

## 리눅스시스템 Lab05

분반: 002

학과: 컴퓨터과학과

학번: 2210701

이름: 김주영

### 1. ps 실습

1) p2-4의 명령 4개 각각 실행한 후, 터미널 창을 캡처한다.

```
u2210701@u2210701-VirtualBox: ~/linux/chap06
u2210701@u2210701-VirtualBox:~/linux/chap06$ ps
  PID TTY          TIME CMD
  4088 pts/0    00:00:00 bash
  4096 pts/0    00:00:00 ps
u2210701@u2210701-VirtualBox:~/linux/chap06$ ps -f
UID            PID    PPID  C  STIME TTY          TIME CMD
u2210701        4088     4062  0   10:04 pts/0    00:00:00 bash
u2210701        4097     4088  0   10:04 pts/0    00:00:00 ps -f
u2210701@u2210701-VirtualBox:~/linux/chap06$
```

```
u2210701@u2210701-VirtualBox: ~/linux/chap06
u2210701@u2210701-VirtualBox:~/linux/chap06$ ps aux
USER          PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.1  0.5 166712 11868 ?        Ss   09:58   0:02 /sbin/init sp
root           2  0.0  0.0      0     0 ?        S    09:58   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<   09:58   0:00 [rcu_gp]
root           4  0.0  0.0      0     0 ?        I<   09:58   0:00 [rcu_par_gp]
root           5  0.0  0.0      0     0 ?        I<   09:58   0:00 [slub_flushwq
root           6  0.0  0.0      0     0 ?        I<   09:58   0:00 [netns]
root           8  0.0  0.0      0     0 ?        I<   09:58   0:00 [kworker/0:0H
root          10  0.0  0.0      0     0 ?        I<   09:58   0:00 [mm_percpu_wq
root          11  0.0  0.0      0     0 ?        I    09:58   0:00 [rcu_tasks_kt
root          12  0.0  0.0      0     0 ?        I    09:58   0:00 [rcu_tasks_ru
root          13  0.0  0.0      0     0 ?        I    09:58   0:00 [rcu_tasks_tr
root          14  0.0  0.0      0     0 ?        S    09:58   0:00 [ksoftirqd/0]
root          15  0.0  0.0      0     0 ?        I    09:58   0:01 [rcu_preempt]
root          16  0.0  0.0      0     0 ?        S    09:58   0:00 [migration/0]
root          17  0.0  0.0      0     0 ?        S    09:58   0:00 [idle_inject/
root          19  0.0  0.0      0     0 ?        S    09:58   0:00 [cpuhp/0]
root          20  0.0  0.0      0     0 ?        S    09:58   0:00 [kdevtmpfs]
root          21  0.0  0.0      0     0 ?        I<   09:58   0:00 [inet_frag_wq
root          22  0.0  0.0      0     0 ?        S    09:58   0:00 [kauditd]
root          23  0.0  0.0      0     0 ?        S    09:58   0:00 [khungtaskd]
root          24  0.0  0.0      0     0 ?        S    09:58   0:00 [oom_reaper]
root          27  0.0  0.0      0     0 ?        I<   09:58   0:00 [writeback]
```

```
u2210701@u2210701-VirtualBox: ~/linux/chap06
u2210701@u2210701-VirtualBox:~/linux/chap06$ ps -ef
```

