6장 리눅스의 프로세스 관리

- 프로세스란?
- 셸 프로세스와 사용자 프로세스
- 프로세스 확인과 강제 종료
- 전면(Foreground)와 후면 (Background)

한빛미디어(주)

프로세스 (process)와 프로그램 (program)

- 프로세스
  - 실행 중인 상태의 프로그램이다.
  - 동일한 프로그램으로 여러 개의 프로세스를 생성할 수 있다.
    - 각 프로세스를 프로그램의 인스턴스(instance)라고 한다.
- 프로그램
  - 파일 형태로 저장되어 있다.
  - 명령어 코드를 담고 있다.

사용자가 프로세스를 생성하는 방법

- 셸 프롬프트 상에서 프로그램을 지정하여 실행
- 실행 중인 사용자 프로세스를 통해서 프로그램을 실행

- 위의 두 가지 방법은 사실 동일하다.
  - 셸 프롬프트를 보여주고 있는 셸 프로세스가 이미 실행 중인 프로세스이기 때문이다.
  - 다른 점은 셸 프로세스는 로그인 과정에서 시스템에 의해 실행된다는 것이다.

셸을 통해서 프로세스 생성하기 (프로그램 실행하기)

```
$ ls -l
-rwxr-xr-x    1 kimyh    graduate    13508 Nov 18 23:36 hello
$ ./hello
hello world!
$
```

두 개 이상의 프로세스를 동시에 생성하기

- 여러 개의 명령어 라인을 ';'을 사용하여 한 줄에 입력한다.

```
$ ls -l
-rwxr-xr-x    1 kimyh    graduate    13508 Nov 18 23:36 hello
$ ./hello
hello world!
$ ls -l ; ./hello
-rwxr-xr-x    1 kimyh    graduate    13508 Nov 18 23:36 hello
hello world!
$
```

서로 협력하는 두 개 이상의 프로세스를 생성하기

- 파이프(|)를 사용하면 서로 협력하는 두 개 이상의 프로세스를 생성할 수 있다.
- 명령어 라인 상에서 앞에 놓여진 프로세스의 표준 출력이 바로 다음에 있는 프로세스의 표준 입력으로 연결된다.

```
$ ls /bin
arch            csh             gawk            mail            red             tcsh
ash             cut             gawk-3.1.0      mkdir           rm              touch
…  (생략했음)
cp              false           login           ps              sync
cpio            fgrep           ls              pwd             tar
$ ls /bin | wc -w
      85
$
```

※"ls /bin"이 표준 출력하는 내용이 "wc –w"의 표준 입력으로 연결된다.

Section 02 셸 프로세스와 사용자 프로세스

셸 프로세스 (shell process)

- 시스템과 사용자의 사이에서 중간자 역할을 하는 프로세스
- 시스템에 로그인 했을 때 가장 먼저 보게 되는 프로세스
- 셸 프로세스의 소유주는 사용자이다

사용자 프로세스 (user process)

- 사용자가 셸 프롬프트 상에서 명령어를 입력하여 생성한 프로세스
- 사용자가 실행한 프로세스가 생성한 또 다른 프로세스
- 프로세스의 주인은 이를 실행한 사용자이다.

- 프로그램 파일의 소유주와 프로세스 소유주는 서로 다를 수 있다.

  예) “ls” 명령은 “ls”라는 이름의 프로그램으로 저장되어 있고 이 프로그램(파일 형태)의 소유주는 관리자이다.

실행 중인 프로세스의 목록 확인하기 (1)

- "ps" 명령을 사용하여 실행 중인 프로세스의 목록을 확인할 수 있다.

```
$ ps -f
UID        PID  PPID  C STIME TTY          TIME CMD
kimyh     2261  2260  0 Nov18 pts/2    00:00:00 -bash
kimyh     6319  2261  0 00:40 pts/2    00:00:00 ps -f
$
```

- 사용자가 소유주인 프로세스는 2개

  "bash"는 셸 프로세스이고, "ps –f"는 현재의 프로세스 목록을 보여주는 프로세스이다.

  "ps –f" 프로세스는 출력을 보여주는 시점에서는 존재했으나, 출력이 끝난 지금은 아마 종료되어 없을 것이다.

실행 중인 프로세스의 목록 확인하기 (2)

```
$ ps -f
UID        PID  PPID  C STIME TTY          TIME CMD
kimyh     2261  2260  0 Nov18 pts/2    00:00:00 -bash
kimyh     6319  2261  0 00:40 pts/2    00:00:00 ps -f
$
```

- UID (User ID) : 프로세스 소유주의 사용자 식별 번호이다.
- PID (Process ID) : 프로세스의 식별 번호이다.
- PPID (Parent Process ID) : 부모 프로세스의 식별 번호이다.

  모든 프로세스는 자신을 생성해준 부모 프로세스가 있다.

  단 한 개 예외가 있다. (init 프로세스)

  대부분의 사용자 프로세스는 셸 프로세스가 부모 프로세스가 된다.

  위의 예에서 “ps –f”의 PPID는 “bash”의 PID와 같다.

