

# Linux Administration

# 1장

## 리눅스 시스템의 개괄

- 1.1 운영체제의 구성
- 1.2 커널 핵심부의 구성
- 1.3 리눅스 시스템 설치
- 1.4 리눅스 시스템의 주요기능
- 1.5 시스템 관리자의 역할 및 주요업무

## Overview

- 이 장에서는 리눅스 시스템을 관리하는데 앞서 어떠한 기본지식이 필요한지 간략하게 설명하고, 또한 시스템 관리자의 역할, 시스템 관리자가 수행하는 작업 등에 대해 간략히 서술했다.
- 우선 기본적으로 시스템 전반적인 구성을 알기 위해 운영체제의 구성과 커널 핵심부의 구성에 대해 알아본 후 리눅스 시스템의 설치 방법 및 리눅스 시스템의 주요기능에 대해 순서대로 살펴본다.

## 1.1 운영체제의 구성

### 1.1.1 리눅스의 특징

리눅스 - 인텔 80386 이상을 사용하는 PC에서 운영되는 유닉스 운영체제의 공개 버전

#### 리눅스의 특징

- 소스 코드를 완전 무료로 공개
- 단일 운영체제의 독점이 아닌 다수를 위한 공개라는 원칙하에 지속적인 업그레이드가 이루어짐
- 파일구성이나 시스템기능의 일부는 유닉스를 기반으로 하면서, 핵심 커널 부분은 유닉스와 다르게 작성

#### 리눅스 운영체제가 다른 운영체제에 비해 가지는 장점

- 파일시스템이 윈도우랑 달라서 바이러스에 걸릴 위험이 거의 없다.
- 리눅스는 오픈소스라서 보안에 취약점이 발견되면 전 세계 프로그래머들이 초스피드로 패치를 내놓는다. (윈도우는 MS가 관리하고 있기 때문에 좀 느림)
- 윈도우처럼 그림환경 (GUI)이 아닌 MS-DOS처럼 CUI환경이다.
- 현재는 리눅스에도 GUI환경 작업이 가능한 X-WINDOW가 있지만은 원래는 CUI이다.
- 윈도우보다 덩치가 작기때문에 시스템 사양이 낮아도 잘 돌아간다.
- 이러한 이점 때문에 임베디드 시스템을 만드는데 리눅스를 많이 이용한다.
- 각종 프로그램들을 무료로 구할 수 있다.
- 전세계 리눅스 커뮤니티들이 있어서 빠른 피드백이 가능

#### 물론 장점만 있는건 아니다. 단점은 아래 내용과 같다.

- CUI환경이기때문에 명령어들을 일일이 외우고 다녀야 한다.
- 초보자들은 배우는데 오랜 시간이 걸린다.

### 1.1.2 리눅스 운영체제의 구성

커널(kernel)과 여러가지 시스템 프로그램(system programs) 들로 구성됨  
업무수행을 위한 몇가지 응용 프로그램(application programs)들도 덧붙여져 있다.

#### 커널

운영체제의 심장부

#### 커널의 역할

- 파일들을 디스크에 적절히 배치
- 프로그램을 시동시켜 작업을 수행하게 함
- 메모리와 같은 시스템의 자원(resource)을 각각의 프로세스에 할당
- 네트워크를 통해 패킷(packet)을 주고받을 수 있게함

### 리눅스 운영체제의 구성

- Linux 운영체제는 커널(kernel)과 여러가지 시스템 프로그램(system programs) 들로 이루어져 있다.
- 여기에는 업무수행을 위한 몇가지 응용 프로그램(application programs)들도 덧붙여져 있다.
- 특히 커널은 운영체제의 심장부라고 할 수 있는 부분이다.

#### 커널의 역할

- 파일들을 디스크에 적절히 배치시킨다.
- 프로그램을 시동시켜 작업을 수행하게 한다.
- 메모리와 같은 시스템의 자원(resource)을 각각의 프로세스에 할당한다.
- 네트워크를 통해 패킷(packet)을 주고받을 수 있게 해준다.

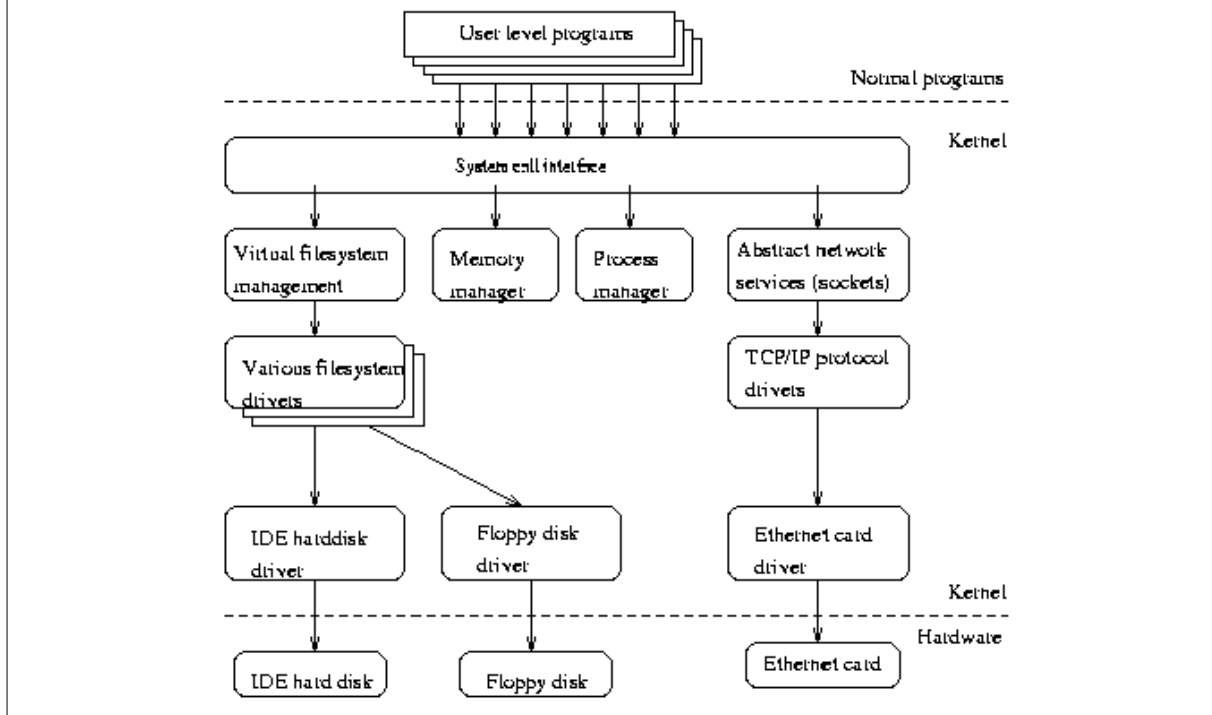
### 커널이 모든 일들을 혼자서 처리하는 것은 아니다.

- 실제로 커널이 혼자서 처리하는 부분은 매우 적다.
- 커널은 기반 설비(tools)들을 제공함으로써 모든 작업을 가능하도록 한다.
- 커널은 기반 설비들을 통하지 않고서는 어떤 것도 직접 하드웨어를 다루지 못하게 한다.
- 이런 방식으로, 하드웨어를 동시에 사용하려는 각각의 사용자들이 서로 충돌하는 일을 막을

수 있다.

- 커널이 갖고 있는 설비들을 사용하는 일은 시스템 콜(system call)을 통해서 이루어진다.

## 1.2 커널 핵심부의 구성



### 커널 핵심부의 구성

커널에서 가장 중요한 구성요소는 메모리관리자와 프로세스 관리자이다.

### 메모리 관리자의 역할

- 프로세스, 커널 일부분, 버퍼 캐쉬를 메모리 영역과 스왑 공간에 적절히 할당하는 역할을 한다.

### 프로세스 관리자의 역할

- 새로운 프로세스를 생성하고 멀티태스킹을 구현한다.

멀티태스킹은 프로세서 상의 프로세스를 계속 스위칭(switching)하는 기법으로 이루어진다.

커널의 가장 밑바탕은 갖가지 종류의 하드웨어 장치 드라이버들로 이루어진다.

하드웨어 장치들을 보면 비슷한 기능하면서도 소프트웨어에 의해 구동되는 방식이 다른 하드웨어가 많은데, 이런 유사성은 비슷한 기능을 통틀어 구동시키는 일반적인 드라이버 클래스를 갖출 수 있게 해준다.

- 그 예로, 모든 디스크 드라이버들은 커널의 나머지 부분에 대해 비슷한 인터페이스를 갖는데, 실제로 디스크 드라이버들은 "드라이브 초기화" "N번째 섹터 읽기" "N번째 섹터 쓰기"와 같은 조작방법을 모두 갖추고 있다.



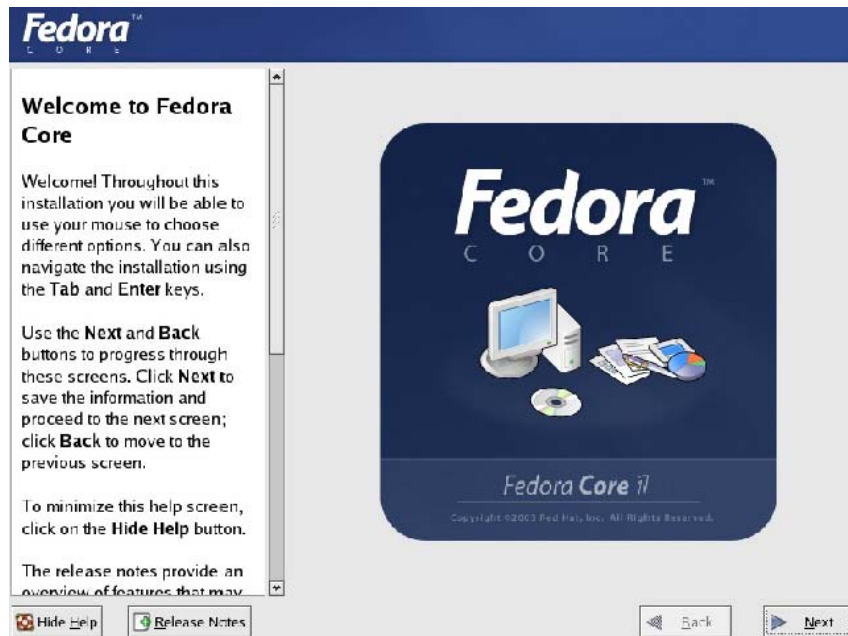
커널이 독립적으로 제공하는 소프트웨어 서비스들 중에도 유사성을 가진 것들이 있다.  
이런 것들은 클래스란 것으로 추상화 될 수 있다.

### 1.3 리눅스 시스템 설치 (Fedora core1 을 기준)

- 우선 리눅스를 다운 받아서 CD 이미지 파일로 굽는다.
- 아래의 주소에서 다운받을 수 있다.  
<http://download.fedora.redhat.com/pub/fedora/linux/core/1/i386/iso/>  
(한국미러) <ftp://ftp.kreonet.re.kr/pub/Linux/fedora/core/1/i386/iso/>
- 여기서 아래 파일들을 다운받는다.
  - yarrow-i386-disc1.iso
  - yarrow-i386-disc2.iso
  - yarrow-i386-disc3.iso
- 이 시디 이미지 파일들을 구워내면 리눅스 시스템을 설치할 준비가 끝난다.

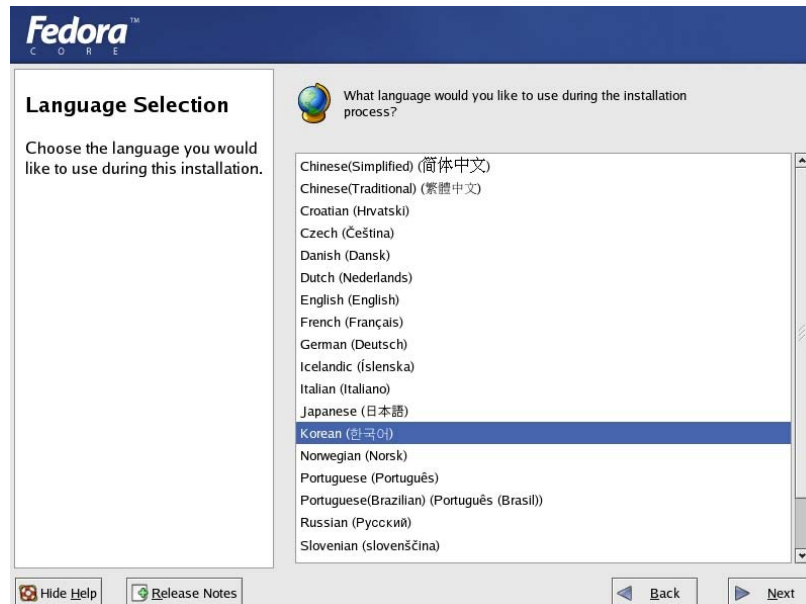
### 1.3 리눅스 시스템 설치 (Fedora core1 을 기준)

- 앞에서 yarrow-i386-disc1.iso 파일을 구운 시디로 부팅을 한다.
- 첫 화면에서 엔터를 치면 아래와 같은 화면으로 넘어 간다.



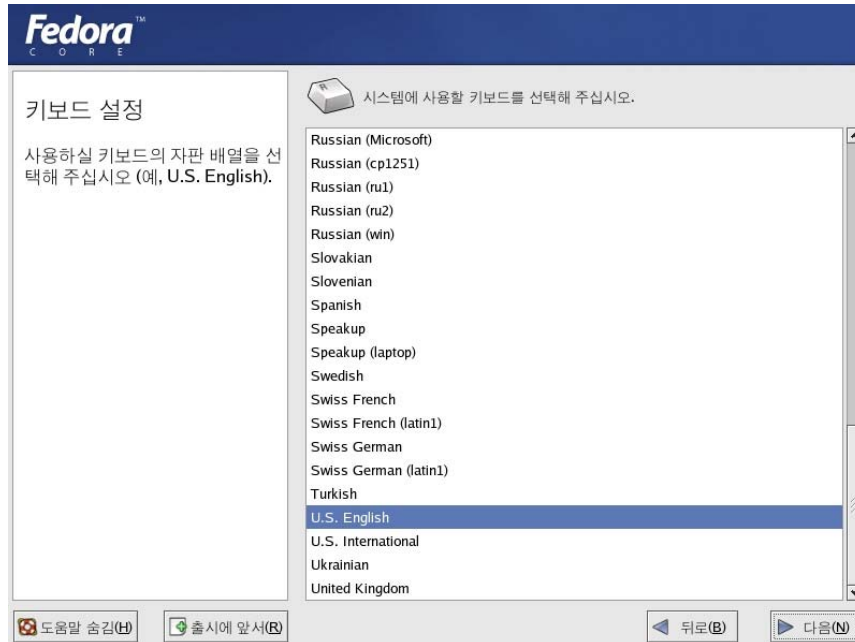
### 1.3 리눅스 시스템 설치 (Fedora core1 을 기준)

- 위 화면에서 Next를 클릭하면 아래와 같은 화면이 나온다.
- 이 화면에서 보통 default 값으로 English로 설정이 되어 있는데, 설치할 때 편의를 위해 Korean(한국어)를 선택한다.



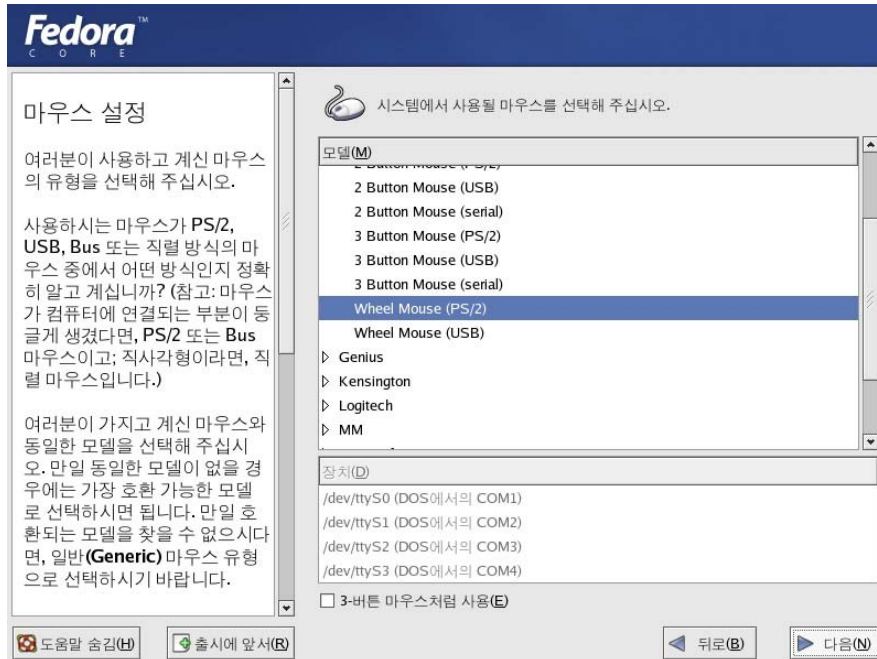
### 1.3 리눅스 시스템 설치 (Fedora core1 을 기준)

- 그 다음으로 넘어가면 아래와 같이 키보드를 선택하는 화면이 나온다.



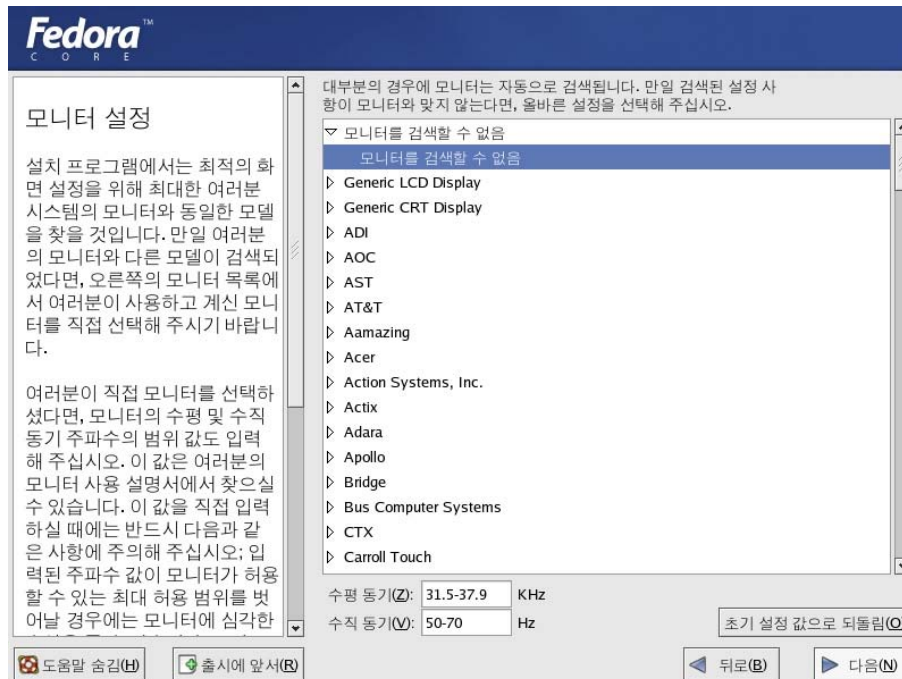
### 1.3 리눅스 시스템 설치 (Fedora core1 을 기준)

- 아래 화면이 나오면 마우스를 선택하고 다음을 클릭한다.



### 1.3 리눅스 시스템 설치 (Fedora core1 을 기준)

- 아래 화면에서 자신의 모니터와 맞는 것을 선택하고 다음을 클릭한다.



### 1.3 리눅스 시스템 설치 (Fedora core1 을 기준)

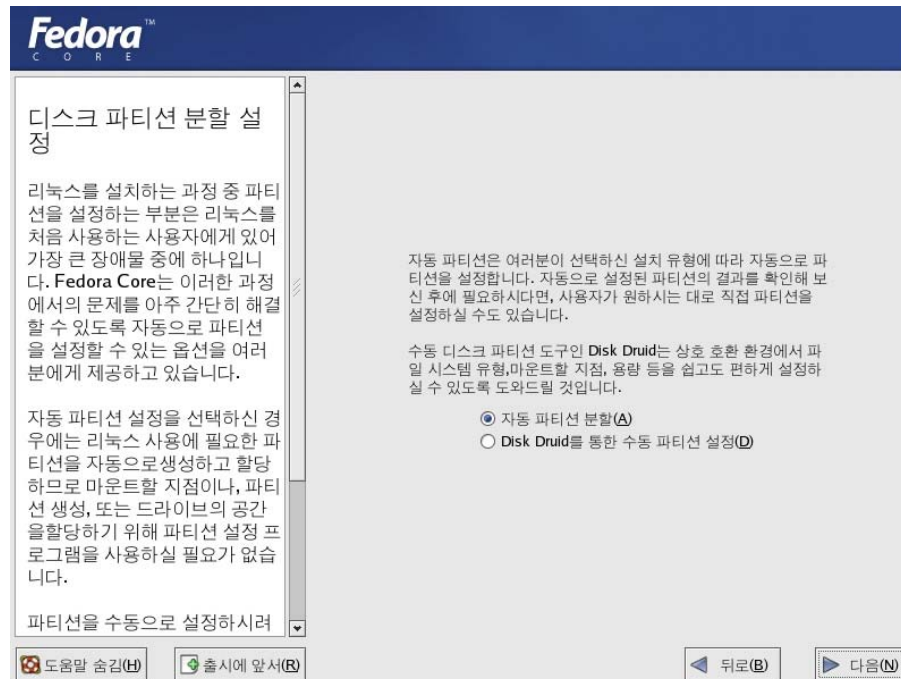
- 아래 화면이 나오면 사용자의 용도에 따라 설치유형을 선택한다.





### 1.3 리눅스 시스템 설치 (Fedora core1 을 기준)

- 파티션을 설정하는 부분이 아래와 같이 나온다.



- 한 하드디스크에 여러 파티션이 존재하고 이 파티션 중에 리눅스를 설치하는 경우라면, ‘Disk Druid를 통한 수동 파티션 설정’을 선택하여 자신이 직접 사용하고자 하는 파티션을 적절한 파일 시스템으로 지정하고 파티션 크기 또한 지정해 주어야 한다.

### 1.3 리눅스 시스템 설치 (Fedora core1 을 기준)

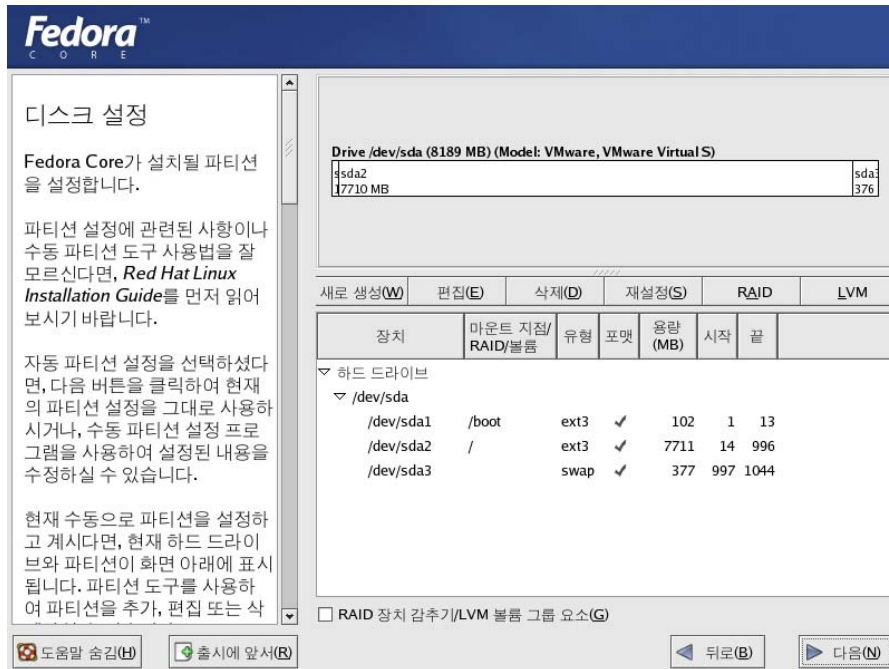
- 앞에서 자동 파티션 분할을 선택하면, 아래와 같은 화면이 나온다.



- 그림에서 보여주는 것처럼 하드디스크에 리눅스에 사용할 파티션이 하나만 필요한 경우에는 '시스템 상의 모든 파티션 삭제'를 선택한 뒤 설치를 진행한다.

### 1.3 리눅스 시스템 설치 (Fedora core1 을 기준)

- 그 다음으로 파티션을 확인하는 화면이 나온다.



### 1.3 리눅스 시스템 설치 (Fedora core1 을 기준)

- 부트로더 설정 뒤, 다음은 네트워크 설정 화면이 나온다.

**네트워크 설정**

설치 프로그램은 시스템에 설치된 모든 네트워크 장치를 검색하여 네트워크 장치 목록에 보여줍니다.

네트워크 장치를 설정하려면, 우선 해당 장치를 선택하신 후 편집 버튼을 클릭해 주십시오. 인터페이스 편집 화면에서 DHCP에 의해 설정된 IP와 넷마스크를 사용하시거나 또는 여러분이 직접 IP 주소, 넷마스크 정보를 입력하실 수 있습니다. 시스템 부팅시에 네트워크 장치가 자동으로 작동되도록 설정도 가능합니다.

DHCP 클라이언트 접속을 하지 않으시거나 이 설정에 대해서 잘 모르시겠다면, 네트워크 관리자에게 문의해 보시기 바랍니다.

**네트워크 장치**

부팅시 활성화	장치	IP/넷마스크
<input checked="" type="checkbox"/>	eth0	DHCP

**호스트명**

호스트명 설정:

☒ DHCP를 통하여 호스트명을 자동으로 설정(A)

☐ 수동 호스트명 설정(M)  (예: "host.domain.com")

**그 외 설정...**

게이트웨이(G):

1차 DNS(P):

2차 DNS(S):

3차 DNS(T):

도움말 숨김(H)    출시에 앞서(R)    < 뒤로(B)    > 다음(N)

## 1.3 리눅스 시스템 설치 (Fedora core1 을 기준)

- 그다음으로 방화벽 설정 화면이 아래와 같이 나온다.



The image shows the 'Fedora Core 1' Firewall Configuration window. The window has a blue header with the 'Fedora' logo and 'CORE' text. The main content area is divided into two panes. The left pane, titled '방화벽 설정' (Firewall Settings), contains text explaining that the firewall protects the computer and network from unauthorized access and that it can be configured to allow or deny specific services. The right pane, titled '방화벽을 사용하여 외부에서 여러분의 컴퓨터에 허가없이 접근하는 것을 방지할 수 있습니다.' (You can use the firewall to prevent unauthorized access to your computer from outside), contains two radio buttons: '방화벽을 사용하지 않음(N)' (Do not use firewall) and '방화벽을 사용함(E)' (Use firewall). The 'Use firewall' option is selected. Below this, there is a section titled '어떤 서비스를 방화벽을 통과할 수 있게 허용하시겠습니까?' (Which services do you want to allow through the firewall?). It lists several services with checkboxes: 'WWW (HTTP)', 'FTP', 'SSH', 'Telnet', and 'Mail (SMTP)'. The 'eth0' interface is selected in the '그 외의 포트(P):' (Other ports) section. At the bottom, there are buttons for '도움말 숨김(H)' (Hide help), '출시에 앞서(A)' (Before launch), '뒤로(B)' (Back), and '다음(N)' (Next).

**Fedora**  
C O R E

### 방화벽 설정

방화벽은 컴퓨터와 네트워크 사이에 위치하며, 다른 네트워크의 사용자들로부터 사설 네트워크의 자원을 보호해줍니다. 적절하게 설정되어진 방화벽은 외부로부터 시스템 보안을 최대한 강화시킬 수 있습니다.

여러분의 시스템에 적용할 보안 수준을 선택해 주십시오.

방화벽을 사용하지 않음 - 이 항목은 모든 접근을 허락하며, 어떠한 보안 검사도 하지 않습니다. 만일 신뢰할 수 있는 (인터넷이 아닌) 네트워크 상에서 시스템을 운영하고 계시거나, 나중에 더욱 세밀한 방화벽 설정을 구상하고 계실 경우에만 이 방법을 사용하시기 바랍니다.

방화벽을 사용하여 외부에서 여러분의 컴퓨터에 허가없이 접근하는 것을 방지할 수 있습니다. 방화벽을 사용하시겠습니까?

☐ 방화벽을 사용하지 않음(N)

☒ 방화벽을 사용함(E)

어떤 서비스를 방화벽을 통과할 수 있게 허용하시겠습니까?

☐ WWW (HTTP)

☐ FTP

☐ SSH

☐ Telnet

☐ Mail (SMTP)

그 외의 포트(P):

특정 장치에서 들어오는 모든 트래픽을 허용하시려면, 아래에서 장치를 선택해 주십시오.

☒ eth0

도움말 숨김(H)    출시에 앞서(A)    뒤로(B)    다음(N)

### 1.3 리눅스 시스템 설치 (Fedora core1 을 기준)

- 그다음 시스템의 기본 언어 선택 화면이 나온다.



### 1.3 리눅스 시스템 설치 (Fedora core1 을 기준)

- 다음은 시간대 선택 화면이 나온다.



## 1.3 리눅스 시스템 설치 (Fedora core1 을 기준)

- 그 다음으로 Root(관리자) 암호를 설정하는 화면이 나온다.



### 루트 암호 설정

루트 계정은 반드시 시스템 관리를 위한 목적으로만 사용하셔야 합니다. 설치가 완료된 후, 일반 용도로 사용할 루트가 아닌 사용자 계정을 만들어 사용하셔야 합니다. 일반 사용자 계정으로 사용하는 도중, 루트 권한으로 작업해야 할 일이 생겼을 경우에는 **su -** 명령을 사용하여 잠시 루트 계정을 이용하실 수 있습니다. 이처럼 일반 사용자 계정을 만들어 사용하는 것은 여러분이 실수로 입력한 명령이나 잘못된 명령으로 인하여 만일의 경우 시스템에 발생될지도 모를 최악의 사태를 최소화 할 수 있기 때문입니다.

 Root (관리자)의 암호를 입력해 주십시오.

Root 암호(P):

확인(C):

 도움말 숨김(H)

 출시에 앞서(R)

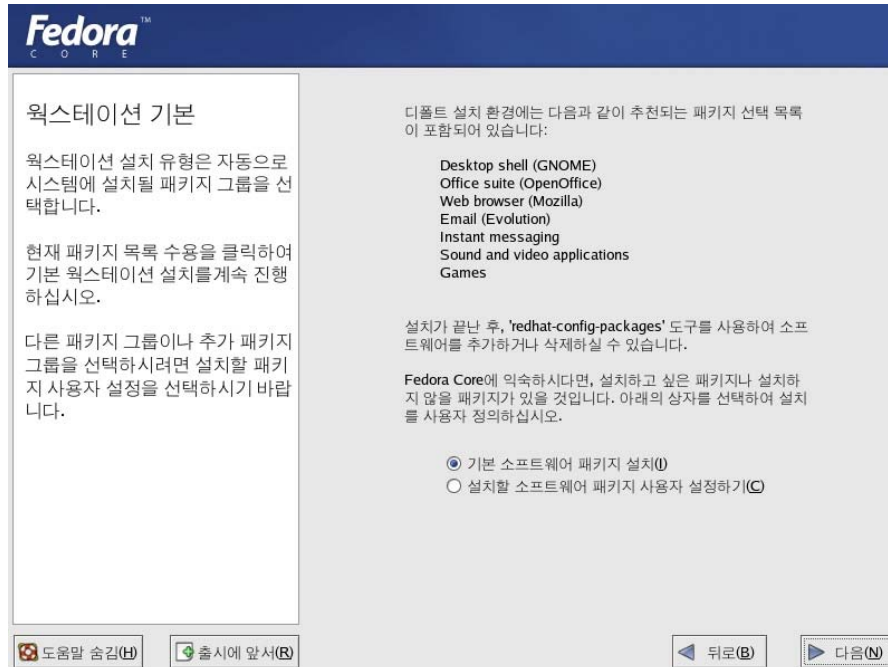
 뒤로(B)

 다음(N)



## 1.3 리눅스 시스템 설치 (Fedora core1 을 기준)

- 다음화면으로 설치할 패키지가 화면에 나온다.



