Operating Systems

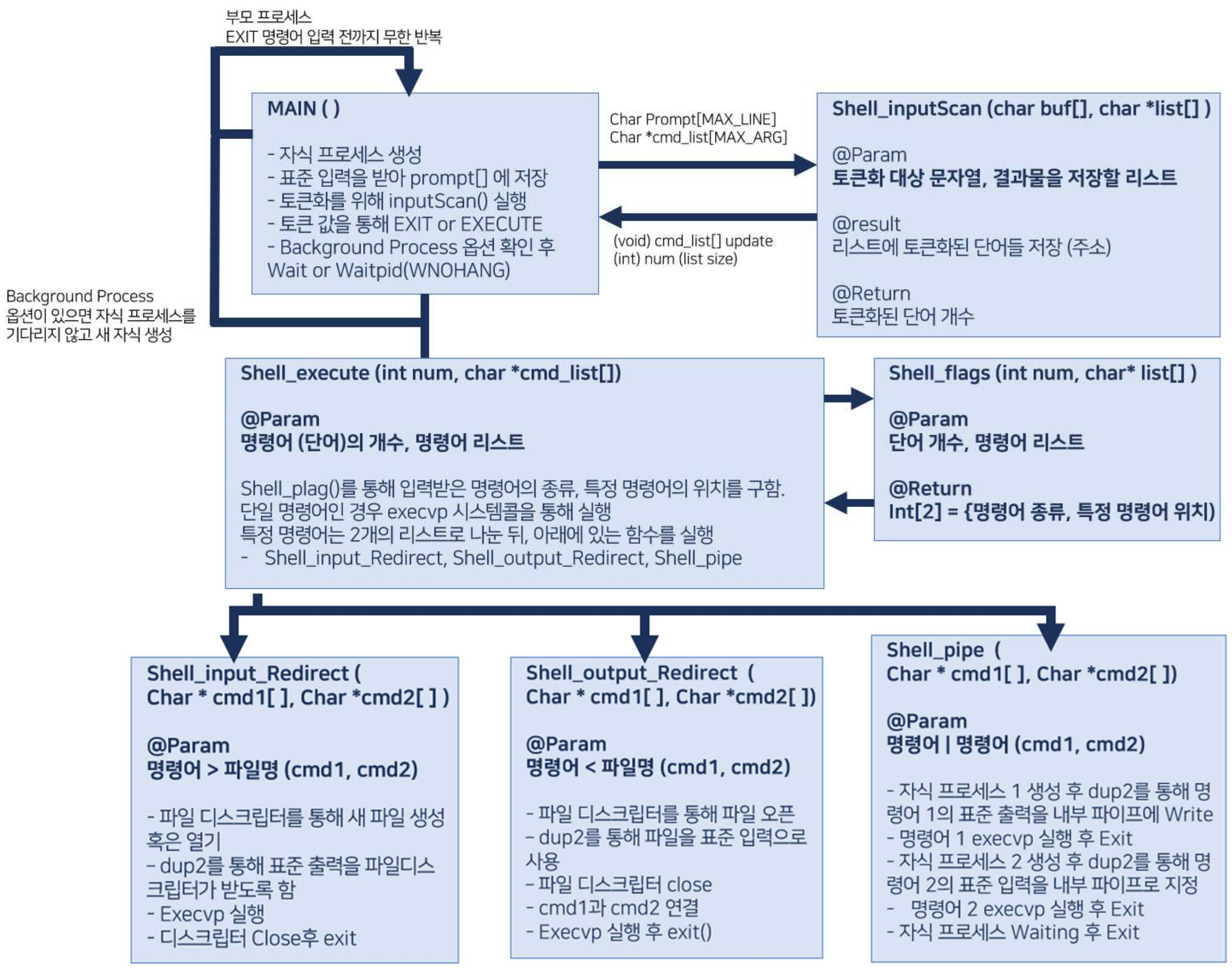
Project #1

COM2005 – 24081

소프트웨어학부 컴퓨터전공

2018045214 최준희

0. Simple Shell Algorithm

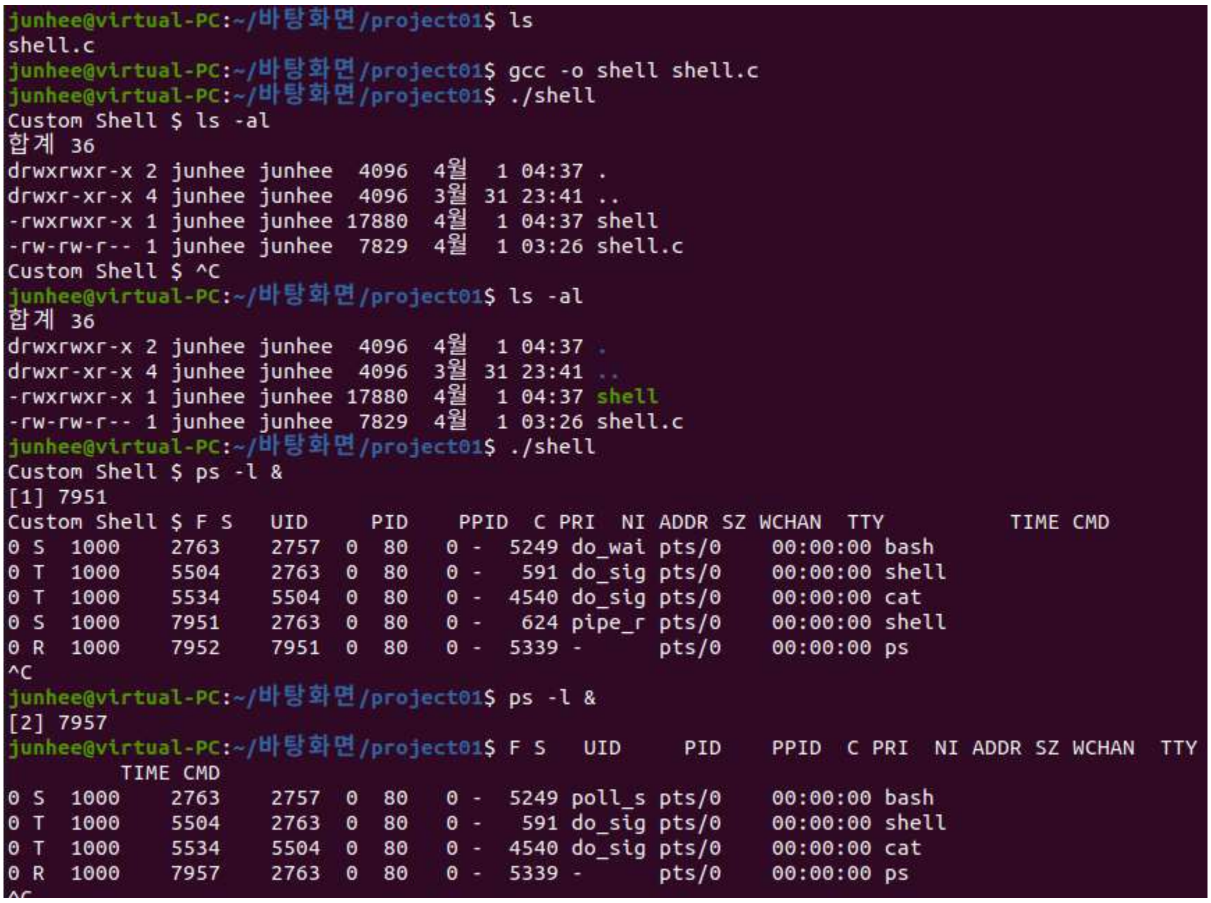

부모 프로세스
EXIT 명령어 입력 전까지 무한 반복
MAIN ( )
- 자식 프로세스 생성
- 표준 입력을 받아 prompt[] 에 저장
- 토큰화를 위해 inputScan() 실행
- 토큰 값을 통해 EXIT or EXECUTE
- Background Process 옵션 확인 후 Wait or Waitpid(WNOHANG)
Char Prompt[MAX_LINE]
Char *cmd_list[MAX_ARG]
(void) cmd_list[] update
(int) num (list size)
Shell_inputScan (char buf[], char *list[])
@Param
토큰화 대상 문자열, 결과물을 저장할 리스트
@result
리스트에 토큰화된 단어들 저장 (주소)
@Return
토큰화된 단어 개수
Background Process
옵션이 있으면 자식 프로세스를 기다리지 않고 새 자식 생성
Shell_execute (int num, char *cmd_list[])
@Param
명령어 (단어)의 개수, 명령어 리스트
Shell_plag()를 통해 입력받은 명령어의 종류, 특정 명령어의 위치를 구함.
단일 명령어인 경우 execvp 시스템콜을 통해 실행
특정 명령어는 2개의 리스트로 나눈 뒤, 아래에 있는 함수를 실행
- Shell_input_Redirect, Shell_output_Redirect, Shell_pipe
Shell_flags (int num, char* list[])
@Param
단어 개수, 명령어 리스트
@Return
Int[2] = {명령어 종류, 특정 명령어 위치}
Shell_input_Redirect (Char * cmd1[ ], Char *cmd2[ ])
@Param
명령어 > 파일명 (cmd1, cmd2)
- 파일 디스크립터를 통해 새 파일 생성 혹은 열기
- dup2를 통해 표준 출력을 파일디스크립터가 받도록 함
- Execvp 실행
- 디스크립터 Close후 exit
Shell_output_Redirect (Char * cmd1[ ], Char *cmd2[ ])
@Param
명령어 < 파일명 (cmd1, cmd2)
- 파일 디스크립터를 통해 파일 오픈
- dup2를 통해 파일을 표준 입력으로 사용
- 파일 디스크립터 close
- cmd1과 cmd2 연결
- Execvp 실행 후 exit()
Shell_pipe (Char * cmd1[ ], Char *cmd2[ ])
@Param
명령어 | 명령어 (cmd1, cmd2)
- 자식 프로세스 1 생성 후 dup2를 통해 명령어 1의 표준 출력을 내부 파이프에 Write
- 명령어 1 execvp 실행 후 Exit
- 자식 프로세스 2 생성 후 dup2를 통해 명령어 2의 표준 입력을 내부 파이프로 지정
- 명령어 2 execvp 실행 후 Exit
- 자식 프로세스 Waiting 후 Exit

프로그램의 흐름은 위 사진과 같으며 세세한 부분은 2. Code의 주석에서 확인할 수 있다. 전반적으로 부모 프로세스를 통해 프로그램이 반복되며, 한 줄의 명령어를 위해 자식 프로세스가 생성, inputScan과 execute를 맡게 된다. 이 후 입력 받은 명령어의 특성에 따라 바로 execvp()가 실행되는 경우도 있고, 약간의 전처리를 거쳐 input, output_Redirect(), shell_pipe() 를 실행하는 경우도 있다.

