Simple Shell

운영체제론 프로젝트 #1

소프트웨어학부 소프트웨어전공

2019054693 강예원

0. Simple Shell 알고리즘

우선 main 함수에서 명령어를 입력받는다. 입력받은 명령어를 strtok_r(char *str, const char *delim, char **saveptr) 함수를 이용하여 delim(" ", "₩t", "₩n")을 기준으로 명령어를 구분한다. 반복문을 이용하여 구분한 명령어가 "|", "<", ">", "&" 와 같은지 비교하고 그에 맞는 함수를 실행시킨다.

execute(int back, char *argv[]) 함수에서는 연산자가 없는 명령어를 실행시킨다. 먼저 fork()를 통해 자식 프로세스를 만들어준다. 부모 프로세스에서는 background process (back == 1) 이면 waitpid 함수를 이용해 options에 WNOHANG을 넣어 자식 프로세스가 끝날 때까지 기다리지 않고 종료시키게 한다. foreground process일 경우 wait() 함수를 이용하여 자식프로세스가 종료될 때까지 대기하도록 한다. 자식 프로세스에서는 execvp 함수를 이용해 명령어(argv)를 실행시킨다.

redir_in(int back, char *argv[], char* input), redir_out(int back, char *argv[], char* output) 함수에서는 각각 input redirection, output redirection을 수행한다. 두 함수 모두 처음에 fork()를 통해 자식 프로세스를 만들어준다. 부모 프로세스는 execute 함수와 동일하다. 자식 프로세스에서 두 함수가 다르게 진행된다. Redirection input의 경우 표준 입력을 특정 파일에서 받도록 대체하는 것이다. Input 파일을 O_RDONLY 로 열어주고 그 file descriptor를 input_fd 라 하자. STDIN_FILENO가 0이므로 dup2(input_fd, 0) 에 의해 표준 입력이 input 파일로 재지정된다. input_fd 는 더 이상 필요가 없으므로 닫아준다. Execvp 함수를 이용해 명령어(argv)를 실행하면 input 파일로부터 입력을 받는다.

Redirection output의 경우 명령이나 프로그램 등의 실행 결과를 특정 파일에 출력하는 것이다. output 파일을 O_CREAT | O_TRUNC | O_WRONLY 으로 열어주고 그 file descriptor를 output_fd 라 하자. STDOUT_FILENO가 1이므로 dup2(output_fd, 1) 을 하면 표준 출력이 output 파일로 재지정된다. 즉, 이후 모든 출력결과가 output 파일로 쓰여진다. output_fd 는 더 이상 필요가 없으므로 닫아준다. execvp 함수를 이용해 명령어(argv)를 실행하면 실행 결과가 output 파일로 저장된다.

pipeAct(int back, char* arg1[], char* arg2[]) 함수에서는 pipe를 수행한다. 먼저 fork()를 통해 자식 프로세스를 만들어준다. 부모 프로세스는 execute 함수와 동일하고 자식 프로세스에서 한번 더 fork()를 하여 프로세스로 데이터를 전달하도록 구현하였다. 우선 2개의 File descriptor를 저장하기 위한 배열 fd[2]를 생성한다. 그 다음에 Pipe() 호출하면 그 배열은 연결된 2개의 file descriptor로 채워지게된다. 이때 fd[0]은 부모 프로세스가 파이프에 데이터를 쓰는데 사용하고, fd[1]은 자식 프로세스가 파이프로 데이터를 읽는데 사용된다. 따라서 부모 프로세스에서는 파이프에 데이터를 쓸 것이므로 dup2(fd[0], 0) 을 하여 fd[0] 을 표준 입력으로 재지정한다. 그 후 execvp 함수를 이용해 파이프 뒤의 명령어(arg2)를 실행한다. 자식 프로세스에서는 파이프의 데이터를 읽을 것이므로 dup2(fd[1], 1)을 하여 fd[1]을 표준 출력으로 재지정한다. 그 후 execvp 함수를 이용해 파이프 앞의 명령어(arg1)을 실행한다.

