제2장 리눅스 시작하기

1. 시스템의 시작과 종료
2. 사용자 관리
3. 리눅스 매뉴얼 보기
4. 파일 및 디렉토리와 관련된 명령어들
5. 텍스트 편집기
6. 리눅스 쉘
7. 텍스트 프로세싱
8. 압축 파일의 이용
9. 데이타 백업
10. 주기적으로 프로그램 실행하기

1. 시스템의 시작과 종료

(1) 리눅스 시스템 처음 시작하기

리눅스 설치를 마치고, CD를 꺼내고 나면 시스템이 다시 시작됩니다.

LILO boot :

리눅스 설치 중 Default로 선택된 값을 변경하지 않고 기본적인 설치를 하였다면, 위와 비슷한 프롬프트가 나타납니다.
만일, MS windows가 설치된 상태에서 리눅스를 설치하였다면, Tab 키를 누르면 부팅 가능한 OS의 Label들(linux, windows98 등)이 나타날 것입니다. 그 중 리눅스 설치 시 리눅스를 위한 Label로 선택했던 이름을 입력하면 됩니다(Default는 "linux"입니다).
최신 버전의 리눅스를 설치하였다면 그래픽 환경의 LILO가 뜨게 됩니다. 그러면, 방향키로 Linux Label에 해당하는 것을 선택한 후 엔터를 누릅니다.
하지만, 선택하지 않아도 LILO는 기본적으로 Linux를 Loading하게 됩니다.
LILO(LInux LOader)는 주로 MBR(Master Boot Record)에 설치되어 여러 개의 커널로 부팅이 가능한 경우, 커널들이 어디에 위치하고 있는지에 관한 정보들을 파악하고 있다가 사용자로 하여금 사용할 운영체제를 선택하도록 하고 선택된 운영체제에 맞는 커널을 읽어 들이는 역할을 합니다.
화면에 메시지를 가득 채우면서 약간의 시간이 경과한 후, 와우리눅스 7.1 파란의 경우 다음과 같은 화면이 나타납니다.

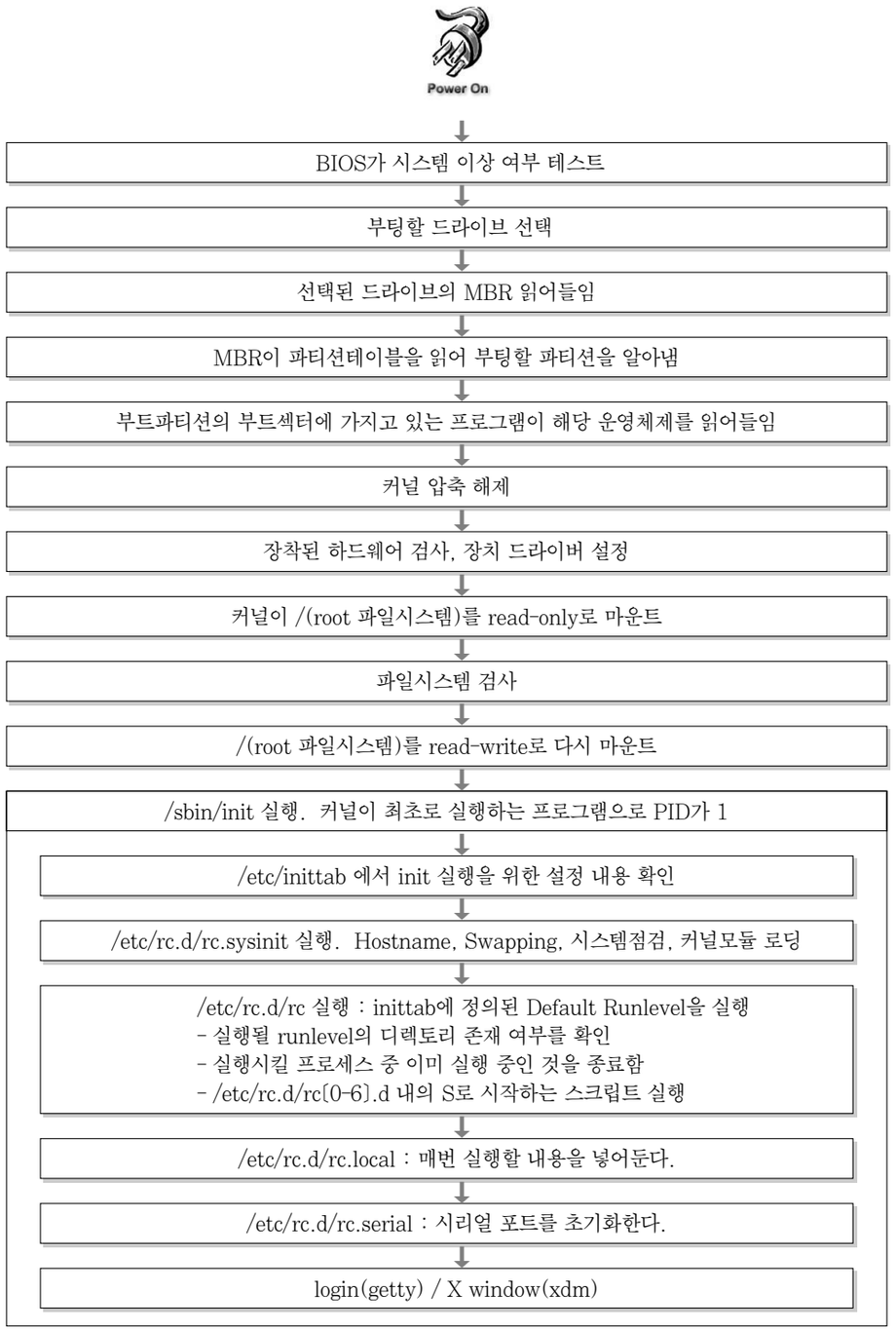
WOWLiNUX Release 7.1 (Paran)
Kernel 2.4.2-2wl on an i686

login:

login : 에 "root"라 입력하고, passwd : 에는 인스톨 할 때 입력했던 패스워드를 입력합니다.
그러면 명령어를 입력하면 반응을 보이게 되는 준비된 쉘 프롬프트 상태가 되었을 것입니다. 잘 되는지 "ls" 명령을 한 번 입력해 보십시오. 파일 이름 같은 리스트가 화면에 출력된다면 성공입니다.

(2) 시스템이 시작되는 과정

로그인 프롬프트를 보여주기 전까지 어떤 작업들이 수행되는지에 관한 절차에 대해 알아 봅니다.
이러한 절차를 알고 있다면 시스템이 시작될 때 어떠한 작업을 추가하거나 삭제하는 것이 가능해 집니다.
다음은 일반적인 운영체제의 로딩과정을 설명한 것입니다.



[그림 Ⅱ-1] 시스템 시작 순서

/etc/rc.d/rc[0-6].d에서 [0-6]은 실행할 실행레벨로 다음과 같은 종류가 있습니다.

0	시스템 종료 halt
1	단일 사용자가 사용하는데 적합하도록, 시스템이 최소화된 상태로 부팅하기 원할 경우 사용
2	다중 사용자 모드. 단 NFS를 지원하지 않음.
3	다중 사용자 모드로 완전한 Networking을 가능하게 합니다.
4	사용되지 않음. 사용자 정의 레벨로 사용될 수 있습니다.
5	X Window 로 부팅 되도록 할 때 사용
6	Reboot

실행레벨은 시스템의 사용 용도나 형편에 따라, 수행되어야 할 서비스들을 묶어 놓은 그룹 정도로 해석할 수 있습니다. 관리자만이 시스템에 접근해야 할 필요가 있을 경우에는 실행레벨 1로, X window 로 로그인 하려면 실행레벨 5로 정해주면, 알아서 그룹으로 묶여 있는 서비스들이 시작되거나 종료되는 것입니다.
해당 디렉토리를 검색해 보면 알 수 있겠지만, 각 실행레벨(runlevel)은 각기 실행할 스크립트들을 각각의 디렉토리에 링크파일로 가지고 있어, init에 의해 해당 스크립트를 실행시키도록 합니다.
앞서 말했듯이 사용자의 편의대로 실행레벨을 별경할 수 있으며, 아래 파일을 수정함으로써 가능합니다.
3 부분을 원하는 실행레벨로 바꾸면 됩니다.

```
#vi /etc/inittab
…
id:3:initdefault:
…
```

당연한 이야기겠지만 위의 실행레벨로 미루어 볼 때, 3 혹은 5가 올 것입니다.
그외, 사용자가 정의 한 특정 실행레벨로 실행 시키고 싶을 때는 실행레벨 4가 올 수도 있을 것입니다.
그러나, 0 혹은 6같은 실행레벨을 준다면 부팅 즉시 종료 혹은 리부팅을 시키는 것이 됩니다.

(3) 부팅 시 실행 할 프로그램 추가하기

위의 절차 중에서 /etc/rc.d/rc 스크립트의 실행내용을 보면 /etc/inittab 파일 중 'id' 부분에 정의된 실행레벨에서 실행될 스크립트들을 실행시키는 일을 합니다.
실행시킬 스크립트들은 /etc/rc.d/rc[0-6].d 디렉토리에 들어 있는데, 이 파일들은 알고 보면, /etc/rc.d/init.d 디렉토리에 실제적인 실행 스크립트들이 들어 있습니다. 그래서, 추가적으로 실행될 스크립트를 추가하려면, /etc/rc.d/init.d 에 스크립트를 추가하고, 실행되기 원하는 레벨의 디렉토리에서 링크를 만들어 주는 것입니다.
위 보기의 경우, 실행 레벨이 '3' 이고, /etc/rc.d/rc3.d 디렉토리에 실행될 스크립트를 가리키는 링크들이 들어 있습니다.

그 링크 이름의 구성을 보면, 앞쪽의 Character가 두가지인 것을 알 수 있습니다.
'K' Stop
'S' Start
를 의미하는 것으로, 'K15httpd' 는
/etc/rc.d/init.d/httpd stop
으로 실행되고, 'S85httpd' 는
/etc/rc.d/init.d/httpd start
로 실행되는 것입니다.
뒤의 숫자는 실행 순서를 나타내는 것으로 번호가 앞서는 것이 먼저 실행됩니다.
이렇게 실행레벨 별로 따로 실행시킬 필요가 없는 경우에는
/etc/rc.d/rc.local
에 실행시킬 스크립드의 내용을 넣어 두면, 해당 실행레벨의 스크립트들이 실행되고, 마지막으로 항상 스크립트가 실행됩니다.

(4) 시작 프로세스 관리

부팅시 자동으로 실행되어야 할 서비스들은 사용자의 필요에 따라 실행 여부를 조정할 수 있습니다. 리눅스에서는 다음과 같은 방법들을 사용하여, 쉽게 서비스를 조정할 수 있습니다. 이것은 위에서 살펴 본 디렉토리의 내용들을 보다 쉽게 관리할 수 있도록 한 것입니다.

① linuxconf

linuxconf는 서비스 설정 뿐만이 아니라 시스템에 관한 대부분의 것들을 모두 설정할 수 있습니다.
그 중, 제어 → 제어판 → 서비스 활동 제어 메뉴를 선택하여,
아래와 같은 메뉴에서 설정을 변경할 수 있습니다.

```
# linuxconf
```

② ntsysv

X window를 사용하지 않는 사용자들이 주로 사용하는 툴로, 아래와 같이 '*' 선택된 부분이 부팅시 실행되는 서비스입니다. 기본적으로는 현재 실행 중인 레벨의 서비스를 조정하지만, 실행시 다른 레벨을 지정해 주면, 다른 레벨의 실행 서비스도 조정할 수 있습니다.

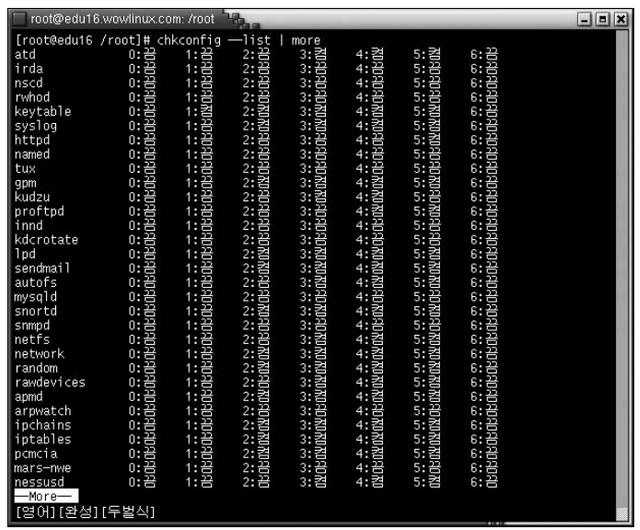
```
# ntsysv
```

[그림 II-2] ntsysv의 실행 화면

3 chkconfig

아래와 같이 모든 실행 레벨의 상태를 텍스트로 보여줍니다. 명령행에서 옵션을 사용해 추가하거나 삭제할 수 있습니다.