## 1.3 리눅스 시스템 설치 (Fedora core1 을 기준)

- 기본적인 설정이 끝나고 설치를 위해 Next버튼을 누른다.



### 1.3 리눅스 시스템 설치 (Fedora core1 을 기준)

- 설치하는 화면이 아래와 같이 나오고 CD를 바꿔 넣으라는 메시지에 CD를 번갈아 넣어준다.



### 1.3 리눅스 시스템 설치 (Fedora core1 을 기준)

- 설치가 완료되면 부팅디스켓을 작성할지 묻는 화면이 나온다.



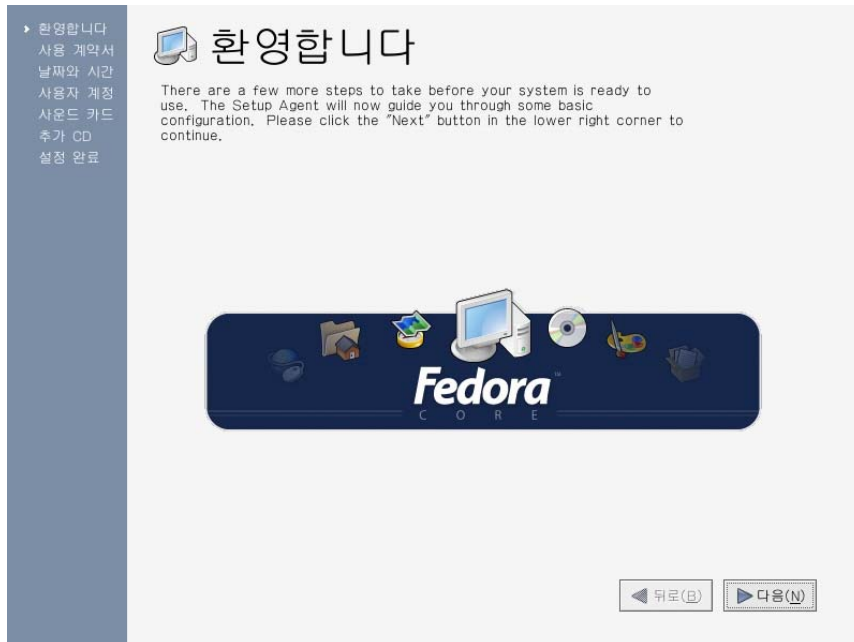
## 1.3 리눅스 시스템 설치 (Fedora core1 을 기준)

- 아래 화면이 나오면 시디를 빼고 재부팅 버튼을 누른다.



## 1.3 리눅스 시스템 설치 (Fedora core1 을 기준)

- 첫 번째로 부팅 시 초기 설정화면이 나온다.



## 1.3 리눅스 시스템 설치 (Fedora core1 을 기준)

- 다음화면으로 사용자 계약서에서 동의를 클릭한다.

환영합니다

▶ 사용 계약서


날짜와 시간

사용자 계정

사운드 카드

추가 CD

설정 완료

 **사용 계약서**

LICENSE AGREEMENT  
FEDORA(TM) CORE 1

This agreement governs the download, installation or use of the Software (as defined below) and any updates to the Software, regardless of the delivery mechanism. The Software is a collective work under U.S. Copyright Law. Subject to the following terms, Fedora Project grants to the user ("User") a license to this collective work pursuant to the GNU General Public License. By downloading, installing or using the Software, User agrees to the terms of this agreement.

1. THE SOFTWARE. Fedora Core (the "Software") is a modular Linux operating system consisting of hundreds of software components. The end user license agreement for each component is located in the component's source code. With the exception of certain image files containing the Fedora trademark identified in Section 2 below, the license terms for the components permit User to copy, modify, and redistribute the component, in both source code and binary code forms. This agreement does not limit User's rights under, or grant User rights that supersede, the license terms of any particular component.

2. INTELLECTUAL PROPERTY RIGHTS. The Software and each of its components, including the source code, documentation, appearance, structure and organization are copyrighted by Fedora Project and others and are protected under copyright and other laws. Title to the Software and any component, or to any copy, modification, or merged notion shall remain with the aforementioned subject to

☐ 네, 사용 계약서에 동의합니다(Y)

☒ 아니오, 동의하지 않습니다(N)

◀ 뒤로(B)

▶ 다음(N)

## 1.3 리눅스 시스템 설치 (Fedora core1 을 기준)

- 그다음으로 날짜와 시간을 설정한다.

환영합니다

사용 계약서


날짜와 시간

사용자 계정

사운드 카드

추가 CD

설정 완료



### 날짜와 시간

시스템 날짜와 시간을 설정해 주십시오.

날짜

← 십이월 → 2003 →

일	월	화	수	목	금	토
30	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10

시간

현재 시간 : 02:30:16

시 : 2

분 : 30

초 : 9

네트워크 시간 프로토콜

네트워크 시간 프로토콜을 사용하여 원격 시간 서버와 현 컴퓨터의 시간을 동기화 시킬 수 있습니다.

☐ 네트워크 시간 프로토콜 활성화(E)

서버(S):

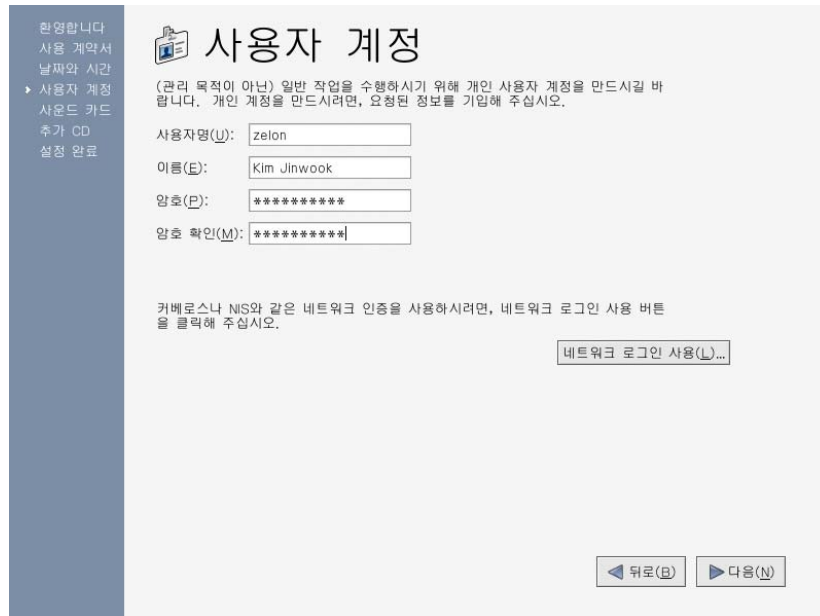
뒤로(B)

다음(N)



## 1.3 리눅스 시스템 설치 (Fedora core1 을 기준)

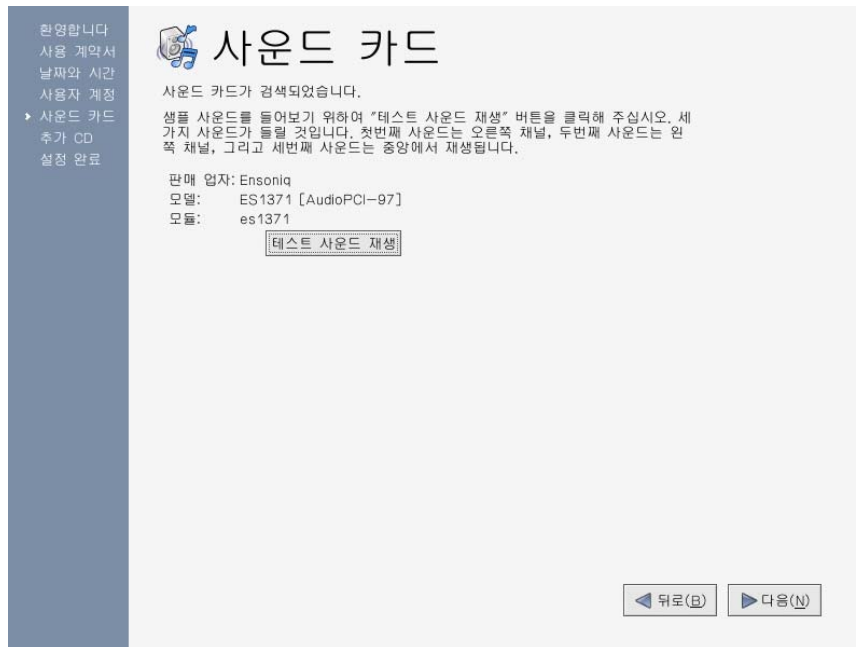
- 다음화면으로 사용자 계정을 설정하는 화면이 나온다.



The screenshot shows the 'User Account' (사용자 계정) setup window in the Fedora Core 1 installer. On the left is a vertical sidebar with options: '환영합니다' (Welcome), '사용 계약서' (License Agreement), '날짜와 시간' (Date and Time), '사용자 계정' (User Account - selected), '사운드 카드' (Sound Card), '추가 CD' (Additional CDs), and '설정 완료' (Finish). The main area is titled '사용자 계정' and contains a note: '(관리 목적이 아닌) 일반 작업을 수행하기 위해 개인 사용자 계정을 만드시길 바랍니다. 개인 계정을 만드시려면, 요청된 정보를 기입해 주십시오.' (For non-admin purposes, we recommend creating a personal user account for general use. To create a personal account, please enter the requested information.) Below this are four input fields: '사용자명(U):' with 'zelon', '이름(E):' with 'Kim Jinwook', '암호(P):' with '\*\*\*\*\*', and '암호 확인(M):' with '\*\*\*\*\*'. At the bottom right, there is a button '네트워크 로그인 사용(L)...' and two navigation buttons: '◀ 뒤로(B)' and '▶ 다음(N)'. A footer note states: '커널로스나 NIS와 같은 네트워크 인증을 사용하시려면, 네트워크 로그인 사용 버튼을 클릭해 주십시오.'

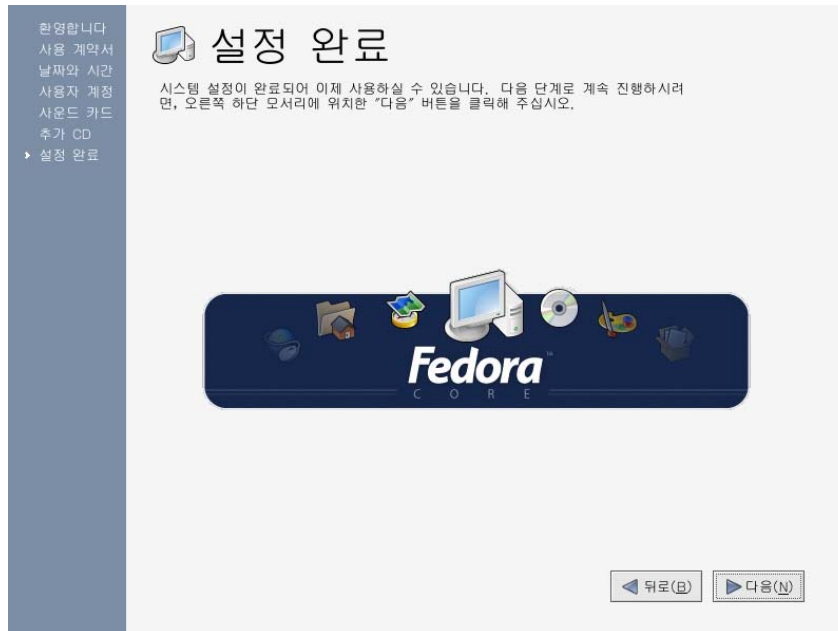
## 1.3 리눅스 시스템 설치 (Fedora core1 을 기준)

- 다음 화면으로 사운드 카드를 설정하는 화면이 나타난다.



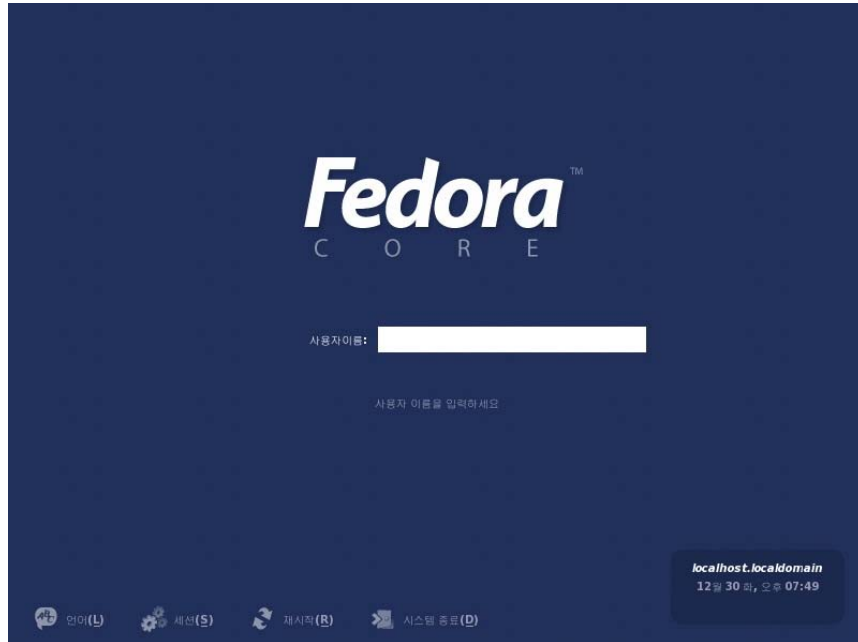
## 1.3 리눅스 시스템 설치 (Fedora core1 을 기준)

- 다음화면으로 설정완료 화면이 나오면 다음을 클릭한다.



### 1.3 리눅스 시스템 설치 (Fedora core1 을 기준)

- 이제 로그인하여 리눅스 시스템을 사용한다.



## 2장

# 디렉토리 트리의 개괄

2.1 전체 디렉토리의 구성

2.2 /etc 디렉토리의 구성

2.3 /usr 디렉토리의 구성

2.4 /var 디렉토리의 구성

2.5 /dev 디렉토리의 구성

2.6 /proc 디렉토리의 구성

## Overview

이 장에서는 리눅스 디렉토리 구성에 대해 설명한다.

- 리눅스 디렉토리를 보면 우리에게 익숙한 OS인 윈도우즈에서의 디렉토리 와 같이 트리구조를 갖는다.
- 리눅스 디렉토리 구성은 비슷한 종류의 데이터를 묶고, 각 그룹간의 경계를 명확히 하는 FHS(Filesystem Hierarchy Standard)의 요구에 의해 결정된다.
- 최상위 디렉토리들은 특정 종류의 데이터 그룹을 갖고 있다. 최상위 디렉토리 내의 서브 디렉토리들은 최상위 디렉토리의 제한조건을 만족하는 데이터를 가진다. 따라서 각각의 최상위 디렉토리는 어떠한 특정한 것을 갖을지 있을 것인가를 정의한다.

## 2.1 전체 디렉토리의 구성

루트 파일시스템 ( / ) - 최상위 디렉토리 계층

루트 디렉토리 ( /root ) - 슈퍼유저의 홈 디렉토리

/bin 디렉토리 - 싱글유저 모드에서 필요로 하는 바이너리 파일들을 모아놓는 디렉토리, 서브 디렉토리 없음

/usr/bin 디렉토리 - 일반 유저들이 사용하는 명령어들을 모아 놓은 디렉토리

/boot 디렉토리 - 부트 설정 파일과 lilo를 제외한 부트 관련 모든 파일을 모아놓은 디렉토리

### 루트 파일시스템과 루트 디렉토리

- /와 /root를 모두 ‘루트’라고 발음하지만 매우 성격이 다른 디렉토리이다.
- /는 리눅스 파일시스템의 최상위 역할
- /root는 슈퍼유저의 홈 디렉토리

### /bin 디렉토리

- 실행 권한과는 상관없이 싱글유저 모드에서 필요로 하는 바이너리 파일들을 가짐
- 이 디렉토리들은 서브 디렉토리를 갖지 않는 반면에, /usr/bin 디렉토리에는 일반 사용자들이 실행할 명령어들이 들어있다.

### /boot 디렉토리

- 부트 설정 파일과 lilo를 제외한 부트 관련 모든 파일을 가지고 있다.
- 커널이 /etc/inittab 파일에 정의된 시스템 초기화 과정 전까지 사용되는 모든 데이터를 저장
- 부트 이미지
  - 부팅에 사용되는 커널 이미지
  - 일반적으로 루트 디렉토리에 넣어 두거나 또는 /boot에 다른 커널 이미지들과 같이 저장
  - 만약 많은 수의 커널 이미지를 갖고 있다면 /boot는 공간을 많이 차지할 것이므로 이런 경우에는 따로 독립적인 파일시스템을 만들어 주는 것이 좋다.



- 리눅스 커널은 /boot 또는 / 내에 위치하고 있으며, /boot 디렉토리 내에는 다수의 커널 이미지가 존재하므로 커널 이미지는 vmlinuz-2.6.11-1.1369\_FC4와 같이 전체 파일명을 지정해야 한다.

**/dev** - 시스템의 모든 디바이스를 액세스할 수 있는 파일들을 모아 놓은 디렉토리

**/etc** - 호스트의 설정 파일을 모아 놓은 디렉토리

**/home** - 개별 사용자의 홈 디렉토리를 모아 놓은 디렉토리

**/lib** - 시스템 부트 때 필요하거나 또는 /bin 디렉토리 내의 명령어들의 실행에 필요한 공유 라이브러리들을 모아 놓은 디렉토리

### **/dev 디렉토리**

- 소켓 및 네임 파이프처럼 블록 디바이스 파일과 문자 디바이스 파일을 가짐
- 시스템의 모든 디바이스를 액세스할 수 있는 파일이 위치
- 만약 필요한 디바이스가 /dev 디렉토리 내에 존재하지 않는다면, mknod 명령어나 MAKEDEV 스크립트를 사용하여 만들 수 있다.

### **/etc 디렉토리**

- 호스트의 설정 파일을 가짐
- 체계적인 설정파일 관리를 위해서 /etc 디렉토리 밑에 많은 디렉토리들이 추가됨
  - 그 예로, X관련 설정 파일은 /etc/X11 디렉토리에, /etc/ppp, /etc/httpd 디렉토리 등 /etc 디렉토리 안에서도 필요한 설정 파일을 쉽게 찾을 수 있도록 그 하위 디렉토리들이 체계적으로 있게 되었다.

### **/home 디렉토리**

- 사용자의 홈 디렉토리가 위치

## **/lib 디렉토리**

- 시스템 부트 때 필요하거나 또는 /bin 디렉토리 내의 명령어들의 실행에 필요한 공유 라이브러리들이 위치함
- /usr 내의 명령어들이 필요로 하는 라이브러리는 /usr/lib 내에 위치
- /lib/modules 경우 로딩 가능한 커널 모듈들이 위치하는 곳이다. 특별한 경우 장애를 복구하기 위해 시스템을 부팅할 때도 커널 모듈들이 필요함

/mnt - 스토리지 디바이스가 임시로 사용하는 마운트 포인트

/opt - 시스템을 지원하는 소프트웨어 패키지의 모든 데이터를 보관하는 디렉토리

/proc - 프로세스들 및 현재 수행되고 있는 커널의 정보를 가지고 있는 파일 시스템, 가상 파일시스템을 위한 공간

/sbin - 기존엔 정적으로 링크된 바이너리 파일만을 가지고 있었으나, 현재는 시스템 관리에 필요한 바이너리 파일을 가지는 디렉토리이다.

### /mnt 디렉토리

- 하드 드라이브 또는 플로피 드라이브 등의 스토리지 디바이스가 임시로 사용하는 마운트 포인트
- 각 마운트 포인트는 실제로 마운트된 후에는 해당 디바이스의 파일시스템을 갖게 되는 것이다.
- 만약 CD-ROM을 마운트 할 경우, /mnt/cdrom으로 액세스 할 수 있다.

### /opt 디렉토리

- 시스템을 지원하는 소프트웨어 패키지의 모든 데이터를 보관하는 곳
- AT&T SysV 유닉스의 계층 파일시스템과 솔라리스 운영체제로부터 차용한 기능
- 수세의 배포판에서는 일반적으로 사용되기도 하지만 레드햇 및 기타 다른 배포판에서는 /opt를 거의 사용하지 않음

### /proc 디렉토리

- 가상 파일시스템을 위한 공간
- 실제로 물리적인 디스크를 사용하지는 않음
- 시스템에서 프로세스들 및 현재 수행되고 있는 커널의 정보를 가짐
- 시스템의 문제를 해결하는데 매우 유용

## /sbin 디렉토리

- 원래 정적으로 링크된 바이너리 파일만을 가지고 있었지만, 현재는 시스템 관리에 필요한 바이너리 파일을 가지고 있으며 슈퍼유저만 사용해야 한다.
- 이러한 바이너리 파일들은 시스템의 부트 과정에서 사용하게 된다.

/tmp - 임시파일을 저장할 때 사용되는 디렉토리

/usr - 공유가 가능하며 변경되지 않는 데이터를 저장하는 디렉토리

/var - 로그, 잠금파일, 프로세스 관련 파일 등 수시로 변경되는 데이터 파일을 저장하는 디렉토리

### /tmp 디렉토리

- 프로그램이 임시 파일을 저장할 때 사용되며, 프로그램이 종료되면 그 파일은 삭제가 된다.
- /var가 독립된 파티션으로 존재하는 경우에는 /tmp가 /var/tmp로 링크되어 있는 경우가 있으며, 이렇게 하면 루트 파티션의 공간을 절약한다.

### /usr 디렉토리

- 공유 가능하며 변경되지 않는 데이터를 저장
- /usr이 독립적인 파티션으로 만들어져 있다면 읽기 전용으로 마운트 될 수 있다.
- 이러한 과정을 쉽게 하기 위해서 /usr에는 변경되지 않는 정적인 데이터만 존재해야 하며, 변경되는 데이터는 /var/ 디렉토리로 링크를 걸어 그 특성을 보호해야 한다.

### /var 디렉토리

- 로그, 잠금파일, 프로세스 관련 파일 등 수시로 변경되는 데이터 파일을 저장
- /usr에 저장될 데이터를 /var로 저장함으로써 /usr이 읽기 전용으로 사용될 수 있도록 도와준다.
- /var는 /var/log/messages 등의 시스템 로그를 가지고 있으며, 에러 메시지를 빈번히 발생시키는 문제가 있다면 /var 디렉토리가 가득차게 되기 때문에, /var는 보통 루트 파티션과는 별도로 생성함으로써 / 파티션이 가득차서 로그인할 수 없게 되는 문제를 막아준다.

## 2.2 /etc 디렉토리

### **/etc/rc or /etc/rc.d or /etc/rc?.d**

- 시스템 시작시나 실행 레벨이 바뀔 때 실행되는 스크립트들이나 그런 스크립트를 모아놓은 디렉토리
- 이곳에서 실행되는 스크립트는 뒤에 7장 부팅과 쉼다운에서 init을 다룬 부분에서 다룸

### **/etc/passwd**

- 사용자들의 데이터베이스 파일이 저장되는 디렉토리

### **/etc/fdprm**

- 플로피 디스크 파라미터 테이블

### **/etc/rc , /etc/rc.d , /etc/rc?.d**

- 시스템 시작시나 실행 레벨이 바뀔 때 실행되는 스크립트 또는 그러한 스크립트를 모아둔 디렉토리
- /etc/rc.d/ 디렉토리 안에는 init.d라는 디렉토리가 존재. 이 init.d 디렉토리에 있는 파일들은 리눅스 시스템의 여러 서비스를 시작하고 중지할 수 있다.
- example  
`$ /etc/rc.d/init.d/network restart`

### **/etc/passwd**

- 사용자들의 데이터베이스 파일로서 사용자들의 username, 실제이름, 홈 디렉토리의 위치, 암호화된 패스워드, 기타 정보들이 수록됨
- 이 파일의 형식에 대한 내용은 뒤에 나오는 ‘9장 사용자 계정의 관리’에서 다룬다.

### **/etc/fdprm**

- 플로피 디스크 파라미터 테이블
- 서로 유사한 플로피 디스크들 사이의 차이점에 대한 정보를 가진다.

### **/etc/fstab**

- 시스템 시작시 마운트 명령에 의해 자동으로 마운트 될 파일시스템들이 나열

### **/etc/group**

- 각 그룹의 정보를 기재

### **/etc/inittab**

- init의 설정파일

### **/etc/issue**

- 로그인 프롬프트가 뜨기 전에 이 파일의 내용을 화면에 보여줌

### **/etc/fstab**

- 시스템 시작시 mount -a 명령(/etc/rc에 설정되어 있음)에 의해 자동으로 마운트 될 파일시스템들이 나열되어 있다.
- 또한 swapon -a 명령에 의해 사용되는 스왑 영역에 대한 정보도 수록이 되어있다.
- 자세한 정보는 ‘4장 파일시스템의 mount & unmount’ 부분을 참고

### **/etc/group**

- /etc/passwd와 비슷하지만, 사용자들의 정보가 아닌 각 그룹들의 정보를 저장
- 자세한 내용은 ‘9장 사용자 계정의 관리의 group 설정’ 부분을 참고

### **/etc/inittab**

- init의 설정파일이다. init에 대해 간략히 알아보면,
- 하드웨어 인식과 초기화 작업을 마친 후 커널은 프로세스 ID(PID) 1번의 init을 실행 시킨다. 시스템의 첫 번째 프로세스인 init은 실행하는 실행하는 모든 프로세스들의 궁극적인 부모역활을 하는 프로세스이다.
- init 이 실행될 때는 제일 먼저 /etc 에 있는 inittab이란 파일을 읽어들인다.
- 이 파일은 init이 해야 할 모든 일이 적혀 있는 파일이라 할 수 있다.



## /etc/issue

- 로그인 프롬프트가 뜨기 전에 이 파일의 내용을 화면에 뿌려준다.
- 시스템의 간단한 정보나 환영메시지 등을 적는 것이 일반적이다.

### **/etc/magic**

- file 명령의 설정 파일

### **/etc/motd**

- 'Message Of The Day'의 약자로 로그인을 할 때마다 이 파일의 내용이 출력됨

### **/etc/mtab**

- 현재 마운트되어 있는 파일 시스템의 목록이 저장

### **/etc/magic**

- file 명령의 설정 파일
- 이 파일에는 다양한 파일 형식들의 정보가 포함되어 있다. file 명령은 이것을 기반으로 파일의 정체를 추측함.

### **/etc/motd**

- Message Of The Day
- 로그인할 때마다 자동으로 이 파일의 내용이 출력됨 (시스템 가동 중지 같은 예고 등을 할 때 주로 쓰임)

### **/etc/mtab**

- 현재 마운트되어 있는 파일 시스템의 목록이 들어있음
- 스크립트에 의해 초기화되며, mount 명령에 의해 그 내용으로 자동으로 갱신됨
- 마운트되어 있는 파일시스템 목록이 필요한 경우 사용하는데, 예를들면 df 명령이 이 파일을 읽는다

### **/etc/shadow**

- 새도우 패스워드를 보관

### **/etc/login.defs**

- 로그인 명령의 설정 파일

### **/etc/printcap**

- 프린터의 설정파일

### **/etc/profile, /etc/csh.login, /etc/csh.cshrc**

- 시스템 시작될 때나 로그인이 이루어질 때, Bourne 셸이나 C 셸에 의해 실행되는 파일들

### **/etc/shadow**

- 새도우 패스워드 소프트웨어가 설치되어 있는 시스템의 경우, 이곳에 새도우 패스워드가 보관 됨
- 새도우 패스워드
  - /etc/passwd 파일에서 암호화된 패스워드 부분만을 떼어내서 /etc/shadow에 보관해 두는 것
- 새도우 패스워드의 목적
  - /etc/shadow는 단지 루트 사용자만이 읽을 수 있기 때문에 패스워드가 쉽게 크랙되는 것을 방지할 수 있음

### **/etc/login.defs**

- login 명령의 설정 파일
- 이 파일을 열어보면 메일을 저장하는 디렉토리, 패스워드 관련사항(암호 만료일, 길이, 경고 일 등), 사용자 추가시 생성되는 UID의 범위, UMASK 등을 설정할 수 있다.

### **/etc/printcap**

- 프린터의 설정파일

`/etc/profile`, `/etc/csh.login`, `/etc/csh.cshrc`

- 시스템이 시작될 때나 로그인이 이루어질 때 Bourne 셸이나 C 셸에 의해 실행되는 파일들
- 이 파일들을 사용하면 모든 사용자들의 기본 환경을 설정해 줄 수 있다.

### **/etc/securetty**

- root의 로그인이 허용되는 안전한 터미널을 지정함

### **/etc/shells**

- 신뢰할 수 있는 셸이 어떤 것인지를 지정함

### **/etc/termcap**

- 여러 가지 터미널들의 특성을 데이터베이스로 만들어 둔 것

### **/etc/securetty**

- 루트의 로그인이 허용되는 안전한 터미널을 지정한다.
- 일반적으로 가상 콘솔들만 나열되어 있는데, 이는 누군가가 모뎀이나 네트워크를 통해 시스템에 침입하여 슈퍼유저 권한을 얻는 일을 막도록 하기 위함이다.

### **/etc/shells**

- 신뢰할 수 있는 셸이 어떤 것인지를 지정한다.
- chsh 명령으로 로그인 셸을 바꿀 때 이 곳에 나열된 셸들만 지정할 수 있다.
- FTP 서비스를 제공하는 ftpd 서버 프로세스의 경우, 사용자의 셸이 /etc/shells에 나열된 것과 일치하는지 확인하고, 만약 일치하지 않는다면 로그인을 거부한다.

### **/etc/termcap**

- 여러 가지 터미널들의 특성을 데이터베이스로 만들어 둔 것.
- 다양한 종류의 터미널들이 각각 어떠한 이스케이프 시퀀스(escape sequence)를 통해 제어될 수 있는지 기재를 하여서, 프로그램들은 현재 터미널의 종류가 어떤 것인지를 확인하고 /etc/termcap에서 해당 터미널에 알맞은 이스케이프 시퀀스를 찾아서 사용하게 된다. 따라서 각각의 프로그램이 터미널들의 특성에 대해 일일이 알고 있을 필요가 없으면서도, 대부분의 터미널에서 잘 동작하게 된다.

## 2.3 /usr 디렉토리

### /usr 디렉토리

- 모든 프로그램들이 이곳에 설치
- /usr 디렉토리에는 배포판에서 제공하는 파일들이 존재
- /usr/local 디렉토리에는 그 밖에 따로 설치되는 프로그램들과 내부적 용도로의 프로그램들이 설치된 파일이 존재

### /usr/X11R6

- X Window System의 모든 파일들이 있음

### /usr/X386

- /usr/X11R6과 비슷한 것으로, X11 Release 5를 위한 것이다.

### /usr 디렉토리

- 리눅스에서 사용하는 모든 애플리케이션 및 시스템 파일들이 위치하는 디렉토리
- Library 파일, 실행 파일들이 존재
- /usr 의 하위 디렉토리들은 시스템에 사용되는 매우 중요한 프로그램들과 설정 파일들을 지니고 있다. 그러므로 /usr 디렉토리는 리눅스 시스템에서 가장 많은 용량을 차지하는 부분이기도 하다.

### /usr/X11R6

- X Window System의 모든 파일들이 있음
- X 윈도 시스템의 개발과 설치를 편리하게 하기 위해 디렉토리를 따로 만듦
- 이 디렉토리 구조는 /usr 자체의 디렉토리 구조와 흡사

### /usr/X386

- /usr/X11R6과 비슷한 것으로, X11 Release 5를 위한 것임.