1. 프로그램 소스 파일

```c
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <sys/wait.h>
#include <fcntl.h>
#define MAXARG 7

// Input Redirection 함수
void redir_in(int back, char *argv[], char* input) {
    int fd;
    int status;
    pid_t pid;

    pid = fork();

    if(pid < 0) {
        perror("Fork Failed\n");
    }

     // 부모일 경우
    else if(pid > 0) {
        if(!back) pid = wait(&status);    // foreground process
                                          // 자식 프로세스가 종료될 때 까지 대기
        else {                            // background process
              printf("[1] %d\n", getpid());      // pid 출력

              //자식 프로세스가 종료될 때 까지 기다리지 않고 종료
              waitpid(pid, &status, WNOHANG);

        }
    }

    // 자식일 경우
    else {
        if((fd = open(input, O_RDONLY)) == 1) {
            perror(argv[0]);
            exit(2);
        }
        dup2(fd, 0);  // fd를 표준 입력으로 redirection
        close(fd);
        execvp(argv[0], argv);  // argv 실행
    }
}

// Output Redirection 함수
void redir_out(int back, char *argv[], char* output) {
```

```
    int fd;
    int status;
    pid_t pid;

    pid = fork();
    if(pid < 0) {
        perror("Fork Failed\n");
    }

    // 부모일 경우
    else if(pid > 0) {
        if(!back) pid = wait(&status);    // foreground process

        else {                            // background process
              printf("[1] %d\n", getpid());
              waitpid(pid, &status, WNOHANG);
        }
    }

    // 자식일 경우
    else {
        fd = open(output, O_CREAT | O_TRUNC | O_WRONLY, 0600);
        dup2(fd, 1);            // fd를 표준 출력으로 redirection
        close(fd);
        execvp(argv[0], argv);  // argv 실행
    }
}


// Pipe 실행 함수
void pipeAct(int back, char* arg1[], char* arg2[]) {
    int fd[2];      // 2개의 fd를 담을 배열 정의
    int status;     // 자식 프로세스의 상태
    pid_t pid1;
    pid_t pid2;

    pid1 = fork();
    if(pid1 < 0) {
        perror("1st Fork Failed\n");
        exit(1);
    }

    // 부모일 경우
    else if(pid > 0) {
        if(!back) pid = wait(&status);    // foreground process


        else {                            // background process
```

