redhat CentOS

ENTERPRISE LINUX ADMIN GUIDE

프로세스 관리

단원목표

- 프로세스 정보
- PID / PPID
- 하드웨어 리소스
- 프로세스 관리
- 프로세스 모니터링

프로세스 정보

- 프로세스 정보

  /proc/PID

1 프로세스(process)란?

프로세스(process)란? 실행중인 프로그램이다.

DISK(한글.exe) ----> MEM(한글 process) ----> CPU(중앙 처리 장치)

프로세스는 프로그램이 메모리에 적재되어 실제로 실행되고 있는 상태를 말한다. 프로세스는 컴퓨터 내에서 실행중인 프로그램의 인스턴스이다. 여러 명의 사용자들에 의해 공유되고 있는 응용프로그램은 일반적으로 각 사용자들의 실행단계에서 하나의 프로세스를 갖는다. 프로세스는 자프로세스라고도 불리는 서브프로세스를 시작시킬 수 있다. 자프로세스는 부프로세스의 복제로서 부프로세스의 자원을 일부 공유하는데, 부프로세스가 종료되면 더 이상 존재할 수 없다. 프로세스들은 몇 가지 IPC 방식을 통하여 정보를 교환하거나 그들의 연산을 동기화할 수 있다.

모든 프로그램은 실행될 때 하나 이상의 프로세스를 갖는다. 하나의 프로세스에는 여러 명령어와 카운트, CPU 레지스터, 그리고 루틴 인자, 복귀 주소, 저장된 변수 등의 데이터 스택이 포함되어 있다. 각 프로세스는 고유의 권한과 책임을 가지고 서로 통신하며, 시스템에서 동작중인 하나의 프로세스가 잘못된 연산을 수행하여 에러를 일으키더라도 다른 프로세스는 정상적으로 작동한다. 개별 프로세스는 자신의 가상 주소 공간에서 실행되며, 커널이 제공하는 인터페이스를 통해서만 다른 프로세스와 연동할 수 있다.

프로세스는 여러 가지 자원을 사용한다. 프로세스는 해당 명령을 수행하기 위해 운영체제에 따라 CPU를 점유할 수 있다. 명령어와 데이터를 저장하기 위해 물리적인 메모리를 사용한다. 프로세스는 운영체제의 제어를 받으면서 실행(running), 대기(waiting), 중단(stopped), 좀비(zombie) 중 한 상태에 있게 된다.

(1) 자식프로세스(Child Process) & 부모프로세스(Parent Process)

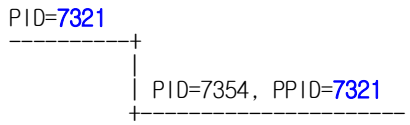
- PID(Process Identification) : 프로세스가 시작할 때 할당받는 프로세스 식별번호
- PPID(Parent Process Identification) : 부모 프로세스 식별번호 (서브 프로세스를 실행시킨 프로세스)

 # ps

```
  PID TTY          TIME CMD
 7321 pts/3    00:00:00 bash
```

 # bash
 # ps -f

```
UID        PID  PPID  C STIME TTY          TIME CMD
root      7321  7319  0 01:19 pts/3    00:00:00 -bash
root      7354  7321  0 01:19 pts/3    00:00:00 bash
```



 # ps -ef | more

```
UID        PID  PPID  C          STIME TTY      TIME     CMD
root         1     0  0           Jan26 ?       00:00:00 init [5]
root         2     1  0           Jan26 ?       00:00:00 [migration/0]
root         3     1  0           Jan26 ?       00:00:00 [ksoftirqd/0]
root         4     1  0           Jan26 ?       00:00:00 [watchdog/0]
root         5     1  0           Jan26 ?       00:00:00 [events/0]
root         6     1  0           Jan26 ?       00:00:00 [khelper]
root         7     1  0           Jan26 ?       00:00:00 [kthread]
```

(ps -ef 명령어 출력 결과 해석)

```
--------------------------------------------------------------------------------
root            UID, 프로세스를 실행시킨 사용자, 프로세스의 주인
1               PID, 프로세스 아이디, 프로세스 식별 번호
0               PPID, 부모 프로세스 아이디
0               C, 현재 사용되지 않는 필드
Jan26           STIME(Start Time), 프로세스 실행 시작 시간
?               TTY, 제어 터미널, 프로세스가 실행된 터미널
00:00:00        TIME, CPU 사용 누적 시간
init [5]        CMD, 명령어(옵션 + 인자 포함)
--------------------------------------------------------------------------------
```

[참고] 데몬(Daemon)의 정의

Deamon 이란?

시스템(System)을 위해 또는 서비스(Service)를 위해 백그라운드에서 동작하는 프로세스이다.
커널상에서 백그라운드 모드로 작동하여 비활성화 상태에서 요청이 있을 때만 동작하는 프로세스를 말한다. 커널상에 백그라운드 모드로 실행되어 작동하지 않고 있을 때는 CPU에 부하를 주지 않지만 메모리의 공간은 차지하고 있으므로 데몬이 커널상에 많이 존재한다면 시스템의 자원을 많이 사용하게 된다.

```
EX) 웹 데몬     ----> httpd
    메일 데몬   ----> sendmail
    ftp 데몬    ----> vsftpd
    telnet 데몬 ----> telnetd
```

2 프로세스의 정보 (Process Information)

(1) 프로세스 정보가 존재하는 디렉토리

/proc 라는 디렉토리에 각 프로세스에 해당되는 PID디렉토리들이 있다.
/proc 라는 디렉토리는 커널 메모리를 마운트 한 것이다.

 # ls /proc

```
1     2525  3065  3780  4103  4330  4431  6165        filesystems  net
10    2526  3224  3793  4108  4334  4443  635         fs           partitions
11    2527  3250  3794  4109  4335  4476  6361        ide          schedstat
12    2528  3278  3802  4110  4337  4478  658         interrupts   scsi
13    2562  3541  3808  4111  4355  4480  689         iomem        self
14    2564  3543  3817  4112  4359  4530  7           ioports      slabinfo
15    2566  3565  3842  4115  4361  5     8           irq          stat
16    2568  3569  3870  4116  4365  507   9           kallsyms     swaps
17    2570  3579  3887  4209  4368  572   acpi        kcore        sys
18    26    3590  3892  4211  4370  573   asound      key-users    sysrq-trigger
19    27    3619  3919  4212  4371  579   buddyinfo   keys         sysvipc
1932  28    3620  3937  4236  4378  580   bus         kmsg         tty
196   283   3621  3945  4238  4391  581   cmdline     loadavg      uptime
197   284   3622  3955  4254  4396  582   cpuinfo     locks        version
198   285   3631  3964  4286  4401  583   crypto      mdstat       vmcore
199   286   3655  3997  4288  4406  5877  devices     meminfo      vmmemctl
2     287   3677  4     4314  4408  599   diskstats   misc         vmstat
202   288   3696  4006  4316  4423  6     dma         modules      zoneinfo
204   289   3702  4014  4322  4427  6159  driver      mounts
25    29    3732  4041  4323  4428  6163  execdomains mpt
2524  3     3770  4042  4328  4430  6164  fb          mtrr
```