여기서 유일하게 pipe()는 자식 프로세스의 자식 프로세스 (손자 프로세스) 를 생성, 명령어 1의 출력을 명령어 2에 넣고 나온 값을 프로세스가 받아 출력하는 구조이다.

1. Compile & Execute Result

1-1. 단일 명령어 조합



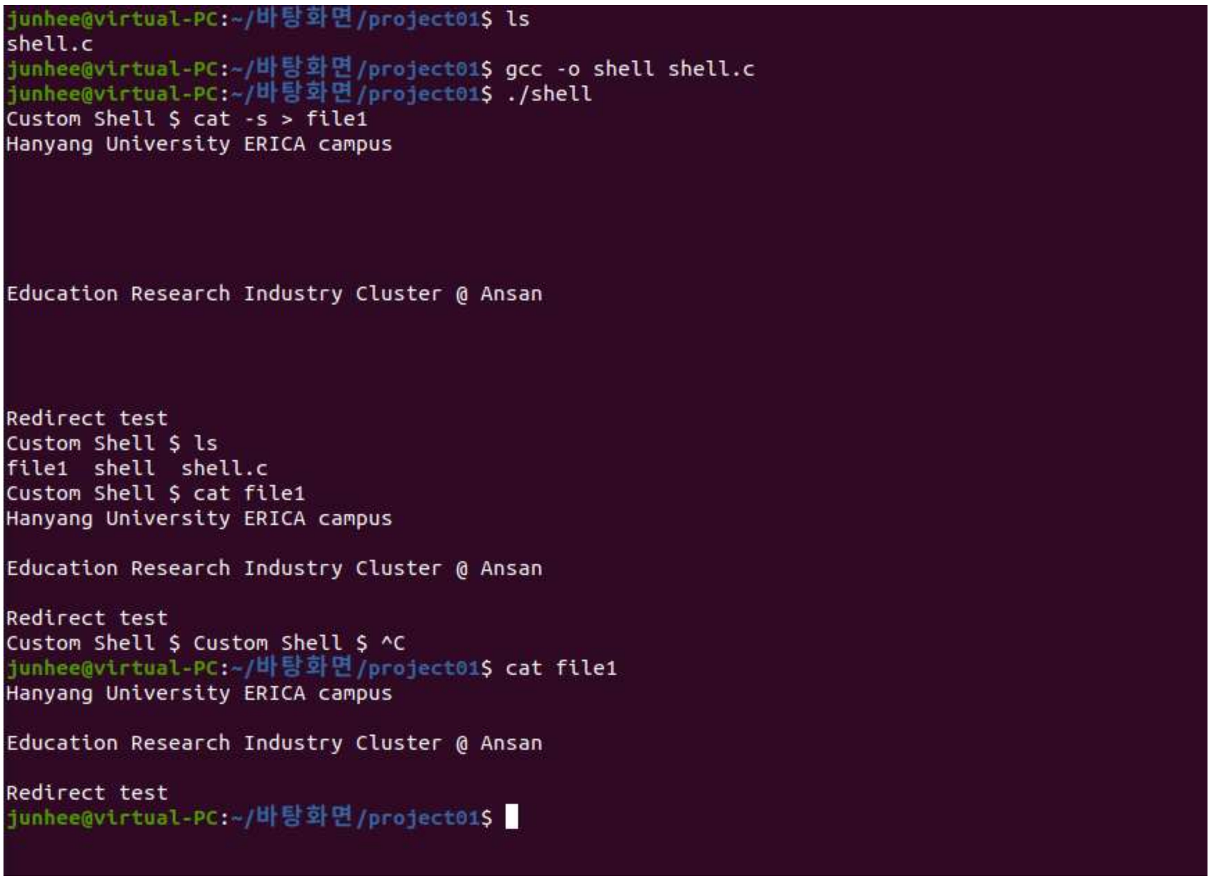
```
junhee@virtual-PC:~/바탕화면/project01$ ls
shell.c
junhee@virtual-PC:~/바탕화면/project01$ gcc -o shell shell.c
junhee@virtual-PC:~/바탕화면/project01$ ./shell
Custom Shell $ ls -al
합계 36
drwxrwxr-x 2 junhee junhee  4096  4월  1 04:37 .
drwxr-xr-x 4 junhee junhee  4096  3월 31 23:41 ..
-rwxrwxr-x 1 junhee junhee 17880  4월  1 04:37 shell
-rw-rw-r-- 1 junhee junhee  7829  4월  1 03:26 shell.c
Custom Shell $ ^C
junhee@virtual-PC:~/바탕화면/project01$ ls -al
합계 36
drwxrwxr-x 2 junhee junhee  4096  4월  1 04:37 .
drwxr-xr-x 4 junhee junhee  4096  3월 31 23:41 ..
-rwxrwxr-x 1 junhee junhee 17880  4월  1 04:37 shell
-rw-rw-r-- 1 junhee junhee  7829  4월  1 03:26 shell.c
junhee@virtual-PC:~/바탕화면/project01$ ./shell
Custom Shell $ ps -l &
[1] 7951
Custom Shell $ F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000    2763    2757  0  80   0 -  5249 do_wai pts/0    00:00:00 bash
0 T  1000    5504    2763  0  80   0 -   591 do_sig pts/0    00:00:00 shell
0 T  1000    5534    5504  0  80   0 -  4540 do_sig pts/0    00:00:00 cat
0 S  1000    7951    2763  0  80   0 -   624 pipe_r pts/0    00:00:00 shell
0 R  1000    7952    7951  0  80   0 -  5339 -      pts/0    00:00:00 ps
^C
junhee@virtual-PC:~/바탕화면/project01$ ps -l &
[2] 7957
junhee@virtual-PC:~/바탕화면/project01$ F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY
         TIME CMD
0 S  1000    2763    2757  0  80   0 -  5249 poll_s pts/0    00:00:00 bash
0 T  1000    5504    2763  0  80   0 -   591 do_sig pts/0    00:00:00 shell
0 T  1000    5534    5504  0  80   0 -  4540 do_sig pts/0    00:00:00 cat
0 R  1000    7957    2763  0  80   0 -  5339 -      pts/0    00:00:00 ps
```

