

Linux System Administration

Linux Internal Structure : 리눅스 내부 구조

리눅스 내부 구조

: 파일 시스템의 이해

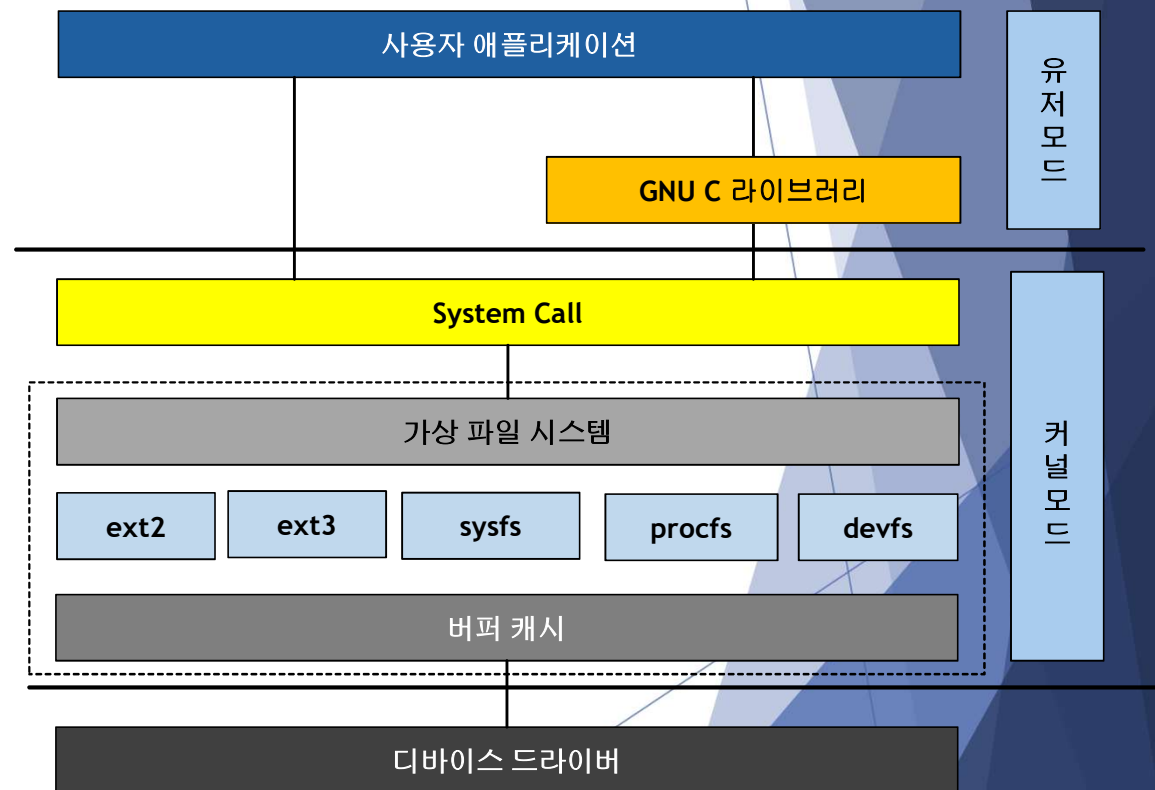
▶ 파일 시스템

- ▶ 운영체제는 커널 이미지, 시스템 실행과 관련된 시스템 파일, 그리고 유틸리티 파일 등을 제공
- ▶ 사용자의 데이터 저장을 위해 사용됨
- ▶ 파일 시스템을 통해 이러한 파일들을 관리
- ▶ 파일 시스템은 파일의 저장, 읽기, 삭제 등의 파일 관리 기능과 파일에 대한 접근 제어 기능을 제공
- ▶ 윈도우에서는 **FAT32**, **NTFS** 등의 파일 시스템을 제공, 리눅스에서는 **EXT3**, **EXT4** 등의 파일 시스템을 제공
- ▶ 디렉터리 안에 디렉터리를 저장할 수 있는 트리형 구조
- ▶ 루트 디렉터리 : 장치의 메인 디렉터리
 - ▶ 여러 장치(하드디스크)를 사용하면 루트 디렉터리 구분에 문제가 생김
 - ▶ 윈도우에서는 분리형 루트라 불리는 방식으로 이를 해결
 - ▶ 유닉스 계열에서는 통합형 파일 시스템을 사용

리눅스 내부 구조

: 가상 파일 시스템(VFS)

- ▶ 유닉스는 디스크, 터미널, 네트워크 카드 등 모든 주변장치들을 파일로 취급
- ▶ 디스크상의 파일 시스템 외에도 다양한 기능의 특수 파일 시스템이 존재
- ▶ 이러한 다양한 파일 시스템을 하나의 파일 시스템처럼 사용할 수 있도록 가상 파일 시스템(VFS) 구조를 사용



