

리눅스시스템 Lab06

분반:001

학과:컴퓨터과학전공

학번:2312282

이름:임다희

1. ps 실습

1) p3-5의 명령 4개 각각 실행한 후, 터미널 창을 캡처한다.

```
u2312282@dahee-VirtualBox:~$ ps
  PID TTY          TIME CMD
 3379 pts/0    00:00:00 bash
 3389 pts/0    00:00:00 ps
u2312282@dahee-VirtualBox:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
u2312282    3379    3353  0 18:46 pts/0    00:00:00 bash
u2312282    3390    3379  0 18:46 pts/0    00:00:00 ps -f
```

```
u2312282@dahee-VirtualBox: ~
u2312282@dahee-VirtualBox:~$ ps aux
USER          PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1   0.6  0.5 166684 11880 ?        Ss   18:45   0:00 /sbin/init sp
root           2   0.0  0.0      0     0 ?        S    18:45   0:00 [kthreadd]
root           3   0.0  0.0      0     0 ?        S    18:45   0:00 [pool_workque
root           4   0.0  0.0      0     0 ?        I<   18:45   0:00 [kworker/R-rc
root           5   0.0  0.0      0     0 ?        I<   18:45   0:00 [kworker/R-rc
root           6   0.0  0.0      0     0 ?        I<   18:45   0:00 [kworker/R-sl
root           7   0.0  0.0      0     0 ?        I<   18:45   0:00 [kworker/R-ne
root           8   0.0  0.0      0     0 ?        I    18:45   0:00 [kworker/0:0-
root           9   0.2  0.0      0     0 ?        I    18:45   0:00 [kworker/0:1-
root          10   0.0  0.0      0     0 ?        I<   18:45   0:00 [kworker/0:0H
root          11   0.0  0.0      0     0 ?        I    18:45   0:00 [kworker/u2:0
root          12   0.0  0.0      0     0 ?        I<   18:45   0:00 [kworker/R-mm
root          13   0.0  0.0      0     0 ?        I    18:45   0:00 [rcu_tasks_kt
root          14   0.0  0.0      0     0 ?        I    18:45   0:00 [rcu_tasks_ru
root          15   0.0  0.0      0     0 ?        I    18:45   0:00 [rcu_tasks_tr
root          16   0.2  0.0      0     0 ?        S    18:45   0:00 [ksoftirqd/0]
root          17   0.1  0.0      0     0 ?        I    18:45   0:00 [rcu_preempt]
root          18   0.0  0.0      0     0 ?        S    18:45   0:00 [migration/0]
root          19   0.0  0.0      0     0 ?        S    18:45   0:00 [idle_inject/
root          20   0.0  0.0      0     0 ?        S    18:45   0:00 [cpuhp/0]
root          21   0.0  0.0      0     0 ?        S    18:45   0:00 [kdevtmpfs]
root          22   0.0  0.0      0     0 ?        I<   18:45   0:00 [kworker/R-in
```

```
u2312282@dahee-VirtualBox:~$ ps -ef
UID          PID     PPID  C  STIME TTY          TIME CMD
root           1         0  0   18:45 ?        00:00:00 /sbin/init splash
root           2         0  0   18:45 ?        00:00:00 [kthreadd]
root           3          2  0   18:45 ?        00:00:00 [pool_workqueue_release]
root           4          2  0   18:45 ?        00:00:00 [kworker/R-rcu_g]
root           5          2  0   18:45 ?        00:00:00 [kworker/R-rcu_p]
root           6          2  0   18:45 ?        00:00:00 [kworker/R-slub_]
root           7          2  0   18:45 ?        00:00:00 [kworker/R-netns]
root           8          2  0   18:45 ?        00:00:00 [kworker/0:0-events]
root           9          2  0   18:45 ?        00:00:00 [kworker/0:1-events]
root          10          2  0   18:45 ?        00:00:00 [kworker/0:0H-kblockd]
root          11          2  0   18:45 ?        00:00:00 [kworker/u2:0-events_unbound]
root          12          2  0   18:45 ?        00:00:00 [kworker/R-mm_pe]
root          13          2  0   18:45 ?        00:00:00 [rcu_tasks_kthread]
root          14          2  0   18:45 ?        00:00:00 [rcu_tasks_rude_kthread]
root          15          2  0   18:45 ?        00:00:00 [rcu_tasks_trace_kthread]
root          16          2  0   18:45 ?        00:00:00 [ksoftirqd/0]
root          17          2  0   18:45 ?        00:00:00 [rcu_preempt]
root          18          2  0   18:45 ?        00:00:00 [migration/0]
root          19          2  0   18:45 ?        00:00:00 [idle_inject/0]
root          20          2  0   18:45 ?        00:00:00 [cpuhp/0]
root          21          2  0   18:45 ?        00:00:00 [kdevtmpfs]
root          22          2  0   18:45 ?        00:00:00 [kworker/R-inet_]
```