사용한 명령어

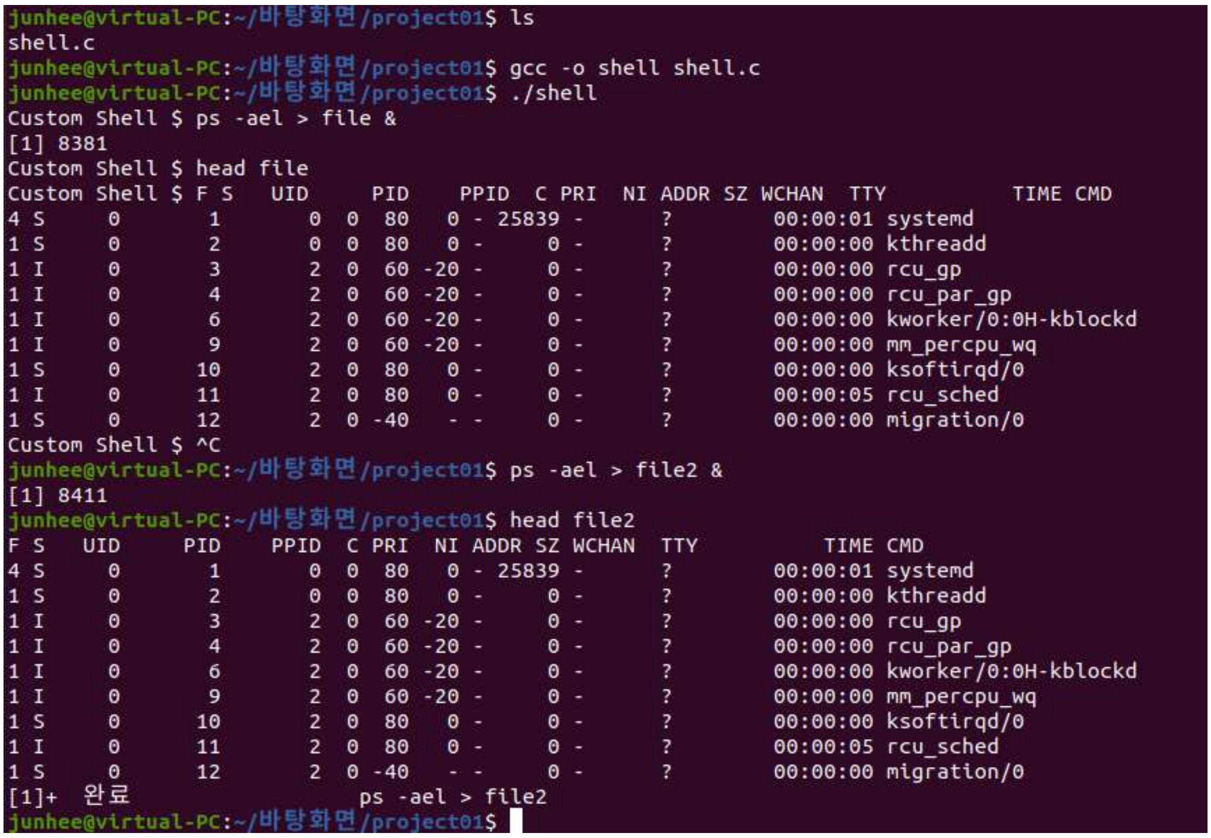
ls -al

ps -l &

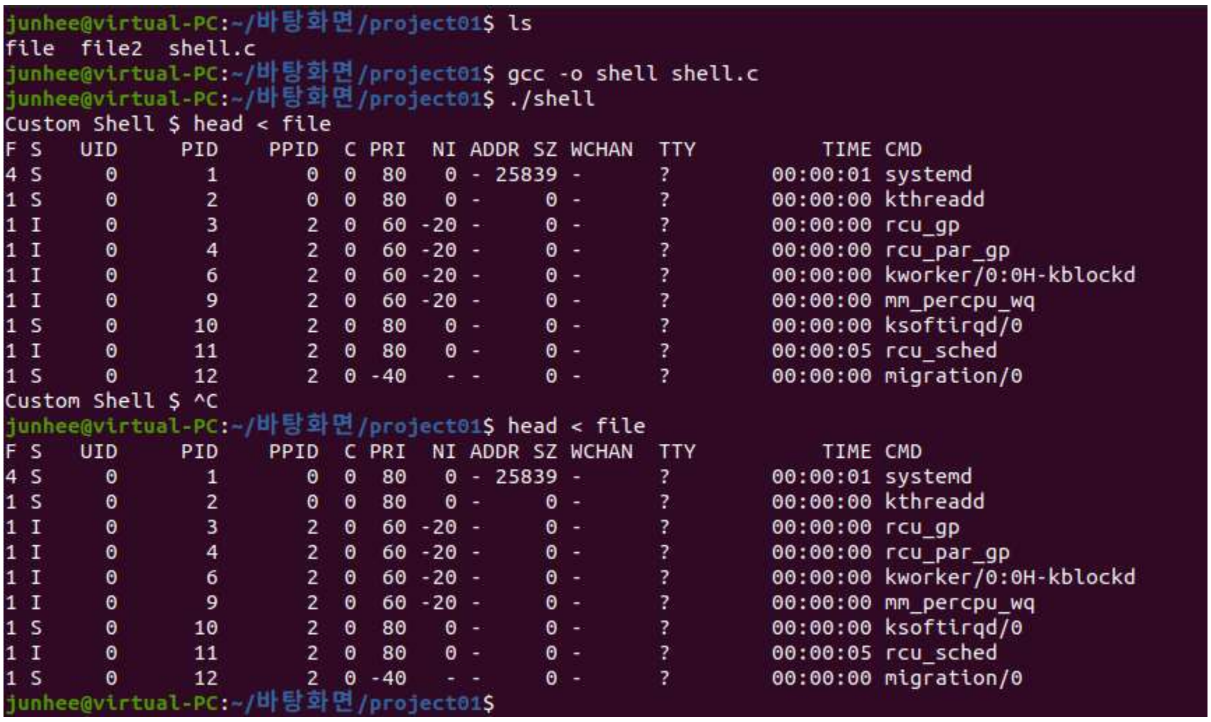
1-2. 리다이렉트 명령어 조합



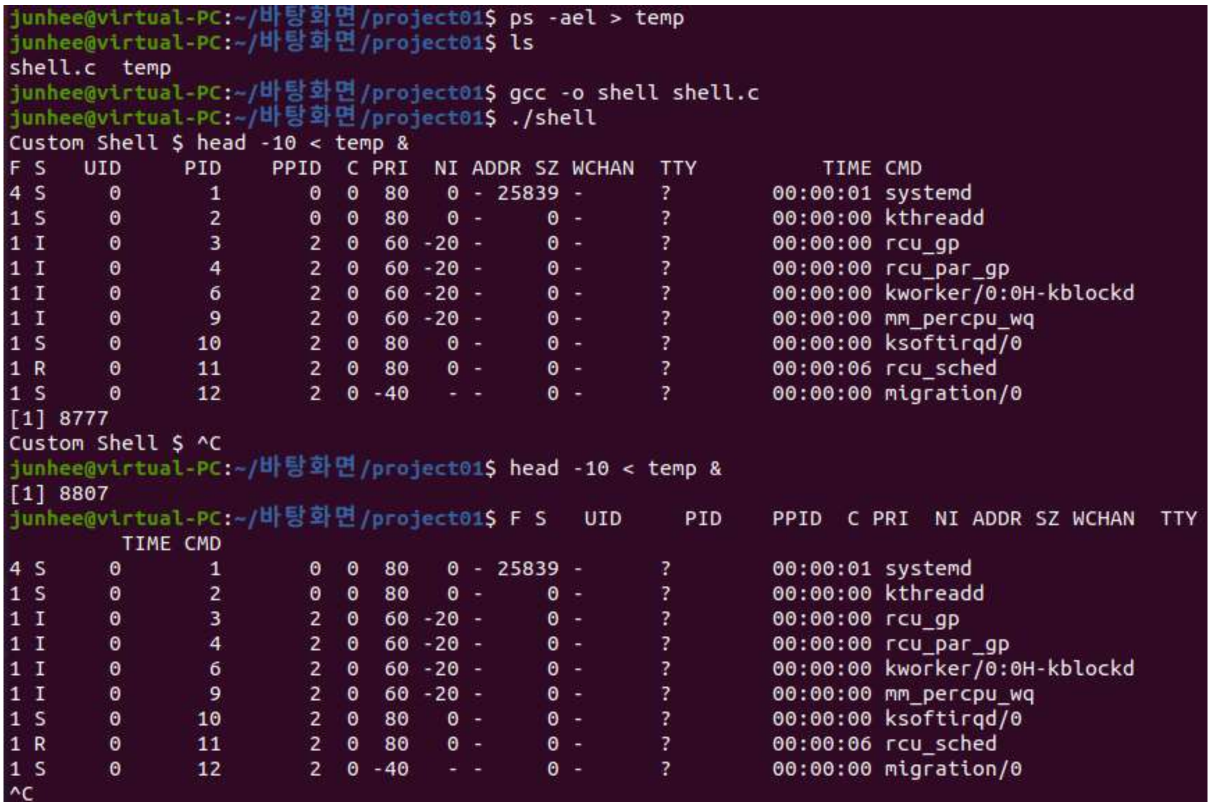
```
junhee@virtual-PC:~/바탕화면/project01$ ls
shell.c
junhee@virtual-PC:~/바탕화면/project01$ gcc -o shell shell.c
junhee@virtual-PC:~/바탕화면/project01$ ./shell
Custom Shell $ cat -s > file1
Hanyang University ERICA campus





Education Research Industry Cluster @ Ansan




Redirect test
Custom Shell $ ls
file1  shell  shell.c
Custom Shell $ cat file1
Hanyang University ERICA campus

Education Research Industry Cluster @ Ansan

Redirect test
Custom Shell $ Custom Shell $ ^C
junhee@virtual-PC:~/바탕화면/project01$ cat file1
Hanyang University ERICA campus

Education Research Industry Cluster @ Ansan

Redirect test
junhee@virtual-PC:~/바탕화면/project01$ 
```



```
junhee@virtual-PC:~/바탕화면/project01$ ls
shell.c
junhee@virtual-PC:~/바탕화면/project01$ gcc -o shell shell.c
junhee@virtual-PC:~/바탕화면/project01$ ./shell
Custom Shell $ ps -ael > file &
[1] 8381
Custom Shell $ head file
Custom Shell $ F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0       1       0  0  80   0 - 25839 -      ?        00:00:01 systemd
1 S     0       2       0  0  80   0 -     0 -      ?        00:00:00 kthreadd
1 I     0       3       2  0  60 -20 -     0 -      ?        00:00:00 rcu_gp
1 I     0       4       2  0  60 -20 -     0 -      ?        00:00:00 rcu_par_gp
1 I     0       6       2  0  60 -20 -     0 -      ?        00:00:00 kworker/0:0H-kblockd
1 I     0       9       2  0  60 -20 -     0 -      ?        00:00:00 mm_percpu_wq
1 S     0      10       2  0  80   0 -     0 -      ?        00:00:00 ksoftirqd/0
1 I     0      11       2  0  80   0 -     0 -      ?        00:00:05 rcu_sched
1 S     0      12       2  0 -40   - -     0 -      ?        00:00:00 migration/0
Custom Shell $ ^C
junhee@virtual-PC:~/바탕화면/project01$ ps -ael > file2 &
[1] 8411
junhee@virtual-PC:~/바탕화면/project01$ head file2
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0       1       0  0  80   0 - 25839 -      ?        00:00:01 systemd
1 S     0       2       0  0  80   0 -     0 -      ?        00:00:00 kthreadd
1 I     0       3       2  0  60 -20 -     0 -      ?        00:00:00 rcu_gp
1 I     0       4       2  0  60 -20 -     0 -      ?        00:00:00 rcu_par_gp
1 I     0       6       2  0  60 -20 -     0 -      ?        00:00:00 kworker/0:0H-kblockd
1 I     0       9       2  0  60 -20 -     0 -      ?        00:00:00 mm_percpu_wq
1 S     0      10       2  0  80   0 -     0 -      ?        00:00:00 ksoftirqd/0
1 I     0      11       2  0  80   0 -     0 -      ?        00:00:05 rcu_sched
1 S     0      12       2  0 -40   - -     0 -      ?        00:00:00 migration/0
[1]+  완료                  ps -ael > file2
junhee@virtual-PC:~/바탕화면/project01$ 
```



```
junhee@virtual-PC:~/바탕화면/project01$ ls
file  file2  shell.c
junhee@virtual-PC:~/바탕화면/project01$ gcc -o shell shell.c
junhee@virtual-PC:~/바탕화면/project01$ ./shell
Custom Shell $ head < file
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0       1       0  0  80   0 - 25839 -      ?        00:00:01 systemd
1 S     0       2       0  0  80   0 -     0 -      ?        00:00:00 kthreadd
1 I     0       3       2  0  60 -20 -     0 -      ?        00:00:00 rcu_gp
1 I     0       4       2  0  60 -20 -     0 -      ?        00:00:00 rcu_par_gp
1 I     0       6       2  0  60 -20 -     0 -      ?        00:00:00 kworker/0:0H-kblockd
1 I     0       9       2  0  60 -20 -     0 -      ?        00:00:00 mm_percpu_wq
1 S     0      10       2  0  80   0 -     0 -      ?        00:00:00 ksoftirqd/0
1 I     0      11       2  0  80   0 -     0 -      ?        00:00:05 rcu_sched
1 S     0      12       2  0 -40   - -     0 -      ?        00:00:00 migration/0
Custom Shell $ ^C
junhee@virtual-PC:~/바탕화면/project01$ head < file
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0       1       0  0  80   0 - 25839 -      ?        00:00:01 systemd
1 S     0       2       0  0  80   0 -     0 -      ?        00:00:00 kthreadd
1 I     0       3       2  0  60 -20 -     0 -      ?        00:00:00 rcu_gp
1 I     0       4       2  0  60 -20 -     0 -      ?        00:00:00 rcu_par_gp
1 I     0       6       2  0  60 -20 -     0 -      ?        00:00:00 kworker/0:0H-kblockd
1 I     0       9       2  0  60 -20 -     0 -      ?        00:00:00 mm_percpu_wq
1 S     0      10       2  0  80   0 -     0 -      ?        00:00:00 ksoftirqd/0
1 I     0      11       2  0  80   0 -     0 -      ?        00:00:05 rcu_sched
1 S     0      12       2  0 -40   - -     0 -      ?        00:00:00 migration/0
junhee@virtual-PC:~/바탕화면/project01$
```