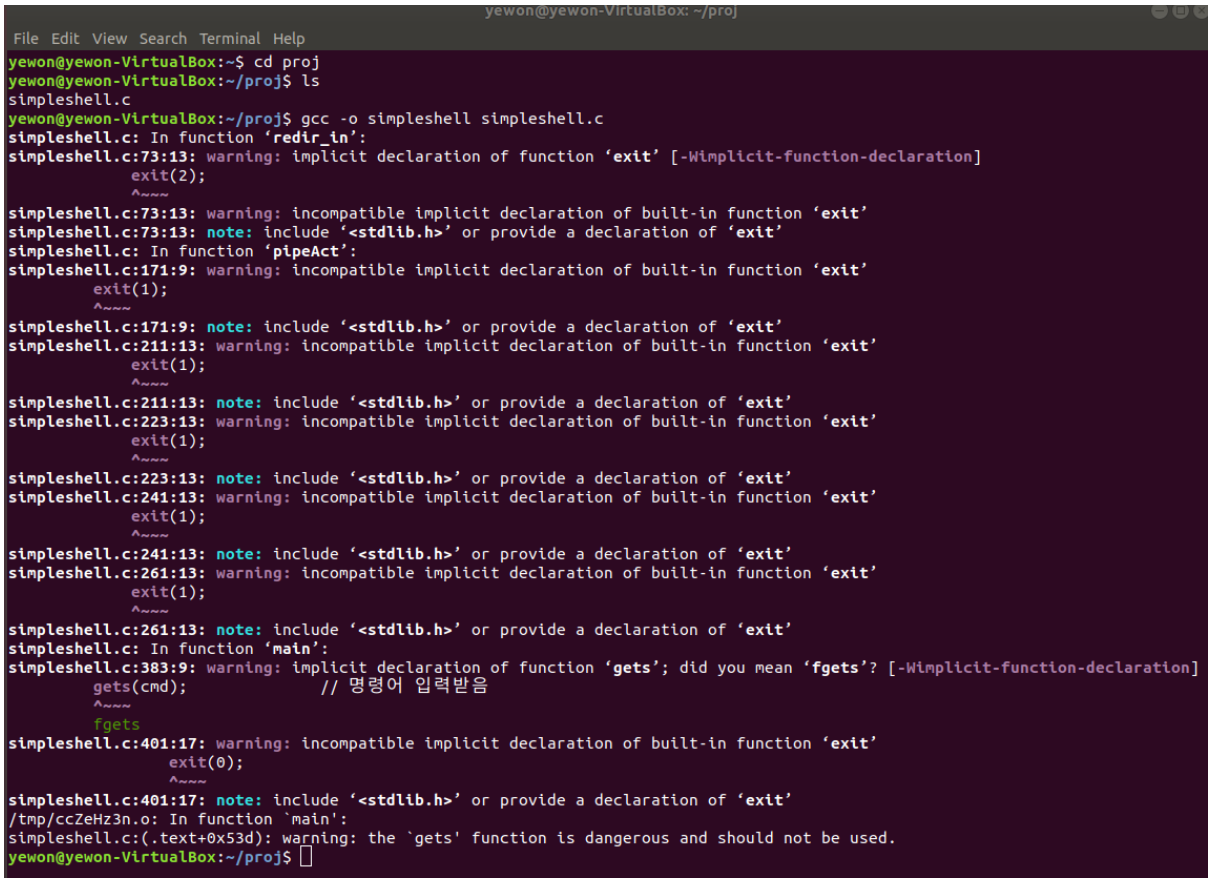
```
            printf("[1] %d\n", getpid());
            waitpid(pid, &status, WNOHANG);
        }
    }

    else {
        // pipe를 호출해 두개의 fd로 배열을 채워줌
        // fd[1] 에 쓰고 fd[0] 으로 읽어야함

        if(pipe(fd) == -1) {
            perror("Pipe Failed\n");
            exit(1);
        }

        pid2 = fork();
        if(pid2 < 0) {
            perror("2nd Fork Failed\n");
            exit(1);
        }
        else if(pid2 == 0) {            // 자식 프로세스 일 경우
            dup2(fd[1], 1);             // fd[1]을 표준 출력으로 redirection
            close(fd[0]);               // fd[0]과 fd[1] 닫기
            close(fd[1]);
            execvp(arg1[0], arg1);      // arg1 실행

            perror("Parent Execvp Failed\n");
            exit(1);
        }

        else {                          // 부모 프로세스 일 경우
            dup2(fd[0], 0);             // fd[0]을 표준 입력으로 redirection
            close(fd[1]);               // fd[1]과 fd[0] 닫기
            close(fd[0]);
            execvp(arg2[0], arg2);      // arg2 실행

            perror("Child Execvp Failed\n");
            exit(1);
        }
    }
}


// 연산자가 없는 일반 명령어 실행 함수
void execute(int back, char *argv[]) {
    int status;
    pid_t pid;

    pid = fork();
```

```
    if(pid < 0) {
        perror("Fork Failed\n");
    }

    // 부모일 경우
    else if(pid1 > 0) {
        if(!back) pid1 = wait(&status);   // foreground process


        else {                            // background process
              printf("[1] %d\n", getpid());
              waitpid(pid1, &status, WNOHANG);
        }
    }

    else execvp(argv[0], argv);
}


// 명령어가 비어있는지 확인하는 함수
int checkCmd(char buf[]) {
    if(strcmp(buf, "") && strcmp(buf, "\t") && strcmp(buf, " "))
        return 0;
    return 1;
}


int main() {
    char cmd[256];          // 명령어 입력받을 배열
    char* args[MAXARG];     // 명령어 임시 저장
    char* arg[MAXARG];      // 연산자 명령어가 아닌 명령어 저장
    char* s;                // 구분자로 나눈 문자열
    char* saveptr;          // 다음 처리를 위한 위치를 저장하는 포인터
    pid_t pid;
    int status;             // 자식 프로세스 상태
    int num;
    int m;                  // <, >, |, & 연산자 명령어 표시할 번호
    int mark;               // 연산자 명령어가 있는 위치

    char *pipe_before[MAXARG];  //pipe 앞부분 명령어
    char *pipe_after[MAXARG];   //pipe 뒷부분 명령어
    char input[20] = "";        //redirection input 할 명령어
    char output[20] = "";       //redirection output 할 명령어
    int back = 0;               //backgroud process(&) 유무

    while(1) {
        printf("osh> ");
```

```
        fflush(stdout);          // printf 출력
        gets(cmd);               // 명령어 입력받음

        if(cmd != NULL && !checkCmd(cmd)) {
            args[0] = cmd;       // args 배열에 cmd 저장
            args[1] = (char *)0;

            // exit 입력 시 쉘 종료
            if(!strcmp(args[0], "exit")) {
                printf("Bye\n");
                exit(0);
            }

            num = 0;
            mark = 0;
            m = 0;
            back = 0;
            // args[0]에서 띄어쓰기, \t, \n 가 나오기 전까지의 문자열
            // saveptr 에 다음 위치 저장
            s = strtok_r(args[0], " \t\n", &saveptr);

            // s가 null이 아닐 동안 실행
            while(s) {
                //s와 문자열 비교
                if(!strcmp(s, "|")) {
                    m = 1;
                    mark = num;    // pipe의 위치를 mark에 저장

                    // pipe 앞의 명령어를 pipe_before 배열에 저장
                    for (int i = 0; i < mark; i++) {
                        pipe_before[i] = arg[i];
                    }
                    pipe_before[mark] = (char *)0;    // 명령어 뒤에 null 처리
                }

                else if(!strcmp(s, "<")) {
                    m = 2;
                    mark = num;    // "<"의 위치를 mark에 저장
                }

                else if(!strcmp(s, ">")) {
                    m = 3;
                }

                else if(!strcmp(s, "&")) {
                    back = 1;      // "&"가 있을 경우 back = 1;
                }
                else {
```

```
                    if(m < 2) {
                        // s가 연산자 기호가 아닌 경우 arg 배열에 저장
                        arg[num] = s;
                        num++;
                    }

                    else if(m == 2) {
                        // 앞에 < 를 만난 경우 input에 s 저장
                        strcpy(input, s);
                    }

                    // 앞에 > 를 만난 경우 output에 s 저장
                    else strcpy(output, s);
                }

                // 이후 띄어쓰기, \t, \n 가 나오기 전까지의 문자열
                s = strtok_r(NULL, " \t\n", &saveptr);
            }

            arg[num] = (char *)0;

            if(m == 0) {            // 아무런 연산자 명령어가 없는 경우
                execute(back, arg);
            }

            else if(m == 1) {       // pipe 명령어가 있는 경우
                // pipe_after에 pipe 뒷부분 명령어 저장
                for(int i = 0; i < num - mark; i++) {
                    pipe_after[i] = arg[i+mark];
                }

                pipe_after[num-mark] = (char *)0;   // 명령어 뒤에 null 처리
                pipeAct(back, pipe_before, pipe_after);
            }

            // Input Redirection 명령어가 있는 경우
            else if(m == 2) redir_in(back, arg, input);

            // Output Redirection 명령어가 있는 경우
            else if(m == 3) redir_out(back, arg, output);
        }
    }
}
```

