제4장 시스템 설정

1. X window System

2. RPM 활용하기

3. LILO (Linux boot Loader)

4. etc 디렉토리 아래의 주요 시스템 설정파일

1. X window System

(1) X Window의 이해

① X Window란?

X 윈도우는 MIT(Massachusetts Institute of Techonology)에서 개발된 그래픽 사용자 환경입니다. 여러 면에서 X 윈도우는 이전의 MS 윈도우와 비교되곤 합니다. DOS 위에서 작동하는 MS의 윈도우와 마찬가지로, X 윈도우 역시 UNIX나 UNIX-Like (리눅스가 대표적이죠.) 운영체제 위에서 작동합니다. 하지만, X 윈도우는 화면에 그림을 뿌려주는 간단한 프로그램입니다. 대신 윈도우 매니저(WindowMaker, Sawfish, Enlightenment등)가 화면에 보여지는 것을 제어하고, 많은 옵션들과 기능들을 제공합니다. 윈도우 매니저에 대해서는 뒷부분에서 다룰 것입니다.
근래의 리눅스 모든 배포판들은 XFree86이라고 불리 우는 공개 X 서버 프로그램을 대부분 제공합니다. XFree86 은 XFree86 프로젝트 비영리 단체에서 관리하고 있습니다. (http://www.xfree86.org 에 접속하면 XFree86 에 대해 더 많은 정보를 얻을 수 있습니다.) XFree86 은 바이너리와 보조파일과 라이브러리, 그리고 설치에 필요한 툴까지 모든 것을 포함하고 있습니다. 그리고 많은 플랫폼(V/i386/386 BSD, 기타 Intel x86 유닉스 등)에서 작동하고 있습니다.
와우리눅스 7.1 파란 릴리즈에서는 XFree86 4.0.3 버전을 제공하고 있습니다.

② X 클라이언트/서버 모델

X 서버 프로그램은 화면과 입력장치(키보드,마우스)등을 제어합니다.
startx 라고 명령을 내림으로써 실행할 수 있습니다.
X 클라이언트는 X 서버의 입출력에 이용할 수 있습니다. 데스크탑과 윈도우매니저 또한 X 서버의 클라이언트일 뿐입니다. X의 이 훌륭한 특징은 각 클라이언트들이 네트워크상의 어떠한 머신에서도 실행될 수 있다는 것을 말하기도 합니다. (X 서버가 동작하고 있는 머신도 포함해서) X 서버에 의해 관리되는 원격 웍스테이션은 각각의 입출력을 가지고 있습니다. 원격 머신에 접근하기 위해서는, Remote X Apps mini-HOWTO를 참고하세요.
http://metalab.unc.edu/LDP/HOWTO/mini/Remote-X-Apps.html
http://kldp.org/HOWTO/mini/html/Remote-X-Apps/Remote-X-Apps.html

③ 하드웨어 정보 얻기

XFree86 윈도우 시스템을 제대로 설치하기 위해서는 가지고 있는 하드웨어 대한 정보를 가지고 있어야 합니다.

XFree86 프로젝트에서는 인텔 기반의 최소 8MB(스왑을 포함해서 총 16MB 정도)의 램을 가진 486 이상을 권장하고 있습니다.
RAM을 늘리는 것은 성능을 향상시키는데 도움이 됩니다. 그리고, S3 (리눅스에서 잘 지원하는)와 같이 가속기능을 가진 비디오 카드를 사용하는 것도 좋은 방법입니다. XFree86 설정하는데 꼭 필요한 시스템 정보는 비디오카드, 모니터, 마우스, 키보드 정도입니다.

① 비디오 카드

새로운 비디오 카드를 사려고 한다면, XFree86 프로젝트에서 지원하는 것인지 확인해야 합니다. 근래의 많은 비디오 카드와 칩셋들이 지원되기는 하지만, 먼저 지원되는 카드 목록을 확인해 보는 것이 좋습니다. 현재 지원되고 있는 카드와 칩셋에 대한 목록은 *http://www.xfree86.org/4.0/Status.html* 에서 보실 수 있습니다.
새로운 카드나, 완제품에서 제공되는 문서는 칩셋과 비디오 메모리 량에 대한 정보를 가지고 있어야 합니다. 이러한 정보가 없다면, 제조업체의 웹사이트에서 제품 정보를 확인하시거나, 직접 연락을 취하십시오. 다른 방법은 XFree86 에서 제공하는 SuperProbe라는 유틸리티를 사용하는 것입니다. SuperProbe는 루트로 실행해야 하고, 알려진 칩셋에 대해서 적당한 결과를 알 수 있습니다. 또한 많은 칩셋들의 비디오 메모리를 알 수 있습니다.

② 모니터

X 설정을 하기 위해 XF86config나 Xconfigurator등의 유틸리티를 사용했다면, 리스트에 있는 모델 중 맞는 것을 고르기만 하면 됩니다. 하지만, 손으로 /etc/xll/XF86config 파일을 편집한다거나, 위 유틸리티의 목록에 모니터가 없다면 수평/수직 주파수(refresh도 역시)를 알고 있어야 합니다. 이러한 것들은 몇몇 곳에서 사용되기도 합니다. 대부분은 사용자 설명서나 케이스에 표시되어 있습니다.
※ 지원되는 refresh 보다 높게 설정한다면, 모니터에 무리가 갈 수 있습니다. 주의하세요.

④ X 윈도우 시작하기

최근 대부분의 배포판들은 시스템 부팅 후, 바로 X 윈도우가 실행되도록 선택할 수 있습니다. X 윈도우가 뜨도록 선택했다면, 사용자는 그래픽컬한 로그인 화면을 볼 수 있습니다. 만일 선택하지 않았다면, 커맨드 라인에서 X 윈도우를 사용하기 위해서 직접 명령을 내려야 합니다. X 세션을 준비하기 위해서, startx 명령을 실행합니다. startx 명령에 옵션이 필요하다면 다음과 같이 실행 시 옵션을 주면 됩니다. (예는 색상 수를 조절하는 옵션을 사용한 것입니다.)
#startx — -bpp 16 (X 서버의 색을 16bit/pixel 을 기본값으로 시작합니다)
실행할 클라이언트를 정하기 위해서, startx 스크립트는 각 사용자 홈디렉토리의 .xinitrc 파일을 살펴봅니다. 이 파일에서 클라이언트가 정의되어 있지 않다면, 스크립트는 xinit 라이브러리의 .xinitrc 를 참조해서 클라이언트를 실행합니다. 실제로 이 스크립트는 xinit 명령을 실행하기 위한 프론트 엔드입니다. xinit은 X 세션 설정에 관한 몇 가지 옵션들을 가지고 실행되는 다소 복잡한 명령입니다. xinit에 대하여 좀 더 자세히 알고 싶으신 분은 man 페이지를 참조하시기 바랍니다.

⑤ xvidtune

X 윈도우를 제대로 설치했더라도, 화면이 모니터에 정확하게 보이지 않을 때도 있습니다. 만일 모니터가(랩탑 같은 경우) 화면의 크기와 위치 조절하는 기능이 없다면, 다른 방법으로 조절해야 합니다. xvidtune 이 이러한 일을 할 수 있는 도구입니다. xvidtune 유틸리티는 많은 배포판에 포함되어 있습니다. 커맨드라인에서 xvidtune 이라고 치면 실행하실 수 있습니다.

xvidtune 유틸리티는 세세한 부분까지 조절이 가능합니다. 화면의 좌/우/상/하 와 높이와 폭의 조절은 물론, 사용자가 시작과 종료시 수평/수직의 동기화까지도 가능합니다.

※ 항상 1에서부터 작은 단위로 조절을 하고, 조절한 값을 적용하기 전에 TEST 버튼을 이용해 테스트를 하십시오. 이 프로그램은 낮은 레벨까지 건들이는 성격이 있기 때문에, 주의하지 않는다면 모니터와 VGA카드를 망가뜨릴 수도 있습니다.

⑥ X에서 글꼴의 설치와 관리

처음 리눅스 시스템을 만지는 사람들에게 X에서 사용하는 글꼴에 대한 부분은 어려울 수 있습니다. 다행히도 많은 사람들의 필요에 의해서 기본적 글꼴들은 제공됩니다. 다른 글꼴을 설치하려는 사람들을 위해서 몇 가지 옵션들을 제공 합니다. XFree86 은 비트맵, 트루타입, 타입1의 글꼴을 지원합니다. X에 포함되어 있는 글꼴 지원과 더불어, XFree86은 별개의 (또는 다수의) 글꼴 서버를 지원할 수 있습니다. 글꼴 서버는 XFree86 에서 사용할 수 있도록 폰트를 설치해 주는 백그라운드 프로세서입니다. 이런 글꼴 서버는 로컬 머신의 XFree86 의 글꼴과 네트워크상에서 운영되고 있는 다른 머신의 폰트들도 사용할 수 있도록 해줍니다.

① Font Path 이해하기

X 상에서 글꼴에 대해 이해한다는 것의 핵심은 Font Path 를 제대로 이해하는 것입니다.

Font Path를 설정하는 방법은 수동으로 XF86Config 파일을 수정하는 것과 xset 명령을 이용해 글꼴들이 있는 디렉토리를 추가해 주는 방법이 있습니다.

주요한 Font Path 는 /etc/X11/XF86Config 파일을 통해 관리됩니다. 이 파일은 X 에서 글꼴들을 사용할 수 있도록 글꼴들이 있는 디렉토리를 참조할 수 있도록 해줍니다. XF86Config 파일을 보면 다음과 유사한 섹션이 존재하는 것을 볼 수 있습니다.

```
# Multiple FontPath entries are allowed (they are
# concatenated together)
FontPath "/usr/X11R6/lib/X11/fonts/misc:unscaled"
FontPath "/usr/X11R6/lib/X11/fonts/75dpi:unscaled"
FontPath "/usr/X11R6/lib/X11/fonts/100dpi:unscaled"
....
....
```


위의 Path 들은 글꼴이 존재하고 있는 디렉토리 경로입니다. 하지만 근래의 레드햇 등의 배포판에서는 다음과 같이 Font Path 설정이 되어 있습니다.

```
# Multiple FontPath enties are allowed (they are
# concatenated together)
# By default, Red Hat 6.0 and later now use a font server
# independent of the X server to render fonts.
FontPath "unix/:-1"
```

위에서 보면, XFree86 과는 별도로 동작하는 글꼴 서버를 볼 수 있습니다. 와우리눅스 7.1의 경우, "unix/:7100"으로 되어 있습니다. 그리고, 이 글꼴 서버는 고유한 설정 파일을 가지고 있습니다. 글꼴 서버는 xfs 와 xfstt, 2가지가 유명합니다. 동일한 XF86Config 파일에서 글꼴 디렉토리를 직접 참조하는 것과, 글꼴 서버를 참조하도록 설정하는 것이 가능합니다.

② Font Path에 디렉토리 추가하기

Font Path를 추가하는 방법 중 하나는 xset 명령을 이용하는 것입니다. 만약 /usr/share/fonts/foo 가 글꼴 디렉토리라면 xset 명령을 다음과 같이 실행하면 Font Path가 추가됩니다.

```
# xset fp+ /usr/share/fonts/foo
# xset fp rehash
```

이렇게 하면 /usr/share/fonts/foo 디렉토리가 사용자의 Font Path에 추가가 되고, X 서버에게 새로 추가된 글꼴을 찾도록 알려주게 됩니다. 위의 명령은 단지 사용자 로그아웃 하기 전까지 글꼴을 사용할 수 있도록 해주는 xset 명령어를 이용하는 예입니다. 위 명령을 자동으로 하기 위해서는 사용자의 홈디렉토리에서 .Xsession 파일이나, .xinitrc 파일에 추가해 주어야 합니다. 모든 사용자에게 Font Path 를 추가해 주기 위해서는 XF86Config 파일에서 File Section 부분에 FontPath 를 추가해 주어야 합니다. (FontPath "/usr/share/fonts/foo" 와 같은 형식)

루트 사용자로 FontPath 를 추가하고 나서 FontPath 를 적용하기 위해서는 X 글꼴 서버를 재시작해야 합니다.

⑦ 글꼴 서버

글꼴 서버로 유명한 두개는 xfs 와 xfstt 라고 위에서 말했습니다. xfs 는 와우리눅스7.1을 포함하여 레드햇 계열의 배포판에서는 (맨드레이크, 터보 리눅스 등) 타입1 과 비트맵 폰트는 물론 트루타잎까지 지원하도록 컴파일 되어 있습니다. 다른 계열의 배포판은 xfs 를 사용하기는 하지만, 트루타잎 글꼴을 지원 않도록 컴파일되어 있습니다. 트루타잎 글꼴을 지원하지 않는 시스템에서는 트루타잎 글꼴을 사용하기 위해서 xfstt 를 설치해야 합니다. (혹은, xfs 의 소스를 받아서 트루타잎 글꼴을 지원하도록 재컴파일 하면 됩니다.)

xfs 는 XFree86 프로젝트의 코드를 기초로 합니다. 과거에는 XFree86 에서 글꼴을 지원하도록 같은 코드가 제공되었지만, 이제는 독립서버로서 작동하게 되었습니다.
xfs 에서 Font Path 는 /etc/X11/fs/config 파일의 제어를 받습니다. 레드햇 계열의 배포판에서는 설정파일을 선택하거나, fontpath(글꼴 경로)를 더해주거나 제거할 때, chkfontpath 유틸리티를 사용할 수 있습니다. 자세한 정보는 man 페이지를 이용하십시오.

⑧ 윈도우 매니저/데스크탑 환경

윈도우 매니저는 무슨 일을 할까?
윈도우 매니저는 미리 정의되어 있는 몇몇 형식으로 프로그램 윈도우를 배치 하거나, 사용자가 윈도우를 이동, 크기 조절, 아이콘화 하기 등을 할 수 있도록 해 줍니다. 윈도우 매니저를 선택하는 것은 X 환경에서의 look and feel 을 결정하는 것입니다.
리눅스에서 사용할 수 있는 윈도우 매니저는 많습니다. 그 중 유명한 것들은 다음과 같습니다.

- Fvwm, fvwm2, fvwm95 : 한때는 유명했지만, 지금은 구식이 된 윈도우 매니저입니다. Fvwm95 는 fvwm2 를 hack 해서 만들었다고 합니다.
  공식 사이트 : *http://www.fvwm.org*
- Enlightenment : 극단적인 설정이 가능한 윈도우 매니저입니다.
  공식 사이트 : *http://www.enlightenment.org*
  한국 사용자 : *http://e.sarang.net*
- Afterstep : NeXT step 환경을 기반으로 만들어졌습니다.
  공식 사이트 : *http://afterstep.org*
- Windowmaker : GNUstep 응용프로그램의 추가적인 통합 지원을 위하여 설계되었습니다. 이것은 NeXTSTEP GUI 의 우아함을 흉내 내고자 하였습니다. 비교적 빠르고, 기능이 풍부하고 설정도 쉽고, 사용하기에 쉽습니다.
  공식 사이트 : *http://www.windowmaker.org*
  참고 사이트 : *http://gnustep.org*
  한국 사용자 : *http://windowmaker.sarang.net/*
- KDE : 윈도우 매니저, 파일 관리자, 패널, 제어 센터, 그리고 많은 최신 데스크탑 환경을 구현하기 위해서 많은 컴퍼넌트들을 가지고 있습니다. KDE 의 최신 버전은 Mac 과 유사한 인터페이스를 제공합니다. KDE는 차세대 리눅스 인터페이스가 되기 위해서 노력하고 있습니다. 레드햇, 수세, 칼데라 시스템에서 배포판에 KDE를 포함하고 있습니다.
  공식 사이트 : *http://www.kde.org*
  한국 사용자 : *http://kde.kldp.org/*
- GNOME : GNOME 은 GNU Network Object Model Environment를 말합니다. GNOME 프로젝트는 완벽하고, 사용자에게 친화적인 인터페이스를 무료로 제공하고자 하고 있습니다. GNOME 은 GNU 프로젝트의 일환으로, OpenSource 정의를 따르는 무료 소프트웨어 입니다. GNOME은 일관성 있는 Look and Feel 을 공유하는 작은 유틸리티와 큰 응용 프로그램으로 구성되어 있습니다. GNOME은 모든 GNOME 응용 프로그램이 사용하는 GUI 툴킷으로서 GTK+ 를 사용합니다. 이것은 Mac 이나, MS-Windows 환경과 비슷하도록 설정할 수 있습니다. 레드햇은 GNOME 개발 프로젝트의 후원자입니다. 그리고, GNOME과 함께 하기를 원하고 있습니다. 매우 뛰어난 유연성을 가진 최근의 리눅스 인터페이스로서의 특징을 가지고 있습니다. GNOME은 윈도우 매니저를 가지고 있지는 않지만, 대체적으로 Enlightenment 윈도우 매니저를 대신 가지고 있습니다. (GNOME 1.2 에서는 Sawfish가 기본으로 사용됩니다.) 최신의 RPM은 다음에서 받으실 수 있습니다.
  공식 사이트 : *http://www.gnome.org*

이외에 IceWM, XFCE, Mwm, Olvwm 등의 윈도우 매니저가 있습니다.
윈도우 매니저의 개략적인 내용은 *http://www.PliG.org/xwinman/* 에서 보실 수 있습니다.
윈도우 매니저는 사용자가 자신만의 환경을 만들 수 있도록 설정파일을 이용할 수 있도록 합니다. 윈도우 매니저에 따라서, .〈매니저〉rc 등의 이름으로 사용자의 홈디렉토리에 위치하면 됩니다. 자세한 것은 각각의 매니저마다 문서를 참조하세요.

⑨ xdm

xdm 은 그래피컬한 사용자 로그인 서비스를 제공합니다. xdm 은 X 를 시작하고, 사용자 로그인이 가능하도록 합니다. 사용자가 로그인 할 때, 홈디렉토리의 .xsession (.xinitrc 대신)을 읽고, 시작 프로그램들과 윈도우 매니저 등을 실행합니다. 사용자가 윈도우 매니저에서 X 세션을 끝냈을 경우에는,
X 는 xdm 로그인 상태로 돌아가게 됩니다.
xdm은 사용자가 볼 수 있는 가장 처음의 인터페이스이기 때문에, 편리한 사용과 몇몇 곳에서 필요에 의한 커스터마이징이 쉽도록 설계되었습니다. xdm은 많은 옵션들이 있습니다. xdm 매뉴얼을 참조하세요.
xdm 시작하기 : xdm 을 부트 후 시작하도록 하려면, /etc/inittab 파일에서 initdefault 를 5로 변경하면 됩니다.

```
#vi /etc/inittab
...
id:5:initdefault:
...
```

※ 팁 : 로그인 상자에서 패스워드를 넣고 Enter 대신 Ctrl + Enter를 치면, xdm 은 failsafe 모드로 들어가게 됩니다. 홈디렉토리의 .xsession 을 읽지 않고 xterm 을 실행시키고, 홈디렉토리의 .xsession 파일이 잘못 되었더라도 사용자가 로그인할 수 있도록 하는데 효과적입니다.