2) ps 명령어의 역할과 출력 결과에 대해 설명한다.

ps 명령어는 현재 시스템 내에 존재하는 프로세스들의 실행 상태를 요약해서 출력한다.

위의 \$ps 명령어의 실행 결과에는 각 프로세스의 번호, CPU 시간, 명령어가 시작된 터미널, 실행되고 있는 명령어(프로그램) 이름이 출력된다.

현재 실행되고 있는 명령어로는 bash 셸, ps 명령어가 있음을 확인할 수 있다.

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

ps -f 옵션은 프로세스의 실행 상태 출력을 더욱 자세하게 해 주는 옵션이다. 기본 ps 명령어의 출력 내용과 더불어 프로세스 사용자의 이름, 부모 프로세스의 ID, 프로세스 실행의 우선순위까지 출력되는 것을 확인할 수 있다.

ps aux 옵션은 시스템 전체에서 실행되는 모든 프로세스를 리스팅하는 명령어이다. 주로 %CPU(사용율), %MEM(메모리 사용율), VSZ(가상 메모리 크기), RSS(실제 메모리 크기), STAT(프로세스 상태)와 같은 프로세스의 상태 정보를 보여준다.

ps -ef 옵션은 시스템 전체에서 실행되는 모든 프로세스를 리스팅하는 명령어이다. aux 옵션과 달리 부모 ID, 우선순위 등 ps -f에서 출력한 식별 정보를 보여준다.

2. pgrep 실습

1) p6의 명령 2개 각각을 실행한 후, 터미널 창을 캡처한다.

```
u2312282@dahee-VirtualBox:~$ pgrep bash
3379
u2312282@dahee-VirtualBox:~$ pgrep -l bash
3379 bash
u2312282@dahee-VirtualBox:~$
```

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

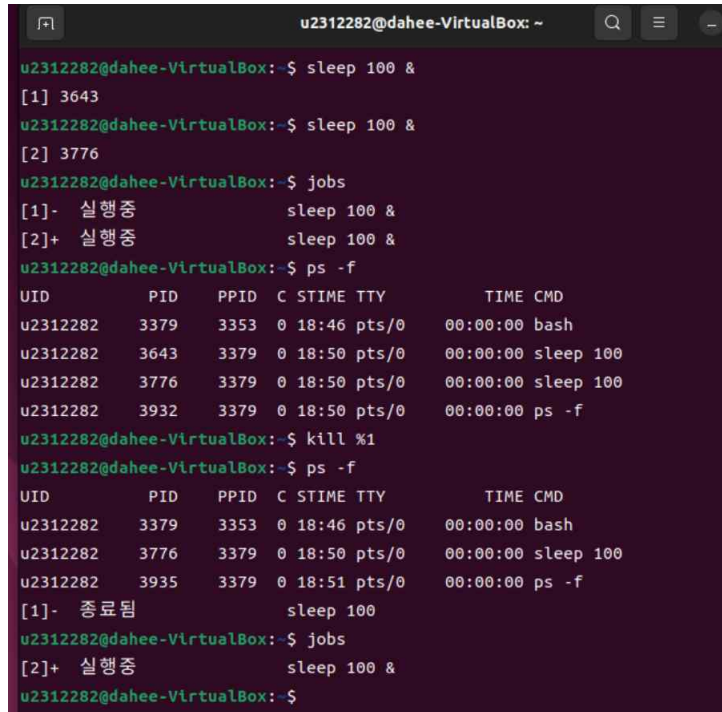
pgrep 명령어는 특정 옵션 및 패턴의 프로세스만을 리스팅하는 명령어이다.