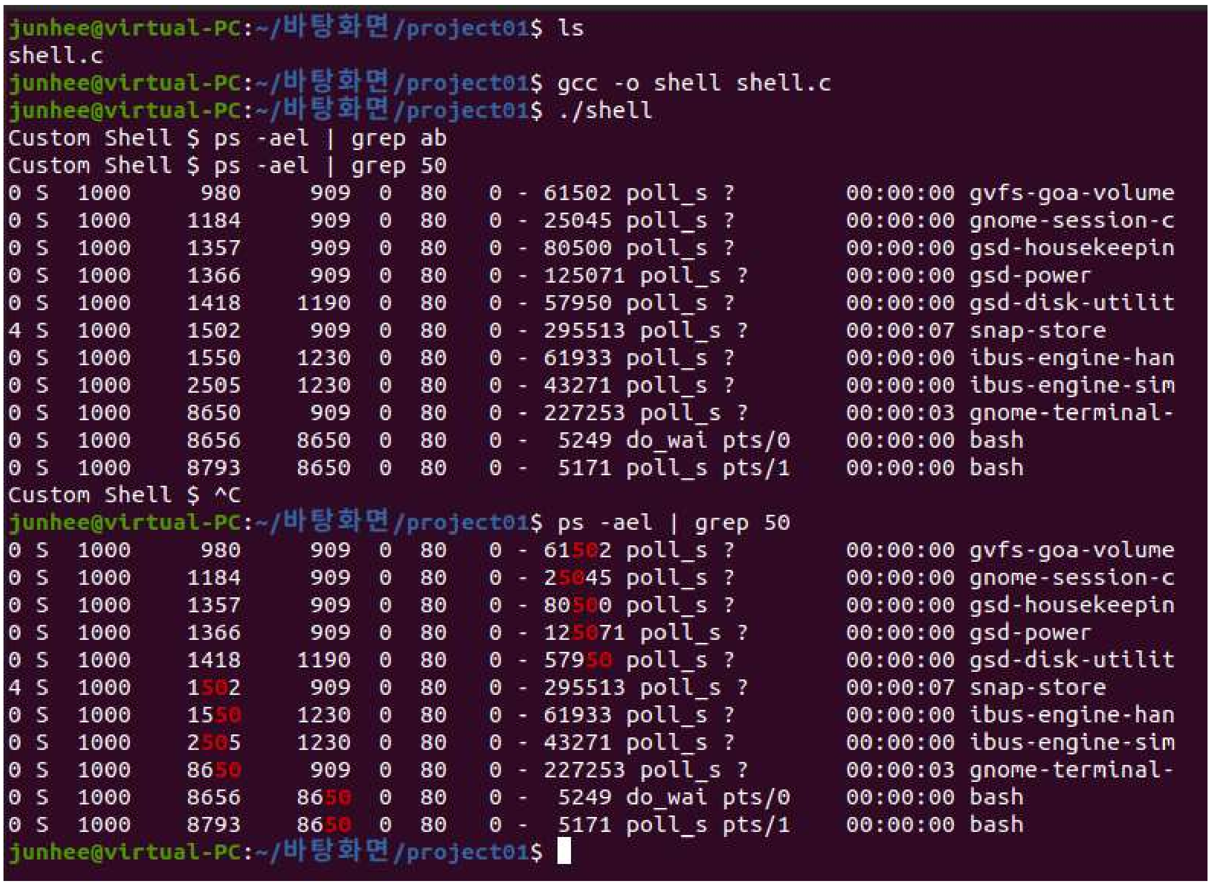
```
junhee@virtual-PC:~/바탕화면/project01$ ps -ael > temp
junhee@virtual-PC:~/바탕화면/project01$ ls
shell.c  temp
junhee@virtual-PC:~/바탕화면/project01$ gcc -o shell shell.c
junhee@virtual-PC:~/바탕화면/project01$ ./shell
Custom Shell $ head -10 < temp &
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0       1       0  0  80   0 - 25839 -      ?        00:00:01 systemd
1 S     0       2       0  0  80   0 -     0 -      ?        00:00:00 kthreadd
1 I     0       3       2  0  60 -20 -     0 -      ?        00:00:00 rcu_gp
1 I     0       4       2  0  60 -20 -     0 -      ?        00:00:00 rcu_par_gp
1 I     0       6       2  0  60 -20 -     0 -      ?        00:00:00 kworker/0:0H-kblockd
1 I     0       9       2  0  60 -20 -     0 -      ?        00:00:00 mm_percpu_wq
1 S     0      10       2  0  80   0 -     0 -      ?        00:00:00 ksoftirqd/0
1 R     0      11       2  0  80   0 -     0 -      ?        00:00:06 rcu_sched
1 S     0      12       2  0 -40   - -     0 -      ?        00:00:00 migration/0
[1] 8777
Custom Shell $ ^C
junhee@virtual-PC:~/바탕화면/project01$ head -10 < temp &
[1] 8807
junhee@virtual-PC:~/바탕화면/project01$ F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY
         TIME CMD
4 S     0       1       0  0  80   0 - 25839 -      ?        00:00:01 systemd
1 S     0       2       0  0  80   0 -     0 -      ?        00:00:00 kthreadd
1 I     0       3       2  0  60 -20 -     0 -      ?        00:00:00 rcu_gp
1 I     0       4       2  0  60 -20 -     0 -      ?        00:00:00 rcu_par_gp
1 I     0       6       2  0  60 -20 -     0 -      ?        00:00:00 kworker/0:0H-kblockd
1 I     0       9       2  0  60 -20 -     0 -      ?        00:00:00 mm_percpu_wq
1 S     0      10       2  0  80   0 -     0 -      ?        00:00:00 ksoftirqd/0
1 R     0      11       2  0  80   0 -     0 -      ?        00:00:06 rcu_sched
1 S     0      12       2  0 -40   - -     0 -      ?        00:00:00 migration/0
^C
```

사용한 명령어

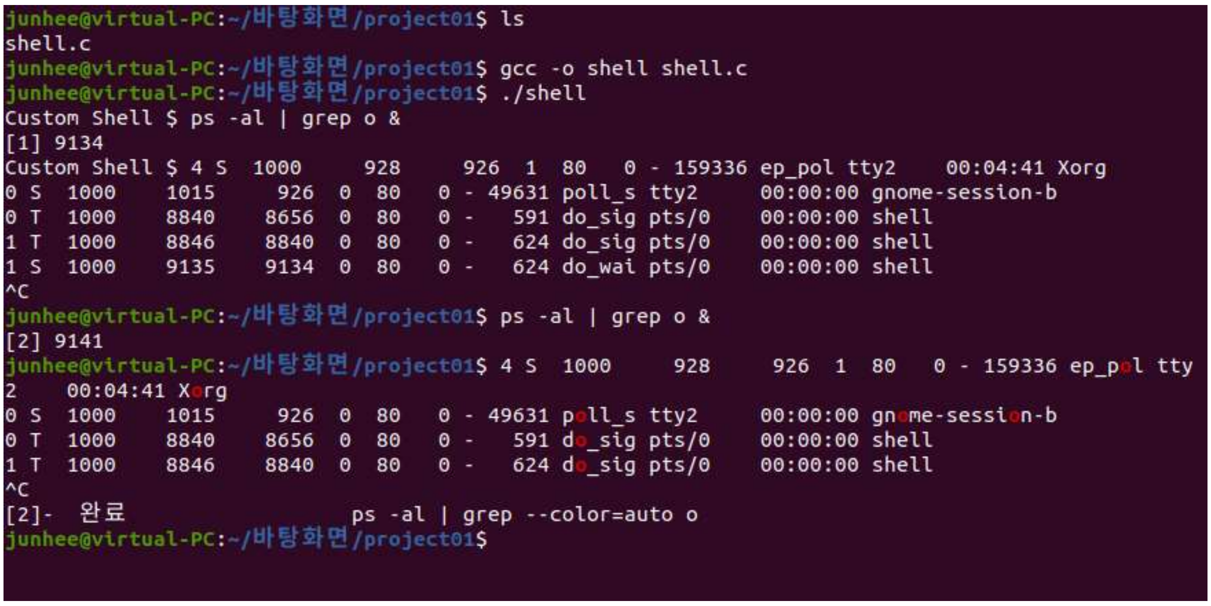
cat -s > file1, cat file1, ps -ael > file &

