리눅스 시스템 관리

6.1 리눅스 파일 시스템

6.1.1 파일 시스템

1. 운영 체제에서는 커널 이미지, 시스템 실행과 관련된 시스템 파일 그리고 유틸리티 파일 등을 제공한다.

2. 사용자의 데이터의 저장을 위해서 사용 됨.

3. 파일 시스템을 통해 이러한 파일들을 관리된다.

4. 파일 시스템은 파일의 저장, 삭제 읽기 등의 파일 관리 기능과 파일에 대한 접근 제어 기능을 제공

5. 윈도우에서는 FAT32, NTFS와 같은 파일 시스템을 제공하고 리눅스 에서는 EXT2, EXT3 와 같은 파일 시스템을 제공

6. 디렉터리 안에 디렉터리를 저장할 수 있는 구조

7. 루트 디렉터리 : 장치의 메인 디렉터리

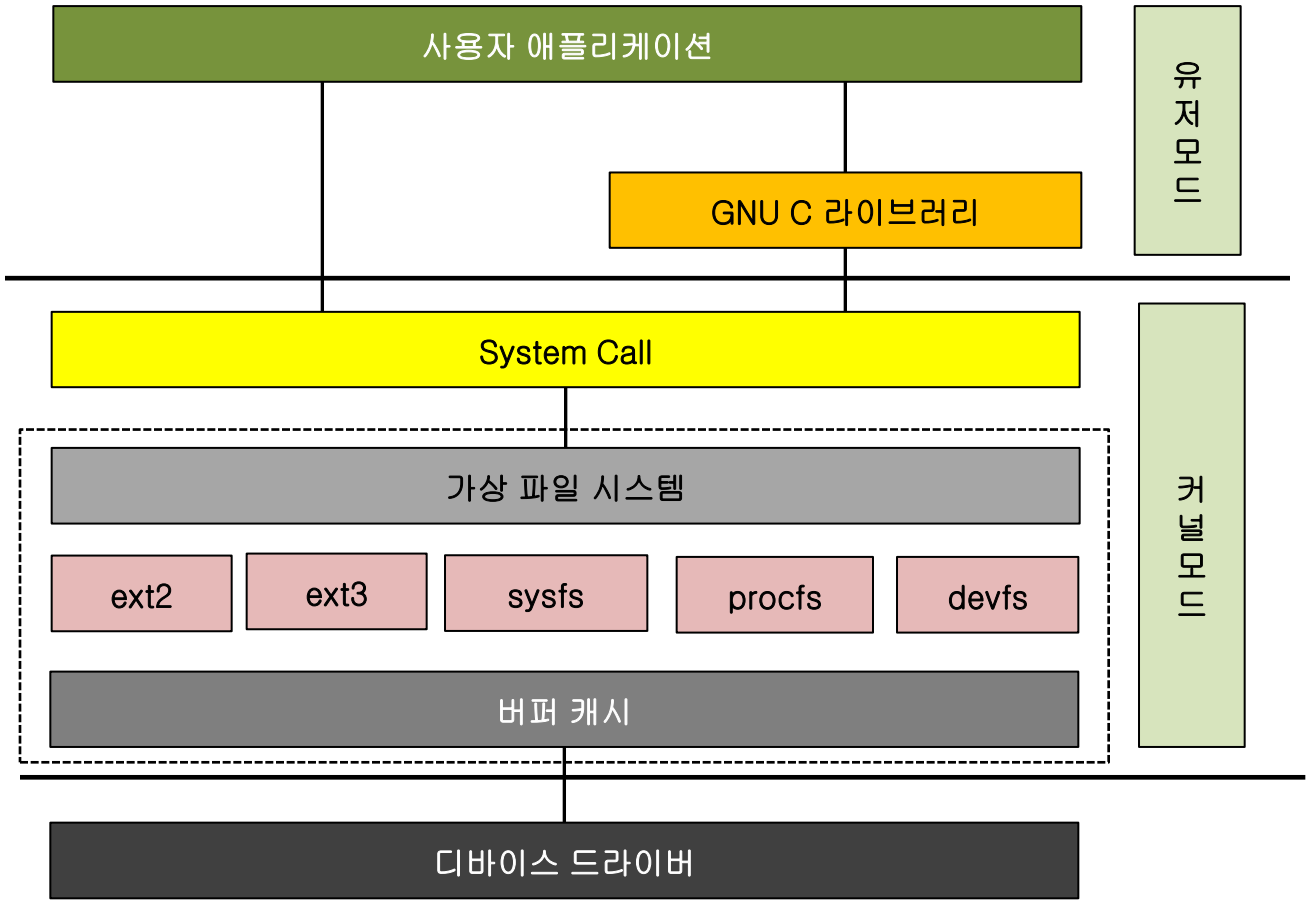
   - 여러 장치(하드 디스크)를 사용하게 되면 루트 디렉터리 구분에 문제가 생긴다.