\$ pgrep bash 와 같이 실행하면 bash를 실행하고 있는 프로세스의 PID를 출력한다.

\$ pgrep -l bash 와 같이 -l 옵션을 실행하면 bash를 실행하고 있는 프로세스의 PID와 함께 프로세스 이름을 출력할 수 있다.

3. 전면처리, 후면처리 실습

1) p7의 명령 6개 각각을 실행한 후, 터미널 창을 캡처한다.



```
u2312282@dahee-VirtualBox: ~  
u2312282@dahee-VirtualBox:~$ sleep 100 &  
[1] 3643  
u2312282@dahee-VirtualBox:~$ sleep 100 &  
[2] 3776  
u2312282@dahee-VirtualBox:~$ jobs  
[1]-  실행중           sleep 100 &  
[2]+  실행중           sleep 100 &  
u2312282@dahee-VirtualBox:~$ ps -f  
UID          PID     PPID  C  STIME TTY          TIME CMD  
u2312282     3379    3353  0  18:46 pts/0        00:00:00 bash  
u2312282     3643    3379  0  18:50 pts/0        00:00:00 sleep 100  
u2312282     3776    3379  0  18:50 pts/0        00:00:00 sleep 100  
u2312282     3932    3379  0  18:50 pts/0        00:00:00 ps -f  
u2312282@dahee-VirtualBox:~$ kill %1  
u2312282@dahee-VirtualBox:~$ ps -f  
UID          PID     PPID  C  STIME TTY          TIME CMD  
u2312282     3379    3353  0  18:46 pts/0        00:00:00 bash  
u2312282     3776    3379  0  18:50 pts/0        00:00:00 sleep 100  
u2312282     3935    3379  0  18:51 pts/0        00:00:00 ps -f  
[1]-  종료됨           sleep 100  
u2312282@dahee-VirtualBox:~$ jobs  
[2]+  실행중           sleep 100 &  
u2312282@dahee-VirtualBox:~$
```

2) 각 명령이 어떤 의미를 가지고 있는지 출력 결과를 중심으로 설명한다. (실습교안 설명 참고)

\$ sleep 100 & 은 sleep 100(100초 휴식)이라는 명령어를 후면 처리하겠다는 의미이다. 해당 명령어를 입력하면 [1] 3643과 같이 작업 번호와 프로세스 번호가 출력된다.

\$ sleep 100 &을 재차 입력해 같은 명령을 반복하면 [2] 3776과 같이 작업 번호와 프로세스 번호가 출력된다.

\$ jobs 명령어를 통해 각 프로세스들의 작업 상태를 확인할 수 있다. 현재 두 차례의 \$ sleep 100 & 명령어가 실행되고 있음을 확인할 수 있다.

\$ ps -f 명령어를 통해 현재 실행중인 프로세스들의 자세한 실행 상태들을 출력할 수 있다. 현재 실행중인 프로세스는 PID 3379의 bash 쉘, 3643의 sleep 100 명령어, 3776의 sleep 100 명령어, 3932의 ps -f 명령어가 있다.

\$ kill %1 명령어를 통해 1번 작업번호에 해당하는 프로세스를 강제로 종료시킬 수 있다.

재차 \$ ps -f 명령어를 통해 실행중인 프로세스를 확인하면 3643번의 sleep 100 명령어가 사라졌음을 확인할 수 있고, 아래에 1번 프로세스가 종료되었다는 메시지가 출력된다.

