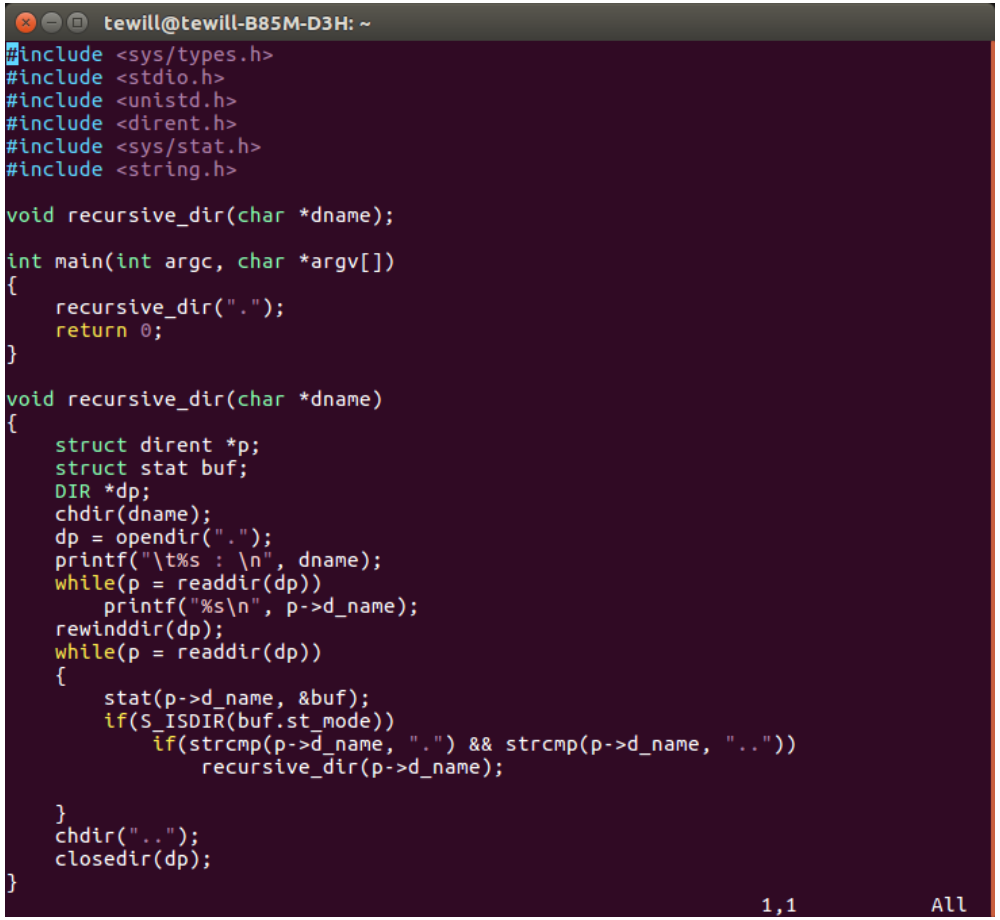
TI DSP, MCU 및 Xilinx Zynq FPGA 프로그램 전문가 과정

강사 - Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 - 은태영
zero_bird@naver.com

파일



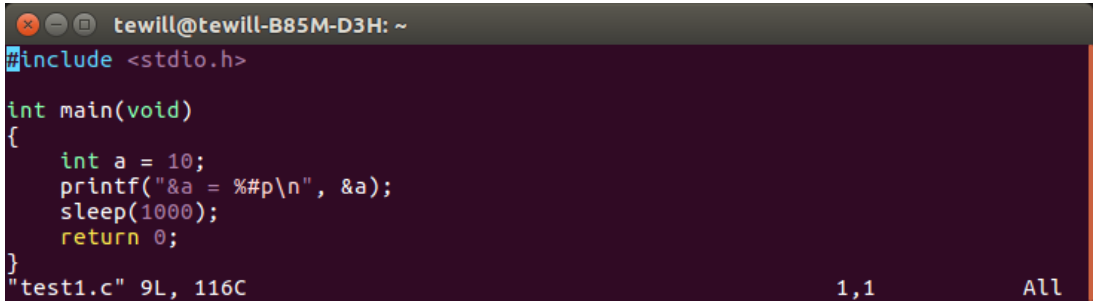
- ❖ rewinddir() : 읽고 있는 파일의 위치를 초기화.