 # gedit &

```
[1] 6180
```

 # ps

```
  PID TTY          TIME CMD
 6165 pts/1    00:00:00 bash
 6180 pts/1    00:00:00 gedit
```

 # ls -ld /proc/6180

```
dr-xr-xr-x 5 root root 0 Jun 18 14:21 /proc/6180
```

 # ls -l /proc/6180
 # file /proc/6180/*

-> gedit 툴을 종료

 # ls -ld /proc/6180

```
ls: /proc/6180: No such file or directory
```

프로세스 관리

- 프로세스 실행

```
fg ) # sleep 300
bg ) # sleep 300 &
```

- 프로세스 확인

```
# ps -ef | grep xinetd
# ps aux | grep xinetd
```

- 프로세스 종료

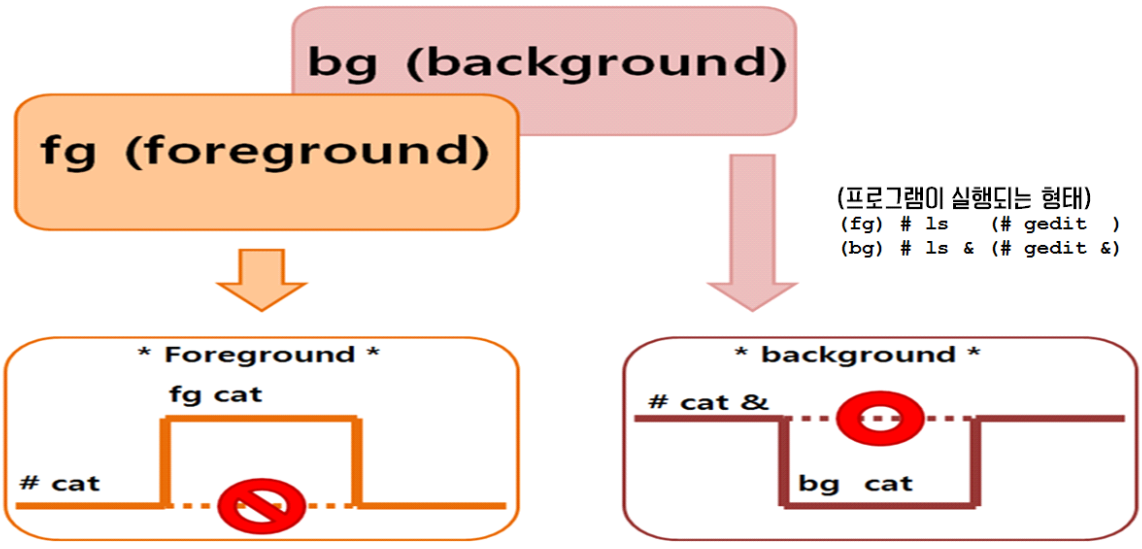
```
# kill -[1|2|9|15] PID
# killall httpd
```

3 프로세스 관리 (백그라운드와 포그라운드 관리 - &, bg, fg, jobs)

쉘 프롬프트에 명령어를 입력해서 포그라운드 상태($ cat > testfile)로 프로그램을 실행시킴. 그 프로세스가 실행되는 동안 셸 프롬프트를 볼 수 없다. <Ctrl + D>로 프로그램이 종료 되어야 쉘프롬프트 상태로 돌아오게 된다. 백그라운드는 명령행 끝에 &(앰퍼샌드)를 붙여서 사용하면 쉘 프롬프트가 떨어지고 대신 프로세스는 사용자가 보지 않는 상태로 실행된다.

백그라운드 프로세스는 $ bg 명령어로 확인가능 하고 $ fg 명령어로 포그라운드 전환이 가능하다. 반대로 하는 방법은 포그라운드에서 돌아가는 중에 <Ctrl+Z> 으로 멈추고 $ bg 명령을 내리면 된다.

백그라운드와 포그라운드 관리



(프로그램이 실행되는 형태)
(fg) # ls (# gedit)
(bg) # ls & (# gedit &)

(fg 형태로 프로그램이 실행되면)
ps

```
  PID TTY          TIME CMD
 8711 pts/1    00:00:00 bash
```

gedit
-> 명령어 창으로 돌아와서 "# ls" 명령어를 수행해도 명령어가 출력결과가 보이지 않는다.
-> 부모 프로세스(bash)가 종료되면 자식 프로세스가 종료된다.

(bg 형태로 프로그램이 실행되면)
ps

```
  PID TTY          TIME CMD
 8711 pts/1    00:00:00 bash
```

gedit &
-> 명령어 창으로 돌아와서 "# ls" 명령어를 수행하면 정상적으로 수행이 된다.
-> 부모 프로세스(bash)가 종료되었다고 해서 자식 프로세스가 종료되지 않는다.(# exit 명령어 사용)

[EX1] 프로그램 실행 실습

cat & <ENTER> /* 백그라운드 상태로 프로세스를 실행 */

```
[1] 7827
```

[1] : 잡 아이디(Job ID)
7827 : 프로세스 아이디(PID)

ps -ef | grep cat

```
root      7827  7601  0 02:46 pts/3    00:00:00 cat
root     27281     1  0 Jan26 ?        00:00:00 /usr/libexec/notification-area-applet --oaf-activate-ii
d=OAFIID:GNOME_NotificationAreaApplet_Factory --oaf-ior-fd=21

[1]+  Stopped                 cat
```

ls -ld /proc/7827

```
dr-xr-xr-x 5 root root 0  1월 27 02:46 /proc/7827/
```

fg (# fg 1)

```
cat
<CTRL + C>  /* 포그라운드 상태의 프로세스를 종료시킴 */
```

ls -ld /proc/7827 /* 지워진 것을 확인 할 수 있다. */

```
ls: /proc/7827/: 그런 파일이나 디렉토리가 없음
```

[EX2] 포그라운드 & 백그라이운드 실행 실습

```
# sleep 500                    /* 포그라운드 상태로 프로세스를 실행 */
[1]+  Stopped                 sleep 500
<CTRL + Z>

# sleep 600
[2]+  Stopped                 sleep 600
<CTRL + Z>

# sleep 700
[3]+  Stopped                 sleep 700
<CTRL + Z>

# jobs
[1]   Stopped                 sleep 500
[2]-  Stopped                 sleep 600
[3]+  Stopped                 sleep 700

# bg 1                        /* 일시중지된 포그라운드 작업을 백그라운드에서 실행 */
[1] sleep 500 &

# bg 2
[2]- sleep 600 &

# bg 3
[3]+ sleep 700 &

# jobs
[1]   Running                 sleep 500 &
[2]-  Running                 sleep 600 &
[3]+  Running                 sleep 700 &

# fg 1
sleep 500
<CTRL + C>

# fg 2
sleep 600
<CTRL + C>

# fg 3
sleep 700
<CTRL + C>

# jobs
#
```

(2) 프로세스 정보 확인

```
[명령어 형식]
# ps                        /* 현재 터미널에서 실행된 프로세스의 간략한 정보 확인 */
# ps -l                     /* 현재 터미널에서 실행된 프로세스의 자세한 정보 확인 */
# ps -a                     /* 사용자 프로세스에 대한 간략한 정보 확인 */
# ps -U <사용자명>          (# ps -U user01, # ps -U user01,user02,user03)
                            /* 특정한 사용자가 실행시킨 프로세스의 간략한 정보 확인 */
# ps -t pts/5               /* 특정한 터미널에서 실행시킨 프로세스의 간략한 정보 확인 */

# ps aux | grep xinetd    (# ps -ef | grep xinetd)
        # ps -aux | grep xinetd
```

[명령어 옵션]

옵션	내용
-e	모든 프로세스 리스트를 출력한다.
-f	프로세스 시작시간, 프로세스의 부모 ID, 그 프로세스에 관련된 사용자 ID, 명령 이름과 가능한 매개변수등 모든 정보를 출력한다. (full format)
-l	자세한 정보 보기
-p (pid)	지정된 프로세스에 대한 정보를 출력한다.
-t (tty)	지정된 터미널에 연관된 프로세스에 대한 정보를 출력한다.
-u (uid)	지정된 사용자에 연관된 프로세스에 대한 정보를 출력한다.
a	다른 사용자의 프로세스 상태도 표시된다.
x	화면에 보이지 않는 프로세스까지 모두 표시