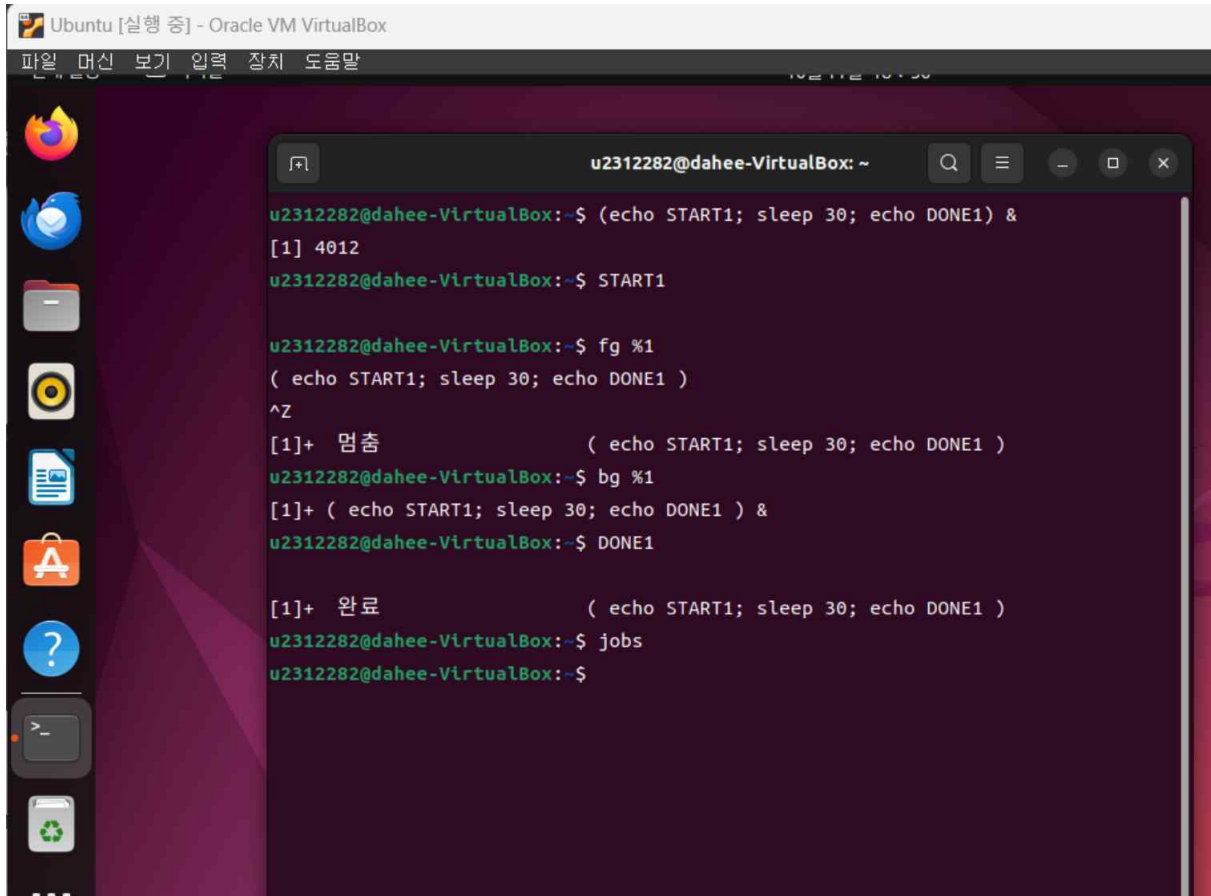
\$ job 명령어를 통해 프로세스들의 작업 상태를 다시 확인하면 2번 작업번호에 해당하는 프로세스만이 현재 실행중임을 확인할 수 있다.

3) 출력된 프로세스들의 부모-자식 관계를 설명하시오.

bash 쉘 프로세스(PID 3379)는 PID 3643의 sleep 100, PID 3776의 sleep 100 명령어, PID 3932의 부모 프로세스이다. 이는 세 개의 자식 프로세스의 PPID(부모의 프로세스 ID)가 3379로 동일함을 통해 알 수 있다.