3. 컴파일 과정


yewon@yewon-VirtualBox: ~/proj
File Edit View Search Terminal Help
yewon@yewon-VirtualBox:~$ cd proj
yewon@yewon-VirtualBox:~/proj$ ls
simpleshell.c
yewon@yewon-VirtualBox:~/proj$ gcc -o simpleshell simpleshell.c
simpleshell.c: In function ‘redir_in’:
simpleshell.c:73:13: warning: implicit declaration of function ‘exit’ [-Wimplicit-function-declaration]
exit(2);
^~~~
simpleshell.c:73:13: warning: incompatible implicit declaration of built-in function ‘exit’
simpleshell.c:73:13: note: include ‘<stdlib.h>’ or provide a declaration of ‘exit’
simpleshell.c: In function ‘pipeAct’:
simpleshell.c:171:9: warning: incompatible implicit declaration of built-in function ‘exit’
exit(1);
^~~~
simpleshell.c:171:9: note: include ‘<stdlib.h>’ or provide a declaration of ‘exit’
simpleshell.c:211:13: warning: incompatible implicit declaration of built-in function ‘exit’
exit(1);
^~~~
simpleshell.c:211:13: note: include ‘<stdlib.h>’ or provide a declaration of ‘exit’
simpleshell.c:223:13: warning: incompatible implicit declaration of built-in function ‘exit’
exit(1);
^~~~
simpleshell.c:223:13: note: include ‘<stdlib.h>’ or provide a declaration of ‘exit’
simpleshell.c:241:13: warning: incompatible implicit declaration of built-in function ‘exit’
exit(1);
^~~~
simpleshell.c:241:13: note: include ‘<stdlib.h>’ or provide a declaration of ‘exit’
simpleshell.c:261:13: warning: incompatible implicit declaration of built-in function ‘exit’
exit(1);
^~~~
simpleshell.c:261:13: note: include ‘<stdlib.h>’ or provide a declaration of ‘exit’
simpleshell.c: In function ‘main’:
simpleshell.c:383:9: warning: implicit declaration of function ‘gets’; did you mean ‘fgets’? [-Wimplicit-function-declaration]
gets(cmd);
// 명령어 입력받음
^~~~
fgets
simpleshell.c:401:17: warning: incompatible implicit declaration of built-in function ‘exit’
exit(0);
^~~~
simpleshell.c:401:17: note: include ‘<stdlib.h>’ or provide a declaration of ‘exit’
/tmp/ccZeHz3n.o: In function `main':
simpleshell.c:(.text+0x53d): warning: the `gets' function is dangerous and should not be used.
yewon@yewon-VirtualBox:~/proj$