[EX] ps 명령어 실습

```
# ps                        /* 자신을 포함하여 자신이 실행한 프로세스를 확인 */
  PID TTY          TIME CMD
 6165 pts/1    00:00:00 bash
```

```
# ps -U root                /* root 사용자가 사용하는 프로세스 정보를 보여줌 */
  PID TTY          TIME CMD
    1 ?        00:00:00 init
    2 ?        00:00:00 migration/0
    3 ?        00:00:00 ksoftirqd/0
    4 ?        00:00:00 watchdog/0
    5 ?        00:00:00 events/0
..... (중략) .....
```

```
# ps                        /* 현재 터미널에서 실행된 프로세스의 간략한 정보 확인 */
  PID TTY          TIME CMD
 7601 pts/3    00:00:00 bash
 7714 pts/3    00:00:00 ps
```

```
# ps -t pts/3               /* 터미널창 번호를 확인 후에 해당 터미널의 프로세스 정보를 확인 */
  PID TTY          TIME CMD
 7601 pts/3    00:00:00 bash
 7717 pts/3    00:00:00 ps
```

```
# ps a                      /* 다른 사용자의 프로세스 상태도 표시 */
  PID TTY      STAT   TIME COMMAND
 2450 tty1     Ss+    0:00 /sbin/mingetty tty1
 2453 tty2     Ss+    0:00 /sbin/mingetty tty2
 2454 tty3     Ss+    0:00 /sbin/mingetty tty3
 2455 tty4     Ss+    0:00 /sbin/mingetty tty4
 2456 tty5     Ss+    0:00 /sbin/mingetty tty5
 2457 tty6     Ss+    0:00 /sbin/mingetty tty6
 2562 tty7     Ss+    0:11 /usr/bin/Xorg :0 -br -audit 0 -auth /var/gdm/:0.Xauth
 7563 pts/1    Ss+    0:00 -bash
 7601 pts/3    Ss     0:00 -bash
 7731 pts/3    R+     0:00 ps a
27308 pts/2    Ss+    0:00 bash
```

```
# ps -ef | less                    /* 모든 프로세스 리스트를 출력하면서 모든 정보를 함께 출력 함 */
```

```
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 Jan26 ?        00:00:00 init [5]
root         2     1  0 Jan26 ?        00:00:00 [migration/0]
root         3     1  0 Jan26 ?        00:00:00 [ksoftirqd/0]
root         4     1  0 Jan26 ?        00:00:00 [watchdog/0]
root         5     1  0 Jan26 ?        00:00:00 [events/0]
```

```
# ps aux | less                    /* a : 다른 사용자의 프로세스 상태도 표시
                                      x : 화면에 보이지 않는 프로세스까지 모두 표시
                                      u : 프로세스를 사용한 사용자와 실행 시간까지 표시 */
```

```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1   2068   576 ?        Ss   Jan26   0:00 init [5]
root         2  0.0  0.0      0     0 ?        S<   Jan26   0:00 [migration/0]
root         3  0.0  0.0      0     0 ?        SN   Jan26   0:00 [ksoftirqd/0]
root         4  0.0  0.0      0     0 ?        S<   Jan26   0:00 [watchdog/0]
root         5  0.0  0.0      0     0 ?        S<   Jan26   0:00 [events/0]
root         6  0.0  0.0      0     0 ?        S<   Jan26   0:00 [khelper]
root         7  0.0  0.0      0     0 ?        S<   Jan26   0:00 [kthread]
root        10  0.0  0.0      0     0 ?        S<   Jan26   0:01 [kblockd/0]
```

[참고] ps aux를 통해 명령어 출력시 나오는 헤더부분 설명

```
user는 프로세스 소유자의 이름, ( u 옵션 )
PID는 프로세스 식별자 번호
%CPU : 마지막 1분 동안 프로세스가 사용한 CPU 점유율 ( u 옵션 )
%MEN : 마지막 1분 동안 프로세스가 사용한 메모리의 점유율 ( u,v 옵션 )
     -u 옵션으로 사용자의 ID나 이름을 지정하여 지정한 사용자의 프로세스만을 살펴 볼수 있다.
VSZ : 가상 메모리에 적재된 프로세스의 kb단위 크기
START는 프로세스가 시작된 시간
```

STAT 필드의 상태

```
-D : 디스크 입출력 대기 상태로 interrupts를 걸 수 없는 상태
-R : 실행중
-S : 짧은 sleep 상태
-T : 정지 상태
-Z : 좀비 상태
-W : 상주한 페이지가 없는 프로세스
-< : 높은 우선권 프로세스
-N : 낮은 우선권 프로세스
-L : 페이지가 락이 걸린 상태
```

4 프로세스를 죽이는 명령 - kill

실행 중인 프로세스를 강제로 종료(인터럽트)시킬 수 있지만 백그라운드에서 실행되는 프로세스는 종료시키지 못하게 된다. 이때 kill 명령을 사용해서 프로세스를 종료시키다. kill명령은 사용자가 프로세스에 일정한 신호를 보내서 프로세스를 종료시킨다, kill -l 명령을 사용하면 신호의 종류를 확인 할 수 있다.

```
(명령어 형식)
# kill PID            (# kill -15 PID, # kill -TERM PID)
# kill PID PID PID
# kill -9 PID         (# kill -KILL PID)
# kill -2 PID         (# kill -INT PID)
```

■ 시그널(Signal)? 프로세스가 다른 프로세스에게 보내는 비동기적 알림 메세지

[참고] 일반적으로 많이 쓰이는 시그널(Signal)

시그널 번호	시그널 이름	설 명
1	SIGHUP	프로세스 재시작(HangUp) (EX: **# kill -1 450**)
2	SIGINT	인터럽트(Interrupt, <Ctrl + C>) (EX: **# kill -2 450**)
9	SIGKILL	강제 종료(force exit signal) (EX: **# kill -9 450**)
15	SIGTERM	정상 종료(exit), 기본 시그널 (EX: **# kill -15 450**)

kill 명령에 아무 신호도 입력하지 않으면 15번 신호인 SIGTERM을 보낸다. (기본값)

 # man 7 signal

```
..... (중략) .....
       Signal     Value     Action   Comment
       ──────────────────────────────────────────────────────────────────────
       SIGHUP       1       Term    Hangup detected on controlling terminal
                                    or death of controlling process
       SIGINT       2       Term    Interrupt from keyboard
       SIGQUIT      3       Core    Quit from keyboard
       SIGILL       4       Core    Illegal Instruction
       SIGABRT      6       Core    Abort signal from abort(3)
       SIGFPE       8       Core    Floating point exception
       SIGKILL      9       Term    Kill signal
       SIGSEGV     11       Core    Invalid memory reference
       SIGPIPE     13       Term    Broken pipe: write to pipe with no readers
       SIGALRM     14       Term    Timer signal from alarm(2)
       SIGTERM     15       Term    Termination signal
       SIGUSR1   30,10,16   Term    User-defined signal 1
       SIGUSR2   31,12,17   Term    User-defined signal 2
       SIGCHLD   20,17,18   Ign     Child stopped or terminated
       SIGCONT   19,18,25   Cont    Continue if stopped
       SIGSTOP   17,19,23   Stop    Stop process
       SIGTSTP   18,20,24   Stop    Stop typed at tty
       SIGTTIN   21,21,26   Stop    tty input for background process
..... (중략) .....
```