head < file, ps -ael > temp, head -10 < temp &

1-3. 파이프 명령어 조합



```
junhee@virtual-PC:~/바탕화면/project01$ ls
shell.c
junhee@virtual-PC:~/바탕화면/project01$ gcc -o shell shell.c
junhee@virtual-PC:~/바탕화면/project01$ ./shell
Custom Shell $ ps -ael | grep ab
Custom Shell $ ps -ael | grep 50
0 S  1000     980     909  0  80   0 - 61502 poll_s ?        00:00:00 gvfs-goa-volume
0 S  1000    1184     909  0  80   0 - 25045 poll_s ?        00:00:00 gnome-session-c
0 S  1000    1357     909  0  80   0 - 80500 poll_s ?        00:00:00 gsd-housekeepin
0 S  1000    1366     909  0  80   0 - 125071 poll_s ?       00:00:00 gsd-power
0 S  1000    1418    1190  0  80   0 - 57950 poll_s ?        00:00:00 gsd-disk-utilit
4 S  1000    1502     909  0  80   0 - 295513 poll_s ?       00:00:07 snap-store
0 S  1000    1550    1230  0  80   0 - 61933 poll_s ?        00:00:00 ibus-engine-han
0 S  1000    2505    1230  0  80   0 - 43271 poll_s ?        00:00:00 ibus-engine-sim
0 S  1000    8650     909  0  80   0 - 227253 poll_s ?       00:00:03 gnome-terminal-
0 S  1000    8656    8650  0  80   0 -  5249 do_wai pts/0    00:00:00 bash
0 S  1000    8793    8650  0  80   0 -  5171 poll_s pts/1    00:00:00 bash
Custom Shell $ ^C
junhee@virtual-PC:~/바탕화면/project01$ ps -ael | grep 50
0 S  1000     980     909  0  80   0 - 61502 poll_s ?        00:00:00 gvfs-goa-volume
0 S  1000    1184     909  0  80   0 - 25045 poll_s ?        00:00:00 gnome-session-c
0 S  1000    1357     909  0  80   0 - 80500 poll_s ?        00:00:00 gsd-housekeepin
0 S  1000    1366     909  0  80   0 - 125071 poll_s ?       00:00:00 gsd-power
0 S  1000    1418    1190  0  80   0 - 57950 poll_s ?        00:00:00 gsd-disk-utilit
4 S  1000    1502     909  0  80   0 - 295513 poll_s ?       00:00:07 snap-store
0 S  1000    1550    1230  0  80   0 - 61933 poll_s ?        00:00:00 ibus-engine-han
0 S  1000    2505    1230  0  80   0 - 43271 poll_s ?        00:00:00 ibus-engine-sim
0 S  1000    8650     909  0  80   0 - 227253 poll_s ?       00:00:03 gnome-terminal-
0 S  1000    8656    8650  0  80   0 -  5249 do_wai pts/0    00:00:00 bash
0 S  1000    8793    8650  0  80   0 -  5171 poll_s pts/1    00:00:00 bash
junhee@virtual-PC:~/바탕화면/project01$
```



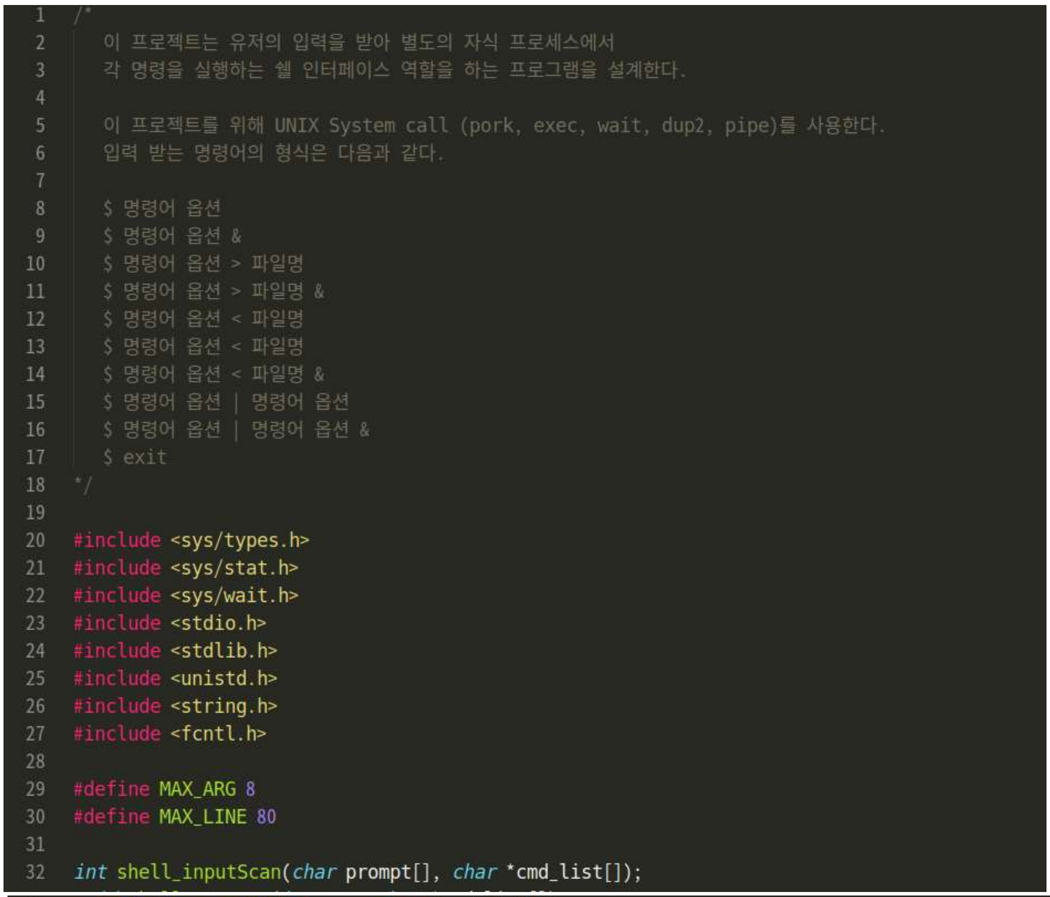
```
junhee@virtual-PC:~/바탕화면/project01$ ls
shell.c
junhee@virtual-PC:~/바탕화면/project01$ gcc -o shell shell.c
junhee@virtual-PC:~/바탕화면/project01$ ./shell
Custom Shell $ ps -al | grep o &
[1] 9134
Custom Shell $ 4 S  1000     928     926  1  80   0 - 159336 ep_pol tty2     00:04:41 Xorg
0 S  1000    1015     926  0  80   0 - 49631 poll_s tty2     00:00:00 gnome-session-b
0 T  1000    8840    8656  0  80   0 -   591 do_sig pts/0    00:00:00 shell
1 T  1000    8846    8840  0  80   0 -   624 do_sig pts/0    00:00:00 shell
1 S  1000    9135    9134  0  80   0 -   624 do_wai pts/0    00:00:00 shell
^C
junhee@virtual-PC:~/바탕화면/project01$ ps -al | grep o &
[2] 9141
junhee@virtual-PC:~/바탕화면/project01$ 4 S  1000     928     926  1  80   0 - 159336 ep_pol tty
2    00:04:41 Xorg
0 S  1000    1015     926  0  80   0 - 49631 poll_s tty2     00:00:00 gnome-session-b
0 T  1000    8840    8656  0  80   0 -   591 do_sig pts/0    00:00:00 shell
1 T  1000    8846    8840  0  80   0 -   624 do_sig pts/0    00:00:00 shell
^C
[2]-  완료                    ps -al | grep --color=auto o
junhee@virtual-PC:~/바탕화면/project01$
```