4. 작업제어 실습

1) p8의 명령 4개 각각을 실행한 후, 터미널 창을 캡처한다.



```
u2312282@dahee-VirtualBox: ~  
u2312282@dahee-VirtualBox:~$ (echo START1; sleep 30; echo DONE1) &  
[1] 4012  
u2312282@dahee-VirtualBox:~$ START1  
u2312282@dahee-VirtualBox:~$ fg %1  
( echo START1; sleep 30; echo DONE1 )  
^Z  
[1]+  멈춤          ( echo START1; sleep 30; echo DONE1 )  
u2312282@dahee-VirtualBox:~$ bg %1  
[1]+ ( echo START1; sleep 30; echo DONE1 ) &  
u2312282@dahee-VirtualBox:~$ DONE1  
[1]+  완료          ( echo START1; sleep 30; echo DONE1 )  
u2312282@dahee-VirtualBox:~$ jobs  
u2312282@dahee-VirtualBox:~$
```

2) 각 명령이 어떤 의미를 가지고 있는지 출력 결과를 중심으로 설명한다. (실습교안 설명 참고)

\$ (echo START1; sleep 30; echo DONE1) & 명령은 START1 메세지 출력-30초간 프로세스 실행 중지-DONE1 메세지 출력을 순차적으로 처리하는 명령을 후면 처리하겠다는 의미이다.

[1] 4012는 해당 명령어의 작업 번호와 프로세스 번호이다. 명령어를 실행하면 즉시 START1 메세지가 출력된다.

\$ fg %1 명령은 작업번호 1번에 해당하는 후면 작업을 전면 작업으로 전환시킨다. (echo START1; sleep 30; echo DONE1) 명령어 작업이 전면 작업으로 전환되었음을 알 수 있다.

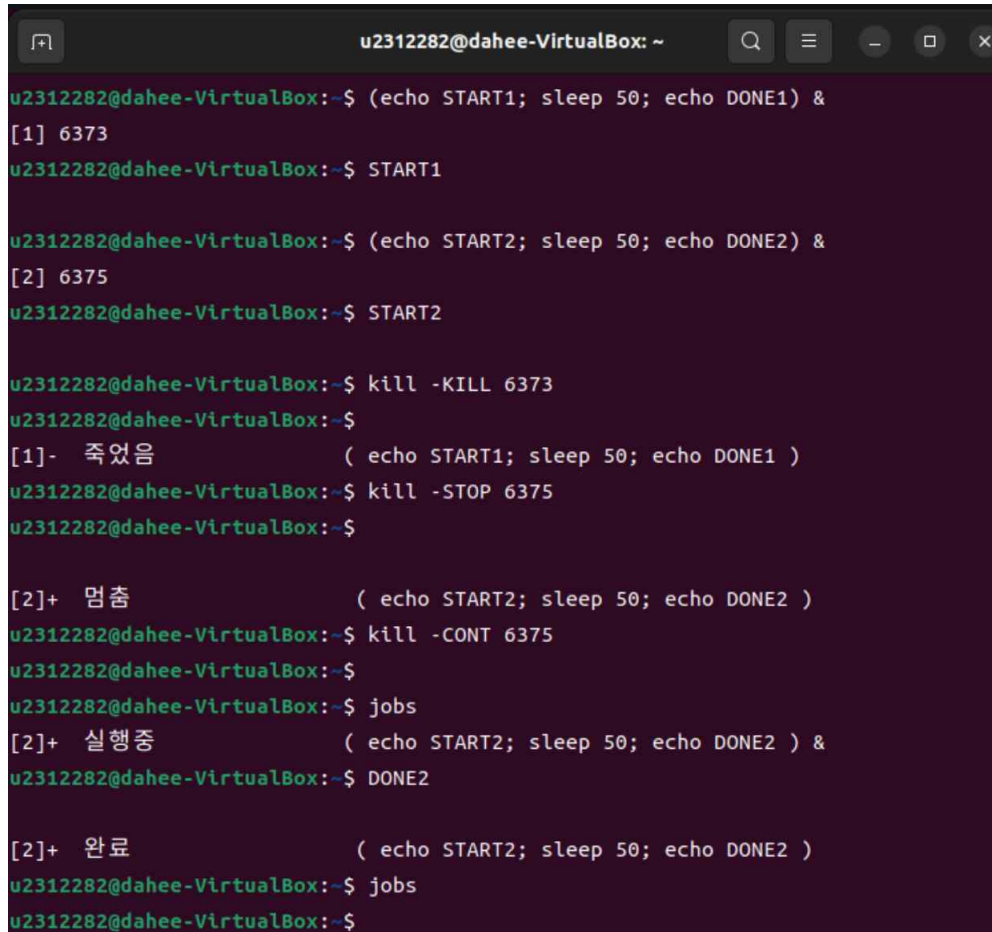
^Z를 통해 전면 실행중인 작업을 중지시킬 수 있다. 작업이 중지되면 작업이 멈추었음을 나타내는 메세지가 출력된다.