 # kill -l

```
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL
 5) SIGTRAP      6) SIGABRT      7) SIGBUS       8) SIGFPE
 9) SIGKILL     10) SIGUSR1     11) SIGSEGV     12) SIGUSR2
13) SIGPIPE     14) SIGALRM     15) SIGTERM     16) SIGSTKFLT
17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU
25) SIGXFSZ     26) SIGVTALRM   27) SIGPROF     28) SIGWINCH
29) SIGIO       30) SIGPWR      31) SIGSYS      34) SIGRTMIN
35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3  38) SIGRTMIN+4
39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12
47) SIGRTMIN+13 48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14
51) SIGRTMAX-13 52) SIGRTMAX-12 53) SIGRTMAX-11 54) SIGRTMAX-10
55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7  58) SIGRTMAX-6
59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```

[EX] kill 명령어 실습

```
# ps -ef | grep sendmail
root     25311     1  0 Jan26 ?        00:00:00 sendmail: accepting connections
smmsp    25319     1  0 Jan26 ?        00:00:00 sendmail: Queue runner@01:00:00 for
                                                 /var/spool/clientmqueue
```

```
# kill 25311      (# kill -15 25311, # kill -TERM 25311)
# ps -ef | grep sendmail
smmsp    25319     1  0 Jan26 ?        00:00:00 sendmail: Queue runner@01:00:00 for
                                                 /var/spool/clientmqueue
```

```
# service sendmail restart
Shutting down sm-client:                                   [FAILED]
Shutting down sendmail:                                    [  OK  ]
Starting sendmail:                                         [  OK  ]
Starting sm-client:                                        [  OK  ]
```

```
# ps -ef | grep sendmail
root      7908     1  0 03:05 ?        00:00:00 sendmail: accepting connections
smmsp    25319     1  0 Jan26 ?        00:00:00 sendmail: Queue runner@01:00:00 for
                                                 /var/spool/clientmq ueue
```

```
# kill -9 7908    (# kill -KILL 7908)
#
```

```
# ps -ef | grep sendmail
smmsp    25319     1  0 Jan26 ?        00:00:00 sendmail: Queue runner@01:00:00 for
                                                  /var/spool/clientmq ueue
```

```
# service sendmail restart
Shutting down sm-client:                                   [FAILED]
Shutting down sendmail:                                    [  OK  ]
Starting sendmail:                                         [  OK  ]
Starting sm-client:                                        [  OK  ]
```

```
# ps -ef | grep sendmail
root      7934     1  0 03:06 ?        00:00:00 sendmail: accepting connections
smmsp    25319     1  0 Jan26 ?        00:00:00 sendmail: Queue runner@01:00:00 for
                                                  /var/spool/clientmq ueue
```

```
# kill -1 7934    (# kill -HUP 7934)
# ps -ef | grep sendmail
root      7939     1  0 03:06 ?        00:00:00 sendmail: accepting connections
smmsp    25319     1  0 Jan26 ?        00:00:00 sendmail: Queue runner@01:00:00 for
                                                 /var/spool/clientmq ueue
```

[참고] kill & killall 명령어

```
      # kill PID
      # killall httpd
```

프로세스 모니터링

- top
  ```
  # top
  # top -u oracle
  # top -p [PID]
  ```

- gnome-system-monitor
  ```
  # gnome-system-monitor
  # gnome-system-log
  ```

- lsof
  ```
  # lsof
  # lsof -I
  # lsof -p [PID]
  # lsof -c sendmail
  ```

5 프로세스 모니터링

(1) top 명령 (ps와 같은 명령어 - 차이점? top은 실시간 모니터링)

[참고] 서버에 접속하여 시스템 모니터링
[TERM1] (TUI) # top (GUI) # gnome-system-monitor
[TERM2] (TUI) # tail -f /var/log/messages (GUI) # gnome-system-log

- 실시간으로 프로세스의 상태나 CPU, 메모리, 접속한 사용자의 수 등을 알 수 있게 된다. 시스템의 성능을 확인하려 할 때 많이 사용하게 된다.

기본설정
- 명령어 수행 시간 간격(time interval)은 3 ~ 4초
- CPU 사용량별로 정열(Sorting)

(명령어 형식)
```
# top
# top -u oracle                 /* 해당 사용자의 프로세스 정보를 실시간 모니터링 */
# top -p PID1,PID2,PID3         /* 해당 번호의 프로세스 정보를 실시간 모니터링 */
# top -d [TIME]                 /* 새로고침의 시간을 변경한다. */
```

[EX1] top 명령어 기본 사용방법 실습

 # top -h (**# top --help**)

```
        top: procps version 3.2.7
usage:  top -hv | -bcisSHM -d delay -n iterations [-u user | -U user] -p pid [,pid ...]
```

 # top

```
top - 17:46:37 up  2:25,  4 users,  load average: 0.03, 0.03, 0.00
Tasks: 136 total,   1 running, 134 sleeping,   0 stopped,   1 zombie
Cpu(s):  2.0%us,  1.7%sy,  0.0%ni, 96.3%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   1035096k total,   718752k used,   316344k free,    68376k buffers
Swap:   522104k total,        0k used,   522104k free,   492140k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 4203 root      16   0 84828  16m  11m S  2.7  1.6   0:09.65 gnome-terminal
 3624 root      15   0 35972  10m 6360 S  0.7  1.1   0:16.60 Xorg
 3813 root      15   0 42812  10m 8520 S  0.2  1.1   0:01.10 vmware-user
    1 root      18   0  2072  616  532 S  0.0  0.1   0:01.26 init
    2 root      RT  -5     0    0    0 S  0.0  0.0   0:00.35 migration/0
    3 root      34  19     0    0    0 S  0.0  0.0   0:00.00 ksoftirqd/0
    4 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 watchdog/0
    5 root      RT  -5     0    0    0 S  0.0  0.0   0:00.15 migration/1
    6 root      36  19     0    0    0 S  0.0  0.0   0:00.00 ksoftirqd/1
    7 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 watchdog/1
    8 root      10  -5     0    0    0 S  0.0  0.0   0:00.03 events/0
    9 root      10  -5     0    0    0 S  0.0  0.0   0:00.00 events/1
   10 root      10  -5     0    0    0 S  0.0  0.0   0:00.01 khelper
   11 root      10  -5     0    0    0 S  0.0  0.0   0:00.00 kthread
   15 root      18  -5     0    0    0 S  0.0  0.0   0:00.03 kblockd/0
   16 root      10  -5     0    0    0 S  0.0  0.0   0:00.02 kblockd/1
   17 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 kacpid
  183 root      11  -5     0    0    0 S  0.0  0.0   0:00.00 cqueue/0
  184 root      11  -5     0    0    0 S  0.0  0.0   0:00.00 cqueue/1
  187 root      10  -5     0    0    0 S  0.0  0.0   0:00.00 khubd
  189 root      10  -5     0    0    0 S  0.0  0.0   0:00.00 kseriod
  256 root      16   0     0    0    0 S  0.0  0.0   0:00.00 pdflush
  257 root      15   0     0    0    0 S  0.0  0.0   0:00.15 pdflush
..... (중략) .....
```

h <----- 'h' 입력

```
Help for Interactive Commands - procps version 3.2.7
Window 1:Def: Cumulative mode Off.  System: Delay 3.0 secs; Secure mode Off.

