LAB09

시스템 프로그래밍_류은경**Prof**

2021115744_권구태

1. (smsh1.c 수정)
- 한 줄에서 여러 명령어 처리할 수 있게 코드 수정
  > ls -l; ps -f; date

```
$ ./p1
> ls; ps; date
a.out  execute.c  smsh.h  smsh1  smsh1.c  splitline.c
   PID TTY          TIME CMD
65444 pts/0    00:00:00 bash
68448 pts/0    00:00:00 a.out
68625 pts/0    00:00:00 ps
2025. 5. 20. (화) 11:36:36 KST
>
```

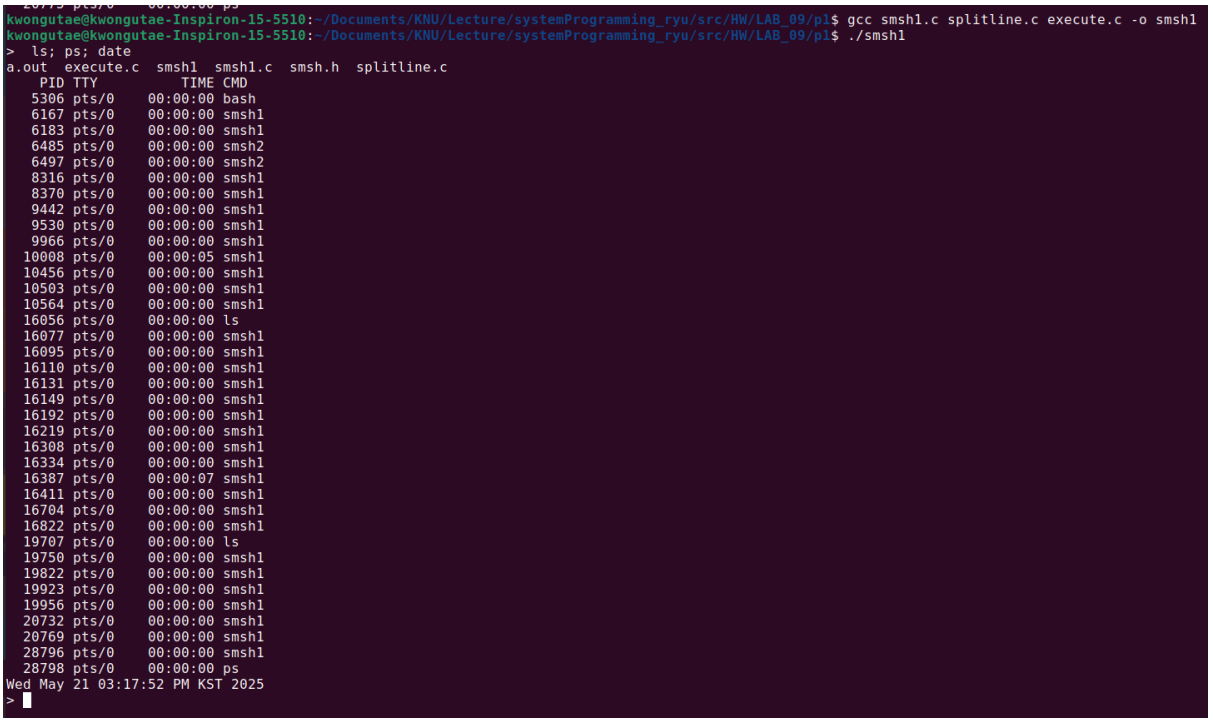
[코드 리뷰**]**

```
while( (cmdline = next_cmd(prompt, stdin) ) != NULL )
   {
       cp_cmdline = cmdline;

       token = strtok(cp_cmdline, "; ");

       while (token != NULL)
       {
           if( (arglist = splitline(token)) != NULL )
           {
               result = execute(arglist);
               freelist(arglist);
           }
           token = strtok(NULL, "; ");
       }

       free(cmdline);
   }
```

next_cmd 함수를 통해 입력 받은 문자열 cmdline을 cp_cmdline으로 복사함. strtok을 사용하면 해당 문자열에 변형이 일어나기 때문에 복사본으로 작업. 이후 문자열을 ;와 스페이스 단위로 자름. 잘린 문자열을 splitline 함수에 넣으면, 새로운 문자열 포인터 배열이 탄생한다. 해당 문자열 포인터 배열 arglist에는 {“token”, NULL} 형식으로 들어가 있게 되고,

이를 execute 함수에 넣으면 fork를 통한 프로세스 복제 후 자식 프로세스에서 명령어를 실행한다. exevcp() 함수 자체가, 최종적으로 명령어를 실행시켜주는 함수인데, 해당 함수의 인자를 보면, 명령어의 이름(즉, ls -al 이면 ls)와 같은 명령어 이름을 넣어주고, arglist[0]이 되겠다. 그리고 나서 어쨌든 원하는 명령어를 전체 실행해야 하기때문에 이중포인터배열을 같이 전달해준다(arglist). 따라서 해당 명령어가 실행이 된다.