 # chkconfig --list¦more



[그림 II-3] chkconfig의 실행 화면

(5) 부팅 과정의 Log 확인하기

시스템에 따라 속도 차이는 있겠지만, 부팅 되는 과정이 모니터에 표시되고 나면, 그 내용을 보고 싶어도 제대로 볼 수가 없습니다.
다음과 같이, 부팅에 관한 Log를 확인할 수 있습니다.

 # dmesg

혹은

 # cat /var/log/messages

혹은 Shift + Pg Up / Pg Dn 버튼을 이용하여 볼 수도 있습니다.

(6) 시스템 종료하기

몇가지 시스템 종료 방법을 알아 보겠습니다.

1 X window 모드로 부팅한 경우

runlevel 5로 부팅한 경우 X window로 Start하게 됩니다.
이런 경우, 좌측의 발바닥(GNOME) 혹은 KDE로고(KDE) → 로그아웃 → 시스템 종료를 선택합니다.

[그림 II-4] GNOME의 로그아웃

[그림 II-5] KDE의 로그아웃

② init

위에서 언급했던 'init'를 이용하여 시스템을 종료하거나, 리부팅하는 것이 가능합니다.
 # init 0 (시스템 종료)
 # init 6 (시스템 리부팅)
그러나, 이 방법은 여러명의 사용자가 시스템에 연결되어 있는 경우 좋지 않은 방법입니다.

③ shutdown

사용자들에게 미리 대처할 수 있도록 시간여유와 시스템 종료에 관한 메시지를 내보내는 'shutdown' 명령을 사용하는 것이 좋습니다.
 # shutdown -h now : 지금 당장 시스템 종료
 # shutdown -r +t5 : 5초 후 리부팅
 # shutdown +10 : 10분 후 셧다운
 # shutdown -c : shutdown 예약 취소

④ halt

halt명령은 아직 정해지지 않은 디스크의 운영 상태를 완성시키고, /var/log/wtmp에 login 과 logout 에 관한 내용을 저장하고, 커널에게 reboot 할 것인지, poweroff 할 것인지를 알려 시스템을 안전하게 종료하도록 합니다.
shutdown -h now 와 같은 명령입니다.
 # halt

⑤ Ctrl + Alt + Del 키로 시스템 종료 및 리부팅

/etc/inittab 파일안의
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
항목에 위와 같이 설정하면, Ctrl + Alt + Del로 리부팅 시킬 수 있습니다.
init 명령으로 실행레벨을 바꾸거나, 종료할 경우 다중 사용자 모드라면 현재 사용중인 사용자들이 아무런 대책없이 종료 당하게 되므로, shutdown 명령을 사용하여 사용자들로 하여금 미리 대비할 여유를 주는 것이 좋습니다.
다음 파일 안에 shutdown 시킬 권한이 있는 사용자들을 정의할 수도 있습니다.
shutdown 명령을 -a 옵션과 함께 사용하며, /etc/inittab 파일 안에 정의해 두어야 합니다.
ca::ctrlaltdel:/sbin/shutdown -a -t3 -r now
-a 옵션에 의해서, 어떤 사용자가 콘솔로 로그인 하여 Ctrl + Alt + Del 키를 입력하면, 다음 파일을 검사합니다.
/etc/shutdown.allow
이 파일에는 셧다운이 가능한 사용자들의 id가 들어 있어 이중 하나의 사용자 혹은 root사용자가 아니라면, 다음과 같은 메시지와 함께 shutdown은 더 이상 진행되지 않습니다.
shutdown: no authorized users logged in
이렇게 함으로써 어떤 사용자가 시스템에 물리적으로 접근하여 Ctrl + Alt + Del 키를 눌러 시스템을 실수로 혹은 고의적으로 종료시키는 것을 방지할 수가 있습니다.

2. 사용자 관리

사용자와 그룹을 관리하는 것은 시스템 관리자로서 중요한 역할입니다. 사용자 계정과 그룹의 생성, 관리, 삭제에 대한 내용을 알아 봅니다.

(1) 사용자

리눅스 설치를 할 때 사용자를 생성하는 부분이 있습니다. root 계정을 비롯하여 일반 사용자를 추가할 수 있습니다. root 사용자는 이유를 불문하고 리눅스 시스템의 모든 파일을 접근할 수 있으며, 작업 제어가 가능합니다.
일반 사용자는 시스템에 로그인이 가능하며 응용프로그램들을 실행시킬 수 있습니다. 리눅스는 멀티 유저 시스템이므로 많은 사용자들이 동시에 시스템에 로그인을 할 수 있습니다. 사용자들은 각각 자신들의 작업환경인 홈디렉토리와 환경설정파일을 소유합니다.

(2) 사용자 추가하기

사용자를 추가하는 방법에는 useradd(adduser)라는 명령어를 사용하는 방법, 윈도우 매니저와 데스크탑 환경에서 제공하는 그래픽 관리 툴을 이용하는 법, 직접 사용자와 패스워드를 관리하는 파일을 편집하여 추가할 수가 있습니다. 사용자를 추가할 때는 다음의 정보들이 자동으로 생성이 되며 특정한 내용을 설정하고 싶다면 사용자 추가 시 옵션을 사용한다거나 파일에서 수정을 합니다.
로그인명, 사용자 전체명(기타 설명), 사용자아이디(UID), 그룹아이디(GID), 패스워드, 홈디렉토리 위치, 로그인 쉘

(3) 명령어로 사용자 계정 설정하기

명령어를 통해 사용자를 추가한다면 다음의 파일들에 대한 내용을 알고 있어야 합니다.

/etc/passwd	사용자 계정에 대한 정보를 보관합니다.
/etc/shadow	shadow passwd 를 사용할 경우 사용자의 패스워드는 이곳에 암호화되어 관리가 됩니다.
/etc/skel	새로운 사용자의 홈디렉토리에 복사가 되는 환경설정 파일의 템플릿들이 위치한 디렉토리입니다.
/etc/group	그룹 설정과 관련된 정보를 저장하는 곳입니다.
/etc/login.defs	패스워드의 자릿수, 유효기간 등의 정보를 설정하고 있습니다.
/etc/default/useradd	사용자의 홈디렉토리, 기본 쉘 환경 등을 설정합니다.

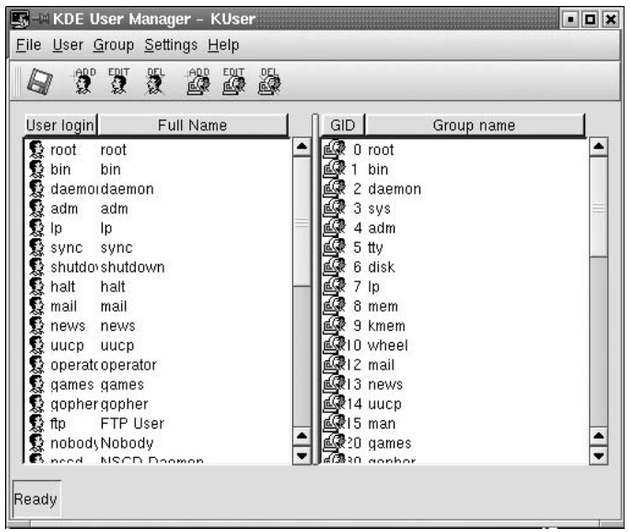
사용자 추가 명령은 root 권한으로만 실행할 수 있으며 위 정의된 파일의 설정을 바탕으로하여 사용자를 생성합니다.
가령, paran 이라는 아이디의 사용자를 추가하고 paran 사용자의 패스워드를 설정하기 위해서는 다음과 같이 명령을 내리며, 사용자명과 패스워드가 정상적으로 입력이 되면
성공적으로 설정이 되었다는 메시지가 출력이 됩니다.

```
# useradd paran

# passwd paran
Changing password for user paran
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully
#
```

(4) 그래픽 사용자 관리도구

GNOME과 KDE 등의 데스크탑환경은 관리자가 편리하게 사용자들을 관리할 수 있는 GUI기반의 계정 관리 도구를 제공합니다. 이러한 도구를 이용하면 편리하게 사용자를 추가하고 설정 내용을 변경할 수 있습니다.



[그림 II-6] kuser의 실행 화면

(5) 사용자 계정 관리

보안 혹은 다른 이유 등으로 일시적으로 사용자가 로그인을 하지 못하도록 막을 수 있습니다. /etc/passwd 파일을 열어 사용자 목록의 제일 앞 줄에 "*"를 추가하거나 사용자 기본 쉘 부분을 /bin/bash에서 /bin/false 로 변경을 하면 됩니다.

(6) 사용자 계정 삭제

이미 만들어진 사용자들을 삭제하는 것은 좋은 방법이 아닙니다. 백업받아 놓은 파일이 어떤 사용자가 사용하던 파일인지 구분하는 방법은 사용자의 UID로 가능하기 때문입니다. 하지만 사용자를 삭제할 경우가 생긴다면 다음의 방법들을 이용합니다.

userdel 사용자명	사용자 계정을 삭제합니다. **# userdel -r paran** : paran이라는 사용자의 사용자 정보와 홈디렉토리 파란 소유의 파일들을 모두 삭제합니다.
usermod -e 사용자명	사용자 계정의 유효기간을 설정합니다. **# usermod -e 2001-06-30 paran** paran 이라는 사용자는 2001년 6월 30일 이후 로그인이 불가능합니다.

(7) 그룹관리

그룹을 만드는 것은 각각의 사용자들을 좀 더 효율적으로 관리를 하기 위함입니다. 그룹에 속한 사용자들은 그룹권한이 부여된 파일과 디렉토리에 대해서는 동일한 권한을 가지고 접근을 할 수 있습니다.
/etc/group 파일은 그룹설정과 관련한 정보를 보관하는 곳입니다. 그룹을 추가 하거나 삭제를 위해서는 다음의 명령을 사용합니다.

 # groupadd 그룹명 : 그룹을 생성합니다.
 # groupdel 그룹명 : 그룹을 삭제합니다.

3. 리눅스 매뉴얼 보기

(1) man 명령어

$ man [section] 명령어이름

MS windows 운영체제에 도움말 기능이 있듯 리눅스에도 맨페이지라는 도움말이 있습니다.
/usr/share/man 디렉토리 섹션별로 저장이 되어 있으며, 이 온라인 도움말 시스템은 각 명령과 용어에 대해 다음과 같은 내용을 포함합니다.
- 명령어 이름(NAME)
- 개요(SYNOPSIS)와 설명(DESCRIPTION)
- 모든 옵션의 목록과 정의
- 환경(ENVIRONMENT)과 매개변수(PARAMETER) 등

관련되는 명령어들과 파일들 또한 맨페이지의 마지막에 표시가 됩니다.
섹션이라는 것은 맨페이지의 처음이나 마지막 관련 명령의 fsck(5)와 같이 숫자로 나타나는 부분입니다.
표는 섹션에 대한 설명입니다.

섹션	설 명
1	User commands ls, grep, find 등과 같은 사용자 명령(쉘명령)이 있습니다.
2	System calls 리눅스 프로그래밍을 위한 시스템 호출과 관계된 내용
3	C library functions 리눅스 라이브러리 함수와 관련된 내용
4	Description of configuration files 특수파일(FIFO, 소켓 등)에 대한 문서
5	File formats (syntax) 중요한 설정 파일에 대한 정보를 포함합니다.
6	Game descriptions 시스템 테스팅 프로그램에 대한 설명
7	Cover text, text format, etc. 표준과 규칙에 대한 정보, 프로토콜, 문자세트, 시그널목록에 대한 정보
8	System administration mount, fsck 등 시스템 관리자가 사용하는 명령에 대한 설명
9	Linux kernel routines 커널 프로그래밍을 위한 정보
n	"New" or commands that didn't fit elsewhere 새로운 명령에 대비한 공간, 주로 Tcl/Tk 프로그래밍에 대한 내용

어떤 주제와 관련하여 하나 이상의 섹션에서 그 내용을 소개하기도 합니다. 가령, crontab 은 1번과 5번 섹션에서 소개가 됩니다.
명령어에 대한 맨페이지를 보고 싶다면 man 1 crontab, 파일의 포맷에 대해 알고 싶다면, man 5 crontab 명령을 입력하면 됩니다.