   - MS 윈도우에서 **분리형 루트**라 불리는 방식으로 이를 해결한다.

   - 유닉스 계열에서는 **통합형 파일 시스템**을 사용한다.

6.1.2 가상 파일시스템(VFS)

유닉스는 디스크, 터미널, 네트워크 카드 등 모든 주변 장치들을 파일로 취급한다. 따라서 디스크상의 파일 시스템 외에도 다양한 기능의 특수 파일 시스템이 존재하는데 리눅스 에서는 이러한 다양한 파일 시스템들을 하나의 파일 시스템 처럼 사용할 수 도록 가상 파일 시스템이라는 구조를 사용 한다.



6.1 리눅스 파일 시스템

1. 리눅스 커널의 특징 중 하나이며 유닉스에 비해 리눅스는 초창기부터 가상 파일 시스템을 지원했다.

2. 모든 파일 시스템을 하나의 파일 시스템으로 보이게 하는 계층(layer)이라 할 수 있다.

3. 파일, 디렉터리, 특수 파일 등을 파일 처리 시스템 호출(system call)을 통해 일관적으로 조작할 수 있다.

주요 특수 파일 시스템

1. procfs : 커널 및 커널 모듈(디바이스 드라이버) 정보를 참조하거나, 설정 변경을 위한 파일 시스템으로 /proc에 마운트

2. sysfs : 시스템에 접속된 디바이스 정보를 참조하거나, 설정 변경을 위한 파일 시스템. /sys 에 마운트

3. devfs : 물리 디바이스에 액세스하기 위한 디바이스 파일을 배치하는 파일 시스템 . /dev 에 마운트

가상 파일시스템(VFS)

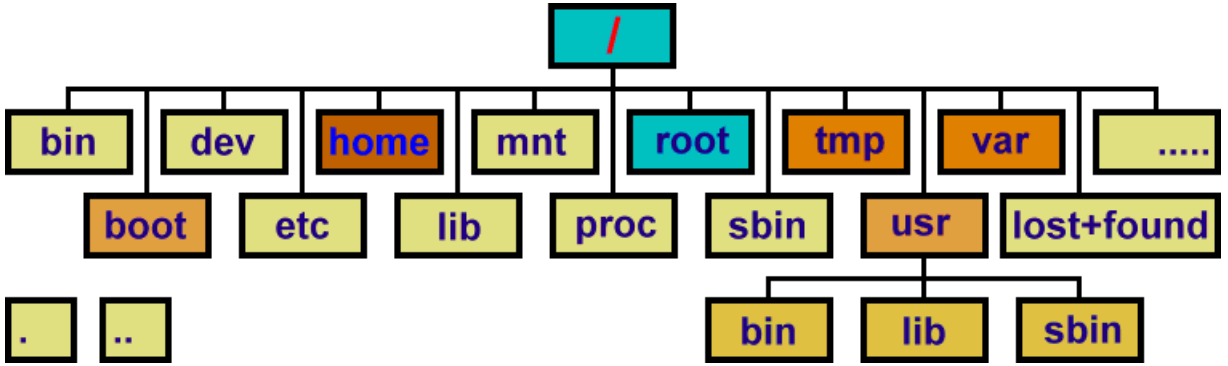
1. 리눅스 커널의 특징 중 하나이며 유닉스에 비해 리눅스는 초창기부터 가상 파일 시스템을 지원했다.