  Z,B       Global: 'Z' change color mappings; 'B' disable/enable bold
  l,t,m     Toggle Summaries: 'l' load avg; 't' task/cpu stats; 'm' mem info
  1,I       Toggle SMP view: '1' single/separate states; 'I' Irix/Solaris mode

  f,o     . Fields/Columns: 'f' add or remove; 'o' change display order
  F or O  . Select sort field
  <,>     . Move sort field: '<' next col left; '>' next col right
  R,H     . Toggle: 'R' normal/reverse sort; 'H' show threads
  c,i,S   . Toggle: 'c' cmd name/line; 'i' idle tasks; 'S' cumulative time
  x,y     . Toggle highlights: 'x' sort field; 'y' running tasks
  z,b     . Toggle: 'z' color/mono; 'b' bold/reverse (only if 'x' or 'y')
  u       . Show specific user only
  n or #  . Set maximum tasks displayed

  k,r       Manipulate tasks: 'k' kill; 'r' renice
  d or s    Set update interval
  W         Write configuration file
  q         Quit
          ( commands shown with '.' require a visible task display window )
Press 'h' or '?' for help with Windows,
any other key to continue
```

PID to kill: <firedox의 PID번호 입력>
Kill PID 23846 with signal [15]: <9>

정렬 기준 변경 : F입력

```
Current Sort Field:  K  for window 1:Def
Select sort field via field letter, type any other key to return

  a: PID        = Process Id              s: DATA       = Data+Stack size (kb)   not
those in column display.  Thus,       t: SHR        = Shared Mem size (kb)   the TTY &
WCHAN fields will violate       u: nFLT       = Page Fault count       strict ASCII
collating sequence.          v: nDRT       = Dirty Pages count      (shame on you if
WCHAN is chosen)
  f: GROUP      = Group Name              x: COMMAND    = Command name/line
  g: TTY        = Controlling Tty         y: WCHAN      = Sleeping in Function
```

```
  h: PR        = Priority               z: Flags      = Task Flags <sched.h>
  i: NI        = Nice value
  j: P         = Last used cpu (SMP)   Note1:
* K: %CPU      = CPU usage               If a selected sort field can't be
  l: TIME      = CPU Time                shown due to screen width or your
  m: TIME+     = CPU Time, hundredths    field order, the '<' and '>' keys
  n: %MEM      = Memory usage (RES)      will be unavailable until a field
  o: VIRT      = Virtual Image (kb)      within viewable range is chosen.
  p: SWAP      = Swapped size (kb)
  q: RES       = Resident size (kb)    Note2:
  r: CODE      = Code size (kb)          Field sorting uses internal values,
```

- (첫 번째 줄) 시스템의 현재 시간과 시스템이 부팅된 후 작동한 시간, 현재 사용자 수를 보여주며, load average는 CPU 로드의 평균 값을 보여준다.
- (두 번째 줄) 시스템에 동작중인 프로세스의 상태를 보여준다.
- (세 번째 줄) CPU의 상태를 보여준다
- (네 번째 / 다섯 번째 줄) 총 사용 가능 메모리, 사용된 메모리, 사용할 수 있는 메모리, 공유 메모리, 버퍼로 사용된 메모리, 스왑 메모리등 메모리에 관한 정보를 보여준다. 각각의 시스템 자원에 따른 프로세스의 점유율을 확인 할 수 있다.

(top 명령어 필드 해석)

---

a: **PID** -- Process Id
The task's unique process ID, which periodically wraps, though never restarting at zero.

b: **PPID** -- Parent Process Pid
The process ID of a task's parent.

c: RUSER -- Real User Name
The real user name of the task's owner.

d: UID -- User Id
The effective user ID of the task's owner.

e: **USER** -- User Name
The effective user name of the task's owner.

f: GROUP -- Group Name
The effective group name of the task's owner.

g: TTY -- Controlling Tty
The name of the controlling terminal. This is usually the device (serial port, pty, etc.) from which the process was started, and which it uses for input or output. However, a task need not be associated with a terminal, in which case you'll see '?' displayed.

h: **PR** -- Priority
The priority of the task.

i: **NI** -- Nice value
The nice value of the task. A negative nice value means higher priority, whereas a positive nice value means lower priority. Zero in this field simply means priority will not be adjusted in determining a task's dispatchability.

j: P -- Last used CPU (SMP)
A number representing the last used processor. In a true SMP environment this will likely change frequently since the kernel intentionally uses weak affinity. Also, the very act of running top may break this weak affinity and cause more processes to change CPUs more often (because of the extra demand for cpu time).

k: **%CPU** -- CPU usage
The task's share of the elapsed CPU time since the last screen update, expressed as a percentage of total CPU time. In a true SMP environment, if 'Irix mode' is Off, top will operate in 'Solaris mode' where a task's cpu usage will be divided by the total number of CPUs. You toggle 'Irix/Solaris' modes with the 'I' interactive command.

l: TIME -- CPU Time
Total CPU time the task has used since it started. When 'Cumulative mode' is On, each process is listed with the cpu time that it and its dead children has used. You toggle 'Cumulative mode' with 'S', which is a command-line option and an interactive command. See the 'S' interactive command for additional information regarding this mode.

m: **TIME+** -- CPU Time, hundredths
The same as 'TIME', but reflecting more granularity through hundredths

of a second.

n: **%MEM** -- Memory usage (RES)
A task's currently used share of available physical memory.

o: **VIRT** -- Virtual Image (kb)
The total amount of virtual memory used by the task. It includes all code, data and shared libraries plus pages that have been swapped out. (Note: you can define the STATSIZE=1 environment variable and the VIRT will be calculated from the /proc/#/state VmSize field.)

VIRT = SWAP + RES.

p: SWAP -- Swapped size (kb)
The swapped out portion of a task's total virtual memory image.

q: **RES** -- Resident size (kb)
The non-swapped physical memory a task has used.

RES = CODE + DATA.

r: CODE -- Code size (kb)
The amount of physical memory devoted to executable code, also known as the 'text resident set' size or TRS.

s: DATA -- Data+Stack size (kb)
The amount of physical memory devoted to other than executable code, also known as the 'data resident set' size or DRS.

t: **SHR** -- Shared Mem size (kb)
The amount of shared memory used by a task. It simply reflects memory that could be potentially shared with other processes.

u: nFLT -- Page Fault count
The number of major page faults that have occurred for a task. A page fault occurs when a process attempts to read from or write to a virtual page that is not currently present in its address space. A major page fault is when disk access is involved in making that page available.

v: nDRT -- Dirty Pages count
The number of pages that have been modified since they were last written to disk. Dirty pages must be written to disk before the corresponding physical memory location can be used for some other virtual page.

w: **S** -- Process Status
The status of the task which can be one of:
'D' = uninterruptible sleep
'R' = running
'S' = sleeping
'T' = traced or stopped
'Z' = zombie

Tasks shown as running should be more properly thought of as 'ready to run' -- their task_struct is simply represented on the Linux run-queue. Even without a true SMP machine, you may see numerous tasks in this state depending on top's delay interval and nice value.

x: **Command** -- Command line or Program name
Display the command line used to start a task or the name of the associated program. You toggle between command line and name with 'c', which is both a command-line option and an interactive command.

When you've chosen to display command lines, processes without a command line (like kernel threads) will be shown with only the program name in parentheses, as in this example:
(mdrecoveryd)

Either form of display is subject to potential truncation if it's too long to fit in this field's current width. That width depends upon other fields selected, their order and the current screen width.

Note: The 'Command' field/column is unique, in that it is not fixed-width. When displayed, this column will be allocated all remaining screen width (up to the maximum 512 characters) to provide for the potential growth of program names into command lines.

y: WCHAN -- Sleeping in Function
Depending on the availability of the kernel link map ('System.map'),

```
          this field will show the name or the address of the kernel function in
          which the task is currently sleeping.  Running tasks  will  display  a
          dash ('-') in this column.