(2) 관련된 명령어들

만약 명령어의 이름을 확실히 모른다면, 키워드를 이용하여 해당 키워드가 포함된 맨페이지를 찾아 낼 수 있습니다.

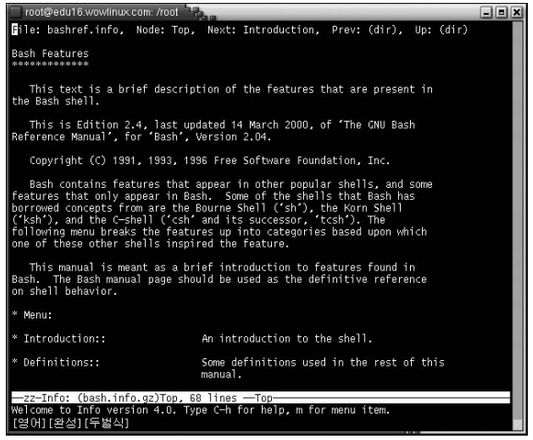
$ apropos manual

```
man             (1) - format and display the on-line manual pages
man [manpath](1) - format and display the on-line manual pages
man2html        (1) - format a manual page in html
perlxs          (1) - XS language reference manual
whereis         (1) - locate the binary, source, and manual page files for a command
xman            (1x) - Manual page display program for the X Window System
```

apropos 명령은 man -k 명령과 같습니다.

(3) info 명령어

info 는 GNU help Utility 이다. man page의 다음 단계로 info page가 기획되었지만, 아직 많이 사용되지는 않고 있습니다. 어떤 경우에는 info page에서 가장 최신의 정보를 구할 수 있고, 하이퍼텍스트 링크와 같은 강력한 기능을 제공합니다.



[그림 II-7] info bash의 실행 화면

4. 파일 및 디렉토리와 관련된 명령어들

리눅스 파일시스템이 어떻게 구성되어 있는지, 디렉토리간 이동, 파일의 생성 및 이동 그리고 삭제 등을 알아 보도록 합니다.

(1) 리눅스 파일시스템

설치하는 동안 리눅스 파티션에 대한 파일시스템이 만들어 집니다. 주로 사용되는 리눅스 파일시스템은 ext2 입니다. 전통적인 유닉스 파일시스템과 유사하며, FAT(MS-DOS)나 ISO-9660(CD-ROM) 파일시스템과는 차이가 있습니다. 파일시스템에 대한 자세한 내용은 후에 다루기로 합니다.
리눅스 및 유닉스와 유사한 운영체제들은 파일과 디렉토리를 계층적(hierarchical system)으로 구성합니다. 파일들은 디렉토리 안에 위치하며 이러한 디렉토리는 상위 디렉토리의 하위 디렉토리로 이루어지게되는 트리(tree)구조입니다. 트리의 시작은 "/(root)" 디렉토리이며, 여기서부터 가지가 뻗어 나가 하위 디렉토리, 하위의 하위 디렉토리 등으로 만들어 지게 됩니다. 파일들은 나뭇가지의 잎과 같이 디렉토리 안에 위치합니다.
파일과 디렉토리 이름은 공백을 포함한 어떠한 문자를 사용해도 상관이 없으나 상위 디렉토리와 하위 디렉토리를 구분하는 역할을 하는 "/"는 제외됩니다. 파일의 속성을 나타내는 확장자(.xyz)는 필요가 없으며, "."로 시작하는 파일과 디렉토리는 숨김파일 혹은 숨김디렉토리입니다.
ls -a 명령을 사용하여 파일, 디렉토리 목록을 확인하면 "."과 ".."로 된 부분을 확인할 수가 있는데 "."은 현재 디렉토리를 ".."은 상위디렉토리를 의미합니다.

(2) ls 명령

ls 명령은 디렉토리 내 파일의 목록을 보기위해 사용합니다.

$ ls

```
count1*  for1*  if1*   kill_netscape*  nsmail/  posparams*
count2*  fubar* ipmail* mbox            path*    postinfo/
```

ls 명령 다음에 -l 플래그를 추가한다면 파일과 디렉토리의 자세한 내용을 확인할 수 있다.

$ ls -l

```
-rwxr--r--   1 root     root        72  3월  5  16:29  count1*
-rwxr--r--   1 root     root       118  3월  5  16:43  count2*
-rwxr--r--   1 root     root       102  3월  5  16:02  for1*
-rwxr--r--   1 root     root        43  3월  5  13:56  fubar*
-rwxr--r--   1 root     root       267  3월  5  21:26  if1*
-rwxr-xr-x   1 root     root        79  7월 29   2000  ipmail*
-rwxr--r--   1 root     root        51  2월 28  15:51  kill_netscape*
-rw-------   1 root     root    145736  8월 16   2000  mbox
drwx------   2 root     root      4096  2월 28  14:11  nsmail/
-rwxr--r--   1 root     root        24  3월  5  14:24  path*
-rwxr--r--   1 root     root       485  3월  5  15:06  posparams*
drwx------   4 root     root      4096  7월 28   2000  postinfo/
```

파일의 유형, 허가권, 링크카운트, 파일소유권, 바이트 크기, 마지막으로 수정한 날짜와 시간, 파일명 등을 보여 줍니다.

다음은 ls 명령과 함께 사용될 수 있는 옵션에 대한 설명입니다.

ls -a	모든 파일 보기
ls -l	자세히 보기
ls -c	시간순으로 정렬해서 보기
ls -d	디렉토리만 보기
ls -f	디스크에 저장된 순서로 보기(정렬 안하기)
ls -i	색인번호(아이노드)와 함께 보기
ls -k	kb단위로 보여주기
ls -m	파일이름을 가로로 나열하기
ls -r	내림차순 정렬로 보기
ls -t	시간순으로 정렬해서 보기
ls -R	하위디렉토리 안의 파일까지 보여주기
ls --color	파일의 형태에 따라 색깔을 다르게 보여주기
ls -F	파일과 디렉토리를 심볼로 구분하여 보여주기

점(.)으로 시작하는 파일과 디렉토리는 ls 명령만으로는 볼 수 없으며 -a 옵션을 함께 사용하여야 합니다.

$ ls -al

```
합계 340
drwxr-x---  14 root     root     4096  5월  8  11:20  ./
drwxr-xr-x  20 root     root     4096  8월 16   2000  ../
-rw-------   1 root     root     3253  8월 16   2000  .ICEauthority
-rw-------   1 root     root      116  8월 16   2000  .Xauthority
-rw-r--r--   1 root     root     1126  8월 24   1995  .Xresources
-rw-rw----   1 operator disk       31  6월  1   2000  .amandahosts
drwxr-x---   2 root     root     4096  5월 31   2000  .ami/
drwxr-xr-x   3 root     root     4096  3월  6  17:33  .kde/
drwxr-xr-x   2 root     root     4096  3월  4  15:43  .mc/
drwxr-xr-x   5 root     root     4096  8월 16   2000  .netscape/
drwxr-xr-x   3 roo      root     4096  8월 16   2000  .sawfish/
drwx------   2 root     root     4096  3월  6  15:45  .ssh/
-rw-------   1 root     root     1912  6월  8   2000  .xsession-errors
-rwxr--r--   1 root     root       72  3월  5  16:29  count1*
-rwxr--r--   1 root     root      118  3월  5  16:43  count2*
```

숨김파일은 주로 다른 사용자들이 함부로 수정을 못하게 하거나 거의 변경이 되지 않는 환경파일과 같은 파일에 설정을 합니다. 가령, 넷스케이프 웹브라우저를 사용하게 되면 사용자의 홈디렉토리에 .netscape 라는 디렉토리가 생성되고 사용자의 브라우저 설정과 관련한 파일들이 위치하게 됩니다.

(3) cd 명령

디렉토리 이동을 할 때 사용하는 명령입니다.

cd 디렉토리명	특정 디렉토리로 이동
cd	홈디렉토리로 이동
cd ~	홈디렉토리로 이동
cd ~/test	홈디렉토리의 하위 디렉토리인 test디렉토리로 이동
cd ..	상위 디렉토리로 이동
cd ../..	상위 디렉토리의 상위디렉토리로 이동
cd -	바로 직전 작업 디렉토리로 이동

(4) pwd 명령

현재 작업 중인 디렉토리를 볼 때 사용합니다(present work directory).

(5) 권한

파일과 디렉토리에 대한 접근권한은 사용자(user), 그룹(group), 기타(other) 별로 나뉘게 됩니다. 사용자는 적어도 하나의 그룹에 속하게 되며, 그룹에 속하지 않은 사용자들을 기타로 분류하게 됩니다.
권한은 읽기(read), 쓰기(write), 실행(execute)가 있습니다. 파일이나 디렉토리에 접근을 하기 위해서는 각 권한에 대한 권한이 설정되어 있어야 하며, 수퍼유저(root)는 권한에 상관없이 접근이 가능합니다.

$ ls -al

```
합계 340
drwxr-x---   14 root     root      4096  5월    8    11:20   ./
drwxr-xr-x   20 root     root      4096  8월   16     2000   ../
-rwxr---r--      1 root     root        72  3월    5    16:29   count1*
--More--
```

위 예제 count1 파일에 대한 접근권한을 살펴 보면, 사용자는 root이고 그룹은 root이며 사용자에 대한 권한은 rwx, 그룹에 대한 권한은 r, 기타에 대한 권한은 r로 설정이 되어 있습니다.

사용자	그룹	기타
rwx	rwx	rwx
421	421	421
4+2+1=7	4+2+1=7	4+2+1=7

(6) 링크

링크는 하나의 파일이 복수개의 이름을 갖도록 하는 것으로 실질적으로 파일은 시스템에서 inode 번호로 인식이 되기 때문에 가능한 것입니다.

$ ln [-snf] file target

-s	심볼릭 링크
-n	파일이 존재하면 겹쳐쓰기를 하지 않음
-f	파일이 존재해도 겹쳐씀

링크에는 하드링크와 심볼릭 링크가 있습니다.

① 하드링크(Hard links)

하드링크는 하나의 파일에 여러 개의 연결을 사용하는 데 사용된다. foo 라는 파일에 대해 bar 라는 하드링크 파일을 만들려면 다음과 같이 합니다.

$ ls -l

```
-rw-r--r--   1 wow      wowlinux     8262  5월   9 11:07 foo
```

$ ln foo bar

$ ls -l

```
-rw-r--r--   2 wow      wowlinux     8262  5월   9 11:07 bar
-rw-r--r--   2 wow      wowlinux     8262  5월   9 11:07 foo
```

$ ls -i

```
 559192 bar    559192 foo
```

foo 파일의 내용을 수정하면 bar 의 내용도 변경이 됩니다. 즉, foo 와 bar 는 항상 같은 내용을 가지는 것이며 다른 이름을 갖는 두개의 파일이고 하나의 inode 로 생성이 됩니다.
rm 명령을 이용하여 foo 파일을 지우게 되면 단지 foo 파일만 삭제되며 bar 파일은 남게 됩니다. 파일을 완전하게 지울려면 foo 파일의 모든 링크를 지워야 합니다. 같은 inode 번호를 가지기 때문에 다른 파일시스템 간에는 링크가 불가능합니다.

① 심볼릭링크(Symbolic links - symlinks)

MS windows 운영체제의 단축아이콘과 비슷한 역할, 링크로 생성된 파일의 내용은 존재하지 않으며, 파일이 어디를 가리키고 있는지 알려주는 역할을 합니다.
심볼릭 링크로 만들 경우 링크를 다른 곳으로 이동시키면 링크가 깨져서 사용할 수 없습니다. 하드링크의 경우에는 링크파일을 이동시켜도 링크상태를 유지합니다.
다른 파일시스템간 생성이 가능하여 다른 파티션에 링크파일을 만들 수 있습니다.