리눅스 내부 구조

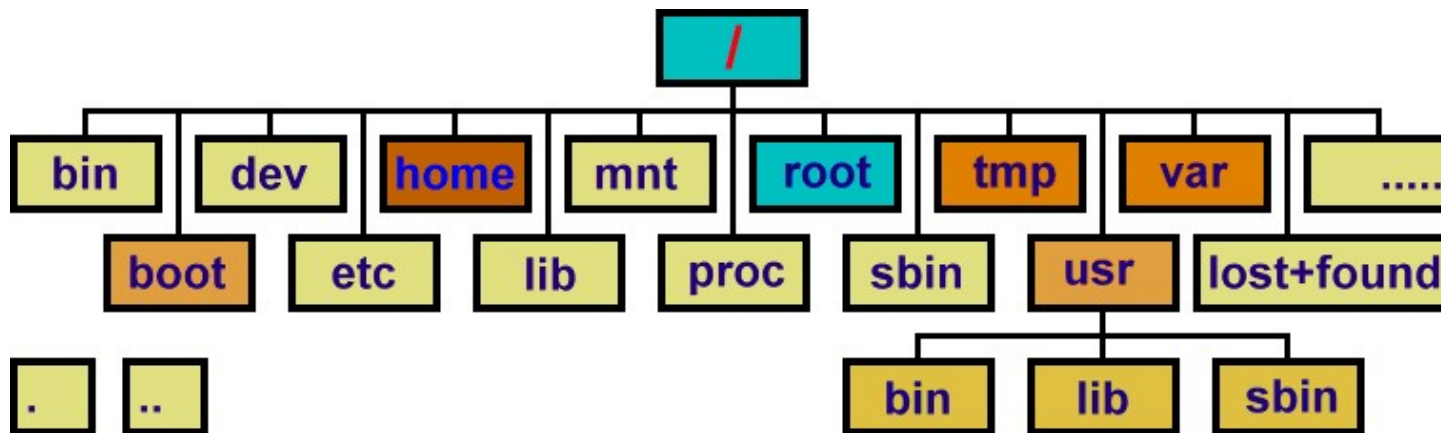
: 리눅스의 가상 파일 시스템

- ▶ 리눅스 커널의 특징 중 하나이며 유닉스에 비해 초창기부터 가상 파일 시스템을 지원
- ▶ 모든 파일 시스템을 하나의 파일 시스템으로 보이게 하는 계층(layer)이라 할 수 있음
- ▶ 파일, 디렉터리, 특수 파일 등을 파일 처리 시스템 호출(System Call)을 통해 일관적으로 조작할 수 있음
- ▶ 주요 특수 파일 시스템
 - ▶ **procfs** : 커널 및 커널 모듈(디바이스 드라이버) 정보를 참조하거나 설정 변경을 위한 파일 시스템 -> /proc 에 마운트됨
 - ▶ **sysfs** : 시스템에 접속된 디바이스 정보를 참조하거나 설정 변경을 위한 파일 시스템 -> /sys 에 마운트
 - ▶ **devfs** : 물리 디바이스에 액세스하기 위한 디바이스 파일을 배치하는 파일 시스템 -> /dev 에 마운트

리눅스 내부 구조

: 리눅스의 디렉터리 구조

- ▶ 리눅스 배포판은 **FHS** 표준에 따르도록 권장
- ▶ 강제사항은 아니지만, 대부분 배포판은 이 표준을 준수
- ▶ 각각의 디렉터리는 그에 맞는 용도가 있음
- ▶ 안전하고 편리한 시스템 운용을 위해서는 용도를 잘 알고, 그에 맞게 활용해야 함



리눅스 내부 구조

: 리눅스의 디렉터리 구조

▶ / : 루트 디렉터리

- ▶ 시스템의 근간이 되는 가장 중요한 디렉터리. 모든 파티션, 디렉터리는 루트 디렉터리 아래 위치하므로 반드시 있어야 함

▶ /bin

- ▶ 시스템 관리자 혹은 일반 사용자가 실행할 수 있는 명령어들이 위치
- ▶ 예) cat, chmod, date, ls, mkdir, rm, touch, vi 등

▶ /sbin

- ▶ 시스템 관리자가 사용할 수 있는 명령어들이 위치.
- ▶ 시스템 수정, 복구에 관한 많은 명령어들이 포함되므로 일반 사용자의 실행 권한 제한 등 보안에 신경을 써야 함
- ▶ 예) ifconfig, reboot, shutdown, halt, mount, fsck 등

리눅스 내부 구조

: 리눅스의 디렉터리 구조

- ▶ **/boot**
 - ▶ 부트로더와 부팅에 관련된 파일들을 포함
 - ▶ 손상되면 시스템이 부팅되지 않으므로 특별한 목적이 아니면 접근하지 말아야 함
- ▶ **/home**
 - ▶ 사용자들의 계정 홈 디렉터리가 하위 디렉터리로 존재
- ▶ **/dev**
 - ▶ 디바이스 파일들이 위치
 - ▶ 시스템의 모든 장치가 파일로 표현되어 있으며 **udev**라는 데몬이 이곳의 장치 파일을 관리
 - ▶ 예) `/dev/sda`, `/dev/hda`, `/dev/tty1`, `/dev/pts/0` 등
- ▶ **/lib**
 - ▶ 시스템의 프로그램이 실행될 때 필요한 공유 라이브러리들을 포함
 - ▶ 특별한 일이 없으면 변경하거나 삭제하지 않는 것이 좋음

리눅스 내부 구조

: 리눅스의 디렉터리 구조

▶ /etc

- ▶ 시스템 혹은 각종 프로그램들의 환경 설정 파일들이 위치
- ▶ 시스템 관리에서는 주로 이곳의 파일들을 수정(백업 권장)
- ▶ 예) /etc/fstab, /etc/group, /etc/passwd, /etc/sysconfig/i18n 등

▶ /mnt

- ▶ 마운트를 위한 임시 디렉터리가 위치. 광학 드라이브나 **USB** 등 이동 디스크를 마운트할 때 이용

▶ /root

- ▶ root 계정의 홈 디렉터리. root 계정만 접근 가능

▶ /var

- ▶ log 파일 등 수시로 업데이트 되는 파일들이 위치
- ▶ 시스템 운영에 필요한 파일들도 위치하므로 수정과 삭제에 주의해야 함

리눅스 내부 구조

: 리눅스의 디렉터리 구조

▶ /proc

- ▶ 실행중인 프로세스 정보, CPU, 메모리 등 시스템 정보가 가상 파일로 저장
- ▶ 대부분 읽기 전용이지만 일부 쓰기가 가능한 파일들이 있는데, 이런 파일들은 커널의 기능을 변경할 수 있음

▶ [실습] CPU 정보 출력

```
[root@lx ~]# cat /proc/cpuinfo
processor       : 0
vendor_id     : GenuineIntel
cpu family    : 6
model         : 42
model name    : Intel(R) Core(TM) i7-2670QM CPU @ 2.20GHz
stepping      : 7
cpu MHz       : 2195.022
cache size    : 6144 KB
physical id   : 0
siblings      : 1
core id       : 0
cpu cores     : 1
apicid        : 0
initial apicid : 0
fpu           : yes
fpu_exception : yes
cpuid level   : 13
wp            : yes
flags         : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 syscall
nx rdtscp lm constant_tsc rep_good nopl xtopology nonstop_tsc pni pclmulqdq monitor ssse3 cx16 sse4_1 sse4_2 popcnt
aes xsave avx hypervisor lahf_lm
bogomips      : 4390.04
clflush size  : 64
...
```