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	1	0	0	09:58	?	00:00:02	/sbin/init splash
root	2	0	0	09:58	?	00:00:00	[kthreadd]
root	3	2	0	09:58	?	00:00:00	[rcu_gp]
root	4	2	0	09:58	?	00:00:00	[rcu_par_gp]
root	5	2	0	09:58	?	00:00:00	[slub_flushwq]
root	6	2	0	09:58	?	00:00:00	[netns]
root	8	2	0	09:58	?	00:00:00	[kworker/0:0H-kblockd]
root	10	2	0	09:58	?	00:00:00	[mm_percpu_wq]
root	11	2	0	09:58	?	00:00:00	[rcu_tasks_kthread]
root	12	2	0	09:58	?	00:00:00	[rcu_tasks_rude_kthread]
root	13	2	0	09:58	?	00:00:00	[rcu_tasks_trace_kthread]
root	14	2	0	09:58	?	00:00:00	[ksoftirqd/0]
root	15	2	0	09:58	?	00:00:01	[rcu_preempt]
root	16	2	0	09:58	?	00:00:00	[migration/0]
root	17	2	0	09:58	?	00:00:00	[idle_inject/0]
root	19	2	0	09:58	?	00:00:00	[cpuhp/0]
root	20	2	0	09:58	?	00:00:00	[kdevtmpfs]
root	21	2	0	09:58	?	00:00:00	[inet_frag_wq]
root	22	2	0	09:58	?	00:00:00	[kauditd]
root	23	2	0	09:58	?	00:00:00	[khungtaskd]
root	24	2	0	09:58	?	00:00:00	[oom_reaper]
root	27	2	0	09:58	?	00:00:00	[writeback]

2) ps 명령어의 역할과 출력 결과에 대해 설명한다.

ps명령어는 현재 시스템 내에 존재하는 프로세스들의 실행 상태를 요약해서 출력한다.

3) 세 개의 옵션이 어떤 의미를 가지고 있는지 출력 결과를 중심으로 설명한다.

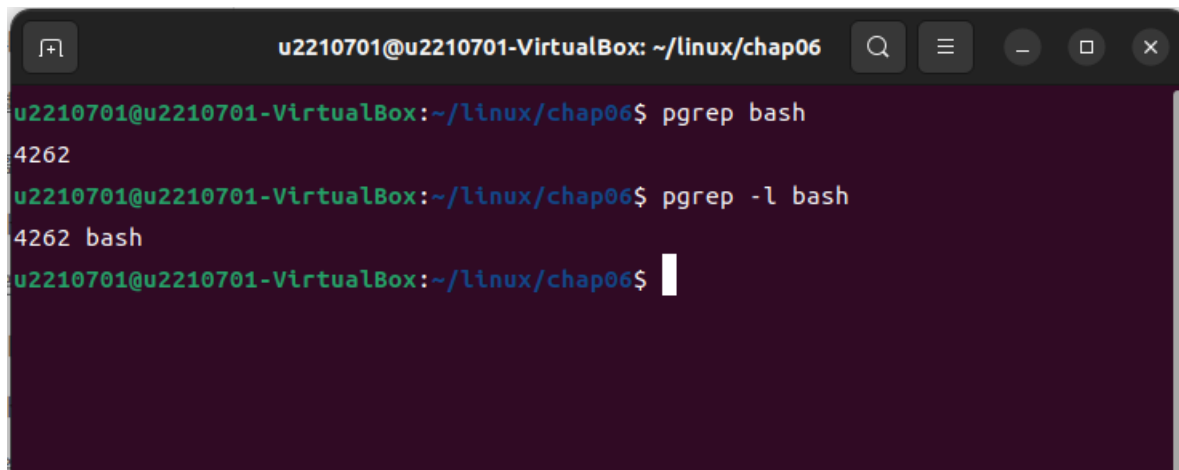
-f: UID, PPID 등의 정보 등을 추가로 출력해 보다 상세한 정보를 표출한다

-aux: 시스템의 동작중인 모든 프로세스의 정보를 표출한다

-ef: 모든 프로세스를 풀 포맷으로 출력한다

## 2. pgrep 실습

1) p5의 명령 2개 각각을 실행한 후, 터미널 창을 캡처한다.