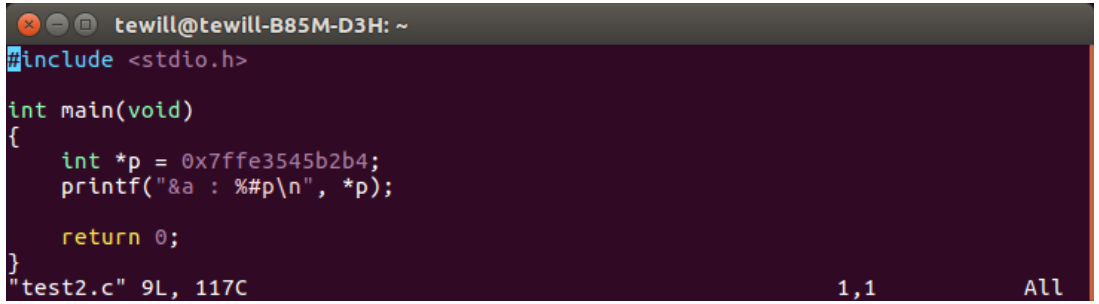
- ❖ 현재 위치의 디렉토리 파일을 열어 dp 에 저장.
- ❖ 현재 위치를 출력하고 반복문을 통해 파일 내부를 읽는다.
- ❖ 파일 내부를 모두 읽은 후 파일의 위치를 초기화.
- ❖ 반복문을 통해 내부에 위치한 디렉토리 탐색.
- ❖ 탐색 결과 “.” 과 “..” 이 아닐 경우 재귀호출.
- ❖ 완료 시, 디렉토리를 닫는다.

메모리 권한

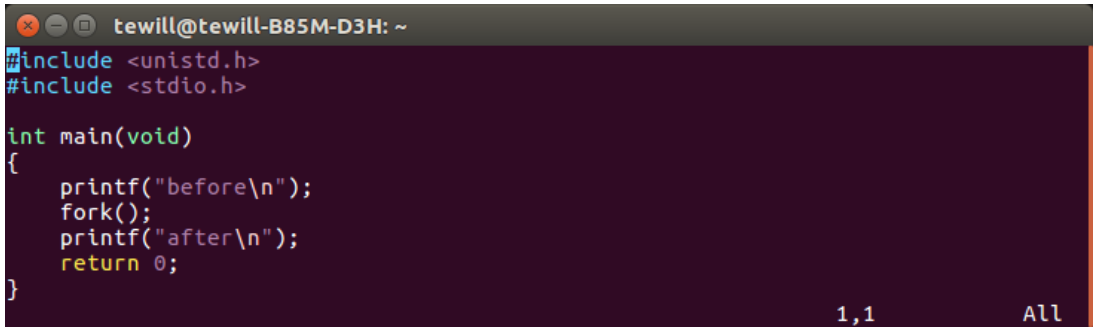
프로세스 별 페이지의 속성을 저장한다. 어떤 녀석이 이 페이지에 대한 읽기와 쓰기 권한을 갖고 있는지
메모리마다 권한을 별도로 갖고 있다.

❖ “%#p” : 주소 값 표시.(0x 표시를 # 이 함(과거).)





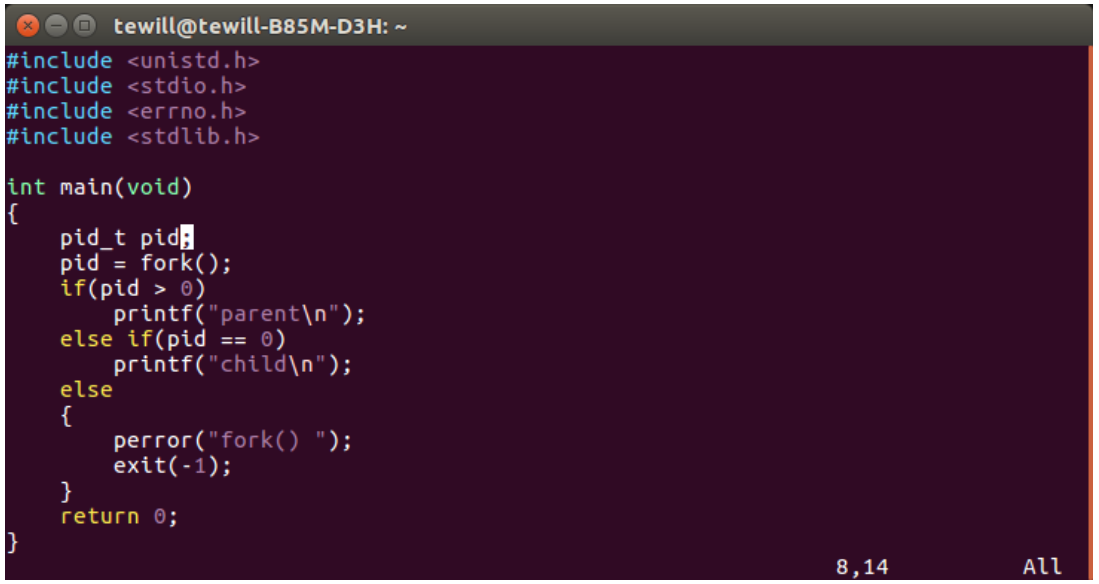
fork()