(2) X Window 환경설정

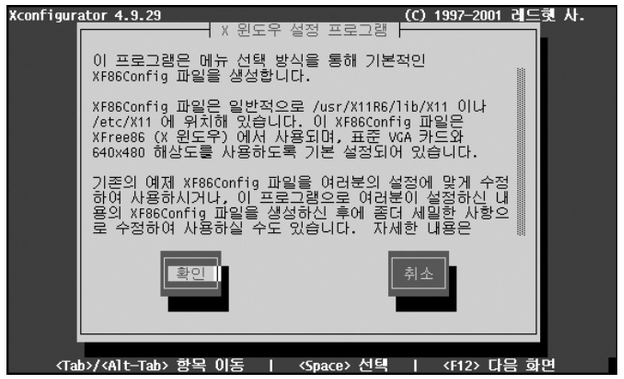
X 윈도우를 설정한다는 것은 다음 정보를 X 윈도우 시스템에게 알려주는 것입니다.

- 비디오 카드가 어떤 칩셋을 사용하며, 따라서 어떤 서버 프로그램을 사용할 것인가?
- 모니터의 수평/수직 주사 능력은 어떻게 되는가?
- 비디오 카드의 능력과 모니터의 능력에 의거하여 어떤 해상도에서 어느 색상 모드로 X 윈도우를 실행하길 원하는가?

이러한 X 윈도우의 설정은 인스톨시 결정하지 않더라도 배포판에서 Xconfigurator, xf86config, XF86Setup 툴을 제공하여 X 윈도우 설정을 돕고 있습니다.

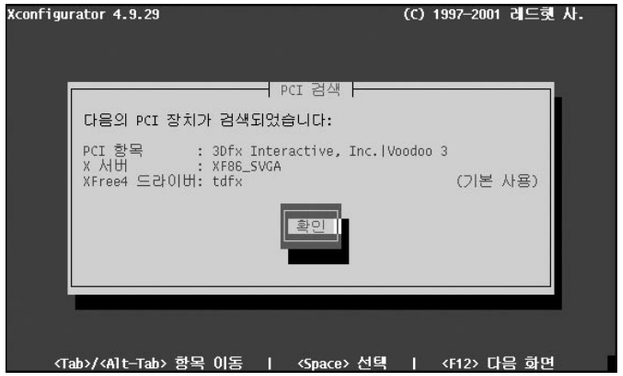
① Xconfigurator

Xconfigurator는 X 윈도우 설정을 돕기 위해 레드헷에서 제공하는 설정 툴로 xf86config나 xf86setup 툴에 비하여 직관적이며, 사용이 간편하여 많이 사용되어 지는 툴입니다.
실제 X 설정의 예를 보며, 살펴보도록 하겠습니다.



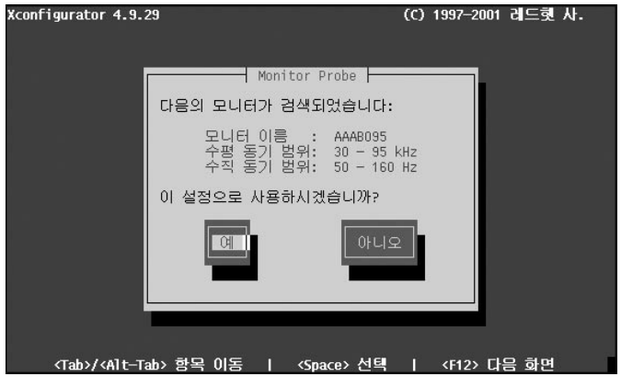
[그림 Ⅳ-1] 사용안내

Xconfigurator 명령을 실행하면, 위와 같이 간단한 설명과 함께 X 설정을 진행하겠음을 알립니다. 시간 여유가 되신다면 꼭 한번 읽어 보십시오. 그리고 확인 버튼을 눌러 다음을 진행합니다.



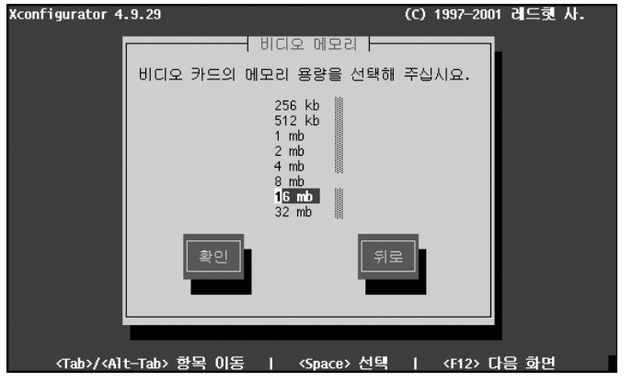
[그림 Ⅳ-2] 비디오카드 검색

그림에서 보는 바와 같이 자동으로 비디오 카드를 찾습니다. 만일 가지고 있는 정보에 없는 비디오 카드라면 엉뚱한 메시지를 보이거나 지원하지 못한다고 경고를 합니다. 이때는 해당 비디오 카드 사이트를 방문하여, Xfree86용 드라이버를 제공하는지 확인하시고, 제조사의 지시에 따라 설치하시면 됩니다.



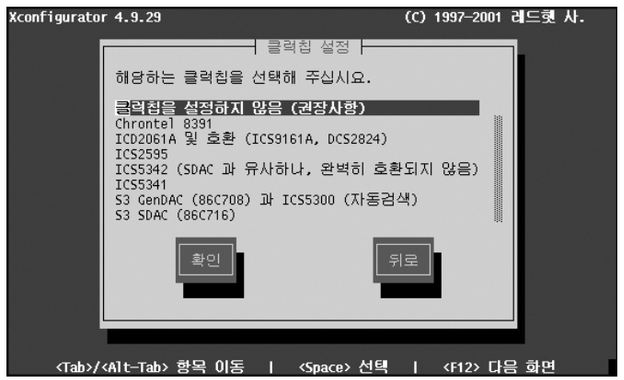
[그림 Ⅳ-3] 모니터 검색

모니터 설정 역시 자동으로 찾아주지만, 목록에 없거나 자동으로 찾지 못할 경우는 모니터 매뉴얼을 참고하여, 수직/수평 주파수 값을 입력해 주시면 됩니다.



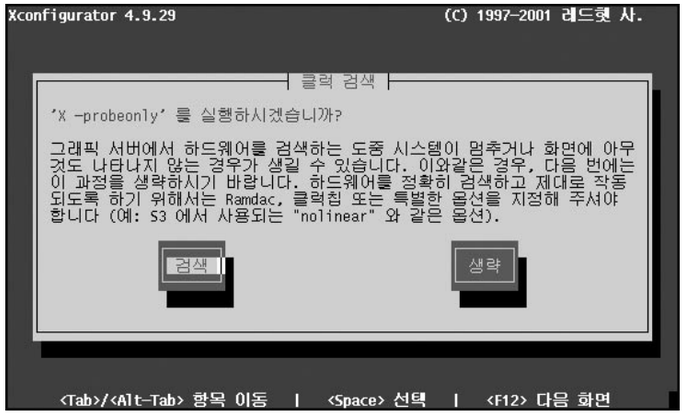
[그림 Ⅳ-4] 비디오 메모리 선책

비디오 카드 메모리를 선택하여 줍니다. 가능한 정확한 값을 입력하셔야 해상도 선택시 원하는 해상도를 사용할 수 있습니다.



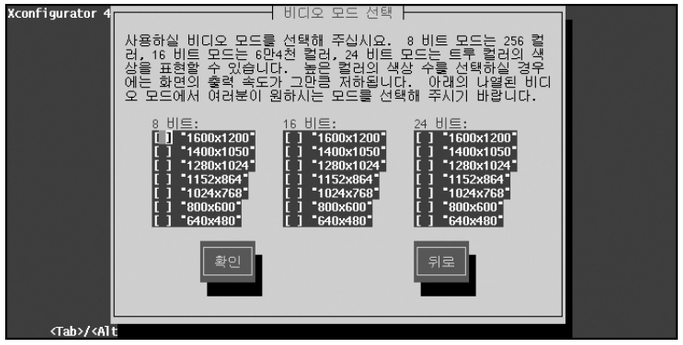
[그림 Ⅳ-5] 클럽칩 선택

클럭칩 셋팅에 관한 질문입니다. 잘 모르는 경우는 '클럭칩을 설정하지 않음' 을 선택하길 권장합니다. 리눅스의 X 윈도우는 자동으로 클럭칩을 설정할 수 있기 때문입니다.



[그림 Ⅳ-6] 비디오 모드 검사

X-probeonly를 통하여 그래픽 서버를 검색할 경우, 사용 가능한 해상도와 비트수를 정확히 출력해 줍니다. 하지만, 몇몇 그래픽 서버에서 문제가 될 수도 있으므로, 생략하셔도 좋습니다.

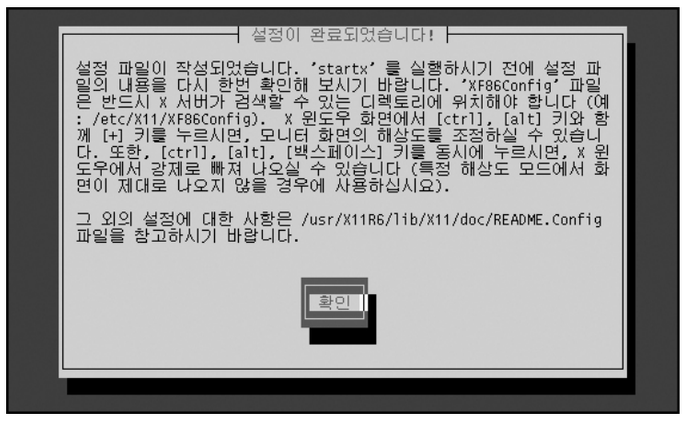


[그림 Ⅳ-7] 비디오 모드 선택

사용하길 원하는 해상도와 비트를 선택합니다. 다중 선택 후 해상도 전환이나 원하는 비트로 선택하여 X 서버 시작도 가능합니다.

[그림 Ⅳ-8] X 윈도우 설정 테스트

지금까지의 설정 내용을 바탕으로 X 윈도우 테스트를 진행합니다. 하지만 몇몇 그래픽 서버는 문제가 될 수 있으므로 생략하셔도 좋습니다.



[그림 Ⅳ-9] 설정완료

X 설정이 완료 되었음을 알립니다. 확인 버튼을 눌러 종료합니다.

2 xf86config

XFree86 에서 제공하는 툴로 [그림Ⅳ-10] 에서 보는 바와 같이 한행 한행 질문과 답을 통하여 X 윈도우를 설정해 가는 방식으로 구성되어 있습니다. 상당히 불편해 보이고, 어려워 보이지만 레드헷의 Xconfigurator가 제공되기 전까지는 아래의 툴을 사용하여 실제 X를 설정했었습니다.

```
You must indicate the horizontal sync range of your monitor. You can either
select one of the predefined ranges below that correspond to industry-
standard monitor types, or give a specific range.

It is VERY IMPORTANT that you do not specify a monitor type with a horizontal
sync range that is beyond the capabilities of your monitor. If in doubt,
choose a conservative setting.

    hsync in kHz; monitor type with characteristic modes
 1  31.5; Standard VGA, 640x480 @ 60 Hz
 2  31.5 - 35.1; Super VGA, 800x600 @ 56 Hz
 3  31.5, 35.5; 8514 Compatible, 1024x768 @ 87 Hz interlaced (no 800x600)
 4  31.5, 35.15, 35.5; Super VGA, 1024x768 @ 87 Hz interlaced, 800x600 @ 56 Hz
 5  31.5 - 37.9; Extended Super VGA, 800x600 @ 60 Hz, 640x480 @ 72 Hz
 6  31.5 - 48.5; Non-Interlaced SVGA, 1024x768 @ 60 Hz, 800x600 @ 72 Hz
 7  31.5 - 57.0; High Frequency SVGA, 1024x768 @ 70 Hz
 8  31.5 - 64.3; Monitor that can do 1280x1024 @ 60 Hz
 9  31.5 - 79.0; Monitor that can do 1280x1024 @ 74 Hz
10  31.5 - 82.0; Monitor that can do 1280x1024 @ 76 Hz
11  Enter your own horizontal sync range

Enter your choice (1-11):
```

[그림 Ⅳ-10] xf86config 설정화면

(3) GNOME (GNU Network Object Model Environment)

1 GNOME 소개

① GNOME 에 대해서

GNOME 은 1984년부터 GNU 프로젝트의 일환으로 사용자들이 쉽고 친숙한 데스크탑 환경을 제공하기 위해 시작되었습니다. GNOME은 완전한 데스크탑을 지향하고 있기 때문에 다양한 응용프로그램을 포함하고 있습니다. 대부분의 GNU 프로그램들처럼, GNOME 역시 모든 Unix-like 운영체제 사용할 수 있도록 설계되어 있습니다.

GNOME 의 주요 구성은 다음과 같습니다.

- GNOME 데스크탑 — 사용자에게 친숙한 윈도우 환경을 제공합니다.
- GNOME 개발 플랫폼 — 각종 개발 도구 및 라이브러리와 유닉스상에서의 개발 도구들을 지원합니다.
- GNOME 오피스 — 스프레드 시트, 워드 프로세서 등의 오피스 군을 제공합니다.

② 와우리눅스 GNOME 에 대해서

와우리눅스에서 제공하는 GNOME은 Ximian GNOME 1.4를 기반으로 사용자들에게 보다 더 친숙한 환경을 제공하고자 노력하였습니다. 각종 응용프로그램의 한글화를 비롯하여, 편리한 데스크탑 환경 설정 도구인 Doorman 을 포함하고 있으며 멋진 데스크탑을 위한 각종 테마와 바탕화면을 제공합니다.

와우리눅스 GNOME 1.4 다음과 같은 응용프로그램을 포함하고 있습니다.

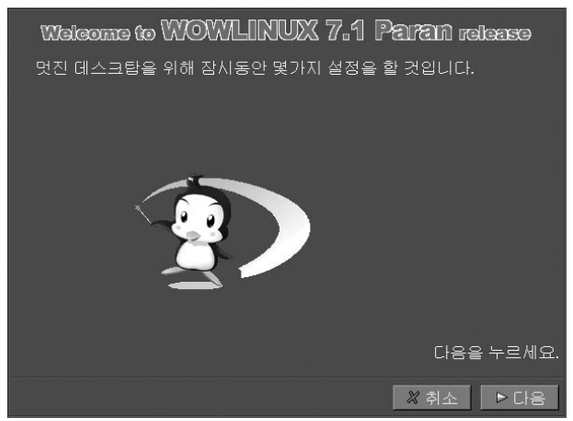
- Nautilus GNOME 1.4 에서 사용할 수 있는 파일 관리자로서 매우 유연합니다.
- Mozilla 인터넷 웹 브라우져의 최신 버전
- Galleon 모질라 엔진을 사용하는 가벼운 웹 브라우져
- GIMP 이미지 리터칭 프로그램으로, 포토샵이나 페인트샵과 유사합니다.
- Xchat IRC 클라이언트
- PAN 뉴스그룹 클라이언트

② GNOME 시작하기

GNOME 데스크탑 환경을 사용하기 위해서는 /etc/sysconfig/desktop 파일에서 DESKTOP="GNOME" 이라고 되어 있어야 합니다. 만약 그래픽 로그인을 사용하신다면 GNOME 세션을 선택하면 쉽게 GNOME 을 사용할 수 있습니다.

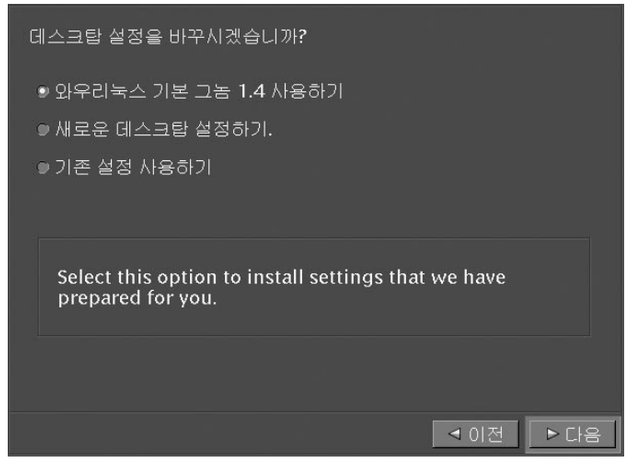
① 문지기(Doorman) 사용하기

처음 GNOME 을 시작하는 사용자는 다음과 같은 화면을 만날 것입니다. 이것은 GNOME 데스크탑 설정 도구로서 Ximian 에 포함된 Doorman 을 와우에서 수정한 와우 문지기(Doorman) 입니다.
설정을 위해서는 "다음"을 누르세요.



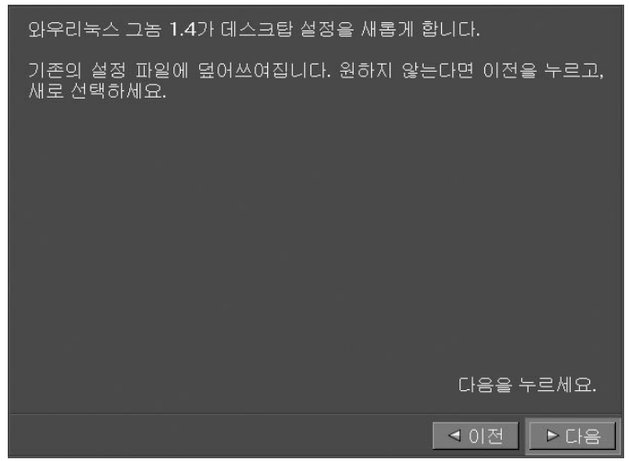
[그림 Ⅳ-11] 환영인사

와우 문지기(Doorman) 초기화면



[그림 Ⅳ-12] 데스크탑 설정 유형

데스크탑 설정 유형을 묻고 있습니다. 3가지 유형은 중 첫번째 "와우리눅스 기본 그놈 1.4 사용하기"는 와우에서 미리 설정해 놓은 것을 적용하게 됩니다. 조금 더 세밀한 설정을 원하시는 분은 두번째의 "새로운 데스크탑 설정하기." 메뉴를 이용하세요. 만약 어떠한 설정도 원하지 않으신다면 마지막의 "기존 설정 사용하기"를 선택하시면 됩니다.
"다음"을 누르면 설정을 시작합니다.



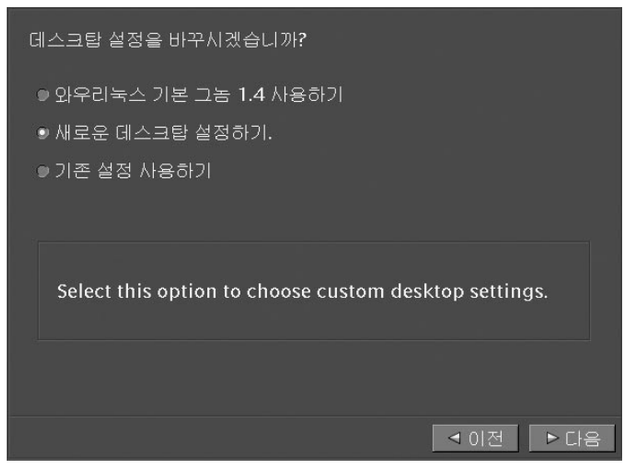
[그림Ⅳ-13] 데스크탑 설정

와우리눅스 기본 그놈 1.4로 데스크탑을 구성합니다.