2. 유닉스는 디스크, 터미널, 네트워크 카드등 모든 주변 장치들을 파일로 취급한다.

3. 따라서 다양한 파일들을 관리하기 위해 다양한 파일 시스템이 존재하게 된다.

4. 리눅스에서는 다양한 파일 시스템들을 하나의 파일 시스템 처럼 사용할 수 있도록 가상 파일 시스템이라는 구조를 사용한다.

6.1.3 리눅스 디렉터리 구조



1. 수많은 리눅스 배포판은 FHS 표준에 따르도록 권장

2. 강제 사항은 아니지만 대부분의 리눅스는 이 표준을 준수하고 있음

3. 각각의 디렉터리는 그에 맞는 용도가 있다.

4. 안전하고 편리하게 시스템을 운용하기 위해서는 용도를 잘 알고 맞게 사용해야 한다.

6.1 리눅스 파일 시스템

/ 루트

루트 디렉터리는 시스템의 근간이 되는 가장 중요한 디렉터리이다. 모든 파티션, 디렉터리는 루트 디렉터리 아래에 위치하기 때문에 반드시 존재해야 한다.

/bin

시스템 관리자 혹은 일반 사용자가 실행할 수 있는 수많은 명령어들이 들어있다.

ex) cat, chmod, date, ls, mkdir, rm, touch, vi . . .

/sbin

시스템 관리자가 사용할 수 있는 명령어들이 들어 있다. 시스템을 수정, 복구에 관한 많은 명령어들이 들어 있으며 일반 사용자의 실행 권한도 제한을 해야 하는 등 보안에 신경을 써야 한다.

ex) ifconfig, reboot, shutdown, mount, fsck

/boot

부트로더와 부팅에 관련된 파일들이 있다. 손상되면 시스템이 부팅이 되지 않으므로 특별한 목적이 아니면 건들지 말아야 한다.

/home

유저들의 홈 디렉터리가 하위 디렉터리로 존재하게 된다.

6.1 리눅스 파일 시스템

/dev
디바이스 파일들이 있다. 시스템의 모든 장치가 파일로 표현되어 있는데 udev라는 데몬이 이곳의 장치 파일을 관리한다.
ex) /dev/sda, /dev/hda, /dev/tty1, /dev/pts/0

/etc
시스템 혹은 각종 프로그램들의 환경 설정 파일들이 위치한다. 시스템 관리에서는 이곳의 파일들을 주로 수정하게 된다. 따라서 이 곳의 설정 파일들은 백업을 해 두는 것이 좋다.

ex) /etc/fstab, /etc/gourp, /etc/initab, /etc/passwd, /etc/sysconfig/i18n

/lib
시스템의 프로그램이 실행 할 때 필요한 공유 라이브러리들이 들어있다 따라서 특별한 일이 없으면 변경하거나 삭제하지 않는 것이 좋다.

/mnt
마운트를 위한 임시 디렉터리가 위치한다. CD나 USB같은 이동 디스크를 마운트할 때 사용하게 된다.

/root
root 계정의 홈 디렉터리이다. root 홈 디렉터리는 root만 접근할 수 있다.

/var
log파일등 수시로 업데이트되는 파일들이 위치한다. 또한 시스템 운영에 필요한 파일들도 위치하기 때문에 수정과 삭제에 주의해야 한다.

6.1 리눅스 파일 시스템

/proc

실행중인 프로세스 정보와 CPU, 메모리 등의 시스템 정보가 가상 파일로 저장되어 있다. 대부분 읽기 전용이나 일부 파일 중에는 쓰기가 가능한 파일들이 있는 데 이런 파일들은 커널의 기능을 변경할 수 있다.

