

제3장 리눅스 시스템 이해하기

1. 하드웨어와 구조
2. 리눅스 파일시스템
3. 프로세스 관리
4. /proc 파일시스템

1. 하드웨어와 구조

(1) 시스템 리소스

컴퓨터는 하드웨어(디바이스)들과 그것을 제어하는 소프트웨어(디바이스 드라이버)로 구성이 됩니다. 디바이스는 시스템 리소스를 할당받습니다. 이러한 값들은 어디서, 어떻게 디바이스들이 프로세서 (CPU)와 메모리간 상호작용을 하게 되는지 결정 짓습니다. IRQ, I/O 주소(ioport), DMA 주소 등이 그것에 해당이 됩니다.

(2) IRQ

인터럽트(interrupt)란 컴퓨터의 CPU에게 현재의 작업을 멈추고 요청한 작업을 수행한 후 원래 하던 작업을 계속하는 처리 방식을 뜻합니다. CPU에게 인터럽트 신호를 보내는 것은 키보드, 통신포트, 기억장치 등 각종 주변장치가 되며 CPU 자신일 수도 있습니다.

IRQ란 하드웨어(키보드, 사운드카드 등)로부터 나오는 신호이며, 인터럽트 제어기에 의해 CPU로 보내 집니다. 그러면 CPU는 해당 IRQ 신호의 주변장치가 요구하는 작업을 처리하게 되는 것입니다.

인텔 베이스 아키텍처인 경우 15개의 IRQ를 할당합니다. 현 리눅스 시스템의 IRQ정보를 확인하고자 한다면 /proc/pci 파일을 살펴 봅니다.

(3) PCI

PCI 란 Peripheral-Component Interconnect의 약자로 33MHz의 전송 속도로 32비트 데이터를 처리하는 버스규약입니다. 메인보드에는 CPU와 데이터를 교환할 수 있도록 PCI 컨트롤러가 기본적으로 장착이 되어 있으며, PCI 방식의 디바이스들은 부팅될 때 PCI 컨트롤러에 의해 자동적으로 IRQ를 할당받습니다.

PCI 버스에 어떠한 장치들이 설정되어 있는지 확인하는 명령은 lspci 입니다.

lspci

00:00.0 Host bridge: Intel Corporation 82810-DC100 GMCH [Graphics Memory Controller Hub] (rev 03)

00:01.0 VGA compatible controller: Intel Corporation 82810-DC100 CGC [Chipset Graphics Controller] (rev 03)

00:1e.0 PCI bridge: Intel Corporation 82801AA PCI Bridge (rev 02)

00:1f.0 ISA bridge: Intel Corporation 82801AA ISA Bridge (LPC) (rev 02)

00:1f.1 IDE interface: Intel Corporation 82801AA IDE (rev 02)

00:1f.2 USB Controller: Intel Corporation 82801AA USB (rev 02)

00:1f.5 Multimedia audio controller: Intel Corporation 82801AA AC' 97 Audio (rev 02)

01:05.0 Ethernet controller: Realtek Semiconductor Co., Ltd. RTL-8139 (rev 10)

(4) PCMCIA

Personal Computer Memory Card International Association 의 약자로 노트북과 같은 휴대용 컴퓨터에 사용되는 확장카드의 표준을 마련하기 위한 국제협회를 뜻합니다.

PCMCIA 디바이스들은 PCI 디바이스들과 유사하게 동작을 합니다. PCMCIA 설정과 관련있는 파일은 /etc/pcmcia/config.opts 입니다.

(5) ISA/Plug and Play

ISA 디바이스들은 IRQ를 장치에 붙어 있는 점퍼로 세팅을 하거나 PnP 기능을 이용하여 설정을 합니다.

하지만 ISA/PnP 기능의 네트워크 인터페이스 카드 등 일부 장치들은 리눅스에서 제대로 인식이 안되므로 BIOS에서 PnP기능을 제거하거나 해당 디바이스를 생산한 업체의 홈페이지에서 PnP기능을 제거하는 유틸리티를 먼저 다운받아 PnP기능을 제거해야 리눅스에서 사용이 가능한 경우도 있습니다.

(6) I/O Addresses

CPU와 각각의 디바이스들이 데이터 통신을 하기 위해서는 각각의 장치에 대해 입출력을 지정한 고유한 주소가 하나 혹은 그 이상 필요한 데 이것을 I/O 주소라고 합니다.

PCI 장치들은 자동적으로 설정이 되며, ISA/PnP 장치들은 수동 혹은 자동으로 설정이 됩니다.

/proc/ioports 파일을 살펴 보면 현재 장착된 디바이스들의 I/O 어드레스들을 확인할 수 있습니다.

(7) DMA

Direct Memory Access의 약자로 DMA 컨트롤러 칩을 이용하여 CPU를 이용하지 않고 메모리와 주변장치 간 데이터를 직접 주고 받을 수 있는 기술입니다. 디바이스 모듈을 메모리에 적재할 때 주로 사용이 됩니다. 사운드카드, 네트워크카드, 하드디스크 등이 DMA를 이용하는 디바이스들입니다.

(8) 디바이스와 드라이버

하드웨어 디바이스들은 리눅스에서는 몇몇 운영체제와는 다른 특별한 방식으로 다뤄지고 있습니다. 물론, 디바이스 드라이버에 의해 제어가 되지만, 커널안에 포함되어 있거나 필요할 때 마다 사용될 수 있도록 모듈화 되어 있을 수도 있습니다.

다음은 리눅스 디바이스 드라이버의 특징들입니다.

- 디바이스들은 /dev 디렉토리 내 특별한 파일들로 존재합니다.
- 디바이스 파일들은 block(디스크 등) 타입과 character special(테이프, 시리얼 라인 등) 타입의 파일로 만들어져 있습니다.
- 리눅스에서는 모든 주변장치를 파일로 간주를 합니다.
- 메모리 또한 파일로 여깁니다(/dev/kmem 과 /dev/mem).
- 디바이스들은 메이저 번호와 마이너 번호를 가집니다.

메이저 번호는 디바이스 드라이버를 대표하는 번호입니다. 가령 플로피디스크를 뜻하는 디바이스 파일은 fd0 의 메이저 번호는 2번입니다. 마이너 번호는 플로피디스크의 유형과 세부정보를 뜻합니다.

```
# ls -l fd0*
brw-rw---- 1 root floppy 2, 0 Mar 24 2001 fd0
brw-rw---- 1 root floppy 2, 4 Mar 24 2001 fd0CompaQ
brw-rw---- 1 root floppy 2, 12 Mar 24 2001 fd0D360
brw-rw---- 1 root floppy 2, 16 Mar 24 2001 fd0D720
brw-rw---- 1 root floppy 2, 28 Mar 24 2001 fd0H1440
brw-rw---- 1 root floppy 2, 12 Mar 24 2001 fd0H360
brw-rw---- 1 root floppy 2, 16 Mar 24 2001 fd0H720
brw-rw---- 1 root floppy 2, 4 Mar 24 2001 fd0d360
-- more --
```

- 디바이스 파일들에 대한 자세한 내용은 다음 파일에서 확인합니다.
/usr/src/linux/Documentation/devices.txt
- 디바이스 파일을 만들거나 부주의로 삭제하였을 경우는 다음처럼 /dev/MAKEDEV 명령을 사용합니다.

가령, 플로피디스크 장치명인 fd0 를 삭제하였을 때 다음과 같이 복구를 합니다.

```
# cd /dev
# ./MAKEDEV fd
```

2. 리눅스 파일시스템

파일시스템(filesystem)이란 운영체제가 파티션 또는 디스크에 파일들이 연속되게 하기 위해 사용하는 방법들이고 자료 구조입니다. 즉, 파일들이 디스크상에서 구성되는 방식입니다.

디스크 위에 데이터를 어떻게 써야할 지에 대한 규칙들이라 할 수 있습니다. 고로, 파일시스템이 없다면, 빈 디스크에 정보를 어떻게 저장할 것인지를 알 수가 없습니다. 여기서 디스크라고 말했지만, 엄밀히 말하면 Random Access가 가능한 저장장치를 말하는 것으로 테이프 디바이스 같은 Sequence Access로 저장되는 경우는 해당되지 않습니다.

파일시스템에는 여러 종류가 있으며, 각각의 파일시스템은 나름대로의 특성과 한계를 가지고 있습니다. MS-DOS 같은 경우에는 파일 이름이 8자 이내이고, 3자리의 확장자를 가져야 한다라든가 하는 것 말입니다.