[그림 Ⅳ-14] 설정완료

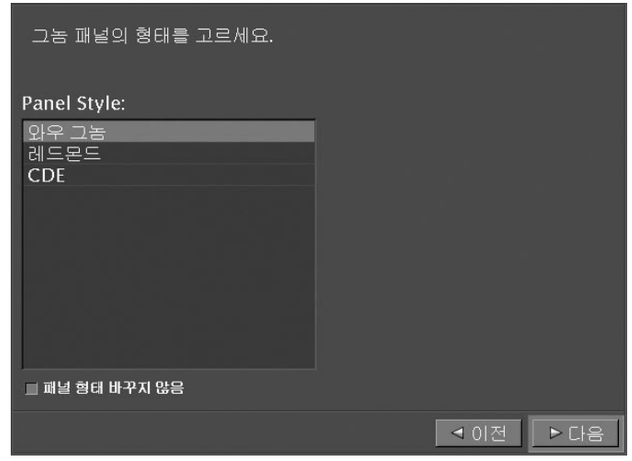
설정이 모두 끝났습니다. "확인"을 누르시고 새롭게 꾸며진 와우리눅스 그놈 1.4를 사용하세요.

사용자 설정



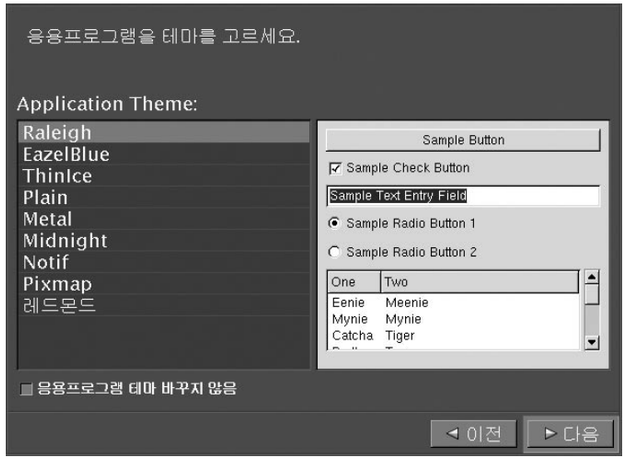
[그림 Ⅳ-15] 데스크탑 설정 유형

사용자 설정을 위해서는 데스크탑 유형에서 "새로운 데스크탑 설정하기"를 선택합니다.



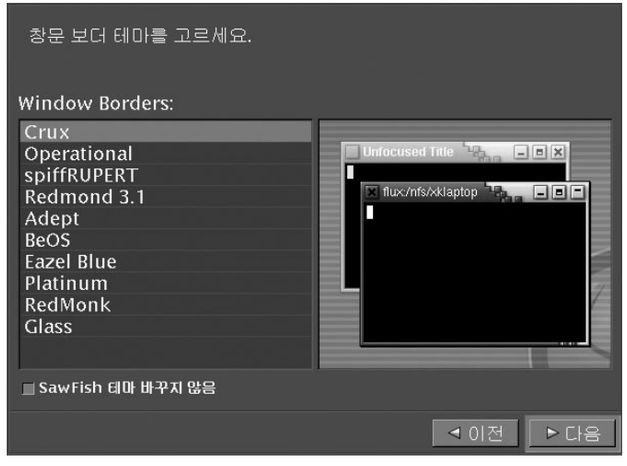
[그림 Ⅳ-16] 패널 형태 선택

패널의 형태를 고르세요. 선택하면 패널의 구성이 화면에 보여집니다. 원하는 것을 고르시고 "다음"을 누르세요



[그림Ⅳ-17] GTK 테마설정

GTK 테마 설정입니다. 원하시는 테마를 선택하시고 "다음"을 누르세요

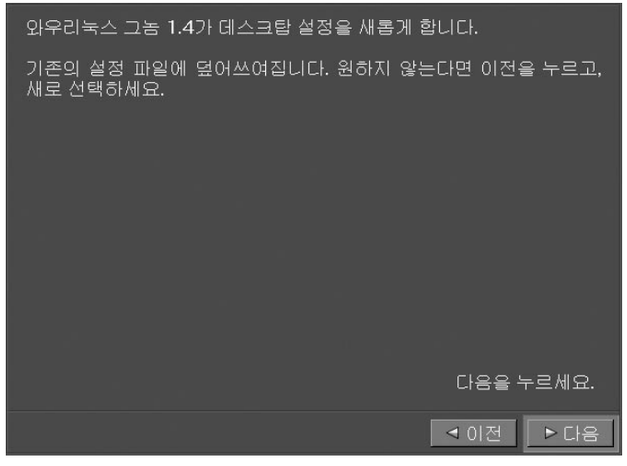


[그림 Ⅳ-18] 창문 보더 테마 설정

Sawfish 윈도우 매니저의 창문 테마 설정입니다. 원하시는 테마를 선택후 "다음"을 누르세요.

[그림 Ⅳ-19] 바탕화면 벽지 선택

와우리눅스 GNOME 1.4 에서 사용할 바탕화면을 고르세요. 그리고 "다음"을 누르세요



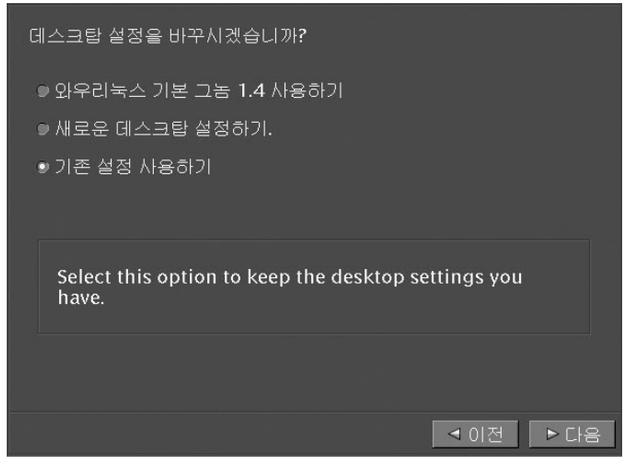
[그림 Ⅳ-20] 데스크탑 설정

사용자가 설정한 데스크탑으로 구성합니다.

[그림 Ⅳ-21] 설정완료

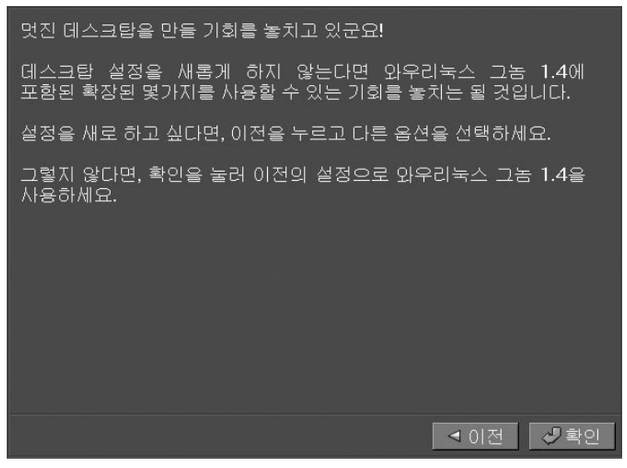
설정이 모두 끝났습니다. "확인"을 누르시고 새롭게 꾸며진 와우리눅스 그놈 1.4를 사용하세요.

기존 설정 사용하기



[그림 22] 데스크탑 설정 유형

데스크탑 설정 유형에서 "기존 설정 사용하기"를 선택하면 아무런 설정을 하지 않고 문지기(Doorman)을 종료합니다. 설정을 원치 않으신다면 선택 후 "다음"을 누르세요

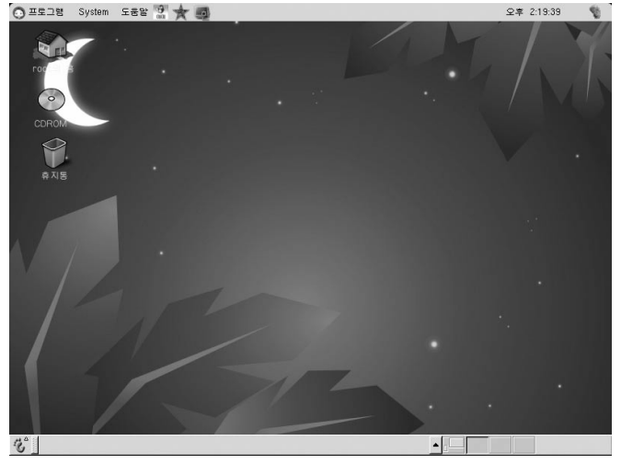


[그림Ⅳ-23] 기존설정 사용

아무런 설정없이 문지기(Doorman) 을 종료합니다. "확인"을 누르세요

② 와우리눅스 GNOME 1.4 실행

문지기(Doorman) 설정이 끝나면 와우리눅스 GNOME 1.4가 실행됩니다. GNOME 에 자세한 설정 및 사용법은 상단의 "도움말"을 선택 후 "User's Manual" 참조하시면 됩니다.



[그림Ⅳ-24] 와우 그놈 기본화면

문지기(Doorman) 설정 후 실행된 GNOME 화면

(4) KDE (K Desktop Environment)

① KDE란?

KDE는 X 윈도우 시스템의 또 다른 데스크탑 환경으로 GNOME과 같이 시스템을 쉽게 설정하고 사용할 수 있는 사용자 인터페이스를 유닉스 시스템에 도입하는 것을 목표로 1996년 Matthias Ettrich에 의해 시작된 프로젝트로 1998년에 1.0 버전이 나왔으며, 2001 현재는 2.1.1 안정버전이 사용되고 있습니다.

KDE는 노르웨이의 트롤테크라는 회사에서 개발된 크로스 플랫폼 라이브러리인 QT를 사용하여 개발되었는데, 이 QT의 초기 라이센스가 GNU의 GPL이 아닌 QPL (무료 소프트웨어를 작성할 때는 QT Free Edition 라이브러리를 사용하고, 유로 소프트웨어를 작성할 경우에는 QT Professional Edition 라이브러리를 사용해야 한다)이라는 독특한 방식이라 리눅스서들 사이에서 의견이 분분했지만, 지금은 GPL로 바뀌어 많은 개발자들이 QT를 사용하여 프로그램을 개발하고 있습니다.

② KDE의 구성

일반적인 KDE 데스크탑은 다음과 같이 세개의 영역으로 구성되어 있습니다.

■ 패널(panel) : 화면의 밑부분에 있는 것으로 데스크탑 사이를 전환하거나 어플리케이션 프로그램들을 시작하는데 사용됩니다. 아래에 있는 것들 중에서 "K"아이콘이 있는데 이것은 응용프로그램들의 메뉴를 보여주고 이것을 클릭하면 프로그램이 실행됩니다.

■ 태스크바(taskbar) : 화면의 왼쪽 상단에 위치해 있으며 현재 실행되고 있는 응용프로그램들을 보여주고 그것들 사이를 전환하는데 사용합니다. 원하는 프로그램을 활성화 하려면 태스크바(taskbar)상의 원하는 프로그램에 대고 클릭하면 됩니다.

■ 데스크탑(desktop) : 자주 사용되는 파일들이나 프로그램 폴더들이 위치하는 곳입니다.
KDE는 다중 데스크탑환경을 제공하며 각각의 데스크탑은 그 자신의 윈도우를 가집니다.
데스크탑 사이를 전환하려면 아래쪽에 위치한 패널(panel)상의 숫자버튼에 대고 클릭해보십시요.

③ 응용프로그램 실행

① Application Starter와 패널(Panel)사용

화면 하단에서 여러분은 데스크탑 패널(panel)을 찾을 수 있습니다. 그리고 응용프로그램을 실행시키기 위해 패널(panel)을 사용하게 되는데, 패널(panel)의 왼쪽부분에 대문자 K 버튼을 Application Starter 라고 부릅니다. 위쪽으로 작은 화살표시가 있고 여러분이 클릭을 하게 되면 화살표가 지시한 방향으로 팝업메뉴가 나타날 것입니다. 그럼 실행해 보도록 합니다. 이 팝업메뉴는 여러분이 여러분의 컴퓨터에 설치된 응용프로그램을 쉽게 사용할 수 있도록 도와 줍니다.

단축키: Alt-F1 을 누르면 Application Starter 메뉴가 나타납니다.

■ 패널(Panel) 조정하기

여러분이 만약 어떤 응용프로그램을 매우 자주 사용한다면, 물론 여러분은 더욱 빠른 방법으로 실행을 시키고 싶을 것입니다. 이러한 경우, 여러분은 단일 응용프로그램이나 아니면 Application Starter 메뉴상의 하위메뉴 전체를 패널(panel)상에 특별히 빠른 실행을 위한 버튼으로 추가 시킬 수가 있습니다. 만약 여러분이 버튼을 누르는 대신 "Find Files" 를 바로 실행시키고 싶다면 간단히 application starter 〉 Panel 〉 Add Application 〉 Find Files 순으로 선택하면 됩니다. (여러분은 Application Starter를 먼저 클릭해야 하고 그리고 난후 패널(Panel)을 선택하고 오른쪽 화살표가 지시하는 방향으로 생긴 또 다른 메뉴를 본 후 거기서 Add application을 선택하고 다음 하위메뉴의 "Find Files"를 선택하는 것을 의미합니다.) 여러분은 패널(panel)상의 모든 메뉴 항목들을 Context 메뉴(마우스 오른쪽 클릭을 하면 생기는 메뉴)상의 "move" 명령으로 옮길 수 있다는 사실을 기억하십시요. 단지 세번째 마우스 버튼을 클릭하십시요. (세번째 마우스 버튼이란 일반적으로 오른쪽 마우스 버튼을 말합니다. 만약 여러분이 왼손잡이 마우스설정으로 해놓았다면 왼쪽 마우스 버튼이 될 것입니다.) 그러면 어떤 메뉴가 뜰 것인데 거기서 "Move"를 선택하면 됩니다. 이제 마우스를 이동해 보십시요. 그리고 패널상에 머물러 있는 동안 어떻게 아이콘이 따라오는지를 지켜보십시요. 이제 되었다면, 첫번째 마우스버튼(기본적으로 왼쪽마우스버튼)을 클릭하십시요. 여러분이 눈치챘듯이, 여기 메뉴에는 여러분이 데스크탑 상의 어떤 실행버튼에 싫증이 났을 경우를 대비해서 "Remove" 명령이 있습니다.

■ Context 메뉴 사용하기

이것은 우리를 또 다른 재밌는 주제로 이끌 것입니다. 많은 곳에서 여러분은 (여러분이 클릭한 것에 적용되는 선택들을 가지고 있는) context menu를 나타나게 하기위해 오른쪽 마우스버튼을 클릭할 수 있습니다. 만약 여러분이 어떤 것으로 무엇을 할지 모른다면, 이때 어떤 것에 대고 세번째 마우스 버튼을 누르는 것은 항상 좋은 생각입니다. 심지어 데스크탑의 배경도 그러한 메뉴를 가지고 있습니다.

■ 다른 패널(Panel) 특징들

패널과 함께 또 다른 재미가 가능한 것들이 있습니다. 그 중 하나는 만약 여러분이 모니터 상에 저해상도를 쓰고 있다면 중요할지 모릅니다. 그것은 "hide-and-show" 기능이라 불리는데, 패널의 왼쪽 끝에 구성된 막대기 같은 버튼에 대고 클릭하면 이 기능을 볼 수 있습니다. 그러나 KDE상의 어떤 것인지 모르겠으면, 그것에 마우스포인터를 올려놓고 잠깐동안 기다려 보십시요. KDE는 작은 도움말 기능을 가지고 있는고, 이것을 "tooltips" 라고 부릅니다. 이것은 몇 개의 단어로 그것의 기능을 설명합니다.

② 커맨드라인을 통한 명령 실행

KDE는 여러분에게서 커맨드라인(때때로 매우 효율적입니다)을 앗아가길 원치 않습니다. 여러분은 데스크 탑에서 여러분의 파일을 옮길 수 있고, 여러분이 익숙한 유닉스 명령을 사용할 수도 있습니다. KDE는 konsole 이라 불리는 아주 세련된 커맨드라인 윈도우를 제공합니다. 시작하려면 application starter 〉 System 〉 Terminal 을 선택합니다. 이것은 여러분의 패널상에 있는 여러분이 원하는 어떤 것 일수 있습니다.

때때로 여러분은 단지 커맨드라인에 하나의 명령만을 입력하길 원할 수도 있습니다. 이러한 경우에, 여러분은 터미널 전체를 필요로 하지 않습니다. Alt-F2를 누르게 되면 여러분은 작은 커맨드라인을 보게 될 것이고 여기에 하나의 명령을 써 넣을 수 있습니다. 그 후에 커맨드라인 윈도우는 사라지지만, 그것은 여러분의 명령을 기억합니다. 여러분이 이 윈도우를 다시 띄우고(우리는 이것을 "minicli"라고 부릅니다.) Up-arrow 키를 눌렀을 때, 여러분이 이전에 입력했던 모든 명령들을 다시 뒤져볼 수(browse) 있습니다. 또한 여러분은 특정 URL로 konqueror을 열기위해 minicli에다가 URL을 입력할 수도 있습니다.

④ 윈도우에서의 작업

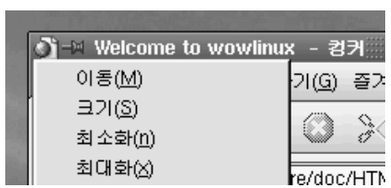
① 윈도우 제어하기

대개 사람들은 윈도우내(inside)에서 작업을 하는데, 때때로 여러분은 윈도우를 다루기를 원할 때도 있을 것 입니다. 여기에서 일반적인 윈도우 관련 기능에 대하여 몇가지 방법을 소개합니다.

- ■ 윈도우 옮기기 : 윈도우 타이틀바를 드래그 하거나, Alt키를 누르고 윈도우내의 아무 곳이나 드래그 합니다.
- ■ 윈도우 크기 조절하기 : 윈도우의 외곽선을 드래그 하거나, Alt 키를 누르고 윈도우내의 아무 곳에서나 마우스 오른쪽 버튼으로 드래그 합니다.
- ■ 윈도우 최대화 시키기 : 화면 가득히 윈도우를 채우기 위해서 또는 최대화되어진 윈도우를 원래 크기로 돌아가게 하기 위해 타이틀바에 있는 최대화버튼(maximize button - X 옆에 있는 사각형)을 클릭합니다. 가운데 마우스 버튼(2버튼일경우 오른쪽 왼쪽 동시 클릭)을 클릭하는 것은 윈도우를 수직으로 최대화 시키고, 오른쪽 버튼을 누르는 것은 수평으로 최대화 시킵니다.
- ■ 윈도우 아이콘화하기 : 윈도우를 숨기고 싶으면 타이틀바에서 아이콘화 버튼(최대화버튼 옆)을 클릭합니다. 태스크바(taskbar)에서 윈도우아이콘을 클릭함으로써 다시 윈도우를 생성시킬 수 있습니다.
- ■ 윈도우간 전환하기 : 다른 윈도우로 전환하기 위해 일반적인 마우스 클릭말고도, 여러분은 윈도우간 전환을 위해 Alt-Tab을 사용할 수도 있습니다

타이틀바 버튼(Titlebar buttons)

KDE 윈도우 창들은 타이틀바 위에 표준의 버튼들을 가지고 있고, 이 버튼들은 여러분에게 통상의 작업들에 대해서 빠른 액세스를 할 수 있도록 도와줍니다. 기본 버튼의 레이아웃은 다음과 같습니다.