- fork() : 프로세스 자기 자신을 복사한다.
- 복사를 하지만 PID 는 다르다. (복제된 VM)
- 실행이 된 줄부터 복사를 한다.
- 리턴 값으로 자식의 PID 를 리턴 한다.
- 복사된 프로세스는 0 을 리턴 한다.

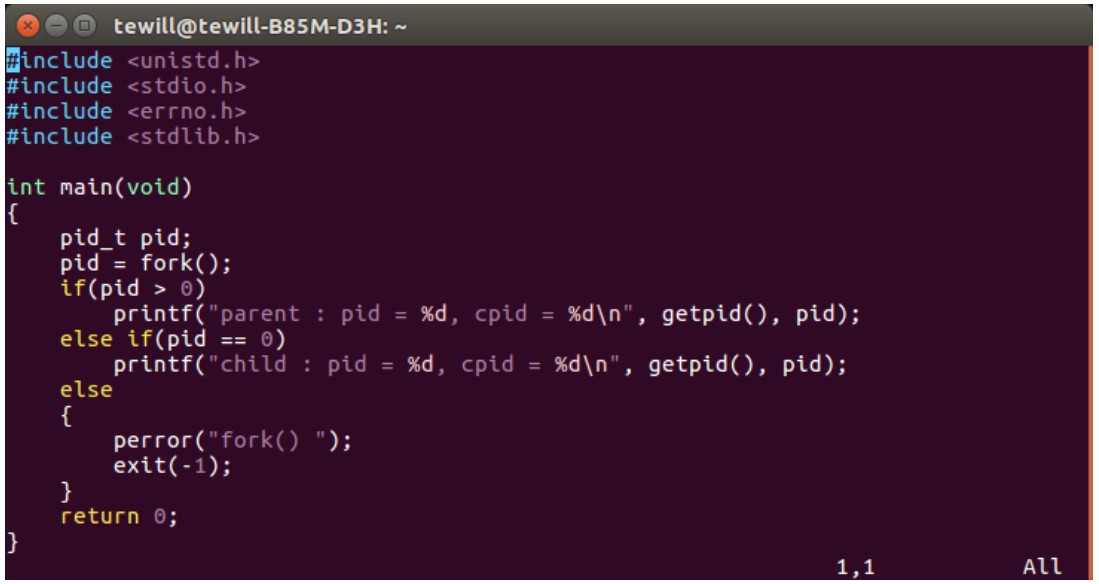
- 복사 시, C.O.W(copy on write) 형식.
- 필요한(쓰기가 발생한) 순간에 복사를 한다.

fork()



- pid_t : int 와 동일.
- fork 로 프로세스를 복제한다.
- 각각의 프로세스에 PID 를 pid 에 저장한다.
- 부모는 0 이상의 PID이기 때문에 parent 를 출력.
- 자식을 fork 의 리턴 이 0 이기 때문에 child 출력.

fork()



- getpid() : 자기 자신의 pid를 리턴 한다.
- 동일한 방식.
- 저장된 pid 의 출력을 통해 눈으로 확인 가능하다.
- 부모의 cpid 와 자식의 pid 가 같음을 알 수 있다.
- 자식의 경우, pid 는 자신의 값을 출력하지만, cpid 는 0 의 값을 출력한다.

fork()

tewill@tewill-B85M-D3H: ~

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

int main(void)
{
    pid_t pid;
    int i;
    pid = fork();
    if(pid > 0)
    {
        while(1)
            for(i = 0; i < 26; i ++)
            {
                printf("%c ", i + 'A');
                fflush(stdout);
            }