A terminal window titled 'u2210701@u2210701-VirtualBox: ~/linux/chap06'. The prompt is 'u2210701@u2210701-VirtualBox:~/linux/chap06\$'. The first command is 'pgrep bash', which outputs '4262'. The second command is 'pgrep -l bash', which outputs '4262 bash'. The prompt is now 'u2210701@u2210701-VirtualBox:~/linux/chap06\$' with a cursor.

```
u2210701@u2210701-VirtualBox:~/linux/chap06$ pgrep bash
4262
u2210701@u2210701-VirtualBox:~/linux/chap06$ pgrep -l bash
4262 bash
u2210701@u2210701-VirtualBox:~/linux/chap06$
```

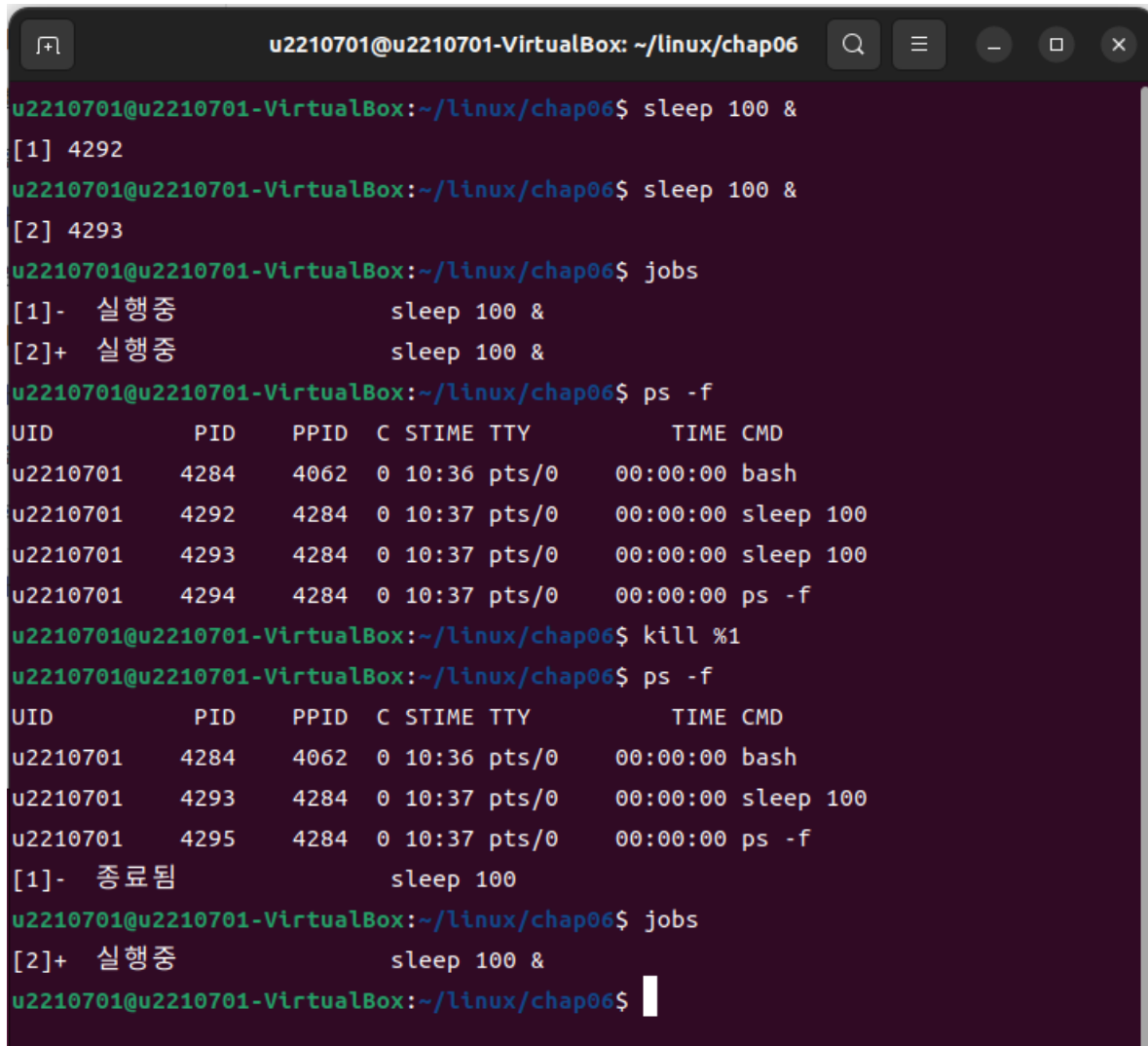
2) 각 명령이 어떤 의미를 가지고 있는지 출력 결과를 중심으로 설명한다.

pgrep bash 명령은 bash 프로세스의 번호만을 출력한다

pgrep -l bash 명령은 -l이 추가되며 프로세스 번호와 이름을 같이 출력한다.