사용한 명령어

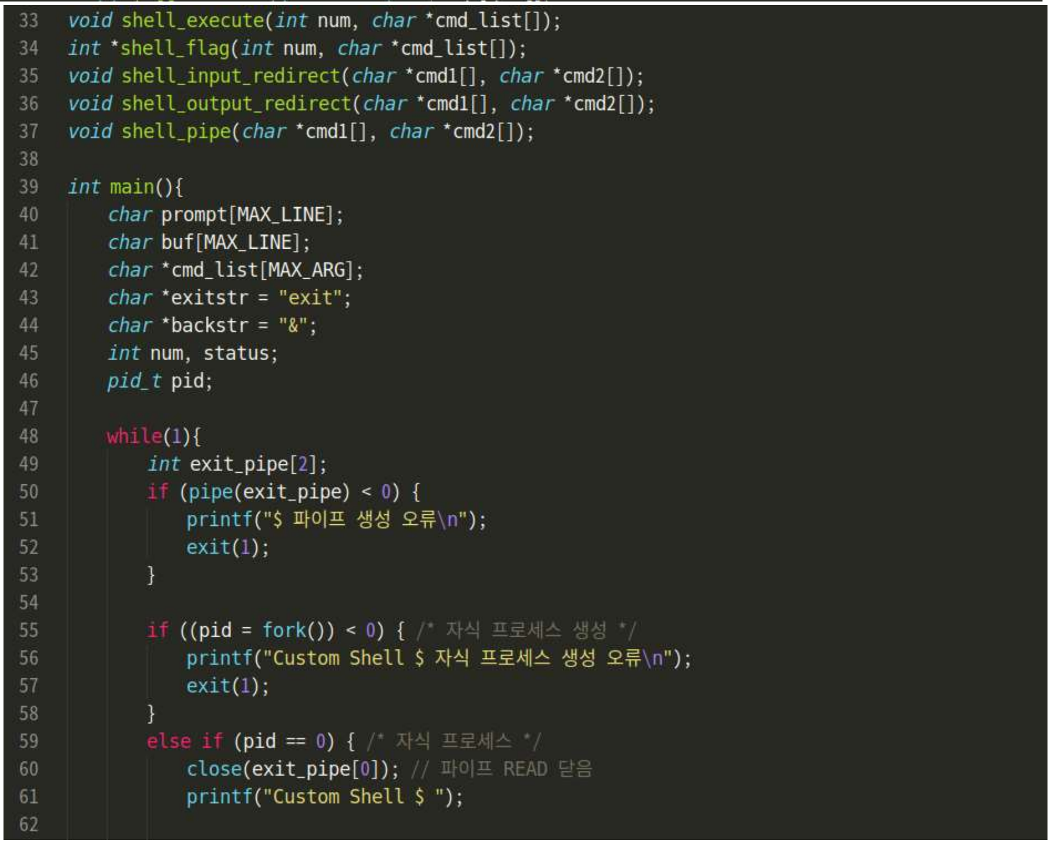
ps -ael | grep 50

ps -al | grep o &

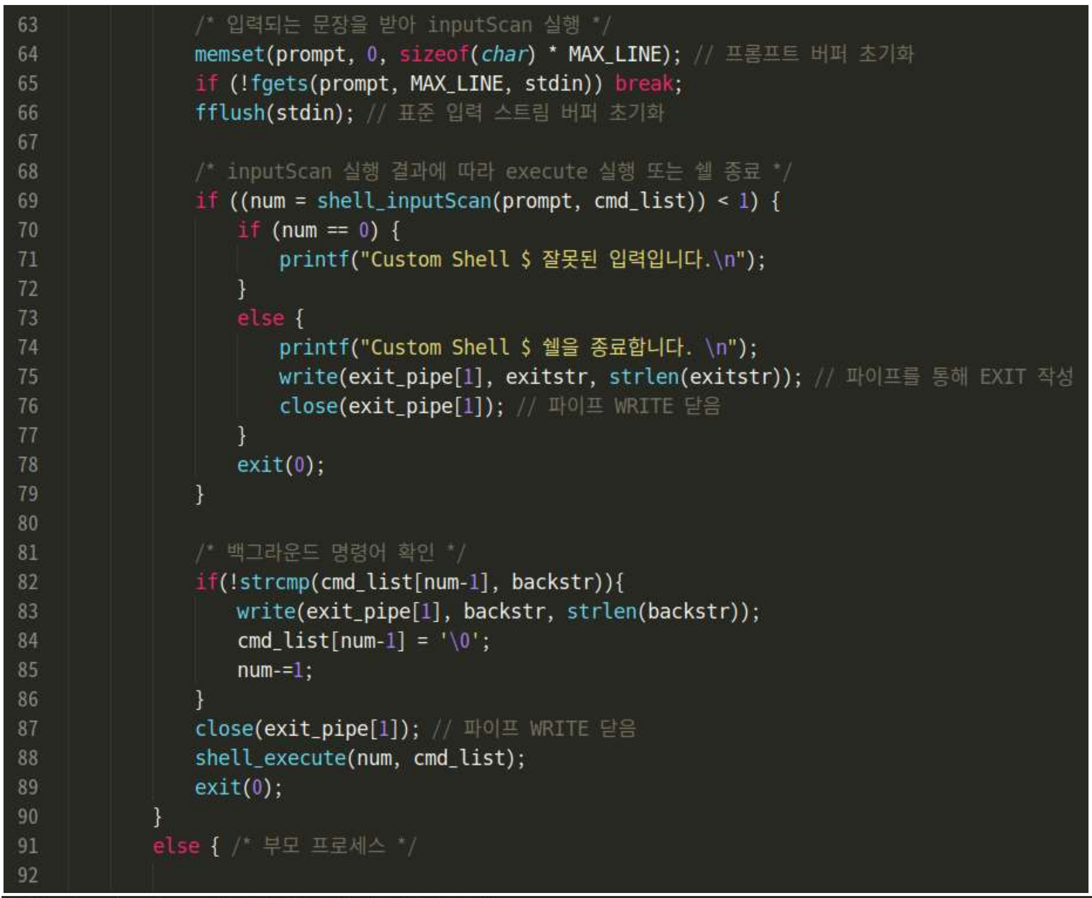
2. Code



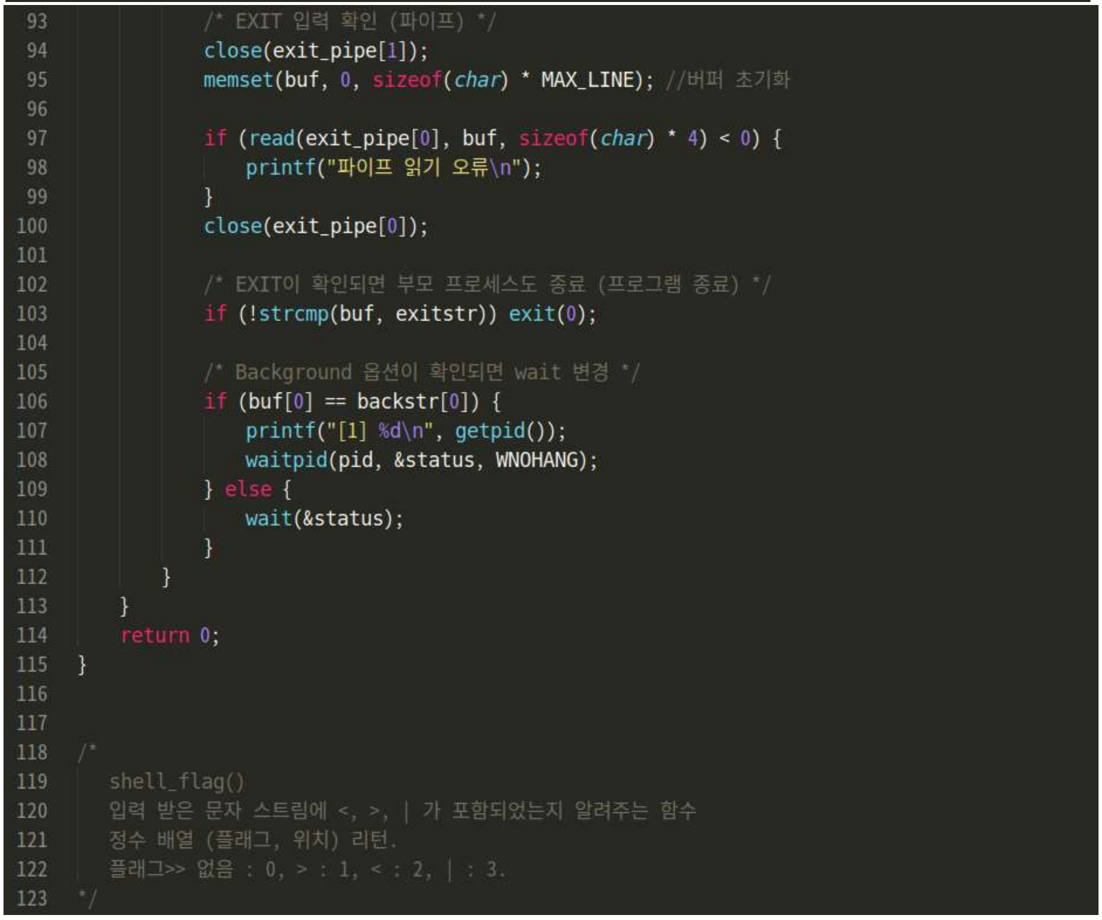
```c
/*
    이 프로젝트는 유저의 입력을 받아 별도의 자식 프로세스에서
    각 명령을 실행하는 쉘 인터페이스 역할을 하는 프로그램을 설계한다.

    이 프로젝트를 위해 UNIX System call (pork, exec, wait, dup2, pipe)를 사용한다.
    입력 받는 명령어의 형식은 다음과 같다.

    $ 명령어 옵션
    $ 명령어 옵션 &
    $ 명령어 옵션 > 파일명
    $ 명령어 옵션 > 파일명 &
    $ 명령어 옵션 < 파일명
    $ 명령어 옵션 < 파일명
    $ 명령어 옵션 < 파일명 &
    $ 명령어 옵션 | 명령어 옵션
    $ 명령어 옵션 | 명령어 옵션 &
    $ exit
*/

#include <sys/types.h>
#include <sys/stat.h>
#include <sys/wait.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <fcntl.h>

#define MAX_ARG 8
#define MAX_LINE 80

int shell_inputScan(char prompt[], char *cmd_list[]);
```



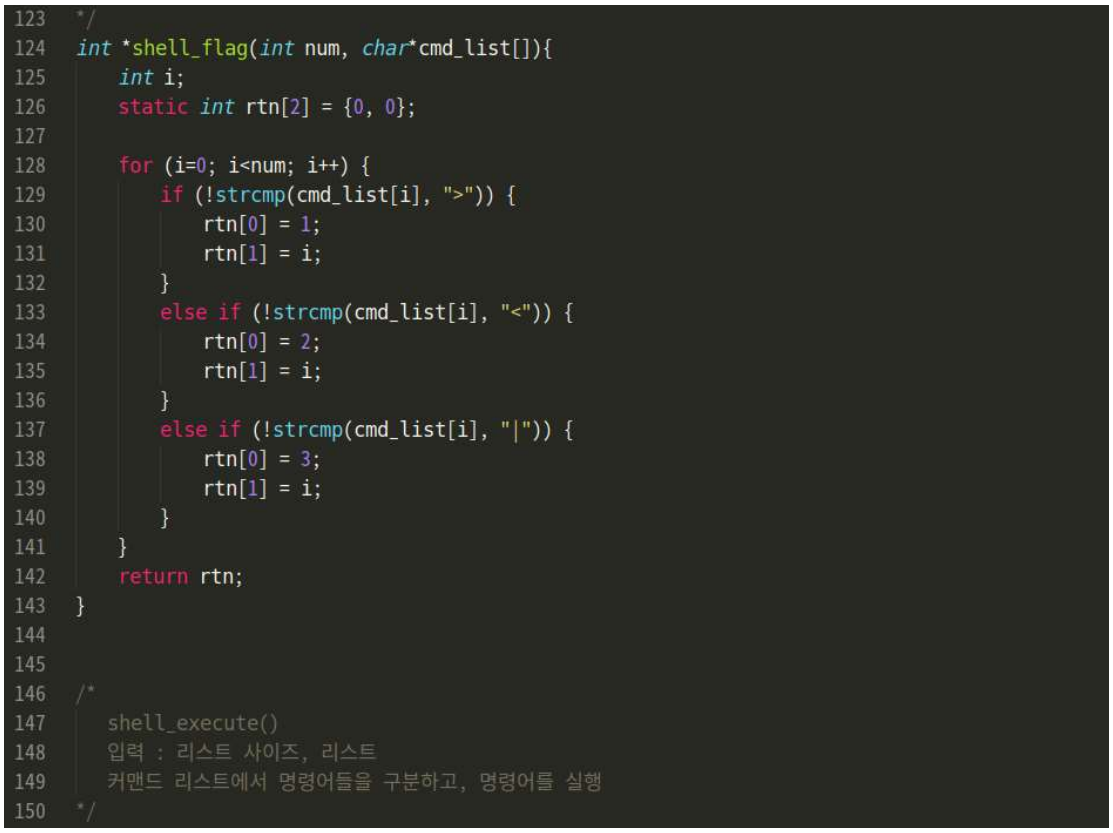
```c
void shell_execute(int num, char *cmd_list[]);
int *shell_flag(int num, char *cmd_list[]);
void shell_input_redirect(char *cmd1[], char *cmd2[]);
void shell_output_redirect(char *cmd1[], char *cmd2[]);
void shell_pipe(char *cmd1[], char *cmd2[]);

int main(){
    char prompt[MAX_LINE];
    char buf[MAX_LINE];
    char *cmd_list[MAX_ARG];
    char *exitstr = "exit";
    char *backstr = "&";
    int num, status;
    pid_t pid;

    while(1){
        int exit_pipe[2];
        if (pipe(exit_pipe) < 0) {
            printf("$ 파이프 생성 오류\n");
            exit(1);
        }

        if ((pid = fork()) < 0) { /* 자식 프로세스 생성 */
            printf("Custom Shell $ 자식 프로세스 생성 오류\n");
            exit(1);
        }
        else if (pid == 0) { /* 자식 프로세스 */
            close(exit_pipe[0]); // 파이프 READ 닫음
            printf("Custom Shell $ ");