### **/usr/bin**

- 사용자들을 위한 대부분의 명령들이 존재

### **/usr/sbin**

- 시스템 관리를 위한 명령들 중 루트파일 시스템에는 있을 필요가 없는 명령들이 위치

### **/usr/man, /usr/info, /usr/doc**

- 각각 매뉴얼 페이지, GNU Info 문서들, 그리고 기타 다른 문서들을 위한 디렉토리들

### **/usr/include**

- C 프로그래밍 언어를 위한 헤더 파일들이 위치

### **/usr/bin**

- 사용자들을 위한 대부분의 명령들이 있으며, 그 밖에 몇몇은 /bin 이나 /usr/local/bin에 위치

### **/usr/sbin**

- 시스템 관리를 위한 명령들 중, 루트 파일시스템에는 필요가 없는 명령들이 이곳에 위치
- 대부분의 서버 프로그램들이 이곳에 위치

### **/usr/man, /usr/info, /usr/doc**

- 각각 매뉴얼 페이지, GNU Info 문서들, 기타 다른 문서들이 위치

### **/usr/include**

- C 프로그래밍 언어를 위한 헤더파일이 위치
- 원칙적으로 /usr/lib 아래에 있어야 하지만, 초기에 이 위치에 있어왔기 때문에, 아직 이곳에 남아있음

### **/usr/lib**

- 프로그램과 서브 시스템들의 고정적인 데이터 파일이 위치
- 전체 시스템에 폭넓게 적용될 수 있는 설정 파일들도 위치

### **/usr/local**

- 내부적인 용도의 프로그램들과 기타 파일을 위한 디렉토리

### **/usr/lib**

- 프로그램들과 서브시스템들의 고정적인 데이터 파일들이 위치
- 전체 시스템에 폭넓게 적용될 수 있는 설정 파일들도 이곳에 위치
- 원래 이 디렉토리는 프로그래밍 서브루틴들의 라이브러리가 있던 곳이기 때문에 지금까지 /lib라는 이름이 붙게 됨

### **/usr/local**

- 내부적인 용도의 프로그램들과 기타 파일들을 위한 디렉토리



## 2.3 /var 디렉토리

### /var 디렉토리

- 시스템 운용 중 계속 갱신되는 데이터들을 모아놓은 디렉토리

### /var/catman

- 포맷된 매뉴얼 페이지들이 잠시 대기하는 디렉토리

### /var/lib

- 일반적인 시스템 운용시 계속 갱신되는 파일들을 위한 공간

### /var 디렉토리

- 시스템 운용 중 계속 갱신되는 데이터들이 위치
- 이 디렉토리 내의 데이터들은 각 시스템에 고유한 것으로서, 네트워크를 통해 공유될 수 있는 성질의 것이 아니다.

### /var/catman

- 포맷된 매뉴얼 페이지들을 잠시 대기시키는 디렉토리
  - 매뉴얼 페이지는 여러 가지 형식으로 출력될 수 있는데, 출력될 형식에 알맞도록 먼저 포맷을 한 후 보게 된다. 포맷되지 않은 매뉴얼 페이지들은 보통 압축된 형태로 /usr/man/man\*에 위치한다.
  - 어떤 매뉴얼 페이지들은 미리 포맷되어 있기도 한데, 이런 것들은 /usr/man/cat\*에 들어가는 것이 일반적이다.
  - 포맷은 처음 볼때만 한번하면 되고, 그 뒤에 같은 페이지를 보는 사람은 /var/man에 있는 것을 바로 꺼내볼 수 있으므로 포맷될 때까지 기다릴 필요가 없어진다.

### /var/lib

- 일반적인 시스템 운용시 계속 갱신되는 파일들을 위한 디렉토리

### **/var/local**

- /usr/local 아래에 설치된 프로그램들의 다양한 데이터가 보관

### **/var/lock**

- 잠금파일이 위치하는 디렉토리

### **/var/log**

- 다양한 프로그램들의 로그파일이 있는 디렉토리

### **/var/local**

- /usr/local 아래에 설치된 프로그램들의 다양한 데이터가 보관되는 디렉토리
- 그 밖에 내부적으로 사용할 목적으로 설치된 프로그램이라 하더라도 /var의 하위 디렉토리를 사용하여 데이터를 보관하기도 한다.

### **/var/lock**

- 잠금파일이 있는 디렉토리
- 일반적으로 많은 프로그램들이 특정한 장치나 파일을 독점적으로 사용하고 있을 때 /var/lock 에다 잠금 파일을 만든다.

### **/var/log**

- 다양한 프로그램들의 로그파일이 있는 디렉토리, 특히 login과 syslog의 로그파일이 이 디렉토리에 있음
  - login의 로그파일
    - /var/log/wtmp에 위치
    - 시스템의 모든 로그인, 로그아웃 정보를 기록

- syslog의 로그파일
  - /var/log/messages에 위치
  - 커널과 시스템 프로그램들의 모든 메시지들을 기록
  - /var/log 안에 있는 파일들은 크기가 무제한으로 커질 수 있으므로 정기적인 삭제가 필요

### **/var/run**

- 시스템 현재 정보들을 담고 있는 디렉토리

### **/var/spool**

- 메일이나 뉴스, 프린트 큐같은 대기상태에 있는 작업들을 위한 곳

### **/var/tmp**

- 임시 파일들보다는 좀 더 오래 유지될 필요가 있는 임시파일들을 보관

### **/var/run**

- 시스템의 현재 정보들을 담고 있는 디렉토리, 부팅을 다시하면 그 내용이 바뀌어 진다.
- 예를들면 /var/run/utmp 파일에는 현재 로그인한 사용자에게 대한 정보가 있다.

### **/var/spool**

- 메일이나 뉴스, 프린터 큐 같은 대기상태에 있는 작업들을 위한 디렉토리가 위치

### **/var/tmp**

- 임시파일들보다는 좀 더 오래 유지될 필요가 있는 임시 파일들이 위치

## 2.5 /dev 디렉토리의 구성

### /dev

- 시스템의 각종 디바이스들에 접근하기 위한 디바이스 드라이버들이 저장 되어있는 디렉토리

### /dev/console

- 시스템의 콘솔

### /dev/hda

- 시스템의 하드디스크

## /dev 디렉토리

- 시스템의 각종 디바이스들에 접근하기 위한 디바이스 드라이버들이 저장
- 물리적인 용량을 갖지 않는 가상 디렉토리
- 대표적으로 하드 드라이브, 플로피, CD-ROM 등이 존재
- 리눅스 시스템의 특징 중 하나는 윈도우와 달리 각종 디바이스 장치들을 하나의 파일로 취급함.
- 따라서 시스템은 각각의 장치들로부터의 정보를 /dev 디렉토리에 존재하는 해당 장치 파일로부터 가져오게 됨.

### /dev/console

- 시스템의 콘솔을 의미

### /dev/hda

- 시스템의 하드디스크 중 첫 번째 하드 디스크를 의미
- /dev/hda1, /dev/hda2 식으로 파티션을 구분
- 여러 개의 하드디스크가 부착되어 있을 경우 /dev/hdb, /dev/hdc 등으로 구분

### **/dev/lp**

- 시스템의 병렬 포트 장치들

### **/dev/null**

- 데이터를 보내면 모두 폐기처리하는 곳

### **/dev/pty**

- 시스템으로 원격접속을 위한 pseudo-terminal들

### **/dev/sda**

- SCSI 장치들

### **/dev/lp**

- 시스템의 병렬 포트 장치들

### **/dev/null**

- 이 장치로 데이터 등을 보내면 모두 폐기되므로 주의해야 함
- 블랙홀이라고도 불리는 특별한 장치

### **/dev/pty**

- 시스템으로의 원격 접속을 위한 'pseudo-terminal'들이다.
- 시스템 계정 사용자들이 원격지에서 시스템으로 텔넷 등을 이용하여 시스템에 접속을 시도하면 이들은 /dev/pty 디바이스들을 사용하게 되는 것이다.

### **/dev/sda**

- SCSI 장치들
- 만약 시스템에 스카시 하드 디스크를 장착했다면, 시스템은 /dev/sda파일에서 정보를 얻어 장치에 접근한다.

### **/dev/ttyS**

- 직렬포트 장치들

### **/dev/cuaS**

- Callout 장치

### **/dev/tty**

- 시스템의 가상 콘솔들

### **/dev/ttyS, /dev/cuaS**

- /dev/ttyS는 직렬포트 장치들
- /dev/cuaS는 Callout 장치

### **/dev/tty**

- 시스템의 가상 콘솔들
- 하나의 화면에 여러 개의 콘솔들을 만드는 기능
- Alt+F1 , Alt+F2 등으로 리눅스에서 제공한 여러 개의 가상 콘솔을 직접 확인할 수 있음

## 2.6 /proc 파일시스템

### /proc 파일시스템

- 커널이 메모리 상에 만들어 놓은 것으로 디스크에는 존재하지 않음
- 시스템의 여러 정보를 제공하는 역할

### /proc/1

- 프로세스 번호 1번에 대한 정보가 있는 디렉토리

### /proc/cpuinfo

- 프로세서 정보가 위치

### /proc/devices

- 현재 커널에 설정되어 있는 장치들의 목록이 위치

### /proc 파일시스템

- 커널이 메모리 상에 만들어 놓은 것으로 디스크에는 존재하지 않음
- 시스템의 여러 정보를 제공
- 원래는 프로세스에 대한 정보를 제공했기 때문에 proc(process)란 이름을 갖게 됨

### /proc/1

- 프로세스 번호 1번에 대한 정보가 있는 디렉토리
- 각 프로세스는 자신만의 디렉토리를 /proc 아래에 갖고 있게 되는데, 자신의 프로세스 식별번호가 그 디렉토리의 이름이 된다.

### /proc/cpuinfo

- 프로세서의 정보가 들어있다.
- cpu의 타입, 모델, 제조회사, 성능 등에 관한 정보가 있음

### /proc/devices

- 현재 커널에 설정되어 있는 장치의 목록을 볼 수 있다.



### **/proc/dma**

- 현재 어느 DMA 채널이 사용 중인지를 알려줌

### **/proc/filesystems**

- 어떤 파일 시스템이 커널에 설정되어 있는지를 알려줌

### **/proc/interrupts**

- 현재 어느 인터럽트가 사용 중이고 얼마나 많이 사용되었는지를 알려줌

### **/proc/ioprots**

- 현재 어느 I/O 포트가 사용 중인지를 알려준다

### **/proc/dma**

- 현재 어느 DMA 채널이 사용 중인지를 알려준다.
  - DMA(Direct Memory Access) - 디바이스가 CPU를 거치지 않고 직접 메모리를 읽고 쓸 수 있는 방법으로 쓰이고 있다. 이 방법을 쓰면 CPU가 작업하는데 있어 할 일의 수고를 어느 정도 덜어주는 역할을 한다.

### **/proc/filesystems**

- 어떤 파일 시스템이 커널에 설정되어 있는지를 알 수 있다.

### **/proc/interrupts**

- 현재 어느 인터럽트가 사용 중인지, 또한 얼마나 많이 사용되었는지를 알 수 있다.

### **/proc/ioprots**

- 현재 어느 I/O 포트가 사용 중인지를 알려 준다

### **/proc/kcore**

- 시스템에 실제 메모리의 이미지 정보를 제공

### **/proc/kmsg**

- 커널이 출력하는 메시지들에 대한 정보를 제공

### **/proc/ksyms**

- 커널이 사용하는 심볼들의 표를 저장한 정보를 제공

### **/proc/loadavg**

- 시스템의 평균 부하량(load average) 정보를 제공

### **/proc/kcore**

- 시스템에 장착된 실제 메모리의 이미지(실제 메모리의 내용을 그대로 본뜬 것)
- 프로그램이 필요로 하는 부분의 이미지만 상황에 따라 만들어 내기 때문에, 실제 메모리의 크기만큼 차지하진 않는다.

### **/proc/kmsg**

- 커널이 출력하는 메시지들 (syslog 파일에도 저장됨)

### **/proc/ksyms**

- 시스템 커널이 사용하고 있는 심볼들에 대한 정보를 저장하고 있는 파일

### **/proc/loadavg**

- 현재 시스템의 평균 부하량(Load Average)에 대한 정보를 저장하고 있는 파일
- 이 파일을 통해 시스템이 현재 수행해야 하는 일이 얼마나 많은지를 알려주는 3가지 지표에 대한 정보를 얻을 수 있다.

### **/proc/meminfo**

- 메모리 사용량에 관한 정보 제공

### **/proc/modules**

- 현재 로드된 커널 모듈에 대한 정보를 제공

### **/proc/net**

- 네트워크 프로토콜들의 상태에 대한 정보를 제공

### **/proc/self**

- 이 디렉토리를 보는 프로그램 자신의 프로세스 디렉토리를 보여 준다

### **/proc/meminfo**

- 현재 시스템이 사용 중인 메모리의 사용량을 저장하고 있는 파일
- 실제 메모리와 가상 메모리에 대한 정보를 포함

### **/proc/modules**

- 현재 어떤 커널 모듈이 사용되고 있는지에 대한 정보를 제공

### **/proc/net**

- 네트워크 프로토콜들의 상태에 대한 정보 제공

### **/proc/self**

- 이 디렉토리를 보고 있는 프로그램 자신의 프로세스 디렉토리로 링크되어 있음
- 만약 서로 다른 2개의 프로세스가 /proc 디렉토리를 보고 있다면 두 프로세스는 서로 다른 링크를 보게 됨.
  - 이를 통하여 프로그램들이 자신의 프로세스 디렉토리를 쉽게 찾을 수 있음

### **/proc/stat**

- 시스템의 상태에 대한 다양한 정보를 제공

### **/proc/uptime**

- 시스템이 동작한 시간 정보를 제공

### **/proc/version**

- 커널의 버전 정보를 제공

### **/proc/stat**

- 시스템의 상태에 관한 다양한 정보 제공
- 예) 부팅 후 page fault가 몇 번 일어났는지에 대한 정보

### **/proc/uptime**

- 시스템이 얼마나 오래 동작했는지에 대한 정보를 저장

### **/proc/version**

- 현재 시스템이 사용 중인 커널 버전에 대한 정보를 저장하고 있는 파일

## 3장 저장장치

- 3.1 하드디스크
- 3.2 저장장치 포맷하기
- 3.3 파티션 활용하기
- 3.4 디스크 공간 할당
- 3.5 플로피 디스크
- 3.6 CD-ROM과 테이프

## Overview

각종 저장장치들의 종류를 알아보고,

저장장치를 포맷하고 파티션을 나누어 공간을 할당하는 방법을 소개한다.

## 3.1 하드디스크

### 하드디스크란?

- 자성체를 입힌 원판형 알루미늄 기판을 회전시키면서 자료를 저장하고 읽어 내도록 한 보조기억장치

### 하드디스크 내부구조

- 플레터
- 헤드
- 액추에이터
- 슬라이더
- 보이스코일
- 회로케이블

## 하드디스크

- 하드디스크는 자성체를 입힌 원판형 알루미늄 기판을 회전시키면서 자료를 저장하고 읽어내도록 한 보조기억장치이다.

## 하드디스크의 구조

- 디스크가 레코드판처럼 겹쳐져 있는 것으로, 디스크 위에는 트랙이라고 하는 동심원이 그려져 있다. 이 동심원 안에 데이터를 전자적으로 기록하게 된다. 헤드는 트랙에 정보를 기록하거나 읽어내는 역할을 한다.



## 하드디스크의 내부구조



\* 하드디스크 내부구조

## 3.2 포맷(Formatting)

### 포맷(Format)

- 디스크와 같은 자기 매체에 트랙과 섹터를 표시하는 과정
- 새 디스크를 자료기록이 가능하도록 그 형식을 지정해 주는 일

### 포맷의 두가지 종류

- 하이레벨 포맷(high-level formatting)
- 로우레벨 포맷(low-level formatting)

### 배드블록(bad blocks) & 배드섹터(bad sectors)

- 디스크에 물리적인 결함이 생긴 블록이나 섹터

## 포맷(Formatting)

- 디스크와 같은 자기 매체에 자료 기록이 가능하도록 그 형식을 지정해 주는 일

### 포맷의 두 가지 종류

- 하이레벨 포맷
  - 디스크의 일부 또는 전체에 파일시스템의 기본적인 틀을 구축하는 것
- 로우레벨 포맷
  - 자기 매체에 트랙과 섹터를 표시하는 과정
  - 일반적으로 논리적인 배드섹터를 삭제할 경우 사용

## 섹터(Sectors)

- 하드디스크나 콤팩트디스크 또는 디스켓에 데이터를 효율적으로 읽고 쓰게 하기 위해 만든 구조
- 하나의 섹터는 통상 512바이트 크기이다. 디스크에 물리적인 결함이 생겨 정보를 저장할 수 없게 되면 배드섹터(bad sector)라고 한다.

## 배드 섹터(bad sector)

- 하드디스크에 배드 섹터가 생겼다는 것은 하드디스크의 핵심부품인 플래터의 일정 부분이 충격을 받아 파손되었다는 것을 의미한다. 따라서 파손된 플래터 부분에는 데이터의 쓰기와 읽기가 불가능하다.
- 특히 한번 생긴 배드 섹터는 시간이 지날수록 그 범위가 점점 커져서 나중에는 아예 복구 불가능 상태로 만들어 버린다. 그 이유는 파손된 플래터의 파편 때문이다. 파손된 플래터의 파편이 플래터 표면 위를 지나는 헤드에 달라붙어서 헤드의 변형을 일으키는 것이다.

### 3.3 파티션(Partition)

#### 파티션

- 컴퓨터에서 디스크나 메모리 등의 저장 매체를 나누는 것

#### MBR (Master Boot Record)

- 운영체제가 어디에, 어떻게 위치해 있는지를 식별하여 컴퓨터의 주기억장치에 적재될 수 있도록 하기 위한 정보
- 파티션 섹터 또는 마스터 파티션 테이블이라고도 불림

#### MBR의 기능

- 부트 파티션을 파티션 테이블에서 찾는 것부터 시작하여 부트섹터의 실행코드의 전송을 중간에서 컨트롤하는 기능을 가짐

#### 파티션

- 컴퓨터에서 디스크나 메모리 등의 저장 매체를 나누는 것

#### MBR(Master Boot Record)

- 운영체제가 어디에, 어떻게 위치해 있는지를 식별하여 컴퓨터의 주기억장치에 적재될 수 있도록 하기 위한 정보
- 파티션 섹터 또는 마스터 파티션 테이블이라고도 불림
- 하드디스크가 포맷될 때 나뉘어지는 각 파티션의 위치에 관한 정보를 가짐
- 하드디스크나 디스켓의 첫 번째 섹터에 저장
- 각 파티션의 위치에 관한 정보를 소유
- 메모리에 적재될 운영체제가 저장되어 있는 파티션의 부트 섹터 레코드를 읽을 수 있는 프로그램 포함

## MBR의 역할

- 부트파티션(active partition)을 파티션 테이블에서 찾는다.
- 부트파티션의 시작섹터를 찾는다.
- 부트파티션 내의 부트섹터(boot record) 복사본을 메모리로 로드 시킨다.
- 부트섹터의 실행코드의 전송을 중간에서 컨트롤 한다.

※ MBR이 위에서 말한 기능을 완전히 끝마치지 못하면 다음 중 하나의 메시지를 화면에 표시하고 시스템이 정지하게 된다.

Invalid partition table. Error loading operating system. Missing operating system.
--

## 파티션의 구조

- 프라이머리 파티션, 익스텐디드 파티션

## 파티션의 형식

- MBR과 확장 파티션에 하나씩 있는 파티션의 정보를 저장하는 곳에는 각 파티션의 형식을 확인하는 1 byte 크기의 파일이 존재
- 이곳에는 해당 파티션에서 사용하는 운영체제를 확인하거나 그 파티션의 사용 목적 등이 담긴 정보들을 저장

## 파티션 나누기

- 리눅스 상에서 파티션을 나눌 경우 일반적으로 fdisk나 cfdisk를 사용

## 파티션의 구조

- 프라이머리(Primary)와 익스텐디드(extended)의 두 종류가 있다.
- 하나의 하드 디스크 드라이브에서는 최대 1개의 프라이머리 파티션과 3개의 익스텐디드 파티션을 만들 수 있다.
- 익스텐디드 파티션은 다시 여러 개의 논리적인 드라이브로 나눌 수 있다.
- 이렇게 한 하드디스크를 여러 개의 드라이브로 나누어 사용하면, 하나의 드라이브에 여러 운영체제를 동작시키기에 용이하며, 자료를 백업하기 용이하다는 장점이 있다.

## 파티션의 형식

- MBR에 하나 그리고 확장파티션에 하나씩 있는 파티션의 정보에는 각 파티션의 형식을 확인하는 1바이트가 파티션당 하나씩 있다.
- 그 1바이트로 파티션을 사용하고 있는 운영체제를 확인하거나, 운영체제가 어떤 목적으로 그 파티션을 사용하는지를 확인한다.

## 파티션 나누기

- 간단한 예제를 통하여 파티션을 나누는 것을 설명한다.

예제) 100M의 새 파티션을 생성하고 /data로 마운트 시켜라 (단, 파일시스템은 ext3로 설정하라)

```
[root@mail root]#fdisk /dev/hda
```

The number of cylinders for this disk is set to 9733.

There is nothing wrong with that, but this is larger than 1024,  
and could in certain setups cause problems with:

- 1) software that runs at boot time (e.g., old versions of LILO)
- 2) booting and partitioning software from other OSs  
(e.g., DOS FDISK, OS/2 FDISK)

Command (m for help): n

First cylinder (9004-9733, default 9004):

Using default value 9004

Last cylinder or +size or +sizeM or +sizeK (9004-9733, default 9733): +100m

Command (m for help): p

Disk /dev/hda: 80.0 GB, 80060424192 bytes

255 heads, 63 sectors/track, 9733 cylinders

Units = cylinders of 16065 \* 512 = 8225280 bytes

Device	Boot	Start	End	Blocks	Id	System
/dev/hda1	* 1		3824	30716248+	7	HPFS/NTFS
/dev/hda2		3825	5099	10241437+	83	Linux
/dev/hda3		5100	5230	1052257+	82	Linux swap
/dev/hda4		5231	9733	36170347+	5	Extended
/dev/hda5		5231	7055	14659281	fd	Linux raid autodetect
/dev/hda6		7056	8880	14659281	fd	Linux raid autodetect
/dev/hda7		8881	9003	987966	83	Linux
/dev/hda8		9004	9016	104391	83	Linux

## 2.파일시스템 지정

```
[root@mail root]#mkfs -t ext3 /dev/hda8
```

↳파일시스템지정

## 3. /data 마운트 시키기

```
[root@mail root]# mkdir /data
```

-> 마운트 시킬 /data 디렉토리를 생성한다.

```
[root@mail root]# mount -t ext3 /dev/hda7 /data
```

-> /dev/hda8 을 /data 디렉토리로 마운트 시킨다.

```
[root@mail root]# mount
/dev/hda2 on / type ext3 (rw)
none on /proc type proc (rw)
usbdevfs on /proc/bus/usb type usbdevfs (rw)
none on /dev/pts type devpts (rw,gid=5,mode=620)
none on /dev/shm type tmpfs (rw)
automount(pid4742) on /home type autofs (rw,fd=5,pgrp=4742,minproto=2,maxproto=3)
/dev/hda8 on /data type ext3 (rw)
```

↳ 마운트된 것을 확인할수 있다.

## 4.fstab 등록하기

-> 재부팅을 할 경우 마운트명령으로 마운트 시킨 건 다시 마운트 되지 않는다.

그러므로 /etc/fstab 에 등록시켜야한다.

```
[root@mail root]# cat /etc/fstab
LABEL=/ / ext3 defaults 1 1
none /dev/pts devpts gid=5,mode=620 0 0
none /proc proc defaults 0 0
none /dev/shm tmpfs defaults 0 0
/dev/hda3 swap swap defaults 0 0
/dev/cdrom /mnt/cdrom udf,iso9660 noauto,owner,kudzu,ro 0 0
/dev/fd0 /mnt/floppy auto noauto,owner,kudzu 0 0
```



```
#/dev/md0 /var ext3 defaults 0 0
#//203.247.50.229/project /wjdgns smbfs defaults,noauto 0 0
```

-> 원래 fstab 정보

```
[root@mail root]# cat /etc/fstab
LABEL=/ / ext3 defaults 1 1
none /dev/pts devpts gid=5,mode=620 0 0
none /proc proc defaults 0 0
none /dev/shm tmpfs defaults 0 0
/dev/hda3 swap swap defaults 0 0
/dev/cdrom /mnt/cdrom udf,iso9660 noauto,owner,kudzu,ro 0 0
/dev/fd0 /mnt/floppy auto noauto,owner,kudzu 0 0
#/dev/md0 /var ext3 defaults 0 0
#//203.247.50.229/project /wjdgns smbfs defaults,noauto 0 0
/dev/hda7 /data ext3 defaults 0 0
↳ 위와같이 등록시키면 부팅시마다 마운트된다.
```

※ /etc/fstab 이나 fdisk 안의 내용은 각 컴퓨터마다 다를 수 있다.

### 3.4 디스크 공간 할당하기

시스템이 돌아가는데 필요한 파일시스템을 하나의 파티션으로 만들고, 사용자들의 필요에 의해 저장되는 사용자 홈 디렉토리를 또 하나의 파티션으로 만들고, 그 외 스왑공간 등을 지정하는 것이 일반적이다.

#### 디스크 공간 할당의 이점

- 시스템이 망가지거나 파일시스템이 망가졌을 경우 해당 문제를 고치기에 용이함
- 사용자들의 홈 디렉토리만의 파티션으로 떼어 놓은 것은 백업을 더 쉽게 할 수 있게함

### 3.5 플로피 디스크

컴퓨터의 외부 기억장치로 사용되는 자기 디스크

자성체를 코팅한 원형의 마일러 기판으로, 특별한 재킷 안쪽에서 회전하게 되어 있음

#### 디스켓의 장점

- 한번 사용한 디스켓을 다시 사용할 수 있어서 입력매체에 소요되는 비용이 절감됨
- 저장된 데이터의 전환이나 수정이 용이
- 데이터를 판독하는 속도가 천공카드에 비해 훨씬 높다.

### 플로피 디스크

- 컴퓨터의 외부 기억장치로 사용되는 자기 디스크
- 자성체를 코팅한 원형의 마일러 기판으로, 특별한 재킷 안쪽에서 회전하게 되어 있다.

#### 플로피 디스켓의 장점

- 한번 사용한 디스켓을 다시 사용할 수 있어서 입력매체에 소요되는 비용이 절감된다.
- 저장된 데이터의 전환이나 수정이 용이하다.
- 데이터를 판독하는 속도가 천공카드에 비해 훨씬 높다.



### 3.6 CD-ROM & 테이프

#### CD-ROM 디스크

- 디지털 데이터를 저장하는 빛이 나는 기층(shiny under layer)을 가진 폴리카보네이트(bulletproof polycarbonate)로 이루어진 직경 120mm의 원형물체
- 정보는 광택이 나는 기층에 구멍(rough pits)으로 기억되고, 레이저 빛의 반사 정도를 측정해 정보를 읽는다.

#### 테이프

- 음악을 위해 사용되는 카세트와 비슷한 테이프를 사용
- 제작 비용이 비교적 저렴
- 저렴한 비용으로 많은 데이터 저장이 가능

#### CD-ROM 디스크

- 디지털 데이터를 저장하는 빛이 나는 기층(shiny under layer)을 가진 폴리카보네이트(bulletproof polycarbonate)로 이루어진 직경 120mm의 원형물체
- 정보는 광택이 나는 기층에 구멍(rough pits)으로 기억되고, 레이저 빛의 반사 정도를 측정해 정보를 읽는다.

#### 테이프

- 음악을 위해 사용되는 카세트와 비슷한 테이프를 사용
- 제작비용이 비교적 저렴함
- 저렴한 비용으로 많은 데이터 저장이 가능
- 테이프에 대한 자세한 사항은 ‘9장 백업과 복원’ 부분 참고

## 4장

# 파일시스템

- 4.1 파일시스템의 개념
- 4.2 파일의 종류
- 4.3 파일시스템의 구조
- 4.4 Mount와 Unmount
- 4.5 스왑공간 할당하기
- 4.6 파일시스템의 생성
- 4.7 파일시스템의 무결성 점검

## Overview

파일시스템이 무엇을 의미하는지, 파일의 종류에는 무엇이 있고 각각 파일은 어떤 특징이 있는지 알아보고 또한 파일시스템을 실질적으로 어떠한 명령어나 툴을 통해 생성하고 관리할 수 있는지 소개하였다.

## 4.1 파일시스템의 개념

### 파일시스템이란?

- 컴퓨터에서 데이터를 기록하기 위해서는 미리 하드디스크 드라이브에 데이터를 읽고, 쓰고, 찾기 위한 준비를 해두어야 하는데, 파일시스템은 그 준비의 규칙을 정리해 놓은 것으로서 파일에 이름은 붙이고, 저장이나 검색을 위해 파일을 어디에 위치시킬 것인지를 나타내는 체계를 의미
- 파일을 디렉토리에 저장시키며 새로 생긴 파일에 이름을 붙이는데, 파일명의 길이를 제한하기도 하고, 어떤 문자들이 사용될 수 있는지를 나타내기도 하며, 파일명 확장자의 길이를 제한함
- 디렉토리 구조를 통하여 파일까지 가는 경로를 설정하는 형식을 포함

### 파일시스템

- 운영체제가 파일을 시스템의 디스크 상에서 구성하는 방식

### 파일시스템의 구조

- 슈퍼블록(super block)
  - 파일시스템에 의존하는 정보를 가지며, 파일시스템의 크기 등과 같은 파일 시스템의 전체적인 정보를 가짐
- 아이노드(inode)
  - 파일의 이름을 제외한 해당 파일의 모든 정보를 가짐
  - 파일이름은 inode 번호와 함께 디렉토리 안에 저장됨
- 데이터 블록(data block)
  - inode에 포함되며, 파일에서 데이터를 저장하기 위해서 사용됨
- 디렉토리 블록(directory block)
  - 파일이름과 inode 번호를 저장하기 위해서 사용됨



- 간접 블록(indirection block)
  - 추가적인 데이터 블록을 위한 포인터들이 사용할 동작으로 할당되는 공간
- 홀(hole)
  - inode나 간접 블록안의 데이터 블록의 주소로 특별한 값을 저장함.
  - 파일시스템에 의해서 파일 안에 자리하게 되지만, 실질적으로 디스크 상에 공간은 할당되지 않음

## 표준화된 파일시스템 계층

### ■ 파일시스템의 계층 표준의 역사적 배경

- 초기 리눅스는 Minix 운영체제를 사용하였으며 이 시점에서 리눅스의 파일시스템은 Minix 파일시스템이었다.
- 그 후, 가상파일시스템(Virtual Filesystem) 계층이 개발되어 이로 인하여 리눅스는 다중 파일시스템을 지원하게 되었다.
- 1992년 4월 VFS가 커널에 포함되었고, 리눅스 버전 0.96b에서는 확장 파일시스템(extfs)이 새로운 파일시스템으로 추가되었다.
- 1993년 1월 ext2fs 파일시스템의 알파 버전이 발표되었다.
- 커널 2.4.15부터는 저널링 파일시스템이 추가되었다.

## 표준화된 파일시스템 계층

### ■ 파일시스템의 계층 표준의 역사적 배경

- 초기 리눅스는 Minix 운영체제를 사용하였으며 이 시점에서 리눅스의 파일시스템은 Minix 파일시스템이었다.
- 그 후, 가상 파일시스템(Virtual Filesystem) 계층이 개발되어 이로 인하여 리눅스는 다중 파일시스템을 지원하게 되었다.
  - 가상 파일시스템 계층 개발이 가져온 효과
    - 파일 관련 시스템 콜을 처리
    - I/O 기능을 물리적으로 파일시스템 코드로 전달
- 1992년 4월 VFS가 커널에 포함되었고, 리눅스 버전 0.96b에서는 확장 파일시스템(extfs)이 새로운 파일시스템으로 추가되었다.