전면 작업이 중지되면 해당 작업을 \$ bg %1 명령어를 통해 후면 작업으로 전환할 수 있다. [1]+ (echo START1; sleep 30; echo DONE1) & 을 통해 해당 작업이 후면작업이 되었음을 확인할 수 있다.

30초가 지나면 작업이 재개되어 DONE1 메세지가 출력되고 작업이 완료되었음을 알리는 출력문이 출력된다. \$ jobs 명령어를 통해 확인해 보면 현재 진행중인 프로세스가 없음을 알 수 있다.

5. 작업제어 실습

1) p9의 명령 6개 각각을 실행한 후, 터미널 창을 캡처한다.



```
u2312282@dahee-VirtualBox: ~  
u2312282@dahee-VirtualBox:~$ (echo START1; sleep 50; echo DONE1) &  
[1] 6373  
u2312282@dahee-VirtualBox:~$ START1  
  
u2312282@dahee-VirtualBox:~$ (echo START2; sleep 50; echo DONE2) &  
[2] 6375  
u2312282@dahee-VirtualBox:~$ START2  
  
u2312282@dahee-VirtualBox:~$ kill -KILL 6373  
u2312282@dahee-VirtualBox:~$  
[1]-  죽었음          ( echo START1; sleep 50; echo DONE1 )  
u2312282@dahee-VirtualBox:~$ kill -STOP 6375  
u2312282@dahee-VirtualBox:~$  
  
[2]+  멈춤            ( echo START2; sleep 50; echo DONE2 )  
u2312282@dahee-VirtualBox:~$ kill -CONT 6375  
u2312282@dahee-VirtualBox:~$  
u2312282@dahee-VirtualBox:~$ jobs  
[2]+  실행중          ( echo START2; sleep 50; echo DONE2 ) &  
u2312282@dahee-VirtualBox:~$ DONE2  
  
[2]+  완료            ( echo START2; sleep 50; echo DONE2 )  
u2312282@dahee-VirtualBox:~$ jobs  
u2312282@dahee-VirtualBox:~$
```

2) 각 명령이 어떤 의미를 가지고 있는지 출력 결과를 중심으로 설명한다. kill 명령어 설명의 경우, 각 옵션들에 대한 의미가 무엇인지에 대해 설명도 포함한다.

\$ (echo START1; sleep 50; echo DONE1) & 명령어를 통해 START1 메시지를 출력하고 프로세스를 50초간 종료한 후 DONE1 메시지를 출력한다. 이는 작업번호 1번, 프로세스 번호 6373을 가지는 후면 작업이다. 실행 즉시 START1 메시지가 출력된다.

\$ (echo START2; sleep 50; echo DONE2) & 명령어를 통해 START2 메시지를 출력하고 프로세스를 50초간 종료한 후 DONE2 메시지를 출력한다. 이는 작업번호 2번, 프로세스 번호 6375를 가지는 후면 작업이다. 실행 즉시 START2 메시지가 출력된다.

\$ kill -KILL 6373을 통해 첫번째 프로세스를 강제종료시킨다. 명령 실행 후 해당 프로세스가 죽었다는 메시지가 출력된다.