    }
    else if(pid == 0)
    {
        while(1)
            for(i = 0; i < 26; i++)
            {
                printf("%c ", i + 'a');
                fflush(stdout);
            }
    }
    else
    {
        perror("fork() ");
        exit(-1);
    }

    printf("\n");
    return 0;
}
```

1,1 All

- ❖ fflush(stdout) : 출력 버퍼를 비우는 함수다.
- ❖ 버퍼가 가득 찰 경우 출력이 무시되는 일이 발생.

- ❖ 복사된 프로세스의 진행 과정을 확인 할 수 있다.
- ❖ 우선순위에 따라 다른 결과 값이 나오지만,
  일반적으로 대소문자가 번갈아 가며 출력된다.

fork()

tewill@tewill-B85M-D3H: ~

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

int global = 100;

int main(void)
{
    int local = 10;
    pid_t pid;
    int i;
    pid = fork();

    if(pid > 0)
    {
        printf("global : %d, local : %d\n", global, local);
    }
    else if(pid == 0)
    {
        global++;
        local++;
        printf("global : %d, local : %d\n", global, local);
    }
    else
    {
        perror("fork() ");
        exit(-1);
    }
    printf("\n");
    return 0;

}
```

1,1 All

- 각각의 가상 메모리가 존재하는지 확인한다.
- 프로세스가 복사된 후 자식에서 전역 변수와 지역 변수의 값을 변경하지만 부모에서 변경되지 않는 모습을 볼 수 있다.

fork()

tewill@tewill-B85M-D3H: ~

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

int main(void)
{
    int fd, ret;
    char buf[1024];
    pid_t pid;
    fd = open("myfifo", O_RDWR);
    if((pid = fork()) > 0)
    {
        for(;;)
        {
            ret = read(0, buf, sizeof(buf));
            buf[ret] = 0;
            printf("keyboard : %s\n", buf);
        }
    }
    else if(pid == 0)
    {
        for(;;)
        {
            ret = read(fd, buf, sizeof(buf));
            buf[ret] = 0;
            printf("myfifo : %s\n", buf);
        }
    }
    else
    {
        perror("fork() ");
        exit(-1);
    }
    close(fd);
    return 0;
}
```

1,1 All

- fifo 의 응용이다.
- 과거 read 는 블럭 함수이기 때문에 fifo 사용 시 한쪽에서 입력을 해야 진행되는 문제가 있었다.
- 하지만, 프로세스의 복사를 통하여 입력 부분과 파일에서 읽어오는 것이 같이 동작하게 된다.
- 즉 두 프로세스가 한개의 프로세스인 것 처럼 출력을 시키는 모습을 볼 수 있다.

fork()

tewill@tewill-B85M-D3H: ~

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

int main(void)
{
    pid_t pid;
    if((pid = fork()) > 0)
        sleep(1000);
    else if(pid == 0)
        ;
    else
    {
        perror("forc() ");
        exit(-1);
    }

    return 0;
}
```

1,1 All

- zombie process 를 확인 가능하다.
- 자식이 종료되었지만, 부모가 이를 확인 불가능한 상태이다.

fork()