4. 여러가지 명령어 실행 화면

(1) 명령어 + 옵션

```
yewon@yewon-VirtualBox:~/proj$ ./simpleshell
osh> ls -al
total 68
drwxr-xr-x  2 yewon yewon  4096  4월  1 18:21 .
drwxr-xr-x 20 yewon yewon  4096  4월  1 18:21 ..
-rwxr-xr-x  1 yewon yewon 13400  4월  1 17:44 simpleshell
-rw-r--r--  1 yewon yewon  8892  4월  1 17:41 simpleshell.c
-rwxr-xr-x  1 yewon yewon  8352  4월  1 17:53 test_printf
-rw-r--r--  1 yewon yewon   119  4월  1 17:52 test_printf.c
-rwxr-xr-x  1 yewon yewon  8448  4월  1 17:54 test_scanf
-rw-r--r--  1 yewon yewon   135  4월  1 17:54 test_scanf.c
```

▲ ls -al 명령어를 입력한 결과

```
osh> ls -s
total 64
 4 out.txt      12 simpleshell.c   4 test_printf.c   4 test_scanf.c
16 simpleshell  12 test_printf    12 test_scanf
```

▲ ls -s 명령어를 입력한 결과

(2) 명령어 + 옵션 &

```
yewon@yewon-VirtualBox:~/proj$ ./simpleshell
osh> ls -al &
[1] 1885
osh> total 68
drwxr-xr-x  2 yewon yewon  4096  4월  1 18:29 .
drwxr-xr-x 20 yewon yewon  4096  4월  1 18:33 ..
-rwxr-xr-x  1 yewon yewon 13400  4월  1 17:44 simpleshell
-rw-r--r--  1 yewon yewon  8892  4월  1 17:41 simpleshell.c
-rwxr-xr-x  1 yewon yewon  8352  4월  1 17:53 test_printf
-rw-r--r--  1 yewon yewon   119  4월  1 17:52 test_printf.c
-rwxr-xr-x  1 yewon yewon  8448  4월  1 17:54 test_scanf
-rw-r--r--  1 yewon yewon   135  4월  1 17:54 test_scanf.c
```

▲ ls -al 을 백그라운드에서 실행시킨 결과

(3) 명령어 + 옵션 > 파일명

```
osh> ls -al
osh> total 68
drwxr-xr-x  2 yewon yewon  4096  4월  1 18:29 .
drwxr-xr-x 20 yewon yewon  4096  4월  1 18:34 ..
-rwxr-xr-x  1 yewon yewon 13400  4월  1 17:44 simpleshell
-rw-r--r--  1 yewon yewon  8892  4월  1 17:41 simpleshell.c
-rwxr-xr-x  1 yewon yewon  8352  4월  1 17:53 test_printf
-rw-r--r--  1 yewon yewon   119  4월  1 17:52 test_printf.c
-rwxr-xr-x  1 yewon yewon  8448  4월  1 17:54 test_scanf
-rw-r--r--  1 yewon yewon   135  4월  1 17:54 test_scanf.c

osh> ls -al > out.txt
osh> ls -al
osh> total 72
drwxr-xr-x  2 yewon yewon  4096  4월  1 18:36 .
drwxr-xr-x 20 yewon yewon  4096  4월  1 18:34 ..
-rw-------  1 yewon yewon   521  4월  1 18:36 out.txt
-rwxr-xr-x  1 yewon yewon 13400  4월  1 17:44 simpleshell
-rw-r--r--  1 yewon yewon  8892  4월  1 17:41 simpleshell.c
-rwxr-xr-x  1 yewon yewon  8352  4월  1 17:53 test_printf
-rw-r--r--  1 yewon yewon   119  4월  1 17:52 test_printf.c
-rwxr-xr-x  1 yewon yewon  8448  4월  1 17:54 test_scanf
-rw-r--r--  1 yewon yewon   135  4월  1 17:54 test_scanf.c

osh> cat out.txt
osh> total 68
drwxr-xr-x  2 yewon yewon  4096  4월  1 18:36 .
drwxr-xr-x 20 yewon yewon  4096  4월  1 18:34 ..
-rw-------  1 yewon yewon     0  4월  1 18:36 out.txt
-rwxr-xr-x  1 yewon yewon 13400  4월  1 17:44 simpleshell
-rw-r--r--  1 yewon yewon  8892  4월  1 17:41 simpleshell.c
-rwxr-xr-x  1 yewon yewon  8352  4월  1 17:53 test_printf
-rw-r--r--  1 yewon yewon   119  4월  1 17:52 test_printf.c
-rwxr-xr-x  1 yewon yewon  8448  4월  1 17:54 test_scanf
-rw-r--r--  1 yewon yewon   135  4월  1 17:54 test_scanf.c
```

▲ ls -al > out.txt를 실행한 결과

ls -al 에 out.txt가 추가된 것과 out.txt에 ls -al의 결과가 저장된 것을 볼 수 있다.