리눅스 내부 구조

: 리눅스의 디렉터리 구조

- ▶ [실습] 파티션 정보를 출력해 봅니다

```
[root@lx ~]# cat /proc/partitions
major minor #blocks name
11      0  1048575 sr0
 8      0 25165824 sda
 8      1   512000 sda1
 8      2 2098176 sda2
 8      3 22554624 sda3
[root@lx ~]#
power management:
```

- ▶ [실습] 다음 정보도 출력해 봅시다
 - ▶ /proc/meminfo
 - ▶ /proc/uptime
 - ▶ /proc/filesystems
 - ▶ /proc/version
 - ▶ /proc/modules
 - ▶ /proc/loadavg

리눅스 내부 구조

: 리눅스의 디렉터리 구조

- ▶ `/usr/bin`
 - ▶ 응용 프로그램들의 실행파일들이 위치
- ▶ `/usr/sbin`
 - ▶ 시스템 관리를 위한 명령어들이 위치
- ▶ `/usr/include`
 - ▶ 시스템, 네트워크 프로그래밍을 위한 **C** 헤더 파일
- ▶ `/usr/lib`
 - ▶ `/usr/bin`, `/usr/sbin` 에 있는 실행 파일들을 위한 라이브러리들이 위치
- ▶ `/tmp`
 - ▶ 임시로 파일을 만들고 삭제하는 공간
- ▶ `/lost+found`
 - ▶ 부팅시 파일 시스템에 문제가 생길 경우 **fsck** 명령어로 복구할 때 사용하는 디렉터리