tewill@tewill-B85M-D3H: ~

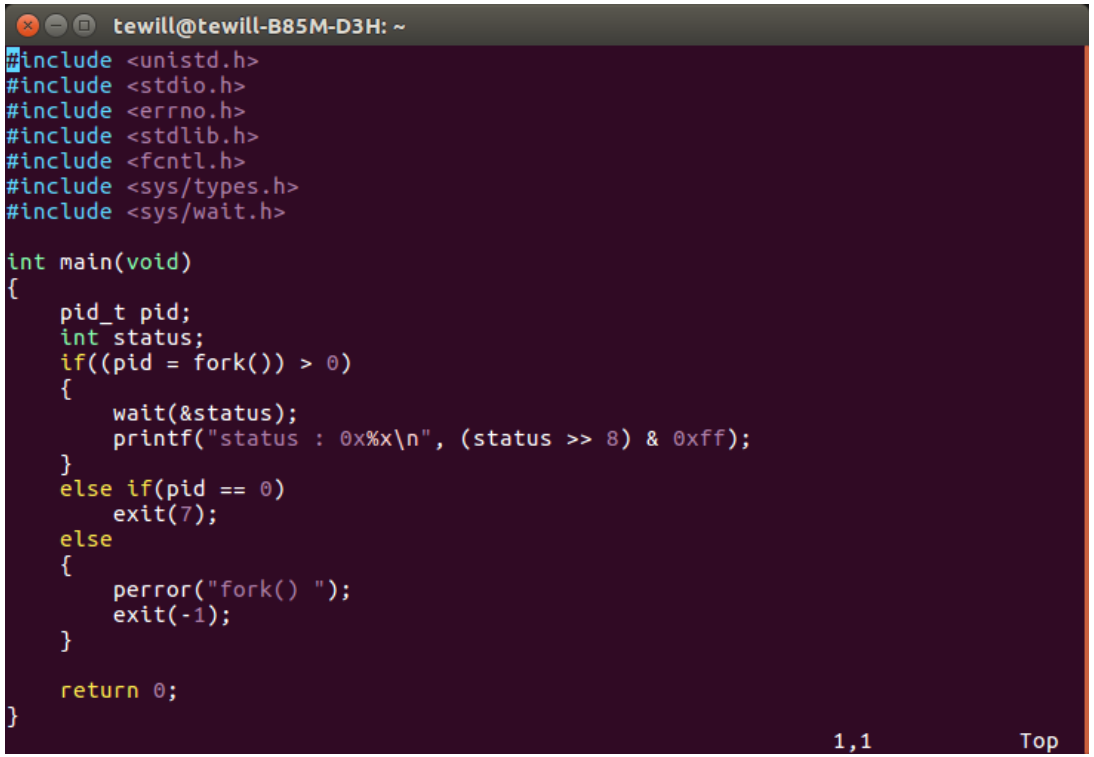
```c
#include <unistd.h>
#include <stdio.h>
#include <errno.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/wait.h>

int main(void)
{
    pid_t pid;
    int status;
    if((pid = fork()) > 0)
    {
        wait(&status);
        printf("status : %d\n", status);
    }
    else if(pid == 0)
        exit(7);
    else
    {
        perror("fork() ");
        exit(-1);
    }

    return 0;
}
```

1,1 All

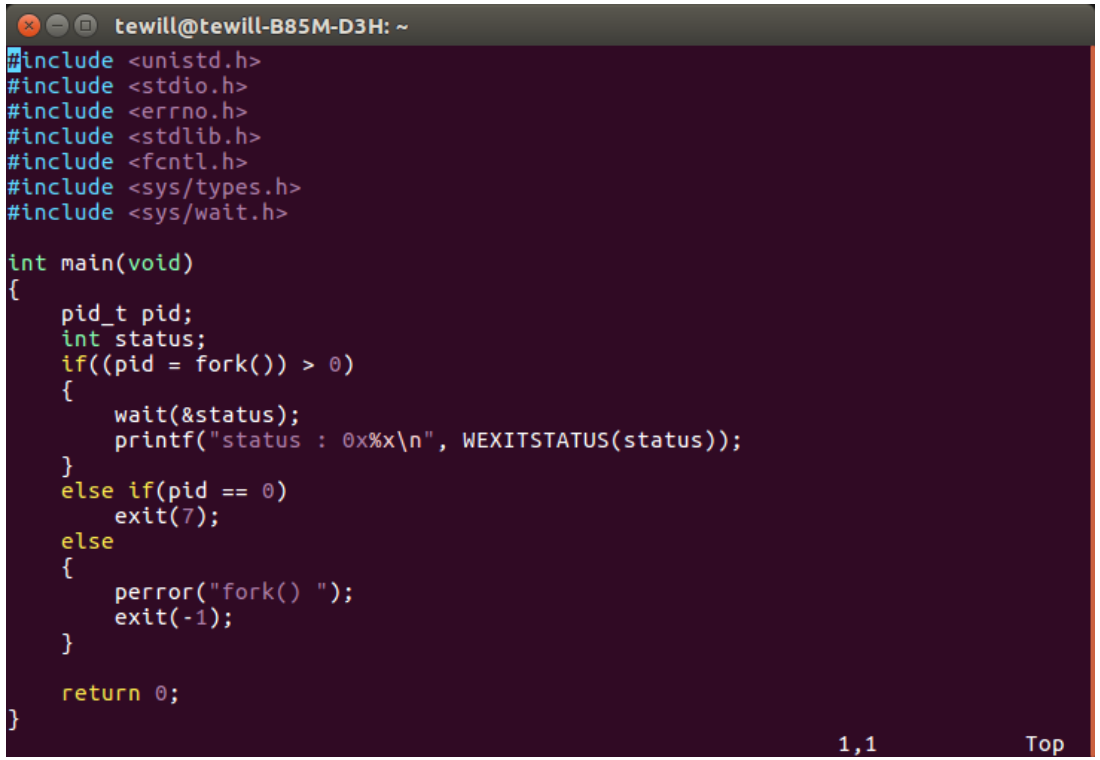
- ❖ wait(&status) : 자식의 상태 값을 받아온다.
- ❖ 시스템 콜 이다.
- ❖ 출력 값으로 1792 가 나오는 것이 확인 가능하다.
- ❖ 이는 1792 / 256 으로 확인 가능하다.
- ❖ 즉 >> 8 을 통해 원하는 값을 확인 가능하다.