여기서는 리눅스에서 사용하고 있는 파일시스템에 대해 알아보도록 하겠습니다.

파일시스템의 종류에 따라 구체적으로는 다른 구조일 수 있지만 대체적으로 다음과 같은 구조에 의해서 자료를 저장하고 다시 접근할 수 있는 방법을 제공합니다.

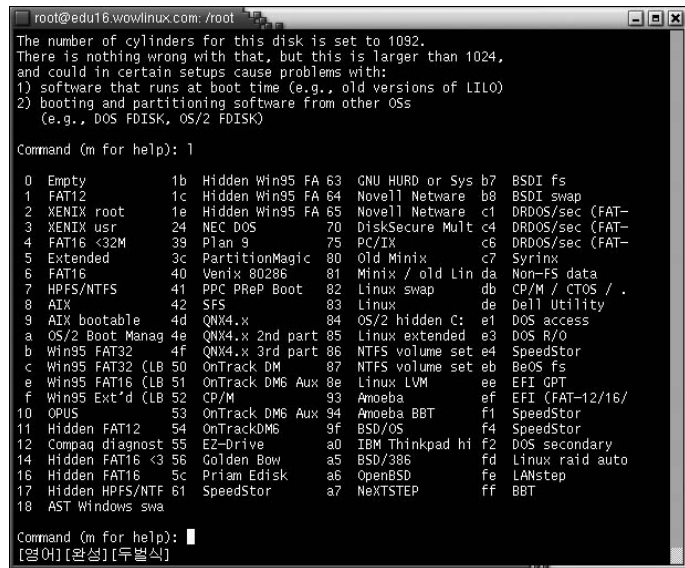
구조	설명
super block	파일시스템의 전체적인 정보를 포함. 파일시스템 Size, 매직넘버, 마운트 횟수, 블록그룹번호, 블록 크기, 첫 번째 inode 등
inode	파일에 대한 모든 정보 (단, 파일 이름은 제외) Data Block의 위치 정보, 타임스탬프, 크기, 소유자, inode의 모드 등
data block	파일에서 데이터를 저장하는 공간
directory block	파일 이름 & inode번호
indirection block	inode에 Data Block의 위치 정보를 저장할 공간이 부족할 때, 이 위치정보를 저장하기 위한 공간을 동적으로 할당하는데 이 공간이 indirection block(간접블록)입니다.

(1) Ext2 파일시스템

리눅스도 사용하는 파일 시스템이 있습니다. 바로 ext2 (Extended2)라는 파일 시스템입니다.

초기의 리눅스는 타넨바움 교수가 교육용으로 제작한 Minix라는 OS의 파일시스템을 사용했지만, 여러 가지 제약 때문에 여러차례의 변화를 거쳐 지금의 ext2를 사용하게 되었습니다.

이 외에도 CD-ROM에서 사용하는 iso9660, 윈도우에서 사용하는 FAT, minix, NFS, sysv, ... 와 같은 여러 종류의 파일 시스템을 지원하며, 이러한 파일시스템 형식으로 작성된 파일이나 디렉토리들을 인식할 수 있습니다(지원하는 파일시스템의 종류는 fdisk 실행 후, l 명령으로 확인할 수 있습니다).



[그림 III-1] 파일시스템의 목록

리눅스 커널이 지원하고 있는 파일시스템 중 리눅스의 루트 파티션으로 사용할 수 있는 것은 몇몇 종류가 있지만, 기본적으로는 ext2가 사용됩니다. 최근에는 캐시(cache)에 저장된 데이터를 실제로 디스크에 기록할 때 시스템이 다운되면 파일시스템이 깨지는 ext2의 약점을 보완한, 저널링 파일시스템이나 TUX2 같은 새로운 개념의 파일 시스템들이 등장하여 보다 더 안정된 파일시스템이 제공될 예정입니다. 비공식 패치에서는 이미 저널링 파일시스템인 ReiserFS가 제공되고 있기도 합니다.

파일시스템은 리눅스를 인스톨하면서 생성되므로 리눅스를 설치 하셨다면, 이미 디스크나 파티션 위에 데이터를 기록할 수 있는 준비를 마친 셈입니다(mkfs 명령으로 만들어 줄 수도 있습니다).

리눅스 파일시스템은 모든 파일들이 /(루트 디렉토리) 아래에 트리구조로 연결되어 있습니다. 모든 디렉토리나 장치들도 모두 이 아래에서 찾아 낼 수 있습니다. 필요한 파일시스템에 접근하려면 마운트라는 명령으로 루트 디렉토리 아래의 한 디렉토리로서 접근 가능한 상태로 만든 후 사용할 수 있습니다.

① ext2fs의 특징

- 255자까지 파일 이름으로 사용할 수 있지만, 필요하다면, 1012자까지도 가능합니다.
- 확장자같은 개념은 필요하지 않습니다.
다만, ".xxx" 형태로 파일 이름을 짓는 경우는 파일 이름으로 그 파일을 분류하거나, 특성을 나타내기 위한 것이 대부분입니다.
- 파일 이름 중에서 "." 으로 시작하는 이름을 가진 파일들은 Hidden 파일로 'ls' 명령에 '-a' 옵션을 붙여야 볼 수 있습니다.
'ls -a' 명령으로 나타나는 list에는 반드시 "."와 ".."이 포함되어 있는데, 전자는 현재 디렉토리를,

후자는 부모 디렉토리를 나타냅니다.

- ext2 파일시스템의 크기는 원래 소스코드에 의하면 2 GB까지로 제한 되어 있지만, 최근 VFS(Virtual File System)의 사용으로, 4 TB까지도 가능합니다. 그래서, 파티션을 여러개로 쪼개지 않고도 큰 사이즈의 디스크 사용이 가능해 졌습니다.

(2) ReiserFS

ReiserFS 는 저널링 파일 시스템의 종류 중 하나로 먼저, 저널링 파일 시스템이란 디스크에 있는 인덱스가 갱신되기 전에 정전을 비롯한 시스템 이상으로 비정상적인 종료하더라도, 인덱스의 관련정보가 로그로 남아 인덱스의 재작성 및 복구를 용이하게 할 수 있는 능력이 있는 파일 시스템을 말합니다.

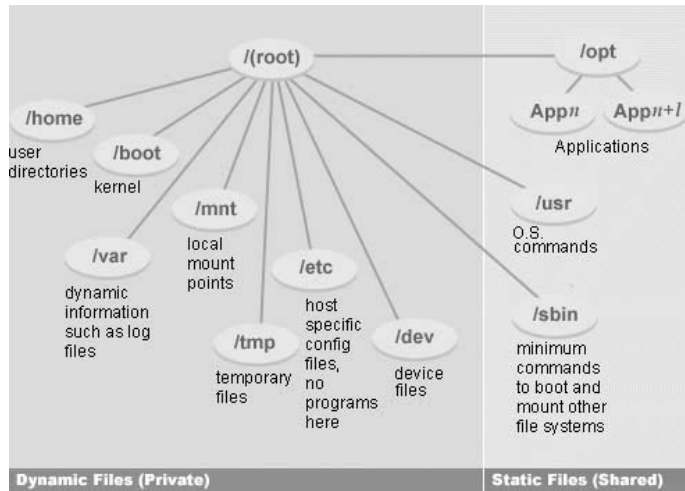
이러한 저널링 파일시스템의 종류중 리눅스 시스템에서 가장 안정으로 작동하는 것이 ReiserFS 파일시스템입니다. ReiserFS 외에도 Ext3, JFS, XFS 등이 있습니다.

ReiserFS가 복구 능력이 있는 좋은 파일시스템이긴 하지만 역으로 생각해 보면 복구능력을 마련하느라 불필요한 로그로 인해 오버헤드와 I/O 작업이 많아집니다. 반면 현재 리눅스 파일시스템의 표준인Ext2의 경우는 성능을 우선시 하여 설계된 파일시스템이라 복구 능력이 떨어지지만 성능은 상당히 뛰어난 파일시스템입니다.

/var 디렉토리와 같이 메일이 스펀링되고, 데이터베이스의 자료가 저장되는 곳은 안전성이 우선시 될 것이고, /usr 디렉토리와 같이 프로그램이 설치되어 읽기 작업만 하는 곳에서는 성능이 우선시 될 것입니다. 여러분의 시스템의 상황을 정확히 파악하고 각 작업환경에 맞게 파티션을 구성하기 바랍니다.