$ ls -l

```
-rw-r--r--   1 wow      wowlinux     8262  5월   9 11:07 foo
```

$ ln - s foo bar

$ ls -l

```
lrwxrwxrwx   1 wow      wowlinux        3  5월   9 13:18 bar -> foo
-rw-r--r--   1 wow      wowlinux     8262  5월   9 11:07 foo
```

$ ls -i

```
 559193 bar    559192 foo
```

foo 에 대한 심볼릭링크 bar 가 만들어진 것을 확인할 수 있습니다.

(7) cp 명령

파일을 대상 파일로 복사(copy)를 하거나 다른 디렉토리로 복사를 할 수 있습니다.

$ cp [option] source target

$ cp [option] source1 source2 souece3 ..... directory

cp -a	파일의 속성, 링크정보 들을 유지하면서 복사
cp -b	이미 파일이 존재할 경우 백업본을 만듦
cp -f	기존의 파일을 강제로 지우고 복사
cp -i	복사할 때 물어봄
cp -I	하드링크 형식으로 복사
cp -P	원본파일에 경로와 함께 지정했을 경우 그 경로를 그대로 복사
	cp -P a/b/c test하면 test/a/b/c 란 결과
cp -p	원본파일의 소유주, 그룹, 권한, 시간정보 복사
cp -r	경로와 함께 경로안의 파일들이 모두 복사

$ ls -l

-rw-r--r-- 1 wow wowlinux 8262 5월 9 11:07 foo

$ cp foo bar

$ ls -l

-rw-r--r-- 1 wow wowlinux 8262 5월 9 13:51 bar

-rw-r--r-- 1 wow wowlinux 8262 5월 9 11:07 foo

$ ls -i

559193 bar 559192 foo

(8) mv 명령

파일명을 변경하거나 현재 위치한 디렉토리에서 다른 디렉토리로 파일을 이동시키고자 할 때 사용되는 move 명령입니다.

$ mv [option] source target

$ mv [option] source1 source2 souece3 ..... directory

mv -b	백업파일을 만듭니다.
mv -f	강제로 복사
mv -i	사용자에게 확인을 시킵니다.
mv -u	업데이트를 합니다.
mv -v	과정을 자세히 보여 줍니다.

$ ls -l

-rw-r--r-- 1 wow wowlinux 8262 5월 9 11:07 foo

$ mv foo bar

$ ls -l

-rw-r--r-- 1 wow wowlinux 8262 5월 9 11:07 bar

(9) rm 명령

파일이나 디렉토리를 삭제(remove)할 때 사용합니다.

$ rm [option] filename

rm -i	사용자에게 확인을 시킵니다.
rm -f	강제로 삭제
rm -r	파일이면 지우고 디렉토리일 경우 경로와 파일을 함께 지움
rm -v	파일 지우는 정보를 자세히 보여줌

디렉토리는 삭제하는 명령에는 rmdir 명령도 있지만 rm -r 명령 다음에 삭제를 원하는 디렉토리를 지정하면 디렉토리와 함께 서브디렉토리, 디렉토리 안 파일들도 동시에 삭제가 가능합니다.
rm 명령으로 삭제한 파일은 복구가 불가능하기 때문에 항상 명령을 내리기 전 주의를 하여야 합니다.

(10) mkdir / rmdir 명령

디렉토리의 생성과 삭제를 합니다.
foo 라는 디렉토리를 만들면서 bar 라는 서브디렉토리를 함께 만들 경우

$ mkdir -p foo/bar

디렉토리를 만들면서 접근권한을 설정할 경우

$ mkdir -m 640 디렉토리명

디렉토리를 삭제할 경우

$ rmdir 디렉토리명

(11) find 명령

검색조건에 따라 파일을 찾아 화면에 표시하거나 쉘스크립트 등을 실행합니다.

$ find 디렉토리들 검색조건 작업명령

검색조건은 다음과 같습니다.

-name name	명명한 이름의 파일을 찾습니다. 와일드카드(*)등을 사용할 수 있습니다.
-user name	지정한 사용자가 소유한 파일을 검색합니다.
-type letter	f, d, l, c, b 등을 문자로 사용하며 f는 파일, d는 디렉토리, l은 심볼릭링크파일, c는 캐릭터파일, b는 블록디바이스 파일로 지정된 형식의 파일을 검색합니다.
-mtime [+/-]n	n일 이전(+), n일 이후(-)를 조건으로 하여 검색을 합니다.
-size [+/-]n[c/K]	n 사이즈의 파일을 검색. +면 n사이즈 이상이 되고, -면 n사이즈 이하가 됩니다. c는 char, K는 kilobytes이며, 기본값은 512 block size입니다.
-newer pathname	지정한 파일 이후에 만들어진 파일을 검색합니다.
-inum number	지정한 아이노드 번호의 파일을 찾습니다.

작업명령은 다음과 같다.

-print	검색한 파일을 stdout(디스플레이 화면)으로 출력합니다. 생략을 해도 무방
-exec command {} \;	검색한 파일에 대해 지정한 명령을 실행합니다.
-ok command {} \;	지정한 명령을 실행하기 전 확인하는 프롬프트가 나타납니다.

계층적인 디렉토리 구조로 인해 특정 파일이 어느 위치에 있는지 기억하기는 쉽지 않습니다. find 명령은 다양한 검색조건을 이용하여 디레토리 내에 위치한 파일들을 찾아 낼 수 있습니다. -name 옵션에서는 *, ?, []등과 같은 쉘에서 사용되는 와일드카드 문자를 사용할 수 있습니다.

소개된 옵션 외에도 매뉴얼 페이지를 보면 강력한 기능들이 있다는 것을 확인할 수 있습니다. find 명령을 큰 규모의 디렉토리에서 사용할 경우는 많은 시간이 걸리므로 백그라운도 작업을 수행하는 것이 편리합니다.

다음의 명령을 수행하고 결과를 확인해 보기 바랍니다.

```
# find . -print
# find /etc -type d -print
# find /home -name .bash_profile -exec more {} \;
# find /etc -type -f -mtime -7 -ok ls -l {} \;
# find / -name core -exec rm -rf {} \;
# find / -user foo -exec ls -al {} \;
# find / -name "*.jpg" -print >/tmp/jpglist 2>/dev/null &
```


(12) 파일해석

① file 명령

파일의 유형을 알아보는 명령입니다. 먼저 파일의 유형 및 파일의 내용을 살핀 다음 그 결과를 알려 줍니다. /usr/share/magic 파일에는 file 명령에 의해 취급되는 파일의 내용에 대해 유형을 결정할 수 있는 설정이 되어 있습니다.

$ file *

```
Desktop:  directory
bar:      ASCII text
gzip:     ELF 32-bit LSB executable, Intel 80386, version 1, dynamically linked (uses shared
          libs), stripped
info.c:   C program text
ld-2.2.so: ELF 32-bit LSB shared object, Intel 80386, version 1, not stripped
libio.h:  C program text
```

② od 명령

ASCII 텍스트가 아닌 파일의 경우는 od 명령을 통하여 8진수 혹은 다른 포맷으로 파일의 내용을 확인할 수 있습니다.

$ od [OPTION] file

od -d	10진수로 보여 줍니다.
od -o	8진수로 보여 줍니다.
od -x	16진수로 보여 줍니다.
od -c	ASCII 문자와 backslash escapes(\n 등)으로 보여 줍니다.

③ strings 명령

파일 내에서 ASCII 문자로 표현 가능한 부분을 표시합니다.

$ strings [OPTION] file

5. 텍스트 편집기

텍스트 편집기를 능숙하게 사용하는 것은 훌륭한 리눅스 시스템 관리자가 되기 위한 필요조건입니다. 설정 파일을 편집하고 스크립트를 작성하는 일 등이 이에 속합니다. 리눅스에서 사용되는 텍스트 편집기들은 강력한 기능을 가지고 있지만 익히기에는 다소 어려움이 있습니다.
리눅스 텍스트 편집기에는 다음과 같은 것들이 있습니다.

- vi
- emacs 와 xemacs
- jed
- joe
- jove
- pico
- ae
- 기타 편집기들과 GUI 편집기 프로그램들이 있습니다.

(1) vi 기본

vi는 visual interface의 약자로 유닉스의 라인 에디터인 ed와 ex의 발전된 형태입니다. vi 에서 사용되는 명령들은 less와 같은 리눅스 명령어들과 elm 혹은 mutt 등의 이메일 프로그램들과도 같습니다.
vi 보다 개선된 프로그램들을 이용할 수도 있습니다. nvi(new vi)나 vim(vi improved)등이 있으며 새로운 버전의 vi 편집기에 vi 명령의 링크를 통해 사용하면 됩니다.
vim은 vi에 unlimited undo, syntax coloring, split windows, visual selection, gui support 등의 기능이 보강된 vi clone 입니다.

[그림 Ⅱ-8] vim의 실행 화면

다음의 사이트에서 vi에 대한 자세한 내용을 살펴 볼 수 있습니다.
http://www.math.fu-berlin.de/ guckes/vi/
http://www.thomer.com/thomer/vi/vi.html
http://www.oreilly.com/catalog/vi6/
파일들은 다음의 명령을 통해서 생성되고 열리게 됩니다.
$ vi 파일명 혹은
$ vi /path/to/파일명
읽기전용으로 파일을 여는 명령은 다음과 같습니다.
$ view 파일명 혹은
$ vi -R 파일명
파일이 존재하지 않으면 위 명령은 vi 다음의 파일명으로 파일을 만들고 편집기가 열리게 되고, 이미 존재하는 파일이라면 편집기가 열리게 됩니다.
파일을 닫을려면 Esc 키를 눌러서 명령모드(command mode)로 들어간 후 다음의 명령을 수행합니다.

ZZ 또는 :wq	저장 후 종료
:w〔파일명〕	저장하기
:w!〔파일명〕	강제 저장(덮어쓰기, 보호모드 무시)
:q!	저장않고 종료
:e!	파일의 마직막 저장상태로 돌아감
:w 새파일명	다른 이름으로 저장
:e 다른파일명	vi 를 나오지 않고 새로운 파일을 편집

(2) vi 모드

vi는 크게 세 가지 모드로 나누어 집니다.

명령모드	키입력이 바로 명령이 되는 모드
입력모드	실제 문서를 편집하는 모드
ex모드	ex명령을 실행시키는 모드

vi를 실행시키면 처음에는 명령모드 상태가 됩니다. 명령모드에서 : 키를 누르면 ex모드로, a, i 등의 키를 누르면 입력모드로 전환되어 문서를 편집할 수 있는 상태로 변경이 됩니다.
그리고 Esc 키를 누르면 항상 명령모드로 전환이 됩니다.

(3) 이동명령

이동명령은 다른 명령(복사, 삭제, 붙이기)등과 조합해서 사용합니다.

① 짧은 이동

명령모드 상에서 h, j, k, l 키를 이용한 이동입니다. 다른 명령모드 명령과 마찬가지로 회수를 명령 앞에 넣어서 반복할 수 있습니다. 단어간의 이동은 w, b, e, E 키입니다.

h, j, k, l	좌, 하, 상, 우
+ 또는 Enter	다음 줄의 첫번째 문자로
-	이전 줄의 첫번째 문자로
e, E	단어의 끝으로
w, W	다음 단어로
b, B	이전 단어로
$	행의 끝으로
0	행의 처음으로
^	행의 처음으로
), (	다음, 이전 문장의 처음으로
}, {	다음, 이전 문단의 처음으로
]], [[	다음, 이전 구절의 처음으로

② 원거리 이동

특정 행으로 직접 이동하는 명령이 G 명령입니다.
10G라고 입력하면 10번째 행으로 이동하게 됩니다. 다른 방법으로는 ex모드에서 :10라고 입력하고 엔터를 입력하면 그 행으로 이동하게 됩니다.
맨 처음 줄로의 이동은 gg 나 1G 나 :1 등을 사용하고 마지막 줄로의 이동은 G 나 :$ 등이 가능합니다. 여기서 $는 ex모드에서는 마지막 줄을 나타냅니다.