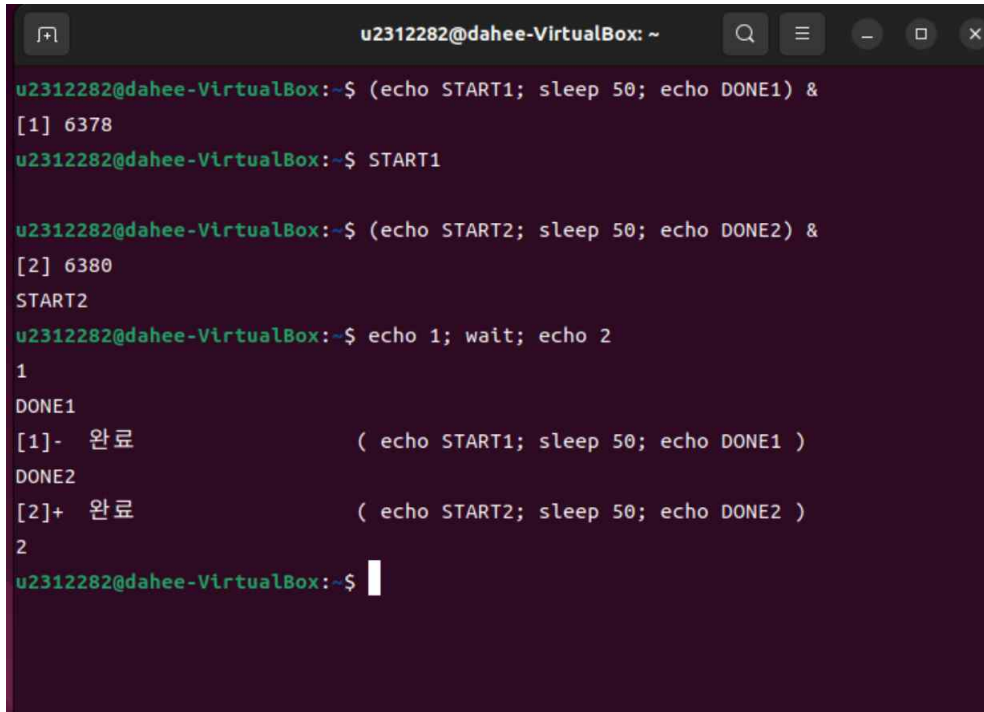
\$ kill -STOP 6375를 통해 두번째 프로세스를 중지시킨다. 명령 실행 후 해당 프로세스가 멈추었다는 메시지가 출력된다.

\$ kill -CONT 6375를 통해 중지되었던 두 번째 프로세스를 다시 재개한다. \$ jobs를 통해 프로세스의 실행 상태를 확인하면 두 번째 프로세스가 실행 중임을 알 수 있다. 50초가 지나면 DONE2 메시지가 출력되고 해당 작업이 완료되었음을 알리는 메시지가 출력된다.

마지막으로 \$ jobs 명령어를 실행하면 현재 실행중인 프로세스가 없음을 알 수 있다.

6. 명령어 열/그룹 실습

1) p10의 명령 3개 각각을 실행한 후, 터미널 창을 캡처한다.



```
u2312282@dahee-VirtualBox: ~  
u2312282@dahee-VirtualBox:~$ (echo START1; sleep 50; echo DONE1) &  
[1] 6378  
u2312282@dahee-VirtualBox:~$ START1  
  
u2312282@dahee-VirtualBox:~$ (echo START2; sleep 50; echo DONE2) &  
[2] 6380  
START2  
u2312282@dahee-VirtualBox:~$ 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  
u2312282@dahee-VirtualBox:~$
```

2) 마지막 명령의 출력 결과에 대해 설명한다. (실습교안 설명 참고)

\$ (echo START1; sleep 50; echo DONE1) &, \$ (echo START2; sleep 50; echo DONE2) & 를 통해 두 개의 후면 작업을 실행한다.

\$ echo 1; wait; echo 2 명령어를 실행하면 우선 1을 출력하고, wait 명령어에 의해 모든 자식 프로세스가 끝날 때까지 기다렸다가 2를 출력하게 되므로 1번 작업이 완료되어 DONE1이 출력되고 2번 작업이 완료되어 DONE2가 출력된 다음에야 2가 출력된다.