fork()

‘ >> 8 ’ 을 통해 값을 받아오는 방식.



‘ WEXITSTATUS() ’ 를 통해 값을 받아오는 방식.



fork()

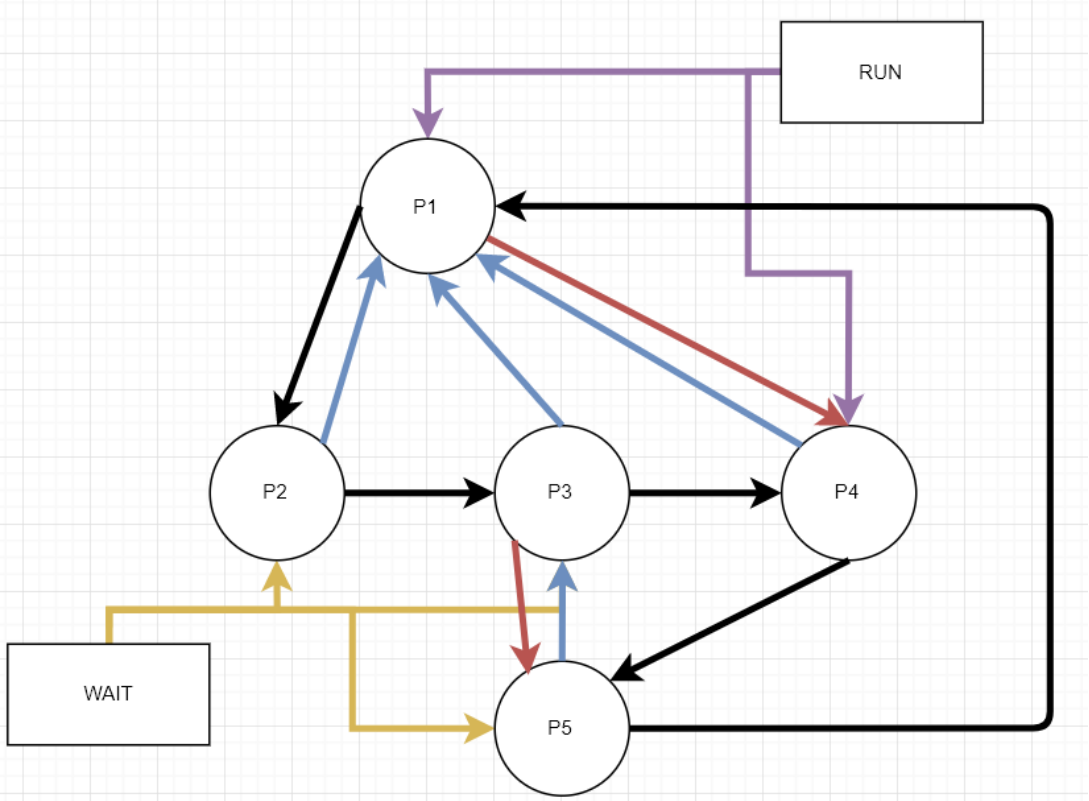
tewill@tewill-B85M-D3H: ~

```c
#include <unistd.h>
#include <stdio.h>
#include <errno.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/wait.h>

int main(void)
{
    pid_t pid;
    int status;
    if((pid = fork()) > 0)
    {
        wait(&status);
        printf("status : 0x%d\n", status & 0x7f);//WEXITSTATUS(status));
    }
    else if(pid == 0)
        abort();
    else
    {
        perror("fork() ");
        exit(-1);
    }

    return 0;
}
```

1,1 All

- abort() : 비정상 종료이다.
- 비정상 종료의 경우 8비트 내부에 값이 들어간다.
- 가장 앞의 1bit 는 죽은 메모리의 상태 값을 갖고 올 것인지에 대한 상태값이다.
- 시그널 값은 8번째 비트에 위치한다.

fork()



- 복제 시, 각 상황을 알려주는 포인터가 생성된다.
- 검정 : 순서를 나타내는 포인터.
- 파랑 : 자식이 부모를 향하는 포인터.
- 빨강 : 부모가 가장 마지막으로 생성한 자식을 향하는 포인터.
- 노랑 : 대기중인 프로세스를 나타내는 포인터.
- 보라 : 진행중인 프로세스를 나타내는 포인터.