[그림Ⅳ-25] 윈도우 메뉴

- 메뉴버튼 : 이것은 대개 응용프로그램을 위한 미니 아이콘을 보여줍니다. 윈도우의 기능메뉴를 얻으려면 그것을 클릭해 보십시오.
- 단축키 : Alt-F3로 윈도우메뉴를 열 수 있습니다.
- 스티키버튼(sticky button), 압정그림이 있는 것 : 가상 데스크탑 사용 섹션에서 살펴보도록 하겠습니다.


[그림 Ⅳ-26] 창 조절 버튼

아이콘화시키는 버튼.
활성창을 최대로 만드는 버튼.
닫기 버튼. (단축키 : Alt-F4)

윈도우간 전환

KDE는 윈도우간 전환을 하는데 있어서 여러 가지 방법을 제공합니다. 여러분은 여러분이 가장 선호하는 방식을 사용하시면 됩니다.
많은 윈도우 시스템에서는 다른 윈도우를 사용하기 위해서 그 윈도우를 마우스로 클릭하는 것을 필요로 합니다. 이것은 KDE의 기본 행동으로 "ClickToFocus" focus정책이라고 부릅니다. 그리고 여러분은 여러분의 데스크 탑을 마우스포인터를 윈도우에 위치시키면 활성화되도록 설정 할 수도 있습니다. 이 것을 FocusFollowMouse 라 부릅니다. 만약 여러분이 KDE Control Panel을 사용하여 이 정책을 선택하게 되면, 마우스가 위치한 윈도우는 항상 활성화됩니다. 이것은 자동으로 맨위로 오지는 않지만 여러분은 여전히 타이틀바나 윈도우의 외곽선을 클릭할 수 있고, 또는 KDE의 특별한 Alt키와 마우스의 가운데 버튼을 윈도우상의 어느 지점에서나 클릭할 수 있습니다.
또한 윈도우간 전환을 위한 몇가지 다른 방법도 있습니다. Window list 메뉴로부터 어떤 윈도우를 클릭하고, 메뉴를 열기 위해 데스크탑의 빈 영역에 마우스를 대고 가운데 버튼을 클릭한 후, 패널상의 여러 가지 윈도우들이 있는 아이콘을 클릭합니다. 또는 Ctrl-Esc 를 누릅니다. 또 Alt 키를 누르고 Tab 키를 눌러 윈도우간 전환을 할 수도 있습니다. 마지막으로 taskbar를 사용하는 간단한 방법도 있습니다.

② 태스크바(Taskbar) 사용

태스크바(taskbar)는 작은 아이콘들의 리스트를 나타내고, 아이콘 하나는 데스크탑상의 각각의 윈도우를 나타냅니다. 기본 KDE설정에서 태스크바(taskbar)는 패널의 가운데 위치합니다. 태스크바(taskbar)는 매우 강력하므로 적절한 태스크바의 버튼에 대고 왼쪽 마우스 버튼 클릭을 한번 하는 것으로 여러분을 즉시 선택된 응용프로그램으로 데려다 줍니다. 리스트중 하나를 가운데 마우스버튼으로 클릭하면 그 윈도우는 아이콘화 하거나 다시 나타나고, 오른쪽 마우스 버튼을 누르는 것은 선택된 윈도우의 윈도우 기능과 함께 context메뉴를 나타나게 할 것입니다.

③ 가상 데스크탑 사용

KDE는 몇 가지 다른 데스크탑(각각이 자신의 윈도우를 가진 데스크탑)을 다룰 수 있습니다. 기본 설정은 여러분에게 네 개의 데스크탑을 제공합니다. 여러분은 패널상의 데스크탑 버튼들 중의 하나에 마우스를 대고 클릭하므로써 쉽게 가상 데스크탑 사이를 전환할 수 있습니다. 또한 Ctrl-F1...F8키로 즉시 일치하는 데스크탑으로 옮길 수도 있고, Ctrl-Tab으로 데스크탑 사이를 전환할 수도 있습니다.

만약에 여러분이 마우스를 화면의 구석쪽으로 이동시켜서 거기서 잠깐 있으면, KDE는 여러분이 지시한 방향으로 데스크탑을 전환 할 것입니다.
때때로 여러분은 윈도우가 모든 데스크탑에서 나타나기를 원할 때가 있을 것 입니다. 예를 들어 자그마한 채팅윈도우 라든지 또는 알람 시계라든지 또는 무엇이든 그러고 싶을 때가 있을 것입니다. 이러한 경우 여러분은 위에서 언급한 "스티키(sticky)버튼"을 사용할 수 있습니다. 이것은 모든 가상 데스크탑에서 보여지게 하기 위해 배경에다가 윈도우를 핀을 꽂는 것입니다.
스티키(sticky)버튼은 또한 윈도우를 하나의 가상 데스크탑으로 부터 다른 데스크탑으로 옮기기 위해서 사용될 수도 있습니다. 윈도우의 스티키 핀버튼을 누르고, 다른 데스크탑으로 전환한 후, 그 핀 버튼을 다시 눌러서 자유롭게 합니다. 그러면 여러분이 태스크바(taskbar)에 있는 윈도우 엔트리(entry)의 context 메뉴(메뉴항목중 "Onto current desktop")를 사용하거나 윈도우 기능 메뉴에서 "Sent to" 옵션을 사용하는 것과 같은 효과를 얻을 수 있습니다.

⑤ 파일 관리

그래픽요소의 데스탑에 대한 가장 일반적인 표현은 여러분의 하드디스크상의 디렉토리들을 폴더를 사용해 나타내는 것입니다. 폴더(Folder)들은 파일들과 다른 폴더를 포함하며, KDE응용프로그램인 konqueror는 여러분이 여러분의 파일들을 관리하는데 도움을 주기위해 이 방법을 사용합니다.

① konqueror 사용하기

application starter 〉 Home Directory 를 실행했을 때 여러분의 파일을 보여주는 윈도우가 konqueror 윈도우입니다. 윈도우의 툴바(toolbar) 아래에 폴더의 패스이름(pathname) 이 표시됩니다. 어떤 파일이나 폴더를 열고 싶다면, 단순히 그것을 왼쪽 마우스 버튼으로 한번 클릭하면 되고, 두개의 점으로(..)이루어진 폴더를 클릭하게 되면 여러분을 한단계 상위 디렉토리 레벨로 옮겨지게 됩니다. 또한 여러분은 보다 직접적인 이동을 위한 폴더 분류 단계를 보기위해서, 메뉴로 부터 Window 〉 Show Sidebar를 선택할 수 있습니다. 또는 특정 디렉토리를 빠르게 찾거나 디렉토리 이름을 완성(completion)하기 위해서 탭(tab)키를 사용하면서 편집을 할 수도 있습니다.

파일을 열기

KDE는 많은 종류의 파일들을 보거나 편집하기 위해서 사용되는 일련의 응용프로그램들을 가지고 있습니다. 그래서 이미지나 문서를 포함을 하고 있는 파일을 클릭하면 konqueror 는 그 파일을 보여주기 위해서 적절한 응용프로그램을 시동할 것입니다. 만약에 여러분이 클릭한 파일에 대해서 무엇을 실행시켜야 할지 모르면 konqueror 는 여러분에게 실행시킬 응용프로그램의 이름을 물어볼 것입니다.

> 파일과 응용프로그램들을 관계짓기위해 MIME types이 사용됩니다.

드래그 앤 드럽 (Dragging and Dropping Icons)

어떤 파일을 이동시키거나 복사하기 위해서, 단순히 그 아이콘을 데스크탑으로 혹은 다른 konqueror 윈도우나 폴더로 아이콘으로 드래그한 후 버튼을 놓게 되면, konqueror는 복사를 할 것인지 이동을 시킬 것인지 아니면 그 파일의 링크를 생성시킬 것인지를 선택하도록 하는 메뉴를 표시합니다. 주의할 점은, 만약에 여러분이 링크를 생성시키게 되면, KDE는 유닉스 "심볼릭 링크(symbolic link)"를 생성하게 되므로 여러분이 원본 파일을 지우거나 옮기게 되면, 그 링크는 깨질 것입니다.
많은 KDE응용 프로그램들이 드래그 앤 드럽 기능을 지원합니다. 어떤 응용프로그램이 그 파일을 실행시키게 하기 위해서, 실행중인 응용프로그램의 윈도우에 아이콘을 끌어다 놓을 수도 있고, 또는 실행하지 않고 있는 응용프로그램의 아이콘 위에 파일아이콘을 끌어다 놓을 수도 있습니다.

파일 속성 설정

파일의 이름이나 권한(퍼미션)같은 속성을 바꾸고 싶으면, 마우스를 아이콘에 대고 오른쪽 버튼을 클릭한 다음 메뉴에서 속성(properties)를 선택하시면 됩니다.

② 압축파일과 네트워크상에서 하는 작업

최근까지 여러분은 인터넷상의 파일들을 액세스 하는 데에 특별한 소프트웨어가 필요했었습니다. 하지만 KDE는 "Network Transparent Access" (NTA) 라고 하는 기술을 지원하고, 이것은 여러분으로 하여금 지구 반대편에 있는 파일을 여러분의 로컬하드 디스크에서 작업하는 것 만큼 쉽게 할 수 있도록 합니다. 예를 들어 FTP서버상의 파일을 액세스 하려면, 단지 konqueror메뉴상의 Location 〉 Open Location 을 선택하고 난 후, FTP서버의 URL을 입력하면, 여러분은 마치 로컬하드 디스크에 있는 것처럼 서버에 있는 폴더로부터 파일들을 드래그 앤 드럽 할 수가 있습니다. 심지어 FTP서버상에서 그 파일들을 여러분의 로컬 하드디스크로 복사할 것 없이, 필요할 때 바로 파일들을 열어 볼 수도 있습니다.
konqueror는 anonymous FTP 액세스를 사용한다는 것을 주의하십시요. 이것은 FTP상의 파일을 액세스 하려는 당신의 행동을 제한할지도 모릅니다. 만약에 여러분이 서버에 계정을 가지고 있다면, 여러분은 URL의 부분과 여러분의 유저 ID를 다음과 같이 덧붙일 수 있습니다.

```
ftp://userid@server/directory
```

Konqueror는 여러분에게 패스워드를 물어볼 것이고, 그런 후에 로그인이 성공하게 되면, 여러분은 서버상의 여러분의 파일에 대해 완전한 액세스 권한을 가지게 될 것입니다.
여러분이 MS-윈도우에 있는 WinZip(tm) 유틸리티를 사용해 본적이 있다면, KDE도 똑같이 tar 파일내부를 들여다 볼 수 있다는 말에 기뻐할 것입니다. KDE는 그러한 압축파일을 일반 폴더와 같이 취급하므로 여러분은 압축파일 내부를 들여다 볼 수 있습니다.
일반적으로, 인터넷상의 파일과 압축파일내의 파일들을 액세스하는 것은 ,네트워크에 의해 지연되는 시간을 제외하고는, 여러분의 로컬 하드디스크에서 파일을 액세스하는 것과 보고 느끼기에 똑 같습니다.

③ 응용프로그램과 디바이스를 액세스하기위해 템플리트(Templates) 사용하기

처음 로그인 했을 때 여러분의 데스크탑에 KDE가 위치시키는 폴더들 중의 하나를 템플리트(Templates)라고 부르고, 그 것들은 .desktop 혹은 .kdelnk 확장자를 가진(보이지 않는) 파일을 가지고 있습니다. 이러한 kdelnk 파일들은 KDE에서 다음을 보이기 위해 사용됩니다.

- 응용프로그램
- 프린터
- (플로피 디스크같이) 마운트 가능한 디바이스
- (WWW문서들이나 FTP디렉토리와 같은)인터넷 자원

Application Starter 상에 있는 그리고 패널상에 있는 거의 모든 항목들은 그것이 디스크상에 있는 kdelnk 파일임을 말합니다. kdelnk파일은 나타낼 아이콘이 무엇인지를 정의할 뿐만 아니라 아이콘이 무엇(응용프로그램,디바이스,또는 URL)을 상징하는 것인지에 대한 정의된 정보를 가집니다. 여러분은 빠른 실행버튼(quick-launch button)을 만들기 위해서 패널에 어떠한 kdelnk파일이라도 끌어다 놓을 수 있습니다.

프린터 설정

여러분은 프린터를 위한 아이콘을 생성할 수 있습니다. 그래서 프린터 아이콘에 파일을 끌어다 놓음으로써 프린트를 할 수 있습니다. 마우스의 오른쪽 버튼을 눌러 Creat New 〉 Link To Application 을 선택한 후, General탭에서, name부분을 Printer.kdelnk로 변경하고, Execute탭에서 첫번째 Execute부분에 "lpr %u" 와 같이 입력합니다.

※주의 : 이 예는 여러분이 lpr명령으로 프린트를 한다고 가정한다. 만약에 여러분이 다른 명령을 쓴다면, 여러분이 사용하는 명령을 입력합니다.

그리고 Execute탭에서, 톱니바퀴 같이 생긴 아이콘을 클릭하고, 나타난 아이콘 가운데 프린터 아이콘을 선택합니다. 이제 여러분은 프린터 아이콘에 파일을 드래그 함으로써 기본 프린터로 부터 출력물을 받을 수 있습니다.

디바이스 마운트하기

유닉스는 주 하드디스크가 아닌 다른 저장 매체에 대해 마운팅(mounting) 이라 불리는 프로세스를 통해야 액세스를 할 수 있습니다. KDE는 CD-ROM드라이브나 플로피 드라이브 같은 제2의 저장매체에 있는 파일을 액세스, 마운트, 언마운트하기 쉽도록, kdelnk를 사용합니다.

마우스의 오른쪽 버튼을 눌러 Create New 〉 Floppy Device 를 선택하고, General 탭에서, name부분을 Floppy.kdelnk로 변경한 후, Device 탭에서, Device부분에 /dev/fd0를 입력합니다. 그리고 Unmounted Icon 을 클릭하고, 녹색점이 없는 플로피 디스크 아이콘을 선택합니다.

이제, 적절히 포맷이 된 플로피 디스켓을 드라이브에 넣고 KDE가 플로피 드라이브를 마운트하고 디스크내의 파일을 보여줄 수 있도록 플로피아이콘을 클릭합니다. 드라이브에서 디스크를 빼기 전에는 플로피 아이콘에 대고 오른쪽 버튼 클릭을 한 후, 나타나는 메뉴에서 Unmount를 선택해야 합니다.

⑥ 데스크탑 설정하기

여러분은 데스크탑이 작업을 수행하는 방식이나 모습이 마음에 들지 않으면, 그것을 바꿀 수 있습니다. KDE는 많은 설정이 가능하고 여러분은 데스크탑의 행동 또는 모습에 대한 거의 모든 면을 바꿀 수가 있습니다. 또한 많은 다른 유닉스 데스크탑 환경과는 달리, 여러분은 암호 같은 설정파일을 편집할 필요도 없습니다. 여러분은 여러분의 데스크탑을 설정하기 위해 특별한 프로그램인 KDE Control Center를 사용합니다.

① KDE Control Center사용하기

[그림 Ⅳ-27] KDE 제어판

Application Starter로 부터 Control Center를 실행시키면, 두개의 창으로 나뉘어진 하나의 윈도우가 나타나고, 왼쪽창에는 항목(module)의 리스트가 나타납니다.

설정을 변경하는 것은 아주 직관적이며, 헬프 버튼은 잘 모르는 설정부분을 설명 하기위해 각 설정패널에서 사용 가능합니다. 각각의 패널은 다음과 같은 이름의 버튼을 가지고 있다. Apply, Reset 그리고 Use Default 이며, 다음과 같은 일을 수행 합니다.

- Apply : 설정변경 된 것을 적용합니다.
- Reset : 변경사항 적용 없이 원 상태로 되돌립니다.
- Use Default : 초기값으로 되돌립니다.

주의 : 만약 여러분의 하나의 설정패널에서 설정을 변경하고 Apply버튼을 누르지 않은 채 다른 항목으로 이동하게 되면, 여러분의 변경된 설정을 모두 잃을 것입니다.

⑦ 로그 아웃(Logging out)

Application Starter 〉 Logout 버튼을 누르면 간단히 로그아웃 할 수 있습니다.

① 세션 관리(Session Management)

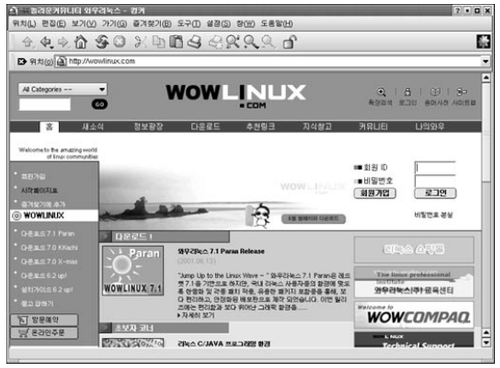
여러분이 로그아웃을 할 때, KDE는 어떤 응용프로그램이 열려있었는지를 기억하고 그 윈도우가 어디에 있었는지를 기억합니다. 그래서 KDE는 다음 번에 여러분이 로그인 을 할 때 기억해 놓은 것들을 다시 열수 있게 됩니다. 이 특징을 Session Management라고 부르고, KDE관련 응용프로그램들은 여러분이 로그아웃을 할 때의 상태를 응용프로그램 자신 스스로 기억을 합니다. 예를 들어, kedit는 어떤 파일을 여러분이 편집하고 있었는지를 기억합니다. Non-KDE(KDE관련 응용프로그램이 아닌) 응용프로그램들은 로그아웃시의 자신의 상태를 기억하지 못합니다. 그래서 KDE는 여러분이 로그아웃 할 때 그 응용 프로그램의 중요한 데이타를 저장을 했는지를 여러분이 확신하는가 경고를 할 것입니다.

세션관리를 경험해 보고 싶으면, application starter 〉 Applications 〉 Editor 순으로 선택을 하여 KEdit 를 실행시킨 후, 편집할 문서를 열고 로그아웃을 하고 다시 들어옵니디다. 여러분은 KEdit가 화면상의 정확히 똑같은 지점에 나타나는 것을 관찰할 것이고, 정확한 가상 데스크탑에 있고, 우리가 로그아웃하기 전에 KEdit에 열었던 문서가 자동으로 다시 열려있음을 알 수 있습니다. KEdit는 심지어 여러분이 로그아웃하기 전에 문서에 편집했던 내용을 저장하였는지 하지 않았는지도 기억합니다. 그래서 KEdit는 만약에 여러분이 'File menu' 로 부터 'Save' 를 선택하면, 여러분이 작업하고 있던 그 파일에 저장을 할 것입니다.