- 확장 파일시스템의 문제점
  - 사용가능한 블록 수를 계산하는 것이라던가 연결 리스트로 구현된 inode는 최적의 성능을 보여주지 못함
  - 파일시스템을 사용할수록 그 리스트가 정렬되지 못하며 파일시스템이 조각남
- 1993년 1월 ext2fs 파일시스템의 알파 버전이 발표되었다.
- 커널 2.4.15부터는 저널링 파일시스템이 추가되었다
  - 리부트시에 파일시스템 점검을 위해 기다릴 필요가 없어짐

## ■ 파일시스템의 계층 표준의 역사적 배경 (Cont)

- 1993년 파일시스템 구조에 대한 일반적인 표준안을 만드는 것에 대한 논의를 시작
- 리눅스 운영체제에 한정된 FSSTND라는 파일시스템 계층 표준으로 형식화 됨
  - FSSTND 내용
    - 리눅스 시스템을 위한 표준 파일시스템 구조를 정의
    - 특정 형태의 파일 및 디렉토리가 속해야 할 곳을 지정
    - 시스템 파일에 포함되어야 할 내용을 정의
- 1995년 문제 해결의 범위를 유닉스 형태의 시스템으로 확대하여 파일시스템 계층 표준 (FHS)가 만들어 졌다.

## ■ 파일시스템의 계층 표준의 역사적 배경 (Cont)

- 1993년 파일시스템 구조에 대한 일반적인 표준안을 만드는 것에 대한 논의를 시작
  - 논의한 내용
    - 실행 파일 중 어떤 것이 /bin에 속할 것이며, 어떤 것이 /usr/bin에 속할 것인가에 대한 정의가 필요함
    - /etc 디렉토리 내에 실행 파일과 설정 파일이 마구 섞여있는 문제
    - 사이트 전체에 대한 설정 파일을 시스템 설정 파일로부터 분리하여, 이러한 파일들을 적절한 위치에서 공유하는 방법의 필요성
    - 파일시스템을 읽기 전용으로 마운트할 수 있도록 하기 위해서 변경될 수 있는 파일을 /usr 디렉토리에서 다른 곳으로 이동할 필요성
- 리눅스 운영체제에 한정된 FSSTND라는 파일시스템 계층 표준으로 형식화 됨
  - FSSTND 내용
    - 리눅스 시스템을 위한 표준 파일시스템 구조를 정의
    - 특정 형태의 파일 및 디렉토리가 속해야 할 곳을 지정
    - 시스템 파일에 포함되어야 할 내용을 정의

- 1995년 문제 해결의 범위를 유닉스 형태의 시스템으로 확대하여 파일시스템 계층 표준(FHS)가 만들어 졌다.

## 4.2 파일시스템의 종류

### 리눅스가 지원하는 다양한 파일시스템

- minix
- xiats
- msdos
- umsdos
- hpfs OS/2
- isofs
- iso9660
- nfs
- sysv
- ext2
- ext3 등

### 리눅스가 지원하는 다양한 파일시스템

- minix
  - 과거 미닉스에서 사용되어졌던 파일시스템으로 리눅스 파일 시스템 대부분의 기능을 제공하는 파일시스템
- xiafs
  - 미닉스의 제한이 있었던 파일 이름과 파일 시스템에 대한 제한을 보안한 미닉스 파일시스템의 수정버전
- msdos
  - 도스의 FAT 파일시스템과 호환을 지원하는 파일시스템
- hpfs OS/2
  - OS/2의 파일시스템. 현재는 읽기만 가능
- umsdos
  - MS-DOS 파일시스템을 리눅스 상에서도 긴 파일명과 소유자, 접근허가, 링크와 장치파일 등을 사용할 수 있도록 확장한 파일 시스템.
  - DOS 파일시스템이 마치 리눅스 파일시스템인 것처럼 보이도록 하는 기능을 제공

- iso9660
  - CD-ROM 표준 파일시스템
  
- nfs
  - 네트워크 파일시스템
  - 네트워크 상의 많은 컴퓨터들이 각각의 시스템에 가진 파일들을 쉽게 공유하기 위해 제공되는 상호간의 공유 파일시스템
  
- sysv
  - System V/386, Xenix , Cogerent 파일시스템
  
- ext
  - 리눅스 초기에 사용되던 파일시스템, ext2의 구버전
  
- ext2
  - ext의 여러 가지 문제점을 보완하여 나온 리눅스 전용 파일시스템

## 4.2.1 리눅스의 파일 종류를 확인하는 방법

### file 명령어

- 사용자가 파일의 종류를 확인할 때 사용하기 위해 만들어진 툴
- `file [option] [-f namefile] [-m magicfiles] file`
- magic파일(일반적으로 `/usr/share/magic` 파일이 기본)을 기본값으로 사용함
- `ls` 명령어보다 보다 자세한 파일의 종류를 확인할 수 있음

### file 명령어

- `file` 명령어는 magic 파일을 사용하고, 이 파일 속에는 `file` 명령어가 파일의 종류를 구분하는데 사용될 정보가 들어있음
- 명령행에서 `-f` 옵션으로 그 목록 파일을 지정하면 한번에 파일의 종류를 확인



## ls 명령어

- 파일 종류를 비롯한 파일에 대한 정보를 제공
- 형식

`ls [option] [file or directory name]`

- 옵션의 종류
  - 아래내용 참고

## ls 명령어

- `ls` 명령어에서 유래된 명령어로 파일 종류를 비롯한 파일에 대한 정보를 알고자할 때 사용
- 아무런 인자없이 `ls` 명령을 실행할 경우 단순히 파일명의 목록만을 보여준다.
- `-l` 옵션 (long listing을 의미)은 퍼미션의 문자열과 소유주, 파일의 크기, 최종수정 일시를 파일명과 함께 표시해준다.
- `ls` 명령어에서 많이 사용되는 인자들
  - `-a` : 숨겨진 파일들(점(.)으로 시작되는 파일)을 모두 출력한다.
  - `--color[=WHEN]` : 다양한 색을 사용하여 파일의 종류를 구분할 것인지 지정한다.
  - `-d, --directory` : 디렉토리의 목록만 출력한다.
  - `-G, --no-group` : 그룹 정보를 출력하지 않는다.
  - `-i, --inode` : 각 파일의 inode 정보를 출력한다.
  - `-n, --numeric-uid-gid` : 이름대신 숫자로 된 UID와 GID를 출력한다.
  - `-R, --recursive` : 모든 하위 디렉토리의 내용을 재귀적으로 출력한다.
  - `-s, --size` : 각 파일의 크기를 블록 단위로 표시하여 출력한다.
  - `-S` : 파일들을 크기 기준으로 정렬한다.
  - `-t` : 파일들의 수정 시간을 기준으로 정렬하여 출력한다.

## 4.2.2 ls 명령어를 사용하여 확인할 수 있는 파일의 종류

‘ls -l’의 결과에서 퍼미션 문자열의 첫 문자는 파일의 종류를 의미

### 1. 일반파일

- 퍼미션에 기호없음(-) 표시
- 실행파일, 텍스트 파일, 셸 스크립트, 이미지 파일 및 기타 다른 범주에 속하지 않는 모든 파일을 포함

### 일반파일

- 실행파일, 텍스트 파일, 셸 스크립트, 이미지 파일 및 기타 다른 범주에 속하지 않는 모든 파일을 포함.
- 대부분의 경우, 바이너리 또는 텍스트 파일을 일반 파일로 분류한다.
  - 텍스트 파일 : ASCII(또는 ANSI) 문자셋만을 사용한 파일
  - 바이너리 파일 : ASCII와 ANSI 문자셋 이외의 범위에 있는 바이트 값을 사용한 파일

## 2. 디렉토리

- 퍼미션에 'd' 표시
- 논리적으로 파일을 보관하는 장소
- 디렉토리 관련 명령어
  - 아래내용 참고

## 디렉토리

- 논리적으로 파일을 보관하는 장소
- 디렉토리를 생성하는 명령어 - **mkdir**
  - 형식
    - mkdir [options] [path] directory\_name
  - mkdir의 인자
    - -m, --mode=MODE : 새로운 디렉토리의 퍼미션 모드를 설정
    - -p, --parents : 디렉토리를 생성할 시 필요하다면 부모 디렉토리도 생성
    - -v, --verbose : 디렉토리 생성시에 자세한 메시지를 출력
- 디렉토리를 삭제하는 명령어 - **rmdir**
  - 형식
    - rmdir [options] [path] directory\_name

- rmdir의 인자

- -ignore-fail-on-non-empty : 디렉토리를 삭제하기 전에는 디렉토리의 내용을 먼저 삭제해야 하는데, 이 옵션을 사용하면 비어있지 않은 디렉토리의 삭제에 대한 실패 메시지를 출력하지 않는다. 비어있지 않은 디렉토리는 삭제되지 않지만 에러를 무시하고 다음 디렉토리를 계속 삭제 진행한다.
- -p, --parents : 지정한 디렉토리 삭제 후 부모 디렉토리가 비어있다면 그것도 삭제
- -v, --verbose : 디렉토리의 삭제를 완료하면 그에대한 검토를 위한 정보를 출력

### 3. 링크

- 퍼미션에 'l' 표시
- 하나의 파일을 여러 개의 파일명으로 액세스 할 수 있도록 해주는 것
- 링크의 종류
  - 하드 링크
    - 서로 다른 경로를 가진 여러 파일들이 하나의 inode를 가리키고 있는 것을 의미
  - 심볼릭 링크
    - 기본적으로 다른 파일을 가리키고 있는 파일

### 링크

- 하나의 파일을 여러 개의 파일명으로 액세스할 수 있도록 해주는 것
- 링크의 종류
  - 하드 링크
    - 서로 다른 경로를 가진 여러 파일들이 하나의 inode를 가리키고 있는 것을 의미
    - 하드링크를 가진 파일들은 모두 같은 파일시스템에 존재해야 한다.  
(∵ inode는 파일시스템마다 고유하기 때문에)
    - 하드 링크를 생성하는 명령어 - ln

```
ln [options] target [link_name]
ln [options] targets directory_name
```

- 심볼릭 링크
  - 기본적으로 다른 파일을 가리키고 있는 파일
  - 링크 파일은 서로 다른 inode를 할당받게 된다.
  - 심볼릭 링크를 사용할 때는 ln 명령어에 -s 옵션을 추가한다.

<example>

```
ln -s /home/user/example_fil /tmp/example
```

- 실제로 두 파일의 inode 값이 서로 다른지를 확인하려면 -i 옵션을 이용하면 된다.

<example>

```
ln -s /home/user/example_fil /tmp/example
```

## 4. 네임 파이프

- 퍼미션에 'p' 표시
- 파이프
  - 프로세스 간의 통신에 사용되는 특별한 파일
  - 네임파이프 생성
    - mkfifo나 mknod 명령어 이용

## 파이프

- 프로세스간의 통신에 사용되는 특별한 파일이다.
- FIFO(First In, First Out)이라고도 불린다.
- 첫 번째 프로세스는 쓰기모드로 열고 다른 프로세스는 읽기 모드로 연다.
- 프로세스와 파이프 사이의 모든 통신은 단일 방향이다.
- 네임 파이프를 생성하는 방법
  - mkfifo 명령어를 사용하거나 이를 사용할 수 없을 경우에 mknod를 사용한다.
  - example

```
$ mkfifo fifo_test  
$ mknod fifo_test p
```

## 5. 소켓

- 퍼미션에 's' 표시
- 두 개의 프로세스를 가상으로 연결하는 것
- 단일 방향과 양쪽 방향 전송이 가능

## 소켓

- 두 개의 프로세스를 가상으로 연결하는 것이다.
- 단일 방향과 양쪽 방향 전송이 가능하다.
- 표준 입출력 라이브러리는 `socket()`이라고 하는 소켓 라이브러리 함수를 가진다.
- 소켓 라이브러리 함수는 자신의 네트워크 주소와 포트 번호를 가진 파일 디스크립터를 생성하여 다른 프로세스가 접속할 수 있도록 해준다.
- 이러한 소켓으로 접속하는 방법은 '<http://211.123.34.45:10>' 식으로 접속하면 된다.



## 6. 블록/문자 디바이스 파일

- 퍼미션에 블록 디바이스 파일은 'b' 표시
- 퍼미션에 문자 디바이스 파일은 'c' 표시
- 디바이스
  - 디바이스 파일의 형태로 파일시스템에 존재하며 /dev/ 디렉토리에 저장되어 있다.
  - 디바이스를 액세스하는 하는 방식
    - 랜덤 액세스 블록 디바이스
    - 문자 디바이스

## 디바이스

- 디바이스 파일의 형태로 파일시스템에 존재하며 /dev /디렉토리에 저장되어있다.
- 리눅스 상에서 디바이스를 액세스하는 방식은 크게 두가지로 나뉜다.
  - 랜덤 액세스 블록 디바이스
    - 블록 디바이스 파일은 버퍼를 가진다.
    - 데이터는 지정된 조건이 만족할 때까지 버퍼에 쌓이면, 데이터는 디바이스로 보내진다.
    - SCSI, 플로피, IDE 디스크 드라이브 등에서 사용한다.
  - 문자 디바이스
    - 버퍼를 사용하지 않고 데이터를 디바이스 파일에서 받는 즉시 보내진다.
    - 테잎 드라이브, 모뎀, 시리얼 마우스 등에서 사용한다.
    - 대표적인 예로 /dev/console을 들 수 있다.

## 4.3 파일 시스템의 구조

### 슈퍼블록(super block)

- 파일 시스템에 의존하는 정보를 가지며 파일 시스템의 크기 등과 같은 파일 시스템의 전체적인 정보를 가진다.

### 아이노드(inode)

- 파일 이름을 제외한 해당 파일의 모든 정보를 가진다.

### 데이터 블록(data block)

- 파일에서 데이터를 저장하기 위해서 사용

### 슈퍼블록(super block)

- 파일 시스템에 의존하는 정보를 가지며 파일 시스템의 크기 등과 같은 파일 시스템의 전체적인 정보를 가진다.

### 아이노드(inode)

- 파일 이름을 제외한 해당 파일의 모든 정보를 가지고 있다.
- 파일 이름은 inode 번호와 함께 디렉토리 안에 저장된다.

### 데이터 블록(data block)

- 파일에서 데이터를 저장하기 위해서 사용된다.
- 데이터 블록은 inode에 포함된다. (inode가 몇 개의 데이터 블록을 포함한다.)

### 디렉토리 블록(directory block)

- 파일 이름과 인노드 번호를 저장하기 위해서 사용

### 간접 블록(indirection block)

- 추가적인 데이터 블록을 위한 포인터들이 사용할 동작으로 할당되는 공간

### 홀(hole)

- inode나 간접 블록 안의 데이터 블록의 주소로 특별한 값을 저장

### 디렉토리 블록(directory block)

- 파일 이름과 인노드 번호를 저장하기 위해서 사용된다.

### 간접 블록(Indirection block)

- 추가적인 데이터 블록을 위한 포인터들이 사용할 동작으로 할당되는 공간이다.
- 실제로 inode에는 적은 수의 데이터 블록을 가지고 있기 때문에, 더 많은 데이터 블록이 필요할 경우 이를 지정할 포인터가 필요하게 된다. 이때 포인터들이 사용할 동적인 블록이 간접 블록이다.

### 홀(hole)

- inode나 간접 블록 안의 데이터 블록의 주소로 특별한 값을 저장한다.
- 홀의 특징으로는 파일시스템에 의해서 파일 안에 자리하게 되지만, 이 홀을 위해 실질적으로 디스크 상에 공간은 할당되지 않는다.(즉, 0바이트가 파일 안에서 특정 공간을 차지하고 있다고 가정하는 것이다.)

### 4.3.1 inode와 디렉토리의 이해

#### inode와 디렉토리

- 각 파일은 생성될 때 고유한 inode를 할당 받음
- 파일명은 디렉토리명 항목 내에 그 inode 정보와 함께 저장
- 디렉토리 데이터의 inode는 해당 파일의 inode 위치를 찾는데 사용되고 inode에는 파일의 정보가 저장

#### inode와 디렉토리

- 파일명과 디렉토리 내의 위치를 제외한 파일의 모든 정보는 inode라고 불리는 데이터 구조에 저장된다.
- inode가 가지고 있는 정보는 파일잠금(locking)에 대한 정보, 액세스 모드, 파일 종류, 파일의 링크수, 소유주와 그룹ID, 바이트 단위의 파일크기, 액세스 및 변경시간, inode를 마지막 변경한 시간, 디스크상의 파일 블록에 대한 주소 등이 있다.
- 각 파일은 생성될 때 고유한 inode를 할당 받는다.
- 파일명은 디렉토리명 항목 내에 그 inode 정보와 함께 저장한다.
- 디렉토리 데이터의 inode는 해당 파일의 inode 위치를 찾는데 사용되고 inode에는 파일의 정보가 저장된다.
- ls 명령어와 같이 디렉토리 목록을 보는 명령어는 실제 파일의 데이터와는 무관하게 inode의 데이터만을 필요로 하므로 빨리 실행되는 것이다.
- 파일을 열면 inode 캐시 내의 해당 파일 항목을 잠금으로 설정하여 파일이 사용 중임을 나타낸다.
- 사용 중에 있지 않은 파일 및 inode 데이터 페이지는 만약 파일시스템에서 그 inode를 갱신하지 않았다면 메모리에 보관한다.

## 4.3.2 EXT2 파일시스템의 구조

### EXT2 inode

- 파일시스템의 가장 기본이 되는 단위
- EXT2 inode에 저장되는 정보
  - 모드(mode)
  - 소유자 정보(owner information)
  - 크기(size)
  - 타임 스탬프(time stamps)
  - 데이터 블록(data blocks)

### EXT2 inode

- EXT2 inode는 파일시스템의 가장 기본되는 단위이다.
- 또한 각 파일을 구분할 수 있는 고유번호를 가지고 파일의 데이터가 어느 블록에 어느 위치에 저장되어 있는지, 파일에 대한 접근 권한, 파일의 최종 수정시간, 그리고 파일의 종류 등의 정보를 inode 테이블에 저장한다.
- 이때 저장되는 정보는 모드, 소유자 정보, 크기, 타임 스탬프, 데이터 블록이다.
  - 모드(mode)에는 inode가 속한 파일에 대한 정보와 파일에 대한 접근 권한정보가 저장된다.
  - 소유자 정보(owner information)는 파일과 디렉토리에 대한 소유자와 그룹에 대한 식별자를 나타낸다.
  - 크기(size)는 파일의 크기 정보를 저장한다.
  - 타임 스탬프(timestamps)는 inode가 생성된 시간과 최종적으로 수정을 가한 시간에 대한 정보를 저장한다.

- 데이터 블록(data blocks)은 inode가 지정하고 있는 데이터 블록에 대한 포인터를 저장한다.

## EXT2 슈퍼블록

- 해당 파일 시스템의 기본적인 크기나 형태에 대한 정보를 저장
- 슈퍼블록에 저장되는 항목
  - 매직넘버
  - 개정 레벨
  - 파운트 횟수
  - 최대 마운트 횟수
  - 블록 그룹번호
  - 블록크기
  - 그룹 당 블록 수
  - 프리 블록

## EXT2 슈퍼블록

- 슈퍼블록은 해당 파일 시스템의 기본적인 크기나 형태에 대한 정보를 저장한다.
- 슈퍼블록에 저장되는 항목은 매직넘버, 개정 레벨, 파운트 횟수, 최대 마운트 횟수, 블록 그룹번호, 블록크기, 그룹 당 블록 수, 프리 블록 등이 있다.
  - 매직넘버(Magic Number)
    - 마운트하는 소프트웨어에게 EXT2 파일 시스템의 슈퍼 블록임을 확인하게 하는 값
  - 개정 레벨(Revision Level)
    - 메이저 레벨과 마이너 레벨로 구성된다.
    - 마운트 프로그램에서 어떤 특정한 버전에서만 지원되는 기능이 이 파일 시스템에서 지원되는지에 대한 확인을 위해 사용된다.
    - 기능 호환성 항목을 포함하여 마운트 프로그램이 해당 파일 시스템에서 안정적으로 사용할 수 있는 기능이 무엇인지를 판단할 수 있는 기준을 제공한다.
  - 마운트 횟수(Mount Count)와 최대 마운트 횟수(Maximum Mount Count)
    - 파일 시스템 전체를 검사할 필요가 있는지에 대한 여부를 확인한다.
    - 마운트가 실행 될 때마다 마운트 횟수는 1씩 증가하고 이 값이 최대 마운트 횟수에 도달하면, 시스템은 e2fsck를 실행하라는 메시지를 내보낸다.

- 블록 그룹 번호(Block Group Number)
  - 슈퍼 블록 복제본을 가지고 있는 블록 그룹의 번호를 나타낸다.
- 그룹 당 블록수(Blocks per Group)
  - 하나의 그룹에 속한 블록의 수를 나타낸다
- 프리블록(Free Block)
  - 시스템 내부적으로 존재하는 프리 블록의 수를 나타낸다.



## EXT2 그룹 기술자

- 블록 그룹에서 블록의 할당 상태를 나타내주는 비트맵
- 그룹 기술자에 저장되는 항목
  - 블록비트맵 (Block Bitmap)
  - inode 비트맵 (inode Bitmap)
  - inode 테이블 (inode Table)

## EXT2 그룹 기술자

- 블록 그룹에서 블록의 할당 상태를 나타내주는 비트맵이다.(블록의 수와 동일)
- 그룹 기술자에 저장되는 항목은 블록비트맵, inode 비트맵, inode 테이블이다.
  - 블록 비트맵(Block Bitmap)
    - 블록을 할당하거나 해제 할 경우 참고되는 정보이다.
  - inode 비트맵(inode Bipmap)
    - inode를 할당하거나 해제할 경우 참고되는 정보이다.
  - inode 테이블(inode Table)
    - 블록 그룹의 inode 테이블에서 시작 블록을 나타낸다.
- 이외에도 프리 블록 개수, 프리 inode 개수, 사용된 디렉토리 개수가 있다.
- 그룹 기술자는 연속적으로 나타나서 전체적으로 하나의 그룹 기술자 테이블을 형성하게 된다.

## EXT2 디렉토리

- 파일에 대한 접근 경로를 생성하고 저장하는 특별한 의미의 파일

## EXT2 디렉토리

- 파일에 대한 접근 경로를 생성하고 저장하는 특별한 의미의 파일로 취급된다.
- 엔트리의 리스트로 나타내어지며, 엔트리 리스트에 저장되는 정보는 inode, 이름 길이, 이름이다.
  - inode : 디렉토리 엔트리에 대항하는 inode
  - 이름 길이(Name Length) : 디렉토리 엔트리의 길이를 바이트로 나타낸 것
  - 이름(Name) : 디렉토리의 이름
- EXT2 파일시스템에서 모든 디렉토리에서 처음 두 엔트리는 항상 “.”과 “..”으로 시작한다.
  - “.”은 현재 디렉토리를 의미하고, “..”은 상위 디렉토리인 부모 디렉토리를 의미한다.

## 저널링 기술이 나오게 된 배경

- EXT2의 단점
  - 파일 시스템 복구 기능을 제공하나 시간이 많이 소요되며, 복구하는 동안 시스템을 사용하지 못한다.
- EXT3에 추가된 기능
  - 저널링 기술을 사용하여 위 단점을 보완

## 저널링 기술

- 데이터를 디스크에 쓰기 전에 로그에 데이터를 남겨 시스템의 비정상적인 셧다운에도 로그를 사용해 fsck보다 빠르고 안정적인 복구기능을 제공하는 기술

## EXT2의 파일 시스템 복구 기능

- 캐시에 저장되어 있는 데이터들을 디스크로 저장하는 도중 시스템이 다운되거나 문제가 발생할 경우 파일시스템이 손상된다.
- 이를 위해 EXT2에선 fsck(File System Check)라는 파일 시스템 복구 기능을 제공한다.
- fsck의 단점
  - 기존 EXT2 파일시스템의 경우에는 시스템이 동작을 멈추기 바로 직전에 파일시스템에 어떠한 수정을 가하고 있었는지 전혀 알 수 없다.
  - 이 이유로 인하여 시스템을 복구하기 위해선 fsck에 의해서 관리되는 슈퍼 블록, 비트맵, inode 등을 모두 검사해야 하기 때문에 시간이 오래 걸린다.
  - 복구하는 동안 시스템을 사용하지 못한다.
  - 슈퍼 블록에 마운트 횟수를 저장하는 영역이 있어서 마운트 횟수가 일정횟수 이상이 될 경우에도 자동적으로 fsck를 실행한다.

## EXT3의 파일 시스템 복구 기능

- 저널링 기능을 추가하여 시스템의 무결성과 뛰어난 복구 기능을 가짐

## 저널링(Journaling) 기술

- 데이터를 디스크에 쓰기 전에 로그에 데이터를 남겨 시스템의 비정상적인 셧다운에도 로그를 사용해 fsck보다 빠르고 안정적인 복구 기능을 제공하는 기술
- 파일 시스템에 수정을 하기 전에 우선 로그에 저장하는 이러한 파일 시스템의 특성 때문에 Log-Ging 파일시스템이라 부르기도 한다.

## 저널링 파일 시스템의 구조

- 로그영역
  - 파일 시스템에 대한 수정을 위한 로그 데이터를 저장
- 파일 시스템 영역
  - 관리 영역과 데이터 영역으로 구성
- 로그(Log) : 원형의 버퍼구조 형태로 순환적으로 데이터들이 저장
- 트랜잭션 : 저널링 파일 시스템에서 로그에 내용을 저장하는 수행단위
- 트랜잭션의 수행단계
  - 트랜잭션 할당 단계 → 트랜잭션 추가 단계 → 트랜잭션 위탁 단계

## 저널링 파일 시스템의 구조

- 저널링 파일 시스템의 구조는 로그영역과 일반적인 파일 시스템 영역으로 구분할 수 있다.
  - 로그영역 : 파일 시스템에 대한 수정을 위한 로그 데이터를 저장
  - 파일 시스템 영역 : 관리 영역과 데이터 영역으로 구성
- 저널링 파일 시스템은 일반적인 파일 시스템에 로그 영역을 추가 한 구조이므로, 저널링 기능을 제거하고 일반적인 파일 시스템과 같이 사용할 수도 있다.
- 로그(Log)
  - 원형의 버퍼 구조 형태로 순환적으로 데이터들이 저장된다.
  - 버퍼의 끝에 도달하면 데이터들은 다시 버퍼의 가장 앞쪽에 저장되는 구조
  - 로그상에는 데이터를 지정하는 포인터가 존재하여 데이터를 구분한다.
- 트랜잭션
  - 저널링 파일 시스템에서 로그에 내용을 저장하는 수행단위이다.
  - 저널링 파일 시스템은 디스크의 빈 블록 비트맵이나 inode가 포함된 블록에 대한 수정을 하기 전에 반드시 로그에 대한 트랜잭션을 수행한다.

■ 트랜잭션의 수행단계

1. 트랜잭션 할당 단계 : 수행할 파일 작성 트랜잭션에 대한 ID를 획득하기 위해서 새로운 트랜잭션 할당을 수행한다.
2. 트랜잭션 추가 단계 : 디스크상의 블록을 변경하기 전에 반드시 로그에 기록해야 하기 때문에 트랜잭션별로 할당되어 있는 관리 데이터와 링크되어 메모리에 저장된다.
3. 트랜잭션 위탁 단계 : 트랜잭션 추가 단계에서 만들어진 트랜잭션의 내용을 로그에 저장하는 동작을 수행한다.

## 4.4 Mount와 Unmount

### 4.4.1 파일시스템의 마운트

#### mount 명령어 사용하기

##### ■ 기본형식

- mount [-t fstype] [-o options] device dir
- mount 명령어 옵션 (아래내용 참고)

#### mount 명령어 사용하기

##### ■ 형식

- mount [-t fstype] [-o options] device dir
- fstype은 파일시스템의 형태를 말하며, ext2,vfat,jfs 등이 있다.
  - -t 옵션을 생략할 경우 리눅스가 해당 파일시스템의 형태를 감지한다.
- device는 파일시스템이 존재하는 리눅스의 디바이스 파일이다.
  - 예를들어 /dev/fd0는 첫 번째 플로피 디스크를 말한다.
- dir은 마운트 포인트를 나타내며 비어있는 디렉토리여야 한다.
  - 비어있지 않은 디렉토리를 사용할 수도 있지만, 그 경우에는 파일시스템이 마운트 되어 있는 동안 그 디렉토리에 들어 있던 내용을 액세스 할 수 없다.

## ■ 옵션

옵 션	지원하는 파일시스템	설 명
defaults	전 체	해당 파일시스템의 기본 옵션을 사용한다. 이 옵션은 주로 /etc/fstab/ 파일 내에서 사용되며, 기본 옵션임을 확실히 하기 위해 많이 사용된다.
loop	전 체	루프백 디바이스를 마운트할 때 사용한다. 파일을 디스크 파티션처럼 마운트할 수도 있다. 예를 들어, <code>mount -t vfat -o loop aaa.img /mnt/image</code> 명령은 aaa.img 파일을 디스크처럼 마운트한다.
auto 또는 noauto 전체	전 체	부팅 때나 <code>mount -a</code> 명령어를 실행했을 때 파일시스템을 자동으로 마운트하도록 또는 그것을 하지 않도록 한다. 기본값은 auto이지만 이동형 미디어에는 noauto 옵션을 사용하는 것이 좋다. /etc/fstab 파일에서 사용된다.
usr 또는 noauto 전체	전 체	일반 사용자가 파일시스템을 마운트하는 것을 허용하거나 또는 허용하지 않는다. 기본값은 nouser이지만 이동형 미디어에는 user 옵션을 설정하는 것이 적합하다. /etc/fstab 파일에 사용된다. 이 파일에 user 옵션이 포함된 경우에는 일반 사용자가 <code>mount /mountpoint</code> 명령을 입력해서 /mountpoint로 지정된 마운트 포인트에 디스크를 마운트할 수 있다.
Owner	전 체	사용자가 디바이스 파일의 소유자이어야 한다는 것을 제외하면 user 옵션과 유사하게 동작한다.
remount	전 체	파티션을 언마운트하지 않고도 마운트 옵션을 변경할 수 있도록 해준다. 이미 마운트되어 있는 파일시스템에 대하여 <code>mount</code> 명령어를 실행할 때 이 remount 옵션을 변경하려는 옵션과 함께 사용한다.
ro	전 체	파일시스템을 읽기전용으로 마운트하도록 지정한다.
rw	읽기와 쓰기가 허용되는 모든 파일시스템	파일시스템을 읽기/쓰기가 가능하도록 마운트한다.
uid=value	vfat,hpfs,ntfs,hfs 등 유닉스 형식의 파티션을 지원하지 않는 대부분의 파일시스템	파일시스템 내의 모든 파일에 대한 소유주를 설정한다. 예를 들면, uid=100으로 설정하면 리눅스의 사용자 ID가 100인 사용자로 파일들의 소유자가 설정된다.
gid=value	vfat,hpfs,ntfs,hfs 등 유닉스 형식의 파티션을 지원하지 않는 대부분의 파일시스템	uid=value와 유사하게 동작하며, 파일시스템 내의 모든 파일의 그룹을 설정한다.
conv=code	vfat,hpfs,ntfs,hfs 등 마이크로소프트나 애플OS에서 사용되는 대부분의 파일시스템	code값이 b 또는 binary로 설정되면 리눅스는 파일의 내용을 수정하지 않는다. code값이 t 또는 text로 설정되면 리눅스는 파일 내에 있는 리눅스 형식, DOS 형식, 또는 매킨토시 형식의 줄바꿈 문자를 자동 변환한다. code값이 a 또는 auto로 설정되면 리눅스는 파일이 이진 형식이 아닌 경우에 한하여 자동 변환을 수행한다.
nonumtail	vfat	리눅스에서 VFAT 파일시스템에 파일을 생성하는 경우에는 윈도우즈와 마찬가지로 파일명이 짧아지게 되는데(예. Sample~1.TXT) 이 옵션을 사용하면 가급적 숫자 형식의 꼬리를 추가하지 않고 파일명을 그대로 사용할 수 있게 된다.
eas=code	hpfs	code값이 no로 설정되면 리눅스는 OS/2의 확장 속성을 무시한다. code값이 ro로 설정되면 리눅스는 EA를 읽어서 리눅스의 소유권과 퍼미션 정보를 찾아내지만 새로운 EA를 만들 수는 없



		다. code 값을 rw로 설정하면 리눅스는 소유권과 퍼미션 정보를 EA에 저장할 수도 있으며, uid,gid 및 umask 옵션에 의해서 값을 바꿀 수도 있다.
case=code	hpfs, hfs	code값이 lower로 설정되면 리눅스는 파일명을 소문자로 변환하며, code값이 asis로 설정되면 리눅스는 파일명을 있는 그대로 유지한다.
fork=code	hfs	HFS 드라이버에서 매킨토시의 자원 포크를 처리하는 방식을 설정한다. 사용할 수 있는 옵션에는 cap, double 및 netatalk가 있다.
afpd	hfs	이 옵션은 Netatalk를 사용하여 파일시스템을 공유하는 경우에 사용한다.
norock	iso9660	ISO-9660 CD-ROM에서 Rock Ridge 확장을 사용하지 않는다.
nojoliet	iso9660	ISO-9660 CD-ROM에서 Joliet 확장을 사용하지 않는다.