^F (Ctrl+F)	한 화면 앞으로 스크롤
^B	한 화면 뒤로 스크롤
^D	반 화면 앞으로 스크롤
^U	반 화면 뒤로 스크롤
^E	한 줄 앞으로 스크롤
^Y	한 줄 뒤로 스크롤
H	화면의 맨 위줄로, nH인 경우 맨 위에서 n행 밑으로
M	화면의 중간 줄로
L	화면의 맨 아래줄로, nL인 경우 맨 밑에서 n행 위로

③ 찾기로 이동

/pattern	문자열의 처음으로 앞으로 검색
?pattern	문자열의 처음으로 뒤로 검색
n	검색을 다시 반복(같은 방향)
N	검색을 다시 반복(반대 방향)
nG 또는 :n	n번째 줄로 이동, G만이면 마지막 줄로
*	현재 커서가 위치한 단어 찾기(앞으로)
#	현재 커서가 위치한 단어 찾기(뒤로)

④ 마크를 이용한 이동

알파벳 갯수만큼의 마크를 하고 마크한 곳이나 그 줄로 바로 이동할 수 있습니다. 우선 마크를 하려면 ma mz를 마크 할 곳에서 입력합니다. 마크한 곳으로 이동은 두 가지가 가능합니다. 즉 마크한 줄과 열이 일치하는 이동은 a z 으로 마크한 줄의 처음으로 이동은 'a 'z를 이용합니다.

mx	현재 위치를 x 이름의 마크로 저장
`x	마크한 위치(행, 열)로 이동
'x	마크한 줄로 이동
``	이전에 마크한 위치로 이동
''	이전에 마크한 줄로 이동

(4) 편집명령

vi 로 파일을 연 후 기본적인 편집모드로 들어가기 위해서는 i 를 입력합니다.

i	현재 위치에서 삽입
I	현재 줄의 처음 위치에서 삽입
a	현재 위치에서 추가
A	현재 줄의 끝에서 추가
o	새로운 한 줄을 커서 아랫줄에 엽니다.
O	새로운 한 줄을 커서 윗줄에 엽니다.
S	줄을 지우고 삽입모드로
R	현재 위치에서 replace 모드로
J	다음줄과 현재줄을 합칩니다.
	대소문자 변환
.	마지막 명령을 반복
u	u 마지막 수정한 것을 취소
U	줄을 처음 상태로 복구

y키는 복사를 하는데 사용되는 키이고 d가 삭제에, p가 붙이기에 사용되는 키입니다. 명령모드에서 사용되는 명령들은 명령 앞에 숫자를 입력한 명령을 내리면 그 횟수만큼 명령을 반복하게 됩니다.
2yy 라고 입력하면 2줄을 복사합니다. 문자 위에서 x를 누르면 위치한 문자가 삭제되고 X를 누르면 Backspace키처럼 동작을 합니다.

Change(변경)	Delete(삭제)	Yank(복사)	설명
cw	dw	yw	한 단어
2cw 또는 c2w	2dw 또는 d2w	2yw 또는 y2w	두 단어
cc	dd	yy	한 행
c$ 또는 C	d$ 또는 D	y$ 또는 Y	커서 위치에서 행의 끝까지
c0	d0	y0	커서 위치에서 행의 처음까지
r	x 또는 X	y1 또는 yh	한 문자 변경

(5) 치환명령

텍스트 내 위치한 단어를 다른 단어로 변경할 때 사용하며 일반적인 치환문의 형식은 다음과 같습니다.
:〈범위〉s/old/new/〈옵션〉
10행부터 50행까지 예전의 내용을 새 내용으로 모두 변경하고 싶다면
:10,50s/old/new/g
전체 파일 내에서 예전의 내용을 새 내용으로 모두 변경하고 싶다면
:1,$s/old/new/g
또는
:%s/old/new/g
1행부터 15행까지 UNIX 라는 내용을 LINUX 로 변경하면서 각각에 대해 확인을 하고자 할 때는
:1,15s/UNIX/LINUX/gc

(6) set명령

ex모드에서 set명령을 사용하여 vi의 상태를 설정할 수 있습니다.
사용 가능한 옵션과 현재의 설정 상태를 확인하는 명령은 다음과 같습니다.

:set all
다음은 vi 옵션의 예들이며 이외에도 많은 옵션들이 있습니다.

showmode	현재 명령/상태모드를 명령어 라인에 보여 줍니다.
autoindent	삽입할 때 들여쓰기를 합니다.
number	행번호를 보여 줍니다.
nomagic	정규 표현식을 무시합니다.
tabstop	탭키의 이동간격 조절
scroll	스크롤 간격을 조절

탭키의 이동간격을 조절할 경우
:set tabstop=6
행번호를 보여주고 싶지 않을 때
:set nonumber

(7) .exrc 파일

/.exrc 파일에 vi 와 관련된 설정을 하면 vi 가 시작할 때 .exrc파일을 읽어들여 환경 설정을 하게 됩니다. 다음은 .exrc 파일의 예입니다.

set tabstop=4
set number
set autoindent

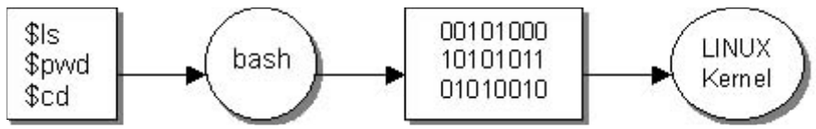
(8) 실행명령

vi 로 편집하는 도중 리눅스 명령어를 사용하여 화면에 표시하거나 파일내 삽입할 수 있습니다.
ex모드에서 다음의 예처럼 입력을 해 봅니다.

:!ls
:!date
명령 결과를 편집 중인 파일에 삽입하고자 한다면 다음과 같이 r 옵션을 사용합니다.
:r !date

6. 리눅스 쉘

영어 단어 Shell 은 조개 껍데기란 뜻입니다. 여기서 말하는 쉘은 그 내용물인 Operating System 을 둘러싸고 있는 껍질과 같습니다. 즉, OS와 사용자가 의사소통을 하기 위한 대화형 환경인 것입니다.
명령어를 입력하면 그 명령을 쉘이 해석하여 운영체제에게 전달합니다.
컴퓨터는 0과 1만을 이해할 수 있습니다. 하지만 사람이 컴퓨터가 이해할 수 있는 0과 1만을 사용하여 명령을 내리기는 너무나 어렵기 때문에 쉘이라는 것이 명령어를 입력받아 이것을 컴퓨터가 이해할 수 있는 0과 1로 된 명령으로 바꾸어 이 명령을 처리하는 커널에 전달합니다.



[그림 II-9] bash의 역할

사용자가 로그인 할 때 자동적으로 쉘이라는 프로그램이 실행됩니다.
최초의 쉘은 UNIX 연구 개발 초기 UNIX 파일 시스템 설계의 일부분으로 Ken Thompson에 의해 만들어 졌으며, AT&T 에서 Steven Bourne 에 의해 본쉘이 만들어 졌습니다.
그 이후 vi의 제작자인 Bill Joy에 의해 C 언어 구조와 유사하고 여러 기능이 추가된 C쉘이 만들어 졌고, AT&T 연구소의 David Korn은 본쉘과 C쉘을 조합한 Korn Shell 을 제작, 배포하였습니다.
쉘에는 여러 종류가 있으나 큰 갈래로는 Bourne Shell 과 C shell이 있습니다. 나머지들은 이들로부터 파생된 것들입니다.
이중 리눅스에서 기본적으로 사용하고 있는 쉘은 bash 쉘로 Bourne Again Shell이란 뜻이며,
이것 역시 Bourne Shell로부터 파생된 것입니다. 사용자가 원한다면 사용하고자 하는 쉘을 변경할 수도 있습니다.
쉘이 프롬프트에 사용하는 기호는 쉘마다 다른 데 기본적으로 본쉘과 콘쉘은 $, C쉘은 %을 사용하며 수퍼유저일 경우에는 모두 #을 사용합니다.
사용 가능한 쉘들을 비교해 보면 다음과 같습니다.

bash	sh	csh	ksh	tcsh
bourne again shell	bourne shell	C shell	korn shell	extended C shell
FSF	Steven Bourne	Bill Joy	David Korn	코넬대학 연구진
파일명 자동완성	속도와 간결성	C언어의 디자인,	본쉘의 구성파일	대화식 명령
강력한 스크립팅		앨리어스, 히스토리,	포함,	EMACS방식의
GPL		작업제어	C쉘의 특징 포함	명령라인 편집

/etc/shells 파일을 열어보면 사용할 수 있는 쉘들의 경로가 설정되어 있습니다.
/bin/bash2
/bin/bash
/bin/sh
/bin/ash
/bin/bsh
/bin/tcsh
/bin/csh
/bin/ksh
/bin/zsh
/etc/passwd 파일을 살펴보면 다음과 같이 사용자가 등록되어 있습니다.
test1:x:502:502:Test User:/home/test1:/bin/bash
마지막 항목(/bin/bash)이 사용자가 사용하는 쉘입니다.

(1) 사용자 쉘 변경

현재 자신이 사용하고 있는 쉘을 확인하고 변경할 수 있습니다.
$ echo $SHELL
/bin/bash
잠시 쉘을 변경하고 싶을 때는 사용하려는 쉘을 프롬프트에서 실행시킵니다.
그 쉘을 빠져나려면, exit 명령을 사용합니다.
기본 쉘을 변경하고 싶을 때는 chsh 명령을 사용하거나 /etc/passwd 파일의 쉘 부분을 변경합니다.
[wow@eduserver wow]$ csh
[wow@eduserver]$ exit
exit
[wow@eduserver wow]$
[wow@eduserver wow]$ chsh
wow으로 쉘 변경.
Password:
새로운 쉘 [/bin/bash]: /bin/csh
쉘이 변경되었습니다.
[wow@eduserver wow]$ cat /etc/passwd | grep wow
wow:x:502:503::/home/wow:/bin/csh

(2) 쉘 환경 설정

① 설정파일

리눅스에서는 여러 가지 환경 설정 파일을 제공합니다. 이것은 리눅스의 사용자를 더욱 자유롭고 융통성 있게 만들어 줍니다.
이런 파일들은 보통 홈디렉토리 안에 위치하고 있으며, .으로 시작하는 파일들입니다.
ls -a 명령으로 파일의 존재를 확인할 수 있습니다. 파일의 이름은 Resource Configuration 이라는 의미의 rc 라는 스펠링으로 끝나는 경우가 많습니다.
이런 파일들은 새로운 사용자를 등록하면 /etc/skel 디렉토리에 기본값으로 저장되어 있는 파일들을 홈디렉토리에 복사하여 생겨나는 것들이고 이 중에는 쉘 구동환경을 설정하는 파일들도 있습니다.

파일	설명
.bashrc	쉘을 위한 쉘스크립트로 서브 쉘, 즉 비로그인 쉘이 실행될 때 명령과 프로그램 구조로 구성할 수 있습니다. 새로운 쉘이 실행될 때마다 실행
.bash_profile	로그인할 때 읽어 들이는 설정 파일 주요 설정 내용은 검색경로, 터미널 종류, 환경변수 등을 설정하고, 그 외 로그인 시점에 실행시키고 싶은 명령, 시스템에 대한 정보를 보여주는 명령 등을 수행
.bash_logout	로그인 쉘이 종료되면서 읽어 들입니다.
/etc/profile	시스템 전역에 영향을 미치는 설정 파일
/etc/bashrc	시스템 전역에 영향을 미치고 새로운 쉘이 실행될 때 마다 실행

② 환경변수

환경변수는 쉘 환경을 입맛에 맞게 혹은 필요에 맞게 설정하는데 사용되는 값들을 가지고 있습니다. 물론 이 변수값을 수정함으로써 사용자마다 원하는 환경을 설정할 수 있으며, 쉘 프로그래밍을 할 때도 유용하게 사용됩니다.
다음은 주요 환경변수들에 대한 설명입니다.