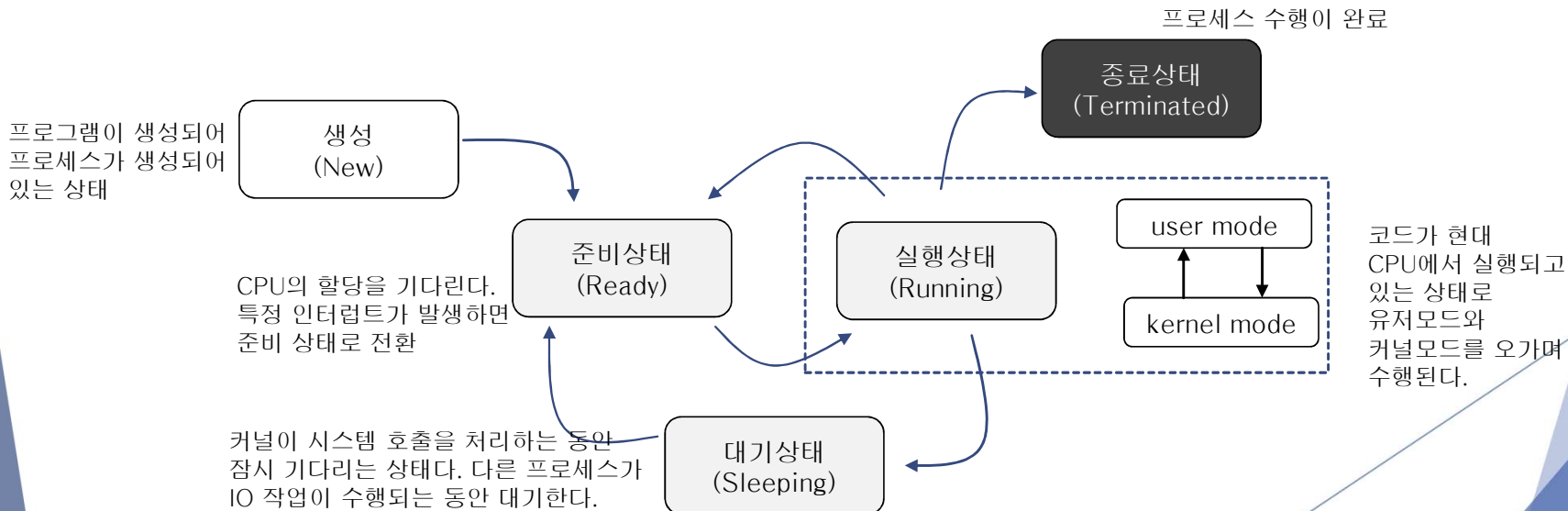
리눅스 내부 구조

: 프로세스

▶ 프로세스

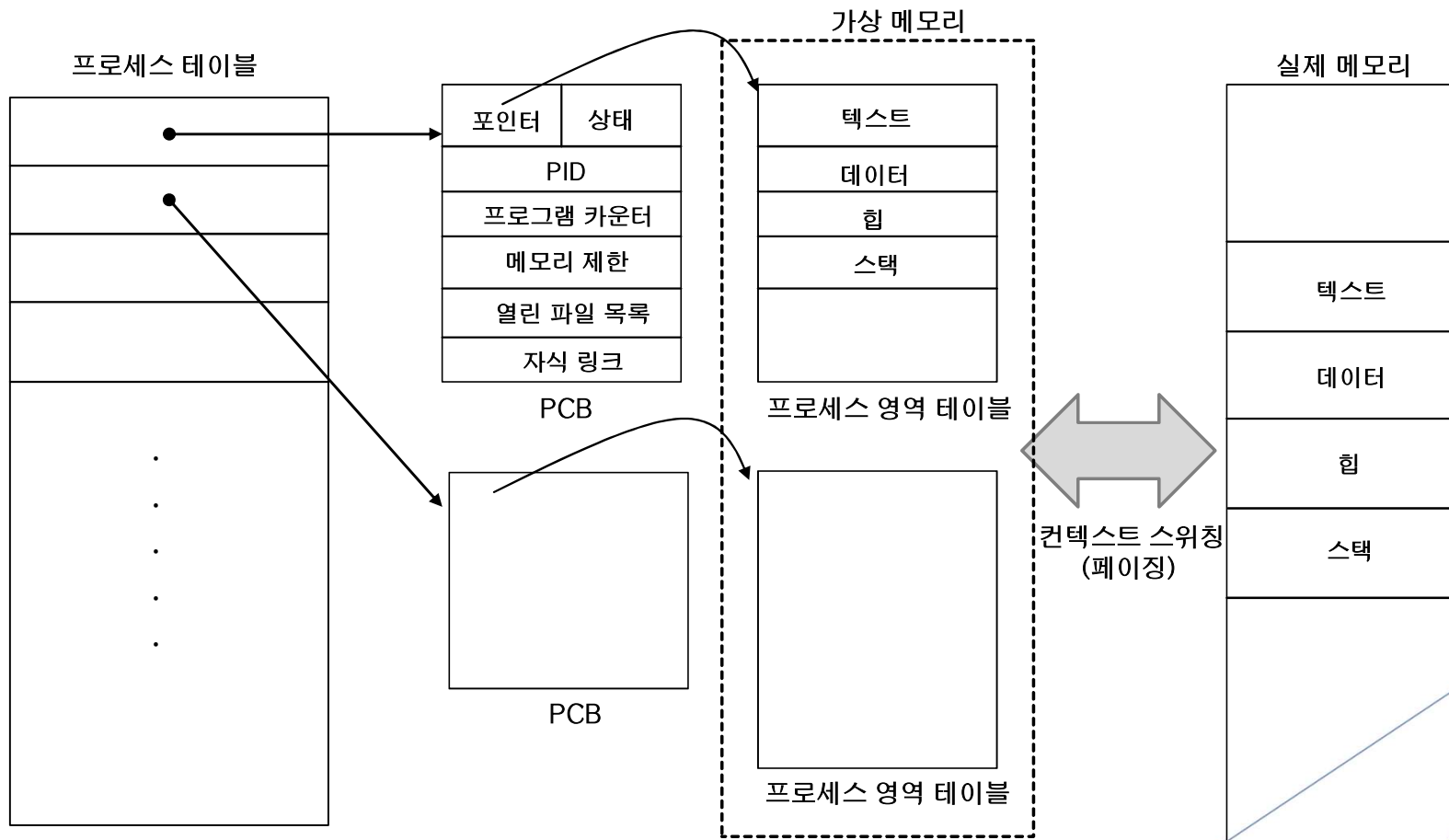
- ▶ 유닉스는 시분할 시스템으로 여러 개의 프로그램을 동시에 실행(멀티 태스킹)
- ▶ 컴퓨터 내에서 실행 중인 프로그램을 프로세스(process) 또는 태스크(Task)라 함
- ▶ 여러 프로세스가 동시에 실행되는 것을 멀티 프로세스라 함

▶ 프로세스의 상태



리눅스 내부 구조

: 프로세스의 자료 구조



리눅스의 내부 구조

: 프로세스 정보

- ▶ 리눅스에서 프로세스 정보는 **task_struct** 구조체를 통해 관리. 이 구조체는 프로세스의 모든 정보를 보관하는 프로세스 서술자로 아주 많은 구조체로 이루어짐
- ▶ 이 정보는 **ps** 명령을 이용하여 가져올 수 있음
- ▶ 현재 실행되는 프로세스 정보는 **/proc** 디렉터리를 통해 확인 가능
- ▶ 실행중인 프로세스에 대한 정보는 **/proc** 디렉터리 내부의 **PID**로 되어 있는 디렉터리에 있으며 **status** 파일을 이용, 확인할 수 있다

: 프로세스 정보

- ```
[root@lx ~]# ps
PID TTY TIME CMD
14877 pts/0 00:00:00 su
14881 pts/0 00:00:00 bash
14896 pts/0 00:00:00 ps

[root@lx ~]# cat /proc/14881/status
Name: bash
State: S (sleeping)
Tgid: 14881
Ngid: 0
Cpus_allowed_list: 0
Mems_allowed:
 00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000
0,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,000
0000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,000
00000001
Mems_allowed_list: 0
voluntary_ctxt_switches: 55
nonvoluntary_ctxt_switches: 0

[root@lx ~]# ^C
```

# 리눅스의 내부 구조

## : 프로세스 관리 명령어 - ps

- ▶ **ps** : 현재 실행되고 있는 프로세스의 목록을 보여줌
  - ▶ 사용법
    - ▶ **ps [옵션]**
  - ▶ 옵션
    - ▶ **-l** : 자세한 정보를 출력
    - ▶ **-a** : 다른 사용자들의 프로세스도 보여줌
    - ▶ **-u** : 프로세스의 사용자 이름과 시작 시간을 출력
    - ▶ **-x** : 터미널과 연결되지 않은 프로세스도 출력
    - ▶ **-e** : 환경을 보여줌
    - ▶ **-f** : 프로세스의 정보를 한 줄로 자세히 출력
    - ▶ **-r** : 현재 실행중인 프로세스를 출력
    - ▶ **-j** : 작업 중심의 형태로 출력
    - ▶ **-c** : 커널 **task\_struct** 구조체 형태로 보여줌
- ▶ 보통 **-aux** 또는 **-ef** 옵션을 사용하여 프로세스 상태를 확인