(3) 디렉토리의 구조와 기능

리눅스를 설치하면 상당히 많은 디렉토리가 자동으로 생성됩니다. 이러한 디렉토리는 대부분 유닉스와 유사합니다. 파일 시스템의 구조는 유닉스의 종류(AT&T 계열과 BSD계열)에 따라 약간 차이가 있으며, 리눅스는 주로 AT&T를 중심으로 BSD가 섞인 형태입니다. 리눅스 연합에서는 이러한 배포판의 파일시스템 차이를 표준화하기 위해 FSSTND(File System Standard) 표준안을 마련해 놓고 있습니다. 아래는 리눅스의 파일시스템 구조를 나타낸 것입니다.



[그림 III-2] 디렉토리 구조

파일과 디렉토리는 카테고리별로 조직화되어 있습니다. 위 그림에서 가장 분명하게 알 수 있는 카테고리는 고정(static)과 유동(dynamic)적인 파일들입니다. 또한 다른 카테고리로는 실행가능 여부, 환경 설정, 데이터 파일들 등이 있습니다. 시스템 관리자라는 입장에서 볼 때, 이와 같은 파일 시스템 구조는 관리하기에 적합합니다.

고정적인 파일들과 디렉토리는 또한 네트워크 상의 다른 호스트들로부터 공유가 가능하기 때문에 공유 가능한(shared) 카테고리 분류에도 해당이 됩니다. /usr, /sbin, /opt 같은 디렉토리들이 여기에 해당합니다.

환경 파일들, 디바이스 파일들, 커널 파일들과 같이 특정한 호스트를 위한 파일들과 로그 파일들, 임시 파일들, 사용자의 홈디렉토리에 있는 파일들은 유동적인 파일들입니다. 관련 디렉토리로는 /boot, /home, /tmp, /dev, /etc, /var 등이 있습니다.

운영체제와 응용프로그램들은 서로 분리 시켜서 유지하여야 합니다. 오랜 시간 동안 많은 응용프로그램을 설치하다 보면 이러한 테크닉이 도움이 된다는 것을 알 수 있을 것입니다. 응용프로그램 제공업체들은 설치되는 위치에 대해서 사용자의 편의를 제공하지 않습니다. 부주의로 인해 운영체제 파일들을 응용프로그램 파일들이 덮어 쓰는 것을 방지하는 것은 시스템 관리자로서 매우 중요한 일입니다. 게다가 운영체제와 분리된 영역에 설치된 응용프로그램을 “모듈화”가 가능하므로 추가, 삭제 및 수정에 있어서 운영체제나 다른 응용프로그램에 영향이 미치지 않도록 할 수 있습니다. 유닉스 계열의 운영체제들은 /opt 라는 디렉토리에 응용프로그램들이 설치되지만, 레드햇 리눅스는 이 디렉토리를 구성하지 않고 /usr/local 디렉토리에 응용프로그램 별로 설치가 되거나 RPM으로 설치를 하면 각각의 구성파일들이 필요한 디렉토리로 설치가 됩니다.

실행 파일들도 시스템 환경 파일들과 구분이 되어져야 합니다. 이는 호스트들 사이에 실행 파일들을 공유하기 위함이며, 운영체제를 업데이트 할 때 환경 파일들에게 영향을 주는 것을 방지할 수도 있습니다. 다음은 레드햇 리눅스를 기준으로 한 디렉토리의 구성과 용도에 대한 간략한 설명입니다.

디렉토리		기능
/		루트 디렉토리
/boot	BOOT	부트 이미지 디렉토리
/bin	BINaries	사용자 명령어 디렉토리
/dev	DEVice	장치 파일 디렉토리
/etc	ETCetera	시스템 환경 설정 디렉토리
/home	HOME	사용자 홈 디렉토리
/lib	LIBraries	공유 라이브러리 및 커널 모듈 디렉토리
/lost + found	LOST + FOUND	파일 시스템 복구를 위한 fsck의 링크 디렉토리
/misc	MISCellaneous	아키텍처 독립 자료 디렉토리
/opt	OPeraTion	애드온(Add-on) 소프트웨어 패키지 디렉토리
/proc	PROCeSS	커널과 프로세스를 위한 가상 파일 시스템 디렉토리
/root	ROOT	루트 사용자 홈 디렉토리
/sbin	System BINaries	시스템 명령어 디렉토리
/tmp	TeMPorary	임시 작업 디렉토리
/usr	USeR	공유 파일 시스템 디렉토리
/var	VARiable data	가변 자료 디렉토리

① 디렉토리별 상세 내용

중요한 디렉토리들에 대한 자세한 내용을 살펴 보도록 하겠습니다.

▶ /

루트 디렉토리로 파일 시스템 계층 구조의 시작점입니다. 따라서 빈 상태로 유지하는 것이 바람직합니다.

▶ /boot

환경 설정 파일을 제외한 부팅 과정에서 필요한 모든 구성 요소들이 포함되어 있습니다.

▶ /bin

실행 파일들이 모두 모여 있습니다. 이 디렉토리에는 많은 필수적인 프로그램들이 포함되어 있습니다. “ls /bin”을 해 보면 이 안의 파일들을 볼 수 있으며 cp, ls, mv 같은 몇 개의 명령어들은 알아 볼 수 있을 것입니다. 이것들은 이들 명령어들의 실제 프로그램들입니다. 따라서 “cp”명령을 입력하면, /bin/cp 프로그램이 실행되는 것입니다.

“ls -l”를 사용하면, /bin의 대부분의 파일들에서 “*”가 파일명 끝에 추가되어 있는 것을 볼 수 있습니다. 이것은 이 파일이 실행 가능한 파일임을 표시하는 것입니다.

▶ /dev

/dev 안의 파일들은 디바이스 드라이버들입니다. 이것들은 디스크 드라이버, 모뎀, 메모리 등과 같은 시스템 디바이스나 자원들을 접근하는데 사용됩니다. 예를 들면, 파일들로부터 정보를 읽어 볼 수 있는 것과 같이, /dev/mouse를 access함으로써 마우스로부터 입력되는 정보를 읽어 올 수 있습니다. fd로 시작하는 파일 이름들은 플로피 장치들입니다. fd0는 첫 번째의 플로피 디스크 드라이브이며, fd1은 두 번째입니다. 이 이외의 것들은 보통 플로피 디스크의 특정 형태를 표시합니다. 예를 들면, fd0H1440은 첫 번째 드라이브(A: 드라이브)의 고밀도 3.5인치 디스켓을 말합니다.

/dev/console

시스템의 콘솔이며, 모니터가 시스템에 직접 연결되어 있음을 말합니다.

/dev/ttyS? 와 /dev/cua?

직렬 포트를 액세스합니다. 예를 들면, /dev/ttyS0는 도스 상의 “COM1”을 뜻하며, /dev/cua는 “callout” 장치로써, 이것은 모뎀을 직접 액세스할 때 사용됩니다.

hd로 시작하는 디바이스 이름

하드 디스크를 액세스합니다. /dev/hda는 첫 번째 하드 디스크 전체를 뜻하며, hda1은 /dev/hda의 첫 번째 파티션을 의미합니다.

sd로 시작하는 디바이스 이름

SCSI 하드 디스크 드라이브나 테이프 드라이브 같은 SCSI 장치들을 의미합니다. 만약 SCSI 하드 디스크를 가지고 있다면, /dev/hda를 접근하는 대신에 /dev/sda를 액세스해야 합니다.

lp로 시작하는 디바이스 이름

병렬 포트를 말하며, /dev/lp1은 도스의 “LPT1”과 같습니다.

/dev/null

“black hole”로써 사용되는 것으로서 어떠한 데이터를 이 장치에 보내면 모두 없어지게 됩니다. 예를 들면, 화면에 아무 것도 출력되지 않기를 바랄 경우, /dev/null로 출력을 보내면 됩니다.

/dev/tty로 시작하는 디바이스 이름

시스템에 있는 “가상 콘솔 : Virtual Console(VC)”입니다. /dev/tty1은 첫 번째 VC이며, /dev/tty2는 두 번째입니다. 가상 콘솔은 한 화면이 여러 개의 가상 터미널을 갖는 것입니다. 각각의 터미널은 [alt]+[F1], [alt]+[F2] 등을 누름으로서 전환할 수 있으며, 같은 사용자나 다른 사용자로 login할 수 있습니다.