실습1

cpu의 정보를 출력해보자

```
[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
cache_alignment         : 64
address sizes   : 36 bits physical, 48 bits virtual
power management:
```

실습2

파티션 정보를 출력해 보자

```
[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:
```

실습3

다음과 같은 정보를 출력 해보자

/proc/meminfo
/proc/uptime
/proc/filesystems
/proc/version
/proc/modules
/proc/loadavg

/usr/bin
응용 프로그램들의 실행파일들이 위치

/usr/sbin
시스템 관리를 위한 명령어들이 있다.

/usr/include
시스템, 네트워크 프로그래밍을 위한 C 헤더파일

/usr/lib
/usr/bin, /usr/sbin 에서 있는 실행 파일들을 위한 라이브러리들이 우치

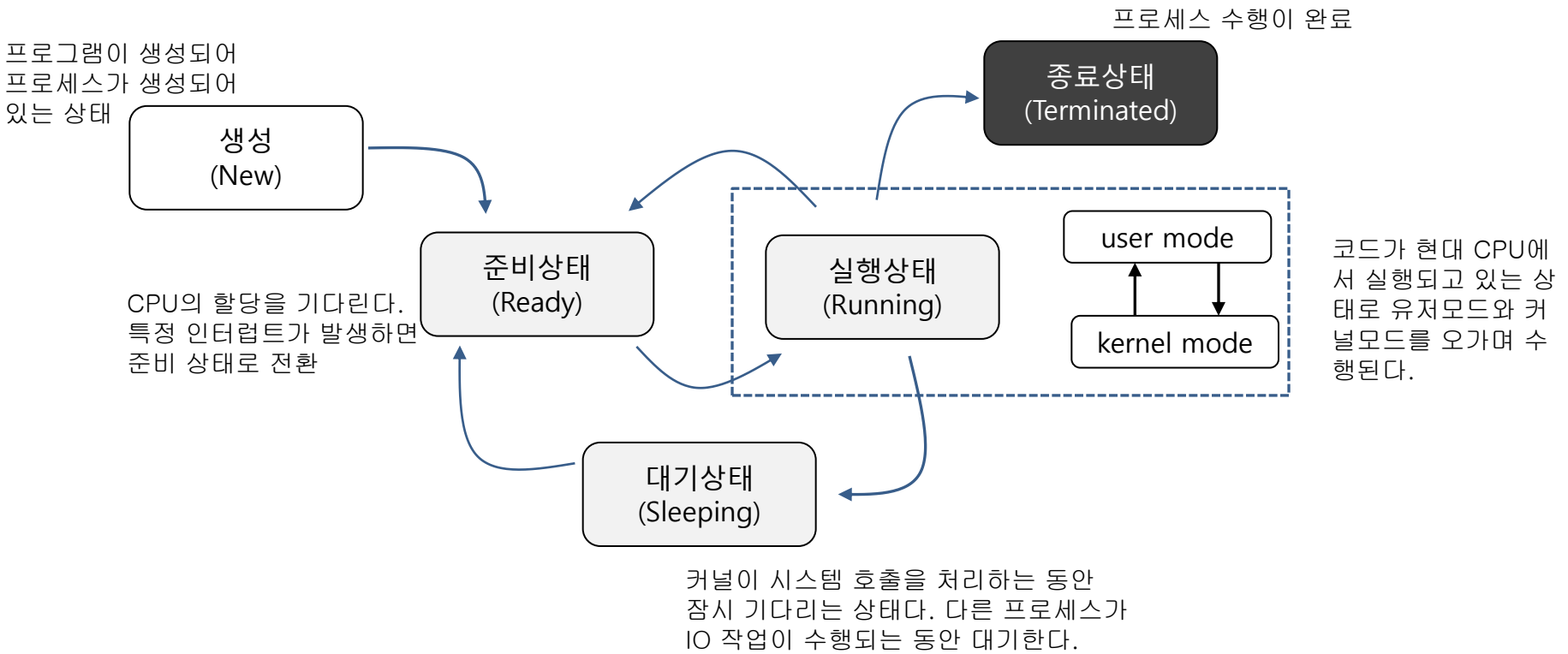
/tmp
임시로 파일을 만들고 삭제하는 공간이다.