          Note:  By  displaying  this  field,  top's  own  working  set  will be
          increased by over 700Kb.  Your only means of  reducing  that  overhead
          will be to stop and restart top.

       z: Flags  --  Task Flags
          This  column  represents the task's current scheduling flags which are
          expressed in hexadecimal notation and with  zeros  suppressed.   These
          flags are officially documented in <linux/sched.h>.  Less formal docu-
          mentation can also be found on the 'Fields select' and 'Order  fields'
          screens.
```

[EX2] 사용자 별로 모니터링

(명령어 형식)
 # top -u root
 # top -u oracle
 # top -u wasuser

[TERM1] root 사용자 윈도우
 # top -u fedora

[TERM2] fedora 사용자 윈도우
 # telnet localhost
fedora 사용자로 로그인

$ id
$ bash -----> 관리자 윈도우 확인
$ ksh -----> 관리자 윈도우 확인
$ vi /etc/passwd
$ exit
$ exit
$ exit

[EX3] 몇개의 프로세스만 모니터링

(명령어 형식)
 # top -p 1
 # top -p 1,3624

 # ps (# pstree 2371)

```
  PID TTY          TIME CMD
 2371 pts/3    00:00:00 bash
 6554 pts/3    00:00:00 ps
```

 # top -p 1,2371

```
top - 17:51:27 up 1 day, 8 min,  4 users,  load average: 0.12, 0.16, 0.10
Tasks:   2 total,   0 running,   2 sleeping,   0 stopped,   0 zombie
Cpu(s):  7.4%us,  1.5%sy,  0.0%ni, 90.9%id,  0.0%wa,  0.0%hi,  0.3%si,  0.0%st
Mem:    515340k total,   500336k used,    15004k free,    63108k buffers
Swap:  1052248k total,        0k used,  1052248k free,   255872k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
    1 root      18   0  2068  620  532 S  0.0  0.1   0:00.49 init
 2371 root      15   0  5872 1496 1188 S  0.0  0.3   0:00.62 bash
```

-> 프로세스 PID 번호 선택은 출력 화면 참조

[EX4] 모니터링 툴 종류

```
        # top                                   - Display Linux tasks
        # free                                  - Display amount of free and used memory in the system
        # /usr/bin/baobab                       - A graphical tool to analyse disk usage
        # /usr/bin/xload                        - System load average display for X
        # /usr/bin/gnome-system-monitor         - System monitor
        # /usr/bin/gnome-system-log             - /var/logs/messages log viewer
```

```
# gnome-[TAB][TAB]
gnome-about                             gnome-pilot-make-password
gnome-about-me                          gnome-power-inhibit-test
gnome-accessibility-keyboard-properties gnome-power-manager
gnome-at-properties                     gnome-power-preferences
gnome-audio-profiles-properties         gnome-screensaver
gnome-background-properties             gnome-screensaver-command
gnome-calculator                        gnome-screensaver-preferences
gnome-cd                                gnome-screenshot
gnome-character-map                     gnome-search-tool
gnome-control-center                    gnome-session
gnome-default-applications-properties   gnome-session-properties
gnome-default-printer                   gnome-session-remove
gnome-desktop-item-edit                 gnome-session-save
gnome-dictionary                        gnome-sound-properties
gnome-display-properties                gnome-system-log
gnome-doc-prepare                       gnome-system-monitor
gnome-eject                             gnome-terminal
gnome-file-share-properties             gnome-text-editor
gnome-font-properties                   gnome-theme-manager
gnome-font-viewer                       gnome-theme-thumbnailer
gnome-help                              gnome-thumbnail-font
gnome-keybinding-properties             gnome-typing-monitor
gnome-keyboard-properties               gnome-ui-properties
gnome-keyring-daemon                    gnome-umount
gnome-mount                             gnome-volume-control
gnome-mouse-properties                  gnome-volume-manager
gnome-network-preferences               gnome-volume-properties
```

```
# gnome-system-[TAB][TAB]
gnome-system-log      gnome-system-monitor
```

[EX5] 시스템 모니터링(CPU/MEM)
(실습 준비)

```
# mkdir -p /root/bin
# cd /root/bin
# scp 172.16.9.252:/root/shell/* /root/bin
# ls
cpu.sh*  cpu2.sh*  cpu3.sh*  mem.c  disk_exhaust.c
```

```
        [TERM2] # top
        [TERM3] # gnome-system-monitor &
```

```
# chmod 755 *.sh
#
```

■ CPU 부하량 테스트

```
# cd /root/bin
# cat cpu.sh
#!/bin/bash

echo "+-----------------------------------------+"
echo "| Control-C will terminate cpu.sh process. |"
echo "+-----------------------------------------+"

START=0
END=1000000000000000000
while [ $START -le $END ]
do
        START=`expr $START + 1`
done
```

```
# ./cpu.sh
```

cat cpu2.sh

```
#!/bin/bash

echo "+-----------------------------------------+"
echo "| Control-C will terminate cpu2.sh process.|"
echo "+-----------------------------------------+"

while true
do
        a=1
done
```

./cpu2.sh

cat cpu3.sh

```
#!/bin/bash

echo "+-----------------------------------------+"
echo "| Control-C will terminate cpu3.sh process.|"
echo "+-----------------------------------------+"

trap 'killall cpu.sh ; exit 1' 2 3

./cpu.sh &
sleep 10
./cpu.sh &
sleep 10
./cpu.sh &
sleep 10
./cpu.sh &
sleep 3600

trap 2 3
```

./cpu3.sh

■ 메모리 부하량 테스트

cat mem.c

```
#include<stdio.h>
#include<stdlib.h>

main()
{
        char *m;

        printf("+-----------------------------------------+₩n");
        printf("| Control-C will terminate mem process.    |₩n");
        printf("+-----------------------------------------+₩n");

        while (1)
                m=malloc(1);
}
```

gcc -o mem mem.c (# yum -y install gcc)
./mem

■ 디스크 부하량 테스트

```
[TERM2] # top
[TERM3] # gnome-system-monitor
[TERM4] # while true
        > do
        > echo "----------------`date`-------------"
        > df -h /
        > sleep 2
        > done
```

cat disk_exhaust.c

```
#include<fcntl.h>
#include<sys/types.h>
#include<sys/stat.h>

main()
{
    int fd;
    char buf[10000];
    fd=open("tempfile", O_WRONLY | O_CREAT, 0777);
    unlink("./tempfile");
    while(1)
        write(fd, buf, sizeof(buf));
}
```

gcc -o disk_exhaust disk_exhaust.c
./disk_exhaust

(2) lsof(List Open File) 명령어

프로세스에 의해 열려진 파일들에 대한 정보를 볼 수 있다. 특정 파일을 접근하고 있는 프로세스인지 특정 포트로 접속한 사용자를 확인 할 때 유용한 프로세스이다.

[명령어 형식]
lsof
lsof -p <PID번호>
lsof -c <데몬명>

[EX] lsof 명령어 실습
 # ps -ef | grep sendmail

```
root     7939     1  0 03:06 ?      00:00:00 sendmail: accepting connections
root     7960  7601  0 03:13 pts/3  00:00:00 grep sendmail
smmsp   25319     1  0 Jan26 ?      00:00:00 sendmail: Queue runner@01:00:00 for /var/spool/clientmqueue
```

 # lsof -p 7939