/dev/pty로 시작하는 디바이스 이름

이것들은 원격 login 세션에서 사용되는 “pseudo-terminal”들입니다. 예를 들어, 사용 중인 컴퓨터가 네트워크에 연결되어 있고, telnet으로 login하려고 할 때, /dev/pty 디바이스를 사용합니다.

▶ /etc

시스템의 부팅, 셧다운 시에 필요한 파일들과 시스템의 전반에 걸친 설정 파일들 및 초기 스크립트 파일들이 있습니다. 시스템의 전반에 걸친 설정 파일들 및 초기 스크립트 파일들이 있습니다. 시스템에 어떠한 문제가 발생한다거나, 시스템 전체 환경에 관한 설정을 바꾸기 위해서는 이들 디렉토리 내에 포함되어 있는 파일들에 대해서 잘 알아야 합니다.

/etc/rc.d/rc

/bin/sh shell이 시스템이 부트되면, 자동적으로 실행되는 스크립트입니다. 이것은 update, crond, inetd 같은 프로그램을 백그라운드로 실행시키며, 파일 시스템 마운팅, 스왑 영역 활성화, 그리고 이런 유사한 다른 작업들을 합니다. /etc/rc.d/rc.local과 /etc/rc.d/rc.# 파일이 포함되기도 합니다.

/etc/passwd

사용자에 대한 정보를 포함하고 있는 문서 파일입니다.

/etc/fdprm

플로피 디스크 파라미터 표입니다.

/etc/fstab

이 파일은 /etc/rc.d/rc 파일 안의 mount -a 명령에 의해 마운팅되는 파일 시스템과 스왑 영역의 목록입니다.

/etc/getty (혹은 /sbin/getty)

이 프로그램은 터미널로 누군가가 login하기를 기다립니다. 명령어 init에 의해 자동적으로 실행되며, login 가능한 터미널 라인이나 가상 콘솔 당 한번씩 실행됩니다. 또한, 사용자의 패스워드를 기다리며, login을 실행합니다.

/etc/gettydefs 혹은 /etc/gettytab

getty가 터미널 라인의 속도, 패리티 검사 등을 어떻게 사용할 것인가를 설정합니다.

/etc/group

/etc/passwd와 유사하며, 사용자 대신에 그룹을 설정합니다.

/etc/init (혹은 /sbin/init)

이 프로그램은 부팅 시에 커널에 의해 첫 번째 프로세스로 실행됩니다. init가 실행된 후에 커널 부팅이 완료됩니다. init는 /etc/rc.d/rc와 getty 등을 실행합니다.

/etc/inittab

init가 시작할 때의 getty의 목록 파일입니다.

/etc/issue

로그인 프롬프트 이전에 출력되는 getty 출력 문서 파일입니다. /etc/issue.net 파일은 remote 로그인 시 프롬프트에 출력됩니다.

/etc/mtab

이 파일은 마운팅된 파일 시스템을 포함하고 있습니다. /etc/rc/rc.d와 mount나 unmount 명령에 의한 셋업이며, 마운팅된 파일 시스템의 목록이 필요할 때 사용됩니다.

/etc/shadow

시스템의 shadow패스워드를 포함하는 파일입니다.

/etc/login.defs

login 명령에 사용되는 설정 파일입니다.

/etc/printcap

/etc/termcap과 유사하며, 프린터를 사용할 때(lpr) 쓰입니다.

/etc/profile

Bourne shell(/bin/sh 혹은 bash)에 의해 로그인 할 때 실행되는 파일입니다.

/etc/securetty

터미널 보안을 위해 사용하는 것으로 root는 이 파일에 열거된 터미널로 로그인 할 수 있습니다. 보통 가상 콘솔들이 열거되어 있으며, 모뎀이나 네트워크로 시스템에 접근하여 슈퍼 유저의 권한을 얻는 것을 막을 수 있습니다.

/etc/shells

shell의 목록으로 chsh 명령으로 사용자 로그인 shell을 바꿀 때, 이 파일의 목록에 있는 shell만 바꿀 수 있도록 합니다.

/etc/termcap

터미널의 기능 데이터베이스입니다. 이것은 문서 파일로서 ESCAPE 문자들로서 터미널을 제어할 때 사용됩니다.

/etc/update (혹은 /sbin/update)

/etc/rc.d/rc 에 의해 백그라운드로 실행되는 프로그램의 하나로 매 30초 마다 버퍼 캐시에 있는 하드 디스크로 쓰여지지 않은 데이터를 저장합니다. 이러한 것은 전원의 단절이나, 커널의 이상 등을 대비하여 매 30초 마다 데이터를 저장함으로써 데이터 손실의 위험부담을 줄이는 것입니다.

/etc/utmp (혹은 /var/run/utmp)

각각의 터미널에 로그인한 사용자나, 로그인에 관한 정보가 기록되어 있는 이진 파일입니다. 사용자가 로그인하면 login은 누가 로그인을 하고 언제 로그아웃을 하였는가에 대한 정보를 기록합니다.

/etc/wtmp (혹은 /var/log/wtmp)

/etc/utmp와 유사하며, 단지 존재하는 정보를 덮어 쓰지 않고 계속 추가합니다. /etc/ftpusers, /etc/rpc, /etc/exports 네트워크에 관한 파일들입니다.

▶ /home

사용자의 홈 디렉토리로써 login 하였을 경우, 처음으로 위치하게 되는 디렉토리입니다. 예를 들어, /home/foo는 사용자 "foo"의 홈 디렉토리입니다. 시스템이 새로 설치되면, 이 디렉토리 안에 아무 것도 포함되어 있지 않습니다.

▶ /lib

부팅과 시스템 운영에 필요한 공유 라이브러리(Shared Library)와 커널 모듈(Kernel Module)이 위치합니다. 공유 라이브러리란 windows 95/98의 DLL(Dynamic Link Library)과 같이 여러 가지 프로그램들에 의해서 사용되는 기능을 별도의 프로그램으로 분리시켜 놓은 것입니다. 커널 모듈도 공유 라이브러리와 같이 커널 안에 자체적으로 포함되지 않고 독립적인 형태로 분리되어 있으면서 부팅 시에 커널에 동적으로 연결되어서 전체적인 커널을 유기적으로 구성하게 되는 별도의 파일(Object File)들입니다.

▶ /lost+found

파일 시스템의 이상 유무를 진단하고 복구하는 프로그램인 fsck(File System Check)에 의해서 사용되는 디렉토리입니다. 손상된 파일이나 디렉토리를 /lost+found디렉토리로 연결한 뒤에 오류를 수정하게 되며, 평상시에는 null 파일 링크에 의해서 비어있는 상태로 존재합니다. 리눅스의 파일 시스템인 ext2에 의한 fsck.ext2(File System Check.extended2) 프로그램도 이 디렉토리를 사용합니다.

▶ /misc

시스템 아키텍처와 무관한 프로그램들과 자료들이 위치합니다. 레드햇 리눅스는 이 디렉토리를 구성하지 않습니다.

▶ /mnt

루트 파일 시스템에 연결된 파일 시스템들의 마운트 디렉토리입니다. 마운트하지 않은 상태에서는 빈 디렉토리로 존재하지만 마운트시키게 되면 해당 파일 시스템의 내용이 그대로 포함됩니다.

▶ /opt

Add-On 소프트웨어 패키지가 설치됩니다. 레드햇 리눅스는 이 디렉토리를 구성하지 않습니다.

▶ /proc

가상 파일 시스템입니다. 이 디렉토리의 내용들은 시스템에서 운영되고 있는 다양한 프로세서들에 관한 내용과 프로그램에 대한 정보를 포함하고 있습니다. 이 디렉토리에서 볼 수 있는 것은 실제 드라이브에 저장되어 있는 내용이 아니며, 메모리 상에 저장되어 있는 것입니다.

▶ /root

시스템 관리자인 root의 홈 디렉토리입니다.

▶ /sbin

시스템 관리를 위한 전반적인 실행 유틸리티를 담고 있습니다. 이 디렉토리에 있는 명령어들은 일반 사용자는 실행할 수 없습니다.

▶ /tmp

프로세스 진행 중 발생하는 임시 파일들이 저장되는 작업 디렉토리입니다. 따라서 이 디렉토리의 파일들은 수시로 생성되고 삭제되므로 중요한 자료일 경우, 이 디렉토리에 보관해서는 안됩니다. /tmp 디렉토리는 모든 사용자에 대해서 읽기와 쓰기 작업이 허용되며, 스틱키 비트(sticky bit)라는 특별한 설정에 의해서 파일의 소유자만이 자신의 소유로 되어 있는 파일을 지울 수 있도록 되어 있습니다.