### 3. 전면처리, 후면처리 실습

1) p6의 명령 6개 각각을 실행한 후, 터미널 창을 캡처한다.



```
u2210701@u2210701-VirtualBox: ~/linux/chap06
u2210701@u2210701-VirtualBox:~/linux/chap06$ sleep 100 &
[1] 4292
u2210701@u2210701-VirtualBox:~/linux/chap06$ sleep 100 &
[2] 4293
u2210701@u2210701-VirtualBox:~/linux/chap06$ jobs
[1]-  실행중                sleep 100 &
[2]+  실행중                sleep 100 &
u2210701@u2210701-VirtualBox:~/linux/chap06$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
u2210701      4284     4062  0  10:36 pts/0        00:00:00 bash
u2210701      4292     4284  0  10:37 pts/0        00:00:00 sleep 100
u2210701      4293     4284  0  10:37 pts/0        00:00:00 sleep 100
u2210701      4294     4284  0  10:37 pts/0        00:00:00 ps -f
u2210701@u2210701-VirtualBox:~/linux/chap06$ kill %1
u2210701@u2210701-VirtualBox:~/linux/chap06$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
u2210701      4284     4062  0  10:36 pts/0        00:00:00 bash
u2210701      4293     4284  0  10:37 pts/0        00:00:00 sleep 100
u2210701      4295     4284  0  10:37 pts/0        00:00:00 ps -f
[1]-  종료됨                sleep 100
u2210701@u2210701-VirtualBox:~/linux/chap06$ jobs
[2]+  실행중                sleep 100 &
u2210701@u2210701-VirtualBox:~/linux/chap06$
```

2) 각 명령이 어떤 의미를 가지고 있는지 출력 결과를 중심으로 설명한다. (실습교안 설명 참고)  
sleep 100 & 명령에서 (sleep 100)명령은 100초동안 중지시키지만 &입력으로 인해 후면처리된다.

jobs 명령으로 현재 후면처리로 실행중인 프로세스 정보를 출력한다.