# 리눅스의 내부 구조

## : 프로세스 관리 명령어 - ps

- ▶ [실습] -ef 옵션을 사용하여 프로세스 상태를 확인해 봅니다

```
[root@lx ~]# ps -ef
UID PID PPID C STIME TTY TIME CMD
root 1 0 0 2월 17 ? 00:00:01 /usr/lib/systemd/systemd --switched-root --system --deseri
root 2 0 0 2월 17 ? 00:00:00 [kthreadd]
root 3 2 0 2월 17 ? 00:00:00 [ksoftirqd/0]
root 6 2 0 2월 17 ? 00:00:00 [kworker/u2:0]
root 7 2 0 2월 17 ? 00:00:00 [migration/0]
root 8 2 0 2월 17 ? 00:00:00 [rcu_bh]
.
.
.
root 14877 14858 0 05:26 pts/0 00:00:00 su -
root 14881 14877 0 05:26 pts/0 00:00:00 -bash
root 14898 2 0 05:28 ? 00:00:00 [kworker/0:2H]
postfix 14900 857 0 05:32 ? 00:00:00 pickup -l -t unix -u
root 14901 2 0 05:33 ? 00:00:00 [kworker/0:0H]
root 14913 2 0 05:53 ? 00:00:00 [kworker/0:1]
root 14914 2 0 05:58 ? 00:00:00 [kworker/0:0]
postfix 14942 857 0 06:01 ? 00:00:00 cleanup -z -t unix -u
postfix 14944 857 0 06:01 ? 00:00:00 trivial-rewrite -n rewrite -t unix -u
postfix 14945 857 0 06:01 ? 00:00:00 local -t unix
root 14946 14881 0 06:01 pts/0 00:00:00 ps -ef
```

**USER:** 프로세스 소유자의 계정   **PID:** 프로세스를 구분하는 프로세스 아이디   **PPID:** 부모 프로세스 PID  
**STIME:** 프로세스 시작 시간   **TTY:** 프로세스의 표준 입출력을 담당하는 터미널   **TIME:** 프로세스의 CPU 점유시간  
**CMD:** 실행 명령어

# 리눅스의 내부 구조

## : 프로세스 관리 명령어 - ps

- ▶ [실습] -aux 옵션을 사용하여 프로세스 상태를 확인해 봅니다

```
USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
root 1 0.0 0.3 44364 7056 ? Ss 2월17 0:01 /usr/lib/systemd/systemd --switched-roo
root 2 0.0 0.0 0 0 ? S 2월17 0:00 [kthreadd]
root 3 0.0 0.0 0 0 ? S 2월17 0:00 [ksoftirqd/0]
root 6 0.0 0.0 0 0 ? S 2월17 0:00 [kworker/u2:0]
root 7 0.0 0.0 0 0 ? S 2월17 0:00 [migration/0]
root 8 0.0 0.0 0 0 ? S 2월17 0:00 [rcu_bh]
root 9 0.0 0.0 0 0 ? S 2월17 0:00 [rcuob/0]
root 10 0.0 0.0 0 0 ? R 2월17 0:00 [rcu_sched]
root 11 0.0 0.0 0 0 ? S 2월17 0:02 [rcuos/0]
root 12 0.0 0.0 0 0 ? S 2월17 0:00 [watchdog/0]
.
.
.
root 14881 0.0 0.1 115504 2160 pts/0 S 05:26 0:00 -bash
root 14898 0.0 0.0 0 0 ? S< 05:28 0:00 [kworker/0:2H]
postfix 14900 0.0 0.1 91232 3892 ? S 05:32 0:00 pickup -l -t unix -u
root 14901 0.0 0.0 0 0 ? S< 05:33 0:00 [kworker/0:0H]
root 14947 0.0 0.0 0 0 ? S 06:03 0:00 [kworker/0:1]
root 14949 0.0 0.0 0 0 ? S 06:08 0:00 [kworker/0:0]
root 14950 0.0 0.0 139496 1656 pts/0 R+ 06:08 0:00 ps -aux
```

**USER:** 프로세스 소유자의 계정 **PID:** 프로세스를 구분하는 프로세스 아이디 **%CPU:** 마지막 분 동안 사용한 CPU의 %  
**%MEM:** 마지막 분 동안 사용한 메모리 양의 % **VSZ:** 프로세스 데이터 스택의 크기 **RSS:** 실제 메모리 양  
**COMMAND:** 실행 명령어 **STAT:** 프로세스의 상태 **START:** 프로세스가 시작된 시간

**stat:**

p: 수행가능, T: 일시 정지, D: 디스크 입출력 대기, S: 20초 미만의 짧은 휴식, I: 20초 이상의 긴 휴식, Z: 좀비 상태

# 리눅스 내부 구조

## : 프로세스 관리 명령어 - pstree

- ▶ **pstree** : 프로세스 정보를 트리 형태로 보여줌
  - ▶ 사용법
    - ▶ **pstree** [옵션]
  - ▶ 옵션
    - ▶ **-n** : PID 순으로 정렬
    - ▶ **-p** : 프로세스명 + PID
- ▶ [실습] **pstree** 명령으로 프로세스의 정보를 확인해 봅시다

```
[root@lx ~]# pstree -n
systemd└─systemd-journal
 └─systemd-udevd
 └─auditd───{auditd}
 └─systemd-logind
 └─NetworkManager└─2*[{NetworkManager}]
 └─dhclient
 └─rsyslogd──2*[{rsyslogd}]
 └─dbus-daemon──{dbus-daemon}
 └─crond
 └─wpa_supplicant
 └─polkitd──5*[{polkitd}]
 └─tuned──4*[{tuned}]
 └─sshd──sshd──sshd──bash──su──bash──pstree
 └─master└─qmgr
 └─pickup
 └─httpd└─httpd
 └─4*[httpd──26*[{httpd}]]
 └─agetty
```

# 리눅스 내부 구조

: 프로세스 관리 명령어 - top

▶ top : 프로세스의 CPU, Memory 사용량 등 전반적 상황을 실시간으로 모니터링

▶ 사용법

▶ top [옵션]

