

TI DSP, MCU 및 Xilinx Zynq FPGA 프로그램 전문가 과정

강사 - Innova Lee(이상훈)

gcccompil3r@gmail.com

학생 - 은태영

zero_bird@naver.com

명령어

```
tewill@tewill-B85M-D3H: ~  
tewill 1826 1821 0 05:23 pts/3 00:00:00 bash  
tewill 1837 1406 0 05:23 ? 00:00:00 zeitgeist-datahub  
tewill 1844 1007 0 05:23 ? 00:00:00 /bin/sh -c /usr/lib/x86_64-linux  
tewill 1848 1844 0 05:23 ? 00:00:00 /usr/bin/zeitgeist-daemon  
tewill 1855 1007 0 05:23 ? 00:00:00 /usr/lib/x86_64-linux-gnu/zeitge  
tewill 1905 1007 0 05:24 ? 00:00:00 /usr/lib/gvfs/gvfsd-network --sp  
tewill 1928 1007 0 05:24 ? 00:00:00 /usr/lib/gvfs/gvfsd-dnssd --spaw  
tewill 1954 1007 0 05:24 ? 00:00:00 /usr/lib/gvfs/gvfsd-metadata  
tewill 1961 1406 0 05:24 ? 00:00:00 update-notifier  
tewill 2027 1406 0 05:25 ? 00:00:00 /usr/lib/x86_64-linux-gnu/deja-d  
root 2138 2 0 05:28 ? 00:00:01 [kworker/u8:0]  
root 2142 2 0 05:28 ? 00:00:00 [kworker/1:2]  
root 2146 2 0 05:28 ? 00:00:00 [kworker/2:1]  
root 2182 2 0 05:30 ? 00:00:00 [kworker/0:0]  
tewill 2208 1007 1 05:31 ? 00:00:00 eog /home/tewill/Pictures/5.png  
tewill 2230 1826 0 05:31 pts/3 00:00:00 ps -ef  
tewill@tewill-B85M-D3H:~$ ps -ef | grep bash  
tewill 1826 1821 0 05:23 pts/3 00:00:00 bash  
tewill 2232 1826 0 05:32 pts/3 00:00:00 grep --color=auto bash  
tewill@tewill-B85M-D3H:~$ ps -ef | grep bash | grep -v grep  
tewill 1826 1821 0 05:23 pts/3 00:00:00 bash  
tewill@tewill-B85M-D3H:~$ ps -ef | grep bash | grep -v grep | awk '{print $2}'  
1826  
tewill@tewill-B85M-D3H:~$
```

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

- ❖ `ps -ef` : 현재 구동중인 프로세스(main)
- ❖ `|` : 파이프
- ❖ `grep bash(터미널)` : `bash(터미널)`을 찾아낸다.
- ❖ `grep -v grep` : 자기 자신(찾는 프로세스)를 제외.
- ❖ `awk '{print $2}'` : 2번째에 위치한 PID 출력.

명령어

```
tewill@tewill-B85M-D3H: ~  
tewill@tewill-B85M-D3H:~$ tail -c 10 c_02.c  
turn 0;  
}  
tewill@tewill-B85M-D3H:~$
```

tail -c 20 oo.c

- ❖ tail : 뒤에서 마지막 부터 읽을 때.
- ❖ -c 0 : 0 번째 문자 수
- ❖ -n 0 : 0 번째 행
- ❖ /var/log/messages :

명령어

```
tewill@tewill-B85M-D3H: ~  
crw----- 1 root root 10, 223 3월 22 05:23 uinput  
crw-rw-rw- 1 root root 1, 9 3월 22 05:23 urandom  
crw----- 1 root root 10, 240 3월 22 05:23 userio  
crw-rw---- 1 root tty 7, 0 3월 22 05:23 vcs  
crw-rw---- 1 root tty 7, 1 3월 22 05:23 vcs1  
crw-rw---- 1 root tty 7, 2 3월 22 05:23 vcs2  
crw-rw---- 1 root tty 7, 3 3월 22 05:23 vcs3  
crw-rw---- 1 root tty 7, 4 3월 22 05:23 vcs4  
crw-rw---- 1 root tty 7, 5 3월 22 05:23 vcs5  
crw-rw---- 1 root tty 7, 6 3월 22 05:23 vcs6  
crw-rw---- 1 root tty 7, 128 3월 22 05:23 vcsa  
crw-rw---- 1 root tty 7, 129 3월 22 05:23 vcsa1  
crw-rw---- 1 root tty 7, 130 3월 22 05:23 vcsa2  
crw-rw---- 1 root tty 7, 131 3월 22 05:23 vcsa3  
crw-rw---- 1 root tty 7, 132 3월 22 05:23 vcsa4  
crw-rw---- 1 root tty 7, 133 3월 22 05:23 vcsa5  
crw-rw---- 1 root tty 7, 134 3월 22 05:23 vcsa6  
drwxr-xr-x 2 root root 60 3월 22 05:23 vfio  
crw----- 1 root root 10, 63 3월 22 05:23 vga_arbiter  
crw----- 1 root root 10, 137 3월 22 05:23 vhci  
crw----- 1 root root 10, 238 3월 22 05:23 vhost-net  
crw----- 1 root root 10, 241 3월 22 05:23 vhost-vsock  
crw-rw-rw- 1 root root 1, 5 3월 22 05:23 zero  
tewill@tewill-B85M-D3H:~$
```

- ❖ d : 디렉토리 파일
- ❖ - : 일반 파일
- ❖ p : 파이프
- ❖ b : 블록. 특정 단위를 갖고 움직임.
- ❖ 4kb(페이지) 물리 메모리의 최소 단위.
- ❖ c : 캐릭터 디바이스. 순서. 키보드 모니터 등.
- ❖ s : 소켓. b로 통합 됨.