변수	설명
DISPLAY	현재 X윈도우 디스플레이 위치
HOME	사용자 홈디렉토리
HOSTNAME	로그인 쉘이 종료되면서 읽어 들입니다.
PATH	쉘을 위한 쉘스크립트로 서브 쉘, 즉 비로그인 쉘이 실행될 때 명령과 프로그램 구조로 구성할 수 있습니다. 새로운 쉘이 실행될 때마다 실행
PS1	로그인할 때 읽어 들이는 설정 파일 주요 설정 내용은 검색경로, 터미널 종류, 환경변수 등을 설정하고, 그 외 로그인 시점에 실행시키고 싶은 명령, 시스템에 대한 정보를 보여주는 명령 등을 수행
PWD	로그인 쉘이 종료되면서 읽어 들입니다.
SHELL	쉘을 위한 쉘스크립트로 서브 쉘, 즉 비로그인 쉘이 실행될 때 명령과 프로그램 구조로 구성할 수 있습니다. 새로운 쉘이 실행될 때마다 실행
TERM	로그인할 때 읽어 들이는 설정 파일 주요 설정 내용은 검색경로, 터미널 종류, 환경변수 등을 설정하고, 그 외 로그인 시점에 실행시키고 싶은 명령, 시스템에 대한 정보를 보여주는 명령 등을 수행
MANPATH	로그인 쉘이 종료되면서 읽어 들입니다.
LS_COLORS	로그인 쉘이 종료되면서 읽어 들입니다.
EDITOR	주로 사용하는 편집기 설정

③ 환경변수값 확인

변수 하나하나에 대해 확인을 할 경우는
[wow@eduserver wow]$ echo $PATH
/bin:/usr/bin:/usr/local/bin:/usr/bin/X11:/usr/X11R6/bin:/usr/kerberos/bin:/home/wow/bin
혹은
[wow@eduserver wow]$ printenv PATH
/bin:/usr/bin:/usr/local/bin:/usr/bin/X11:/usr/X11R6/bin:/usr/kerberos/bin:/home/wow/bin
현재 설정되어 있는 변수값을 모두 확인한다면 다음의 명령을 사용합니다.
$ printenv

④ 환경변수값 설정/변경

다음과 같은 명령을 통해 현재 쉘 프롬프트 상에서 환경변수값을 설정하거나 변경할 수 있습니다.

$ export [환경변수명]=[변수값]

[wow@eduserver wow]$ echo $PATH

/bin:/usr/bin:/usr/local/bin:/usr/bin/X11:/usr/X11R6/bin:/usr/kerberos/bin:/home/wow/bin

[wow@eduserver wow]$ export PATH=/tmp:$PATH

[wow@eduserver wow]$ echo $PATH

/tmp:/bin:/usr/bin:/usr/local/bin:/usr/bin/X11:/usr/X11R6/bin:/usr/kerberos/bin:/home/wow/bin

외에도 $HOME/.bash_profile 파일을 수정하여 변수값 설정, 변경이 가능합니다. 이 파일을 수정하면 쉘 실행 시 마다 변경할 필요가 없습니다.

$ vi $HOME/.bash_profile

변수명=변수값

⑤ 쉘 활용하기

쉘 프롬프트 상에서 명령을 수행하는 데 알고 있으면 편리한 기능들이 있습니다.

■ 명령 완성 기능

명령어가 길거나 이름의 일부만 생각날 경우, 명령어의 일부만 입력하고 Tab 키를 입력하면 가능한 명령어를 완성시켜 줍니다. 만일, 해당하는 명령어가 둘 이상인 경우 해당하는 명령어의 리스트를 보여 줍니다.

■ 화살표 키로 사용했던 명령어 찾아내기

전에 사용했던 명령을 다시 사용하고 싶은 경우 아래, 위 화살표를 사용하여 사용했던 명령어를 아래, 위로 검색해 낼 수 있고 원하는 명령을 실행할 수 있습니다. 여기 나타나는 명령어들은 /.bash_history 파일에 저장되어 있습니다. 저장되는 명령어의 갯수는 HISTSIZE 변수로 결정됩니다.

■ 마우스로 텍스트 복사하기

X window를 사용하는 경우 여러개의 터미널을 띄우고, 여러개의 작업을 동시에 수행할 수 있습니다. 이 경우 이쪽 터미널에서 저쪽 터미널로 어떤 텍스트를 복사하고자 한다면, 마우스로 원하는 부분을 드래그 하고, 복사되기 원하는 위치로 가서 가운데 버튼, 2버튼인 경우 두 개 버튼을 동시에 누르면 복사가 됩니다. 물론, 에디터에서 에디터로의 복사도 가능합니다.

■ 여러개의 콘솔화면 사용하기

Remote로 떨어져 있는 터미널에서 telnet으로 접속한 경우가 아니고, 서버에 연결되어 있는 콘솔에서 여러개의 콘솔을 이동하며 사용할 수 있다. Alt + F1 F6를 눌러 이동하면서 사용할 수 있습니다.

■ 명령에 메타문자 사용하기

명령어에 파일을 인수로 사용할 경우 하나가 아닌 여러 개를 동시에 지정할 필요가 있을 때 파일을 하나씩 처리한다면 매우 불편할 것입니다. 이런 작업을 쉽게 하기 위해 와일들카드를 사용합니다.

? : 어떤 문자이던지 한문자

* : 어떤 문자도 없거나 그 이상인 경우

지우고자 하는 파일 이름이 test 로 시작하고 다섯글자인 파일이면,

$ rm test?

지우고자 하는 파일 이름이 test 로 시작하는 모든 파일이면,

$ rm test*

그 외에 사용되는 특수 문자들은 다음과 같습니다. 이러한 문자를 메타문자라 하고 이 문자들을 명령어 사용시 함께 사용하면 그 문자 나름대로의 기능을 수행하게 됩니다.

문자	의미
.〉	표준출력을 파일에 기록하는 출력 리다이렉션
〉〉	표준출력을 파일 끝에 덧붙이는 출력 리다이렉션
〈	파일로부터 표준입력을 읽는 입력 리다이렉션
*	0개 이상의 문자와 일치하는 파일 치환 대표문자
?	단일 문자와 일치하는 파일 치환 대표문자
[...]	대괄호 사이의 어떤 문자와도 일치하는 파일 치환 대표문자
|	어떤 프로세스의 출력을 다른 프로세스의 입력으로 보내는 파이프 기호
;	명령 순서에 사용
||	이전의 명령이 실패하면 실행하는 조건부 실행
&&	이전의 명령이 성공하면 실행하는 조건부 실행
&	명령어를 백그라운드로 실행
#	#문자에 뒤따르는 모든 문자들을 주석 처리
$	변수 접근

(3) 입출력 리다이렉션

리눅스에서 명령을 수행할 때 표준입력(stdin), 표준출력(stdout), 표준오류출력(stderr) 장치가 있습니다. 보통 표준입력은 키보드, 표준출력 및 표준오류출력은 모니터가 될 것입니다.

필요에 따라서 이 입출력 대상을 변경할 수 있습니다. 예를 들어 명령의 실행 결과를 화면에 뿌리지 않고, 파일로 저장하고 싶은 경우에 출력을 모니터가 아닌, 결과를 지정 할 파일을 지정하거나, 반복되는 입력이 필요한 경우, 입력해야할 데이터를 파일에 저장하여, 표준 입력을 키보드가 아닌, 이 파일로 저장하려는 경우 등에 사용됩니다.

각 입출력 스트림을 번호로 지정을 하면 stdin, stdout, stderr 순서대로 0, 1, 2 가 됩니다.

명령	시행
명령 〉 파일명	명령 실행 결과를 파일로 출력
명령 〉〉 파일명	명령 실행 결과를 이 파일에 덧붙여 출력
명령 〉& 파일명	명령 실행 결과와 에러를 파일로 출력
명령 〉〉& 파일명	명령 실행 결과와 에러를 이 파일에 덧붙여 출력
명령 〉! 파일명	명령 실행 결과를 이 파일이 존재하더라도 무시하고 출력
명령 〉&! 파일명	명령 실행 결과와 에러를 이 파일이 이미 존재하더라도 무시하고 출력
명령1 | 명령2	명령1의 output을 명령2의 input으로 사용하여 실행
명령 〈 파일명	파일의 내용을 데이터로 삼아 명령을 실행
명령 2〉 파일명	에러를 파일로 출력

찾은 결과를 부모 디렉토리에 find_result.txt 라는 이름의 파일로 저장하는 명령입니다.

$ find /home -name *.conf 〉 ../find_result.txt

ls -al 명령의 결과를 grep conf 명령의 입력값으로 사용하는 명령입니다.

$ ls -al | grep conf

다음은 메일의 내용을 미리 mailtext 라는 파일에 저장을 하고 이 내용을 다른 사람에게 메일을 보내는 명령입니다.

$ mail edu@wowlinux.com 〈 mailtext

find 명령을 실행 시 결과는 화면으로 출력을 하지만 에러는 파일로 출력을 할 때 다음과 같습니다.

$ find / -name core 2〉 errmesg

(4) Background 와 Foreground 작업

리눅스에서는 명령을 백그라운드로 수행하는 것이 가능합니다. 터미널 상의 명령 프롬프트 상에서 명령어를 수행할 때, 이것을 백그라운드로 처리하려면 수행하려는 명령 뒤에 &를 붙여 수행합니다.
이것은 명령 수행으로부터 발생하는 output 조차도 보이지 않는 것을 의미하는 것은 아니며 다만, 백그라운드로 실행되고 있는 명령과 상관없이 명령 프롬프트 상에서 다른 작업을 수행할 수 있는 것입니다.
그러므로, 화면으로 output을 나타내고 싶지 않다면 출력을 재지정하여 명령을 수행합니다.
jobs 명령은 현재 백그라운드로 수행되고 있는 작업이 어떤 것이 있는지 확인할 때 사용하는 명령입니다. 작업번호, 상태, 수행명령 순서로 화면에 나타납니다.

 # find / -name core 〉 corelist 2〉 errmesg &
〔1〕 22645
 # jobs
〔1〕+ Running find / -name core 〉corelist 2〉errmesg &
fg 명령은 백그라운드로 수행되고 있는 작업을 foreground로 실행시키려 할 때 사용하며 백그라운드로

수행 중인 작업이 여러 개인 경우, 작업번호 혹은 같은 명령이 수행되고 있는 경우가 아니라면 그 명령어를 적어 줍니다.

 # find -user wow 〉 wowlist 2〉 wowerr &
〔1〕 22665
 # find / -name core 〉 corelist 2〉 errmesg &
〔2〕 22666
 # jobs
〔1〕- Running find -user wow 〉wowlist 2〉wowerr &
〔2〕+ Running find / -name core 〉corelist 2〉errmesg &
 # fg 1
find -user wow 〉 wowlist 2〉 wowerr
Ctrl+z 는 포그라운드로 수행되고 있는 명령을 잠시 멈추는 명령으로 끝내는 것이 아니고, 잠시 멈추는 것입니다.
 # find / -name core 〉 corelist 2〉 errmesg
(Ctrl+z)
〔1〕+ Stopped find / -name core 〉corelist 2〉errmesg
 # jobs
〔1〕+ Stopped find / -name core 〉corelist 2〉errmesg
bg 명령은 현재 멈춰진 명령을 다시 수행시키되 백그라운드로 수행시키려 할 때 사용합니다. 수행해야 할 작업이 여러개라면 작업번호를 인수로 줍니다.
 # find / -name core 〉 corelist 2〉 errmesg
(Ctrl+z)
〔1〕+ Stopped find / -name core 〉corelist 2〉errmesg
 # jobs
〔1〕+ Stopped find / -name core 〉corelist 2〉errmesg
 # bg
〔1〕+ find / -name core 〉 corelist 2〉 errmesg &
 # jobs
〔1〕+ Running find / -name core 〉corelist 2〉errmesg &
Ctrl+c 는 foreground 로 수행중인 명령을 중지, 끝내고 싶은 경우 사용합니다.
작업이 바로 종료됩니다.

(5) Bash 스크립트

쉘에서는 명령이 입력되자마자 실행되는 인터프리터 환경을 제공합니다. 즉, 컴파일 같은 작업이 필요 없이 텍스트 파일 안에 명령 프롬프트에서 실행이 가능한 명령들을 적고, 이 파일에 실행권한을 주고 명령어 라인에서 명령어처럼 실행시키면 마치 프로그램처럼 실행이 됩니다. 이런 것을 Shell Script라고 합니다. 이 쉘스크립트에서는 여러 개 명령어들을 연속적으로 실행시킬 수 있으며, 제어문과 변수 선언 등이 가능한 프로그래밍 언어와 유사합니다.
쉘 스크립트를 사용하는 이유는 어떤 작업을 자동으로 수행하도록 하여 시스템 관리를 쉽고 효율적으로 만들기 위해서 라는 것이 가장 큰 이유일 것입니다.실제로 처음 리눅스를 설치했을 때도, 이미 시스템 관리를 쉽도록 하기 위한 많은 쉘 스크립트들이 존재합니다.
사실 명령을 실행시키는데 있어서는 쉘 종류에 따른 차이점을 크게 느낄 수 없지만, 쉘 프로그래밍을 하게 되면 그 차이점을 더 많이 느낄 수 있습니다. 사용하는 함수나 문법 등에 차이가 있습니다. 여기서는 리눅스의 기본 쉘인 bash를 기준으로 설명하며, 자세한 프로그래밍 방법을 설명하려는 것이 아니라 쉘 프로그래밍의 능력과 필요성을 아는 것이 목적입니다.
다음은 bash 스크립트의 예제로 vi 편집기를 통해 getline 파일을 생성합니다.