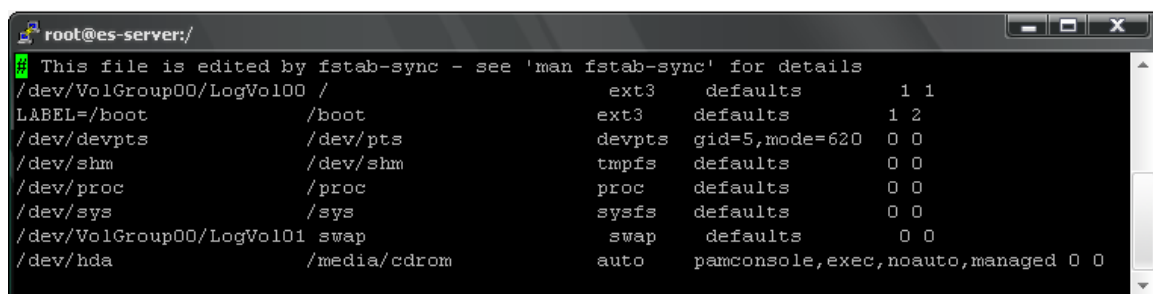
## 파일시스템 마운트 하기

### /etc/fstab 내에 마운트 할 항목을 추가하여 마운트 하기

- /etc/fstab 파일에는 마운트시의 설정값들이 정의되어 있다.
- 이 파일은 여러 개의 라인으로 구성되어 있는데 하나의 라인이 하나의 파일시스템에 해당한다.

### /etc/fstab 내에 마운트 할 항목을 추가하여 마운트 하기

- /etc/fstab 파일에는 마운트시의 설정값들이 정의되어 있다.
- 이 파일은 여러 개의 라인으로 구성되어 있는데 하나의 라인이 하나의 파일시스템에 해당한다.
- Example



```
root@es-server:/
This file is edited by fstab-sync - see 'man fstab-sync' for details
/dev/VolGroup00/LogVol100 /                ext3      defaults    1 1
LABEL=/boot              /boot      ext3      defaults    1 2
/dev/devpts               /dev/pts   devpts     gid=5,mode=620 0 0
/dev/shm                  /dev/shm   tmpfs      defaults    0 0
/dev/proc                 /proc      proc       defaults    0 0
/dev/sys                  /sys       sysfs      defaults    0 0
/dev/VolGroup00/LogVol01 swap                swap       defaults    0 0
/dev/hda                  /media/cdrom auto       pamconsole,exec,noauto,managed 0 0
```

## 파일시스템 언마운트 하기

### umount 명령어 사용하기

- 파일시스템의 사용을 중지하기 위해서 사용
- 형식
  - `umount [-a][-f][-t fstype] mountpoint | device`

### umount 명령어 사용하기

- 형식
  - `umount [-a][-f][-t fstype] mountpoint | device`
- 각 옵션의 의미
  - `-a` : `/etc/fstab` 내의 모든 파티션을 언마운트한다.
  - `-f` : 언마운트 동작을 강제로 실행한다.
  - `-t fstype` : 언마운트할 파일시스템의 형태를 지정해 준다.
  - `mountpoint` : 마운트되어 있는 장치의 마운트 포인트를 지정해 준다.
  - `device` : 파일시스템이 위치해 있는 디바이스를 지정해 준다.

## 4.5 스왑 공간 할당하기

### 스왑 공간 할당하기

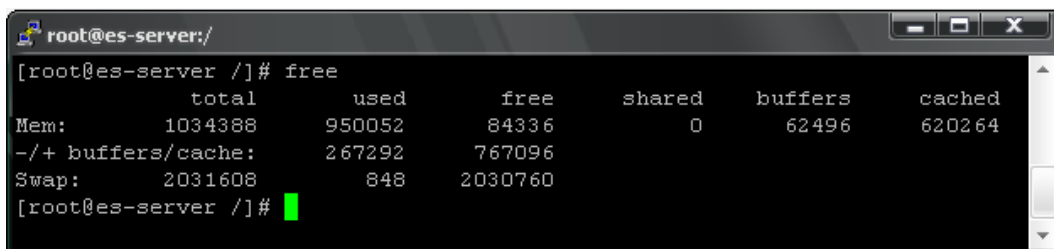
- 스왑 공간
  - 시스템 메모리(RAM)의 보조적인 수단으로 사용되는 디스크 공간
- 현재 시스템의 스왑 공간 확인하는 명령어 - free
- 스왑 공간을 추가하기 위한 절차
  - 아래 내용 참고

#### ※ 참고

☞ 컴퓨터가 어떠한 용도로 사용되느냐에 따라 스왑 공간의 크기가 달라질 수 있지만 일반적으로 시스템의 물리적 메모리 크기의 두배의 공간을 스왑 공간으로 잡는다.

### 스왑 공간 할당하기

- 스왑 공간 - 시스템 메모리(RAM)의 보조적인 수단으로 사용되는 디스크 공간
- 현재 시스템의 스왑 공간을 확인하는 명령어 - free
  - example



```
root@es-server:/  
[root@es-server /]# free  
              total        used         free       shared    buffers     cached  
Mem:           1034388      950052       84336          0         62496       620264  
-/+ buffers/cache:    267292       767096  
Swap:          2031608         848       2030760  
[root@es-server /]#
```

- Swap 라인을 살펴보면, total 칼럼의 숫자는 시스템에 할당된 전체 스왑 공간이 몇 킬로바이트인지를 알려주고, used와 free 칼럼의 숫자를 통해 스왑공간을 얼마나 사용하는지 알 수 있다.
- used 값이 total 값에 근접하기 시작하면 스왑 공간을 추가하는 것을 고려해야 한다.

■ 스왑 공간을 추가하는 절차

1. 적절한 크기의 디스크 공간을 잡는다.
  - 새로운 디스크 파티션을 생성하거나, 기본 파일시스템에서 적절한 크기의 빈 파일을 생성한다. 예를들면 'dd if=/dev/zero of=/swap bs=1024 count=n' 식으로 입력하여 n 킬로바이트 크기의 파일(/swap)을 생성한다.
2. 생성해 놓은 새로운 스왑 공간에 mkswap 명령어를 실행하여 해당 공간에 스왑 정보를 저장한다.
3. swapon 명령어를 사용하여 새로운 스왑 공간을 추가하고 free명령어를 통해 전체 스왑의 크기가 제대로 변경되었는지 확인한다.

## 4.6 파일시스템의 생성

파일시스템의 종류에 따라서 고유의 파일시스템 생성 유틸리티가 존재

파일시스템 생성 명령어 - mkfs

- 형식
  - mkfs [-t fsname] [option] device [size]
  - 각 옵션의 설명은 아래 내용 참고

### 파일시스템의 생성

- 리눅스에서는 파일시스템의 종류에 따라서 고유의 파일시스템 생성 유틸리티가 있다.
- 이러한 파일시스템 생성 유틸리티는 일반적으로 mkfs.fsname와 같은 이름을 가지며, 여기서 fsname은 파일시스템의 종류 코드를 의미한다.
  - 예를 들면, mkfs.ext2는 ext2 또는 ext3 파일시스템을 생성하는 유틸리티이다.

### 파일시스템 생성 명령어 - mkfs

- mkfs 명령어의 형식
  - mkfs [-t fsname] [options] device [size]
- mkfs 명령어의 각 옵션
  - -t fsname : 이 옵션은 파일시스템의 종류를 지정한다.  
mkfs는 mkfs.fsname을 호출한다.

- options : 각 파일시스템 고유의 옵션을 실제 파일시스템 생성 프로그램으로 전달할 수 있게 해준다.
  - -c : 디바이스의 비정상 블록 유무를 점검한다.
  - -l filename : 비정상 블록의 목록을 지정한 filename으로부터 읽는다.
  - -v : 파일시스템 생성 과정에 대한 추가적인 정보를 출력한다.
- device : 반드시 필요한 인자로서 프로그램이 파일시스템을 어떠한 디바이스에 생성할지를 알려준다.
- size : 지정한 크기만큼의 파일시스템을 생성하도록 하며, 블록의 단위는 보통 1024 바이트이다. 이 옵션을 생략하면 프로그램은 파티션이나 디바이스 전체의 크기에 맞도록 파일시스템을 생성한다.

## 4.7 파일시스템의 무결성 점검

### 파일시스템의 무결성 점검

- 파일시스템이 비정상인 경우에는 `fsck.fsname` 유틸리티를 사용하여 파일 시스템을 점검함

### 파일시스템의 무결성을 점검하는 유틸리티 - `fsck`

- 큰 문제가 발생하지 않는다면 사용자의 개입없이 모든 과정이 자동으로 진행됨
- 사용자의 개입이 필요한 경우엔 어떤 작업이 실패했는지 그에 대한 메시지를 출력된다.
  - ☞ 점검되는 파일시스템에 대해 자세히 모르는 경우에는 `fsck` 명령이 물어보는 각 질문에 대하여 기본값을 선택하는 좋다.

### 파일시스템의 무결성 점검

- 리눅스는 부트할 때마다 파일시스템이 정상적으로 언마운트되었는지 확인한다.
- 파일시스템이 비정상인 경우에는 `fsck.fsname` 유틸리티를 사용하여 파일시스템을 점검한다.
- 파일시스템 무결성 점검의 필요성
  - `ext2fs`에서 파일시스템의 점검은 매우 중요하다. 왜냐하면, 이전에 정상적으로 언마운트되지 않아서 일관성이 깨진 상태의 `ext2fs`를 그대로 사용하면 파일의 손실 또는 손상을 초래할 수도 있다.
- 파일시스템 점검 과정은 큰 문제가 발생하지 않는다면 사용자의 개입없이 모든 과정이 자동으로 진행된다.
- 가끔 파일시스템의 점검과정에서 사용자의 개입이 필요한 경우가 있다. 그런 경우에는 어떤 작업이 실패했다는 메시지가 출력되며, 시스템 관리자가 파일시스템을 수동으로 점검해야 한다.



## 5장 프로세스

- 5.1 프로세스의 개념
- 5.2 프로세스의 종류
- 5.3 부모 프로세스와 자식 프로세스
- 5.4 Init 프로세스
- 5.5 프로세스의 관리

## Overview

프로세스의 개념을 이해하고, 프로세스의 종류에 대해 알아본 뒤  
프로세스 중에서 가장 중요한 Init 프로세스에 대해 알아보고,  
마지막으로 프로세스들을 어떻게 관리하는지에 대하여 알아보자.

## 5.1 프로세스의 개념

### 프로세스

- 컴퓨터의 CPU에서 실행되는 모든 프로그램
- 하나의 명령어가 여러 개의 프로세스로 실행될 수 있음
- 리눅스에서는 여러 개의 프로세스가 항상 실행되고 있음

### 프로세스에 대한 시스템 관리자의 역할

- 시스템의 프로세스들이 정상적으로 실행될 수 있도록 운영
- 일부 프로세스는 특정 시간에 실행되도록 관리
- 높은 우선순위를 가진 프로세스가 CPU를 많이 사용할 수 있도록 관리
- 시스템의 모든 프로세스가 정상적으로 수행될 수 있도록 충분한 시스템 자원을 확보

### 프로세스란?

- 컴퓨터의 CPU에서 실행되는 모든 프로그램

### 프로세스의 특징

- 멀티태스킹을 지원한다.(뒷장에서 설명)
- 하나의 명령어가 여러 개의 프로세스로 실행될 수 있다.
- 리눅스에서는 여러 개의 프로세스가 항상 실행되고 있다.

### 프로세스에 대한 시스템 관리자의 역할

- 시스템의 프로세스들이 정상적으로 실행될 수 있도록 운영해야 한다.
- 일부 프로세스는 특정 시간에 실행되도록 관리해야 된다.
- 높은 우선순위를 가진 프로세스가 CPU를 많이 사용할 수 있도록 관리를 해주어야 한다.
- 시스템의 모든 프로세스가 정상적으로 수행될 수 있도록 충분한 시스템 자원을 확보하는 것이 중요하다.

## 멀티태스킹의 개념

- 한 사람의 사용자가 한 대의 컴퓨터로 2가지 이상의 작업을 동시에 처리하거나 프로그램들을 동시에 구동시키는 것
- 멀티프로세싱 시스템에서는 동시에 여러 개의 프로세스가 메모리에 상주하며, 실행 중인 프로세스가 중지되면 CPU 자원은 다른 프로세스를 실행하는데 사용된다.
- 이러한 방식으로 다수의 프로세스(태스크)가 CPU를 공유하며 이것을 멀티태스킹이라고 한다

## 멀티태스킹의 개념

- 한 사람의 사용자가 한 대의 컴퓨터로 2가지 이상의 작업을 동시에 처리하거나 프로그램들을 동시에 구동시키는 것

## 리눅스 시스템에서의 멀티태스킹

- 리눅스는 다중 처리 방식의 운영체제이며 운영체제는 CPU시간을 다양한 프로세스에게 적절히 분배해 주어야 한다.
- CPU의 개수보다 더 많은 프로세스가 존재한다면 일부 프로세스들은 CPU를 사용할 수 있을 때까지 대기해야 한다.
- 프로세스는 강제로 수행이 될 때까지 실행을 계속하고, 수행이 중지되는 경우는 보통 필요한 시스템 자원의 사용이 가능하지 않을 때이다.
- 멀티프로세싱 시스템에서는 동시에 여러 개의 프로세스가 메모리에 상주하며, 실행 중인 프로세스가 중지되면 CPU 자원은 다른 프로세스를 실행하는데 사용된다.

- 이러한 방식으로 다수의 프로세스(태스크)가 CPU를 공유하며 이것을 멀티태스킹이라고 한다.

## 5.2 프로세스의 종류

대화형 프로세스와 배치 프로세스와 데몬 프로세스로 나뉜다.

### 대화형 프로세스(사용자 프로세스)

- 사용자와 정보를 주고받으며 실행되는 프로세스

### 대화형 프로세스

- 사용자 프로세스라고도 불림
- 시스템 사용자들은 셸을 통해 프로세스를 수행시킬 수 있다.
- 대화형 프로세스 제어 또는 작업제어(job control)는 프로세스를 포그라운드나 백그라운드로 전환하는 역할을 하고, 포그라운드나 백그라운드에서 실행을 계속 할 수 있도록 제어한다.

## 배치 프로세스

- 일련의 작업을 몰아서 특정 시각에 실행시키는 것

## 배치 프로세스

- 일련의 작업을 몰아서 특정 시각에 실행시키는 것
- 일반적으로 터미널과의 입출력 교류가 없다.
- 미뤄두었던 작업들을 시스템의 사용률이 낮을 때 처리하는데 유용하다.



## 데몬 프로세스

- 특정 서비스를 위해 백그라운드 상태에서 계속 실행되는 서버 프로세스

## 데몬 프로세스

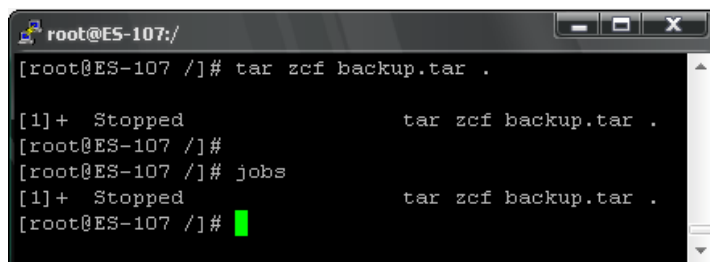
- 특정 서비스를 위해 백그라운드 상태에서 계속 실행되는 서버 프로세스
- 다른 데몬들에게 할당된 포트들을 감시하는 특별 한 용도의 데몬도 존재한다.
  - 오래전부터 inetd라는 데몬이 이러한 용도로 사용되었고, 최근엔 대신 xinetd라는 것을 많이 사용하고 있다. 이런 데몬들을 TCP/IP 슈퍼서버라고 한다.
  - 이것과 관련된 내용은 뒤에 네트워크 장에서 상세히 설명한다.

## 대화형 프로세스 관련 명령어

- 수행 중인 프로세스를 백그라운드로 전환시키기
  - 'CTRL+Z'로 전환
- 프로세스의 상태 확인하기
  - 'jobs' 명령어 사용
- 백그라운드로 전환된 프로세스를 포그라운드 상태로 전환하기
  - 'fg' 명령어 사용
- 명령어 처음부터 백그라운드로 실행하기
  - 명령행 제일 끝에 '&' 기호 추가

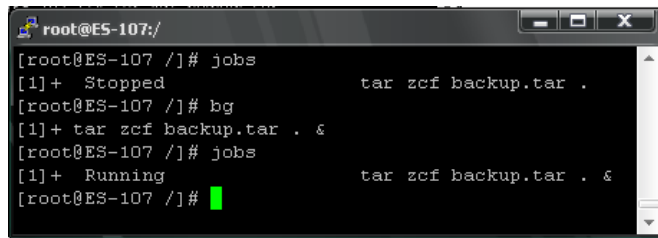
## 대화형 프로세스 관련 명령어

- example
  - tar 명령어를 이용하여 / 디렉토리 내용을 압축하고 있는 작업을 백그라운드로 전환하고 jobs 유틸리티를 사용하여 백그라운드 프로세스의 목록을 확인한다.



```
root@ES-107:/  
[root@ES-107 /]# tar zcf backup.tar .  
[1]+  Stopped                  tar zcf backup.tar .  
[root@ES-107 /]#  
[root@ES-107 /]# jobs  
[1]+  Stopped                  tar zcf backup.tar .  
[root@ES-107 /]#
```

- 위에서 일시 중지된 프로세스를 백그라운드 상태에서 계속 실행되게 하려면, bg 명령어를 사용한다. bg명령어를 사용한 후에 jobs 유틸리티로 프로세스를 확인



```
root@ES-107:/  
[root@ES-107 /]# jobs  
[1]+  Stopped                  tar zcf backup.tar .  
[root@ES-107 /]# bg  
[1]+  tar zcf backup.tar . &  
[root@ES-107 /]# jobs  
[1]+  Running                  tar zcf backup.tar . &  
[root@ES-107 /]#
```

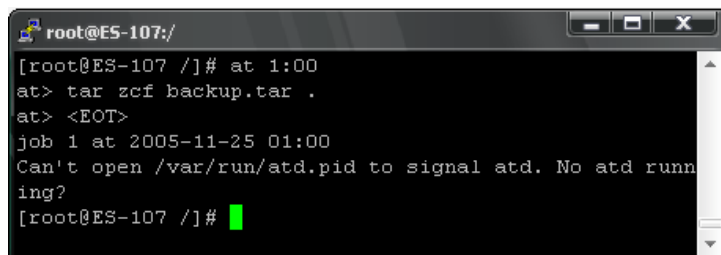
- 백그라운드 상태에서 작업 중인 프로세스를 다시 포그라운드 상태로 전환할 경우, fg 명령어를 사용하면 된다.

## 배치 프로세스 관련 명령어

- 특정시간에 특정 명령어를 실행시키는 명령어
  - 'at' 명령어
- 예정된 작업을 삭제하는 명령어
  - 'atrm' 명령어
- 예정된 작업을 확인하는 명령어
  - 'atq' 명령어
- 시스템의 사용량이 낮을 때에만 명령어를 실행하는 명령어
  - 'batch' 명령어

## 배치프로세스 관련 명령어

- example
  - 새벽 1시에 루트 디렉토리 전체를 백업을 하고 싶을 경우 아래 예와 같이 at 명령어를 사용하면 간단히 해결할 수 있다.
  - at 명령어 뒤에 작업이 진행될 시간을 적고 at> 프롬프트 상에서 그 시간에 할 작업을 지정하여 준 뒤, Ctrl+D 키를 이용하여 저장한다.



```
root@ES-107:/  
[root@ES-107 /]# at 1:00  
at> tar zcf backup.tar .  
at> <EOT>  
job 1 at 2005-11-25 01:00  
Can't open /var/run/atd.pid to signal atd. No atd running?  
[root@ES-107 /]#
```

- at 명령어는 시간만 지정할 수 있는 것이 아니라 명령어 뒤에 12/31/05 형식으로 날짜 부분을 추가하여 날짜까지 지정해 줄 수 있다.
- atrm 명령어 : 예정된 작업을 삭제하는 명령어
- atq 명령어 : 예정된 작업을 확인하는 명령어
- batch 명령어 : 시스템의 사용량이 낮을 때에만 명령어를 실행하는 명령어

## 5.3 부모 프로세스와 자식 프로세스

### 부모 프로세스의 특징

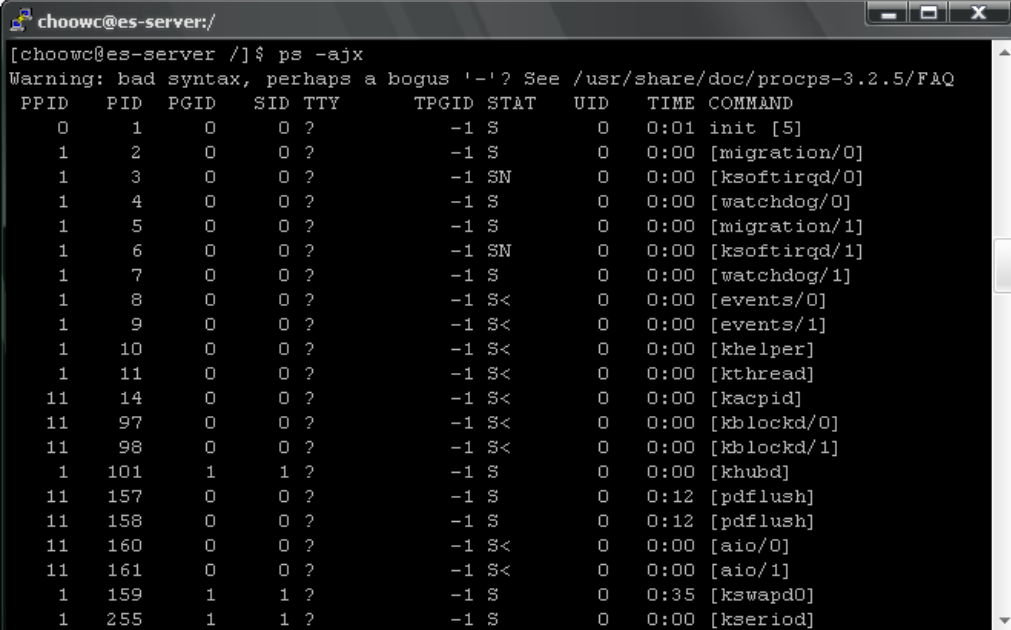
- 여러 개의 자식 프로세스들을 실행해서 다수의 작은 작업들을 동시에 처리할 수 있다. - 포크(fork)

### 부모 프로세스와 자식 프로세스의 관계

- 부모 프로세스와 자식 프로세스의 관계를 알 수 있는 방법
  - 각 프로세스의 PID와 PPID 필드를 살펴보면 알 수 있다

### 각 프로세스의 관계를 확인하는 방법

- Example



```
choowc@es-server:/[choowc@es-server /]$ ps -ajx
Warning: bad syntax, perhaps a bogus '-'? See /usr/share/doc/procps-3.2.5/FAQ
  PPID   PID  PGID   SID TTY      TPGID  STAT   UID    TIME  COMMAND
    0     1    0      0 ?        -1 S      0    0:01  init [5]
    1     2    0      0 ?        -1 S      0    0:00  [migration/0]
    1     3    0      0 ?        -1 SN     0    0:00  [ksoftirqd/0]
    1     4    0      0 ?        -1 S      0    0:00  [watchdog/0]
    1     5    0      0 ?        -1 S      0    0:00  [migration/1]
    1     6    0      0 ?        -1 SN     0    0:00  [ksoftirqd/1]
    1     7    0      0 ?        -1 S      0    0:00  [watchdog/1]
    1     8    0      0 ?        -1 S<     0    0:00  [events/0]
    1     9    0      0 ?        -1 S<     0    0:00  [events/1]
    1    10    0      0 ?        -1 S<     0    0:00  [khelper]
    1    11    0      0 ?        -1 S<     0    0:00  [kthread]
   11    14    0      0 ?        -1 S<     0    0:00  [kacpid]
   11    97    0      0 ?        -1 S<     0    0:00  [kblockd/0]
   11    98    0      0 ?        -1 S<     0    0:00  [kblockd/1]
    1   101    1      1 ?        -1 S      0    0:00  [khubd]
   11   157    0      0 ?        -1 S      0    0:12  [pdflush]
   11   158    0      0 ?        -1 S      0    0:12  [pdflush]
   11   160    0      0 ?        -1 S<     0    0:00  [aio/0]
   11   161    0      0 ?        -1 S<     0    0:00  [aio/1]
    1   159    1      1 ?        -1 S      0    0:35  [kswapd0]
    1   255    1      1 ?        -1 S      0    0:00  [kseriod]
```

- `ps -ajx` 명령어로 확인할 수 있다.
- 위의 예에서 `init` 프로세스를 살펴보면 PPID가 0 PID가 1인 것을 확인할 수 있다.
- `migration`, `ksoftirqd`, `watchdog`, `envents`, `khelper` 등 PPID가 1인 프로세스들은 모두 `init` 프로세스의 자식 프로세스가 됨을 알 수 있다.

## 5.4 Init 프로세스

### Init 프로세스

- 리눅스에서 가장 중요한 프로세스
- 리눅스 시스템에서 실행되는 첫 번째 프로세스
- PID가 1로 할당됨
  
- 역할
  - 파일시스템의 점검과 마운트
  - 필요한 데몬들의 시작
  - 시스템 구성에 필요한 기타의 것들을 포함하는 부트의 나머지 과정을 제어
  - getty를 띄워서 사용자들이 로그인 할 수 있게 해준다.

### Init 프로세스

- 리눅스 시스템에서 Init이 첫 번째 프로세스인 이유
  - 커널이 자신의 부팅을 진행할 때, 부트 프로세스로서 커널이 마지막으로 해야할 일은 Init를 실행시키는 것이다. 즉, 사용자 레벨의 프로세스인 init를 실행시킴으로 해서, 커널은 부트 프로세스로서의 역할을 마치게 된다. 그래서, init는 언제나 첫번째 프로세스가 되는 것이다(따라서, init의 프로세스 번호도 언제나 1이 된다).
  
- 부팅이 정상적으로 진행되었을 때 Init의 역할
  - init는 사용자들이 로그 아웃한 터미널에 다른 사용자들이 다시 로그인 할 수 있도록 getty를 재실행시킨다.
  - init는 고아 프로세스들을 거두어 양자로 삼는다.
    - 이 말의 의미는 한 프로세스가 자식 프로세스를 생성하고서 자식 프로세스보다 자신이 먼저 죽었을 경우를 의미하는 것으로서 이때 이 자식 프로세스의 부모 프로세스는 자동적으로 init 프로세스가 된다.

- getty를 띄워서 사용자들이 로그인할 수 있게 한다.
  - getty를 실행하기 위한 init의 설정 파일은 /etc/inittab 이다.



## 5.5 프로세스의 관리

시스템에서 수행 중인 프로세스에 대한 정보 확인

ps 명령어 사용하기

- ps -e 명령어
- ps -au 명령어
- ps -aef 명령어

ps 명령어의 사용

- ps -e 명령어
  - 프로세스의 PID를 확인하여 특정 프로세스를 중지하거나 재시작하기에 용이
- ps -au 명령어
  - 현재 터미널의 모든 프로세스들을 사용자 정보가 우선된 형태로 출력
- ps -aef 명령어
  - 특정 대문에 대한 정보를 찾을 때 용이

```
choowc@es-server:/
top - 14:40:00 up 4 days, 13 min, 13 users, load average: 0.15, 0.04, 0.01
Tasks: 155 total, 3 running, 152 sleeping, 0 stopped, 0 zombie
Cpu(s): 42.5% us, 8.6% sy, 0.0% ni, 45.8% id, 2.5% wa, 0.2% hi, 0.5% si
Mem: 1034388k total, 990316k used, 44072k free, 56896k buffers
Swap: 2031608k total, 508076k used, 1523532k free, 232012k cached

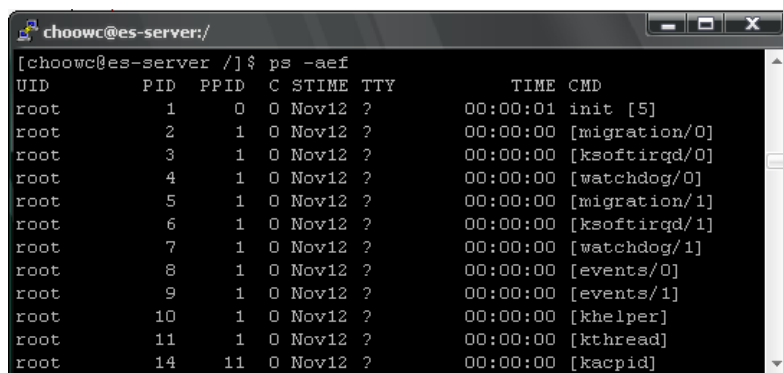
  PID USER      PR  NI  VIRT  RES  SHR  S %CPU  %MEM    TIME+  COMMAND
 25825 vicente   15   0 8988 3232 2004  S  2.0   0.3   0:00.69  xterm
 4869  vicente   16   0 5720 1748  764  S  1.3   0.2   0:00.04  make
 4878  vicente   19   0 5724 1696  764  S  1.0   0.2   0:00.03  make
 4882  vicente   20   0 5692 1652  724  R  1.0   0.2   0:00.03  make
 476   root      15   0    0    0    0   S  0.3   0.0   0:26.00  kjournald
27963 vicente   15   0 47392 12m 8248  S  0.3   1.2   0:01.88  gnome-terminal
4279  choowc    16   0 2020 1020  784  R  0.3   0.1   0:00.27  top
```

## top 명령어의 사용

- /proc 파일시스템을 사용
- 실행 중인 모든 프로세스에 대한 CPU 사용률과 메모리 사용률 등을 표시

## top 명령어의 사용

- /proc 파일시스템을 사용
- 실행 중인 모든 프로세스에 대한 CPU 사용률과 메모리 사용률 등을 표시
- Example



```
[choowc@es-server ~]$ ps -aef
UID          PID  PPID  C  STIME TTY          TIME CMD
root           1      0  0 Nov12 ?        00:00:01 init [5]
root           2      1  0 Nov12 ?        00:00:00 [migration/0]
root           3      1  0 Nov12 ?        00:00:00 [ksoftirqd/0]
root           4      1  0 Nov12 ?        00:00:00 [watchdog/0]
root           5      1  0 Nov12 ?        00:00:00 [migration/1]
root           6      1  0 Nov12 ?        00:00:00 [ksoftirqd/1]
root           7      1  0 Nov12 ?        00:00:00 [watchdog/1]
root           8      1  0 Nov12 ?        00:00:00 [events/0]
root           9      1  0 Nov12 ?        00:00:00 [events/1]
root          10      1  0 Nov12 ?        00:00:00 [khelper]
root          11      1  0 Nov12 ?        00:00:00 [kthread]
root          14     11  0 Nov12 ?        00:00:00 [kacpid]
```

## Kill 명령어를 이용하여 프로세스 중지/재시작 하기

- 프로세스 ID를 인자로 주고 kill 명령어를 실행하면 그 프로세스는 중지 시그널을 받게 된다.
- 중지 시그널을 받고서도 쉽게 죽지 않는 프로세스인 경우
  - -9 인자를 갖는 명령어를 사용
- 데몬을 재시작할 경우 kill 명령어의 프로세스 ID 앞에 -1 인자를 붙이거나 -HUP 인자를 붙여서 명령을 실행

## 스크립트를 사용하여 프로세스 중지/재시작 하기

- 데몬의 실행을 제어하는 스크립트 - /etc/rc.d/init.d 디렉토리 내에 존재
- 데몬의 실행을 제어하기 위한 다른 방법 - service 명령어 사용
- 특정 데몬을 중지하기 위해선 stop 옵션을 사용
- 현재 실행되고 있지 않은 데몬을 시작할 경우 start 옵션 사용
- 특정 데몬이 실행중인지 확인을 할 경우 status 옵션 사용
- 현재 실행중이 데몬 재시작 할 경우 restart 옵션 사용

## 6장

### 부팅과 섯다운

- 6.1 부팅과 셧다운의 개괄
- 6.2 부팅의 세부과정
- 6.3 셧다운의 세부과정
- 6.4 부트로더
- 6.5 사용자 초기화 파일관리

## Overview

일반적으로 컴퓨터를 그냥 스위치를 눌러 키고, 끄는 일련의 행위를 부팅 혹은 셧다운으로 오해하는 경우가 있는데 이 장을 통해서 부팅과 셧다운의 정확한 개념을 이해하고 각 세부과정을 알아보고, 부트로더는 무슨 역할을 하는지, 어떤 것들이 있는지 알아보고 마지막으로 사용자 초기화 파일관리에 필요한 것들이 무엇인지 서술하였다.