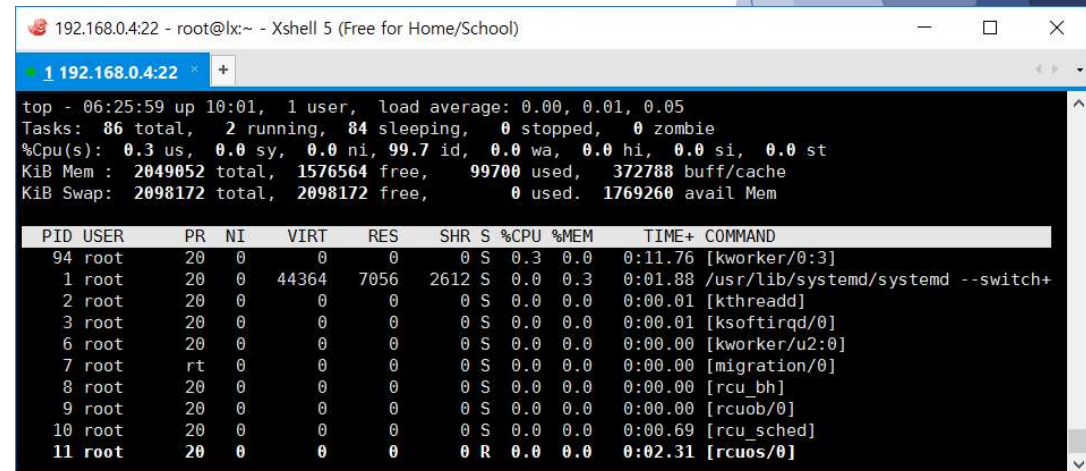
▶ 옵션

▶ -d : 시간, 화면 갱신 시간 지정

▶ -c : 명령행 전체를 보여줌

▶ -q : 화면을 계속 갱신

▶ [실습] top 명령으로 프로세스를 확인해 봅니다



```
192.168.0.4:22 - root@lx:~ - Xshell 5 (Free for Home/School)
1 192.168.0.4:22
top - 06:25:59 up 10:01, 1 user, load average: 0.00, 0.01, 0.05
Tasks: 86 total, 2 running, 84 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.3 us, 0.0 sy, 0.0 ni, 99.7 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 2049052 total, 1576564 free, 99700 used, 372788 buff/cache
KiB Swap: 2098172 total, 2098172 free, 0 used. 1769260 avail Mem

 PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
 94 root 20 0 0 0 0 S 0.3 0.0 0:11.76 [kworker/0:3]
 1 root 20 0 44364 7056 2612 S 0.0 0.3 0:01.88 /usr/lib/systemd/systemd --switch+
 2 root 20 0 0 0 0 S 0.0 0.0 0:00.01 [kthreadd]
 3 root 20 0 0 0 0 S 0.0 0.0 0:00.01 [ksoftirqd/0]
 6 root 20 0 0 0 0 S 0.0 0.0 0:00.00 [kworker/u2:0]
 7 root rt 0 0 0 0 S 0.0 0.0 0:00.00 [migration/0]
 8 root 20 0 0 0 0 S 0.0 0.0 0:00.00 [rcu_bh]
 9 root 20 0 0 0 0 S 0.0 0.0 0:00.00 [rcuob/0]
 10 root 20 0 0 0 0 S 0.0 0.0 0:00.69 [rcu_sched]
 11 root 20 0 0 0 0 R 0.0 0.0 0:02.31 [rcuos/0]
```

# 리눅스 내부 구조

: 프로세스 관리 명령어 - top

top 첫 번째 줄

| 이름           | 설명                |
|--------------|-------------------|
| up           | 리눅스 부팅 후 총 구동 시간  |
| users        | 접속하여 사용중인 총 사용자 수 |
| load average | 시스템 평균 부하         |

top 두 번째 줄 - Tasks

| 이름       | 설명                                |
|----------|-----------------------------------|
| total    | 전체 프로세스 수                         |
| running  | 현재 실행되고 있는 프로세스 수                 |
| sleeplng | 백그라운드에서 잠자고 있는(대기 모드) 프로세스 수      |
| stopped  | 실행을 일시적으로 중단하고 있는 프로세스 수          |
| zombie   | 실행을 종료했지만 어떤 이유로 메모리에 남아있는 프로세스 수 |

```
morenice
top - 09:59:05 up 48 days, 12:30, 29 users, load average: 0.00, 0.00, 0.00
Tasks: 249 total, 1 running, 247 sleeping, 1 stopped, 0 zombie
Cpu(s): 0.0%us, 0.0%sy, 0.0%ni,100.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 8176444k total, 8067892k used, 108552k free, 566432k buffers
Swap: 0k total, 0k used, 0k free, 6859604k cached

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
31368 morenice 20 0 18960 1456 960 R 0 0.0 0:00.17 top
1 root 20 0 3720 144 52 S 0 0.0 1:21.01 init
2 root 15 -5 0 0 0 S 0 0.0 0:00.00 kthreadd
3 root RT -5 0 0 0 S 0 0.0 0:02.72 migration/0
4 root 15 -5 0 0 0 S 0 0.0 0:10.15 ksoftirqd/0
5 root RT -5 0 0 0 S 0 0.0 0:00.04 watchdog/0
6 root RT -5 0 0 0 S 0 0.0 0:02.74 migration/1
7 root 15 -5 0 0 0 S 0 0.0 0:09.07 ksoftirqd/1
8 root RT -5 0 0 0 S 0 0.0 0:00.03 watchdog/1
9 root RT -5 0 0 0 S 0 0.0 0:02.77 migration/2
10 root 15 -5 0 0 0 S 0 0.0 0:09.19 ksoftirqd/2
11 root RT -5 0 0 0 S 0 0.0 0:00.02 watchdog/2
12 root RT -5 0 0 0 S 0 0.0 0:02.74 migration/3
13 root 15 -5 0 0 0 S 0 0.0 0:09.52 ksoftirqd/3
14 root RT -5 0 0 0 S 0 0.0 0:00.02 watchdog/3
15 root RT -5 0 0 0 S 0 0.0 0:04.03 migration/4
16 root 15 -5 0 0 0 S 0 0.0 0:43.86 ksoftirqd/4
17 root RT -5 0 0 0 S 0 0.0 0:00.05 watchdog/4
18 root RT -5 0 0 0 S 0 0.0 0:04.09 migration/5
19 root 15 -5 0 0 0 S 0 0.0 0:43.29 ksoftirqd/5
20 root RT -5 0 0 0 S 0 0.0 0:00.04 watchdog/5
21 root RT -5 0 0 0 S 0 0.0 0:04.10 migration/6
22 root 15 -5 0 0 0 S 0 0.0 0:43.05 ksoftirqd/6
23 root RT -5 0 0 0 S 0 0.0 0:00.04 watchdog/6
24 root RT -5 0 0 0 S 0 0.0 0:04.12 migration/7
25 root 15 -5 0 0 0 S 0 0.0 0:44.11 ksoftirqd/7
26 root RT -5 0 0 0 S 0 0.0 0:00.04 watchdog/7
```