ps -f 명령으로 각 프로세스들의 상세정보를 출력한다

kill %1 명령에서 %1입력으로 현재 실행중인 첫번째 후면 프로세스를 강제종료한다

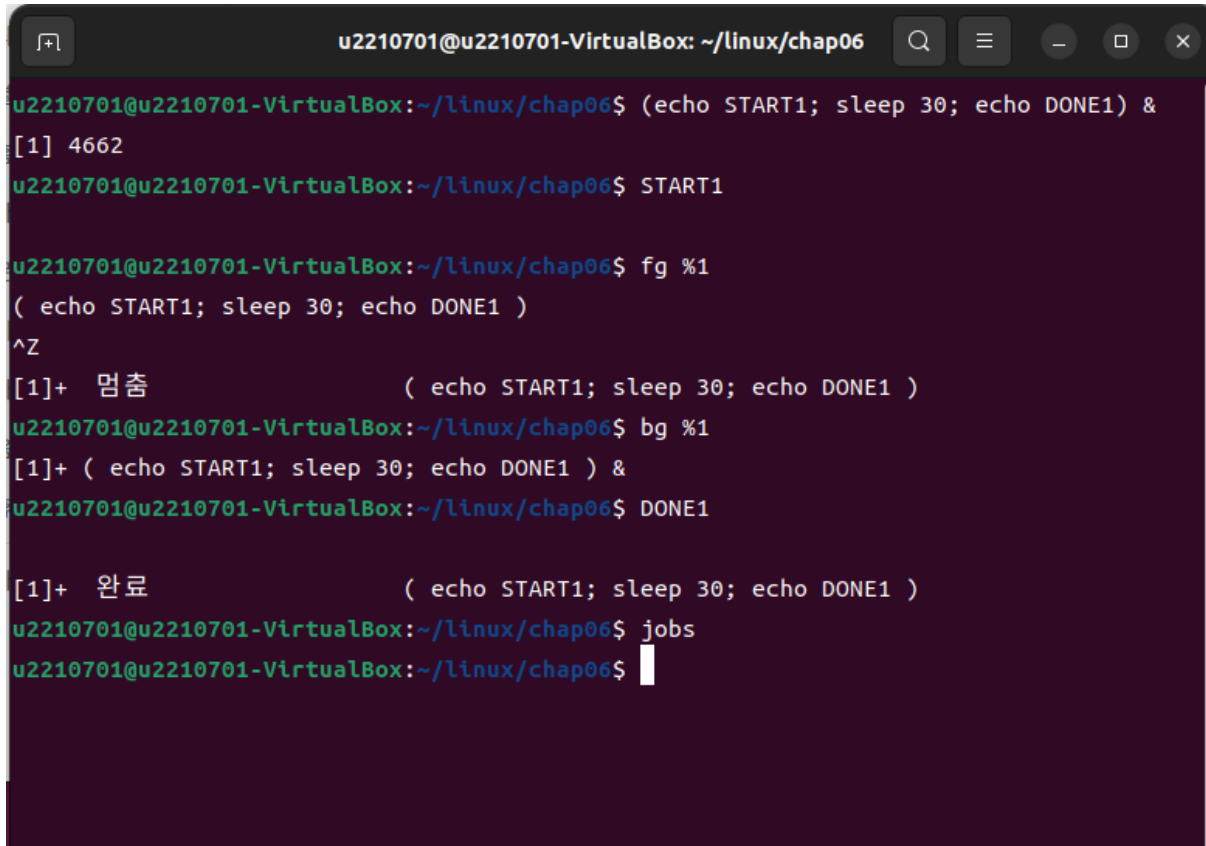
다시 ps -f 명령과 jobs 명령으로 프로세스 정보와 실행중인 후면 프로세스 정보를 출력한다.

3) 출력된 프로세스들의 부모-자식 관계를 설명하시오.

출력된 결과를 보면 bash 쉘의 PID와 sleep 100, ps -f 의 PPID값이 같다. 그 이유는 각 명령들이 bash 쉘 위에서 실행되었기 때문에 부모 프로세스인 bash의 PID가 자식 프로세스의 PPID로 나타난다. (PPID = Parent Process ID)

#### 4. 작업제어 실습

1) p7의 명령 4개 각각을 실행한 후, 터미널 창을 캡처한다.



```
u2210701@u2210701-VirtualBox: ~/linux/chap06
u2210701@u2210701-VirtualBox:~/linux/chap06$ (echo START1; sleep 30; echo DONE1) &
[1] 4662
u2210701@u2210701-VirtualBox:~/linux/chap06$ START1

u2210701@u2210701-VirtualBox:~/linux/chap06$ fg %1
( echo START1; sleep 30; echo DONE1 )
^Z
[1]+  멈춤                ( echo START1; sleep 30; echo DONE1 )
u2210701@u2210701-VirtualBox:~/linux/chap06$ bg %1
[1]+ ( echo START1; sleep 30; echo DONE1 ) &
u2210701@u2210701-VirtualBox:~/linux/chap06$ DONE1

[1]+  완료                ( echo START1; sleep 30; echo DONE1 )
u2210701@u2210701-VirtualBox:~/linux/chap06$ jobs
u2210701@u2210701-VirtualBox:~/linux/chap06$
```

2) 각 명령이 어떤 의미를 가지고 있는지 출력 결과를 중심으로 설명한다. (실습교안 설명 참고)  
(echo ... DONE1) & 괄호 안의 명령을 &입력으로 후면처리 한다.

fg %1 명령으로 %1에 해당하는, 첫번째로 실행중인 후면작업을 전면작업으로 전환한다.