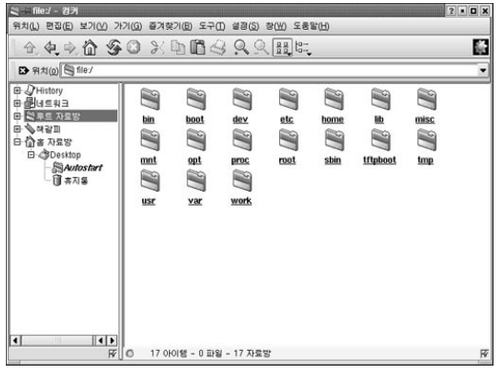
⑧ 유용한 KDE용 프로그램들

① 컹커 (konqueror) : 파일관리자 / 웹브라우져

윈도우의 탐색기를 연상케 하는 컹커는 파일 관리자겸 웹브라우져로서 다양한 기능을 가지고 있습니다. 대부분의 네트웍 프로트콜을 지원하고 있으며, 자바 스크립트 또한 완벽하게 지원하여 웹브라우져로서 전혀 손색이 없습니다. 더불어 파일 관리자로서도 그 역할을 충실히 실행하고 있습니다.



[그림 Ⅳ-28] 웹브라우져



[그림Ⅳ-29] 파일관리자

② K 메일

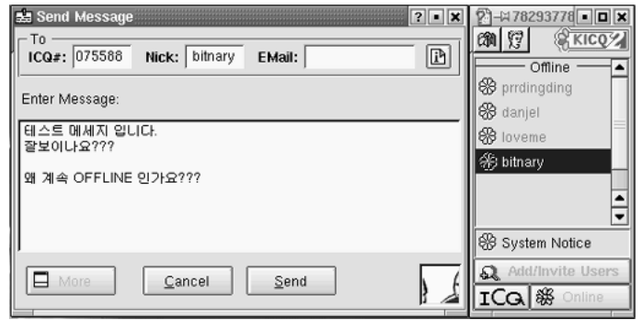
K 메일은 윈도우의 Outlook Express와 같이 pop3 혹은 imap 을 지원하는 메일 클라이언트로 다중 사용자 지원 및 메시지 규칙을 포함하는 매우 유용한 프로그램입니다.

[그림 Ⅳ-30] 편지읽기

[그림 Ⅳ-31] 편지쓰기

③ Kicq

가장 널리 사용되는 인터넷 메시지 프로그램으로 윈도우 icq 인터페이스를 그대로 사용하고 있습니다. 윈도우 사용자와 자유롭게 의사 소통을 할 수 있는 또 다른 수단이 되어 많은 리눅스 유저들이 사용하고 있습니다.



[그림 Ⅳ-32] Kicq

④ Ksnapshot

리눅스의 멋진 X 윈도우를 캡쳐할 때 사용됩니다. 간단한 기능을 가지고 있지만, 여러모로 유용하게 사용되는 유틸리티입니다. 멋진 스크린샷을 캡쳐하여 당당히 리눅스 유저임을 뽑내 보십시오!

[그림 Ⅳ-33] Ksnapshot

2. RPM 활용하기

(1) RPM이란?

RPM은 레드햇 패키지 관리자를 말합니다. 여기에 레드햇 이라는 이름이 들어 있지만, 누구나 사용할 수 있는 개방된 패키징 시스템으로 만들어졌습니다. RPM은 사용자가 새로운 프로그램의 소스 코드를 소스와 바이너리로 패키징이 가능하도록 합니다. 이렇게 되면 바이너리를 쉽게 설치하고 찾아내고 소스를 쉽게 다시 빌드할 수 있게 됩니다. 이것은 모든 패키지와 파일의 데이터베이스를 관리하는데, 이는 패키지를 검증하고 파일과 패키지의 질의에 쓰여집니다.
레드햇 소프트웨어는 다른 배포본 제작자들이 RPM을 살펴보고 배포본에 채용할 것을 권장합니다. 이렇게 많은 부문에서 확장 가능한 기반을 제공함에도 불구하고도, RPM은 매우 유연하고 사용하기 쉽습니다. RPM은 전부 개방되었고 사용가능해서 우리는 버그리포트와 수정을 보내 주는 것에 감사하는 바입니다. RPM은 GPL을 따라 로얄티 없이 배포됩니다.

(2) RPM 사용하기

① 요구사항

RPM을 사용하기 위해 주된 요구 사항은 cpio 버전 2.4.2 이상이 필요합니다. RPM은 물론 리눅스에서 사용하고자 만들어졌지만, 다른 유닉스 시스템에도 포팅 할 수 있을 것입니다. 사실은 SunOS, Solaris, AIX, Irix, AmigaOS, 그 외 여러 유닉스 에서 모두 컴파일됩니다. 다만 주의할 것이 있다면, 서로 다른 유닉스 시스템에서 만들어진 바이너리 패키지는 호환되지 않습니다.
RPM을 설치하기 위한 최소 요구 사항은 다음과 같다. RPM을 소스에서 빌드하기 위해서, 여러분은 패키지를 컴파일하는데 필요한 것들, 즉, gcc, make 등이 필요합니다.

※ 레드헷 기반의 와우리눅스는 패키지 관리를 위해 RPM을 사용하고 있습니다.

② 패키지 설치 / 제거

가장 간단한 형태로, RPM 은 패키지를 설치할 때 다음과 같이 사용할 수 있습니다.

```
# rpm -i foobar-1.0-1.i386.rpm
```

다음의 간단한 명령은 패키지를 제거할 때 쓰는 것입니다.

```
# rpm -e foobar
```

매우 쓸모 있지만 더욱 복잡한 명령중 하나는 여러분이 FTP를 통하여 설치하는 것입니다. 여러분이 네트웍에 연결되어 있고 새로운 패키지를 설치하기를 원한다면, 여러분에게 필요한 것은 파일의 정확한 URL과 함께 파일의 위치를 정하는 것인데, 다음과 같습니다


```
# rpm -i ftp://ftp.redhat.com/------/foobar-1.0-1.i386.rpm
```

이제는 FTP 를 통한 설치와 질의 기능을 사용할 수 있습니다. (ftp/bin 디렉토리에 rpm 바이너리를 가져다 놓는 다면 여러분의 FTP 서버는 rpm 질의를 받을 수 있습니다.)

③ 패키지 관리

RPM은 매우 유용한 도구이고, 다양한 옵션을 갖추고 있습니다. RPM을 이해하는 가장 좋은 방법은 몇 가지 예제를 살펴보는 것입니다.
실수로 몇 가지 파일들을 지웠는데, 어느 것을 지웠는지 확신할 수 없면, 여러분은 전체 시스템을 점검해 보고 어떠한 파일이 빠져 있는지 보기 위해 다음과 같이 명령을 내립니다.

```
# rpm -Va
```

알 수 없는 파일을 보게 되고, 어떠한 패키지 안에 속해 있는지 보려면, 다음과 같이 명령을 내립니다.

```
# rpm -qf /usr/X11R6/bin/xjewel
```

위의 질의 결과는 다음과 같습니다.

```
# xjewel-1.6-1
```

여러분은 kouls RPM을 발견하였지만, 이것이 무엇인지 알 수 없다면, 이 파일에 대한 정보를 알기 위해 다음과 같이 질의를 할 수 있습니다.

```
# rpm -qpi koules-1.2-2.i386.rpm
```

질의 결과는 다음과 같습니다.

```
Name    : koules        Distribution: Red Hat Linux Colgate
Version : 1.2            Vendor: Red Hat Software
Release : 2          Build Date: Mon Sep 02 11:59:12 1996
Install date: (none)        Build Host: porky.redhat.com
Group   : Games          Source RPM: koules-1.2-2.src.rpm
Size    : 614939
Summary : SVGAlib action game with multiplayer, network, and sound support
Description :
This arcade-style game is novel in conception and excellent in execution.
No shooting, no blood, no guts, no gore. The play is simple, but you
still must develop skill to play. This version uses SVGAlib to
run on a graphics console.
```

또한 여러분은 kouls RPM을 설치할 때 어떠한 파일이 설치되는지 알고자 한다면, 다음과 같이 질의를 할 수도 있습니다.

```
# rpm -qpl koules-1.2-2.i386.rpm
```

질의의 결과는 같습니다.

```
/usr/doc/koules
/usr/doc/koules/ANNOUNCE
/usr/doc/koules/BUGS
/usr/doc/koules/COMPILE.OS2
/usr/doc/koules/COPYING
/usr/doc/koules/Card
/usr/doc/koules/ChangeLog
/usr/doc/koules/INSTALLATION
/usr/doc/koules/Icon.xpm
/usr/doc/koules/Icon2.xpm
/usr/doc/koules/Koules.FAQ
/usr/doc/koules/Koules.xpm
/usr/doc/koules/README
/usr/doc/koules/TODO
/usr/games/koules
/usr/games/koules.svga
/usr/games/koules.tcl
/usr/man/man6/koules.svga.6
```

(3) RPM 패키지 만들기

RPM을 만드는 것은 무척 쉽습니다. 특히 여러분이 만들고자 하는 패키지를 얻었을 때는 더욱 그렇습니다. RPM을 만드는 기본적인 절차는 다음과 같습니다.

- /etc/rpmrc가 있는지 확인합니다.
- RPM을 만들고자 하는 소스 코드를 구합니다.
- 정확하게 빌드하기 위해서 소스에 필요한 패치를 가합니다.
- 패키지에 대한 명세 파일을 만듭니다.
- 모든 것이 정확한 위치에 있는지 확인합니다.
- RPM을 사용하여 패키지를 만듭니다.

보통, RPM은 바이너리와 소스 모두 만듭니다.

① rpmrc 파일 (The rpmrc File)

RPM 설정 파일은 /etc/rpmrc 파일에서만 이루어집니다. 예를 들면 다음과 같습니다.

```
require_vendor: 1
distribution: I roll my own!
require_distribution: 1
topdir: /usr/src/me
vendor: Mickiesoft
packager: Mickeysoft Packaging Account <packages@mickiesoft.com>

optflags: i386 -O2 -m486 -fno-strength-reduce
optflags: alpha -O2
optflags: sparc -O2

signature: pgp
pgp_name: Mickeysoft Packaging Account
pgp_path: /home/packages/.pgp

tmppath: /usr/tmp
```

require_vendor에는 RPM이 vender 줄을 찾을 때 필요합니다. 이 줄은 /etc/rpmrc 명세 파일의 헤더에서 나옵니다. 이 기능을 끄려면, 숫자를 0으로 바꿉니다. 같은 방법은 require_distribution과 require_group에서도 적용이 가능합니다.
다음 줄은 distribution 줄입니다. 여러분은 여기 또는 명세 파일 헤더의 뒷부분에 정의할 수 있습니다. 특정한 배포본에서 만들 때, 이 줄이 맞는지 확인하는 것은 매우 좋은 생각입니다. 이것이 필요한 것은 아니지만, vender 줄도 같은 방법으로 이루어집니다.
RPM은 역시 다양한 아키텍처에서 패키지를 만드는 기능을 지원하고 있습니다. rpmrc 파일에는 아키텍처에 종속적인 플래그가 필요한 것을 빌드할 때 'optflags' 변수를 설정할 수 있습니다. 다음 단락에서 이러한 변수를 어떻게 사용하는지 볼 수 있습니다.
위에 있는 매크로에 더해서, 여기에는 몇 가지 더 있습니다. 여러분은 이렇게 사용할 수 있습니다.

```
# rpm —showrc
```

태그가 어떻게 세팅되는지, 사용 가능한 플래그가 어떤 것이 있는지 알기 위해 서는 위와 같은 명령을 내립니다.

② 명세 파일 (The Spec File)

우리는 명세 파일에 대해 논의할 것입니다. 명세 파일은 패키지를 만드는데 필요하며, 명세 파일에는 소프트웨어와 설치할 모든 바이너리와 그 파일 리스트를 어떻게 만드는지에 대하여 지시한 설명이데, 이는 소프트웨어에 따라오는 것입니다.
여러분은 명세 파일을 표준 관례에 따라 이름 짓기를 원할 것입니다. 명세 파일 이름은 이름-버전번호-발표 번호.spec 이 됩니다.
여기에 간단한 명세 파일이 있습니다. (vim-3.0-1.spec)

```
Summary: ejects ejectable media and controls auto ejection
Name: eject
Version: 1.4
Release: 3
Copyright: GPL
Group: Utilities/System
Source: sunsite.unc.edu:/pub/Linux/utils/disk-management/eject-1.4.tar.gz
Patch: eject-1.4-make.patch
Patch1: eject-1.4-jaz.patch
%description
This program allows the user to eject media that is autoejecting like
CD-ROMs, Jaz and Zip drives, and floppy drives on SPARC machines.
%prep
%setup
%patch -p1
%patch1 -p1
%build
make RPM_OPT_FLAGS="$RPM_OPT_FLAGS"
%install
install -s -m 755 -o 0 -g 0 eject /usr/bin/eject
install -m 644 -o 0 -g 0 eject.1 /usr/man/man1
%files
%doc README COPYING ChangeLog
/usr/bin/eject
/usr/man/man1/eject.1
```


③ 헤더 (The Header)

헤더에는 여러분이 채워넣을 필요가 있는 표준적인 필드를 담고 있습니다. 여기에는 몇 가지 주의할 것이 있습니다. 필드에는 반드시 다음과 같이 채워야 합니다.

- Summary : 패키지에 대한 간단한 설명을 한 줄로 씁니다.
- Name : 여러분이 사용하고자 하는 rpm 파일 이름이 와야 합니다.
- Version : 여러분이 사용할 파일 이름의 버전 번호가 와야 합니다.
- Release : 여기에는 같은 버전의 패키지 번호가 옵니다. (예를 들어 우리가 패 키지를 만들었는데 잘못된 것을 알고 다시 만들었을 때 다음 패키지의 release 번호는 2가 됩니다.)
- Icon : 여기에는 다른 (레드햇의 glint:;와 같은) 시각적인 설치 도구 에서 사용할 아이콘 파일의 이름이 옵니다. 아이콘은 반드시 gif 포맷이어야 하고 SOURCES 디렉토리에 위치하여야 합니다.
- Source : 이 줄에서는 원래 소스 파일의 위치를 가리킵니다. 이것은 여러분이 소스 파일을 다시 얻거나 새로운 버전을 체크하는데 쓰입니다.

※주의: 이 줄에서 파일 이름은 여러분의 시스템에 있는 파일 이름과 일치해야 합니다.
(예를 들어, 소스 파일을 다운로드 받아서 이름을 바꾸지 말아야 합니다.)

여러분은 하나 이상의 소스 파일을 다음과 같은 라인을 사용하여 지정할 수 있습니다.

Source0 : blah-0.tar.gz
Source1 : blah-1.tar.gz
Source2 : fooblah.tar.gz

이 파일들은 SOURCE 디렉토리에 위치합니다(이 디렉토리 구조는 뒤의 "소스 디렉토리 트리" 단락에서 다룰 것입니다.)

- Patch : 패치를 찾을 수 있는 위치입니다. 다시 다운로드 받을 때 필요합니다. 주의: 여기서의 파일 이름은 "여러분의" 패치를 만들 때 사용하는 것과 일치하여야 합니다. 여러 소스에서 여러 패치 파일을 가지고 있을 때 주목할 필요가 있습니다.
  Patch0 : blah-0.patch
  Patch1 : blah-1.patch
  Patch2 : fooblah.patch
  이 파일들은 SOURCES 디렉토리 안에 있어야 합니다.
- Copyright : 이 줄에서는 패키지의 저작권을 알려줍니다. 여러분은 APL, BSD, MIT, 공개(public domain), distributable, 또는 상용 (commercial)과 같이 쓸 수 있습니다.
- BuildRoot : 이 줄에서는 새로운 패키지를 설치하고 만드는 root'' 디렉토리를 지정하도록 합니다. 여러분은 설치하기 전에 여러분의 패키지를 테스트하는데 이를 사용할 수 있습니다.
- Group : 이 줄은 (레드햇 glint'' 와 같은) 시각적 설치 프로그램에서 특정한 프로그램이 트리 구조에서 어디에 위치하는지 알려줍니다. 그룹 트리는 현재 이러한 구조를 가지고 있습니다.

Applications
  Communications
  Editors
    Emacs
  Engineering
  Spreadsheets
  Databases
  Graphics
  Networking
  Mail
  Math
  News
  Publishing
    TeX
Base
  Kernel
Utilities
  Archiving
  Console
  File
  System
  Terminal
  Text
Daemons
Documentation
X11
  XFree86
    Servers
  Applications
    Graphics
    Networking
  Games
    Strategy
    Video
  Amusements
  Utilities
  Libraries
  Window Managers
Libraries
Networking
  Admin
  Daemons
  News
  Utilities
Development
  Debuggers
  Libraries
    Libc
  Languages
    Fortran
    Tcl
  Building
  Version Control
  Tools
Shells
Games

- %description 이것은 헤더 아이템이 아니지만, 여기서 설명해 두어야 합니다. 여러분의 패키지, 서브 패키지마다 설명을 하나씩 필요로 할 것입니다. 이것은 패키지에 대해서 참고적인 설명을 쓰는데 사용하는 것이고 여러 줄에 걸쳐 쓸 수 있습니다.

4 준비 (Prep)

이것은 명세 파일의 두번째 단락입니다. 이것은 소스를 빌드 할 준비를 하는데 쓰입니다. 여기에서는 여러분이 소스 패치, make 실행과 같은 셋업하는데 필요한 것들을 할 수 있습니다.
한가지 주의할 점: 각각 단락에는 실행할 쉘 스크립트가 위치하여야 합니다. 여러분은 간단히 소스를 풀고 패치할 쉘스크립트를 만들어 %prep 뒤에 위치 시킬 수 있습니다. 여기에서 우리는 도움이 되도록 매크로를 만들어 두었습니다.
매크로의 첫 번째는 %setup 매크로입니다. 이것은 간단한 양식으로써 (명령행 옵션은 없습니다), 소스를 풀고 소스 디렉토리로 들어가는 것입니다. 여기에는 다음과 같은 옵션이 있습니다.
-n name 에서는 리스트된 이름에 빌드할 디렉토리의 이름을 정하는데, 기본값은 $NAME-$VERSION입니다. 다른 가능성이 있는 $NAME, ${NAME}${VERSION} 또는 사용하는 tar 파일

이 올 수도 있습니다. (명세 파일 안에 있는 $' 변수는 실제 변수가 아니라는 점에 주의하기 바랍니다. 그것은 실제로 샘플 이름이 위치할 곳을 나타내는데 쓰입니다. 여러분은 변수가 아닌 패키지의 실제 이름과 버전을 사용할 필요가 있습니다.)
-c untar를 실행하기 전에 디렉토리를 만들고 그곳으로 이동하는 것입니다.
-b # 는 그 디렉토리로 이동하기 전에 소스#의 압축을 풀 것입니다.(untar) (-c와 함께 사용할 수는 없습니다.) 이것은 여러 개의 소스 파일이 있을 때만 유용합니다
-a # 는 디렉토리로 이동한 후에 소스#의 압축을 풀 것입니다.
-T 옵션은 압축을 푸는 기본 기능을 무시하고 압축 풀린 소스 파일을 얻기 위하여 -b 0 또는 -a 0 를 필요로 합니다. 부차적인 소스가 있을 때 이 옵션이 필요합니다.
-D 는 소스를 풀기 전에 디렉토리를 지우지 않는 옵션입니다. 이것은 여러분이 하나 이상의 셋업 매크로를 가지고 있을 때만 유용합니다. 이것은 셋업 매크로 중 첫번째 것을 사용한 후에 쓰입니다. (첫번째에 있으면 절대 안됩니다.)
사용 가능한 매크로중 다음으로는 %patch 매크로입니다. 이 매크로는 소스에 패치를 가하는 과정을 자동화 하는 것을 돕습니다. 여기에는 몇 가지 옵션이 있습니다.
는 패치 파일로 패치 # 를 적용합니다.
-p #는 패치 명령(patch(1))을 strip할 디렉토리의 수를 지정합니다.
-P 의 기본 기능은 패치를 적용하는 것입니다. 이 플래그는 기본 기능이고 압축 풀린 메인 소스 파일을 얻기 위해서 0이 하나 필요합니다. 이 옵션은 첫 번째 매크로와 다른 숫자를 필요로 하는 두번째 %patch 매크로에서 유용합니다.
여러분은 역시 실제 명령을 내리는 대신 %patch# 를 쓸 수 있습니다: %patch # -P
%build 매크로에서 여러분이 포함하고자 하는 모든 것(다음 단락에서 논의할 것 입니다.)은 쉘을 통하여 실행하는 것입니다.