(4) 명령어 + 옵션 > 파일명 &

```
osh> ls -al
osh> total 72
drwxr-xr-x  2 yewon yewon  4096  4월   1 18:45 .
drwxr-xr-x 20 yewon yewon  4096  4월   1 18:34 ..
-rw-------  1 yewon yewon   521  4월   1 18:36 out.txt
-rwxr-xr-x  1 yewon yewon 13400  4월   1 17:44 simpleshell
-rw-r--r--  1 yewon yewon  8892  4월   1 17:41 simpleshell.c
-rwxr-xr-x  1 yewon yewon  8352  4월   1 17:53 test_printf
-rw-r--r--  1 yewon yewon   119  4월   1 17:52 test_printf.c
-rwxr-xr-x  1 yewon yewon  8448  4월   1 17:54 test_scanf
-rw-r--r--  1 yewon yewon   135  4월   1 17:54 test_scanf.c

osh> ls -al > out2.txt &
[1] 1989
osh> ls -al
osh> total 76
drwxr-xr-x  2 yewon yewon  4096  4월   1 18:47 .
drwxr-xr-x 20 yewon yewon  4096  4월   1 18:34 ..
-rw-------  1 yewon yewon   577  4월   1 18:47 out2.txt
-rw-------  1 yewon yewon   521  4월   1 18:36 out.txt
-rwxr-xr-x  1 yewon yewon 13400  4월   1 17:44 simpleshell
-rw-r--r--  1 yewon yewon  8892  4월   1 17:41 simpleshell.c
-rwxr-xr-x  1 yewon yewon  8352  4월   1 17:53 test_printf
-rw-r--r--  1 yewon yewon   119  4월   1 17:52 test_printf.c
-rwxr-xr-x  1 yewon yewon  8448  4월   1 17:54 test_scanf
-rw-r--r--  1 yewon yewon   135  4월   1 17:54 test_scanf.c

osh> cat out2.txt
osh> total 72
drwxr-xr-x  2 yewon yewon  4096  4월   1 18:47 .
drwxr-xr-x 20 yewon yewon  4096  4월   1 18:34 ..
-rw-------  1 yewon yewon     0  4월   1 18:47 out2.txt
-rw-------  1 yewon yewon   521  4월   1 18:36 out.txt
-rwxr-xr-x  1 yewon yewon 13400  4월   1 17:44 simpleshell
-rw-r--r--  1 yewon yewon  8892  4월   1 17:41 simpleshell.c
-rwxr-xr-x  1 yewon yewon  8352  4월   1 17:53 test_printf
-rw-r--r--  1 yewon yewon   119  4월   1 17:52 test_printf.c
-rwxr-xr-x  1 yewon yewon  8448  4월   1 17:54 test_scanf
-rw-r--r--  1 yewon yewon   135  4월   1 17:54 test_scanf.c
```

▲ ls -al > out2.txt를 백그라운드에서 실행한 결과

ls -al 에 out2.txt가 추가된 것과 out2.txt에 ls -al의 결과가 저장된 것을 볼 수 있다.

(5) 명령어 + 옵션 < 파일명

```
osh> cat out.txt
total 68
drwxr-xr-x  2 yewon yewon  4096  4월  1 18:36 .
drwxr-xr-x 20 yewon yewon  4096  4월  1 18:34 ..
-rw-------  1 yewon yewon     0  4월  1 18:36 out.txt
-rwxr-xr-x  1 yewon yewon 13400  4월  1 17:44 simpleshell
-rw-r--r--  1 yewon yewon  8892  4월  1 17:41 simpleshell.c
-rwxr-xr-x  1 yewon yewon  8352  4월  1 17:53 test_printf
-rw-r--r--  1 yewon yewon   119  4월  1 17:52 test_printf.c
-rwxr-xr-x  1 yewon yewon  8448  4월  1 17:54 test_scanf
-rw-r--r--  1 yewon yewon   135  4월  1 17:54 test_scanf.c
osh> sort -r < out.txt
osh> total 68
-rwxr-xr-x  1 yewon yewon  8448  4월  1 17:54 test_scanf
-rwxr-xr-x  1 yewon yewon  8352  4월  1 17:53 test_printf
-rwxr-xr-x  1 yewon yewon 13400  4월  1 17:44 simpleshell
-rw-r--r--  1 yewon yewon  8892  4월  1 17:41 simpleshell.c
-rw-r--r--  1 yewon yewon   135  4월  1 17:54 test_scanf.c
-rw-r--r--  1 yewon yewon   119  4월  1 17:52 test_printf.c
-rw-------  1 yewon yewon     0  4월  1 18:36 out.txt
drwxr-xr-x  2 yewon yewon  4096  4월  1 18:36 .
drwxr-xr-x 20 yewon yewon  4096  4월  1 18:34 ..
```

▲ sort -r < out.txt 를 실행한 결과

out.txt 가 sort -r 명령어에 의해 내림차순으로 정렬된 것을 볼 수 있다.