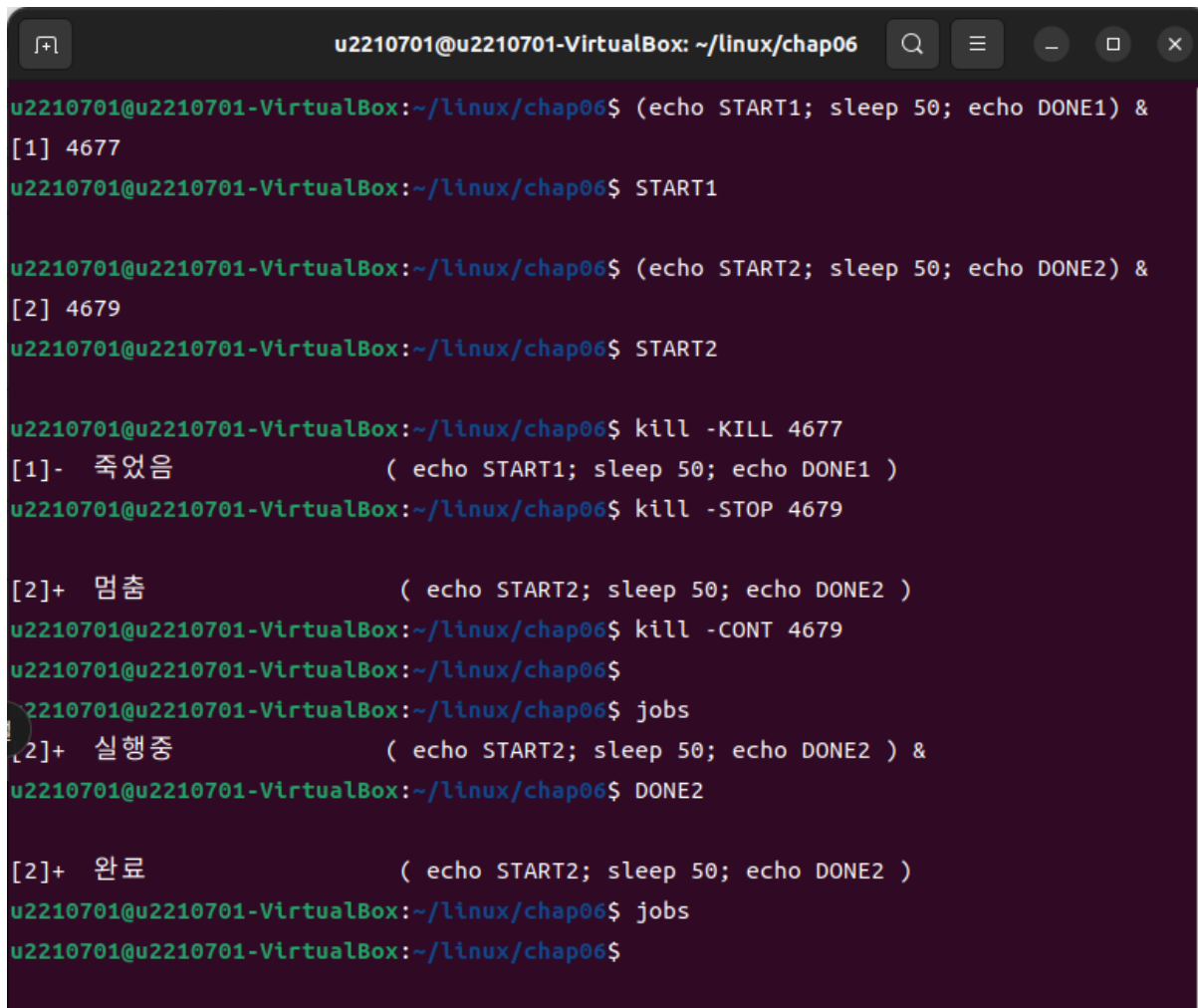
sleep 30으로 인해 대기중인 작업을 ctrl z로 정지시킨다.

bg %1로 중지된 작업을 후면작업으로 전환한 뒤 다시 실행한다.

jobs명령을 입력해도 실행중인 후면작업이 없으므로 출력값은 없다.