/lost+found
부팅시 파일시스템에 문제가 생길 경우 fsck 명령어로 복구할 때 사용하는 디렉터리

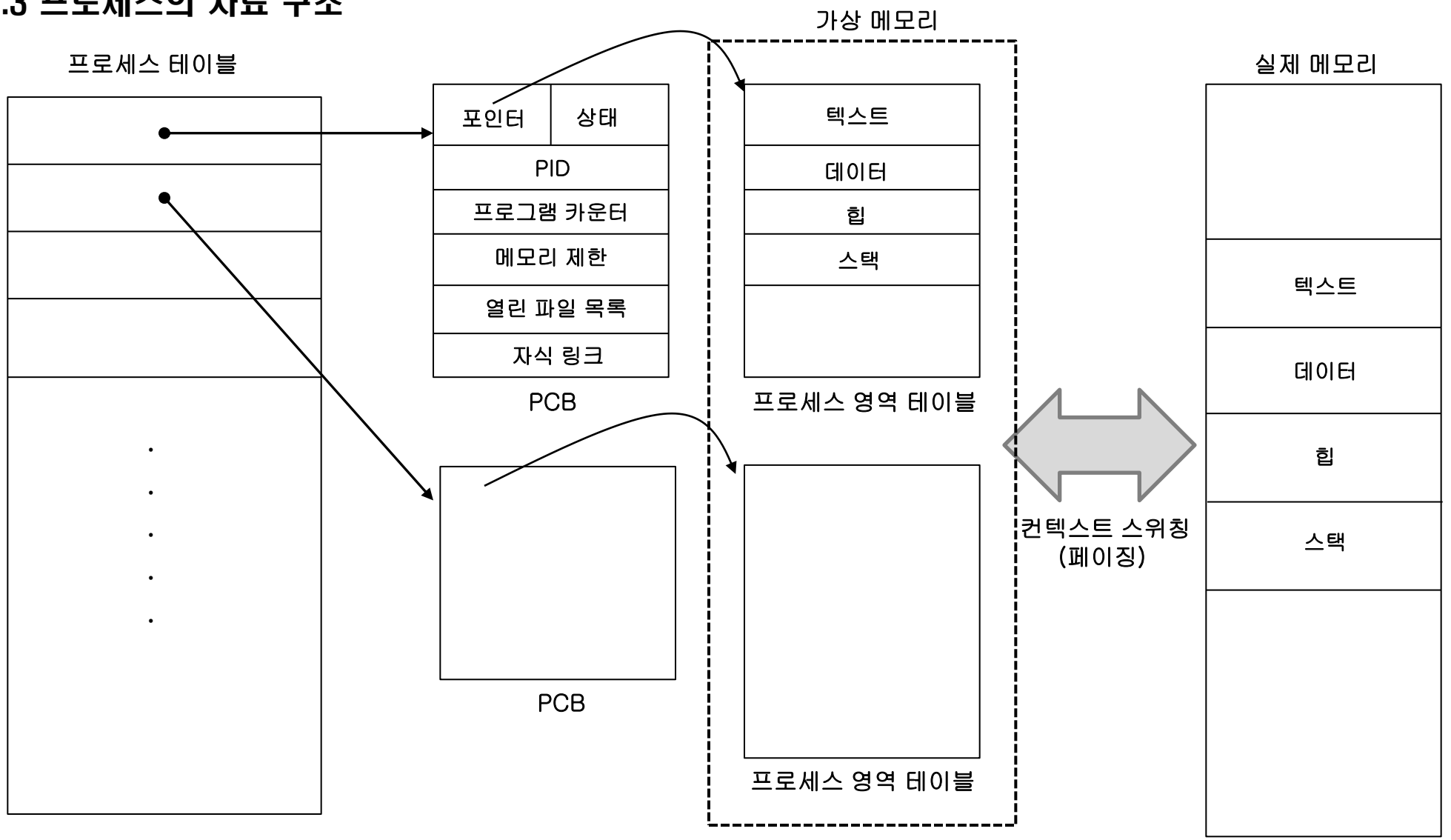
6.2.1 프로세스

- 유닉스는 시분할 시스템으로 여러 개의 프로그램을 동시에 실행한다.(멀티태스킹)
- 컴퓨터 내에서 실행 중인 프로그램을 프로세스(process) 또는 태스크(Task)라고 한다.
- 여러 프로세스가 동시에 실행되는 것을 멀티프로세스라 한다.