▶ /usr

루트 디렉토리와 함께 중요한 시스템 디렉토리 계층을 구성합니다. /usr 디렉토리에는 공유 가능한 대부분의 프로그램들이 설치되며 네트워크를 이용해서 여러 개의 시스템을 연결할 경우, 이 디렉토리를 공유해서 설치된 프로그램들을 활용할 수 있게 됩니다. 따라서 /usr 디렉토리는 읽기 전용으로 마운트되어야 하며, 가변 자료들은 /var 디렉토리로 심볼릭 링크시켜서 사용하게 됩니다. /usr 디렉토리는 다른 시스템과 연결될 경우, 시스템 운영과 연관되기 때문에 부적으로 전체 루트 디렉토리와 유사한 구조를 갖게 됩니다.

/usr/X11R6

X윈도우 시스템을 설치하였다면 이 디렉토리에 설치가 됩니다. X윈도우 시스템은 방대하며, 많은 그래픽 유틸리티와 프로그램들이 그래픽 윈도우로 출력되는 강력한 그래픽 사용자 환경입니다. 이 디렉토리에는 X윈도우 실행 파일, 사양 파일, 자원 파일들을 포함하고 있습니다. X11R6은 X의 버전과 릴리즈 번호를 뜻합니다.

/usr/bin

시스템이 소유하고 있는 소프트웨어를 담기 위한 warehouse입니다. /bin과 같은 곳에는 없는 유용한 실행 파일들을 가지고 있습니다.

/usr/etc

유틸리티와 파일들이 있습니다. 일반적으로 /usr/etc 에 있는 파일들은 /etc 에 있는 것만큼 반드시 필요로 한 것들은 아닙니다.

/usr/include

C 컴파일러를 위한 include 파일을 포함합니다. 이 파일은 데이터 구조 이름과 서브루틴, 상수 같은 C로 작성된 프로그램에서 사용되는 내용을 담고 있습니다. /usr/include/sys 에 있는 파일들은 리눅스 시스템 레벨의 프로그래밍을 할 때 사용됩니다. 만약, C프로그래밍 언어에 익숙하다면, 여기에 printf() 함수가 선언되어 있는 stdio.h 같은 헤더 파일을 찾을 수 있을 것입니다.

/usr/lib

/lib에서 찾을 수 있는 "stub"와 "static"와 같은 라이브러리를 포함하고 있습니다. 프로그램을 컴파일할 때, 프로그램은 /usr/lib에 있는 파일들과 link되며 이 라이브러리 안에 실행 코드가 필요로 할 때, /lib를 찾습니다. 또한, 많은 프로그램들이 /usr/lib 안에 사양 파일을 저장합니다.

/usr/local

/usr 에 포함된 것과 매우 유사하고, 시스템에 반드시 필요로 하는 것은 아니지만, 매우 유용한 것들을 포함하는 데 사용됩니다. /usr/local 에 있는 프로그램들은 시스템의 특성의 결정 짓는 소프트웨어들이 있을 수 있습니다.

/usr/man

이 디렉토리는 실제적인 man page를 포함하고 있습니다.

/usr/src

시스템에 있는 다양한 프로그램의 컴파일 되지 않은 소스 코드를 포함하고 있습니다. 여기서 가장 중요한 것은 /usr/src/linux 이며, 이것은 리눅스 커널의 소스 코드를 포함하고 있습니다.

/usr/dict
사전 파일 디렉토리입니다.

/usr/doc
프로그램의 문서와 관련한 파일들이 저장되는 곳입니다.

/usr/games
게임 프로그램들이 위치하고 있습니다.

/usr/info
GNU info파일을 위한 디렉토리입니다.

/usr/sbin
시스템 명령어들이 위치합니다.

/usr/share
아키텍처 독립적인 자료들의 디렉토리입니다.

▶ /var
내용이 수시로 변경될 수 있는 변수를 담고 있는 파일들이 위치합니다. 예를 들면, 부팅 중의 시스템 확인 과정은 부팅 때마다 달라질 수 있으므로 부팅 과정을 기록하는 파일은 이 디렉토리에 위치하게 됩니다. /tmp 디렉토리가 파일 자체에 대한 임시 디렉토리인데 반해서 /var 디렉토리는 변경될 수 있는 자료를 포함하고 있는 파일들을 위한 디렉토리입니다.

/var/adm
시스템 관리적인 문제의 파일, 로그 파일, 커널 크래쉬 덤프를 기록한 파일들을 포함합니다.

/var/spool
스풀 파일들이 위치합니다.

(4) 마운트와 언마운트

리눅스에서 파일시스템에 접근을 하기 위해서는 마운트가 되어야 합니다. 명령어를 통해서 마운트가 가능하며, /etc/fstab 파일에 있는 목록을 포함하고 있습니다.
먼저 마운트 명령은 다음과 같이 사용합니다.

mount -t 파일유형 -o 옵션 /dev/device /mnt/device_directory

CD-ROM 드라이브나 플로피 디스크 드라이브, NFS로 공유된 원격 디렉토리를 마운트하는 명령의 예입니다.

mount /dev/fd0 /mnt/floppy
mount -t iso9660 /dev/cdrom /mnt/cdrom
mount -t nfs 10.0.0.1:/usr/local /usr/local

-t 옵션 후 파일시스템의 유형을 적어 주는 것은 생략을 하여도 무방합니다.

프롬프트 상에서 mount명령만을 사용하면 현재 마운트되어 있는 디바이스 장치와 마운트되어 있는 위치를 확인할 수 있습니다. 이 내용은 /etc/mtab 파일에도 있습니다.

mount
/dev/hda1 on / type ext2 (rw)
none on /proc type proc (rw)
none on /dev/pts type devpts (rw,gid=5,mode=620)
automount(pid1002) on /misc type autofs (rw,fd=5,pgrp=1002,minproto=2,maxproto=3)

새로운 파일 시스템(/dev/hda3)을 마운트 하기 위해서, mount point가 될 디렉토리(/home2)를 생성하고 mount 시킵니다.

mkdir /home2
mount /dev/hda3 /home2
cd /home2
ls -a
./ ../ lost+found/

새로 생성한 파일 시스템을 부팅시 항상 mount 되도록 하려면, /etc/fstab 파일에 포함시킵니다.

LABEL=/	/	ext2	defaults	1	1
/dev/fd0	/mnt/floppy	auto	noauto,owner	0	0
none	/proc	proc	defaults	0	0
none	/dev/pts	devpts	gid=5,mode=620	0	0
/dev/hda5	swap	swap	defaults	0	0
/dev/cdrom	/mnt/cdrom	iso9660	noauto,owner,kudzu,ro	0	0
/dev/hda3	/home2	ext2	defaults	0	0

/etc/fstab 파일의 형식은 디바이스명, 마운트포인트, 파일시스템 유형, 옵션, Dump 유무, fsck 시 체크 유무입니다.

umount 명령은 mount된 상태를 해제합니다.
현재 파일시스템이 사용 중이라면, mount가 해제되지 않습니다. 그 디렉토리에서 빠져나온 후 해제합니다.
cd /
umount /home2

(5) 사용자별 디스크 할당 Quota

quota는 사용자가 그들에게 부여된 디스크의 제한 용량 이상으로 디스크를 사용하지 못하게 하는 것으로, 사용자가 가질 수 있는 inode의 수를 제한하거나 사용자에게 할당된 디스크 블록의 수를 제한함으로써 설정을 할 수 있습니다.

① 커널 설정

리눅스를 설치하면 quota지원이 됩니다.
만약, 지원하지 않는다면 커널을 재설정해야 합니다.
/usr/src/linux 디렉토리로 이동을 한 후 make menuconfig 라는 명령으로 커널 컴파일 설정작업을 합니다.
File systems 메뉴를 선택하면 Quota support 라는 항목이 있습니다. 이 항목을 체크한 후 커널 컴파일을 다시 합니다. 커널 컴파일은 8장 리눅스 활용하기를 참조합니다.

② /etc/fstab 파일의 수정

가령 /home 디렉토리에 설정을 원한다면, /etc/fstab 파일을 편집하여 옵션의 defaults 부분에 "usrquota"부분을 추가합니다.
/dev/hda1 / ext2 defaults 1 1
/dev/hda2 /usr ext2 defaults,usrquota 1 1

③ quota.user, quota.group 파일의 생성

quota 기록 파일인 quota.user, quota.group 파일을 해당 파일시스템의 root 수준에서 만듭니다.
root로 로그인한 후 quota를 설정하려는 파티션의 최상위 디렉토리에 만들면 됩니다.
touch /usr/quota.user
chmod 600 /usr/quota.user
이제 시스템을 리부팅합니다.

