18. 프로세스

프로세스(process)란 현재 컴퓨터 시스템의 자원을 점유하고 실행중인 프로그램을 뜻한다. 최근의 운영체제는 여러 개의 프로세스를 동시에 실행(Multi Processing)할 수 있다.

유닉스 시스템의 모든 프로세스는 각기 고유한 번호를 가지게 되는데 이를 PID(Process Identification Number)라고 한다. PID외에도 PPID(Parent PID), UID(User ID), GID(Group ID) 정보를 포함하고 있다.

Ubuntu 시스템은 부팅 될 때 모든 프로세스의 조상격이 되는 systemd라는 데몬이 실행되며 이 프로세스에 의해 다른 프로세스가 생성된다.

```
kihee@kihee-VirtualBox:~$ pstree
systemd─┬─ModemManager───2*[{ModemManager}]
        ├─NetworkManager───2*[{NetworkManager}]
        ├─accounts-daemon───2*[{accounts-daemon}]
        ├─acpid
        ├─avahi-daemon───avahi-daemon
        ├─colord───2*[{colord}]
        ├─cron
        ├─cups-browsed───2*[{cups-browsed}]
        ├─cupsd
        ├─dbus-daemon
```

18. 1 프로세스 목록보기

ps명령은 현재 실행 중인 프로세스에 대한 정보를 출력한다.

ps [옵션]

ps 명령의 옵션은 다음과 같다.

옵션	설명
-e	시스템에 실행중인 모든 프로세스 정보 출력
-f	각 프로세스에 대한 자세한 정보 출력
-u uid	특정 사용자에 대한 모든 프로세스 출력

다음 명령을 각각 입력하고 출력된 결과를 살펴보자.

```
kihee@kihee-VirtualBox:~$ ps
    PID TTY          TIME CMD
   2175 pts/0    00:00:00 bash
   2919 pts/0    00:00:00 ps
```

아래의 ps 명령은 kihee(UID) 계정에 의해 bash(CMD)가 2149번 프로세스(PID)로 첫 번째 가상 콘솔(pts/0)에서 구동 되고 있으며 2149번 프로세스(PPID)가 부모 프로세스임을 나타낸다.

```
kihee@kihee-VirtualBox:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
kihee       2175    2149  0 23:07 pts/0    00:00:00 bash
kihee       2920    2175  0 23:22 pts/0    00:00:00 ps -f
```

-e 옵션은 시스템에서 실행중인 모든 프로세스들을 출력한다. TTY값이 "?"인 것은 시스템이 실행시킨 프로세스들로 대부분 데몬이다.

```
kihee@kihee-VirtualBox:~$ ps -ef
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 23:06 ?        00:00:01 /sbin/init splash
root           2       0  0 23:06 ?        00:00:00 [kthreadd]
root           3       2  0 23:06 ?        00:00:00 [rcu_gp]
root           4       2  0 23:06 ?        00:00:00 [rcu_par_gp]
root           5       2  0 23:06 ?        00:00:00 [slub_flushwq]
root           6       2  0 23:06 ?        00:00:00 [netns]
root           7       2  0 23:06 ?        00:00:00 [kworker/0:0-events]
root           8       2  0 23:06 ?        00:00:00 [kworker/0:0H-events_highpri]
root          10       2  0 23:06 ?        00:00:00 [mm_percpu_wq]
root          11       2  0 23:06 ?        00:00:00 [rcu_tasks_kthread]
root          12       2  0 23:06 ?        00:00:00 [rcu_tasks_rude_kthread]
root          13       2  0 23:06 ?        00:00:00 [rcu_tasks_trace_kthread]
root          14       2  0 23:06 ?        00:00:00 [ksoftirqd/0]
root          15       2  0 23:06 ?        00:00:00 [rcu_preempt]
root          16       2  0 23:06 ?        00:00:00 [migration/0]
root          17       2  0 23:06 ?        00:00:00 [idle_inject/0]
root          19       2  0 23:06 ?        00:00:00 [cpuhp/0]
```

18. 2 프로세스 강제종료

응답이 없는 프로세스나 불필요한 프로세스를 강제로 종료하기 위해서는 해당 프로세스의 PID를 알아야 한다. 경우에 따라서는 PPID로 부모 프로세스를 종료해야 할 때도 있다. 이때 부모프로세스를 종료하면 자식 프로세스도 함께 종료된다.

kill [시그널] PID...

시그널	설명
-9	프로세스를 강제로 종료
-15	소프트종료로, 프로세스가 관련 파일들을 정리하고, 프로세스를 종료한다.