# 리눅스 내부 구조

: 프로세스 관리 명령어 - top

top 세 번째 줄 - Cpu(s)

| 이름 | 설명                                      |
|----|-----------------------------------------|
| us | 사용자 어플리케이션에 할당 된 CPU 비중                 |
| sy | 시스템 어플리케이션에 할당 된 CPU 비중                 |
| ni | CPU 우선순위를 낮추기 위해 (nice) 할당 된 CPU 비중     |
| id | idle (휴식) 상태의 CPU 비중                    |
| wa | I/O를 기다리는 프로세스에 할당 된 CPU 비중             |
| hi | 하드웨어 인터럽트를 기다리는 프로세스에 할당 된 CPU 비중       |
| si | 소프트웨어 인터럽트를 기다리는 프로세스에 할당 된 CPU 비중      |
| st | 하이퍼바이저 (가상플랫폼을 실행하는 소프트웨어)에 할당 된 CPU 비중 |

```

top - 09:59:05 up 48 days, 12:30, 29 users, load average: 0.00, 0.00, 0.00
Tasks: 249 total, 1 running, 247 sleeping, 1 stopped, 0 zombie
Cpu(s): 0.0%us, 0.0%sy, 0.0%ni,100.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 8176444k total, 8067892k used, 108552k free, 566432k buffers
Swap: 0k total, 0k used, 0k free, 6859604k cached

 PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
31368 morenice 20 0 18960 1456 960 R 0.0 0.0 0:00.17 top
 1 root 20 0 3720 144 52 S 0.0 0.0 1:21.01 init
 2 root 15 -5 0 0 0 S 0.0 0.0 0:00.00 kthreadd
 3 root RT -5 0 0 0 S 0.0 0.0 0:02.72 migration/0
 4 root 15 -5 0 0 0 S 0.0 0.0 0:10.15 ksoftirqd/0
 5 root RT -5 0 0 0 S 0.0 0.0 0:00.04 watchdog/0
 6 root RT -5 0 0 0 S 0.0 0.0 0:02.74 migration/1
 7 root 15 -5 0 0 0 S 0.0 0.0 0:09.07 ksoftirqd/1
 8 root RT -5 0 0 0 S 0.0 0.0 0:00.03 watchdog/1
 9 root RT -5 0 0 0 S 0.0 0.0 0:02.77 migration/2
 10 root 15 -5 0 0 0 S 0.0 0.0 0:09.19 ksoftirqd/2
 11 root RT -5 0 0 0 S 0.0 0.0 0:00.02 watchdog/2
 12 root 15 -5 0 0 0 S 0.0 0.0 0:02.74 migration/3
 13 root 15 -5 0 0 0 S 0.0 0.0 0:09.52 ksoftirqd/3
 14 root RT -5 0 0 0 S 0.0 0.0 0:00.02 watchdog/3
 15 root 15 -5 0 0 0 S 0.0 0.0 0:04.03 migration/4
 16 root 15 -5 0 0 0 S 0.0 0.0 0:43.86 ksoftirqd/4
 17 root RT -5 0 0 0 S 0.0 0.0 0:00.05 watchdog/4
 18 root 15 -5 0 0 0 S 0.0 0.0 0:04.09 migration/5
 19 root 15 -5 0 0 0 S 0.0 0.0 0:43.29 ksoftirqd/5
 20 root RT -5 0 0 0 S 0.0 0.0 0:00.04 watchdog/5
 21 root 15 -5 0 0 0 S 0.0 0.0 0:04.10 migration/6
 22 root 15 -5 0 0 0 S 0.0 0.0 0:43.05 ksoftirqd/6
 23 root RT -5 0 0 0 S 0.0 0.0 0:00.04 watchdog/6
 24 root 15 -5 0 0 0 S 0.0 0.0 0:04.12 migration/7
 25 root 15 -5 0 0 0 S 0.0 0.0 0:44.11 ksoftirqd/7
 26 root RT -5 0 0 0 S 0.0 0.0 0:00.04 watchdog/7