④ 사용자에게 할당량을 부여하기

이것은 edquota명령으로 설정합니다. 예를들면 "edquota bob"이란 명령을 사용함으로써 vi에디터(또는 \$EDITOR 환경변수에 설정한 에디터)로 quota가 설치된 각 파티션의 "bob"이라는 사용자의 할당량을 수정할 수 있습니다.

Quotas for user bob:
/dev/hda2: blocks in use: 2594, limits (soft = 5120, hard = 6400)
 inodes in use: 356, limits (soft = 1000, hard = 1500)

현재의 작업은 /tmp/EdP.aa01023 와 같은 임시파일에 만들어진 후 /usr/quota.user 파일에 저장됩니다.
"blocks in use"는 사용자가 사용 중인 한 파티션의 총 블록(kilobyte단위)을 의미합니다. "inodes in use"는 사용자가 사용 중인 전체 파일의 갯수입니다. edquota에서 당신은 Soft Limit, Hard Limit, 유예기간(Grace Period)이라는 단어를 알아야 합니다.

- Soft Limit
일반적으로 soft limit는 한 사용자가 사용할 수 있는 최대 용량을 가리킵니다. 그러나 유예기간(grace period) 내에 있어서는 사용자는 사용용량 초과에 대해서 경고를 받게되는 경계선처럼 동작합니다.
- Hard Limit
hard limit는 유예기간(grace period)이 설정되어 있을 때에만 동작합니다. 이것은 디스크의 최대 사용용량을 의미합니다. 즉, 사용자는 hard limit 용량 이상을 사용할 수 없습니다.
- Grace Period
유예기간은 사용자의 사용 용량이 soft limit용량을 넘은 후부터 적용되는 시간 제한입니다. 시간은 sec(onds), min(utes), hour(s), day(s), week(s), month(s) 단위로 사용할 수 있습니다. "edquota -t" 명령을 실행하면 유예 기간을 보거나 수정할 수 있습니다.

Time units may be: days, hours, minutes, or seconds Grace period before enforcing soft limits for users: /dev/hda2: block grace period: 0 days, file grace period: 0 days
"0 days"부분을 적당한 기간으로 바꿔 줍니다. 개인적으로는 7일("7 days" or "1 week")로 설정해 주었습니다.

5 기타 Quota 명령어

- Quotacheck

Quotacheck는 파일시스템의 디스크 사용상태를 검색하거나 quota기록화일인 “quota.user” 파일을 최근의 상태로 갱신하기 위해 사용합니다. quotacheck를 정기적으로(1주) 시스템을 부팅할때나 cronjob을 통해서 실행할 것을 권장합니다.

- Repquota

Repquota는 파일시스템의 quota 를 간략화해서 보여 줍니다. repquota가 출력하는 예제입니다.

repquota -a

		Block limits				File limits			
User		used	soft	hard	grace	used	soft	hard	grace
root	—	175419	0	0		1467	9	0	
bin	—	18000	0	0		735	0	0	
uucp	—	729	0	0		23	0	0	
man	—	57	0	0		10	0	0	
user1	—	13046	15360	19200		806	1500	2250	
user2	—	2838	5120	6400		377	1000	1500	

- Quotaon 와 Quotaoff

Quotaon은 quota를 가동시킬 때 사용합니다. 반대로 quotaoff는 중단시킬 때 사용합니다. 실제로 이 두화일은 비슷합니다. 이것들은 시스템이 부팅되거나 셧다운될 때 수행됩니다.

3. 프로세스 관리

프로세스란 간단히 말해 리눅스에서 작동하고 있는 프로그램을 말합니다.

프로세스는 컴퓨터의 CPU, 메모리, 디스크 등을 활용하여 작업을 수행하므로, 많은 프로세스를 수행시키려면 그만큼 많은 자원이 필요하게 됩니다. 그러므로, 프로세스를 잘 관리해야 시스템이 최적의 성능을 발휘할 수 있는 것입니다.

이 섹션에서는 프로세스의 개념에 대해 간단히 설명하고 어떻게 프로세스를 모니터링하고 관리하는지, 어떤 사용자가 시스템에 접근하여 시스템의 자원을 활용하는지, 어떤식의 접근을 하고 있는지, 그리고 사용자가 접근하는 디스크 스페이스가 얼마나 되는지 등등의 프로세스를 관리하는데 필요한 내용들을 알아보려고 합니다.

(1) 프로세스란

1 프로세스의 생성

프로세스는 수행시키려는 부모 프로세스를 가집니다. 이 부모 프로세스가 자신과 동인한 프로세스를 복제(fork)한 후 수행시키려는 내용으로 덮어 씌워 새로운 프로세스가 생성되고, 그 프로세스는 자신이 수행할 임무를 완수하게 되는 것입니다. 모든 임무를 완수한 후에는 종료됩니다. 이런 프로세스는 반대로 자식프로세스라고 하는데, 이들은 부모의 환경을 물려 받아 사용합니다.

각각의 프로세스는 프로세스 ID(PID)를 할당 받게 됩니다.

가장 먼저 실행 되는 프로세스인 init의 PID는 1입니다.

2 사용자 프로세스(대화형 프로세스)

시스템 사용자들은 셸을 통해 프로세스를 수행 시킬 수 있습니다.

위에서 언급한 것처럼, 셸에서 사용자가 프로그램 이름을 입력하면, 셸은 그 프로그램 이름이 유효한지 확인한 후, 자신을 복제한 새로운 프로세스를 만들고 이 프로세스를 사용자가 지정했던 프로그램으로 바꾸어 이 프로그램에 대한 프로세스를 수행하도록 합니다. 이 프로세스들은 Background와 Foreground 중의 하나로 수행되며, 실행되는 상태를 변경할 수 있습니다.

3 데몬 (Daemon)

사용자가 생성하는 프로세스들 외에도, 시스템이 시작되는 동안 리눅스 시스템이 실행시키는 프로그램들이 있습니다. 이런 프로세스들은 데몬(Daemon)이라고 부릅니다.

데몬은 백그라운드로 항상 실행되고 있으며, 지시가 있을 경우에만 종료됩니다. 보통 일반 사용자들은 이 프로세스들을 볼 수 있는 권한이 없습니다.

시스템이 시작 될 때 가장 먼저 실행 되는 프로세스인 init는 서버 시스템에 대해 보통 inetd 데몬을 수행시키는데, 이 프로세스는 네트워크 관련 서비스 요청이 들어 오는지를 관찰하고 있다가 요청이 들어 오면 이들을 위한 프로세스를 실행 시키는 역할을 합니다. 이런 inetd와 같은 프로세스들이 데몬입니다.(Red Hat 7.0부터는 inetd 대신 xinetd라는 데몬을 사용합니다.)

또, sendmail의 경우, 시스템이 전자우편을 위한 설정이 되어 있다면, 메일이 배달 되도록 하기 위해 백그라운드로 계속 실행되고 있습니다. 이런 sendmail 프로세스도 데몬인 것입니다.

프로세스 중 가장 먼저 시작되는 것은 init입니다. 그리고, 이 프로세스에 의해 다른 모든 프로세스들이 시작됩니다. 시작되고 종료되는 데몬들은 runlevel에 따라 달라집니다.

(2) 프로세스 관리

사용자 프로세스들과 데몬들을 모니터링하고, 다시 시작, 혹은 종료 시킬 수 있는 몇가지 방법에 관하여 알아 보도록 하겠습니다.

① ps 명령

현재 실행되고 있는 프로세스들을 리스트로 보여 줍니다. 이 명령은 매우 강력한 명령으로 필요에 따라 다양한 옵션으로 실행시킬 수 있습니다.

옵션	설명
a	전체 사용자의 모든 프로세스 출력
e	명령문 실행 후 프로세스 환경 변수 출력
l	자세히 출력
u	사용자 이름과 프로세스 시작 시간을 출력
w	출력 결과가 한줄이 넘어도 모두 출력
x	제어하는 터미널이 없는 프로세스 출력

다음은 프로세스의 상태를 조회하기 위해 가장 흔하게 사용되는 옵션을 사용한 예입니다. 모든 사용자의 프로세스와 제어하는 터미널이 없는 프로세스들의 사용자 이름과 시작시간을 포함하여 리스트 하라는 명령입니다. TTY가 ?인 것이 제어하는 터미널이 없는 것으로 시스템에서 자체에서 시작된 프로세스들 입니다.