시그널

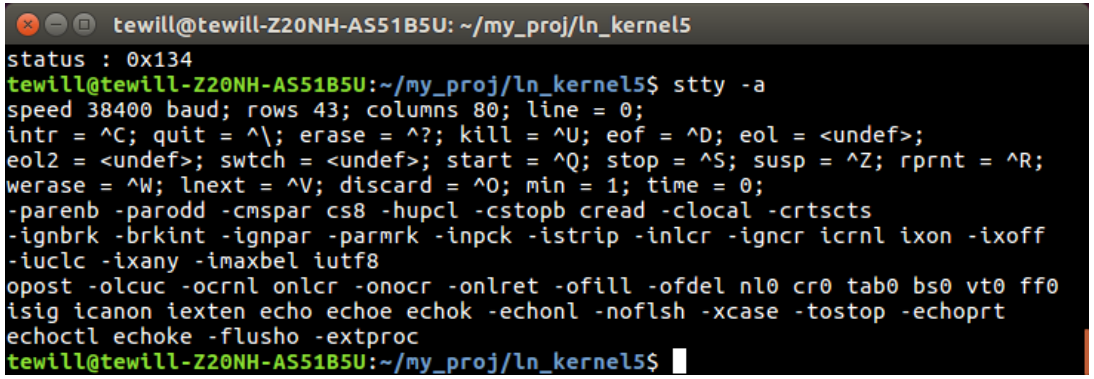
정상 종료

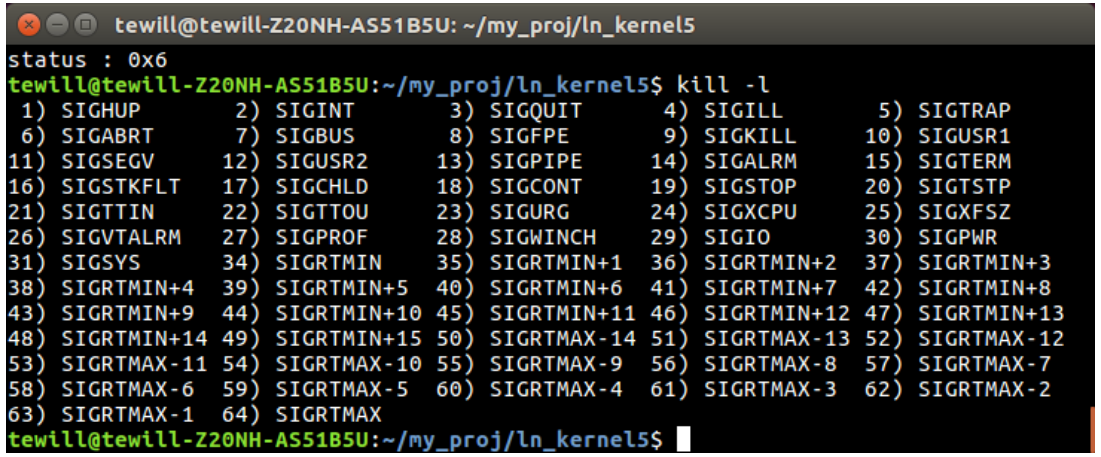
비정상 종료

명령어

sty -a : 사용되는 시그널을 출력한다.

kill -l : 리눅스 상에 존재하는 모든 시그널 번호를 출력한다.





ls 기능 구현

tewill@tewill-B85M-D3H: ~

```c
#include <stdio.h>
#include <unistd.h>
#include <dirent.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <string.h>
#include <pwd.h>
#include <grp.h>
#include <time.h>

void opt_l(char *);

void opt_a(char *,int);

int setopt(int, char **);

void recursive_dir(char *, int);

int main(int argc, char **argv)
{
        int flag = 0;

        flag = setopt(argc, argv);
        recursive_dir(".", flag);

        return 0;
}
```