⑤ 빌드

이 단락에서는 실제로 어떤 매크로가 있는 것은 아니다. 여러분은 압축 풀린 소스를 가지고 있을 때 사용하기 원하는 소프트웨어를 빌드하고, 그것을 패치하고 그 디렉토리로 이동하는 등의 어떠한 명령이든지 여기에 넣어야 한다. 이것은 명령들의 쉘에 전달되는 또다른 셋으로써, 어떠한 쉘 명령이든지 (설명을 포함해서) 여기에 쓸 수 있다. 여러분의 현재 작업 디렉토리는 각각의 단락마다 소스 디렉토리의 최상위 레벨 디렉토리로 리셋되므로, 따라서 이것을 기억하기 바란다. 여러분은 필요하다면 서브 디렉토리로 이동할 수 있다.

⑥ 설치

이것 역시 실제 어떠한 매크로가 아닙니다. 여러분은 기본적으로 설치하는데 필요한 어떠한 명령이든지 여기에 넣기를 원할 것입니다. 여러분이 빌드하는 패키지 안에서 make install을 사용할 수 있다면, 여기에 넣어 두도록 합니다. 아니면, 여러분은 make install에 쓰일 makefile을 패치하거나 make

install을 여기서 할 수 있습니다, 또는 수동적인 쉘 명령으로 설치할 수도 있습니다. 여러분은 현재 디렉토리가 소스 디렉토리의 가장 상위 디렉토리가 된다는 것을 생각하여야 합니다.

⑦ 설치와 제거의 선행/후행 스크립트

여러분은 바이너리 패키지의 설치나 제거 전후에 스크립트를 넣을 수 있습니다. 주된 이유는 공유 라이브러리를 담고 있는 패키지를 설치하거나 제거하고 나서 ldconfig와 같은 명령을 실행하기 위해서입니다. 각각의 스크립트에 대한 이 매크로들은 다음과 같습니다.

%pre 설치하기 전에 실행되는 스크립트입니다.
%post 설치한 후에 실행되는 스크립트입니다.
%preun 제거하기 전에 실행되는 스크립트입니다.
%postun 제거한 후에 실행되는 스크립트입니다.

이 단락의 내용은 #!/bin/sh 를 필요로 하지 않더라도 어떠한 쉘 스타일의 스크립트가 될 수 있습니다.

⑧ 파일

여기는 여러분이 반드시 파일을 반드시 리스트해야 하는 단락입니다. RPM은 make install의 결과로 어떠한 바이너리가 설치되는지 알 방법이 없습니다. 이것을 할 수 있는 방법은 "없습니다". 어떤 이는 패키지를 설치 전후에 find를 실행하기를 제의하기도 하지만, 다중 사용자 시스템에서는, 패키지 빌드가 이루어지는 동안 패키지 자체와는 아무런 관련이 없는 다른 파일이 생성될 수 있기 때문에 받아들이기 어렵습니다.
여기에는 특별한 작업에 사용 가능한 몇 가지 매크로가 있습니다.
%doc 여러분이 바이너리로 설치하기를 원하는 소스 패키지 내의 문서를 표시하는데 사용됩니다. 문서는 /usr/share/doc/$NAME-$VERSION-$RELEASE에 설치될 것입니다. 여러분은 매크로를 써서 명령행에서 여러 문서를 리스트하거나, 모두 각각 매크로를 써서 리스트할 수도 있습니다.
%config 는 패키지에서 설정 파일을 표시하는데 사용합니다. 여기엔 sendmail.cf, passwd와 같은 파일을 포함합니다. 여러분이 나중에 설정 파일을 담고 있는 패키지를 제거하고자 한다면, 수정하지 않은 파일은 모두 제거되고 수정된 파일은 .rpmsave 를 붙여 이름을 바꾸어 둡니다. 여러분은 역시 이러한 매크로로 여러 개의 설정 파일을 리스트할 수 있습니다.
%dir 파일 안의 패키지에 포함되는 파일 리스트 안의 단일 디렉토리를 표시합니다. 기본값으로, 여러분은 디렉토리 이름을 %dir 없이 나열할 수 있습니다. 디렉토리의 모든것은 파일 리스트 안에 포함되고 나중에 패키지의 한 부분으로 설치됩니다.
%files -f 〈filename〉 로는 소스의 빌드 디렉토리 안에 있는 몇몇 임의의 파일에서 여러분의 파일을 리스트 할 수 있습니다. 이는 여러분이 파일 리스트를 직접 빌드할 수 있는 패키지를 가지고 있는 경우에 좋습니다. 여러분은 여기 있는 파일 리스트만 포함시키고, 여러분은 특별히 파일을 리스트할 필요가 없습니다.

파일 리스트에서 가장 주의해야 할 것은 디렉토리 리스트이다. 여러분이 실수로 /usr/bin을 써 두었다면, 여러분의 바이너리 패키지는 여러분 시스템의 /usr/bin 안의 모든 파일을 담게 될 것이다.

⑨ 빌드하기

소스 디렉토리 트리

여러분에게 가장 필요한 것은 적절히 맞추어진 빌드 트리입니다. 이것은 /etc/rpmrc 파일에서 설정 가능합니다. 대부분의 사람들은 /usr/src 를 사용할 것입니다.

여러분은 빌드 트리를 만들기 위해 다음과 같은 디렉토리를 생성할 필요가 있습니다.

BUILD 는 RPM에 의해서 모든 빌드가 이루어지는 디렉토리입니다. 여러분은 특정한 곳에서 빌드 테스트를 할 필요는 없지만, 이 디렉토리가 RPM이 빌드할 위치입니다.

SOURCES 오리지널 소스 tar 파일과 패치를 넣어 두어야 하는 디렉토리입니다. 이는 기본적으로 RPM이 참고하는 곳입니다.

SPECS 은 명세 파일이 위치할 디렉토리입니다.

RPMS 는 RPM이 바이너리 RPM을 빌드할 디렉토리이다.

SRPMS 모든 소스 패키지가 놓여질 곳입니다.

빌드 테스트

아마도 여러분이 가장 원하는 것은 RPM 없이 깨끗하게 컴파일되는 소스를 구하는 것이다. 이를 위해서는, 소스를 풀고 $NAME.orig 디렉토리로 이동한다. 그리고 소스를 다시 풉니다. 빌드에서 이 소스를 사용하도록 합니다. 소스 디렉토리로 이동하고 빌드하기 위해서 명령을 따릅니다. 여러분이 편집해야 할 것이 있다면, 패치를 필요로 할 것입니다. 빌드하고 나면, 소스 디렉토리를 지웁니다. 설정 스크립트에서 만들어진 모든 파일들이 지워지는지 확인합니다. 그 다음 다시 소스 디렉토리로 이동합니다. 그 다음 여러분은 다음과 같은 명령을 내립니다.

```
# diff -uNr dirname.orig dirname 〉../SOURCES/dirname-linux.patch
```

이는 여러분이 명세 파일에서 사용할 수 있는 패치를 만드는 것입니다. 여러분이 패치 파일 안에서 보는 "linux"는 동일한지 확인하기 위한 것(identifier)에 불과하다는 것에 주목합니다. 또한 바이너리가 실수로 포함되지 않는지 확인하기 위해서 사용하기 전에 패치 파일을 들여다 보는 것 역시 좋은 습관입니다.

파일 리스트 생성

이제 여러분은 빌드할 소스를 가지고 있습니다. 그리고 빌드하고 설치하는 방법을 알고있습니다. 설치 과정의 출력을 보고 명세 파일 안에서 사용할 파일 리스트를 만듭니다. 우리는 일반적으로 명세 파일을 이러한 과정으로 동시에 만듭니다. 여러분은 초기화된 것을 만들고 쉬운 부분을 채울 수 있습니다. 그리고, 진행할 때 다른 곳을 채워 나갑니다.

RPM으로 패키지 만들기

여러분이 명세 파일을 갖게 되면, 여러분은 패키지를 빌드할 준비가 된 것입니다. 가장 쓸만한 방법으로는 명령행에서 다음과 같은 명령을 내리는 것입니다.

```
# rpm -ba foobar-1.0.spec
```

여기에는 유용한 -b 스위치와 함께 다른 옵션이 있습니다.

p 는 명세 파일의 prep 단락을 실행한다는 것을 의미합니다.

l 은 리스트 체크입니다.

c 는 prep를 하고 컴파일합니다. 이것은 여러분이 어떠한 소스를 빌드해야 할지 정확하지 않을 때 유용합니다. 소스를 빌드하고 RPM을 사용하기 시작할 때까지는 여러분이 소스만 가지고 작업할지도 모르기 때문에 쓸모 없게 보입니다. 그렇지만 RPM을 사용하는데 익숙해지면, 여러분은 이것을 사용할 때, 실례로써 찾을 수 있을 것입니다.

i 는 prep 컴파일, 설치를 합니다.

b 는 prep 컴파일, 설치와 바이너리 패키지만 만듭니다.

a 는 소스와 바이너리 모두 만듭니다.

-b 스위치에는 몇 가지 수정 옵션이 있습니다. 다음과 같습니다.

—short-circuit 은 특정한 단계를 바로 건너뜁니다. (c와 i에서만 쓸 수 있습니다.)

—clean 은 작업이 끝나면 빌드 트리를 지웁니다.

—keep-temps /tmp에 만들어진 모든 임시 파일과 스크립트 을 그대로 둡니다. 여러분은 -v 〉 옵션을 사용하여 실제로 tmp에 어떠한 파일이 만들어지는지 볼 수 있다.

—test 는 실제 어떠한 단계도 실행하지 않습니다, 다만 임시로 보존합니다.

⑩ 테스트

여러분이 소스와 바이너리의 rpm 패키지를 가지고 있으면, 시험해 볼 필요가 있습니다. 가장 좋은 방법은 여러분이 빌드한 것을 다른 머신에서 사용해 보는 것입니다. 결국, 여러분은 make install을 여러분의 머신에서 여러번 해볼 것인데, 그것은 반드시 잘 설치되어 있어야 합니다.

여러분은 패키지를 시험하기 위해 rpm -U 〔패키지 이름〕 를 실행할 수 있습니다. 그러나 여러분은 패키지를 빌드할 때, make install을 하였기 때문에 속을 수 있습니다. 파일 리스트에서 어떤 파일을 빠뜨렸다면, 그것은 제거되지 않을 것입니다. 여러분은 바이너리 패키지를 다시 설치하면 여러분의 시스템은 다시 완전해질 것이지만, rpm은 그렇지 않습니다. rpm -ba 〔패키지 이름〕 을 실행시켰기 때문에, 대부분의 사람들은 rpm -i 〔패키지〕로 설치한다는 것을 확인하고 명심하십시요. 바이너리가 홀로 설치될 때 여러분이 빌드하거나 설치할 때 수행되어야 할 필요가 있는 것 중 하지 않은 일이 있는지 확인하십시요.

3. LILO (Linux boot Loader)

(1) LILO 란?

LILO는 x86용 리눅스에서 가장 많이 사용되는 리눅스 로더입니다. 이것은 도스에서의 멀티 부트와 NT의 부트로더, 그리고 OS/2의 부트로더와 같이 여러 운영체제를 선택적으로 부팅할 수 있도록 합니다. 대부분의 리눅스 배포판 설치시 기본적으로 설치하여 사용할 수 있다는 것과 실제로 별다른 어려움 없이 설정하여 사용할 수 있다는 점에서 많은 리눅스 유저가 사용하고 있습니다.

① LILO를 어디서 구할 수 있는가?

ftp://ftp.ezlink.com/pub/ 에서 최신의 리로를 구할 수가 있습니다.
그러나, 배포판에는 리로가 기본적으로 포함되어 있으며 설치시 자동으로 함께 설치할 수 있게 되어 있습니다. 설사 LILO가 설치되어 있지 않다고 하더라도 배포판CD에서 언제든지 설치 할 수 있습니다.

> 와우리눅스 7.1에서는 레드헷 7.1과 달리, 저널링 파일 시스템인 ReiserFS를 지원함에 따라서, 최신 버전인 20.7 버전의 리로를 택하고 있으며, 미려한 인터페이스를 제공하기 위해 GFX 패치를 가하여 640x480의 그래픽컬 한 화면을 제공합니다.

(2) LILO 설치하기

앞서 언급을 드린 바와 같이 이렇게 리로를 따로 설치할 필요는 없습니다. 단지 새로운 LILO 버젼을 설치하시거나 혹시라도 재설치가 필요할 때, 다음과 같이 설치하기 바랍니다.
먼저 압축을 풀어줍니다. 가장 간단한 방법으로 다음과 같이 하시면 됩니다.

```
# gzip -cd 파일명.gz.tar | tar xvf -
```

그리고 나서 Makefile로 정의되어지는 REWRITE_TABLE로 컴파일하고 인스톨합니다.
그냥 명령어라인에서 REWRITE_TABLE을 써 넣고 엔터를 치시면 됩니다.
만일 RPM으로 제공된 LILO 패키지를 설치한다면, RPM 사용하기 섹센에서 다루었던 RPM 명령이 그대로 적용되므로, 간단히 RPM 명령을 사용하여 패키지를 설치하고, 설정하면 됩니다.
그런 후에 /etc/lilo.conf 를 편집하고 /sbin/lilo 명렁을 내리셔서 MBR에 이것을 설치하면 됩니다.

- LILO는 두 군데 설치가 가능합니다. 즉, 일반적으로 MBR(master boot record (/dev/hda))이나 리눅스 root partition (대체로 /dev/hda1 나 /dev/hda2)을 특정하여 설치가 가능하다. 여기에서 /etc/lilo.conf안에 있는 'boot=' 지시어는 LILO에게 주 부트로더를 어디에 놓아야 하는 지를 말합니다.
- 만약 여러분의 또 다른 운영체제가 당신의 하드 드라이브에 설치되도록 한다면 당신은 LILO를 MBR보다는 루트파티션에 인스톨 하는 것이 좋습니다. 이 경우에 당신은 그 파티션을 fdisk의 a 명령어를 사용하므로써 또는 cfdisk의 b 명령어로써 bootable 로 한다. 만약 여러분이 마스터 부트 섹터를 덮어쓰지 않는다면 리눅스와 LILO를 언인스톨 할 때 좀더 쉽습니다.

〔참고〕

1] LILO 설치시 주의할 점

LILO가 시스템을 부팅할 때 리눅스 커널을 디스크(IDE drive, floppy 또는 다른 것들)로부터 읽어들이기 위해 'BIOS calls'를 이용합니다. 그러므로, 커널은 BIOS에 의해서 연결될 수 있는 어떤 장소에 있어야만 합니다.
부팅시 LILO는 파일시스템의 데이타를 읽을 수가 없습니다. 여러분이 /etc/lilo.conf에 넣는 어떤 패스네임이라도 인스톨시킬 때(/sbin/lilo 명령을 내릴 때) 정해(resolved)집니다. 인스톨시에는 프로그램이 어떤 섹터들이, 운영체제가 읽어들이기 위해 사용되는 파일들에 의해, 사용될 지 리스트하는 테이블들을 만듭니다. 결과적으로 이 모든 파일들은 BIOS에 의해 연결될 수 있는 파티션에 있어야만 합니다. (이 파일들은 대체적으로 /boot 디랙토리에 위치하며, 이것은 여러분의 리눅스 시스템의 root 파티션만이 BIOS를 거쳐서 연결될 필요가 있다는 것을 의미합니다.)
BIOS에 기초하고 있는 또 다른 결과는 당신이 LILO 셋업을 편집하는 어떤 때라도 당신이 로더를 재인스톨 해야 한다는 것입니다. (/sbin/lilo 명령을 다시 실행하여 줍니다). 당신이 당신의 커널을 재컴파일해서 당신의 이전 이미지를 덮어씌울 때마다 당신은 LILO를 재인스톨해야 합니다.

2] 어떻게 대용량 커널을 다룰 것인가?

만약 zImage 커널을 컴파일하는데 너무 커서 0.5MB (이것은 커널 2.1이후로 공통된 것입니다.) 에 맞출 수가 없다면, 'big zImage' 를 만들기 위해 'bzImage'를 만들어야 할 것입니다. 큰 커널 이미지를 부팅하기 위해 특별한 것은 필요하지 않습니다. 그러나, LILO 버젼 18이상이 필요합니다. 만약 그것보다 오래된 것이 인스톨되어 있다면, 새로운 것으로 업그래이드해야 합니다.

(3) LILO 설정

앞서 기술한 바와 같이 LILO는 대체적으로 자동으로 설치되도록 되어 있습니다. 그러나, 만약 특정 운영체제를 더 사용하게 되거나, 경우에 따라서는 LILO를 수동으로 편집하고 재설치하는 과정이 필요하게 됩니다. 그래서, 여기서는 기본적인 편집과 개념에 대하여 알아보도록 하겠습니다.