$ vi getline

```
#!/bin/bash
if [ $# -lt 2 ]
then
        echo Usage: $0 검색대상파일 검색할단어
        exit
fi
if [ ! -f $1 ]
then
        echo "$1 : 그런 파일은 존재하지 않습니다!"
else
    grep $2 $1 > $1_$2
fi
```

이 스크립트는 두 개의 파라미터를 넘겨 받아 첫번째 인수와 일치하는 이름의 파일에서 두번째 인수와 일치하는 라인을 찾아 "파일이름_찾은단어" 라는 이름의 파일에 저장하는 스크립트입니다.
쉘스크립트를 작성한 후에는 실행권한을 부여하여야 합니다.

$ chmod a+x getline

이제 작성한 스크립트를 이용하여 임의의 파일에서 원하는 단어를 찾아내 보도록 합니다.

$ ls

getline* passwd

$ ./getline passwd bash

$ ls

getline* passwd passwd_bash

passwd_bash 파일이 만들어 졌으며 이 파일의 내용을 보면 bash라는 단어를 포함하는 행들만으로 이루어져 있다는 것을 확인할 수 있습니다.
bash 쉘스크립트는 #!/bin/bash 와 같이 시작합니다(실제 bash 의 디렉토리를 적는다).
$# 는 스크립트에 넘겨진 인수의 갯수
$0 는 실행된 쉘스크립트 이름
$1 $nnn 는 넘겨진 인수들을 순서대로 나타냅니다.
이 외에도 변수를 지정하거나 조건문, 반복문, case문 등의 문법을 이용하여 원하는 작업을 수행하는 쉘을 만들 수 있습니다.
http://kldp.org/KoreanDoc/Shell_Programming-KLDP

7. 텍스트 프로세싱

리눅스에서 어떤 명령어를 수행하면 대부분 모니터 혹은 파일에 어떤 결과를 나타내는 텍스트가 디스플레이됩니다.
이를 통해 우리는 알고 싶어하던 결과를 확인할 수가 있습니다. 혹은 이런 결과들을 서로 조합하거나 분리하거나 원하는 내용만을 뽑아서 다른 이름의 파일로 저장할 수도 있습니다.
이런 작업을 편하고 깔끔하게 할 수 있도록 도와주는 명령 및 유틸리티들을 알아 봅니다.

(1) 정규표현식(Regular Expression)

정확한 문자열을 패턴, 즉 골라내고자 하는 대상으로 정하지 않고, 보다 일반적인 문자열을 패턴으로 지정할 때 사용하게 되는 것이 정규 표현식입니다.
예를들면 "A로 시작하고 세글자인 단어가 포함된 줄을 모두 골라라", "/bin/bash로 끝나는 라인을 모두 보여달라"라는 식의 명령을 말하는 것입니다.
정규 표현식은 grep 뿐 아니라, 비슷한 용도로 ed, sed, awk, ...에서도 사용되며, vi 와 emacs 에서도 제한적으로 사용됩니다.
무엇을 골라내라고 좀 더 포괄적으로 명령하기 위해서는 정규 표현식에 사용되는 기호가 무엇을 의미하는지에 대해 알고 있어야 합니다.
사용되는 기호를 정리해 보면 다음과 같습니다.

기호	설명
.	한문자 일치 c..l 이라면 cool, coal 등이 해당
*	0개 이상의 문자와 매치 c*l 이라면 corel, cool, coal, cristal ... 모두 해당 c로 시작해서 l로 끝나는 모든 것이 해당
^	라인의 처음이 패턴과 일치
$	라인의 끝이 패턴과 일치
〈	단어의 시작이 패턴과 매치되는 단어를 가진 라인을 고를 때
〉	단어의 끝이 패턴과 매치되는 단어를 가진 라인을 고를 때
[ ]	괄호 안의 문자 중 하나 혹은 정해진 범위와 매치
[^]	괄호 밖의 문자와 매치
\	다음에 나오는 특수문자를 글자 그대로 인식하여 매치

(2) grep Family

어떤 패턴을 포함하고 있는 라인을 찾고자 할 때, 유용하게 사용하는 명령이 바로 grep 명령입니다.
grep은 기본적으로 주어진 파일 혹은 표준입력을 통해 들어온 내용(예를 들어 명령의 결과를 입력으로 사용할 수도 있다)에서 원하는 패턴을 포함하고 있는 라인을 찾아서 화면에 보여 줍니다.
grep(Generalized Regular ExPressions)은 큰 사이즈의 텍스트 파일 혹은 표준 입력사항에서 특정 패턴을 포함하고 있는 라인을 찾아서 보여 줍니다.
다음 세가지의 종류가 있습니다.

명령	설명
grep [옵션] 패턴 [대상파일들]	강력한 패턴 매칭 템플릿을 정의하기 위해 정규표현식을 사용할 수 있습니다.
egrep [옵션] "패턴|패턴|..." [대상파일들]	확장된 정규표현식을 사용하며 찾아낼 패턴을 여러 개 지정할 수 있습니다. | 기호는 불린 연산자 OR에 해당하므로 정해진 패턴들에 포함되는 모든 라인을 보여 줍니다.
fgrep [옵션] 패턴 [대상파일들]	패턴과 정확히 일치하는 것만을 찾아 줍니다.

명령과 함께 쓰이는 옵션들에 대한 설명입니다.

옵션	설명
-v	패턴과 일치하는 않는 라인을 보여 줍니다.
-c	패턴과 일치하는 라인의 개수만을 보여 줍니다.
-i	대소문자 구분없이 보여 줍니다.
-n	패턴과 일치하는 라인의 라인넘버를 앞에 붙여 보여 줍니다.

다음의 명령을 수행하고 결과를 확인하시기 바랍니다.

```
# ls -l | grep '^d'
# grep '50.' /etc/passwd
# ls -a | grep '^.[a-zA-Z0-9]'
# fgrep httpd /var/log/messages
# ls /etc  | egrep "conf|cfg"
```

(3) 텍스트 처리를 위한 명령어들

유닉스는 전통적으로 표준입력으로부터 얻어지는 캐릭터들을 읽어서 처리하고 그 결과를 표준출력으로 내보내기 위한 여러종류의 유틸리티들 (혹은 명령어들)을 제공합니다. 복잡한 텍스트 처리를 이런 유틸리티들과 파이프들을 통해 연결하므로써 처리할 수 있도록 해 줍니다. 이 유틸리티들의 대부분과 이와 연관된 프로그램들은 GNU textutils 패키지에 포함되어 있습니다.

 # rpm -ql textutils | grep /usr/bin

이러한 유틸리티들은 명확하지 않은 사용법을 가질 수도 있습니다. 즉, 꼭 이렇게 쓰여야만 한다는 단순한 사용법이 아니고 필요에 따라 여러가지 형태로 사용될 수 있으며 그 활용 범위가 매우 다양합니다.
예를 들어 리눅스는 파일의 내용을 모니터에 보여주는 명령을 따로 가지지 않습니다. 대신, cat 같은 명령으로 이러한 작업이 가능합니다.

 # cat /etc/passwd

이 명령은 모니터에 /etc/passwd 파일의 내용을 보여줄 것입니다. 이것을 조금 응용해 본다면, 출력 위치를 재지정하여 모니터가 아닌 파일로 그 결과를 저장할 수도 있습니다. 이렇게 하면 cp 명령을 쉽게 만들어 낼 수도 있습니다.
cat의 다른 사용 예를 보면,

 # cat test{1,2,3} 〉 test.results

이것의 결과는 test1, test2, test3 라는 파일의 내용을 모두 포함하는 test.results라는 이름의 새로운 파일을 생성합니다.

또 다른 예로, 다음은 cut을 사용하여 현재 등록되어 있는 사용자들이 사용하는 쉘을 보여 주는 명령입니다.

 # cut -d: -f7 /etc/passwd | sort | uniq

먼저 사용자 정보가 들어 있는 /etc/passwd 파일을 읽고, sort 명령으로 정렬한 뒤, 중복되는 쉘인 경우는 출력에 포함시키지 않도록 uniq처리를 한 것입니다.
다음은 텍스트 처리를 위한 명령어들입니다. 자세한 설명과 용법은 info 혹은 man 페이지로부터 도움을 받도록 합니다.

명령	설명
comm	정렬된 두 파일의 공통된 라인을 비교
csplit	주어진 인수를 기준으로 파일을 분리
cut	하나 혹은 그 이상의 파일들로부터 선택된 컬럼들과 필드들을 잘라냅니다.
expand	주어진 파일의 tab을 그에 맞는 space로 변경
fmt	라인을 추가하거나 삭제하여 특정 폭의 텍스트로 변환
fold	정해진 길이 보다 크지 않도록 라인 폭을 자릅니다.
head	주어진 파일의 처음 부분을 원하는 라인 수 만큼 출력
join	주어진 두 파일의 같은 라인을 출력
nl	주어진 파일의 라인을 붙여서 출력
paste	주어진 파일들을 병합, 각 컬럼은 기본적으로 tab으로 나눔 **# paste data1 data2 〉 data3** data1 과 data2 두 개의 컬럼으로 된 파일 data3 를 만듭니다.
pr	파일을 출력을 위한 형식으로 바꿔 줍니다.
split	큰 파일을 여러개의 파일로 나눈다. 기본적으로 1000라인.
sum	체크섬으로 파일의 블록수 등을 계산하고 출력, 파일의 전송 후 확인하는데 유용
tac	라인을 반대로 위치시켜 출력, 맨 끝줄이 맨 앞으로.
tail	파일의 마지막 열줄을 출력
tr	문자를 변환합니다. **# cat test | tr '[A-Z]' '[a-z]'** test 라는 파일에 있는 대문자를 소문자로 바꿔서 출력합니다.
uniq	중복된 라인을 지웁니다. **# sort names | uniq -c** name 파일을 정렬하고 중복된 경우 중복된 개수를 앞에 출력
unexpand	space 문자를 tab 문자로 변환
wc	문자, 단어, 줄수를 출력합니다. **# who | wc -l** who 명령 결과의 라인 수(현재 사용자의 수를 출력)

8. 압축 파일의 이용

리눅스는 파일을 압축하거나 묶는 많은 방법을 제공합니다.
유닉스의 기본 철학인 '작은 것이 아름답다' 는 정신을 이어 받아 여러 프로그램이 엮어져서 사용됩니다.
리눅스에서는 디렉토리나 여러 파일들을 압축하기 위해서 파일들을 묶고 압축하는 순서로 합니다. 물론, 하나의 파일을 할 경우에는 바로 압축을 하면 됩니다. 리눅스에서 압축 방법은 소프트웨어를 설치하거나 업그레이드 할 때와 시스템을 백업하는데 사용되므로, 반드시 익혀두시기 바랍니다.

(1) tar를 이용하여 파일들을 하나로 묶기

리눅스에서 여러 파일들을 압축하기 위해서는 파일들을 묶어야 한다고 했습니다.
이러한 역할을 하는 프로그램이 바로 tar 입니다. tar 은 단순히 여러 파일들을 묶어서 하나의 덩어리로 만드는 작업을 하며, 압축의 효과는 없습니다. 그러므로, 다음에 소개할 압축 프로그램들로 이 덩어리 파일을 압축하면 됩니다. 한가지 주의 할 점은, tar 사용할 때 반드시 f 옵션을 사용하여 대상 파일을 지정해야 한다는 것입니다. 그래야 다른 옵션을 사용할 때 대상 파일을 사용할 수 있습니다.

사용법 : tar 〈옵션〉〈만들 파일이름〉〈묶을 파일이름〉

옵션	시행
-c	tar 파일을 만들때 사용합니다.
-x	묶여진 tar 파일을 해체할 때 사용합니다.
-v	파일들을 묶거나 해체할 때 파일들의 이름과 크기를 표시합니다.
-f	사용할 tar 파일을 지정합니다.
-t	묶여진 tar 파일의 내용물을 출력합니다. * 위의 옵션들은 -cvf 와 같이 이어서 사용할 수 있습니다.

tar 의 사용방법은 몇가지 예제로 알아 보도록 하겠습니다.