(6) 명령어 + 옵션 < 파일명 &

```
osh> cat out.txt
total 68
drwxr-xr-x  2 yewon yewon  4096  4월  1 18:36 .
drwxr-xr-x 20 yewon yewon  4096  4월  1 18:34 ..
-rw-------  1 yewon yewon     0  4월  1 18:36 out.txt
-rwxr-xr-x  1 yewon yewon 13400  4월  1 17:44 simpleshell
-rw-r--r--  1 yewon yewon  8892  4월  1 17:41 simpleshell.c
-rwxr-xr-x  1 yewon yewon  8352  4월  1 17:53 test_printf
-rw-r--r--  1 yewon yewon   119  4월  1 17:52 test_printf.c
-rwxr-xr-x  1 yewon yewon  8448  4월  1 17:54 test_scanf
-rw-r--r--  1 yewon yewon   135  4월  1 17:54 test_scanf.c
osh> sort -r < out.txt &
[1] 1989
osh> total 68
-rwxr-xr-x  1 yewon yewon  8448  4월  1 17:54 test_scanf
-rwxr-xr-x  1 yewon yewon  8352  4월  1 17:53 test_printf
-rwxr-xr-x  1 yewon yewon 13400  4월  1 17:44 simpleshell
-rw-r--r--  1 yewon yewon  8892  4월  1 17:41 simpleshell.c
-rw-r--r--  1 yewon yewon   135  4월  1 17:54 test_scanf.c
-rw-r--r--  1 yewon yewon   119  4월  1 17:52 test_printf.c
-rw-------  1 yewon yewon     0  4월  1 18:36 out.txt
drwxr-xr-x  2 yewon yewon  4096  4월  1 18:36 .
drwxr-xr-x 20 yewon yewon  4096  4월  1 18:34 ..
```

▲ sort -r < out.txt 를 백그라운드에서 실행한 결과

out.txt 가 sort -r 명령어에 의해 내림차순으로 정렬된 것을 볼 수 있다.

(7) 명령어+옵션 | 명령어+옵션

```
yewon@yewon-VirtualBox:~/proj$ ps -ef
UID          PID  PPID  C STIME TTY          TIME CMD
root           1     0  4 19:48 ?        00:00:08 /sbin/init splash
root           2     0  0 19:48 ?        00:00:00 [kthreadd]
root           3     2  0 19:48 ?        00:00:00 [rcu_gp]
root           4     2  0 19:48 ?        00:00:00 [rcu_par_gp]
root           5     2  0 19:48 ?        00:00:00 [kworker/0:0-cgr]
root           6     2  0 19:48 ?        00:00:00 [kworker/0:0H-kb]
root           7     2  0 19:48 ?        00:00:00 [kworker/0:1-eve]
root           8     2  0 19:48 ?        00:00:00 [kworker/u2:0-ev]
root           9     2  0 19:48 ?        00:00:00 [mm_percpu_wq]
root          10     2  0 19:48 ?        00:00:00 [ksoftirqd/0]
root          11     2  0 19:48 ?        00:00:00 [rcu_sched]
root          12     2  0 19:48 ?        00:00:00 [migration/0]
root          13     2  0 19:48 ?        00:00:00 [idle_inject/0]
root          14     2  0 19:48 ?        00:00:00 [cpuhp/0]
root          15     2  0 19:48 ?        00:00:00 [kdevtmpfs]
root          16     2  0 19:48 ?        00:00:00 [netns]
root          17     2  0 19:48 ?        00:00:00 [rcu_tasks_kthre]
root          18     2  0 19:48 ?        00:00:00 [kauditd]
root          19     2  0 19:48 ?        00:00:00 [khungtaskd]
root          20     2  0 19:48 ?        00:00:00 [oom_reaper]
root          21     2  0 19:48 ?        00:00:00 [writeback]
root          22     2  0 19:48 ?        00:00:00 [kcompactd0]
root          23     2  0 19:48 ?        00:00:00 [ksmd]
root          24     2  0 19:48 ?        00:00:00 [khugepaged]
root          26     2  0 19:48 ?        00:00:00 [kworker/u2:1-ev]
root         116     2  0 19:48 ?        00:00:00 [kintegrityd]
root         117     2  0 19:48 ?        00:00:00 [kblockd]
root         118     2  0 19:48 ?        00:00:00 [blkcg_punt_bio]
root         119     2  0 19:48 ?        00:00:00 [tpm_dev_wq]
root         120     2  0 19:48 ?        00:00:00 [ata_sff]
root         121     2  0 19:48 ?        00:00:00 [md]
root         122     2  0 19:48 ?        00:00:00 [edac-poller]
root         123     2  0 19:48 ?        00:00:00 [devfreq_wq]
root         124     2  0 19:48 ?        00:00:00 [watchdogd]
root         128     2  0 19:48 ?        00:00:00 [kswapd0]
```