프로세스의 가계도 확인해보기

- 모든 프로세스는 최초의 "init" 프로세스부터 시작된다.
- "ps" 명령과 "grep" 명령을 사용하여 init 프로세스까지 거슬러 가본다.

```
$ ps -ef | grep 'kimyh' | grep 'bash'
kimyh     2261  2260  0 Nov18 pts/2    00:00:00 -bash
kimyh     7333  2261  0 01:00 pts/2    00:00:00 grep bash
$ ps -ef | grep '2260' | head -1
kimyh     2260  2258  0 Nov18 ?        00:00:00 /usr/local/ssh/sbin/sshd
$ ps -ef | grep '2258' | head -1
root      2258   534  0 Nov18 ?        00:00:00 /usr/local/ssh/sbin/sshd
$ ps -ef | grep '534' | head -1
root       534     1  0 Sep11 ?        00:02:44 /usr/local/ssh/sbin/sshd
$ ps -ef | grep '1' | head -1
root         1     0  0 Sep11 ?        00:00:38 init [3]
$
```

최초의 프로세스인 init는 부모 프로세스가 없다.

프로세스 확인

앞에서 예로 든 것과 같이 "ps" 명령을 사용한다.

[예제] 동일한 계정을 사용하여 두 개의 터미널로 접속

```
[1]$ tty
/dev/pts/2
[1]$ ps -f
UID        PID  PPID  C STIME TTY          TIME CMD
kimyh     2261  2260  0 Nov18 pts/2    00:00:00 -bash
kimyh     8528  2261  0 01:14 pts/2    00:00:00 ps -f
[1]$ ps -ef | grep 'kimyh'
kimyh     2260  2258  0 Nov18 ?        00:00:00 /usr/local/ssh/sbin/sshd
kimyh     2261  2260  0 Nov18 pts/2    00:00:00 -bash
kimyh     8250  8238  0 01:11 ?        00:00:00 /usr/local/ssh/sbin/sshd
kimyh     8251  8250  0 01:11 pts/3    00:00:00 -bash
kimyh     8535  2261  0 01:14 pts/2    00:00:00 ps -ef
kimyh     8536  2261  0 01:14 pts/2    00:00:00 grep kimyh
[1]$ kill 8250
[1]$ ps -ef | grep 'kimyh'
kimyh     2260  2258  0 Nov18 ?        00:00:00 /usr/local/ssh/sbin/sshd
kimyh     2261  2260  0 Nov18 pts/2    00:00:00 -bash
kimyh     8946  2261  0 01:19 pts/2    00:00:00 ps -ef
kimyh     8947  2261  0 01:19 pts/2    00:00:00 grep kimyh
[1]$
```

```
[2]$ tty
/dev/pts/3
[2]$ ps -f
UID        PID  PPID  C STIME TTY          TIME CMD
kimyh     8251  8250  0 01:11 pts/3    00:00:00 -bash
kimyh     8295  8251  0 01:11 pts/3    00:00:00 ps -f
[2]$
```

- **두 개의 연결 상태**
  - 각각 pts/2, pts/3으로 표현된다
- **프로세스의 소유주**
  - 계정이 동일하므로 소유주의 ID 역시 동일하다.
    하지만, 연결 상태가 다르다. (pts/2, pts/3)
- **프로세스 강제 종료**
  - pts/2에서 kill 명령을 사용하여 pts/3에 속해 있는 8250번 프로세스를 종료한다.

전면과 후면

- 실행 중인 프로세스는 각각 전면 프로세스와 후면 프로세스로 나눌 수 있다.
  전면이냐 후면이냐 하는 구분은 언제든지 바뀔 수 있다.

- 전면 프로세스
  사용자의 표준 입력을 받을 수 있는 프로세스
  하나의 연결 상태에서 한 개의 프로세스만 전면 프로세스가 될 수 있다.

※ 표준 출력은 모든 프로세스가 가능하다.

- 후면 프로세스
  실행 중이지만 사용자의 표준 입력을 받을 수 없는 프로세스
  하나의 연결 상태에서 전면 프로세스를 제외한 나머지 사용자 프로세스들이 모두 포함된다.

전면과 후면 프로세스 전환 예제 (1)

관련 명령어 : ps, jobs, bg, fg

```
$ cat > 111 &
[1] 9268
$ cat > 222 &
[2] 9269

[1]+  Stopped                 cat >111
$ cat > 333 &
[3] 9280

[2]+  Stopped                 cat >222
$
```

※ 명령어 라인의 마지막에 &를 입력하면 해당 명령은 실행되면서 후면으로 돌려진다.

※ jobs 명령은 후면에서 실행 중인 작업의 목록을 보여준다.

```
$ jobs
[1]   Stopped                 cat >111
[2]-  Stopped                 cat >222
[3]+  Stopped                 cat >333
$ kill %2

[2]-  Stopped                 cat >222
$ jobs
[1]   Stopped                 cat >111
[2]-  Terminated              cat >222
[3]+  Stopped                 cat >333
$
```

전면과 후면 프로세스 전환 예제 (2)

```
$ cat > 111
Ctrl+Z
[1]+  Stopped                 cat >11
1
$ jobs
[1]+  Stopped                 cat >11
1
$ cat > 222
Ctrl+Z
[2]+  Stopped                 cat >22
2
$ jobs
[1]-
  Stopped                 cat >111
[2]+  Stopped                 cat >22
2
$
```

※ 전면 프로세스는 Ctrl+Z를 입력하여 후면으로 돌릴 수 있다.
※ fg 명령어를 사용하여 후면의 작업을 전면으로 돌릴 수 있다.

```
$ fg
cat >222

[2]+  Stopped                 cat >22
2
$ fg 1
cat >111

[1]+  Stopped                 cat >11
1
$ jobs
[1]+  Stopped                 cat >11
1
[2]-
  Stopped                 cat >222
$ fg
cat >111
$
```