```



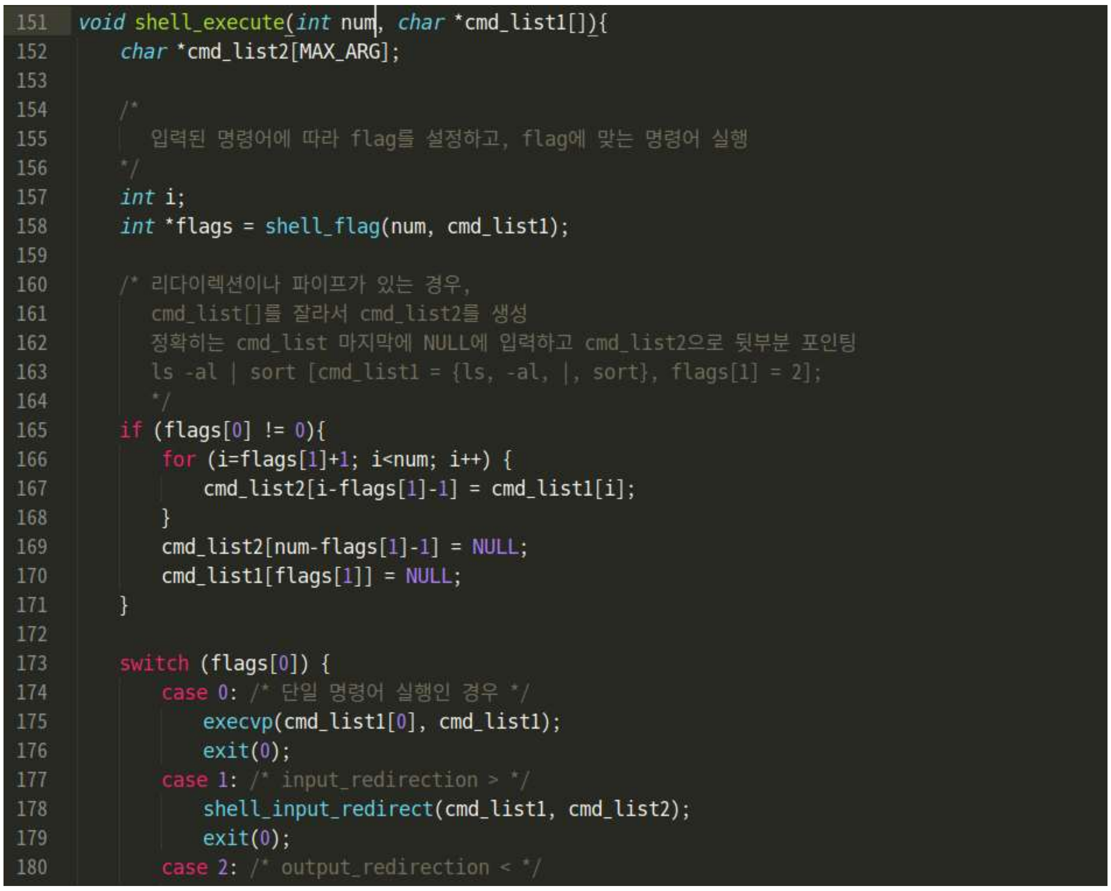
```c
            /* 입력되는 문장을 받아 inputScan 실행 */
            memset(prompt, 0, sizeof(char) * MAX_LINE); // 프롬프트 버퍼 초기화
            if (!fgets(prompt, MAX_LINE, stdin)) break;
            fflush(stdin); // 표준 입력 스트림 버퍼 초기화

            /* inputScan 실행 결과에 따라 execute 실행 또는 쉘 종료 */
            if ((num = shell_inputScan(prompt, cmd_list)) < 1) {
                if (num == 0) {
                    printf("Custom Shell $ 잘못된 입력입니다.\n");
                }
                else {
                    printf("Custom Shell $ 쉘을 종료합니다. \n");
                    write(exit_pipe[1], exitstr, strlen(exitstr)); // 파이프를 통해 EXIT 작성
                    close(exit_pipe[1]); // 파이프 WRITE 닫음
                }
                exit(0);
            }

            /* 백그라운드 명령어 확인 */
            if(!strcmp(cmd_list[num-1], backstr)){
                write(exit_pipe[1], backstr, strlen(backstr));
                cmd_list[num-1] = '\0';
                num-=1;
            }
            close(exit_pipe[1]); // 파이프 WRITE 닫음
            shell_execute(num, cmd_list);
            exit(0);
        }
        else { /* 부모 프로세스 */

```



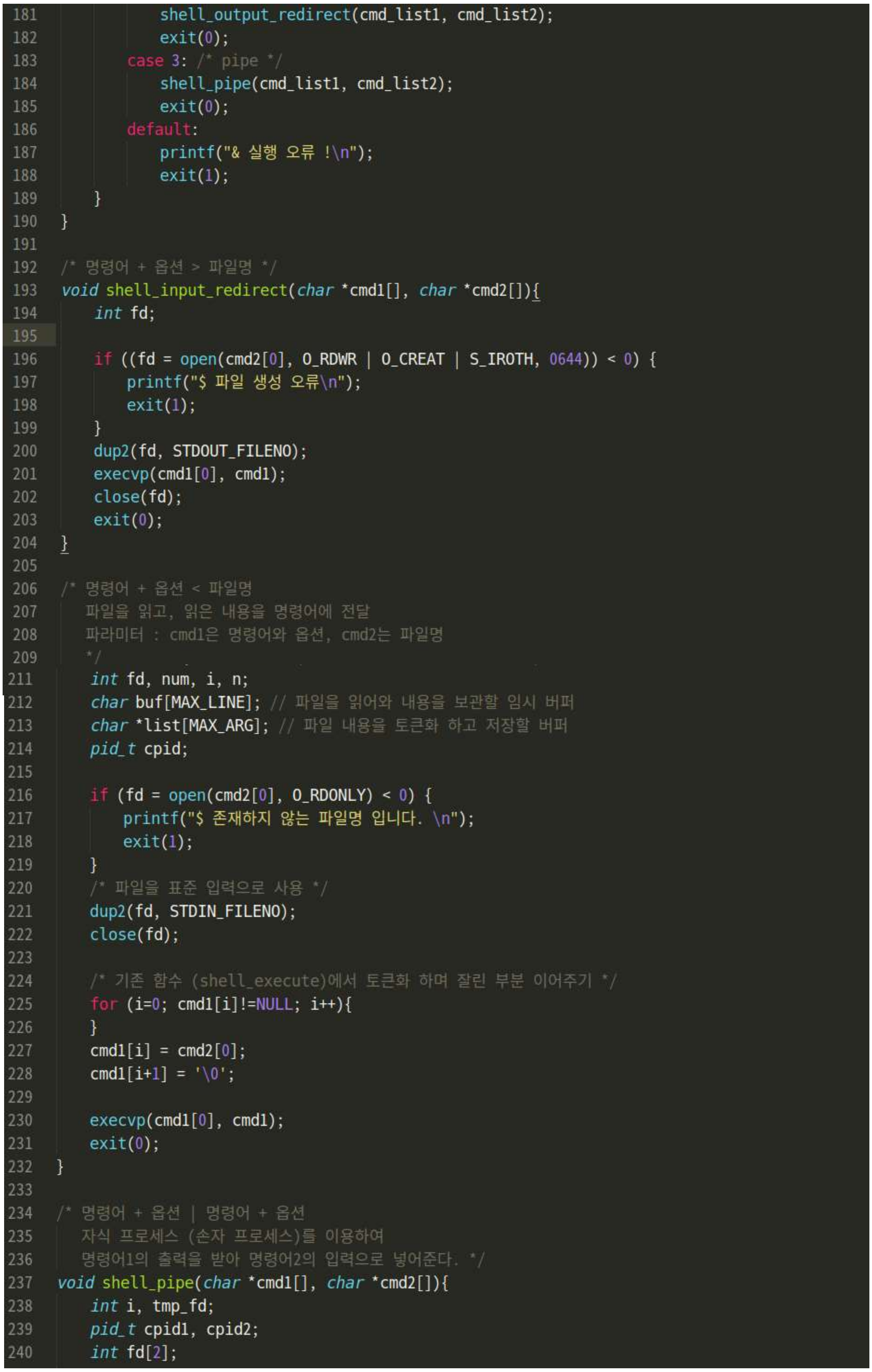
```c
            /* EXIT 입력 확인 (파이프) */
            close(exit_pipe[1]);
            memset(buf, 0, sizeof(char) * MAX_LINE); //버퍼 초기화

            if (read(exit_pipe[0], buf, sizeof(char) * 4) < 0) {
                printf("파이프 읽기 오류\n");
            }
            close(exit_pipe[0]);

            /* EXIT이 확인되면 부모 프로세스도 종료 (프로그램 종료) */
            if (!strcmp(buf, exitstr)) exit(0);

            /* Background 옵션이 확인되면 wait 변경 */
            if (buf[0] == backstr[0]) {
                printf("[1] %d\n", getpid());
                waitpid(pid, &status, WNOHANG);
            } else {
                wait(&status);
            }
        }
    }
    return 0;
}


/*
    shell_flag()
    입력 받은 문자 스트림에 <, >, | 가 포함되었는지 알려주는 함수
    정수 배열 (플래그, 위치) 리턴.
    플래그>> 없음 : 0, > : 1, < : 2, | : 3.
*/
```