6.2.2 프로세스의 상태



6.2.3 프로세스의 자료 구조



6.2 프로세스

1. 리눅스에서는 프로세스에 대한 정보는 task_struct 구조체를 통해 관리된다. 이 구조체는 프로세스의 모든 정보를 보관하는 프로세스 서술자로 아주 많은 구조체 로 이루어져 있다.

2. 이 정보는 ps 명령을 통해 가져올 수 있다.

3. 현재 실행 되는 프로세스틔 정보는 /proc 디렉토리를 통해서 확인할 수 있다.

4. 실행 중인 프로세스에 대한 정보는 /proc 디렉터리 안의 PID로 되어 있는 디렉터리에 존재 하고 있으며 status 파일을 통해 확인할 수 있다.

실습1

```
[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:
                00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000001
Mems_allowed_list:              0
voluntary_ctxt_switches:        55
nonvoluntary_ctxt_switches:     0
[root@lx ~]# ^C
```

6.2.4 프로세스 관리 명령어

PS
현재 실행되고 있는 프로세스의 목록을 보여준다

사용법
ps [옵션]

옵션
-l : 자세한 형태의 정보를 출력한다.
-a : 다른 사용자들의 프로세스도 보여준다.
-u : 프로세스의 사용자 이름과 시작 시가능 출력한다.
-x : 터미널과 연결되지 않은 프로세스도 보여준다.
-e : 환경을 보여준다.
-f : 프로세스의 정보를 한 둘로 자세히 출력한다.
-r : 현재 실행중인 프로세스들을 표시한다.
-j : 작업 중심의 형태로 출력한다.
-c : 커널 task_struct 구조체 형태로 보여준다.

보통 -aux 또는 -ef 옵션을 사용해서 프로세스 상태를 확인한다.

6.2 프로세스

실습1

-ef 옵션을 사용해서 프로세스 상태를 확인해 보자.

```
[root@lx ~]# ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0  2월17 ?      00:00:01 /usr/lib/systemd/systemd --switched-root --system --deseri
root         2     0  0  2월17 ?      00:00:00 [kthreadd]
root         3     2  0  2월17 ?      00:00:00 [ksoftirqd/0]
root         6     2  0  2월17 ?      00:00:00 [kworker/u2:0]
root         7     2  0  2월17 ?      00:00:00 [migration/0]
root         8     2  0  2월17 ?      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

6.2 프로세스

실습1

-aux 옵션을 사용해서 프로세스 상태를 확인해 보자.

```
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

6.2.4 프로세스 관리 명령어

pstree
프로세스 정보를 트리형태로 보여준다.

사용법
pstree [옵션]

옵션
-n : PID 순으로 정렬
-p : 프로세스 명 + PID

실습1

```
[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
```

6.2.4 프로세스 관리 명령어

top

프로세스의 CPU, Memory 사용량등 전반적인 상황을 실시간으로 모니터링 한다.