프로세스 실습을 위해 sleep 명령을 사용하여 프로세스를 하나 생성할 것이다. sleep 명령은 인자로 주어진 값(초 단위) 만큼 대기한다. sleep 명령을 실행하면 아래와 같이 더 이상 터미널을 사용할 수 없게 된다.

```
kihee@kihee-VirtualBox:~$ sleep 1000
```

위의 명령을 실행한 후에도 터미널을 사용하고자 한다면 위의 명령을 백그라운드로 실행해야 한다. 우선 Ctrl + c 키를 눌러 프로세스를 종료한다. Ctrl + c 키는 현재 포그라운드로 실행 중인 프로세스에 중지 시그널을 전송함으로써 프로세스를 강제로 중지시킨다.

```
kihee@kihee-VirtualBox:~$ sleep 1000
^C
kihee@kihee-VirtualBox:~$
```

백그라운드로 명령을 실행하고자 한다면 명령행의 뒤에 &를 붙여 명령을 실행한다. &를 붙여 명령을 실행하면 백그라운드로 명령이 실행되며 다음과 같이 1번 작업, 2961번 프로세스로 sleep 1000 명령이 실행되었음을 보여준다.

```
kihee@kihee-VirtualBox:~$ sleep 1000 &
[1] 2961
kihee@kihee-VirtualBox:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
kihee       2175    2149  0 23:07 pts/0    00:00:00 bash
kihee       2961    2175  0 23:34 pts/0    00:00:00 sleep 1000
kihee       2963    2175  0 23:34 pts/0    00:00:00 ps -f
```

아래는 kill 명령으로 sleep 1000 명령에 의해 발생된 2961번 프로세스를 강제적으로 종료하는 명령 예이다. kill 명령 사용시 시그널을 지정하지 않는다면 -9 시그널이 전송되어 프로세스를 강제적으로 종료하게 된다.

```
kihee@kihee-VirtualBox:~$ kill 2961
[1]+  종료됨                  sleep 1000
kihee@kihee-VirtualBox:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
kihee       2175    2149  0 23:07 pts/0    00:00:00 bash
kihee       2967    2175  0 23:37 pts/0    00:00:00 ps -f
```

18. 3 포그라운드와 백그라운드 프로세스

사용자가 명령을 입력하면 그 결과가 출력될 때까지 기다려야 하는데 이런 경우 포그라운드 처리라고 하며 명령을 실행시키는 일반적인 방식이다.

백그라운드 처리는 명령의 처리와 관계없이 프롬프트가 출력되어 사용자는 다른 작업을 계속할 수 있다. 백그라운드 처리는 명령의 실행시간이 많이 걸릴 것으로 예상되거나 명령을 실행시킨 후 다른 작업을 할 필요가 있을 때 많이 사용된다.

$ sleep 1000	전면작업 처리
$ sleep 1000 &	후면작업 처리

명령을 실행하면 전면작업으로 실행되므로 더 이상 터미널을 사용할 수 없게 된다.

반면 명령을 실행함에 있어서 명령행에 "&"를 붙이면 후면작업으로 실행된다. 후면작업으로 작업을 실행하면 터미널을 계속적으로 사용할 수 있다.

18. 4 포그라운드와 백그라운드의 작업 제어

현재 실행 중인 백그라운드 명령을 모두 보여주는 명령이 jobs이다.

$ jobs [옵션] [%작업번호]

옵션

-p : 프로세스 ID를 출력한다.
-l : (소문자 L) 나열되는 각 작업에 대한 추가 정보를 제공한다.
이 정보에는 작업 번호, 프로세스 ID, 상태, 작업을 시작한 명령이 포함된다.

작업번호 지정 방법

%번호 : 해당 번호의 작업 정보 출력
%+ 또는 %% : 작업 순서가 +인 작업 정보 출력
%- : 작업 순서가 –인 작업 정보 출력

```
kihee@kihee-VirtualBox:~$ jobs -l
[1]   2970 실행중               sleep 1000 &
[2]-  2972 실행중               sleep 2000 &
[3]+  2974 실행중               sleep 3000 &
```

위의 예에서는 세 개의 작업이 있으며 +가 표시된 3번 작업이 가장 최근에 실행된 작업이다. 또한 세 개의 작업 모두가 후면작업으로 작업중이다. 후면작업을 전면작업으로 전환하는 명령은 fg 명령이다.

$ fg %작업번호

```
kihee@kihee-VirtualBox:~$ fg %1
sleep 1000
```

반대로 전면작업을 후면작업으로 전환시키는 명령은 bg 명령이다. 하지만 현재 작업이 전면작업으로 진행중이므로 터미널을 사용할 수 없는 상태이다. 그러므로 우선 Ctrl + z 키를 눌러서 현재 진행중인 전면작업을 일시 중지시켜야 한다. 이후 프롬프트상에서 bg 명령을 이용하여 후면작업으로 전환하도록 한다.

Ctrl + z를 눌러 전면작업을 일시중지 시킨 후
$ bg %작업번호

```
kihee@kihee-VirtualBox:~$ fg %1
sleep 1000
^Z
[1]+  멈춤                  sleep 1000
kihee@kihee-VirtualBox:~$ bg %1
[1]+ sleep 1000 &
kihee@kihee-VirtualBox:~$
```