```
COMMAND    PID USER   FD   TYPE     DEVICE    SIZE     NODE NAME
sendmail 7939 root  cwd    DIR        8,1    4096 3095078 /var/spool/mqueue
sendmail 7939 root  rtd    DIR        8,1    4096       2 /
sendmail 7939 root  txt    REG        8,1  806460  928207 /usr/sbin/sendmail.sendmail
sendmail 7939 root  mem    REG        8,1 1011184 1140250 /lib/libdb-4.3.so
..... (중략) .....
```

Column	설 명
Command	프로세스와 관련된 Unix command 이름
PID	Process IDentification number
PPID	Parent Process IDentification number (해당 프로세스의 부모 프로세스 ID)
PGRP	Process Group IDentification number (해당 프로세스와 관련된 프로세스 그룹 ID)
USER	해당 프로세스를 소유한 사용자 ID 또는 login name
FD	File Descriptor number (ex) cwd : current working directory r : read access / w : write access / u : read and write access
TYPE	해당 파일과 관련한 노드 타입 (ex) inet : Internet domain socket
DEVICE	device number
SIZE SIZE/OFF OFFSET	file 이나 file offset의 사이즈
INODE NODE-ID	local file 의 node number 또는 Internet protocol type 또는 서버 호스트의 NFS file의 inode number
NAME	해당 파일이 소속된 mount point나 파일 시스템의 이름

 # lsof -i /* 현재 사용하고 있는 모든 소켓 상태를 확인 가능 */

```
COMMAND     PID   USER   FD   TYPE DEVICE SIZE NODE NAME
portmap    1930    rpc    3u   IPv4   5376       UDP *:sunrpc
portmap    1930    rpc    4u   IPv4   5377       TCP *:sunrpc (LISTEN)
rpc.statd  1962   root    3u   IPv4   5552       UDP *:869
rpc.statd  1962   root    6u   IPv4   5529       UDP *:866
rpc.statd  1962   root    7u   IPv4   5559       TCP *:872 (LISTEN)
```

 # ssh root@172.16.9.252
root 사용자로 로그인

 # lsof -i@172.16.9.252 (# netstat -apn | grep EST)

```
COMMAND   PID  USER   FD   TYPE DEVICE SIZE NODE NAME
sshd     7561  root    3u  IPv6 152794       TCP linux200:ssh->192.168.0.2:50259 (ESTABLISHED)
sshd     7599  root    3u  IPv6 152898       TCP linux200:ssh->192.168.0.2:50261 (ESTABLISHED)
named   25482 named   21u  IPv4 125320       TCP linux200:domain (LISTEN)
named   25482 named  513u  IPv4 125319       UDP linux200:domain
```

 # exit
 # lsof -c sshd | more /* 특정 데몬이 사용하고 있는 모든 파일의 정보 출력 */

```
COMMAND  PID USER   FD   TYPE DEVICE    SIZE    NODE NAME
sshd    7476 root  cwd    DIR    8,1    4096       2 /
sshd    7476 root  rtd    DIR    8,1    4096       2 /
sshd    7476 root  txt    REG    8,1  408384  935283 /usr/sbin/sshd
sshd    7476 root  mem    REG    8,1   76400 2023272 /lib/libresolv-2.5.so
sshd    7476 root  mem    REG    8,1   45288 2023297 /lib/libcrypt-2.5.so
....
```

프로세스 모니터링

- pmap 명령어
  ```
  # pmap PID
  ```
- pstree 명령어
  ```
  # pstree
  # pstree PID
  ```
- nice / renice 명령어
  ```
  # nice -n (-20 ~ 18) CMD
  # renice -n (-20~20 PID
  ```

```
pCMD(ps, pmap, pstree, pwdx, ....)
# pmap PID
# pstree PID
# pwdx PID
```

(3) pmap 명령어

프로세스가 사용하고 있는 메모리의 주소를 확인 할 수 있다. 응용프로그램이 실행될 때 얼마나 많은 라이브러리가 로드되는가를 확인 할 수 있다.

[명령어 형식]
```
# pmap PID
```

[EX] pmap 명령어 사용법 실습
 # ps -ef | grep senmail (# pgrep -lf sendmail)

```
root      8024     1  0 03:28 ?        00:00:00 sendmail: accepting connections
smmsp     8032     1  0 03:28 ?        00:00:00 sendmail: Queue runner@01:00:00 for /var/spool/clientmq
                                                                                    ueue
root      8073  8040  0 03:28 pts/1    00:00:00 grep sendmail
```

 # pmap 8024 /* 프로세스가 사용하고 있는 메모리의 주소를 확인 */

```
8024:   sendmail: accepting connections
00110000   1192K r-x--  /lib/libcrypto.so.0.9.8e
0023a000     76K rwx--  /lib/libcrypto.so.0.9.8e
0024d000     16K rwx--    [ anon ]
00251000    980K r-x--  /lib/libdb-4.3.so
00346000     12K rwx--  /lib/libdb-4.3.so
00349000     36K r-x--  /lib/libcrypt-2.5.so

..... (중략) .....

00fcf000     68K r-x--  /lib/libresolv-2.5.so
00fe0000      4K r-x--  /lib/libresolv-2.5.so
00fe1000      4K rwx--  /lib/libresolv-2.5.so
00fe2000      8K rwx--    [ anon ]
09c2e000    404K rw---    [ anon ]
b7f9f000     28K rw---    [ anon ]
bf95a000    200K rw---    [ stack ]
 total     9432K
```

(4) pstree 프로세스 상관도

실행중인 프로세스 상태를 트리 구조로 보여주는 명령어. 프로세스의 부모 자식관계를 형태로 보여주는 명령어

[명령어 형식]

```
# pstree        (# pstree -l)
# pstree PID
# pstree user01
```

(선수작업) 필요하면 명령어 수행

```
# ps
# sleep 500 &
# sleep 600 &
# sleep 700 &
# ps
```

 # ps

```
  PID TTY          TIME CMD
 8040 pts/1    00:00:00 bash
 8077 pts/1    00:00:00 sleep
 8082 pts/1    00:00:00 sleep
 8083 pts/1    00:00:00 sleep
 8099 pts/1    00:00:00 ps
```

 # pstree -l /* 프로세스의 관계를 길게 출력 */

```
init─┬─acpid
     ├─atd
     ├─auditd─┬─audispd─────{audispd}
     │        └─{auditd}
     ├─automount─────4*[{automount}]
     ├─avahi-daemon─────avahi-daemon
     ├─bonobo-activati─────{bonobo-activati}
     ├─brcm_iscsiuio─────3*[{brcm_iscsiuio}]
     ├─bt-applet
     ├─clock-applet
     ├─crond
     ├─cupsd
     ├─2*[dbus-daemon]
     ├─dbus-launch
     ├─eggcups
     ├─escd─────{escd}
     ├─events/0
     ├─gam_server
     ├─gconfd-2
     ├─gdm-binary─────gdm-binary─┬─Xorg
     │                           └─gnome-session─────Xsession
     ├─gdm-rh-security─────{gdm-rh-security}
     ├─gnome-keyring-d
     ├─gnome-panel
     ├─gnome-power-man
     ├─gnome-screensav
     ├─gnome-settings─────{gnome-settings-}
     ├─gnome-terminal─┬─bash
     │                ├─gnome-pty-helpe
     │                └─{gnome-terminal}
     ├─gnome-vfs-daemo
     ├─gnome-volume-ma
     ├─gpm
     ├─hald─────hald-runner─┬─hald-addon-acpi
     │                      ├─hald-addon-keyb
     │                      └─hald-addon-stor
     ├─hcid
     ├─hidd
     ├─hpiod
     ├─httpd─┬─8*[httpd]
     │       └─{httpd}
     ├─2*[iscsid]
     ├─khelper
     ├─klogd
     ├─krfcommd
     ├─ksoftirqd/0
     ├─kthread─┬─aio/0
     │         ├─ata/0
     │         ├─ata_aux
     │         ├─cqueue/0
     │         ├─ib_addr
     │         ├─ib_cm/0
```