기타

- ❖ man 함수 : 함수에 대한 설명.
- ❖ man -s2 함수 : 시스템 콜 확인.
- ❖ atoi(문자열) : 문자열의 숫자를 정수로 리턴 한다.
- ❖ sprintf(저장할 문자열, "%d", 숫자 변수) : 숫자 등 2 번째 인자의 값을 1 번째 인자로 저장한다.
- ❖ open(파일 명, O_APPEND) : O_APPEND 는 해당 파일의 가장 끝으로 pos 를 위치시킨다.
- ❖ slab 할당자가 32byte 로 관리하기 때문에 32의 배수로 하는 것이 성능이 빠르다.

파일

```
tewill@tewill-B85M-D3H: ~  
#include <stdio.h>  
#include <unistd.h>  
#include <fcntl.h>  
  
int main(void)  
{  
    int fd;  
    fd = open("a.txt", O_WRONLY | O_CREAT | O_TRUNC, 0644);  
    close(1);  
    dup(fd);  
    printf("출력될까 ?\n");  
    return 0;  
}
```

1,5 All

- ❖ dup : 가장 마지막으로 close 한 파일을 복사한다.
- ❖ 표준 출력을 닫고, dup 을 통해 fd 에 복사한다.
- ❖ printf 로 출력되어야 할 문자열이 파일에 저장된다.

파일

```
tewill@tewill-B85M-D3H: ~  
#include <stdio.h>  
#include <unistd.h>  
#include <fcntl.h>  
  
int main(void)  
{  
    int fd;  
    char buff[1024];  
    fd = open("a.txt", O_RDONLY);  
    close(0);  
    dup(fd);  
    gets(buff);  
    printf("출력이 될까?\n");  
  
    printf("%s\n", buff);  
    return 0;  
}
```

1,1 All

- ❖ gets() : 문자열을 입력받는다.
- ❖ 표준입력을 닫고 dup 을 통해 fd 에 복사한다.
- ❖ gets 를 통해 입력된 것이 파일에 저장된다.

파일

```
tewill@tewill-B85M-D3H: ~  
#include <sys/types.h>  
#include <unistd.h>  
#include <fcntl.h>  
  
int main(int argc, char *argv[])  
{  
    int i;  
    char ch = 'a';  
    int fd = open(argv[1], O_WRONLY | O_CREAT | O_TRUNC, 0644);  
    lseek(fd, 512 - 1, SEEK_SET);  
    write(fd, &ch, 1);  
    close(fd);  
    return 0;  
}
```

1,1 All

❖ lseek 를 통해 파일의 pos 를 조작할 수 있다.

❖ fd 파일의 512에 위치한 값을 a로 바꾼다.

파일

```
tewill@tewill-B85M-D3H: ~  
#include <fcntl.h>  
#include <stdio.h>  
#include <string.h>  
  
int main(void)  
{  
    int fd, ret;  
    char buf[1024];  
    mkfifo("myfifo");  
    fd = open("myfifo", O_RDWR);  
    for(;;)  
    {  
        ret = read(0, buf, sizeof(buf));  
        buf[ret - 1] = 0;  
        printf("Keyboard input : [%s]\n", buf);  
        read(fd, buf, sizeof(buf));  
        buf[ret - 1] = 0;  
        printf("Pipe input : [%s]\n", buf);  
    }  
    return 0;  
}
```

1,1 All

- ❖ mkfifo : 파이프 파일을 생성.
- ❖ 파이프 파일 : 데이터 값을 연결하는 통로.
- ❖ read : 블로킹. 즉, 제어권을 갖고 있다.
- ❖ 파이프 파일을 통해 통신을 하지만, read 로 인해 실행 위치에서 입력을 해야 받아올 수 있다.
- ❖ fcntl(위치, 설정, 내용) 을 통해 변경 가능하다.
- ❖ F_SETFL : 파일 권한 설정.
- ❖ O_NONBLOCK

CPU와 Multi-tasking

- ❖ CPU 는 오로지 한 순간에 한가지 연산만 수행한다.
- ❖ 1Clock 은 한 개의 연산을 하는데 걸리는 시간이다.
- ❖ CPU 주파수가 일반적으로 Ghz 단위로 동작하는데, 이는 10^9 로서 1Clock의 동작 속도는 10^{-9} 가 된다.
- ❖ 아주 빠른 속도로 연산처리가 되기 때문에 Context switching 이 가능하다.
- ❖ Context switching : DRAM 에 CPU의 진행중인 연산 처리를 저장하고 다른 프로세스의 연산을 불러와 진행한다. 이때, 중요도에 따라 우선순위가 지정되며, 낮은 우선순위의 연산이 진행 안되는 것을 방지하기 위하여 할당시간을 비율로 하여, 일정 비율 이상이 될 경우 동작을 넘기도록 한다.
- ❖ 리눅스 커널의 경우, Run Queue 와 Wait Queue 가 존재하여 연산을 불러오고 실행한다.
- ❖ PIC : 하드웨어 장치의 입력을 전기 신호로 CPU 에 전달하는 역할을 한다.