사용법

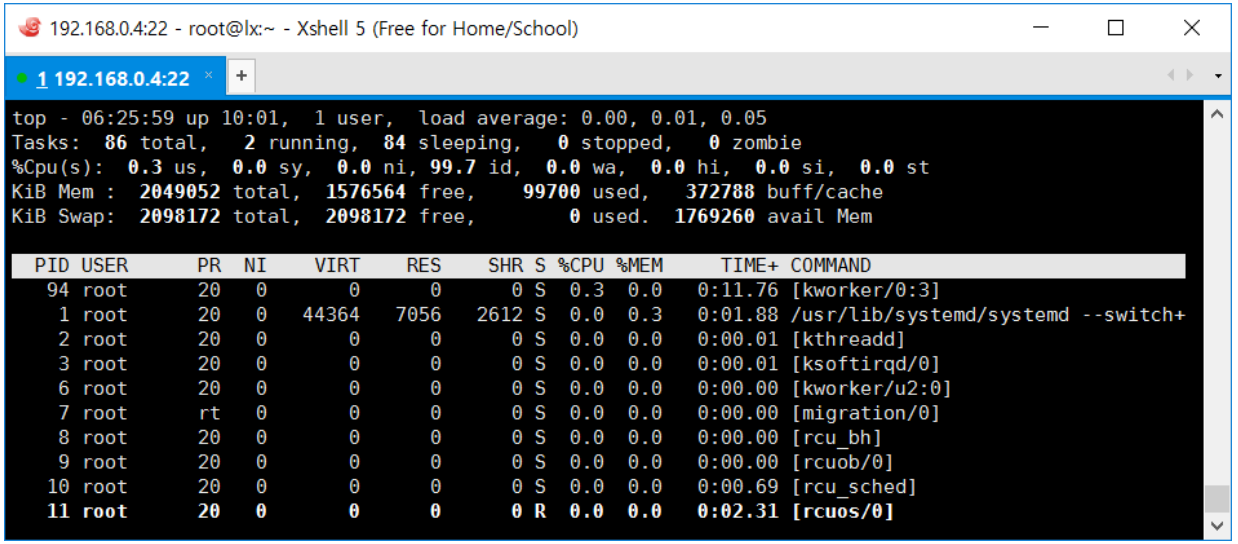
top [옵션]

옵션

-d 시간: 화면 갱신 시간 지정
-c : 명령행 전체를 보여준다
-q : 화면으 계속 갱신한다.

실습1

별 옵션 없이 top를 실행 시켜보자.



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
sleepling	백그라운드에서 잠자고 있는(대기 모드) 프로세스 수
stopped	실행을 일시적으로 중단하고 있는 프로세스 수
zombie	실행을 종료했지만 어떤 이유로 메모리에 남아있는 프로세스 수

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

6.2 프로세스

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

실습2

top명령을 실행 시킨 후, 다음 주요 단축 키를 눌러보자

M: 메모리 사용량 순으로 정렬
P: CPU 사용량 순으로 정렬
T: 실행시간이 긴 순서로 정렬
R: 정렬의 순서 변경

6.2.4 프로세스 관리 명령어

kill

지정한 프로세스에게 시그널을 보낸다.

사용법

top [옵션] PID

옵션

-l : 시그널 목록을 보여준다.
-c : 명령행 전체를 보여준다
-q : 화면으 계속 갱신한다.

실습1

- l 옵션을 사용해서 프로세스가 받는 시그널 목록을 확인해 보자.

```
[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-1451) SIGRTMAX-1352) SIGRTMAX-12
53) SIGRTMAX-1154) SIGRTMAX-1055) 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
```

6.2.4 프로세스 관리 명령어

실습2
-9(kill) 시그널을 보내서 프로세스를 죽여 보자.

```
[root@lx ~]# ps -ef | grep hhtpd
root     14999 14881  0 06:39 pts/0    00:00:00 grep --color=auto hhtpd
[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 ~]#
```

번호	신호	의미
1	SIGHUP	hang-up 을 줄인 말. 애플리케이션에 현재 연결을 끊으라고 알린다. 애플리케이션을 재 초기화 할 때 사용
3	SIGQUIT	quit 애플리케이션에 정상 종료하라고 알림
6	SIGABRT	about 프로그램이 중단된다고 알림. 프로그램은 곧바로 닫힘
9	SIGKILL	강제로 애플리케이션을 종료 함

6.3 메모리 관리

6.3.1 free

시스템의 메모리 정보를 출력한다.

사용법

free [옵션]

옵션

-b : 바이트 단위 표시
-k : Kb 단위로 표시
-m : Mb 단위로 표시
-t : 총합을 표시

실습1

메모리 사용량 출력하기

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