## 6.1 부팅과 셧다운 개괄

### 부팅과 셧다운 개괄부팅(booting)

- 컴퓨터 시스템에 전원을 넣고 운영체제를 불러 들이는 과정
- 부트스트래핑(bootstrapping)
  - 컴퓨터는 부트스트랩 로더(bootstrap loader)라는 작은 기계어 코드를 불러들이게 되는데, 이 프로그램은 다시 운영체제를 불러 들어서 그것을 작동시킨다. 이 과정을 부트스트래핑이라 한다.
- 부트스트래핑 과정이 필요한 이유
  - 컴퓨터가 맨 처음 읽어들이 수 있는 코드의 크기가 제한되어 있기 때문이다.
- 셧다운 : 파일시스템과 파일들을 보존하기 위해 안정된 절차로 종료해야 한다.

### 부팅과 셧다운 개괄

- 컴퓨터 시스템에 전원을 넣고 운영체제를 불러들이는 과정을 부팅이라고 한다.
- 운영체제가 부팅하기 위해선 부트스트래핑 과정을 거쳐야 한다.
  - 컴퓨터는 부트스트랩 로더(bootstrap loader)라는 작은 기계어 코드를 불러들이게 되는데, 이 프로그램은 다시 운영체제를 불러들여서 그것을 작동시킨다.
  - 이 과정을 부트스트래핑이라 한다.
  - 부트스트래핑 과정이 필요한 이유
    - 컴퓨터가 맨 처음 읽어들이 수 있는 코드의 크기가 제한되어 있기 때문이다.
- 리눅스 시스템을 셧다운 시키기 위해서는 먼저 모든 프로세서들을 종료하라는 지시가 필요하다. 그 다음 파일시스템과 스왑 공간을 언마운트 해야하며, 이 작업이 모두 마친 뒤에 콘솔창에 시스템 전원을 꺼도 좋다는 메시지를 보내야 한다.
- 이러한 셧다운 과정이 제대로 이루어 지지 않아서 파일시스템의 버퍼캐쉬가 제대로 비어지지 않는다면, 파일 안에 데이터가 없어지고 파일시스템이 불안해져서 결국 못쓰게 되는 상황을 이룰 수 있다.



## 6.2 부팅의 세부과정

### 리눅스의 기본적인 부트 절차

- ① 기본 입출력 시스템(BIOS, Basic Input/Output System)이 하드웨어 장치들을 시동하고 점검한다.
- ② BIOS는 1차 부트 로더로 제어권을 넘기고, 1차 부트로더는 파티션에 존재하는 2차 부트 로더를 찾는다.
- ③ 1차 부트 로더는 2차 부트 로더를 실행한다. 2차 부트 로더는 커널 이미지를 찾아서 실행한다.
- ④ 커널의 압축된 부분을 원래대로 복원해서 커널을 실행한다.
- ⑤ 커널은 /sbin/init 코드를 실행함으로써 init 프로세스를 시작한다.
- ⑥ init 프로세스는 getty 프로그램을 실행하고 그 외 다른 프로그램들도 실행하며 시스템이 중지될 때까지 그 프로그램들의 상태를 관찰한다.

### 리눅스의 기본적인 부트 절차

1. 일단 PC에 전원이 들어오면 BIOS는 시스템의 하드웨어에 문제가 없는지 여러 Test를 거친다.
2. BIOS는 Test 후 하드웨어의 문제점이 없다면 어느 디스크 드라이브로부터 부팅을 시작할 것인지 선택한다. (어떤 디스크로부터 부팅할 것인지 사용자가 BIOS Setup 메뉴에서 직접 설정할 수 있다.)
3. 선택된 드라이브의 첫 번째 섹터(부트섹터)를 읽어들인다.
  - 해당 드라이브에 여러 파티션이 존재할 경우 부트 섹터를 각각 따로 갖게 되는데, 이때 디스크의 첫 번째 섹터를 마스터 부트 레코드(MBR)이라 한다.
4. 마스터 부트 레코드의 프로그램이 어느 파티션이 활성화되어 있는지 알아본 후 그 파티션의 부트섹터를 읽는다.
  - 플로피 디스크로부터 리눅스를 부팅할 경우, 부트 섹터에 있는 작은 프로그램이 디스크의 첫째 몇 블록을 메모리의 특정 장소로 읽어들인다.
  - 리눅스 부트 플로피 디스크에는 파일시스템이 없기 때문에, 부팅과정의 간편화를 위해 커널은 연속된 섹터들 안에 그대로 저장된다.

5. 이러한 검사 후 그 파티션의 부트섹터를 읽어서 그 코드를 실행시킨다
6. 리눅스 커널이 메모리에 읽혀지게 되면 부팅이 본격적으로 시작된다.
  - 하드 디스크의 경우 각 파티션에는 파일시스템이 존재하므로 플로피 디스크처럼 단순히 순차적으로 디스크를 읽을 수 없으므로 이를 해결하기 위해 부트로더를 사용한다.

## 본격적인 부팅 과정

- ① 압축된 형태로 설치된 리눅스 커널 형태이므로 압축을 푼다.
- ② 커널은 어떠한 하드웨어가 장착되어 있는지 체크한다.
- ③ 적절한 장치 드라이버를 설정한다.
- ④ 커널은 루트 파일시스템을 마운트 한다.
- ⑤ 커널은 Init 프로그램을 백그라운드로 실행시킨다.
- ⑥ Init은 다중 사용자 모드로 전환되고, getty를 가상 콘솔과 시리얼 라인 터미널에 띄운다.
- ⑦ 이로써 모든 부팅 과정이 마치게 된다.

## 본격적인 부팅 과정

- ① 압축된 형태로 설치된 리눅스 커널 형태이므로 압축을 푼다.
- ② 커널은 어떠한 하드웨어가 장착되어 있는지 체크한다.
- ③ 적절한 장치 드라이버를 설정한다.
  - 이동안에 어떠한 하드웨어가 인식되었는지 메시지를 출력한다.
- ④ 커널은 루트 파일시스템을 마운트 한다.
  - 적합한 파일시스템 드라이버를 커널에 포함시키지 않았든지 하는 이유로, 파일시스템을 마운트하는 데 실패한다면 커널은 공황상태(panic)에 빠져든다.
- ⑤ 커널은 Init 프로그램을 백그라운드로 실행시킨다.
  - 최소한 몇가지 필수적인 백그라운드 데몬을 실행하도록 되어 있다.
- ⑥ Init은 다중 사용자 모드로 전환되고, getty를 가상 콘솔과 시리얼 라인 터미널에 띄운다.
- ⑦ 이로써 모든 부팅 과정이 마치게 된다.

## 셸다운의 세부과정

### 셸다운의 중요성

- 리눅스는 디스크에 쓰기를 바로하지 않고, 디스크 캐쉬를 거치기 때문에 리눅스 시스템을 제대로 셸다운하지 않을 경우 파일시스템이나 파일들이 손상될 수 있다.
- 많은 백그라운드 작업들이 멀티태스킹 환경에서 돌아가고 있기 때문에 데이터를 안전하게 저장하고 종료하는 것이 중요하다.

### 리눅스 시스템을 셸다운 시키는 명령어 - shutdown

- 단일 사용자일 경우 : `$shutdown -h now`
- 다중 사용자일 경우 : `$shutdown -h +time message` 방식
  - 모든 사용자가 안전하게 자신의 작업을 저장할 시간을 준다.

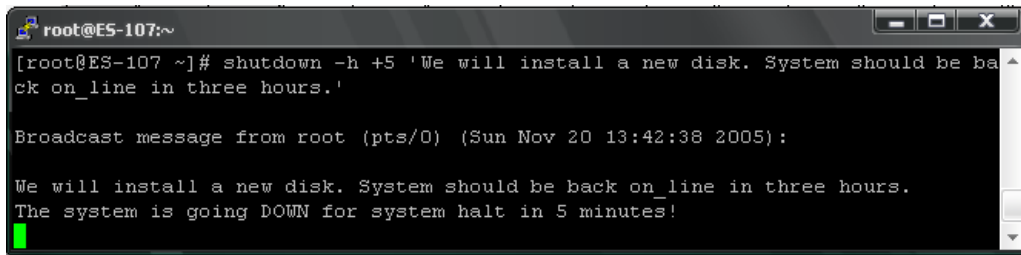
## 셸다운의 중요성

- 리눅스 시스템을 셸다운시킬때 적절한 절차를 밟아야 되는 이유
  - 리눅스는 디스크에 쓰기를 바로하지 않고, 디스크 캐쉬를 거치기 때문에 리눅스 시스템을 제대로 셸다운하지 않을 경우 파일시스템이나 파일들이 손상될 수 있다.
  - 많은 백그라운드 작업들이 멀티 태스킹 환경에서 돌아가고 있기 때문에 데이터를 안전하게 저장하고 종료하는 것은 중요하다.

## shutdown 명령어

- 단일 사용자인 경우 바로 자신의 작업하던 것을 저장하고 바로 종료하면 되기 때문에 시스템을 즉시 셸다운시키는 명령어(`shutdown -h now`)를 입력하면 된다.
- 다중 사용자인 경우 모든 사용자가 안전하게 자신의 작업을 저장하고 정리할 시간을 주고 이를 알려주고 셸다운시키기 위해 `shutdown -h +time message` 형태의 명령어를 사용한다.

- Example - 5분후에 시스템을 종료한다는 메시지를 해당 서버 사용자 모두에게 보낸다.



```
root@ES-107:~  
[root@ES-107 ~]# shutdown -h +5 'We will install a new disk. System should be back on_line in three hours.'  
  
Broadcast message from root (pts/0) (Sun Nov 20 13:42:38 2005):  
  
We will install a new disk. System should be back on_line in three hours.  
The system is going DOWN for system halt in 5 minutes!  
█
```

### 셸다운의 세부과정

- ① 루트를 제외한 모든 파일시스템이 언마운트 된다.
- ② 로그아웃을 하지 않은 사용자들의 프로세스들은 종료된다.
- ③ 데몬들이 종료된다.
- ④ 루트파일시스템이 언마운트된다.
- ⑤ init은 전원을 꺼도 좋다는 메시지를 화면에 뿌려준다.
- ⑥ 전원을 끈다.

### 리부팅

- 셸다운을 하고, 전원을 내린 뒤, 다시 전원을 올리는 과정
- 리부팅 방법 : shutdown -r 명령을 주거나 ctrl+alt+del 키 입력

## 6.4 부트로더

### 부트로더의 역할

- OS의 커널을 로드하고 몇몇 커널 파라미터를 커널에 넘겨주는 일을 한다

### 부트로더의 종류

- 시스템 커맨더(System Commander)
- NTLDR
- 리눅스 로더(LILO, Linux L0ader)
- GRUB(Grand Unified Bootloader) 등

## GRUB의 기능

- a.out 포맷과 ELF 포맷의 커널을 읽어들이 수 있다.
- Linux, FreeBSD, NetBSD, OpenBSD등 비-멀티부트 커널을 지원한다.
- 멀티플 모듈을 로드할 수 있다.
- 텍스트 형식의 설정 파일을 제공한다.
- 메뉴 인터페이스를 제공한다.
- 유연한 커맨드라인 인터페이스를 제공한다.
- BSD FFS, FAT16, FAT32, Minix, ext2 그리고 ReiserFS 파일시스템을 지원한다.
- gzip으로 압축된 파일을 다룰 수 있다.
- BIOS에서 인식되는 모든 장치에 액세스할 수 있다

## GRUB

- GRUB은 다음 위치에서 다운받을 수 있다.
  - <ftp://alpha.gnu.org/gnu/grub/>
- GRUB의 기능
  - a.out 포맷과 ELF 포맷의 커널을 읽어들이 수 있다.
  - Linux, FreeBSD, NetBSD, OpenBSD등 비-멀티부트 커널을 지원한다.
  - 멀티플 모듈을 로드할 수 있다.
  - 텍스트 형식의 설정 파일을 제공한다.
  - 메뉴 인터페이스를 제공한다.
  - 유연한 커맨드라인 인터페이스를 제공한다.
  - BSD FFS, FAT16, FAT32, Minix, ext2 그리고 ReiserFS 파일시스템을 지원한다.
  - gzip으로 압축된 파일을 다룰 수 있다.
  - BIOS에서 인식되는 모든 장치에 액세스할 수 있다.
- LILO와 비교했을 때 장점
  - 매번 구성을 변경할 때마다 설치 프로그램을 실행 필요가 없다.
  - 부트 프롬프트에서 시스템의 정보를 직접 확인할 수 있다.



## LILO의 기능

- 모든 부트 가능 파티션의 부트섹터에 존재하는 다수의 커널 이미지로 부트할 수 있다.
- 부트 가능 파티션으로는 윈도우즈 파티션등 어떤 운영체제도 선택하여 부트할 수 있다.
- 선택된 커널 이미지 및 그에 따른 initrd 이미지를 메모리로 읽어들이고 그 제어권을 커널로 넘겨준다.

## LILO 설정

- /etc/lilo.conf 파일을 사용하여 설정한다.

## LILO 기능

- 모든 부트 가능 파티션의 부트섹터에 존재하는 다수의 커널 이미지로 부트할 수 있다.
- 부트 가능 파티션으로는 윈도우즈 파티션등 어떤 운영체제도 선택하여 부트할 수 있다.
- 선택된 커널 이미지 및 그에 따른 initrd 이미지를 메모리로 읽어들이고 그 제어권을 커널로 넘겨준다.

## LILO 설정

- 각 운영체제들에 대하여 LILO가 인식할 수 있는 이미지를 만들어야 한다.
- 각 커널 이미지와 운영체제의 정보를 /etc/lilo.conf 파일에 추가하고, 해당 이미지를 대표할 수 있는 이름을 지정해 주어야 한다.
- lilo.conf 파일을 변경한 후, LILO 프로그램을 사용하여 MBR에 설치해야 한다.
  - 이 때 다음과 같은 명령어를 실행한다.
    - /sbin/lilo
  - LILO에는 다양한 옵션들이 존재 하는데 이는 BootPrompt-HOWTO 문서를 참고한다.
  - 구할 수 있는 곳
    - <http://www.linuxrx.com/HOWTO/sunsite-sources/BootPrompt-HOWTO.html>

## 6.5 사용자 초기화 파일관리

### 사용자 초기화 파일

- 각 사용자들마다 서로 다른 초기화 절차를 가진다.
- 시스템 관리자는 사용자들의 초기화 파일을 관리할 책임이 있다.

### 절차

- ① 사용자가 로그인하면 먼저 사용자 구성파일들이 읽어들여진다.
  - ② 첫 번째 읽혀져 반영되는 파일은 /etc/profile 이다.
  - ③ /etc/profile 스크립트가 완료된 후, 시스템은 사용자의 홈 디렉토리에서 .bash\_profile, .bash\_login, .profile 파일을 적혀있는 순서대로 찾아 읽어들인다.
- 사용자가 Bash 셸에서 로그아웃할 때는 .bash\_logout 파일이 읽혀진다.

### 사용자 초기화 파일

- 사용자가 로그인했을 때 초기화 되는 내용 사용자에게 따라 달라지며, 각 사용자들마다 서로 다른 초기화 절차를 가진다.
- 시스템 관리자는 사용자들의 초기화 파일을 관리할 책임을 가진다.
- ‘점’ 명령어를 사용하여 파일의 내용을 읽는다.

### 절차

- ① 사용자가 로그인하면 먼저 사용자 구성파일들이 읽어들여진다.
- ② 첫 번째 읽혀져 반영되는 파일은 /etc/profile 이다.
  - /etc/profile 파일은 시스템 전반에 걸친 환경변수들을 정의하고, 사용자들이 로그인할 때 실행해야 할 프로그램들을 지정하고 있다.
  - /etc/profile 파일은 /etc/profile.d 디렉토리 내의 모든 파일을 읽어들인다.
  - /etc/profile.d 디렉토리를 사용하는 이유
    - 새로운 RPM 패키지를 추가할 때 로그인 스크립트에 추가할 내용이 있다면, /etc/profile 파일의 내용을 수정하는 것보다는 /etc/profile.d 디렉토리에 파일을 추가하는 것이 편리하기 때문

- ③ /etc/profile 스크립트가 완료된 후, 시스템은 사용자의 홈 디렉토리에서 .bash\_profile, .bash\_login, .profile 파일을 적혀있는 순서대로 찾아 읽어들이는다.
- 일반적으로 .bash\_profile이 사용된다. 이 파일이 하는 일은 .bashrc 파일이 사용자의 홈 디렉토리에 존재한다면 그 파일을 읽어들이는 것이다. 그런 후, 몇 개의 환경변수들을 설정한다.
  - 사용자가 Bash 셸에서 로그아웃할 때는 .bash\_logout 파일이 읽혀진다

## 6.5 사용자 초기화 파일관리

### 사용자 초기화 파일

- 각 사용자들마다 서로 다른 초기화 절차를 가진다.
- 시스템 관리자는 사용자들의 초기화 파일을 관리할 책임이 있다.

### 사용자 초기화 파일

- 사용자가 로그인했을 때 초기화 되는 내용 사용자에게 따라 달라지며, 각 사용자들마다 서로 다른 초기화 절차를 가진다.
- 시스템 관리자는 사용자들의 초기화 파일을 관리할 책임을 가진다.
- ‘점’ 명령어를 사용하여 파일의 내용을 읽는다.

## 절차

- ① 사용자가 로그인하면 먼저 사용자 구성파일들이 읽어 들어진다.
  - ② 첫 번째 읽혀져 반영되는 파일은 `/etc/profile` 이다.
  - ③ `/etc/profile` 스크립트가 완료된 후, 시스템은 사용자의 홈 디렉토리에  
서 `.bash_profile`, `.bash_login`, `.profile` 파일을 적혀있는 순서대로  
찾아 읽어들인다.
- 사용자가 Bash 셸에서 로그아웃할 때는 `.bash_logout` 파일이 읽혀진다.

## 절차

- ① 사용자가 로그인하면 먼저 사용자 구성파일들이 읽어들여진다.
- ② 첫 번째 읽혀져 반영되는 파일은 `/etc/profile` 이다.
  - `/etc/profile` 파일은 시스템 전반에 걸친 환경변수들을 정의하고, 사용자들이 로그인할 때  
실행해야 할 프로그램들을 지정하고 있다.
  - `/etc/profile` 파일은 `/etc/profile.d` 디렉토리 내의 모든 파일을 읽어들인다.
  - `/etc/profile.d` 디렉토리를 사용하는 이유
    - 새로운 RPM 패키지를 추가할 때 로그인 스크립트에 추가할 내용이 있다면, `/etc/profile`  
파일의 내용을 수정하는 것보다는 `/etc/profile.d` 디렉토리에 파일을 추가하는 것이 편리  
하기 때문
- ③ `/etc/profile` 스크립트가 완료된 후, 시스템은 사용자의 홈 디렉토리에서 `.bash_profile`,  
`.bash_login`, `.profile` 파일을 적혀있는 순서대로 찾아 읽어들인다.
  - 일반적으로 `.bash_profile`이 사용된다. 이 파일이 하는 일은 `.bashrc` 파일이 사용자의 홈  
디렉토리에 존재한다면 그 파일을 읽어들이는 것이다. 그런 후, 몇 개의 환경변수들을 설정  
한다.
  - 사용자가 Bash 셸에서 로그아웃할 때는 `.bash_logout` 파일이 읽혀진다.

## 7장

# 사용자 계정의 관리

- 7.1 사용자 계정의 종류
- 7.2 윈도우 사용자와 윈도우 암호
- 7.3 사용자 계정의 관리
- 7.4 그룹
- 7.5 그룹의 관리

## Overview

사용자 계정의 종류를 알아보고 그러한 계정들을 어떻게 생성하고 삭제하고, 수정하는지에 대한 방법과 그룹에 대한 내용을 서술하고 그룹을 관리하는 방법을 서술하였다.



## 7.1 사용자 계정의 종류

### 일반적인 계정의 종류

- 모뎀을 통해 서버에 PPP 프로토콜로 접속해서 TCP/IP 네트워크를 액세스 하기 위한 계정
- 평범한 로그인 계정(소위 셸 계정)
- 메일만 사용하기 위한 메일 계정(POP, 가상POP, 또는 IMAP)
- FTP만 사용하기 위한 계정
- 이외에도 지금은 거의 이용되지 않고 한국에는 없는 오래된 프로토콜인 UUCP 네트워크 계정 등이 있다.

### 계정의 종류

- PPP 계정
- 셸 계정
- 메일 계정(POP, 가상 POP, IMAP)
- FTP만 사용하기 위한 계정 등

## 사용자의 홈 디렉토리가 필요하지 않는 계정

- PPP(Point-to Point Protocol) 계정
- POP(Post Office Protocol) 계정

## 사용자의 홈 디렉토리가 필요하지 않은 계정

- PPP 계정
  - PPP 사용자는 PPP 데몬 프로그램을 로그인 셸로 설정해야 한다.
  - PPP 접속이 연결되었을 때 사용자 인증이 이루어진다.
- POP 계정
  - POP 사용자의 MUA(Mail User Agent, 메일 클라이언트 프로그램)은 메일 서버 시스템을 접속해서 사용자 인증을 받는다.
- 이러한 계정들은 셸에 로그인을 하지 않기 때문에 사용자를 생성할 때 로그인 셸을 /bin/false로 설정한다.
  - telnet이나 콘솔에서 로그인을 시도하면 바로 세션이 종료되며 로그인이 실패

## /etc/passwd 파일을 이용한 사용자 정보 확인하기

- 각 필드의 구분은 콜론(:)으로 되어 있다.
- 각 필드의 내용 (순서대로)
  - 사용자명
  - 암호화된 암호
  - 사용자 ID
  - 기본 그룹
  - 사용자 정보
  - 사용자의 홈 디렉토리
  - 사용자의 로그인 셸

## /etc/passwd 파일을 이용한 사용자 정보 확인하기

- 각 필드의 구분은 콜론(:)으로 되어 있다.
- 각 필드의 내용 (순서대로)
  - 사용자명
    - 사용자명은 일반적으로 소문자를 쓴다.(대소문자로 인한 혼동을 피하기 위해)
  - 암호화된 암호
    - 대부분의 리눅스 시스템은 별도의 쉘도우 암호를 /etc/shadow 파일에 담아 그것을 이용한다. 따라서 이 필드는 실제 암호가 다른 곳에 있다는 것을 알려주는 x 기호만을 담고 있다.
  - 사용자 ID(UID값이 표기)
  - 기본 그룹(GID값이 표기)
  - 사용자 정보
    - 사용자에 대한 정보(이름, 전화번호 등)를 담고 있다.

- 사용자의 홈 디렉토리
  - 사용자가 인증을 받은 후에 login 프로그램이 \$HOME 변수의 값을 이 필드의 값으로 정의한다. 기본적으로 /home/username을 이용한다.
- 사용자의 로그인 셸
  - 사용자가 인증을 받은 후에 login 프로그램이 \$SHELL 변수의 값도 이 필드의 값으로 정의한다. 기본적으로 /bin/bash로 지정된다.

## 7.2 쉘도우 암호

### 쉘도우 암호

- 소금(salt)라 불리는 난수값에 의해 암호화가 되어 저장
- 소금을 이용함으로 암호는 4096 가지의 서로다른 방법으로 암호화가 됨
- 문제점
  - /etc/passwd 파일에는 각 사용자의 UID와 GID가 저장되어 있으며, 이 정보는 시스템 내의 누구라도 읽을 수 있어야 되기 때문에 결론적으로 이곳에 저장되는 암호 파일은 보안 공격에 취약하게 된다.
- 해결방안
  - /etc/passwd 파일에서 암호 부분만 뽑아내서 root로만 접근이 가능한 별도의 /etc/shadow 파일을 저장

### 쉘도우 암호

- 소금(salt)라 불리는 난수값에 의해 암호화가 되어 저장
- 소금을 이용함으로 암호는 4096 가지의 서로다른 방법으로 암호화가 됨
- 문제점
  - /etc/passwd 파일에는 각 사용자의 UID와 GID가 저장되어 있으며, 이 정보는 시스템 내의 누구라도 읽을 수 있어야 되기 때문에 결론적으로 이곳에 저장되는 암호 파일은 보안 공격에 취약하게 된다.
  - 만약 어느 해커가 /etc/passwd 파일에서 암호화된 암호 문자열을 읽어낼 수 있다면, 리눅스의 암호 알고리즘과 동일한 알고리즘을 써서 영어 사전의 단어들을 암호화해서 서로 같은지 비교해 볼 수 있게 된다. 즉, 단어집에서 단어를 하나씩 뽑아서 4096 가지의 소금값에 적용 후 시스템의 암호 문자열과 비교를 하여 만약 두 문자열이 서로 같으면 그 사용자 계정의 암호는 밝혀지는 것이다.
- 해결방안
  - /etc/passwd 파일에서 암호 부분만 뽑아내서 root로만 접근이 가능한 별도의 /etc/shadow 파일을 저장
  - 기본적으로 모든 리눅스 배포판에서는 쉘도우 암호를 이용한다.

## /etc/shadow 파일의 구성

### ■ example

```
hwlee:GcX5dT8cpoeEd:11088:0:99999:7:0:::
```

### ■ 필드의 구성

- 사용자명
- 암호화된 암호
- 암호의 최근 변경시간
- 변경 허용 최소 일수
- 강제 변경 일수
- 암호만료 경고 일수
- 만료와 계정 취소 사이의 일수
- 계정 만료
- 특수 플래그

## /etc/shadow 파일의 구성

### ■ example

```
hwlee:GcX5dT8cpoeEd:11088:0:99999:7:0:::
```

### ■ 필드의 구성

- 사용자명 : 사용자의 로그인명으로 /etc/passwd 파일에 있는 필드와 같은 값을 갖음
- 암호화된 암호 : 암호화된 암호가 실제로 저장되는 위치
- 암호의 최근 변경시간 : 1970년 1월 1일 이후로 해당 날짜까지 몇일이나 경과되었는지를 표기
- 변경 허용 최소 일수 : 암호를 변경했다면 그로부터 몇일 뒤 다시 암호를 바꿀수 있는지 설정
- 강제 변경 일수 : 이 필드에 지정된 날짜가 흐르면 사용자가 암호를 바꾸도록 만듦. 99999로 설정하면 암호 변경을 강요받지 않는다.
- 암호만료 경고 일수 : 암호가 만료되기 전 사용자에게 언제 알려줄 지를 지정
- 만료와 계정 취소 사이의 일수 : 계정 만료 후 몇일 뒤에 계정이 비활성화 될지를 결정
- 계정 만료 : 해당 계정이 무효화되는 날짜 입력
- 특수 플래그 : 현재 사용하지 않는 필드이다.

## 셴도우 암호를 관리하는 유틸

- pwconv
  - /etc/login.defs 파일에 정의된 PASS\_MIN\_DAYS, PASS\_MAX\_DAYS, 그리고 PASS\_WARN\_AGE 값들을 이웅해서 /etc/shadow 파일에 새로운 항목을 추가
  - /etc/shadow 파일의 항목들 중 /etc/passwd 파일 속에 정의되어 있지 않은 것들을 삭제
- pwunconv
  - /etc/passwd 파일과 /etc/shadow 파일을 비교 검사하면서 /etc/shadow의 암호필드를 /etc/passwd의 관련 라인에 병합
- grpconv
- grpunconv

## 셴도우 암호를 관리하는 유틸

- pwconv
  - /etc/login.defs 파일에 정의된 PASS\_MIN\_DAYS, PASS\_MAX\_DAYS, 그리고 PASS\_WARN\_AGE 값들을 이웅해서 /etc/shadow 파일에 새로운 항목을 추가
  - /etc/shadow 파일의 항목들 중 /etc/passwd 파일 속에 정의되어 있지 않은 것들을 삭제
- pwunconv
  - /etc/passwd 파일과 /etc/shadow 파일을 비교 검사하면서 /etc/shadow의 암호필드를 /etc/passwd의 관련 라인에 병합
- grpconv
  - pwconv와 동일한 기능을 하지만 /etc/group 파일의 그룹들에 대해 동작한다.
- grpunconv
  - pwunconv와 동일한 기능을 하지만 /etc/group 파일의 그룹들에 대해 동작한다

## 7.3 사용자 계정의 관리

### 새로운 사용자 추가과정

- ① /etc/passwd에 새 사용자에게 대한 항목을 생성한다.
- ② 사용자의 암호를 설정한다.
- ③ 로그인 쉘을 지정한다.
- ④ 사용자의 홈 디렉토리를 생성하고 적절한 퍼미션을 부여한다.

### 새로운 사용자 추가과정

(4번과 5번은 선택 사항)

- ① /etc/passwd에 새 사용자에게 대한 항목을 생성한다.
- ② 사용자의 암호를 설정한다.
- ③ 로그인 쉘을 지정한다.
- ④ 사용자의 홈 디렉토리를 생성하고 적절한 퍼미션을 부여한다.
- ⑤ 사용자의 홈 디렉토리 속에 여러 가지 도움이 될만한 파일들을 복사해준다.