⋮

```
yewon      1747  1373  0 19:48 tty2     00:00:00 /usr/lib/gnome-disk-utility/gsd-
yewon      1774  1543  0 19:48 ?        00:00:00 /usr/lib/gvfs/gvfsd-trash --spaw
yewon      1803  1346  0 19:48 ?        00:00:00 /usr/lib/evolution/evolution-cal
yewon      1812  1803  0 19:48 ?        00:00:00 /usr/lib/evolution/evolution-cal
yewon      1814  1346  0 19:48 ?        00:00:00 /usr/lib/dconf/dconf-service
yewon      1827  1346  0 19:48 ?        00:00:00 /usr/lib/evolution/evolution-add
yewon      1836  1827  0 19:48 ?        00:00:00 /usr/lib/evolution/evolution-add
yewon      1850  1569  0 19:48 tty2     00:00:00 /usr/lib/ibus/ibus-engine-simple
yewon      1878  1346  0 19:49 ?        00:00:02 /usr/lib/gnome-terminal/gnome-te
yewon      1887  1878  0 19:49 pts/0    00:00:00 bash
yewon      1902  1373  0 19:49 tty2     00:00:00 update-notifier
yewon      1904  1373  0 19:49 tty2     00:00:02 /usr/bin/gnome-software --gappli
root       1928     1  0 19:49 ?        00:00:00 /usr/lib/fwupd/fwupd
yewon      1996  1887  0 19:55 pts/0    00:00:00 ./simpleshell
yewon      1997  1996  0 19:55 pts/0    00:00:00 ps -ef
```

▲ ps -ef 를 실행한 결과

```
osh> ps -ef | grep -b bash
17519:yewon      1887  1878  0 19:49 pts/0    00:00:00 bash
17862:yewon      1999  1996  0 19:55 pts/0    00:00:00 grep -b bash
```

▲ ps -ef | grep -b bash 를 실행한 결과

(8) 명령어+옵션 | 명령어+옵션 &

```
osh> ps -ef | grep -b bash &
[1] 2145
osh> 17303:yewon     1877  1868  0 18:33 pts/0    00:00:00 bash
17650:yewon     2082  1868  0 19:28 pts/1    00:00:00 bash
17831:yewon     2155  2145  0 19:44 pts/0    00:00:00 grep -b bash

osh>
```

▲ ps -ef | grep -b bash 를 백그라운드에서 실행한 결과

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

int main() {
        printf("Hello\n");
        sleep(20);
        return 0;
}
```

▲ test_printf.c 코드

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

int main() {
        char buf[100];
        scanf("%s", buf);
        printf("%s\n", buf);
        sleep(20);
        return 0;
}
```

▲ test_scanf.c 코드

```
yewon@yewon-VirtualBox:~/proj$ gcc -o test_printf test_printf.c
yewon@yewon-VirtualBox:~/proj$ gcc -o test_scanf test_scanf.c
```

▲ test_printf.c 와 test_scanf.c 를 컴파일

```
osh> ps aux | grep test_
yewon     1894  0.0  0.0  22828  1056 pts/0    S+   20:08   0:00 grep test_
osh> ./test_printf | ./test_scanf &
[1] 1893
osh> ps aux | grep test_
yewon     1900  0.0  0.0   4512   804 pts/0    S+   20:08   0:00 ./test_scanf
yewon     1901  0.0  0.0   4512   804 pts/0    S+   20:08   0:00 ./test_printf
yewon     1902  0.0  0.0  22828  1040 pts/0    S+   20:08   0:00 grep test_
osh> Hello

osh> ps aux | grep test_
osh> yewon     1994  0.0  0.0  22828  1092 pts/0    S+   20:09   0:00 grep test_

osh>
```

▲ ./test_printf | ./test_scanf 를 백그라운드에서 실행한 결과

실행 전에는 ./test_scanf 와 ./test_printf 프로세스가 없었지만 실행한 후 프로세스가 동작 중인 것을 볼 수 있다. 'Hello' 를 출력하고 프로세스 또한 종료된 것을 볼 수 있다.