```

summary

task information

# 리눅스 내부 구조

: 프로세스 관리 명령어 - top

Process Table

| 이름      | 설명                             |
|---------|--------------------------------|
| PID     | 프로세스의 ID 번호                    |
| USER    | 프로세스를 소유한 사용자                  |
| PR      | 프로세스의 우선 순위                    |
| NI      | 프로세스의 nice 값                   |
| VIRT    | 프로세스가 소비하는 가상 메모리의 양           |
| RES     | 실제 상주하는 가상 메모리의 크기             |
| SHR     | 프로세스가 사용하고 있는 공유 메모리의 양        |
| S       | 프로세스 상태 (ex 잠자기 상태, 실행 중 상태 등) |
| %CPU    | CPU 사용 률                       |
| %MEM    | 메모리 사용 률                       |
| TIME+   | Task 가 시작된 이후 사용한 시간           |
| COMMAND | 명령어 이름                         |

```

top - 09:59:05 up 48 days, 12:30, 29 users, load average: 0.00, 0.00, 0.00
Tasks: 249 total, 1 running, 247 sleeping, 1 stopped, 0 zombie
Cpu(s): 0.0%us, 0.0%sy, 0.0%ni,100.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 8176444k total, 8067892k used, 108552k free, 566432k buffers
Swap: 0k total, 0k used, 0k free, 6859604k cached

 PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
31368 morenice 20 0 18960 1456 960 R 0.0 0.0 0:00.17 top
 1 root 20 0 3720 144 52 S 0.0 0.0 1:21.01 init
 2 root 15 -5 0 0 0 S 0.0 0.0 0:00.00 kthreadd
 3 root RT -5 0 0 0 S 0.0 0.0 0:02.72 migration/0
 4 root 15 -5 0 0 0 S 0.0 0.0 0:10.15 ksoftirqd/0
 5 root RT -5 0 0 0 S 0.0 0.0 0:00.04 watchdog/0
 6 root RT -5 0 0 0 S 0.0 0.0 0:02.74 migration/1
 7 root 15 -5 0 0 0 S 0.0 0.0 0:09.07 ksoftirqd/1
 8 root RT -5 0 0 0 S 0.0 0.0 0:00.03 watchdog/1
 9 root RT -5 0 0 0 S 0.0 0.0 0:02.77 migration/2
 10 root 15 -5 0 0 0 S 0.0 0.0 0:09.19 ksoftirqd/2
 11 root RT -5 0 0 0 S 0.0 0.0 0:00.02 watchdog/2
 12 root RT -5 0 0 0 S 0.0 0.0 0:02.74 migration/3
 13 root 15 -5 0 0 0 S 0.0 0.0 0:09.52 ksoftirqd/3
 14 root RT -5 0 0 0 S 0.0 0.0 0:00.02 watchdog/3
 15 root RT -5 0 0 0 S 0.0 0.0 0:04.03 migration/4
 16 root 15 -5 0 0 0 S 0.0 0.0 0:43.86 ksoftirqd/4
 17 root RT -5 0 0 0 S 0.0 0.0 0:00.05 watchdog/4
 18 root RT -5 0 0 0 S 0.0 0.0 0:04.09 migration/5
 19 root 15 -5 0 0 0 S 0.0 0.0 0:43.29 ksoftirqd/5
 20 root RT -5 0 0 0 S 0.0 0.0 0:00.04 watchdog/5
 21 root RT -5 0 0 0 S 0.0 0.0 0:04.10 migration/6
 22 root 15 -5 0 0 0 S 0.0 0.0 0:43.05 ksoftirqd/6
 23 root RT -5 0 0 0 S 0.0 0.0 0:00.04 watchdog/6
 24 root RT -5 0 0 0 S 0.0 0.0 0:04.12 migration/7
 25 root 15 -5 0 0 0 S 0.0 0.0 0:44.11 ksoftirqd/7
 26 root RT -5 0 0 0 S 0.0 0.0 0:00.04 watchdog/7

```

summary

task information

# 리눅스 내부 구조

## : 프로세스 관리 명령어 - top

- ▶ top은 다른 프로세스 관리 명령(ps, pstree 등)과는 달리, 인터랙티브 셸 프로그램
- ▶ 주요 단축키
  - ▶ q : 모니터링 종료
  - ▶ m : 메모리 사용량 순으로 정렬
  - ▶ t : 실행시간이 긴 순서로 정렬
  - ▶ r : 정렬 순서 변경
- ▶ [실습] top으로 모니터링 도중 단축키를 입력, 정렬 순서를 변경해 봅니다



# 리눅스 내부 구조

## : 프로세스 관리 명령어 - kill

- ▶ kill : 지정한 프로세스에 시그널을 보내는 명령
  - ▶ 사용법
    - ▶ kill [옵션] PID
  - ▶ 옵션
    - ▶ -l : 사용할 수 있는 시그널을 출력
    - ▶ -Signal\_ID : 프로세스에 Signal\_ID에 해당하는 시그널을 보냄
- ▶ [실습] -l 옵션을 이용, 사용할 수 있는 시그널 목록을 확인해 봅니다

```
[root@lx ~]# 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
```

# 리눅스 내부 구조

## : 프로세스 관리 명령어 - kill

- ▶ [실습] -9 (-KILL) 시그널을 프로세스에 전송, 프로세스를 죽여 봅니다

```
[root@lx ~]# ps -ef | grep httpd
root 13691 1 0 2월17 ? 00:00:01 /usr/local/apache/bin/httpd -k start
daemon 13692 13691 0 2월17 ? 00:00:00 /usr/local/apache/bin/httpd -k start
daemon 13693 13691 0 2월17 ? 00:00:00 /usr/local/apache/bin/httpd -k start
daemon 13694 13691 0 2월17 ? 00:00:00 /usr/local/apache/bin/httpd -k start
daemon 13695 13691 0 2월17 ? 00:00:00 /usr/local/apache/bin/httpd -k start
daemon 13779 13691 0 2월17 ? 00:00:00 /usr/local/apache/bin/httpd -k start
root 15001 14881 0 06:39 pts/0 00:00:00 grep --color=auto httpd
[root@lx ~]# kill -9 13691
[root@lx ~]# ps -ef | grep httpds
root 15003 14881 0 06:40 pts/0 00:00:00 grep --color=auto httpds
[root@lx ~]#
```

- ▶ [Tip] KILL 시그널을 받게 되면 프로그램은 **Clean Up** 코드를 거치지 않게 됩니다. 바로 -9(-KILL) 시그널을 보내기보다는 두 세번 정도 -15(-TERM) 시그널을 보내고 정상 종료 되지 않을 때 KILL 시그널을 보내는 것이 좋습니다

# 리눅스 내부 구조

## : 메모리 관리 - free

- ▶ free : 시스템의 메모리 정보를 출력
  - ▶ 사용법
    - ▶ free [옵션]
  - ▶ 옵션
    - ▶ -b : 바이트 단위 출력
    - ▶ -k : KB 단위 출력
    - ▶ -m : MB 단위 출력
    - ▶ -t : 총 합을 표시
- ▶ [실습] 메모리 사용량을 출력해 봅니다

```
[root@lx ~]# free
 total used free shared buff/cache available
Mem: 2049052 99024 1577308 25424 372720 1770020
Swap: 2098172 0 2098172
```