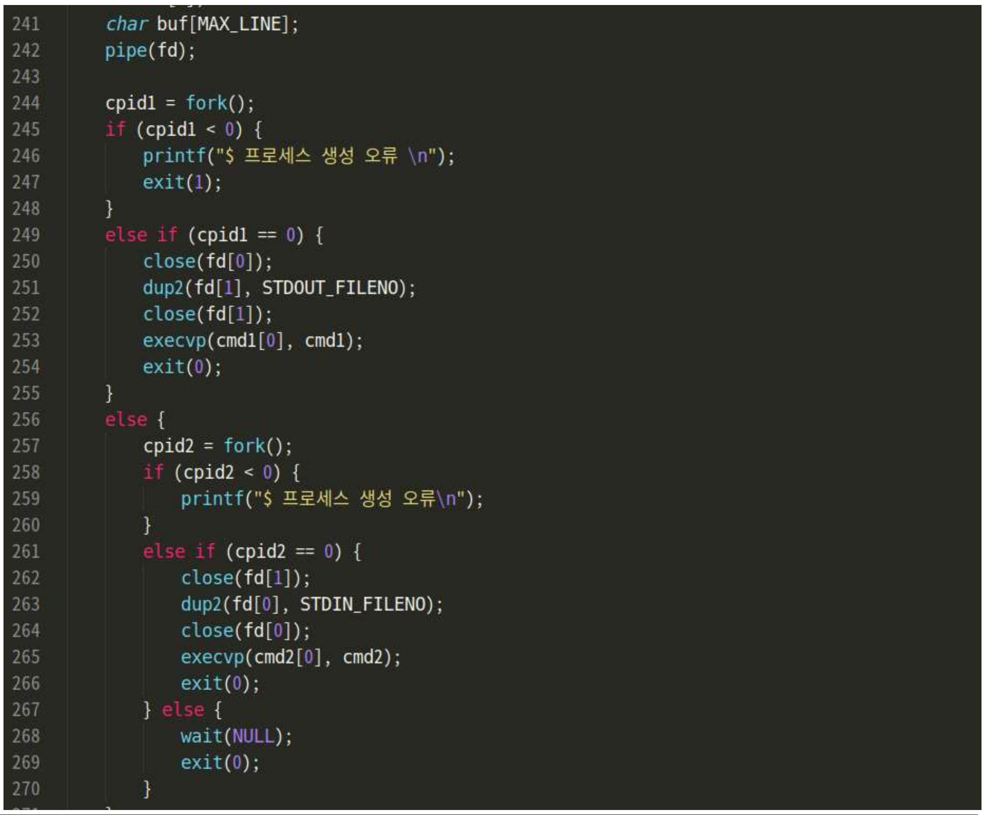
```c
*/
int *shell_flag(int num, char*cmd_list[]){
    int i;
    static int rtn[2] = {0, 0};

    for (i=0; i<num; i++) {
        if (!strcmp(cmd_list[i], ">")) {
            rtn[0] = 1;
            rtn[1] = i;
        }
        else if (!strcmp(cmd_list[i], "<")) {
            rtn[0] = 2;
            rtn[1] = i;
        }
        else if (!strcmp(cmd_list[i], "|")) {
            rtn[0] = 3;
            rtn[1] = i;
        }
    }
    return rtn;
}


/*
   shell_execute()
   입력 : 리스트 사이즈, 리스트
   커맨드 리스트에서 명령어들을 구분하고, 명령어를 실행
*/
```



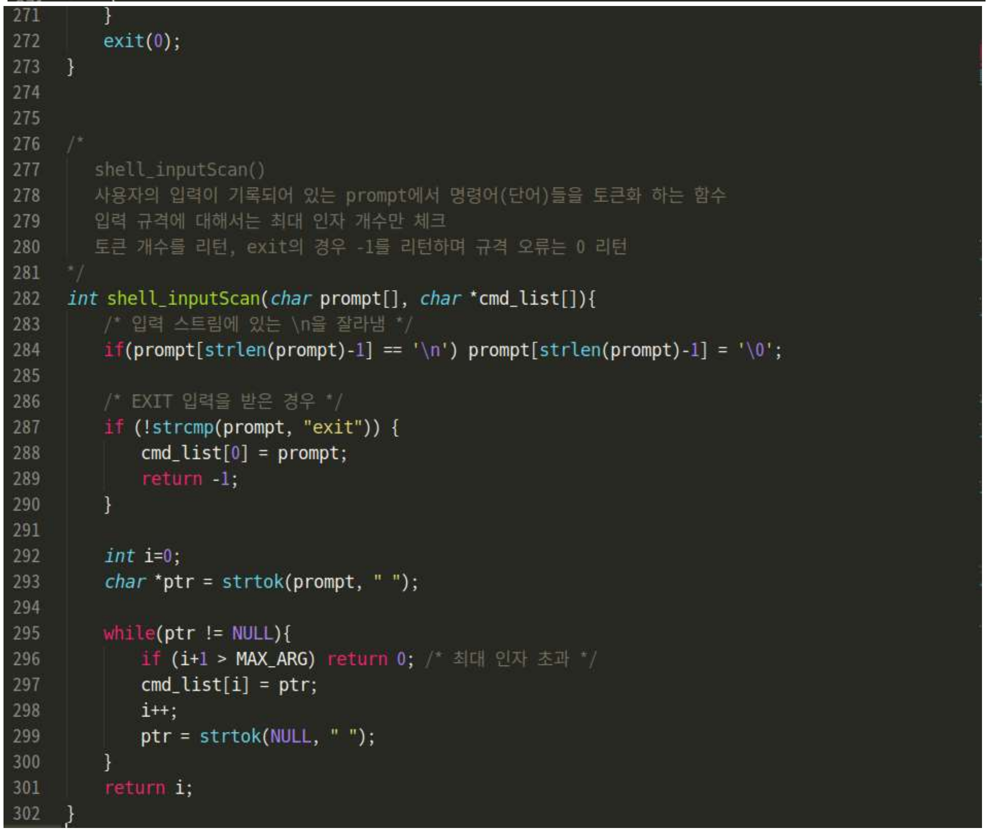
```c
void shell_execute(int num, char *cmd_list1[]){
    char *cmd_list2[MAX_ARG];

    /*
       입력된 명령어에 따라 flag를 설정하고, flag에 맞는 명령어 실행
    */
    int i;
    int *flags = shell_flag(num, cmd_list1);

    /* 리다이렉션이나 파이프가 있는 경우,
       cmd_list[]를 잘라서 cmd_list2를 생성
       정확히는 cmd_list 마지막에 NULL에 입력하고 cmd_list2으로 뒷부분 포인팅
       ls -al | sort [cmd_list1 = {ls, -al, |, sort}, flags[1] = 2];
       */
    if (flags[0] != 0){
        for (i=flags[1]+1; i<num; i++) {
            cmd_list2[i-flags[1]-1] = cmd_list1[i];
        }
        cmd_list2[num-flags[1]-1] = NULL;
        cmd_list1[flags[1]] = NULL;
    }

    switch (flags[0]) {
        case 0: /* 단일 명령어 실행인 경우 */
            execvp(cmd_list1[0], cmd_list1);
            exit(0);
        case 1: /* input_redirection > */
            shell_input_redirect(cmd_list1, cmd_list2);
            exit(0);
        case 2: /* output_redirection < */
```



```c
            shell_output_redirect(cmd_list1, cmd_list2);
            exit(0);
        case 3: /* pipe */
            shell_pipe(cmd_list1, cmd_list2);
            exit(0);
        default:
            printf("& 실행 오류 !\n");
            exit(1);
    }
}

/* 명령어 + 옵션 > 파일명 */
void shell_input_redirect(char *cmd1[], char *cmd2[]){
    int fd;

    if ((fd = open(cmd2[0], O_RDWR | O_CREAT | S_IROTH, 0644)) < 0) {
        printf("$ 파일 생성 오류\n");
        exit(1);
    }
    dup2(fd, STDOUT_FILENO);
    execvp(cmd1[0], cmd1);
    close(fd);
    exit(0);
}

/* 명령어 + 옵션 < 파일명
   파일을 읽고, 읽은 내용을 명령어에 전달
   파라미터 : cmd1은 명령어와 옵션, cmd2는 파일명
   */
    int fd, num, i, n;
    char buf[MAX_LINE]; // 파일을 읽어와 내용을 보관할 임시 버퍼
    char *list[MAX_ARG]; // 파일 내용을 토큰화 하고 저장할 버퍼
    pid_t cpid;

    if (fd = open(cmd2[0], O_RDONLY) < 0) {
        printf("$ 존재하지 않는 파일명 입니다. \n");
        exit(1);
    }
    /* 파일을 표준 입력으로 사용 */
    dup2(fd, STDIN_FILENO);
    close(fd);

    /* 기존 함수 (shell_execute)에서 토큰화 하며 잘린 부분 이어주기 */
    for (i=0; cmd1[i]!=NULL; i++){
    }
    cmd1[i] = cmd2[0];
    cmd1[i+1] = '\0';

    execvp(cmd1[0], cmd1);
    exit(0);
}

/* 명령어 + 옵션 | 명령어 + 옵션
   자식 프로세스 (손자 프로세스)를 이용하여
   명령어1의 출력을 받아 명령어2의 입력으로 넣어준다. */
void shell_pipe(char *cmd1[], char *cmd2[]){
    int i, tmp_fd;
    pid_t cpid1, cpid2;
    int fd[2];
```



```c
    char buf[MAX_LINE];
    pipe(fd);

    cpid1 = fork();
    if (cpid1 < 0) {
        printf("$ 프로세스 생성 오류 \n");
        exit(1);
    }
    else if (cpid1 == 0) {
        close(fd[0]);
        dup2(fd[1], STDOUT_FILENO);
        close(fd[1]);
        execvp(cmd1[0], cmd1);
        exit(0);
    }
    else {
        cpid2 = fork();
        if (cpid2 < 0) {
            printf("$ 프로세스 생성 오류\n");
        }
        else if (cpid2 == 0) {
            close(fd[1]);
            dup2(fd[0], STDIN_FILENO);
            close(fd[0]);
            execvp(cmd2[0], cmd2);
            exit(0);
        } else {
            wait(NULL);
            exit(0);
        }
```



```c
    }
    exit(0);
}


/*
    shell_inputScan()
    사용자의 입력이 기록되어 있는 prompt에서 명령어(단어)들을 토큰화 하는 함수
    입력 규격에 대해서는 최대 인자 개수만 체크
    토큰 개수를 리턴, exit의 경우 -1를 리턴하며 규격 오류는 0 리턴
*/
int shell_inputScan(char prompt[], char *cmd_list[]){
    /* 입력 스트림에 있는 \n을 잘라냄 */
    if(prompt[strlen(prompt)-1] == '\n') prompt[strlen(prompt)-1] = '\0';

    /* EXIT 입력을 받은 경우 */
    if (!strcmp(prompt, "exit")) {
        cmd_list[0] = prompt;
        return -1;
    }

    int i=0;
    char *ptr = strtok(prompt, " ");

    while(ptr != NULL){
        if (i+1 > MAX_ARG) return 0; /* 최대 인자 초과 */
        cmd_list[i] = ptr;
        i++;
        ptr = strtok(NULL, " ");
    }
    return i;
}
```

감사합니다.