## 새로운 사용자 추가 명령어 - useradd

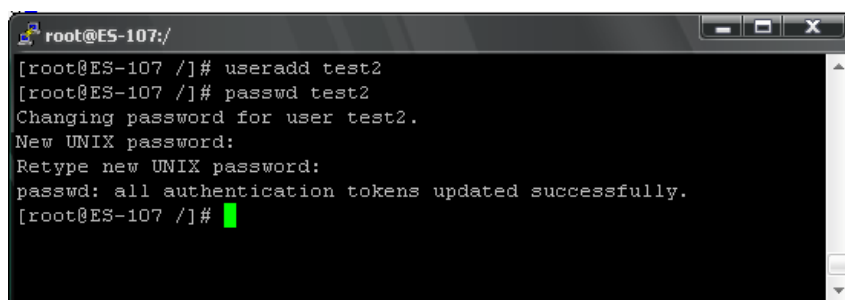
### ■ 형식

- useradd [-D] [-g default\_group] [-b default\_home] [-s default\_shell] username

## 사용자 추가 명령어 - useradd

### ■ 형식

- useradd [-D] [-g default\_group] [-b default\_home] [-s default\_shell] username
- /etc/passwd 파일의 각 필드에 들어갈 정보를 인자로 받는다
- 각 인자들을 안적으면 해당 옵션의 디폴트로 적용된다.
- 사용자를 추가 하면 passwd 명령어를 이용하여 암호를 설정해야 한다.
- example



```
root@ES-107:/  
[root@ES-107 /]# useradd test2  
[root@ES-107 /]# passwd test2  
Changing password for user test2.  
New UNIX password:  
Retype new UNIX password:  
passwd: all authentication tokens updated successfully.  
[root@ES-107 /]#
```

## 사용자 계정 수정

- /etc/passwd 파일 내의 항목을 편집
  - 가장 간편한 방법
- etc/shadow 파일의 내용을 편집
  - 위의 방법보다 복잡함
    - ▶ 암호 같은 경우 암호화가 되어있음
    - ▶ 날짜값들은 1970년 1월 1일 기준으로 계산된 값임

## 사용자 계정의 수정

- /etc/passwd 파일 내의 항목을 수정하는 방법
  - 일반적으로 가장 간편한 방법
  - example
    - 만약 testuser1 계정의 /bin/bash 쉘을 C셸로 변경한다면,



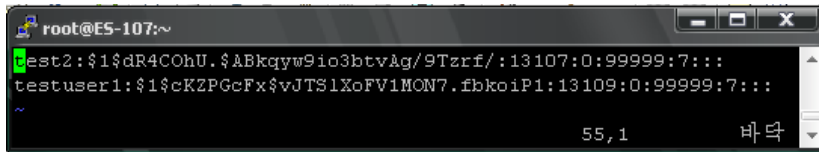
```
root@ES-107:~  
test2:x:504:504:./home/test2:/bin/bash  
testuser1:x:505:505:./home/testuser1:/bin/bash  
55,1 바닥
```

- 위 그림과 같은 상태에서 /bin/bash 부분을 아래와 같이 /bin/csh로 변경한다.



```
root@ES-107:~  
test2:x:504:504:./home/test2:/bin/bash  
testuser1:x:505:505:./home/testuser1:/bin/csh  
55,1 바닥
```

- /etc/shadow 파일 내의 항목을 수정하는 방법



```
root@ES-107:~  
test2:$1$dR4COhU.$ABkqyw9io3btvAg/9Tzrf/:13107:0:99999:7:::  
testuser1:$1$cKZPGcFx$vJTS1XoFV1MON7.fbko1P1:13109:0:99999:7:::  
~  
55,1      바닥
```

- 이 화면에서 보듯이 암호 같은 경우 암호화가 되어있고, 날짜값들은 1970년 1월 1일을 기준으로 계산된 값이기 때문에 수정하는데 어려움이 있고 위험요소도 있다.

## 사용자 계정 수정 (Cont)

### usermod를 이용해서 사용자 계정 수정하기

#### ■ 형식

- usermod [-c comment] [-d home\_dir [-m]] [-e expire\_date] [-f inactive\_time] [-g initial\_group] [-G group[,]] [-l login\_name] [-p passwd] [-s shell] [-u uid] [-o] [-L|-U] login

#### ■ 각 옵션들의 의미

- |                    |                 |
|--------------------|-----------------|
| ▶ -c comment       | ▶ -l login_name |
| ▶ -d home_dir [-m] | ▶ -p password   |
| ▶ -e expire_date   | ▶ -s shell      |
| ▶ -f inactive_time | ▶ -u uid        |
| ▶ -g initial_group | ▶ -L, -U        |
| ▶ -G group         | ▶ login         |

### usermod를 이용해서 사용자 계정 수정하기

#### ■ 형식

- usermod [-c comment] [-d home\_dir [-m]] [-e expire\_date] [-f inactive\_time] [-g initial\_group] [-G group[,]] [-l login\_name] [-p passwd] [-s shell] [-u uid] [-o] [-L|-U] login

#### ■ 각 옵션들의 의미

- |                    |   |
|--------------------|---|
| ▶ -c comment       | : 현재의 설명 필드의 값을 변경  |
| ▶ -d home_dir [-m] | : 새로운 홈디렉토리의 위치 설정(-m 옵션을 주면 디렉토리 내의 내용도 같이 옮겨짐)                |
| ▶ -e expire_date   | : 사용자 계정을 더 이상 사용하지 못하게 되는 날짜 설정                                |
| ▶ -f inactive_time | : 암호 만료일이 지난 후 몇일이 지나야 사용자 계정을 비활성화 하는지 설정                      |
| ▶ -g initial_group | : 새로운 로그인 그룹을 지정  |
| ▶ -G group         | : 사용자가 속할 그룹들을 콤마로 연결해서 지정                                      |
| ▶ -l login_name    | : 사용자의 로그인명을 변경   |
| ▶ -p password      | : 새로운 암호 지정   |
| ▶ -s shell         | : 사용자의 로그인 셸을 변경  |
| ▶ -u uid           | : 사용자 UID 변경  |
| ▶ -L               | : /etc/passwd 파일내 해당 사용자 암호값 부분 앞에 느낌표를 붙여서 변경, 즉 로그인 할 수 없게 만들 |
| ▶ -U               | : 위 -L 옵션에서 생성된 느낌표를 제거함  |
| ▶ login            | : 사용자의 로그인명   |

## 사용자 계정 막기

- /etc/shadow 파일의 관련항목을 수정하기
  - 암호를 변경하고 계정이 만료될 날짜를 수정
- change 명령어를 이용하기
  - change -E 2005-11-11 username 형식으로 사용자의 암호 만료일을 수정

## 사용자 계정 막기

- /etc/shadow 파일의 관련항목을 수정하기
  - 암호를 변경하고 계정이 만료될 날짜를 수정
  - 절차
    - ① 우선 계정을 막을 사용자의 암호를 변경한다.
    - ② /etc/shadow 파일에서 막을 계정을 찾아 세 번째 필드의 숫자를 확인한다.
      - ▶ 세 번째 필드의 숫자는 암호가 가장 최근에 변경된 날짜를 의미하고, 앞서서도 얘기했듯이 1970년 1월 1일로부터 최근에 변경된 날짜 즉 오늘날짜까지 흘러간 날짜수가 기록된다.
    - ③ 세 번째 필드의 수에서 1을 뺀 숫자를 8번째 필드에 입력한다.
      - ▶ 8번째 필드는 계정이 만료될 날짜를 나타낸다. 이 수를 세 번째 필드의 수에서 1을 뺀 값을 적으므로써 해당 사용자가 더 이상 로그인하지 못하도록 만들 수 있다.
- change 명령어를 이용하기
  - change -E 2005-11-11 username 형식으로 사용자의 암호 만료일을 수정

## 사용자 계정 삭제

### ■ 절차

- ① /etc/passwd에서 그 사용자에 대한 항목을 삭제한다.
- ② 사용자의 파일들을 백업한다.
- ③ 사용자의 파일들을 지운다.
- ④ 사용자의 홈 디렉토리를 지운다.

### ■ 사용자 계정을 삭제하는 방법

- 수작업으로 계정을 삭제하는 방법
- userdel 유틸리티를 이용하여 계정을 삭제하는 방법
  - userdel -r username 형식으로 작성하면, 해당 username에 관한 항목을 삭제하고 홈 디렉토리까지 삭제한다

## 사용자 계정 삭제

### ■ 절차

- ① /etc/passwd에서 그 사용자에 대한 항목을 삭제한다.
- ② 사용자의 파일을 백업한다.
- ③ 사용자의 파일들을 지운다.
- ④ 사용자의 홈 디렉토리를 지운다.
- ⑤ 파일시스템을 검색하여 삭제된 사용자 소유의 파일들을 찾아 지우거나 다른 사용자의 소유로 바꿔주어야 한다.

### ■ 사용자 계정을 삭제하는 방법

- 수작업으로 계정을 삭제하는 방법
  - ① /etc/passwd 파일에서 삭제하려는 사용자에 대한 항목을 삭제한다.
  - ② pwconv 명령으로 그 사용자에 대한 /etc/shadow 항목도 삭제한다.
  - ③ 사용자의 홈 디렉토리와 파일들을 모두 지운다.
    - 형식은 rm -r /home/delete\_user 이런 식으로 한다.
  - ④ 해당 컴퓨터 전체를 검색하여 삭제된 사용자의 권한으로 된 파일을 찾는다.
    - 만약 삭제할 사용자의 UID와 GID가 500 이라면, find / -uid 500 -gid 500 이러한 명령으로 처리해야 할 파일들을 찾을 수 있다.
  - ⑤ 찾은 파일들을 삭제하거나 chown 명령어를 이용하여 소유주를 변경한다.

▸ `chown aaa.bbb /opt/somefile`

`/opt/somefile`의 소유주를 `aaa`로 변경하고 그룹을 `bbb`로 바꿔주는 명령어이다.

## 7.4 그룹

### 그룹

- 모든 사용자들은 각각 로그인 그룹이라고 하는 기본 그룹 ID를 가짐
- 로그인시 /bin/login에 인증을 받으면 /etc/passwd 파일에 지정되어 있는 그룹값이 기본그룹이 된다.
- 일반적으로 레드햇 리눅스에서는 사용자 ID와 기본 그룹 ID가 같다.
- 다른 배포판은 일반적으로 모든 사용자들이 users라는 하나의 그룹에 속한다.
- 리눅스 관리자들은 필요에 따라 그룹들을 생성하여 필요에 따라 유저들을 적절한 그룹에 배분하여야 하며, 필요가 없는 그룹들은 삭제시켜줘야 한다.



## 그룹의 특징

- 파일에 그룹을 설정하고 또한 사용자들을 특정 그룹에 포함시킴으로써 리눅스의 보안성을 향상 시킴
- 예를들면, 사용자마다 특정 그룹을 지정함으로써 한 그룹의 사용자들이 다른 그룹의 사용자의 파일들을 읽거나 쓸 수 없도록 만들 수 있다.

## 그룹의 설정

- 리눅스 배포판에 따라 그룹을 설정하는 방법이 다르다.

## 그룹의 특징

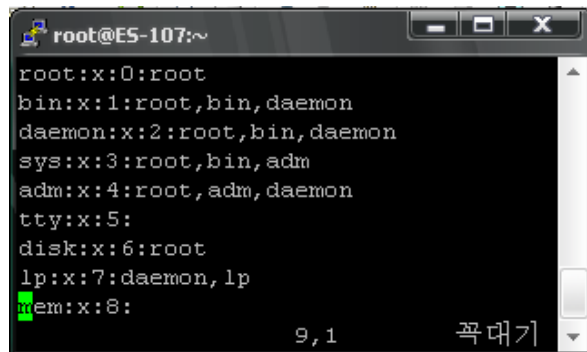
- 파일에 그룹을 설정하고 또한 사용자들을 특정 그룹에 포함시킴으로써 리눅스의 보안성을 향상 시킴
- 예를들면, 사용자마다 특정 그룹을 지정함으로써 한 그룹의 사용자들이 다른 그룹의 사용자의 파일들을 읽거나 쓸 수 없도록 만들 수 있다.
- 사용자 본인과 해당 그룹은 읽기와 쓰기 퍼미션을 갖고, 그 외의 사용자들은 읽기만 가능하다.

## 그룹의 설정

- 리눅스 배포판에 따라 그룹을 설정하는 방법이 다르다.
  - 레드햇의 경우, 새로운 사용자를 생성하면 사용자명과 같은 번호의 그룹 ID가 해당 사용자에게 할당된다.
  - 다른 리눅스 배포판 중에는 users라는 하나의 그룹을 만들고 모든 사용자를 그 그룹의 멤버로 지정하는 경우도 있다.
  - 이런 형식의 경우 모든 사용자 계정은 모두 users 그룹의 멤버가 된다. 보통 umask의 값은 022로 설정되어 있다. 따라서 다른 사용자의 파일을 읽을 수는 있지만 쓸 수는 없다. 이러한 방식을 ‘공유방식’이라 한다.

## /etc/group 파일

- 각 그룹에 대한 정보를 담은 파일
- 각 항목은 그룹명, 암호, 숫자로 된 그룹 ID(GID), 그룹 멤버 목록으로 구성되어 있다.
- /etc/group 파일의 예



```
root@ES-107:~  
root:x:0:root  
bin:x:1:root,bin,daemon  
daemon:x:2:root,bin,daemon  
sys:x:3:root,bin,adm  
adm:x:4:root,adm,daemon  
tty:x:5:  
disk:x:6:root  
lp:x:7:daemon,lp  
mem:x:8:
```

9,1      꼭대기

- 기본리눅스 그룹
  - 아래 내용 참고

## /etc/group 파일

- 각 그룹에 대한 정보를 담은 파일
- 각 항목은 그룹명, 암호, 숫자로 된 그룹 ID(GID), 그룹 멤버 목록으로 구성되어 있다.
- 기본리눅스 그룹
  - 다음 장에 나오는 표를 참고

그룹	GID	멤버	기능
root	9	root	슈퍼유저 그룹
bin	1	root, bin, daemon	프로그램 실행
daemon	2	root, bin, daemon	프로그램 실행
sys	3	root, bin, adm	시스템 그룹
adm	4	root, bin, daemon	관리 그룹
tty	5		터미널 액세스
disk	6	root	디스크 디바이스 파일 액세스
lp	7	daemon, lp프	린트 인쇄 그룹
mem	8		커널 메모리 액세스
kmem	9		커널 메모리 액세스
wheel	10	root	루트 권한 허용 사용자들
mail	12	mail	메일 유틸리티가 사용함
news	13	news	유즈넷 뉴스 유틸리티가 사용함
uucp	14	uucp	UUCP 네트워킹이 사용함
man	15		매뉴얼 페이지 액세스를 위해 사용함
games	20		게임 고득점을 저장하기 위한 그룹
gopher	30		Gopher 유틸리티가 사용함
dip	40		다이얼업 IP 그룹(PPP, SLIP)
ftp	50		FTP 데몬을 위한 그룹
nobody	99		저수준 보안 그룹
users	100		기본 사용자 그룹
utmp	22		utmp 유틸리티가 사용함
xfx	43		X 글꼴 서버가 사용함
apache	48		Apache 데몬이 사용함
named	25		DNS가 사용함
floppy	19		저수준 플로피 디스크 디바이스를 액세스하기 위한 그룹
pppusers	44		PPP만 액세스하는 사용자들
popusers	45		POP만 액세스하는 사용자들
slipusers	46		SLIP만 액세스하는 사용자들

## 7.5 그룹의 관리

### 그룹 추가

- /etc/group 파일의 내용을 수정함으로 추가하는 방법
  - 추가하고자 하는 그룹을 작성하고, 뒤에 그 그룹 멤버들을 콤마로 연결해서 추가
- groupadd 명령어를 이용하여 그룹을 추가하는 방법
  - 형식
    - groupadd [-g GID[-o]] [-r] [-f] group

### 그룹 추가

- /etc/group 파일의 내용을 수정함으로 추가하는 방법
  - 추가하고자 하는 그룹을 작성하고, 뒤에 그 그룹 멤버들을 콤마로 연결해서 추가
  - example
    - test111:x:111:testuser1,testuser2,testuser3
  - 참고로 어떤 사용자가 어느 그룹에 속해 있는지 알고 싶을 때는 id 명령어를 사용한다. 또한 groups 명령어도 비슷한 기능을 한다.
    - example
- groupadd 명령어를 이용하여 그룹을 추가하는 방법
  - 형식
    - groupadd [-g GID[-o]] [-r] [-f] group
  - 각 옵션들의 의미
    - -g GID : 그룹 ID의 숫자값
    - -r : 새로 생성하는 그룹이 시스템 그룹임을 알려주는 기능
    - -f : 만약 주어진 그룹 이름이 기존 시스템에 존재한다면, 아무런 에러 메시지도 출력하지 않고 조용히 groupadd 명령을 끝내도록 한다.  
레드햇 리눅스에서 이 옵션은 /etc/group 내에서 가장 큰 GID값의

다음값을 저장하는 그룹을 생성하도록 하는 옵션으로도 사용된다.

- group : 새로 생설될 그룹의 이름

## 그룹 수정

### ■ 수작업으로 그룹 수정하기

- /etc/group 파일을 편집해서 그룹명, GID, 멤버 등을 수정할 수 있다.
- example  
아래내용 참고

### ■ groupmod를 이용해서 그룹 수정하기

- 형식  
groupmod [-g GID [-o]] [-n group\_name] group

## 그룹 수정

### ■ 수작업으로 그룹 수정

- /etc/group 파일을 편집해서 그룹명, GID, 멤버 등을 수정할 수 있다.
- example

```
test111:x:111:testuser1,testuser2,testuser3
```

이러한 부분에서 그룹 멤버에 testuser4를 추가하고 그룹명도 testgroup으로 변경하고자 한다면,

```
testgroup:x:111:testuser1,testuser2,testuser3,testuser4
```

이와같이 수정하여 주면 된다.

### ■ groupmod를 이용해서 그룹 수정

- 형식  
groupmod [-g GID [-o]] [-n group\_name] group
- 옵션 설명
  - -g : GID 그룹 ID의 숫자값
  - -n group\_name : 그룹의 새로운 이름
  - group : 기존 그룹명

## 그룹 삭제

- 수작업으로 그룹 삭제
  - 앞에서 수정할 때와 같은 방법으로 /etc/group 파일을 수정
- groupdel 명령어를 이용해서 그룹 삭제
  - 형식  
`groupdel group_name`

## 소속된 그룹 임시적으로 변경하기

- newgrp 명령어를 이용하여 소속 그룹 변경하기
  - 형식  
`newgrp - change_group`

## 그룹 삭제

- 수작업으로 그룹 삭제
  - 앞에서 수정할 때와 같은 방법으로 /etc/group 파일을 수정
- groupdel 명령어를 이용해서 그룹 삭제
  - 형식  
`groupdel group_name`

## 소속된 그룹 임시적으로 변경하기

- 필요성
  - 사용자가 수행하는 모든 작업은 그 사용자의 UID와 GID 값을 기반으로 이루어 진다.
  - 또한 한 사용자가 많은 그룹에 속해있다고 하더라도, 어떤 한 순간에는 단 하나의 그룹에 속한 형태로 동작한다.
  - 어떤 파일을 특정 그룹의 멤버들만이 액세스 할 수 있게 하려면 그 그룹에 속한 신분으로 파일을 생성해야 한다.

- newgrp 명령어를 이용하여 소속 그룹 변경하기
    - 형식 : newgrp 명령어 뒤에 하이픈(-)을 붙이고 한 칸을 띄워 바꿀 그룹명을 지정
      - newgrp - change\_group
      - example  
newgrp - testgroup
- 이렇게 사용자가 testgroup으로 변경한 후 파일을 하나 생성하면 그 파일의 그룹 소유권은 새로 바꾼 testgroup에 속하게 된다. 이는 ls -l 명령으로 확인할 수 있다.



## 8장 네트워크

8.1 TCP/IP 리눅스 네트워킹

8.2 메일서버 설정

## Overview

## 8.1 TCP/IP 리눅스 네트워킹

### 개요

- 인터넷에서 이용되는 프로토콜의 핵심은 전송미션 컨트롤 프로토콜 (Transmission Control Protocol)과 인터넷 프로토콜(Internet Protocol)이다.
- TCP/IP 네트워킹의 기본 디자인을 이해하고 있으면 시스템을 설정해서 여러 기능을 이용할 수 있도록 준비하는데 크게 유리할 뿐만 아니라 특히 TCP/IP 문제가 생겼을 때 정확한 처리가 가능하다.
- 따라서 TCP/IP 네트워킹의 개념을 이해하는 일은 시스템 관리자로서 기본적인 것이고 꼭 필요하다.

## TCP/IP의 개념

### ■ TCP/IP란?

- 미국방성 ARPANET 프로젝트에서 시작된 프로토콜
- 특정 운영체제에 한정된 프로토콜이 아닌 개방되어 있는 프로토콜이므로 서로 다른 운영체제, 네트워크 장치가 서로 통신이 가능한 프로토콜
- TCP
  - ▶ Transmission Control Protocol
  - ▶ 전송을 제어하는 프로토콜
- IP
  - ▶ Internet Protocol
  - ▶ IP주소 : 인터넷에 연결된 컴퓨터들끼리 서로를 구별하기 위해 붙인 주소

## TCP/IP의 개념

### ■ TCP/IP 란?

- TCP/IP는 원래 미국방성 ARPANET 프로젝트에서 시작된 프로토콜이다.
- 미국방성 내부 프로젝트이었으나 대학이 참여함으로써 세상에 널리 알려지는 계기가 되었다. TCP/IP는 특정 운영체제에 한정된 프로토콜이 아닌 개방되어 있는 프로토콜이므로 서로 다른 운영체제, 네트워크 장치가 서로 통신이 가능한 프로토콜이다. 또한 TCP/IP는 네트워크 크기와 그 네트워크를 구성하는 호스트들의 수에 따라 클래스를 나누어서 네트워크를 구축하여 관리할 수 있으므로 확장성과 네트워크 관리의 융통성을 제공한다.

### ■ TCP

- Transmission Control Protocol의 약자이며 말 그대로 전송을 제어하는 프로토콜이다. IP위에서 구현되어 안정적인 연결, 메시지 교환, 패킷 손실 문제를 다루는 프로토콜이다

### ■ IP

- IP는 Internet Protocol의 약자이며, TCP/IP상에서의 데이터 교환 단위를 패킷이라고 하는데 IP는 패킷의 생성방법, 보내는 사람의 주소(발신지), 받는 사람의 주소(수신지), 전송하는 문제등을 다룬다. 즉, IP주소는 인터넷에 연결된 컴퓨터들끼리 서로를 구별하기 위해 붙인 주소라고 보면 된다.

## 프로토콜이란?

- 다른 종류의 컴퓨터를 네트워크를 통하여 통신이 가능하기 위해 서로를 연결해주는 하드웨어나 운영체제를 연결하기 위한 통신 규약

## TCP/IP를 기반으로한 프로토콜의 종류

- telnet
- FTP
- SMTP
- SNMP
- HTTP

## 프로토콜이란?

- 다른 종류의 컴퓨터를 네트워크를 통하여 통신이 가능하기 위해 서로를 연결해주는 하드웨어나 운영체제를 연결하기 위한 통신 규약

## TCP/IP를 기반으로한 프로토콜의 종류

- telnet
  - 네트워크 터미널 프로토콜(Network Terminal Protocol)
  - 네트워크를 통한 원격 로그인을 제공한다.
- FTP
  - 파일 전송 프로토콜(File Transfer Protocol)
  - 양방향 파일 전송에 사용된다.
- SMTP
  - 메일 전송 프로토콜(Simple Mail Transfer Protocol)
  - 전자 메일을 전달한다.

■ SNMP

- 네트워크 관리 프로토콜(Simple Network Management Protocol)
- 네트워크 구성을 관리한다.

■ HTTP

- 하이퍼텍스트 전송 프로토콜(Hypertext Transfer Protocol)
- 네트워크를 통해 웹페이지를 전달한다.

## IP 주소

- 인터넷상에서 각각의 컴퓨터를 구분해주는 유일한 번호
- 이 번호는 컴퓨터에 연결된 각각의 네트워크 연결 인터페이스에 부여
- IP주소는 4바이트의 크기로 나타낸다.
- 각 바이트는 '.(dot)'으로 구분하여 표시하고 각 바이트는 0부터 255까지의 십진수로 바뀌어 표시된다.

## IP 주소의 구분 (클래스)

- IP 네트워킹 프로토콜이 이해할 수 있는 것은 32비트 수이다.
- 모든 컴퓨터는 네트워크 환경에서 유일한 수의 IP를 지정해야 한다.
- 네트워크의 크기와 그 네트워크를 구성하는 호스트들의 수에 따라 A, B, C 등의 클래스로 나눌 수 있다.

## IP 주소

- 인터넷상에서 각각의 컴퓨터를 구분해주는 유일한 번호
- 이 번호는 컴퓨터에 연결된 각각의 네트워크 연결 인터페이스에 부여된다.
- IP주소는 4바이트의 크기로 나타낸다.
- 각 바이트는 '.(dot)'으로 구분하여 표시하고 각 바이트는 0부터 255까지의 십진수로 바뀌어 표시된다.

## IP 주소의 구분 (클래스)

- IP 네트워킹 프로토콜이 이해할 수 있는 것은 32비트 수이다.
- 모든 컴퓨터는 네트워크 환경에서 유일한 수의 IP를 지정해야 한다.
- 네트워크의 크기와 그 네트워크를 구성하는 호스트들의 수에 따라 A, B, C 등의 클래스로 나눌 수 있다.



- 할당된 범위는 아래와 같다.

Network Class	Network Addresses	Netmask
A	0.0.0.0 - 127.255.255.255	255.0.0.0
B	128.0.0.0 - 191.255.255.255	255.255.0.0
C	192.0.0.0 - 223.255.255.255	255.255.255.0

### 넷마스크(Netmask) 란?

- 인터넷에 연결된 컴퓨터의 네트워크 소속을 알 수 있는 주소
- 넷마스크 주소는 점으로 구분되는 4자리 수로 구성되어 있다.

### 게이트웨이(Gateway)

- TCP/IP에 연결된 컴퓨터에서 어디로 통해서 다른 네트워크를 통할 수 있는지 알려주는 주소
- 게이트웨이는 둘 또는 그 이상의 물리적 네트워크에 동시에 연결되어 있고, 그들간에 패킷을 교환하도록 설정된 것이다

### 넷마스크(Netmask) 란?

- 인터넷에 연결된 컴퓨터의 네트워크 소속을 알 수 있는 주소
- 넷마스크 주소는 점으로 구분되는 4자리 수로 구성되어 있다.

### 게이트웨이(Gateway)

- TCP/IP에 연결된 컴퓨터에서 어디로 통해서 다른 네트워크를 통할 수 있는지 알려주는 주소
- 게이트웨이는 둘 또는 그 이상의 물리적 네트워크에 동시에 연결되어 있고, 그들간에 패킷을 교환하도록 설정된 것이다.
- 호스트가 직접적으로 대화할 수 있는 것은 그 네트워크 내에 있는 것들뿐이다. 그 외의 모든 호스트들에는 게이트웨이(gateway)라는 것을 통해서만 접근할 수 있다.

## 도메인 네임 서버(DNS : Domain Name Server)

- 이름과 IP 주소를 매핑하여주는 거대한 분산 네이밍 시스템
- IP 주소와 도메인 주소를 일대일로 대응시켜 DNS를 이용해 IP 주소를 도메인 주소로 또는 도메인 주소를 IP 주소로 변환해주는 역할을 하는 것

## 유동 IP란? (DHCP)

- 현재의 IP주소가 부족한 상황을 해결하기 위한 미봉책으로 생겨난 것이다.
- 여러대의 컴퓨터가 하나의 IP주소를 공유하도록 설계된 것
- DHCP는 부족한 IP주소를 할당할 뿐만 아니라, 사용자가 자신의 컴퓨터에서 복잡하게 IP주소를 할당하는 불편함을 덜어주기도 한다.

## 도메인 네임 서버(DNS : Domain Name Server)

- 이름과 IP 주소를 매핑하여주는 거대한 분산 네이밍 시스템
- 인터넷에서 사용되는 IP(Internet Protocol), 그리고 IP의 상위에서 동작하는 넷스케이프 같은 응용 프로그램들은 210.123.34.103 과 같이 표현되는 IP 주소만을 인식하게 되는데, 이러한 IP 주소는 프로그램입장에선 해석하기 수월하지만 사람이 기억하여 사용하기가 어렵고, IP 주소만으로는 서비스 유형을 예측하기 힘들다는 단점이 있다.
- 그래서 IP 주소와 도메인 주소를 일대일로 대응시켜 DNS를 이용해 IP 주소를 도메인 주소로 또는 도메인 주소를 IP 주소로 변환해주는 역할을 하는 것이다.
- IP 주소로는 연결이 되지만 도메인 주소로 연결이 되지 않을 경우는 DNS에 문제가 있거나 DNS 설정이 되지 않았기 때문이다.

## 유동 IP란? (DHCP)

- 현재의 IP주소가 부족한 상황을 해결하기 위한 미봉책으로 생겨난 것이 유동 IP주소이다.
- 여러대의 컴퓨터가 하나의 IP주소를 공유하도록 설계된 것이다

- 전원을 켜서 인터넷에 접속한 컴퓨터에게만 IP주소를 할당하고, 전원이 꺼질 때 IP주소를 환수하여 다음에 전원을 켜고 인터넷에 접속하는 컴퓨터에게 할당하는 방식이다.
- 기본적으로 DHCP란 기술을 통하여 이루어진다. DHCP는 이렇게 부족한 IP주소를 할당할 뿐만 아니라, 사용자가 자신의 컴퓨터에서 복잡하게 IP주소를 할당하는 불편함을 덜어주기도 한다.

## 9장

# 백업과 복원

9.1 백업정책

9.2 백업 미디어

9.3 리눅스 백업 및 복원 툴

9.4 운영체제 백업

9.5 장애 복구 기법

## 9.1 백업 정책

### 백업정책이란?

- 어떤 정보를 얼마나 자주 백업할 것인지를 결정하는 것

### 백업정책의 종류

- 완전 백업(full backup)
  - 모든 작업을 백업한다는 의미
- 증진 백업(incremental backup)
  - 가장 최근에 백업한 후의 작업만을 백업한다는 의미

### 백업정책의 사용

- 완전 백업과 증진 백업을 적절히 섞어서 시간과 노동을 아낌
- 백업은 시스템 사용량이 가장 적을 때 하는 것이 좋다.

### 백업정책이란

- 어떤 정보를 얼마나 자주 백업할 것인지를 결정하는 것

### 백업정책의 종류

- 완전 백업(full backup)
  - 모든 작업을 백업한다는 의미
- 증진 백업(incremental backup)
  - 가장 최근에 백업한 후의 작업만을 백업한다는 의미

### 백업정책의 사용

- 완전 백업과 증진 백업을 적절히 섞어서 시간과 노동을 아낌
- 백업은 시스템 사용량이 가장 적을 때 하는 것이 좋다.
- 컴퓨터 시스템의 특성에 잘 맞는지 확인해야 한다.
- 데이터의 종류, 정적 데이터인지 동적 데이터 인지 확인해야 하고, 시스템의 사용량 등에 따라 이러한 요구사항들에 가장 잘 만족시키는 적절한 백업 정책을 사용해야 한다.

## 차등 백업(differential backup)

- 의미 : 바로 전에 수행한 백업 이후로 바뀐 파일들만을 백업
- 증진 백업과의 차이 : 증진 백업은 완전 백업 이후로 바뀐 파일들만을 백업하는 것을 의미
- 차등 백업의 장점
  - 백업 시간이 증진 백업에 비해 빠르다.
- 차등 백업의 단점
  - 백업을 풀 경우 최근 완전 백업한 자료를 풀고 그 뒤로 해온 모든 차등 백업들을 차례대로 모두 풀어야 하기 때문에 복원하는데 걸리는 시간은 증진 백업을 사용하였을 경우보다 훨씬 오래 걸린다.