① LILO 설정 파일의 편집

LILO를 수작업으로 설정해야 한다면, /etc/lilo.conf 파일을 편집해야 할 것입니다. 여기서 /etc는 lilo.conf 파일이 있는 디랙토리이고 그 파일을 열기 위해서 전체경로를 적어준 것입니다. 다음과 같이 파일을 열어 편집을 시작한다.

```
# vi /etc/lilo.conf
```

② LILO 편집을 위한 기본적인 설명

LILO를 편집하기 위하여 몇가지 기본적인 개념들을 알고 있어야 합니다.

```
boot=/dev/hda append="mem=128M hdc=621,128,63 hdd=827,64,63"
map=/boot/map install=/boot/boot.b prompt timeout=50
image=/boot/vmlinuz-2.0.36-1kr
        label=linux root=/dev/hdb5 inird=/boot/initrd-2.0.36.img read-only
other=/dev/hda1
        label=dos table=/dev/had
```

boot=

LILO가 설치 될수 있는 곳

1. /dev/hda라면 첫번째 IDE 하드디스크의 MBR에 설치
2. /dev/sda라면 SCSI 첫번째 부팅 가능한 하드디스크의 MBR에 설치
3. 부팅 가능한 첫번째 플로피 장치명인 경우 /dev/fd0

MBR에 설치하면 LILO가 시스템의 기본 부트 로더가 됩니다.

prompt , timeout=
prompt는 boot: 프롬프트를 표시합니다. timeout은 지정한 시간 안에 키보드 입력이 없을 때 첫번째 등록된 부팅 항목 또는 default= 설정행에서 지시하는 항목으로 부팅됩니다. 시간 설정은 1/10초 단위로 설정 되므로 50 이면 5 초를 의미합니다.

append=""
커널옵션은 boot: 프롬프트에서도 입력할 수 있지만 매번 입력해야 한다면 이곳에 적어줍니다.

default=
LILO는 dsfault=설정행이 없으면 image= 또는 other= 설정행으로 첫번째 등록된 항목으로 일정시간 후에 부팅합니다.

image=
리눅스 커널 이미지를 등록하는 부분입니다.

label=
부팅하고자 구별하는 인식단어를 표시합니다.

root=
리눅스가 설치된 루트 장치명을 표시합니다.

read-only
리눅스는 대부분의 경우 일단 루트 파티션을 읽기 전용으로 마운트합니다.

initrd=
초기화에 필요한 루트 디스크 이미지를 표시합니다.

other=
리눅스 이외의 도스/윈9X/윈NT/윈2K 등의 운영체제가 설치된 파티션 위치를 지정하고 등록하여 설정합니다.

③ LILO 의 기본적 편집

대부분 LILO 인스톨은 다음 예들과 같이 파일을 작성하여 사용합니다. 물론 처음 자동으로 설치되어 나올 때도 바로 이런 형태로 작성되어 있습니다. 여기에서는 기본적으로 하나의 드라이브 또는 두개의 드라이브에 두개의 운영체제를 설치했을 때의 편집 예를 소개합니다.
다음은 하나의 드라이브에 리눅스와 도스(또는 윈95)를 설치하여 사용할 경우의 예제입니다.

```
boot = /dev/hda                 # 또는 root 파티션
delay = 10                      # delay, in tenth of a second (so you can interact)
vga = 0                         # optional. Use "vga=1" to get 80x50
#linear                         # try "linear" in case of geometry problems.
image = /boot/vmlinux           # 당신의 커널 zImage 파일 이름
root = /dev/hda1                # 당신의 root 파티션
    label = Linux               # 또는 마음에 드는(fancy) 이름
    read-only                   # root를 read-only로 마운트
other = /dev/hda4               # 만약 있다면 당신의 dos 파티션
    label = dos                 # 또는 또다른(non-fancy) 이름
```

만약 원한다면 여러가지 image 와 other 부분들을 가질 수 있습니다. lilo.conf안에 작성된 여러가지 커널 이미지들을 가지는 것은 낯선 일은 아닙니다. 적어도 최신의 개발된 커널을 유지하려고 한다면 말입니다.

다음은 또 다른 LILO 편집 예제입니다. 이 예제는 리눅스 루트 파티션은 /dev/hda2에 자리를 잡고 있고, MS-DOS는 /dev/hdb1에 (두번째 하드 드라이브) 설치되어 있습니다.

```
# Tell LILO to install itself as the primary boot loader on
/dev/hda.
boot = /dev/hda
# The boot image to install; you probably shouldn't change this
install = /boot/boot.b
# for booting Linux.
image = /vmlinuz                        # The kernel is in /vmlinuz
label = linux                           # Give it the name "linux"
root = /dev/hda2                        # Use /dev/hda2 as the root filesystem
vga = ask                               # Prompt for VGA mode
append = "aha152x=0x340,11,7,1"         # Add this to the boot options,
                                        # for detecting the SCSI controller
                                        # for booting MSDOS
other = /dev/hdb1                       # This is the MS-DOS partition
label = msdos                           # Give it the name "msdos"
```

일단 /etc/lilo.conf 파일을 편집한 후에는 root의 권한을 가지고서 /sbin/lilo 를 실행시켜야 합니다. 그렇게 하므로서 하드 드라이브에 LILO가 설치될 것입니다. 또한 명심해야 할 것은, 커널을 재컴파일 한 후에도 부트 로더에게 알려주어야 하며, 그렇게 하려면 /sbin/lilo를 다시 실행시켜야 한다는 것입니다.
여러분은 이제 하드 드라이브로부터 시스템을 다시 부팅할 수 있습니다. 디폴트로서 LILO는 설정파일에 나열된 것 중 첫번째 오퍼레이팅 시스템을 부팅할 것입니다. 이 경우에는 리눅스가 첫번째로 나열되어 있으므로, 다른 오퍼레이팅 시스템을 선택하여 부팅하기 위해서는 부팅시에 아래와 같은 것이 나타날 것입니다. 나타나지 않는다면 부팅시에 shift또는 control 키를 눌러봅니다.

Boot:

여기에 부팅할 오퍼레이팅 시스템명을 (설정파일의 label 라인에 명시되어 있다. 여기에서는 linux 또는 msdos) 입력합니다. 또는 목록을 보려면 tab 키를 누릅니다.

와우리눅스 7.1에 사용된 리로는 사용의 편리함을 더하기 위해, 라벨명이 나열되므로, 화살표키를 사용하여 선택한 후, 엔터키를 누르면 해당 운영체제로 부팅이 가능하다.

〔참고〕

The 'init=' Argument

커널은 부트시에 일반적으로 init' 프로그램으로 시작합니다. 이 것은 컴퓨터를 설정하는 getty 같은 프로그램을 실행시키거나 rc' 스크립트나 그와 비슷한 것들을 실행함으로서 가능합니다. 커널은 처음에 /sbin/init를 찾습니다. 그 다음엔 /etc/init를 찾습니다. 그리고 마지막 수단으로 /bin/sh를 사용하려 할 것입니다. (/etc/rc도 가능) 예를 들면 init 프로그램이 중지되고 부트 할 수 없어졌다면 부트시에 직접 쉘을 가동시키기 위하여 간단히 init=/bin/sh 인자만을 사용하면 됩니다. 그리고 잘못되었던 프로그램을 제대로 되돌리면 됩니다.

④ 여러개의 운영체제 멀티부팅을 위한 LILO의 편집

① 리눅스, 윈도우즈9x, 윈도우즈 2000(NT 4.0)

이것은 /etc/lilo.conf 파일을 편집하고 LILO를 재인스톨하는 것에 관련된 것입니다. 다음 파일에 미리 존재해야 합니다.

boot=/dev/hda

리눅스에 대한 것이 또한 이미 존재해야만 합니다.

image=/vmlinuz (해당 이미지의 이름을 정확히 기재합니다.)

root=/dev/hdc1

label=Linux

이제 윈95/98에 대한 것을 편집하도록 합니다.

other=/dev/hda1

table=/dev/hda

label=Windows95

이제 윈 2000(NT 4.0)에 대한 것을 편집하도록 합니다.

other=/dev/hdb1

table=/dev/hda

loader=/boot/any_d.b

label=WindowsNT

이 부분은 2000(NT 4.0)이 다른 디스크에 설치될 때의 예 입니다. 그리고, 만약 같은 디스크에 파티션으로 존재한다면 위의 윈95/98의 예를 따르면 됩니다.

이제 LILO를 다시 설치하시고(/sbin/lilo) 컴퓨터를 재부팅합니다. 만약 모두가 잘 된다면 당신은 윈95/98와 윈 2000(NT 4.0)와 리눅스를 LILO로부터 선택할 수 있도록 되는 것입니다. 또한, 리눅스와 윈 2000(NT 4.0)의 멀티 부팅시나 리눅스와 윈9x간의 멀티부팅시에도 위의 것을 응용하면 됩니다.

② 리눅스, 윈도우즈95/98, 또다른 윈도우즈 95/98 내지는 도스 설치

각각 다른 언어의 윈도우를 설치하여 부팅하려 할 때나 MS사의 여러 운영체제를 설치할 때 문제가 생기곤 합니다. 이것을 해결하기 위해 LILO를 사용할 수 있습니다.

윈95/98와 dos(또는 다른 윈도우)는 하나 이상의 파티션이 활성화(active)로 표시되면 문제가 생길 수 있습니다(confused). 그래서, 그들이 부팅하기 전에 부트 메니저가 그들의 파티션이 활성화해서 어떤 다른 것들도 표시되지 않도록(unmark) 할 필요가 있습니다. 그래서, 활성화 파티션을 하나만 지정하고 부팅하면 드라이브명을 운영체제에서 다시 표시하도록 합니다.

또한 이들은 몇가지 이유로 부팅된 디바이스에 파티션들을 다시 명시합니다(relabel). 그래서, OS는 항상 C드라이브에 위치하도록 나타납니다. 예를 들어 당신이 도스를 파티션 E에 설치한다고 하더라도 그것은 그것이 부팅할 때 파티션 C로 나타나게 됩니다. 이 경우 LILO를 이용하여 하드 디스크 하나 또는 하드디스크 두 개에 나누어 설치하여 LILO를 설치하여 부팅이 가능하도록 할 수 있습니다. 이 경우 다른 하드디스크에 각각 다른 윈도우95/98을 설치합니다.

가. 하나의 하드 디스크에 설치하기

- 당신의 드라이브에 세개의 파티션을 만들기 위해 리눅스 fdisk나 파티션 매직을 사용합니다. 하나의 파티션에 윈95를 설치하고 또다른 하나에 도스(또는 또다른 윈도우)를 설치합니다. 그리고, 세번째에 리눅스를 설치합니다. 만약 당신이 시작할 드라이브에 하나의 파티션(도스)만을 가지고 있다면 파티션 매직이 그것을 세개로 나눌 수 있는 가장 쉬운 방법일 것입니다. FIPS는 무료로 같은 기능을 제공하지만 기술면에서 조금 떨어집니다.
- 부팅시 엑티브 표시(flag)를 업데이트하는 능력을 가지 유일한 버전인 lilo.17.tar.gz 사본을 가져옵니다.

```
ftp://ftp.ezlink.com/pub/lilo.17.tar.gz
```

다운받은 리로를 Makefile 로 정의되는 REWRITE_TABLE로 컴파일하고 인스톨합니다.
/etc/lilo.conf 을 다음과 같이 편집하여 설치하고, /sbin/lilo를 실행하여 드라이브의 MBR 기록을 업데이트합니다.

```
boot = /dev/sda                  # 드라이브명
compact delay = 5                # optional, for systems that boot very quickly
vga = normal                     # force sane state
ramdisk = 0                      # paranoia setting
root = current                   # use "current" root
image=/boot/vmlinuz              # 커널 이름
    append="aha1542=0x230 ro"
    label=linux
other=/dev/sda1
    table=/dev/sda rewrite-table     # 덧붙여진 곳
    label = dos
other=/dev/sda2
    table=/dev/sda rewrite-table     # 덧붙여진 곳
    label = w95
```

나. 두개의 하드 디스크에 설치하기

lilo가 버젼 17이상이라면 위와 같이 한 하드에 설치가 가능하다. 그러나, 경우에 따라서 다음과 같이 하드 두개에 나누어 설치하여 LILO를 통해 모두 부팅이 가능하도록 할 수 있습니다. 즉, 다른 하드에 윈95/98을 설치하여 사용할 수 있습니다.

우선 첫 번째 하드디스크의 첫 파티션에 윈도우95/98(도스)를 설치하고, 나머지 파티션에 리눅스를 설치합니다. 다른 하드디스크에 윈도우95/98을 설치합니다. 물론, 각각의 운영체제는 부팅 가능하도록 설정합니다. 그리고, 리눅스로 부팅해서 lilo.conf를 편집합니다.

- 파티션 테이블

```
/dev/hda1 * win9x partition (C: or D: depending upon boot)
/dev/hda2 / (Linux root) # 리눅스의 fdisk로 파티션을 잡아야 합니다.
/dev/hda3 swap
/dev/hdb1 * Win9x partition (C: or D: depending upon boot)
〔/dev/hdb2 DOS partition (E: always) - 옵션〕
```

* 를 가진 파티션은 fdisk에 의해 부팅할 수 있도록('active') 설정된 상태.

- /etc/lilo.conf 의 예제

```
boot=/dev/hda
〔append="mem=128M hda=621,128,63 hdb=827,64,63"#대용량 하드웨어 옵션〕
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
default=linux
image=/boot/vmlinuz
  label=linux
  root=/dev/hda2
  read-only
other=/dev/hdb1
```

```
   table=/dev/hdb
     label=win98_2
 other=/dev/hda1
     table=/dev/hda
     label=win98_1
```

⑤ LILO 지우기

LILO가 부트섹터에 설치될 때 그것의 백업이 /boot/boot.xxyy로 생깁니다. xxyy는 hex로 그 디바이스의 major 그리고 minor 숫자이다. 그것은 ls -l /dev/device 를 실행하여 확인할 수 있습니다. 예를 들어 /dev/hda (major 3, minor 0)의 처음 섹터의 숫자들은 /boot/boot.0300에 저장됩니다. 이에 대해 /dev/fd0 상의 LILO가 /boot/boot.0200를 만들고 /dev/sdb3 (major 8, minor 19)가 /boot/boot.0813를 만듭니다. 이것은 이미 하나가 존재한다면 다시 그 파일을 만들 필요가 없습니다. /boot/에 있는 백업본들은 항상 어떤 릴로를 설치하기 전에 그 상황을 확인합니다.
참고로 다음과 같이 하면 백업된 섹터 이미지를 복원할 수 있습니다.

```
#/sbin/lilo -u /dev/hda
```

만약 Lilo가 /dev/hda에 설치되어 있다면 단지 다음과 같이 해 주면 됩니다.

```
#dd if=/boot/boot.0300 of=/dev/hda bs=446 count=1
```

또는

```
#cat /boot/boot.0300 〉/dev/hda
```

그러나, 이것은 안전하지 못하다. 원래 파티션 테이블 정보를 복원하게 되므로...
가장 간단하고 쉽게 다음과 같이 실행합니다.

```
#/sbin/lilo -U 또는 lilo -u /dev/hda(MBR이 있는 곳)
```

도스의 Fdisk를 사용한다면, 다음과 같이 실행합니다.

```
fdisk /mbr
```


4. etc 디렉토리 아래의 주요 시스템 설정파일

(1) 부팅과정의 이해

/etc 디렉토리 아래는 시스템의 주요 설정에 관한 대부분의 설정파일이 위치하게 되는데, 이 설정 파일을 살펴보기에 앞서 우리는 시스템의 부팅과정을 이해할 필요가 있습니다.

① 전원을 켜는 순간으로부터 init 프로세스가 시작될 때까지

PC 의 전원을 켜면 PC 는 롬에 저장되어 있는 초기화 프로그램을 실행합니다. 초기화 프로그램에 의해 메모리 체크 등이 수행되고 필요한 초기화가 완료되고 나면, 하드 디스크 혹은 다른 부팅 매체 (플로피 디스크나 CDROM 등)의 0 번 섹터의 부트 프로그램을 읽습니다. 보통, 0번 섹터를 MBR (Master Boot Record) 이라고 한다. 여기는 리눅스를 위한 lilo 가 들어 있을 수도 있고, NT 나 OS/2 등 다른 운영체제의 부트 로더가 들어 있을 수도 있습니다. 여기서는 lilo 가 MBR 에 있다고 가정하고 진행합니다. lilo 는 커널을 실행하기 위해서 사용자의 입력을 기다립니다. 여러분이 컴퓨터를 켰을 때 나오는

```
Boot:
```

라는 프롬프트가 그것입니다.
이때 우리는 boot 라는 프롬프트 뒤에 커널에 주고싶은 옵션이나, 부팅하고자 하는 커널의 이미지를 지정해 줄 수 있습니다. 이에 관한 자세한 내용은 lilo 섹션을 참조하십시요.
이제, 사용자는 lilo 에 실행시키기를 원하는 커널의 이미지를 커널에 넘겨주고자 하는 옵션값과 함께 알려줍니다. 그리고, lilo 는 해당하는 커널의 이미지를 로딩해서 실행하게 됩니다. 기본 설정으로는 /vmlinuz 또는 /boot/vmlinuz 이미지가 실행됩니다.
vmlinuz 는 리눅스 커널의 압축 이미지입니다. 여기서 swapper 라고도 불리 우는 프로세스 id 0 인 프로세스가 실행됩니다. 이 프로세스는 운영체제 그 자체라고도 할 수 있는 프로세스로써, 메모리 관리, 디스크 관리, 프로세스 관리 등을 수행합니다. 이 프로세스는 프로세스 id 1 인 init 라는 프로세스를 실행시키고는 본연의 기능인 swapper 로써의 기능을 수행하기 시작합니다.

swapper 는 무슨 일을 하는가?

유닉스 시스템에서 실행되는 모든 프로세스들은 "lifetime"을 가지고 있습니다. 프로세스 생성에서부터 종료시까지, 그 동안 cpu 를 점유하면서 실행되는 시간도 있을 것이고, 할 일 없이 사용자로부터의 입력을 기다리며 "잠들어" 있는 시간도 있을 것입니다.
swapper 프로세스에 대해 설명하면서 이 이야기를 하는 이유는 swapper 가 하는일이 바로, "잠들어" 있는 프로세스를 메모리에서 내려서 디스크 공간에 잠시 "스왑"시켰다가, 그 프로세스가 깨어나야만 할 시기가 오면, 디스크의 프로세스를 다시 메모리로 적재해 주는 등의 일을 하기 때문입니다.

예를 들어서, 지금 시스템에 너무 많은 프로세스가 실행되고 있어서 그 것들이 모두 들어갈 만큼 메모리가 크지 않다고 가정할 때, 시간이 좀 많이 걸리는 I/O 요청을 한 프로세스라든지, 사용자의 입력을 기다리는 프로세스 라든지, 지금 sleep 상태로 있는 프로세스 (예를 들면, httpd...) 와 같은 것들은 지금 당장 메모리에 있을 필요가 없는 것입니다. 그러면, 지금 메모리가 모자라니까 디스크로 "스왑" 을해서 당장 실행되어야 하는 프로세스를 위한 메모리 공간을 늘리게 되는데, 이때 swapper 가 작동을 해서 메모리에 있는 프로세스를 디스크로 잠시 옮겨 두는 (swap out) 것이다. 또, swapper 는 반대로 디스크에 스왑되어 있는 프로세스가 메모리로 적재되어서 실행되어야 할 필요가 있을 때에도 스왑된 프로세스를 메모리로 다시 올리는(swap in) 일을 하기도 합니다.
리눅스의 모든 프로세스는 모두 "부모" 프로세스를 가지고 있습니다. 즉, 그 프로세스를 생성시킨 프로세스가 존재한다는 인데 단, 하나 pid 가 0 인 swapper 프로세스만은 부모 프로세스가 존재하지 않고, lilo 등에 의해서 "수동"으로 실행이 됩니다. 그 외 나머지 프로세스는 모두 fork() 시스템 콜과 exec() 시스템 콜을 이용해서 생성이 됩니다.