# ps -aux more										
USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.5	0.0	1368	52	?	S	05:18	0:04	init [5]
root	2	0.0	0.0	0	0	?	SW	05:18	0:00	[keventd]
root	3	0.0	0.0	0	0	?	SW	05:18	0:00	[kapm-idled]
root	4	0.0	0.0	0	0	?	SW	05:18	0:00	[kswapd]
root	5	0.0	0.0	0	0	?	SW	05:18	0:00	[kreclaimd]
root	6	0.0	0.0	0	0	?	SW	05:18	0:00	[bdfush]
root	7	0.0	0.0	0	0	?	SW	05:18	0:00	[kupdated]
root	8	0.0	0.0	0	0	?	SW<	05:18	0:00	[mdrecoveryd]
root	496	0.0	0.0	0	0	?	SW	05:18	0:00	[eth0]
root	585	0.0	0.1	1428	168	?	S	05:18	0:00	syslogd -m 0
root	590	0.0	0.0	1924	0	?	SW	05:18	0:00	klogd -2
rpc	623	0.0	0.0	1512	0	?	SW	05:18	0:00	portmap
rpcuser	657	0.0	0.0	1560	0	?	SW	05:18	0:00	rpc.statd
root	833	0.0	0.0	1352	0	?	SW	05:18	0:00	/usr/sbin/apmd -p
root	901	0.0	0.0	1480	0	?	SW	05:18	0:00	/usr/sbin/automou
daemon	935	0.0	0.0	1400	0	?	SW	05:18	0:00	/usr/sbin/atd
root	997	0.0	0.0	2588	0	?	SW	05:18	0:00	/usr/sbin/sshd
root	1055	0.0	0.3	2252	424	?	S	05:18	0:00	xinetd -stayalive
root	1173	0.0	0.0	5348	0	?	SW	05:18	0:00	sendmail: accepti
root	1205	0.0	0.0	1396	52	?	S	05:18	0:00	gpm -t ps/2 -m /d
root	1240	0.1	0.0	13656	56	?	S	05:18	0:00	/usr/sbin/httpd -
apache	1271	0.0	0.0	13704	0	?	SW	05:18	0:00	/usr/sbin/httpd -
apache	1272	0.0	0.0	13704	0	?	SW	05:18	0:00	/usr/sbin/httpd -
apache	1273	0.0	0.0	13704	0	?	SW	05:18	0:00	/usr/sbin/httpd -
apache	1274	0.0	0.0	13704	0	?	SW	05:18	0:00	/usr/sbin/httpd -
apache	1275	0.0	0.0	13704	0	?	SW	05:18	0:00	/usr/sbin/httpd -
apache	1276	0.0	0.0	13704	0	?	SW	05:18	0:00	/usr/sbin/httpd -
apache	1277	0.0	0.0	13704	0	?	SW	05:18	0:00	/usr/sbin/httpd -
apache	1278	0.0	0.0	13704	0	?	SW	05:18	0:00	/usr/sbin/httpd -
root	1279	0.0	0.0	4440	0	?	SW	05:18	0:00	/usr/sbin/nessusd
nobody	1334	0.0	0.0	2108	36	?	S	05:18	0:00	proftpd (acceptin
root	1365	0.0	0.0	1552	96	?	S	05:18	0:00	crond
xf86	1524	0.0	0.0	7344	0	?	SW	05:19	0:00	xf86-dmccprv -da
root	1597	0.0	0.0	1340	0	tty1	SW	05:19	0:00	/sbin/mingetty tt
root	1598	0.0	0.0	1340	0	tty2	SW	05:19	0:00	/sbin/mingetty tt
root	1599	0.0	0.0	1340	0	tty3	SW	05:19	0:00	/sbin/mingetty tt
root	1600	0.0	0.0	1340	0	tty4	SW	05:19	0:00	/sbin/mingetty tt

출력 결과를 이해하려면 각각의 필드들이 의미하는 것을 알아야 합니다. 다음은 주요 필드들에 대한 설명입니다.

필드	설명
UID	프로세스를 실행시킨 소유자명
PID	Process ID
%CPU	CPU 사용 비율
%MEM	메모리 사용 비율
SIZE	사용중인 가상 메모리 사이즈(KB)
RSS	사용중인 실제 메모리 사이즈(KB)
TTY	실행중인 터미널(tty), 일반적으로 약어로 표시(/dev/tty7 -> p7)
STAT	프로세스의 상태 R : 실행중/실행준비완료 S : sleeping I : idle Z : zombie 수행이 종료되어 상위 프로세스의 종료를 기다리는 프로세스 D : disk wait P : page wait W : swapped out N : nice에 의해 우선 순위가 낮아짐 T : 종료 < : 슈퍼유저가 우선순위를 높임
START	프로세스 시작 시간/날짜
TIME	프로세스가 사용한 CPU 시간
COMMAND	실행되고 있는 명령
NI	nice 우선순위 번호
PR	프로세스 우선순위 번호
PPID	상위 프로세스의 PID
WCHAN	sleeping 상태의 프로세스가 있는 커널 함수 이름
FLAGS	프로세스와 관련된 숫자 값

② pstree 명령

pstree는 앞서 언급했던 부모 프로세스와의 관계를 트리 모양으로 보여주는 명령으로, 프로세스의 근원을 비주얼하게 볼 수 있습니다. 프로세스의 실행 옵션을 보거나, 특정 프로세스를 하이라이팅하여 볼 수 있습니다.

pstree | more

```
init-+-apmd
      |-atd
      |-automount
      |-bdflush
      |-crond
      |-eth0
      |-gdm--gdm-+-X
      |           -gdmlogin
      |-gpm
      |-httpd--8*[httpd]
      |-kapm-idled
      |-keventd
      |-klogd
      |-kreclaimd
      |-kswapd
      |-kupdated
      |-mdrecoveryd
      |-6*[mingetty]
      |-nessusd
      |-portmap
      |-proftpd
      |-rpc.statd
      |-sendmail
      |-sshd
      |-syslogd
      |-xfs
      -xinetd--in.telnetd--login--bash--su--bash-+-more
                                     -pstree
```

(3) 프로세스에게 말걸기

만일 원하지 않는 프로세스가 진행되고 있다면, 그 프로세스를 중지 시켜야 할 것입니다. 이런 경우에 프로세스에 신호를 보내, 현재 진행중인 정상적인 작업이 아닌 예외 상황을 일으키도록 할 수 있습니다. 이렇게 프로세스에 보낼 수 있는 신호에는 여러 종류가 있습니다. 프로세스를 정지시키거나(STOP), 계속(CONT), 종료(KILL), 설정파일을 다시 읽어 들이는 (HUP) 등의 신호가 있습니다. 이런 신호를 보낼 때 사용하는 명령이 kill입니다. 이 kill 명령이 프로세스에 보낼 수 있는 신호의 종류는 kill -l 로 확인할 수 있습니다. 여러종류의 신호가 있지만, 일반 사용자들이 쓸 일은 별로 없습니다. 기껏해야, 자신의 프로세스를 종료시키거나, 설정파일을 다시 읽어 들이거나 하는 신호를 보낼 뿐입니다. 프로세스를 종료시키는 신호는 KILL 인데 신호들은 모두 번호를 가지고 있어서, 다음 두 명령은 같은 결과를 보여줍니다.

```
# kill KILL 1192
# kill -9 1192
```

PID가 1192 번인 프로세스를 종료합니다. kill 명령은 신호를 받을 프로세스를 PID로 지정해 줍니다. 하지만, 일일이 프로세스 번호를 알아낼 필요 없이 프로세스의 이름으로 그 프로세스를 종료시킬 수도 있습니다. 이때 사용하는 명령어가 killall입니다. 이 killall은 편한 만큼 주의를 기울여야 합니다. 자신의 권한(특히 root user의 경우)을 남용하여, 한창 작업중인 다른 사람의 프로세스도 종료시킬 수도 있습니다.

(4) 프로세스 실행에 우선순위 주기

프로세스가 수행되는 데에도 순서가 있습니다. nice 명령으로 이 우선순위를 변경할 수 있는데, 우선순위에는 다음과 같은 규칙들이 있습니다.