[실행 결과**]**



[주의해야 할 점 및 새롭게 알게 된 점**]**

1. strtok사용 -> 해당 함수를 사용할 때는 분리하고자 하는 문자열을 첫번째 인자로 넘겨준 뒤, 해당 문자열을 반복해서 분리하고자 할 때, 그 다음부터는 NULL을 첫번째 인자로 넣어주어야 한다. 그렇지 않고 처음 넣었던 문자열을 계속 넣어주면 해당 문자열의 첫번째 주소를 계속해서 가리키기 때문에 문자열 토킹이 되지 않는다.
2. execvp()함수 사용 -> 해당 함수는 실제 터미널에서 명령어를 실행하게 해주는 함수인데, 해당 함수에 인자를 넘길때는 execvp((char *)명령어의 이름, (char **)해당 명령어의 문자열)의 형식이 되어야 한다. 그리고 해당 함수에 명령어를 넣기 위해선 마지막 문자열에는 NULL 값이 들어가 있어야 한다. 이를 위하여 splitline 함수에 해당 문자열을 넣어 작업을 진행한 것이다.

2. (smsh2. 수정)
- 아래 smsh2에서 "else" 구문이 처리될 수 있도록 코드 수정

```
$ ./p2
> date
2025. 5. 20. (화) 11:56:28 KST
> if grep ekryu /etc/passwd
> then
> echo ok
> else
> echo err
err
> fi
> ps
    PID TTY          TIME CMD
  69716 pts/1    00:00:00 bash
  69896 pts/1    00:00:00 a.out
  70020 pts/1    00:00:00 ps
>
```

[코드 리뷰]

```
enum states { NEUTRAL, WANT_THEN, THEN_BLOCK, ELSE_BLOCK };
```

enum states에 else_block 추가

```
    else if ( if_state == ELSE_BLOCK && if_result == SUCCESS )
        rv = 1;
    else if ( if_state == ELSE_BLOCK && if_result == FAIL )
        rv = 0;
```

else_block 명령어가 들어왔을 때, if_state가 정상적으로 변경이 되었고,
if_result가 변경에 성공했다면 flag를 1로 설정.
취소 했을 때를 대비한 rv flag 변수 설정

```
    else if( strcmp( cmd, "else" ) == 0 ) {
        if( if_state != WANT_THEN )
            rv = syn_err( "else unexpected");
        else {
            if_state = ELSE_BLOCK;
            rv = 0;
        }
    }
```

cmd 값이 else일 때 if_state를 else_block으로 변경하고, rv를 0으로 설정

[실행결과]

```
kwongutae@kwongutae-Inspiron-15-5510:~/Documents/KNU/Lecture/systemProgramming_ryu/src/HW/LAB_09/p2$ ./p2
> if grep kwongutae /etc/passwd
kwongutae:x:1000:1000:KwonGutae:/home/kwongutae:/bin/bash
> then
> echo ok
ok
> echo err
err
> fi
> ps
    PID TTY          TIME CMD
   5306 pts/0    00:00:00 bash
   6167 pts/0    00:00:00 smsh1
   6183 pts/0    00:00:00 smsh1
   6485 pts/0    00:00:00 smsh2
   6497 pts/0    00:00:00 smsh2
   8316 pts/0    00:00:00 smsh1
   8370 pts/0    00:00:00 smsh1
   9442 pts/0    00:00:00 smsh1
   9530 pts/0    00:00:00 smsh1
   9966 pts/0    00:00:00 smsh1
  10008 pts/0    00:00:05 smsh1
  10456 pts/0    00:00:00 smsh1
  10503 pts/0    00:00:00 smsh1
  10564 pts/0    00:00:00 smsh1
  16056 pts/0    00:00:00 ls
  16077 pts/0    00:00:00 smsh1
  16095 pts/0    00:00:00 smsh1
  16110 pts/0    00:00:00 smsh1
  16131 pts/0    00:00:00 smsh1
  16149 pts/0    00:00:00 smsh1
  16192 pts/0    00:00:00 smsh1
  16219 pts/0    00:00:00 smsh1
  16308 pts/0    00:00:00 smsh1
  16334 pts/0    00:00:00 smsh1
  16387 pts/0    00:00:07 smsh1
  16411 pts/0    00:00:00 smsh1
  16704 pts/0    00:00:00 smsh1
  16822 pts/0    00:00:00 smsh1
  19707 pts/0    00:00:00 ls
  19750 pts/0    00:00:00 smsh1
  19822 pts/0    00:00:00 smsh1
  19923 pts/0    00:00:00 smsh1
  19956 pts/0    00:00:00 smsh1
  20732 pts/0    00:00:00 smsh1
  20769 pts/0    00:00:00 smsh1
  28796 pts/0    00:00:00 smsh1
  30029 pts/0    00:00:00 shsm2
  30056 pts/0    00:00:00 p2
  30205 pts/0    00:00:00 p2
  30218 pts/0    00:00:00 ps
>
```

[주의할 점 및 새롭게 알게된 점]

1. 1번에서 배운 것과 동일,
2. flag 변수 rv의 사용 위치 중요.