② init 프로세스와 inittab 파일

프로세스 id 1 번인 init 프로세스는 사용자들을 위해서 시스템을 설정하게 됩니다. pid 0 인 프로세스가 초기화한 커널을 바탕으로 나머지 작업을 수행하는 것입니다.
init 프로세스가 하는 일은 파일시스템의 구조를 검사하고, 파일시스템을 마운트하고, 서버 데몬을 띄우고, 사용자 로그인을 기다리고, 사용자가 로그인 하면, 사용자를 위한 쉘을 띄우는 것까지의 것들입니다.
init 가 처음 시작해서 수행해야 할 작업들을 설정한 파일은 /etc/inittab 파일입니다. init 는 새로운 실행레벨에서 실행할 프로세스를 결정하기 위해서 이 파일을 참조합니다. 다시 말하면, inittab 파일은 시스템의 상태에 따라서 해당하는 런레벨에서 init 프로세스가 수행해야 할 일들을 서술해 놓은 파일입니다.
init 프로그램은 inittab 파일을 참조하여서 모든 새로운 런레벨에서 실행할 수 없는 프로세스가 만약 지금 실행 중이면, 그 프로세스를 죽이고, 새로운 런레벨에서 실행해야만 하는 프로세스 중 현재 실행되고 있지 않은 프로세스는 새로이 실행을 시킵니다.

```
# inittab       This file describes how the INIT process should set up
#               the system in a certain run-level.
#
# Author:       Miquel van Smoorenburg,
#               Modified for RHS Linux by Marc Ewing and Donnie Barnes
#
# Default runlevel. The runlevels used by RHS are:
# 0 - halt (Do NOT set initdefault to this)
# 1 - Single user mode
# 2 - Multiuser, without NFS (The same as 3, if you do not have networking)
# 3 - Full multiuser mode
# 4 - unused
# 5 - X11
# 6 - reboot (Do NOT set initdefault to this)
#
id:3:initdefault:
# System initialization.
si::sysinit:/etc/rc.d/rc.sysinit
l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6
# Things to run in every runlevel.
ud::once:/sbin/update
# Trap CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
# When our UPS tells us power has failed, assume we have a few minutes
# of power left.  Schedule a shutdown for 2 minutes from now.
# This does, of course, assume you have powerd installed and your
# UPS connected and working correctly.
pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"
# If power was restored before the shutdown kicked in, cancel it.
pr:12345:powerokwait:/sbin/shutdown -c "Power Restored; Shutdown Cancelled"
# Run gettys in standard runlevels
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6
# Run xdm in runlevel 5
# xdm is now a separate service
x:5:respawn:/etc/X11/prefdm -nodaemon
```

〔inittab 파일〕

inittab 파일을 한줄씩 살펴보도록 하겠습니다.
으로 시작하는 줄은 주석문으로 시스템 설정에 아무런 영향을 주지 않습니다. 간략히 inittab 파일의 용도와 저자가 나오고, 각 런레벨(ren level) 에 대한 설명이 나열됩니다.

- 렌레벨 0 은 시스템을 종료할 때 사용되는 런레벨입니다.
- 런레벨 1 은 싱글 유저 모드에서 사용되는 레벨입니다. 여러분이 lilo: 프롬프트에서 linux single 이라고 입력하면 런레벨 1 에서 리눅스가 시작하게 됩니다. 이 때는 디폴트로 root 로 로그인되며, 대다수의 중요한 데몬들은 실행되지 않게 설정되어 있습니다.
- 런레벨 2 는 NFS 를 지원하지 않는 다중 사용자 모드를 정의하며, 런레벨 3 은 네트워킹을 지원하는 다중 사용자 모드입니다. (디폴트 런레벨로 주로 지정됩니다.)
- 런레벨 4 는 여러분이 나름대로 정의해서 쓸 수 있는 런레벨입니다. 필요에 따라서 실행시키기 원하는 데몬이나 서비스를 /etc/rc.d/rc4.d 디렉토리 밑에 설정함으로써 여러분만의 특성을 갖는 고유의 런레벨을 정의할 수 있습니다.
- 런레벨 5 는 X 를 실행시키기 위한 런레벨로 약속되어 있습니다.
- 런레벨 6 은 시스템을 재부팅 시키도록 정의된 런레벨입니다.

만약, inittab 파일의 처음에 나오는
id:3:initdefault:
의 라인을
id:6:initdefault:
혹은
id:0:initdefault:
등으로 지정해 버리면, 여러분의 리눅스 시스템은 부팅하자마자 종료되어 버리거나, 재시작을 거듭 반복하게 되어 버리므로 주의하시기 바랍니다.
이제 inittab 파일의 형식을 살펴 보도록 하겠습니다. 자세히 inittab 파일을 보신 분이라면 모든 줄이 다음과 같은 형식으로 되어 있다는 것을 아셨을 겁니다.
id:run-levels:action:process
제일 처음에 나오는 id 는 해당 state를 구분하기 위한 레이블이라고 보시면 됩니다. 그리고, 그 다음의 rul-levels 는 그 줄의 내용을 적용하기 위한 런레벨의 목록입니다. 그리고, action 은 그 줄(엔트리), 엔트리에 의해 실행되는 프로세스를 어떻게 할 것인가에 대한 설명입니다. 마지막으로 나오는 process 는 프로세스의 실행파일의 경로와 프로세스에 넘겨줄 인수입니다. (쉘에서 실행시키는 명령어와 같은 형식이라고 생각할 수 있습니다. 즉, 해당 엔트리를 실행할 때 process 부분에 나오는 명령어로 실행하라는 뜻입니다.)
action 부분에 올 수 있는 키워드는 다음과 같습니다.

- wait : 프로세스를 실행하고, 다음 줄의 엔트리로 가지 말고, 실행한 프로세스가 종료하기까지 기다리라는 뜻입니다.
- respawn : 프로세스를 실행하고, 그 프로세스가 죽게 되면, 다시 실행시키라는 의미입니다. 주로 getty 등의 프로세스입니다.

- initdefault : 디폴트 런레벨을 지정하겠다는 뜻입니다. 위에 예시한 inittab 파일의 첫 줄에 이 키워드가 나오는데, 해석하면, 런레벨 3 을 디폴트 런레벨로 지정한다는 의미입니다. 즉, initdefault 엔트리는 시스템의 부트 프로세스가 종료된 후에 진입할 런레벨을 가리키는 엔트리입니다. process 필드는 아무런 의미가 없게 됩니다.
- off : 아무것도 하지 말라는 뜻입니다.
- once : 이미 실행되고 있는 프로세스라면 실행하지 말고, 실행되고 있지 않으면 단지 한번만 실행시키라는 뜻입니다. 단, wait 처럼 기다리거나 하지는 않 도록 지정합니다.
- boot : 시스템 부팅시에 실행되어야 할 프로세스를 가리킵니다. 런레벨 필드는 아무런 의미가 없게 됩니다. (무시됩니다.)
- bootwait : 프로세스가 시스템 부팅시에 실행되도록 지정합니다. 단지, init 가 그 프로세스가 종료되기를 기다린다는 점에서 boot 와 다릅니다. 예를 들어 /etc/rc 와 같은 것이 있습니다.
- sysinit : 프로세스가 시스템 부트시에 실행되게 합니다. 그리고, 이 엔트리는 다른 boot 나 bootwait 엔트리들이 실행되기 전에 실행되는 엔트리가 됩니다. 런레벨 필드는 무시합니다.
- powerwait : init 프로세스가 SIGPWR 시그널을 받으면 실행되는 프로세스입니다. SIGPWR 시그널은 전원과 관련해서 무엇인가 문제가 있음을 가리키는 시그널입니다. 이때 init 는 프로세스가 종료되기까지 대기합니다.
- powerfail : powerwait 항목과 마찬가지이지만, 프로세스가 종료되기까지 기다리지않는다는 점이 다릅니다.
- powerokwait : 이 엔트리 역시 init 가 SIGPWR 시그널을 받으면 실행될 프로세스를 지정합니다. 그러나, 이 엔트리에서 지정된 프로세스는 /etc/powerstatus 파일에 OK 라는 단어가 있을 때만 실행됩니다. 즉, 전원이 다시 돌아왔을 때만 실행됩니다.
- ctrlaltdel : init 가 SIGINT 시그널을 받게 되면 실행할 프로세스를 지정합니다. 즉, 시스템 콘솔에서 누군가가 CTRL-ALT-DEL 키를 눌렀을 때 이 엔트 리에서 지정한 프로세스가 실행되는 것입니다.
- kbrequest : 이 엔트리에서 지정하는 프로세스는 init 가 키보드 핸들러로부터 콘솔에서 특수키 조합이 눌려졌다는 시그널을 받으면 실행되는 프로세스입니다. 키맵 파일과 함께 쓰일 수 있습니다.

③ init 그 이후

앞 절에서 설명한 바와 같이 init 프로세스가 자신에게 할당된 초기화 과정을 모두 끝마치고 나면, inittab 파일의 맨 끝부분에 명시된 것처럼 mingetty 혹은 getty 프로세스를 실행시킵니다. 실행시키는 방법은 최초에 pid 0 의 프로세스에 의해 실행된 init 가 fork() 시스템 콜을 한번 수행하고, 그에 의해 생성된 자식 init 가 exec() 시스템 콜을 이용해서 mingetty 등의 프로세스를 수행하게 됩니다. 그리고 나서, mingetty 프로세스는 다시 exec() 를 호출하여 login 프로세스를 수행하고, login 프로세스는 사용자의 userid 와 passwd 를 입력받아서 사용자 인증과정을 거칩니다. 만약, 승인된 사용자라면, /etc/passwd 파일에 명시된 사용자에 해당하는 쉘을 띄우게 됩니다. 쉘을 띄우는 것 또한 exec() 시스

템 콜을 이용해서 입니다. 그러면, 사용자는 자신의 쉘 (주로 bash) 로 작업을 하고, 쉘에서 입력하는 명령어들은 먼저 쉘이 fork() 를 한 후 쉘의 자식프로세스가 exec() 를 하여 실행되고, 종료하는 것입니다.

(2) /etc/rc.d/* 파일

① rc.sysinit 스크립트

이 스크립트는 inittab 정의에 따라, 시스템 초기화시 맨 먼저 딱 한 번 실행되는 초기화 스크립트로 다음과 같은 내용을 포함하고 있습니다.

- 기본적인 path 설정
- /etc/sysconfig/network 파일이 있으면 그 스크립트를 실행시킴
- 키맵의 로딩
- 시스템 폰트의 로딩
- 스왑 영역의 활성화
- 디스크 검사 (fsck)
- /proc 파일시스템의 마운트
- 루트 파일시스템을 rw 모드로 다시 마운트하기
- /etc/HOSTNAME 파일의 설정
- /etc/mtab 파일에 루트와 /proc 파일시스템의 엔트리 추가하기
- 커널 모듈들 로드하기
- 시스템 시간 설정
- 등등....

의 일을 수행합니다. 여기서 이 스크립트를 분석하는 것은 커다른 의미가 없으므로 관심이 있는 분들은 rc.sysinit 파일을 천천히 훑어 보시기 바랍니다. 충분히 이해할 수 있으리라 생각됩니다.

② rc 스크립트

정작, 우리가 주의를 기울일 초기화 스크립트는 각종 데몬들을 실행시키고, 죽이는데 사용되는 /etc/rc.d/rc 스크립트입니다. 이 스크립트는 인수로 실행 레벨을 받습니다.
즉,

```
/etc/rc.d/rc n
```

과 같은 형식으로 실행이 됩니다.
/etc/rc.d 디렉토리 아래는 다음과 같이 각 런레벨에 해당하는 디렉토리들이 있습니다.

```
rc0.d/
rc1.d/
rc2.d/
rc3.d/
rc4.d/
rc5.d/
rc6.d/
```

각각의 디렉토리는 rc런레벨.d 로 이름이 지어져 있습니다. 각 디렉토리 아래에는 해당 런레벨에서 실행할 서비스나 프로세스들이 정의되어 있습니다. 한 rc3.d 디렉토리를 살펴볼 때, 다음과 비슷한 내용을 볼 수 있습니다.

```
$ ll /etc/rc.d/rc3.d
    drwxr-xr-x  ./
    drwxr-xr-x  ../
    lrwxrwxrwx  K05keytable -> ../init.d/keytable*
    lrwxrwxrwx  K15gpm -> ../init.d/gpm*
    lrwxrwxrwx  K15proftpd -> ../init.d/proftpd*
    lrwxrwxrwx  K60atd -> ../init.d/atd*
    lrwxrwxrwx  K60crond -> ../init.d/crond*
    lrwxrwxrwx  K80random -> ../init.d/random*
    lrwxrwxrwx  K89portmap -> ../init.d/portmap*
    lrwxrwxrwx  K92apmd -> ../init.d/apmd*
    lrwxrwxrwx  K96pcmcia -> ../init.d/pcmcia*
    lrwxrwxrwx  S10network -> ../init.d/network*
    lrwxrwxrwx  S30syslog -> ../init.d/syslog*
    lrwxrwxrwx  S50inet -> ../init.d/inet*
    lrwxrwxrwx  S55named -> ../init.d/named*
    lrwxrwxrwx  S80sendmail -> ../init.d/sendmail*
    lrwxrwxrwx  S90mysql -> ../init.d/mysql*
    lrwxrwxrwx  S91smb -> /etc/rc.d/init.d/smb*
    lrwxrwxrwx  S99local -> ../rc.local*
```

보시는 바와 같이 S 로 시작하는 파일들과, K 로 시작하는 파일들이 있는데, 대부분의 파일이 /etc/rc.d/init.d 아래의 파일로 링크가 되어 있는 것을 확인할 수 있습니다.
이처럼, rcN.d 디렉토리의 모든 파일들은 오로지 링크로만 되어 있으며, 실제 서비스를 시작하거나 종료하는 스크립트는 /etc/rc.d/init.d 디렉토리 아래에 존재합니다.
S 로 시작하는 파일은 해당 서비스를 실행시키는데 사용되고, (Start)
K 로 시작하는 파일은 해당 서비스를 죽이는데 사용됩니다.(Kill)
레드헷 기반의 배포판에서는 데몬 관리의 편의를 돕기 위해 ntsysv 툴을 제공하고 있으므로, 데몬을 링크하거나, 삭제하는 수고를 덜 수 있다.

② rc.local 스크립트

사용자가 부팅시 마다 실행시키고자 하는 명령이 있을 때는 rc.local 파일을 사용하는데, 기본적으로 레드햇 계열의 리눅스에서는 로그인시 출력되는 메시지를 매번 갱신하도록 하는 내용으로 구성되어 있습니다. 만일 사용자가 로그인온시 출력되는 메시지를 바뀌기 위해서 /etc/issue 혹은 /etc/issue.net 의 내용을 바꾼다 하더라도 rc.local 스크립트에 의해 부팅시 마다 매번 초기화 되므로 출력메세지를 바꾸고자 한다면 rc.local 스크립를 손봐야 합니다.

(3) 시스템 고유 설정 디렉토리 (/etc/sysconfig)

여러분이 X 윈도우 제어판이나 각종 제어 프로그램을 사용하여 설정한 시스템의 기본 설정들은 대부분 /etc/sysconfig 디렉토리 아래 파일이나 하부 디렉토리 밑에 일관되게 놓입니다. 즉, 제어 프로그램을 사용하지 않더라도 이 디렉토리 아래에 놓인 설정 파일을 편집하여 시스템의 설정을 변경할 수도 있습니다.

Network : 리눅스 기본 네트웍 설정 파일
Network-scripts : 네트웍 인터페이스 관련 스크립트 모음 디렉토리
Mouse : 마우스 타임, 3버튼 에뮬레이션 여부 기록
Keyboard : 자판 타입 설정
Clock : BIOS 시간 저장 유형 설정
Pcmcia : PCMCIA 장치 설정
Ipchains : 인스톨시 설정한 방화벽 설정
Desktop : 기본 데스크탑 환경 설정
I18n : 시스템에 사용될 언어 설정
Harddisks : 하드디스크 파라미터 옵션 설정
.
.
.

(4) 파티션 마운트를 위한 fstab 파일

/etc/fstab은 파일 시스템 테이블이다. 이 안에는 시스템 부팅과 함께 자동으로 마운트되어야 할 항목과 옵션이 들어 있습니다. 자동 마운트 기능 외에도 마운트를 편하게 하기 위해 /etc/fstab이 존재합니다.

```
/dev/hda1     /boot      ext2       defaults                  1 1
/dev/hda2     /          ext2       defaults                  1 1
/dev/hda3     /home      reiserfs   defaults                  1 1
none          /proc      proc       defaults                  0 0
none          /dev/pts   devpts     gid=5, mode=620           0 0
/dev/fd0      /mnt/floppy auto      noauto, owner             0 0
/dev/cdrom    /mnt/cdrom iso9660    noauto, owner, kudzu, ro  0 0
```

Fstab은 다음과 같이 6개의 필드로 구성되어 있습니다.
〈장치명〉〈디렉토리〉〈파일 시스템〉〈옵션〉〈덤프〉〈파일시스템 점검 순서〉
와우리눅스 7.0 이후 버전부터는 장치명에 파티션 포멧시 라벨명을 부여하고, 장치명 대신에 라벨명을 부여할 수도 있습니다.

```
LABEL=/boot                /boot          ext2          defaults          1 1
```

일반적인 경우를 한번 살펴보도록 합시다.

```
/dev/hda1                  /boot          ext2          defaults          1 1
```

/dev/hda1 파티션을 부트(/boot) 디렉토리에 마운트하고, 파일 시스템은 ext2 임을 나타냅니다. 마운트할 때의 옵션은 defaults 이며, 이 defaults는 rw, suid, dev, exec, auto, nouser, async 옵션을 합한 것과 같은 내용입니다. 다섯 번째 필드인 〈덤프〉값은 dump 라는 프로그램에 의해 덤프할 파일 시스템인지 여부를 적는 것인데, ext2 파티션인 경우 대부분 1로 설정하고, 나머지는 0으로 설정합니다. 〈파일 시스템 점검 순서〉 부분은 ext2 파일 시스템의 하드 디스크 파티션에서만 사용합니다. 이 곳이 1 이상의 값으로 설정되어 있는 파티션은 부팅 과정에서 파일 시스템의 이상 여부 확인 후, 문제가 있다고 판단되면 자동으로 점검하게 됩니다.
시디롬과 플로피 디스크의 옵션부의 noauto 옵션은 시스템 부팅중 자동으로 마운트되지 않습니다. 시디롬과 플로피 디스크는 쓰고 싶을 때만 넣고 마운트하는 매체이기 때문에 부팅중 마운트 하려 들면 귀찮은 메시지만 나올 뿐입니다.

```
Mount: wrong fs type, bad option, bad superblock on /dev/cdrom,
       Or too many mounted file systems
```

그러면 시스템 부팅중 자동으로 마운트하지도 않을 항목을 뭐하러 /etc/fstab에 넣는 것일까? 그것은 마운트 명령이 편해지기 때문입니다.
위의 예에서 /dev/cdrom 장치가 시디롬이고, /mnt/cdrom에 마운트한다는 사실을 기록해 두었기 때문에 시디롬을 마운트하기 위해서는 다음과 같이 하기만 해도 마운트를 할 수가 있습니다.

```
# Mount /mnt/cdrom
```