- 일반유저는 우선순위를 낮출수만 있습니다.
- 슈퍼유저만이 우선순위를 높일 수 있습니다.
- 우선순위의 범위는 -20에서 19까지입니다.
- 적은 수가 우선순위는 높습니다.

어떤 스크립트에 우선순위를 낮게 주어, 실행시키고자 한다면,

```
# nice -n +10 xxx.pl
```

과 같은 형식으로 실행시킵니다. ps 명령 항목 중에, 우선순위와 관련된 두 개의 항목이 있는데, PRI와 NI입니다. 사용자에 의해 변경된

NI값을 바탕으로 실제적인 프로세스의 우선순위인 PRI가 결정되는 것입니다. 각 프로세스의 우선순위를 확인하려면, ps -l 명령을 실행했을 때, NI 항목을 살펴 보십시오.

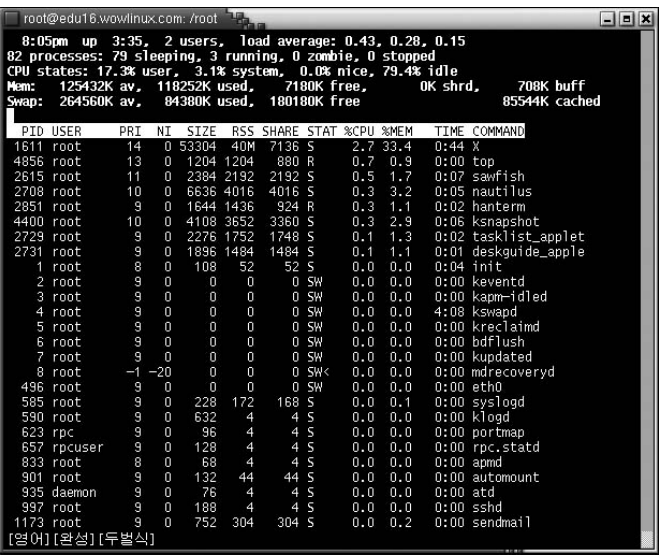
```
# ps -l
```

F	S	UID	PID	PPID	C	PRI	NI	ADDR	SZ	WCHAN	TTY	TIME	CMD
100	S	0	1785	1784	0	69	0	-	578	wait4	pts/0	00:00:00	login
000	S	0	1817	1786	0	69	0	-	519	wait4	pts/0	00:00:00	su
100	S	0	1818	1817	0	74	0	-	552	wait4	pts/0	00:00:00	bash
100	R	0	2015	1818	0	79	0	-	757	-	pts/0	00:00:00	ps

(5) 프로세스 모니터링 하기

① top

top명령은 CPU를 가장 많이 사용하고 있는 프로세스들을 실시간으로 보여 주는 명령입니다.



[그림 III-3] top 실행 화면

- 첫 번째 행에서 부터,
- uptime에서 얻은 값으로 시스템의 최근 프로세스 수행수를 평균낸 값,
- 실행되고 있는 프로세스의 통계값,
- 프로세스들의 CPU 사용 상태,
- 메모리 사용상태,
- 스왑메모리 사용상태를 보여주고 있으며,

그 아래는 ps 명령과 유사한 결과 값으로 현재 활발한 프로세스의 자원 사용상태를 일목요연하게 나타내고 있습니다.

CPU 사용(P), 메모리 사용(M), 실행시간(T)에 따라 작업들을 정렬해서 볼 수 있습니다. top 명령이 수행되는 동안 h를 누르면 사용 가능한 명령들을 볼 수 있습니다.

gtop과 ktop은 X 윈도우에서 같은 기능을 수행합니다.

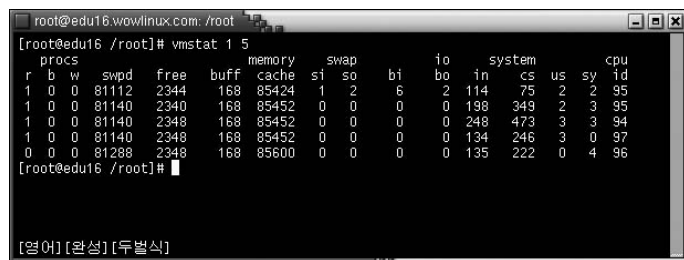


The image shows the gtop utility running in a terminal window. It displays a table of system processes with columns for PID, 사용자 (user), Pri, 크기 (size), 상주 (resident), CPU, MEM, 시간 (time), and 명령 (command). The processes are sorted by CPU usage. At the bottom, there is a summary bar showing system statistics like CPU usage (6.37%), memory usage (3.59%), and load averages.

[그림 III-4] gtop 실행 화면

2 vmstat

시스템에 관한 다양한 상태를 보여주는 vmstat 명령에서도 프로세스의 정보를 확인 할 수 있습니다.



The image shows the output of the vmstat command in a terminal window. The first line is the command prompt and the command itself: root@edu16:/root# vmstat 1 5. The output is a table with columns forprocs, r, b, w, swpd, free, buff, cache, si, so, bi, bo, in, cs, us, sy, id, and cpu. The data shows system statistics over time.

[그림 III-5] vmstat 실행 화면

뒤의 옵션은 1초 간격으로 5번을 디스플레이 하라는 옵션입니다.

첫 번째 항목에 해당하는 procs 부분이 프로세스에 관한 정보로,

r : 대기 프로세스의 수

b: blocked된 프로세스의 수

w: swapped out 된프로세스의 수를 나타냅니다.

그 외에도, 메모리, 스왑상태, 블록 IO, 시스템, CPU에 대한 정보를 모니터링 할 수 있습니다.

4. /proc 파일시스템

리눅스에서 시스템의 정보를 알고 싶으면 /proc 디렉토리의 내용을 확인해 보면 됩니다.

/proc 디렉토리는 실제로 존재하지 않는 임시 데이터입니다.

즉, 커널이 메모리 상에 만들어 놓은 것으로 실제 디스크 공간에는 존재하지 않습니다. /proc은 시스템의 여러가지 정보를 제공해 주는데, 원래는 주로 프로세스에 대한 정보를 제공했기 때문에 proc(process)이란 이름을 가지고 있습니다. 이 곳에 있는 중요한 파일과 디렉토리들을 아래에 설명하였습니다. /proc 파일시스템에 관한 더욱 자세한 정보는 /usr/src/linux/Documentation/proc.txt 를 보시기 바랍니다.

/proc/asound

이 디렉토리는 시스템내에 설치되어진 사운드카드에 대한 정보를 볼수 있습니다.

/proc/cpuinfo

프로세서의 정보가 들어있습니다. cpu의 타입, 모델, 제조회사, 성능 등에 관한 정보를 알려줍니다.

/proc/devices

현재 커널에 설정되어 있는 장치의 목록을 볼 수 있습니다.

/proc/dma

현재 어느 DMA 채널이 사용 중인지를 알려줍니다.

/proc/filesystems

어떤 파일시스템이 커널에 설정되어 있는지를 알 수 있습니다.

/proc/ide

이 디렉토리는 IDE-인테페이스와 이것과 연결된 IDE-device들에 대한 정보를 표시합니다.

/proc/interrupts

현재 어느 인터럽트가 사용 중인지, 그리고 얼마나 많이 사용되었는지를 알 수 있습니다.

/proc/ioports

현재 어느 I/O 포트가 사용 중인지를 알려줍니다.

/proc/loadavg

시스템의 평균부하량(load average)을 보여줍니다.

/proc/meminfo

메모리 사용량에 관한 정보를 보여줍니다. 실제 메모리와 가상 메모리를 모두 다룹니다.

/proc/modules

현재 어떤 커널 모듈이 사용되고 있는지를 알려줍니다.

/proc/partitions

이 페이지는 분할영역에 대한 정보를 표시합니다.

/proc/pci

이 페이지는 PCI-버스와 설치되어진 PCI-카드들, 그리고 주변장치연결 (PCI) 버스를 사용하는 다른 device들에 대한 정보를 표시합니다.

/proc/scsi/scsi

이 페이지는 SCSI-인테페이스와 이것과 연결된 SCSI-device들에 대한 정보를 표시합니다.

/proc/stat

이 곳에는 시스템의 상태에 관한 다양한 정보가 있습니다. 즉, 부팅된 후 page fault가 몇번 일어났는가 하는 것들을 알아 볼 수가 있습니다.

/proc/uptime

시스템이 얼마나 오랫동안 살아 있었는지 보여줍니다.

/proc/version

커널의 버전을 알려줍니다.