1,1 Top

tewill@tewill-B85M-D3H: ~

```c
void recursive_dir(char *dname, int flag)
{
        if(flag & 4)
        {
                struct dirent *p;
                struct stat buf;
                DIR *dp;
                chdir(dname);
                dp = opendir(".");
                printf("\t%s : \n", dname);

                while(p = readdir(dp)) {
                        if(!(flag & 1))
                                if(p->d_name[0] == '.')
                                        if(p->d_name != "." && p->d_name != "..")
                                                continue;
                        if(flag & 2)
                                opt_l(p->d_name);
                        printf("%s\n", p->d_name);
                }
                rewinddir(dp);

                while(p = readdir(dp))
                {
                        stat(p->d_name, &buf);

                        if(!(flag & 1))
                                if(p->d_name[0] == '.')
                                        if(p->d_name != "." && p->d_name != "..")
                                                continue;

                        if(S_ISDIR(buf.st_mode))
                                if(strcmp(p->d_name, ".") && strcmp(p->d_name, ".."))
                                        recursive_dir(p->d_name, flag);

                }
                chdir("..");
                closedir(dp);
        } else
                opt_a(".", flag);
        return;
}
```

30,1 20%

ls 기능 구현

tewill@tewill-B85M-D3H: ~

```
void opt_l(char *argv)
{
        struct stat buf;

        struct passwd *pw;
        struct group *gr;

        struct tm *tm;

        char perm[11] = "__________";

        char rwx[4] = "rwx";
        char sst[4] = "sst";
        int i;

        stat(argv, &buf);
        if(S_ISDIR(buf.st_mode))
                perm[0] = 'd';
        if(S_ISREG(buf.st_mode))
                perm[0] = '-';
        if(S_ISFIFO(buf.st_mode))
                perm[0] = 'p';
        if(S_ISLNK(buf.st_mode))
                perm[0] = 'l';
        if(S_ISSOCK(buf.st_mode))
                perm[0] = 's';
        if(S_ISCHR(buf.st_mode))
                perm[0] = 'c';
        if(S_ISBLK(buf.st_mode))
                perm[0] = 'b';
                                                        63,1          43%
```

tewill@tewill-B85M-D3H: ~

```
        for(i = 0; i < 9; i++)
                if((buf.st_mode >> (8 - i)) & 1)
                        perm[i + 1] = rwx[i % 3];

        for(i = 0; i < 3; i++)
                if((buf.st_mode >> (11 - i)) & 1)
                        if(perm[(i + 1) * 3] == '-')
                                perm[(i + 1) * 3] = sst[i]^0x20;
                        else
                                perm[(i + 1) * 3] = sst[i];
        printf("%s ", perm);
        printf("%ld ", buf.st_nlink);
        pw = getpwuid(buf.st_uid);
        printf("%s ", pw->pw_name);
        gr = getgrgid(buf.st_gid);
        printf("%s ", gr->gr_name);
        printf("%ld ", buf.st_size);
        tm = localtime(&buf.st_mtime);
        printf("%d-%02d-%02d %02d:%02d ", tm->tm_year + 1900,
                tm->tm_mon + 1, tm->tm_mday, tm->tm_hour, tm->tm_min);

        return;
}
                                                        93,0-1        62%
```

ls 기능 구현

tewill@tewill-B85M-D3H: ~

```
void opt_a(char *d_name,int flag)
{
        DIR *dp;
        int i = 0;
        struct dirent *p;

        dp = opendir(d_name);
        while(p = readdir(dp))
        {
                if(!(flag & 1))
                        if(p->d_name[0] == '.')
                                continue;

                if(flag & 2)
                        opt_l(p->d_name);
                printf("%-16s ", p->d_name);

                if(flag & 2)
                        printf("\n");
                else if((i + 1) % 5 == 0)
                        printf("\n");
                i++;
        }
        printf("\n");
        closedir(dp);
        return;
}
                                                        117,0-1       80%
```

tewill@tewill-B85M-D3H: ~

```
int setopt(int argc, char **argv)
{
        int cmd;
        int flag = 0;
        while((cmd = getopt(argc, argv, "alRi")) > 0)
        {
                switch(cmd)
                {
                        case 'a':
                                flag |= 1;
                                break;
                        case 'l':
                                flag |= 2;
                                break;
                        case 'R':
                                flag |= 4;
                                break;
                        case 'i':
                                flag |= 8;
                                break;
                        default:
                                break;
                }
        }
        return flag;
}
                                                        172,1         Bot
```