## 5. 작업제어 실습

1) p8의 명령 6개 각각을 실행한 후, 터미널 창을 캡처한다.



```
u2210701@u2210701-VirtualBox: ~/linux/chap06
u2210701@u2210701-VirtualBox:~/linux/chap06$ (echo START1; sleep 50; echo DONE1) &
[1] 4677
u2210701@u2210701-VirtualBox:~/linux/chap06$ START1

u2210701@u2210701-VirtualBox:~/linux/chap06$ (echo START2; sleep 50; echo DONE2) &
[2] 4679
u2210701@u2210701-VirtualBox:~/linux/chap06$ START2

u2210701@u2210701-VirtualBox:~/linux/chap06$ kill -KILL 4677
[1]-  죽었음                ( echo START1; sleep 50; echo DONE1 )
u2210701@u2210701-VirtualBox:~/linux/chap06$ kill -STOP 4679

[2]+  멈춤                  ( echo START2; sleep 50; echo DONE2 )
u2210701@u2210701-VirtualBox:~/linux/chap06$ kill -CONT 4679
u2210701@u2210701-VirtualBox:~/linux/chap06$
u2210701@u2210701-VirtualBox:~/linux/chap06$ jobs
[2]+  실행중                ( echo START2; sleep 50; echo DONE2 ) &
u2210701@u2210701-VirtualBox:~/linux/chap06$ DONE2

[2]+  완료                  ( echo START2; sleep 50; echo DONE2 )
u2210701@u2210701-VirtualBox:~/linux/chap06$ jobs
u2210701@u2210701-VirtualBox:~/linux/chap06$
```

2) 각 명령이 어떤 의미를 가지고 있는지 출력 결과를 중심으로 설명한다. kill 명령어 설명의 경우, 각 옵션들에 대한 의미가 무엇인지에 대해 설명도 포함하시기 바랍니다. (실습교안 설명 참고)  
(echo ... DONE1) &, (echo ... DONE2) & 명령 두가지는 모두 &입력으로 인해 후면처리 된다.

kill명령의 -KILL, -STOP, -CONT 옵션을 사용해 출력한다. 이때, 옵션 뒤에 명령을 적용시킬 프로세스 번호를 입력해 특정한다.

-KILL 옵션은 프로세스를 무조건 종료시킨다.