## 9.2 백업 미디어

### 적합한 백업 미디어 고르기

- 백업을 하는데 있어서 어떤 미디어를 이용해 백업을 할지를 정하는 것은 중요하다.
- 고려해야 할 사항
  - 백업할 데이터의 크기와 지원하는 미디어가 무엇인지 확인
  - 현재 소유하고 있는 백업 미디어로 무엇이 있고 얼마나 효율적으로 백업을 할 수 있을지 확인

### 백업 미디어의 종류

- 테이프
- 플로피티컬 디스크
- CD-R
- 플로피 디스크 등

### 적합한 백업 미디어 고르기

- 백업을 하는데 있어서 어떤 미디어를 이용해 백업을 할지를 정하는 것은 중요하다.
- 고려해야 할 사항
  - 백업할 데이터의 크기와 지원하는 미디어가 무엇인지 확인
  - 현재 소유하고 있는 백업 미디어로 무엇이 있고 얼마나 효율적으로 백업을 할 수 있을지 확인

### 백업 미디어의 종류

- 테이프
- CD-R
- 플로피티컬 디스크
- 플로피 디스크 등

## 테이프

- 용량과 가격 면에서 가장 우수한 백업 미디어
- 테이프의 종류
  - 8mm 나선형 기록(helical scan) 테이프
  - 4mm 나선형 기록 테이프 (일명 DAT)
  - 1/4인치 선형 카트리지 테이프 (QIC)
  - Travan (QIC의 변형)

## 테이프

- 최초로 개발된 보조저장장치
- 비트당 가격이 가장 저렴
- 순차적 액세스 방식 이용
- 액세스 속도가 매우 느리기 때문에 주로 백업(backup)용 저장장치로 사용
- 조직
  - 적은 수의 병렬 트랙들로 구성
  - 레코드(record) : 테이프에 저장된 데이터 블록
  - 레코드간 갭(inter-record gap) : 레코드들 간의 구분을 위한 간격
- 용량과 가격 면에서 가장 우수한 백업 미디어
- 테이프의 종류
  - 8mm 나선형 기록(helical scan) 테이프
  - 4mm 나선형 기록 테이프 (일명 DAT)
  - 1/4인치 선형 카트리지 테이프 (QIC)
  - Travan (QIC의 변형)

## 나선형 기록 테이프

- 헤드/드럼이 돌면서 읽기와 쓰기를 수행
- 헤드는 테이프 면에 비스듬한 사선 방향으로 기록
- 대부분의 나선형 기록 테이프들은 내부적으로 데이터 압축을 수행
- 장점
  - 하드웨어로 압축을 수행하기 때문에 컴퓨터 CPU 부하를 덜어줌
  - 소프트웨어로 압축하는 것보다 신뢰도가 높다.

## QIC와 Travan 선형 테이프

- QIC는 1972년에 3M 주식회사가 개발
- 오디오 카세트 테이프와 유사한 외형을 갖으며, 두 개의 감개(reel)가 들어있어 하나가 테이프를 감고 다른 하나가 받쳐주는 형식
- 헤드부분은 쓰기 헤드와 양쪽의 읽기 헤드로 구성
- 장점
  - 드라이브 가격 면에서 나선형 기록 테이프보다 저렴
- 단점
  - 용량과 신뢰성이 다소 떨어지며 소음도 약간 있음
  - 테이프는 나선형 기록 테이프보다 약간 비쌈

## 디지털 선형 테이프(DLT)

- 구성에 따라 10GB 드라이브로부터 1.5TB를 저장할 수 있는 테이프 라이브러리까지 다양함
- 0.5인치 금속 입자 테이프를 이용
- 일부 신형 DLT 테이프 드라이브는 헤드에 두 개의 읽기/쓰기 장치가 있어서 두 개의 채널을 동시에 기록이 가능
- 장점
  - 테이프가 가이드를 통과하면서 청소되는 메커니즘도 있기 때문에 테이프의 수명이 길다.
  - 전송률을 효율적으로 두배까지 늘릴 수 있다.
  - 테이프의 논리적인 끝 단에 파일 색인을 기록하여 이용함으로써 검색 시간이 짧다.

## 맘모스

- 엑사바이트(Exabyte)사에서 개발
- 전용 테이프 뿐 아니라 구형 엑사바이트 테이프도 읽기 가능
- 맘모스 라이브러리로 테이프 라이브러리 시스템을 제작할 경우 수 테라바이트 이상 저장할 수 있음
- 장점
  - 테이프를 적재하거나 빼낼 때마다 읽기헤드와 쓰기 헤드를 자동으로 청소시켜주기 때문에 수명이 길어진다.
  - 전송률도 180MB 정도로 빠른편이다.
- 단점 - 가격면에서 비싸다.

### 지능형 테이프(AIT, Advanced Intelligent Tape)

- 순수한 자기화 계층과 다이아몬드 성분 코팅(DLC) 기술을 접목하여 기록 용량이 크고, 내구성도 튼튼하면서, 더 얇아짐
- 헤드 청소가 필요할 때마다 드라이브가 사용자에게 알림
- 테이프에 대한 정보를 담는 플래시 메모리 칩을 내장하여 읽기 속도를 향상시킴

### CD-R과 CD-RW

- 장점 : 가격이 저렴하다.
- 단점 : 용량이 650-700MB 정도로 작기 때문에 고용량 백업에는 적합하지 않다.

## 기타 백업 장치

- 플롭티컬
  - 플로피 디스크를 대체하는 장비
  - 플로피 디스크보다 정교하고, 용량도 큼
- 착탈식 드라이브
  - 대표적인 예 : ZIP 드라이브, Jaz 드라이브 등
  - 플로피보다는 훨씬 많은 데이터를 담아 옮길 수 있지만 백업 솔루션으로는 용도가 적절치 못함
- 광자기 드라이브
  - 레이저 부품과 자석 부품 두종류의 헤드를 함께 이용하므로 구성이 복잡하고 드라이브 무게가 무겁고 가격도 비싼 편

## 기타 백업 장치

- 플롭티컬
  - 플로피 디스크를 대체하는 장비
  - 읽기와 쓰기를 레이저로 하기 때문에 기존 플로피 디스크보다 정교하고, 용량도 크다.
  - 장점 : 특별한 포맷을 필요로 하지 않는다.
  - 단점 : 용량에 비해 가격이 비싸다.
- 착탈식 드라이브
  - 대표적인 예로 ZIP 드라이브와 Jaz 드라이브 등이 있다.
  - 플로피보다는 훨씬 많은 데이터를 담아 옮길 수 있지만 백업 솔루션으로는 용도가 적절치 못하다.
- 광자기 드라이브
  - 레이저 부품과 자석 부품 두종류의 헤드를 함께 이용하므로 구성이 복잡하고 드라이브 무게가 무겁고 가격도 비싼 편이다.



## 장치에 따른 백업 명령어와 유틸리티

### ■ 테이프 유틸리티

- mt 유틸리티
  - 테이프의 장력을 팽팽하게 잡아주는 작업
  - 테이프 지우기 / 앞으로 빨리 감기
  - 한 테이프에 하나 이상의 백업 넣기
- rmt 유틸리티
  - 원격 dump나 원격 restore 유틸리티가 원격 백업 서버에 달려있는 테이프 드라이브를 제어하기 위해 이용하는 유틸

## 장치에 따른 백업 명령어와 유틸리티

- 테이프 유틸리티
  - mt 유틸리티
    - 테이프의 장력을 팽팽하게 잡아주는 작업  
ex) # mt -f /dev/st0 reten
    - 테이프 지우기  
ex) # mt -f /dev/st0 erase
    - 앞으로 빨리 감기  
ex) # mt -f /dev/st0 fsf 2
    - 한 테이프에 하나 이상의 백업 넣기 위해 테이프의 기록된 데이터의 끝(eof)가 나타날때까지 테이프를 진행 방향으로 감는 작업  
ex) # mt -f /dev/st0 eof
  - rmt 유틸리티
    - 원격 dump나 원격 restore 유틸리티가 원격 백업 서버에 달려있는 테이프 드라이브를 제어하기 위해 이용하는 유틸
    - 사용방법은 mt와 거의 같음

## 테이프 장치명

- 백업 명령어 등을 이용시 사용할 테이프 디바이스명을 지정
- 테이프 디바이스명은 연결 인터페이스의 종류에 따라 틀림
- 테이프 드라이브의 종류
  - IDE 디바이스
  - SCSI 디바이스
  - ftape 디바이스
  - USB 디바이스 (SCSI 디바이스와 유사하게 동작함)

## 백업 명령어와 유틸리티

- 테이프 장치명
  - 백업 명령어 등을 이용시 사용할 테이프 디바이스명을 지정
  - 테이프 디바이스명은 연결 인터페이스의 종류에 따라 틀림
  - 테이프 드라이브의 종류
    - IDE 디바이스
      - /dev/ht0 또는 /dev/nht0 라는 디바이스 파일명을 통해 액세스 가능
      - /dev/ht0 : 자동 되감기 기능이 탑재된 디바이스
      - /dev/nht0 : 자동 되감기 기능이 없는 디바이스
      - 일반적으로 두 개의 IDE 포트를 지원
    - SCSI 디바이스
      - 자동 되감기 기능에 따라 /dev/st0와 /dev/nst0라는 디바이스 파일명 사용
      - SCSI 어댑터는 7개(내로우 SCSI) 또는 15개(와이드 SCSI)의 디바이스를 지원

- ▶ ftape 디바이스
  - 자동 퇴감기 기능에 따라 /dev/rft0와 /dev/nrft0라는 디바이스 파일명 사용
  - 호환성을 위해 /dev/ftape와 /dev/nftape라는 심볼릭 링크가 위 파일명을 가리킴
- ▶ USB 디바이스 (SCSI 디바이스와 유사하게 동작함)

## CD-R 및 CD-RW 백업툴

- CD-R 명령어
  - mkisofs
    - ISO-9660 파일시스템 이미지를 생성
    - example
      - \$ `mkisofs -r -J -o cd_image.iso /home/testuser1`
      - /home/testuser1 디렉토리 안에 내용을 cd\_image.iso라는 이미지로 만들라고 하는 예
      - 자세한 옵션 사용법은 mkisof 유틸리티의 매뉴얼 페이지를 참고

## CD-R 및 CD-RW 백업툴

- CD-R 명령어
  - mkisofs
    - ISO-9660 파일시스템 이미지를 생성
    - example
      - \$ `mkisofs -r -J -o cd_image.iso /home/testuser1`
      - 위 예제는 /home/testuser1 디렉토리 안에 내용을 cd\_image.iso라는 이미지로 만드는 예이다.
      - 옵션 설명
        - r : Rock Ridge 확장을 추가
        - J : Joliet 파일시스템을 추가
      - 이 옵션 외 자세한 옵션 사용법은 mkisofs 유틸리티의 매뉴얼 페이지를 참고
  - 위의 과정을 마치면 리눅스의 루프백 기능을 이용하여 mkisofs로 생성한 이미지를 검사할 수 있다. 쉽게 말해서, 이미지 파일을 마치 디스크 파티션처럼 마운트 할 수 있는 것이다.
    - 마운트를 하는 예
      - # `mount -t iso9660 -o loop cd_image.iso /mnt/tmp`

## CD-R 명령어

### cdrecord

- 공CD-R에 이미지를 굽는 명령어
- SCSI 방식과 ATAPI 방식의 CD 레코더를 지원
- example
 

```
# cdrecord -v speed=2 dev=0,5,0 cd_image.iso
```
- CD-RW에 기록할 경우 blank=blanktype 옵션을 추가해서 기존의 내용을 지우고 이미지를 구어야 한다.
- 기존 CD의 내용을 지우는 옵션
  - 아래 내용 참고

## CD-R 명령어

- cdrecord
  - 공CD-R에 이미지를 굽는 명령어
  - SCSI 방식과 ATAPI 방식의 CD 레코더를 지원
  - example
    - ISO-9660 파일시스템 형식의 데이터 이미지를 SCSI 타겟ID 5에 물려있는 레코더를 이용해서 구울 경우,

```
# cdrecord -v speed=2 dev=0,5,0 cd_image.iso
```

 이런 식으로 명령을 주면 된다.
  - CD-RW에 기록할 경우 blank=blanktype 옵션을 추가해서 기존의 내용을 지우고 이미지를 구어야 한다.
  - 기존 CD의 내용을 지우는 옵션

blanktype 값	설 명
help	사용 가능한 값들을 모두 보여준다.
all	디스크 전체를 완전히 지운다. 시간이 오래 걸림
fast	디스크를 빠르게 지운다.
track	하나의 트랙을 지운다.
unreserve	예약된 트랙의 예약을 해지한다.
trtail	트랙의 끝부분을 지운다
unclose	마지막 세션을 닫지 않도록 한다.
session	마지막 세션을 지운다.

## 9.3 리눅스 백업 및 복원 툴

리눅스에서는 다양한 명령행 백업 툴을 지원한다.

리눅스 백업 및 복원 툴의 종류

- dump
- rdump
- restore
- tar
- cpio
- afio 등

## dump 유틸리티

- 인자로 주어진 파일시스템을 지정된 dump 레벨로 백업
- inode 기반으로 동작
- 모든 디렉토리를 백업한 후에 파일들을 백업함
- dump 명령어 사용법
  - 형식

```
dump [-level] [-b blocksize] [-B records] [-f file] [-u] directory
```

- 각 옵션의 의미 : 아래내용 참고

## dump

- 인자로 주어진 파일시스템을 지정된 dump 레벨로 백업
- inode 기반으로 동작
- 모든 디렉토리를 백업한 후에 파일들을 백업함
  - 디렉토리들을 inode 값에 따라 작은값부터 큰값 순으로 백업하고, 그 후 파일들도 같은 순서로 백업한다.
- dump 명령어 사용법
  - 형식

```
dump [-level] [-b blocksize] [-B records] [-f file] [-u] directory
```
  - 각 옵션의 의미
    - level : 증진 백업 레벨. 레벨이 0일 경우 완전 백업을 의미
    - b blocksize : dump 블록 크기(바이트 단위)
    - B records : dump 블록 단위로 나타낸 백업 테이프 크기
    - f file : 백업이 저장될 파일명
    - u : 백업 내역을 /etc/dumpdates라는 파일에 저장하도록 함(반드시 필요)
    - directory : 백업할 디렉토리명이나 파일시스템

## restore 유틸리티

- dump로 받은 백업을 풀어내는 유틸리티
- 복원작업
  - 대화식 모드 : 원하는 파일들을 지정해서 복원
  - 비대화식 모드 : 백업 전체를 복원
- 대화식 모드로 복원하는 방법
  - example

```
# restore -i -f /dev/nst0
```

-i 옵션 : restore 명령을 대화식 모드로 실행
  - 이러한 방식으로 실행 후 restore> 프롬프트가 뜬 상태에서 add 복원할 디렉토리 명을 실행 후 extract 명령을 주면 해당 디렉토리가 복원이 된다.

## restore 유틸리티

- dump로 받은 백업을 풀어내는 유틸리티
- 복원작업
  - 대화식 모드 : 원하는 파일들을 지정해서 복원
  - 비대화식 모드 : 백업 전체를 복원
- 대화식 모드로 복원하는 방법
  - example

```
# restore -i -f /dev/nst0
```

-i 옵션 : restore 명령을 대화식 모드로 실행
  - 이러한 방식으로 실행 후 restore> 프롬프트가 뜬 상태에서 ‘add 복원할 디렉토리 명’ 을 실행 후 extract 명령을 주면 해당 디렉토리가 복원이 된다.

```
restore> add directory_name
```

```
restore> extract
```



## restore 유틸리티

- 비대화식 모드로 복원
    - 옵션 -t를 이용하여 테이프에 들어있는 파일들의 목록을 복원
    - 특정 파일을 복원하려면 아래의 예처럼 -x 옵션을 사용
- ```
# restore -x /usr/local/bin -f /dev/nst0
```

## tar

- 오래된 백업/아카이브 툴이지만 이식성이 제일 뛰어난 툴
- 근본적으로 에러 복구 기능에 결점이 있음
- tar의 기본적인 기능과 명령어 그리고 한정자 사용 등에 대해서는 뒷장을 참고

## restore 유틸리티

- 비대화식 모드로 복원
    - 옵션 -t를 이용하여 테이프에 들어있는 파일들의 목록을 복원
    - 특정 파일을 복원하려면 아래의 예처럼 -x 옵션을 사용
- ```
# restore -x /usr/local/bin -f /dev/nst0
```
- 테이프에 담겨있는 모든 파일들의 목록을 보려면 아래와 같이 실행하면 된다.
- ```
# restore -t -f /dev/nst0
```

## tar

- 오래된 백업/아카이브 툴이지만 이식성이 제일 뛰어난 툴
- 근본적으로 에러 복구 기능에 결점이 있음
  - 만약 tar 파일이 깨져 있으면 깨진 부분 근처로 수백킬로바이트가 넘는 데이터를 살릴수 없고 그 뒷부분부터 다시 데이터를 복원할 수 있다.
  - tar 압축 백업 파일이 깨진 경우는 복원할 방법이 없다.
    - 이유 : tar 압축 아카이브는 하나의 큰 압축 이미지 형태로 만들어지기 때문

## tar

- 압축기능이 내장되어 있지 않으며, 외부 유틸리티를 이용하여 압축기능을 제공
- 압축 아카이브의 단점
  - ① 에러 복구 기능 결점
  - ② 특정 파일을 풀어나기 위해 전체를 압축해제 후 검색
  - ③ 버퍼링(buffering) 기능이 거의 없음
- tar 실행할 때는 반드시 한가지 명령과 하나 이상의 한정자를 지정해 주어야 함
- tar 주로 사용되는 명령어와 한정자는 아래 내용을 참고

## tar

- 압축기능이 내장되어 있지 않으며, 외부 유틸리티를 이용하여 압축기능을 제공
- 압축 아카이브의 단점
  - ① 에러 복구 기능 결점
  - ② 특정 파일을 풀어나기 위해 전체를 압축해제 후 검색
  - ③ 버퍼링(buffering) 기능이 거의 없음
- tar 실행할 때는 반드시 한가지 명령과 하나 이상의 한정자를 지정해 주어야 함
- tar 사용시 주로 사용되는 명령어

| 명령어                 | 단축어 | 기 능                            |
|---------------------|-----|--------------------------------|
| --create            | c   | 아카이브를 생성                       |
| --concatenate       | A   | 아카이브에 tar 파일들을 추가              |
| --append            | r   | 아카이브 파일들을 추가                   |
| --update            | u   | 파일이 아카이브에 있는 것보다 최신이면 아카이브에 추가 |
| --diff 또는 --compare | d   | 디스크의 파일들과 아카이브를 비교             |
| --list              | t   | 아카이브의 내용을 나열                   |
| --extract 또는 --get  | x   | 아카이브로부터 파일들을 복원                |

- tar 사용시 주로 사용되는 한정자

| 한정자                       | 단축어 | 기 능                                            |
|---------------------------|-----|------------------------------------------------|
| --directory dir           | C   | 작업을 수행하기 전에 디렉토리 dir로 이동                       |
| --file [aaa:]bbb          | f   | aaa라는 컴퓨터의 bbb라는 파일을 아카이브 파일로 지정               |
| --listed-incremental file | g   | 증진 백업이나 복원을 수행하며, file을 이전에 아카이브한 파일들의 목록으로 이용 |
| --one-file-system         | 1   | 단 하나의 파일시스템(파티션)을 백업하거나 복원                     |
| --multi-volume            | M   | 여러 개의 테이프에 걸쳐있는 아카이브를 생성하거나 풀어냄                |
| --tape-length N           | L   | N 킬로바이트 이후에 테이프를 교환                            |
| --same-permissions        | p   | 퍼미션 정보를 보존                                     |
| --absolute-paths          | P   | 절대 경로명 이용(파일명의 제일 앞의 /를 유지)                    |
| --verbose                 | v   | 백업하거나 복원하는 파일들 목록을 나열                          |
| --verify                  | W   | 아카이브를 생성한 후 다시 검증 확인                           |
| --exclude file            | 없음  | 아카이브로부터 file을 풀어냄                              |
| --exclude-from file       | X   | file 속에 나열된 파일들을 제외하고 풀어냄                      |
| --gzip 또는 --ungzip        | z   | 아카이브를 gzip/ungzip을 통해 압축 처리                    |

- tar 명령어를 이용하여 한 컴퓨터의 데이터를 모두 테이프에 백업하는 예

```
# tar --create --verbose --gzip --one-file-system --same-permissions --file /dev/st0 / /home /usr/local
```

- 위에 명령어를 명령 축약어를 통해 간단하게 아래와 같이 표현할 수 있다.

```
# tar cvz1pf /dev/st0 / /home /usr/local
```

## cpio 유틸리티

- cp 명령과 입출력의 I/O를 합쳐서 지은 이름
- 특수 파일을 백업하거나 복원할 수 있음
- cpio 아카이브에서 파일이나 디렉토리를 골라서 풀어내는 기능 지원
- 표준 입력으로부터 입력받고 결과를 표준 출력으로 내보내는 기능 지원
- 옵션
  - -o : 아카이브를 생성
  - -i : 아카이브를 풀어냄
  - -t : 아카이브의 내용을 나열함

## cpio 유틸리티

- cp 명령과 입출력의 I/O를 합쳐서 지은 이름
- 특수 파일을 백업하거나 복원할 수 있음
- dump와 tar의 장점을 갖고 있음
- cpio 아카이브에서 파일이나 디렉토리를 골라서 풀어내는 기능 지원
- 표준 입력으로부터 입력받고 결과를 표준 출력으로 내보내는 기능 지원
- 옵션
  - -o : 아카이브를 생성
  - -i : 아카이브를 풀어냄
  - -t : 아카이브의 내용을 나열함
- example
  - 현재 디렉토리에 있는 파일들을 sample.cpio라는 이름으로 묶는 경우 아래와 같이 한다.  

```
# find . | cpio -o > sample.cpio
```
  - 아카이브를 현재 디렉토리 풀어내려면 아래와 같이 한다.  

```
# cpio -i < sample.cpio
```
  - 아카이브에 들어있는 파일들의 목록을 볼 경우 아래와 같이 한다.  

```
# cpio -t < sample.cpio
```

## afio

- 파일단위로 데이터를 압축
- 장점
  - ① 에러 복구 기능이 뛰어남
  - ② 더블 버퍼링 기능 제공
- afio는 cpio와 같이 아카이브할 파일들의 목록을 표준 입력으로부터 읽기 때문에 일반적으로 find 명령과 함께 사용함
- 옵션
  - 백업할 때 : -o
  - 아카이브를 풀어낼 때 : -i
  - 아카이브의 내용을 볼 때 : -t

## afio

- 파일단위로 데이터를 압축
- 장점
  - ① 에러 복구 기능이 뛰어남
  - ② 더블 버퍼링 기능 제공
- afio는 cpio와 같이 아카이브할 파일들의 목록을 표준 입력으로부터 읽기 때문에 일반적으로 find 명령과 함께 사용함
- 옵션
  - 백업할 때 : -o
  - 아카이브를 풀어낼 때 : -i
  - 아카이브의 내용을 볼 때 : -t
- 사용하는 예
  - 현재 디렉토리 파일들을 sample.afio라는 이름의 아카이브로 묶을 경우 아래 예와 같이 입력한다.  

```
# find . | afio -o /root/sample.afio
```
  - 아카이브를 현재 디렉토리에 풀어내려면 -i 옵션을 사용하여 아래 예와 같이 한다.  

```
# afio -i /root/sample.afio
```

- 아카이브에 들어있는 파일들의 목록을 볼때는 -t 옵션을 사용하여 아래 예와 같이 한다.

```
# afio -t /root/sample.afio
```

- 압축 아카이브를 생성하려면 -Z 옵션을 추가하여 아래 예와 같이 한다.

```
# afio . | afio -oZ /root/sample.afio.gz
```

## 9.4 운영체제 백업

### 최소 백업

- 디스크 장애 등의 이유로 하드웨어를 새로 설치했을 경우, 리눅스를 다시 설치한 후 테이프나 CD-R 등의 작은 백업으로부터 설정 파일들을 복원해서 원래 이용하던 시스템과 거의 같은 상태로 만드는 백업 방법을 의미
- 최소 백업의 장점
  - 시간상 빠른 백업이 가능하고, 테이프가 적게 든다.
  - 복잡한 장애 복구를 하지 않고 시스템을 복원할 수 있다.
- 최소 백업의 단점
  - 운영체제 업데이트라던가 별도의 소프트웨어 설치를 다시 해야 하며, 복원 후 예전처럼 설정이 잘 되었는지 꼼꼼히 확인이 필요하다.

### 최소 백업

- 디스크 장애 등의 이유로 하드웨어를 새로 설치했을 경우, 리눅스를 다시 설치한 후 테이프나 CD-R 등의 작은 백업으로부터 설정 파일들을 복원해서 원래 이용하던 시스템과 거의 같은 상태로 만드는 백업 방법을 의미
- 최소 백업을 하기 위해 사용자 파일, 환경설정 파일, 시스템 관련 설정 파일들이 모여있는 /etc, /home, /usr/local, /var 등의 특정 디렉토리들을 백업을 하면 된다.
  - 위 4개의 디렉토리를 tar 명령과 SCSI 테이프 드라이브를 이용해 백업한다면 아래와 같은 작업으로 쉽게 할 수 있다.

```
# tar cvplf /dev/st0 /etc /home /usr/local /var
```

- 최소 백업의 장점
  - 시간상 빠른 백업이 가능하고, 테이프가 적게 든다.
  - 복잡한 장애 복구를 하지 않고 시스템을 복원할 수 있다.

- 최소 백업의 단점
  - 운영체제 업데이트라던가 별도의 소프트웨어 설치를 다시 해야하며, 복원 후 예전처럼 설정이 잘 되었는지 꼼꼼히 확인이 필요하다.



## 전체백업

- 컴퓨터에 있는 모든 것을 백업하는 방법
- 일반적으로 / 디렉토리를 백업
- 단, /proc 파일시스템은 컴퓨터의 상태를 담아놓은 가상 파일시스템이기 때문에 이것까지 복원하게 되면 다운되는 등의 문제가 발생
- 전체 백업의 장점
  - 백업을 수행할 때 상태 그대로 복원이 됨
- 전체 백업의 단점
  - 시간이 오래 걸리며, 테이프도 많이 필요함
  - 장애가 발생한 후 전체 백업으로부터 복원하려면 사전 계획과 준비가 필요

## 전체백업

- 컴퓨터에 있는 모든 것을 백업하는 방법이다.
- 일반적으로 / 디렉토리를 백업한다.
- 단, /proc 파일시스템은 컴퓨터의 상태를 담아놓은 가상 파일시스템이기 때문에 이 파일 시스템까지 복원하게 되면 다운되는 등의 문제가 발생하기 때문에 /proc 파일시스템은 전체백업에서 제외해야 한다.
- 전체 백업의 장점
  - 백업을 수행할 때 상태 그대로 복원이 된다. 즉, 복원 후 프로그램을 업데이트한다거나 설정 파일을 다시 고쳐줘야 하는 번거로움이 없다.
- 전체 백업의 단점
  - 시간이 오래 걸리며, 테이프도 많이 필요하다.
  - 장애가 발생한 후 전체 백업으로부터 복원하려면 사전 계획과 준비가 필요하다.
- 전체 백업을 하는 방법
  - 기본적으로 / 디렉토리를 백업하고, 제외할 파일이나 디렉토리는 별도로 명시한다.
  - tar 명령과 SCSI 테이프 드라이브를 이용해서 전체 백업을 한다면 아래 예와 같이 한다.

```
# tar cvpf /dev/st0 / --exclude=proc
```

## 9.5 장애 복구 기법

### 장애 복구 기법이 필요한 경우

- 백업을 이용한 데이터 복원과 상관없는 문제들이 발생할 경우
  - LILO나 GRUB과 같은 부트 로더가 동작하지 않는 경우
  - 새로 컴파일한 커널로 바꿀 경우 부팅이 안되는 상황인 경우

### 장애 복구 기법의 장점

- 시스템이 멈추기 전에 로그인해서 문제를 진단하고 해결하거나 중요한 파일들을 뽑아 올 수 있음

### 장애 복구 기법의 종류

- 싱글유저 모드로 부팅하기
- 응급 부트 디스크 이용하기
- 복구 모드를 이용하기

## 장애 복구 기법

### ■ 싱글유저 모드

- 싱글유저 모드로 부팅하면 컴퓨터가 실행 레벨 1로 부팅된다.
- 시스템이 부팅된 후에 로그인 할 수 없는 장애일 경우
  - LILO 프롬프트에서 linux single 이라고 입력
- 이 모드는 아무 데몬도 띄우지 않기 때문에 데몬 때문에 발생한 문제라면 싱글유저 모드로 로그인하여 고칠 수 있다.
- 루트 암호를 잊어버렸을 경우나 커널 이미지를 새로 만들고 LILO와 같은 부트로더를 실행하는 것을 잊어먹었을 때 요긴하게 쓰임
  - 루트 권한이 필요한 시스템 유지 보수 작업 수행이 가능

## 장애 복구 기법

### • 싱글유저 모드

- 싱글유저 모드로 부팅하면 컴퓨터가 실행 레벨 1로 부팅된다.
- 시스템이 부팅된 후에 로그인 할 수 없는 장애일 경우
  - LILO 프롬프트에서 linux single 이라고 입력
- 이 모드는 아무 데몬도 띄우지 않기 때문에 데몬 때문에 발생한 문제라면 싱글유저 모드로 로그인하여 고칠 수 있다.
- 루트 암호를 잊어버렸을 경우나 커널 이미지를 새로 만들고 LILO와 같은 부트로더를 실행하는 것을 잊어먹었을 때 요긴하게 쓰임
- 루트 권한이 필요한 시스템 유지 보수 작업 수행이 가능

## 장애 복구 기법

### 응급 부트 플로피 이용

- 리눅스 설치하는 과정 중에 생성하거나 /sbin/mkbootdisk 명령을 실행해서 생성한 부트 이미지
- 해당 시스템에 적합하도록 맞춰진 부트 데이터와 루트 파일시스템이 담겨 있음
- 부트 플로피는 시스템을 부팅하면서 기존 시스템에 들어있는 파일들을 즉, 부트로더 파일, 초기화 파일, 커널 이미지를 필요로 한다.
- 만약 부팅이 안되는 경우 위 파일이 문제라면 부트 플로피를 이용해서 부팅이 불가능 하다. 이럴 경우 다음에 나올 복구 모드를 이용해야 한다.

## 장애 복구 기법

- 복구 모드
  - 리눅스 배포판이 지원하는 복구 모드로 부팅시 작지만 완전한 리눅스 시스템이 뜬다.
  - 사용 가능한 작업
    - 컴퓨터의 디바이스 마운트
    - 설정 파일 편집
    - 파일시스템 검사
    - 기타 장애 보수 작업 등
  - 복구 모드를 사용해야 할 상황
    - 부트로더(LILO 또는 GRUB)가 다른 운영체제에 의해 지워졌을 경우 등

## 장애 복구 기법

- 복구 모드
  - 싱글유저 모드 부팅이나 부트 디스크를 이용한 부팅 모두 시스템이 이미 들어있는 파일을 필요로 하는데 반해서, 복구 모드는 그런 방법으로도 고칠 수 없는 경우를 해결할 수 있는 유일한 방법이다.
  - 리눅스 배포판이 지원하는 복구 모드로 부팅시 작지만 완전한 리눅스 시스템이 뜬다.
  - 사용 가능한 작업
    - 컴퓨터의 디바이스 마운트
    - 설정 파일 편집
    - 파일시스템 검사
    - 기타 장애 보수 작업 등
  - 복구 모드를 사용해야 할 상황
    - 부트로더(LILO 또는 GRUB)가 다른 운영체제에 의해 지워졌을 경우
    - 하드웨어에 문제가 생겨서 정상 부팅이 안되는 경우
    - /etc/fstab 같은 파일에 잘못된 수정을 하여 부팅이 안될 경우







































































































































































































































