커널창을 열고 cat >myfifo 입력하여 테스트

-> 글자를 입력해도 read()함수에 걸리면 입력실행 전까지 다른걸 blocking 한다.

2. read()함수를 non_blocking 상태로 사용하여 테스트

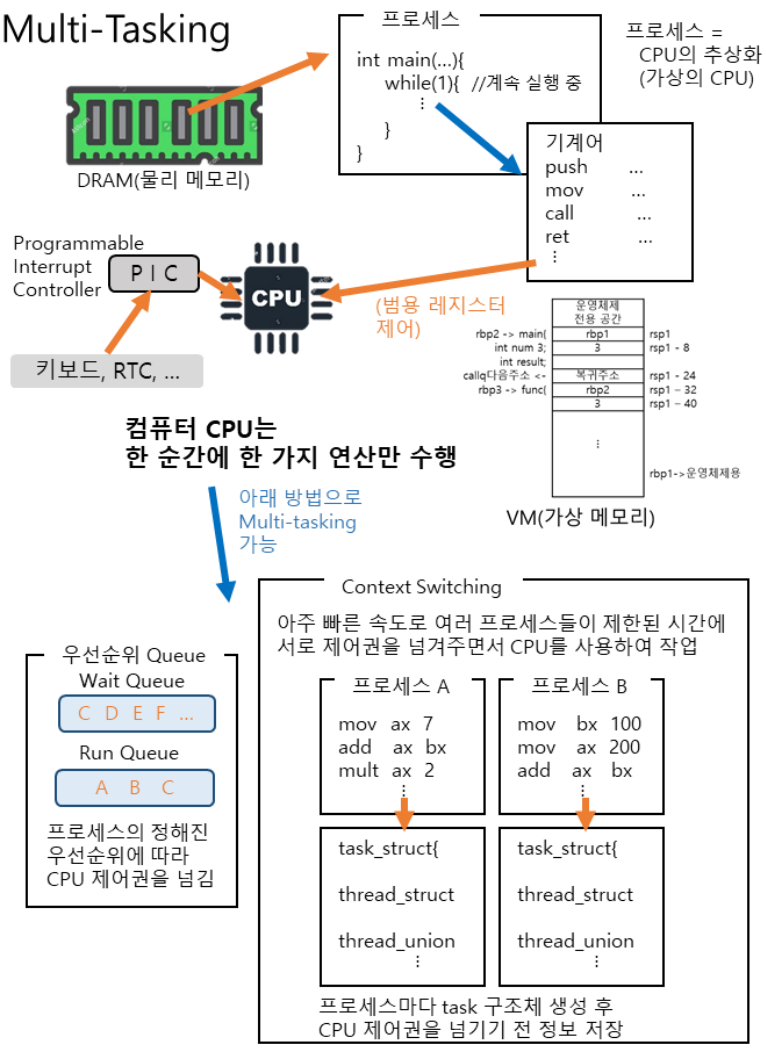
```c
#include <string.h>
#include <fcntl.h>
#include <stdlib.h>
#include <stdio.h>
int main() {
        int fd, ret;
        char buf[1024];
        mkfifo("myfifo");
        fd = open("myfifo", O_RDWR);
        fcntl(0, F_SETFL, O_NONBLOCK);
        fcntl(fd, F_SETFL, O_NONBLOCK);
        for (;;) {
                if ((ret = read(0, buf,
sizeof(buf))) > 0) {
                        buf[ret - 1] = 0;
                        printf("Keyboard 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;
}
```

fcntl([fd index], [F_SETFL], [O_NONBLOCK])

-> 파일 컨트롤 함수

[F_SETFL] -> 파일 권한 세팅

[O_NONBLOCK] -> non_bloking으로 설정



프로세스는 CPU의 추상화(가상의 CPU)이다.

실행파일(a.out 등)에서 무한루프 돌고있는 함수 자체가 프로세스이다.