```
        │           ├─ib_inform
        │           ├─ib_mcast
        │           ├─iscsi_eh
        │           ├─iw_cm_wq
        │           ├─iw_cxgb3
        │           ├─kacpid
        │           ├─kauditd
        │           ├─kblockd/0
        │           ├─kgameportd
        │           ├─khubd
        │           ├─kjournald
        │           ├─kmpath_handlerd
        │           ├─kmpathd/0
        │           ├─kpsmoused
        │           ├─kseriod
        │           ├─kstriped
        │           ├─kswapd0
        │           ├─local_sa
        │           ├─mpt_poll_0
        │           ├─2*[pdflush]
        │           ├─rdma_cm
        │           ├─rpciod/0
        │           └─scsi_eh_0
        ├─mapping-daemon
        ├─metacity
        ├─migration/0
        ├─6*[mingetty]
        ├─mixer_applet2─────{mixer_applet2}
        ├─named─────3*[{named}]
        ├─nautilus
        ├─nm-applet
        ├─nm-system-setti
        ├─notification-ar
        ├─nss_pcache
        ├─pam-panel-icon─────pam_timestamp_c
        ├─pcscd─────3*[{pcscd}]
        ├─portmap
        ├─puplet
        ├─python
        ├─rpc.idmapd
        ├─rpc.statd
        ├─scim-bridge
        ├─scim-helper-man
        ├─2*[scim-launcher]
        ├─scim-panel-gtk─────{scim-panel-gtk}
        ├─sdpd
        ├─2*[sendmail]
        ├─smartd
        ├─sshd─────sshd─────bash─┬─pstree
        │                        └─2*[sleep]
        ├─syslogd
        ├─trashapplet
        ├─udevd
        ├─uuidd
        ├─watchdog/0
        ├─wnck-applet
        ├─xfs
        ├─xinetd
        └─yum-updatesd
```

```
# pstree 8040              /* 프로세스간의 부모 자식관계를 트리 형태로 보여 줌 */
bash─┬─pstree
     └─3*[sleep]
```

[참고] 실습을 위한 준비

```
# bash
# bash
# ps
# pstree 8040
bash─┬─bash─────bash─────pstree
     └─3*[sleep]
# exit
# exit
# jobs
# kill %1
# kill %2
# kill %3
```

(5) pwdx CMD

대상 프로세스의 현재 작업 디렉토리를 얻을 수 있다.

ps

```
  PID TTY          TIME CMD
21716 pts/2    00:00:00 bash
```

pwdx 21716

```
21716: /root
```

cd /etc/sysconfig
pwdx 21716

```
21716: /etc/sysconfig
```

(6) nice / renice 명령어

프로세스를 실행하는데 있어서 프로세스에게 우선권을 부여 할 수 있다. 일반적으로 프로세스들은 설정된 우선권 순위대로 실행이 되는데 nice명령을 사용하게 되면 프로세스의 실행 우선권을 바꿀 수 있다. 우선권 순위는 -20에서 19까지의 범위를 가지며 nice 값이 적을수록 우선순위 값이 높아지게 된다.

프로세스의 우선순위: 프로세스가 운영체제의 CPU를 선점할 수 있는 권한

(프로세스의 우선순위를 조정하는 명령어)
■ nice 명령어 : 프로그램을 실행할 때 프로세스의 우선순위를 설정할 수 있는 명령어
■ renice 명령어 : 실행중인 프로그램의 우선순위를 조정할 수 있는 명령어

```
#  ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S     0  4768  4744  0  75   0 -   644 wait   pts/1    00:00:00 bash
```

(명령어 형식)
```
# CMD                          (EX: # backup.sh)
# nice -(-20 ~ 19)   CMD   (EX: # nice  -10 backup.sh (NI:  10))
# nice -n (-20 ~ 19) CMD   (EX: # nice --10 backup.sh (NI: -10))

# renice (-20 ~ 20)    PID (EX: # renice  10 PID (NI:  10))
# renice -n (-20 ~ 20) PID (EX: # renice -10 PID (NI: -10))
```

[EX] nice 명령어 실습
```
# sleep 500 &
[1] 8077

# sleep 600 &
[2] 8082
```

```
# nice -(-20 ~ 19)   CMD
# nice -n (-20 ~ 19) CMD
```

```
# nice -15 sleep 700 &    (# nice -n 15 sleep 700 &)
[3] 8083

# ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S     0  4943  4914  0  75   0 -  1163 wait   pts/1    00:00:00 bash
0 S     0  6438  4943  0  78   0 -   954 -      pts/1    00:00:00 sleep
0 S     0  6442  4943  0  77   0 -   954 -      pts/1    00:00:00 sleep
0 S     0  6459  4943  0  92  15 -   954 -      pts/1    00:00:00 sleep
4 R     0  6461  4943  0  77   0 -  1080 -      pts/1    00:00:00 ps

# nice --15 sleep 800 &   (# nice -n -15 sleep 800 &)
# ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S     0  4943  4914  0  76   0 -  1163 wait   pts/1    00:00:00 bash
0 S     0  6438  4943  0  78   0 -   954 -      pts/1    00:00:00 sleep
0 S     0  6442  4943  0  77   0 -   954 -      pts/1    00:00:00 sleep
0 S     0  6459  4943  0  92  15 -   954 -      pts/1    00:00:00 sleep
4 S     0  6478  4943  0  62 -15 -   954 -      pts/1    00:00:00 sleep
4 R     0  6480  4943  0  78   0 -  1080 -      pts/1    00:00:00 ps
```

```
# renice (-20 ~ 20)    PID
# renice -n (-20 ~ 20) PID
```

```
# renice -5 6459
6459: old priority 15, new priority -5

# ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S     0  4943  4914  0  75   0 -  1163 wait   pts/1    00:00:00 bash
0 S     0  6438  4943  0  78   0 -   954 -      pts/1    00:00:00 sleep
0 S     0  6442  4943  0  77   0 -   954 -      pts/1    00:00:00 sleep
0 S     0  6459  4943  0  72  -5 -   954 -      pts/1    00:00:00 sleep
4 S     0  6478  4943  0  62 -15 -   954 -      pts/1    00:00:00 sleep
4 R     0  6542  4943  0  77   0 -  1080 -      pts/1    00:00:00 ps
```

```
# renice 5 6459
6459: old priority -5, new priority 5

# ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S     0  4943  4914  0  75   0 -  1163 wait   pts/1    00:00:00 bash
0 S     0  6438  4943  0  78   0 -   954 -      pts/1    00:00:00 sleep
0 S     0  6442  4943  0  77   0 -   954 -      pts/1    00:00:00 sleep
0 S     0  6459  4943  0  82   5 -   954 -      pts/1    00:00:00 sleep
4 S     0  6478  4943  0  62 -15 -   954 -      pts/1    00:00:00 sleep
4 R     0  6560  4943  0  77   0 -  1080 -      pts/1    00:00:00 ps
```

(실무 예) nice, renice 사용 예
(백업스크립트/데이터수집 스크립트)
(X) # /root/bin/backup.sh &
(O) # nice -n 10 /root/bin/backup.sh &
(부하량을 주는 프로그램)
top
renice 10 PID (PID : 부하량을 주는 프로그램's PID)