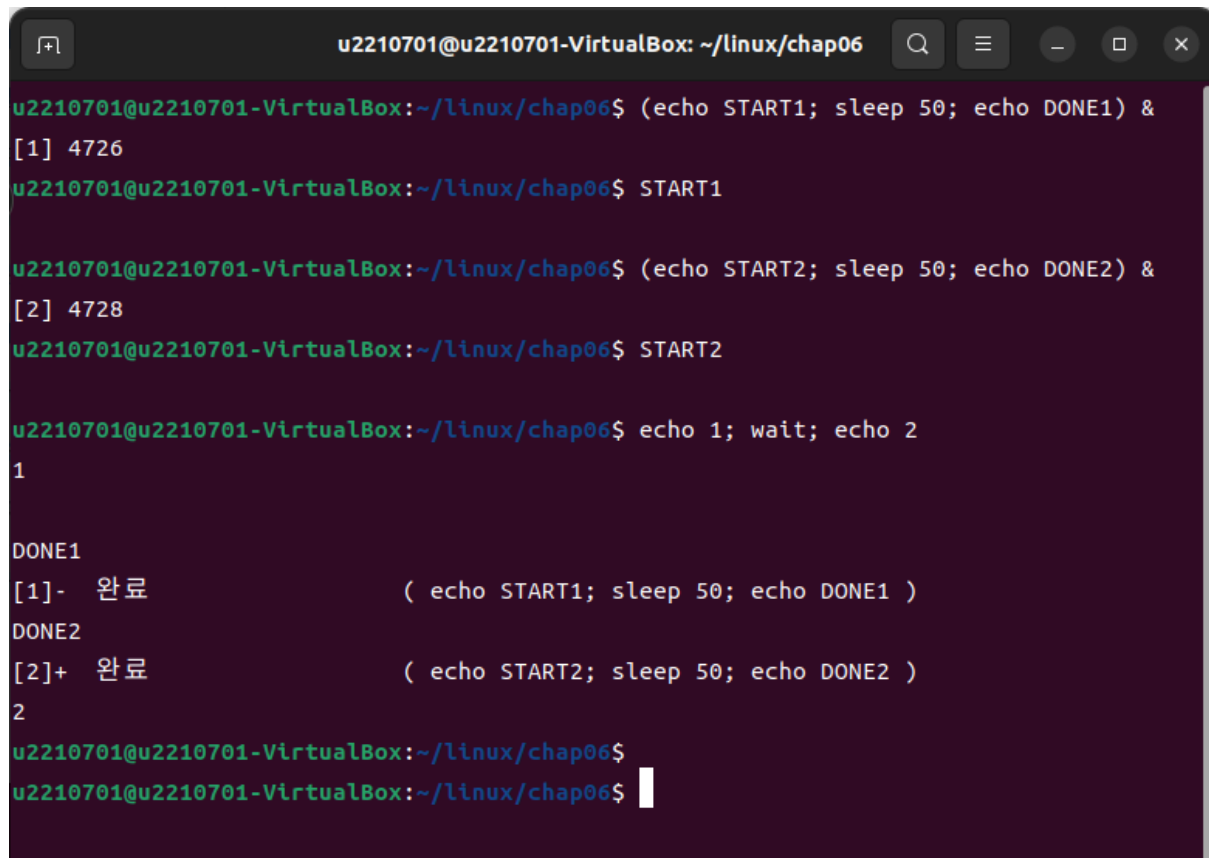
-STOP 옵션은 프로세스를 중지시킨다.

-CONT 옵션은 중지된 프로세스를 다시 실행시킨다.

모든 프로세스가 실행완료되었으므로 jobs명령을 입력해도 출력되는 것은 없다.

## 6. 명령어 열/그룹 실습

1) p9의 명령 3개 각각을 실행한 후, 터미널 창을 캡처한다.



```
u2210701@u2210701-VirtualBox: ~/linux/chap06
u2210701@u2210701-VirtualBox:~/linux/chap06$ (echo START1; sleep 50; echo DONE1) &
[1] 4726
u2210701@u2210701-VirtualBox:~/linux/chap06$ START1

u2210701@u2210701-VirtualBox:~/linux/chap06$ (echo START2; sleep 50; echo DONE2) &
[2] 4728
u2210701@u2210701-VirtualBox:~/linux/chap06$ START2

u2210701@u2210701-VirtualBox:~/linux/chap06$ echo 1; wait; echo 2
1
DONE1
[1]- 완료                ( echo START1; sleep 50; echo DONE1 )
DONE2
[2]+ 완료                ( echo START2; sleep 50; echo DONE2 )
2
u2210701@u2210701-VirtualBox:~/linux/chap06$
u2210701@u2210701-VirtualBox:~/linux/chap06$
```

2) **마지막 명령의 출력 결과**에 대해 설명한다. (실습교안 설명 참고)

(echo ... DONE1) &, (echo ... DONE2) & 두가지 명령은 모두 &입력으로 후면처리되어있다.

echo 1; wait; echo 2 명령에서 echo 1이 실행되어 1이 출력된 후, wait 명령으로 인해 실행중인 모든 프로세스가 끝나기를 기다린다. 따라서 후면처리된 명령들이 각각 50초에 걸쳐 실행이 완료된 후, 마지막 명령의 echo 2 가 실행되어 2가 출력되며 모든 프로세스 실행이 끝난다.