예제	시행
tar -xvf example.tar	example.tar의 파일들을 해체하고 해체되는 파일들을 표시합니다.
tar -cvf backup.tar /home/ftp/pub	/home/ftp/pub 디렉토리와 그 안의 파일들을 backup.tar 이름으로 묶습니다.
tar -tvf example.tar	example.tar 안에 묶여있는 파일들의 리스트를 표시합니다.

(2) gzip 사용하기

가장 널리 사용되는 압축 방법은 gzip 을 사용하는 것입니다.
gzip 파일은, 확장자 .gz 으로 구분할 수 있으며, gzip 파일의 압축을 해제 하려면, 단순히 gunzip 〈파일이름.gz〉 라고 입력하거나 gzip -d 〈파일이름.gz〉 과 같이 합니다.
gzip 으로 압축된 tar 파일의 확장자는 tgz 일때도 있습니다. 이러한 파일의 압축을 해제하고 아카이브를 푸는 방법에는 두 가지가 있습니다. .tgz 파일에 대해 gunzip 을 실행한 다음 tar를 실행하여 파일을 푸는 방법과 tar 명령에서 옵션 z를 사용하여 두 가지 작업을 한꺼번에 하는 것입니다. gzip 으로 압축명령을 실행하면 압축을 하는 대상파일은 삭제되며 또한 gzip으로 압축을 푸는 경우에는 압축 되었던 파일이 삭제됩니다.

사용법 : gzip 〈옵션〉〈파일이름〉

압축하기	gzip cvs.pdf
압축풀기	gzip -d cvs.pdf.gz 또는 gunzip cvs.pdf.gz
-l	gzip으로 압축된 파일의 정보를 출력합니다.
-1	빠르게 압축을 하는 대신에 압축율이 떨어집니다.
-9	압축률이 높은 대신 압축시간이 길어집니다.

(3) bzip2 사용하기

bzip2 는 gzip 보다 10%-20% 정도 압축률이 훨씬 뛰어나서 빠르게 인기를 얻고 있습니다.
bzip2 압축은 bz2 파일 확장자로 알 수 있으며, 단순히 bunzip2 〈파일이름.bz2〉라고 입력하여 압축을 해제할 수 있습니다.
또한 gzip 압축과 마찬가지로 tar 에서 j 옵션(tar-1.13 이하는 I -대문자 i)을 사용하여 압축을 해제 할 수 있습니다. tar xvfj 〈파일이름.tar.bz2〉 와 같이 사용합니다. gzip과 마찬가지로 bzip2으로 작업하면 대상파일은 작업 후 삭제됩니다.

사용법 : bzip2 〈옵션〉 〈파일이름〉

압축하기	bzip2 cvs.pdf
압축풀기	bzip2 -d cvs.pdf.bz2 또는 bunzip2 cvs.pdf.bz2

마지막으로, compress 명령을 사용하는 유닉스의 오래된 압축 방식이 있습니다.
이 방식의 파일은 더 이상 보기 힘들지만, 여전히 어딘가에서 마주칠 때가 있습니다. 파일 이름에 .Z 확장자가 있으면 compress 유틸리티로 압축한 것임을 알 수 있습니다.
이러한 파일의 압축을 해제 하려면 uncompress 〈파일이름.Z〉 을 사용합니다.

(4) tar 와 gzip, bzip2 같이 사용하기

여러 개의 파일들을 하나로 묶어 압축하는 것은 상당히 번거로운 작업이며 이러한 작업들은 tar의 확장옵션을 사용하여 더 간편하게 할 수 있습니다.
여러 개의 파일들을 묶어 압축을 하기 전에 디렉토리를 만들어 그 안에 파일들을 옮긴 후 압축하는 것이 좋습니다. 이렇게 하면 압축을 풀었을 경우, 다른 파일들은 섞이지 않게 되므로, 좀 더 파일들을 관리하기가 편하게 됩니다.
가령, junilove라는 사용자가 2001년 5월 23일에 다운로드한 pdf 파일들을 압축한다고 했을 경우, 그냥 묶어서 압축하는 경우보다 junilove_20010523_pdf 와 같이 짧고 쉽게 알아 볼 수 있도록 디렉토리를 만들어서 하는 것이 나중에 관리하는데 더 편리합니다.

1 tar로 묶어서 gzip으로 압축된 파일 풀어보기

가장 많이 사용되는 형식입니다. 이렇게 압축된 파일은 tar.gz 또는 tgz 과 같은 확장자 형식을 지니며 아래와 같이 간편하게 풀 수 있습니다.

```
#tar  xvfz  test.tar.gz
```

tar 로 묶고 bzip2로 압축된 파일은 tar.bz2 와 같은 확장자 형식을 가지며 다음과 같이하여 풀면 됩니다.

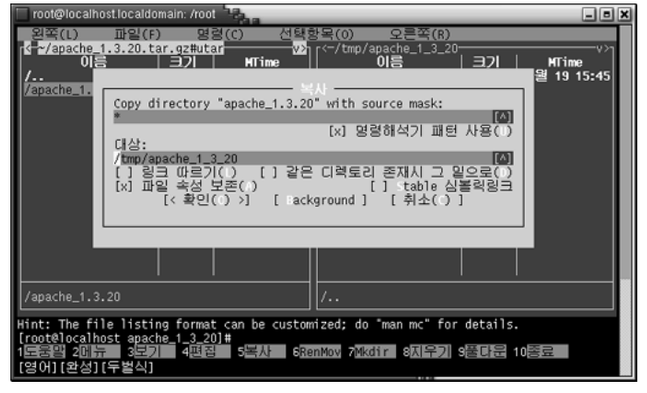
```
#tar  xvfj   test.tar.bz2
```

2 tar로 묶고 gzip으로 압축해 보기

tar 의 확장옵션을 사용하여 편하게 압축을 할 수 있습니다.
tar 에서 압축하는 옵션이 c 옵션과 gzip 을 수행할 수 있는 z 옵션을 사용하면 됩니다.
tar cvfz example.tar.gz example_dir
bzip2로 압축하고 싶다면, z 옵션 대신에 j 옵션(tar-1.13 이하는 I -대문자 i)을 사용하여 bzip2 가 수행되도록 하면 됩니다.
tar cvfj example.tar.bz2 example_dir

(5) MC 에서의 압축파일

손쉬운 사용방법과 편리한 인터페이스로 인기를 끌고 있는 미드나잇 커맨더 (MC)에서도 압축파일을 쉽게 관리할 수 있습니다.
MC에서는 ZIP, tar.gz, tgz, tar.bz2 등을 지원하며 원하는 파일에서 Enter 키를 누르면 압축파일 내용이 보입니다. 여기서 원하는 파일을 선택하여 F5를 눌러 원하는 디렉토리를 입력하면 자동으로 대상 디렉토리에 압축이 풀리게 됩니다.



[그림 II-9] mc 를 이용한 압축파일의 복사

9. 데이터 백업

(1) 백업의 중요성에 대해

일반적으로 데이터를 잃어버리게 되는 요인은 다음과 같은 것 들이 있습니다.
하드웨어의 문제, 소프트웨어의 버그, 사람의 실수, 그리고 자연 재해로 인한 경우입니다. 요즈음은 하드웨어들은 신뢰도가 높기는 하지만 문제가 발생할 가능성은 언제나 존재합니다. 중요한 데이터가 보관되는 하드웨어 중 가장 흔한 장치는 하드디스크 일 것입니다. 그렇지만 하드디스크는 비교적 불안한 장치입니다. 또한 소프트웨어도 별로 믿을 만한 것이 못되어서, 신뢰도 높은 견고한 소프트웨어라는 것은 드물다고 볼 수 있습니다. 더구나 사람은 정말로 믿어선 안 되는 존재이며 언제나 실수를 저지르게 마련입니다.
백업이란 것은 데이터가 지닌 가치를 보전하는 작업입니다. 데이터를 여러 개 복사해 둔다면, 그 중에 하나가 망가지더라도 별 문제가 되지 않을 것입니다.
백업을 평소에 철저히 해두는 것은 무척 중요합니다. 그러나 현실의 모든 일이 그러하듯이, 백업 작업 자체도 언젠가는 실패할 수 있습니다. 백업을 제대로 해내기 위한 방편 중 하나는, 모든 일에 철저를 기하는 것입니다. 그렇게 하지 않는다면, 언젠가 여러분의 백업이 더 이상 제 역할을 하지 못하는 심각한 사태에 직면하게 될 것입니다.
적어도 백업에 관해서 라면, 병적인 강박관념은 시스템 운영 담당자의 필수 조건이라고 할 수 있습니다.

(2) 무엇을 백업해야 할 것인가

많은 사람들이 가능한 모든 것을 백업하고 싶어합니다. 예외가 있다면 재설치가 가능한 소프트웨어들은 보통 백업할 필요가 없지만 그 설정 파일들은 나중에 다시 설정할 필요가 있게 되므로 꼭 백업해 두어야 합니다.
또 하나의 예외는 /proc 파일 시스템입니다. 이 곳에는 커널이 언제나 자동으로 생성하는 데이터들이 위치하므로, 백업을 받아둘 필요는 절대 없습니다.
어느 정도 중요한 부분으로서는 뉴스, 메일 스풀 디렉토리와 각종 로그 파일들, 그리고 /var 디렉토리 아래의 여러 파일들이 있습니다. 이들 중 무엇을 백업해야 할지는 관리자의 판단에 달려있습니다.
반드시 백업을 꼭 해야만 하는 가장 중요한 것은 각 사용자들의 개인 파일들이 있는 /home 과 시스템 설정 파일들 입니다. 이들은 주로 /etc 아래에 있지만, 그밖에 많은 설정 파일들이 파일시스템 전역에 흩어져 있습니다.

(3) 단순 백업

단순 백업 방식이라는 것은, 먼저 모든 것을 한꺼번에 백업하고 그 다음부터는 앞선 백업에서 변경된 부분만을 골라 백업하는 것을 말합니다. 여기서 맨 처음 하는 백업을 ful backup(완전 백업)이라고 하며, 그 다음부터는 incremental backups(변경분 백업) 이라고 합니다.
보통 풀 백업은 양이 많기 때문에, 여러 장의 플로피와 테이프를 사용해야 하는 고된 작업이 됩니다. 반면, 풀 백업을 해두면 복원하기는 변경분 백업보다 훨씬 쉽습니다. 풀 백업 이후에도 언제나 모든 것을 백업해 두도록 하면 복원 작업은 좀 더 효율적일 수 있을 것입니다. 다만 이렇게 하면 일이 좀 많아지는데, 물론 풀 백업과 변경분 백업을 사용해서 복원할 때의 작업량보다도 더 과중한 작업을 하면서까지 이렇게 할 필요는 없습니다.

1 tar를 사용해 백업하기

tar를 사용하면 풀 백업을 쉽게 할 수 있습니다.

```
# tar --create --file /dev/ftape /usr/src
tar: Removing leading / from absolute path names in the archive
#
```

위는 tar의 GNU 버전과 긴 이름 옵션을 사용한 예입니다. 전통적인 tar는 원래 한 문자 옵션만을 인식합니다. 또한 GNU tar는 한개 테이프나 플로피에 다 들어가지 않는 큰 용량의 백업도 다룰 수 있으며, 아주 긴 경로명도 사용할 수 있습니다.
만일 백업이 한 개 테이프에 다 들어가지 않는다면, multi-volume (-M) 옵션을 사용하면 된다:

```
# tar -cMf /dev/fd0H1440 /usr/src
tar: Removing leading / from absolute path names in the archive
Prepare volume #2 for /dev/fd0H1440 and hit return:
#
```

플로피를 사용할 때에는 백업 받기 전에 꼭 포맷을 하여야 한다는 점을 주의해야 합니다.
 tar가 새 플로피를 요구할 때, 새 플로피를 넣고 다른 가상 터미널에서 포맷을 먼저 한 뒤 백업을 계속해서 받을 수도 있습니다.
백업을 받고 나서는 그것이 제대로 되었는지 확인, 비교를 해야 합니다.
이를 위해서는 --compare (-d) 옵션을 사용하면 됩니다.

```
# tar --compare --verbose -f /dev/ftape
usr/src/
usr/src/linux
usr/src/linux-1.2.10-includes/
....
```

#
확인, 비교 과정에서 실패한 백업본을 그대로 방치한다면, 나중에 원본이 손상되고 난 후에야 그 백업본이 무용지물이었다는 사실을 깨닫게 될 것입니다.
—newer (-N) 옵션을 사용하면, tar를 사용해 변경분 백업을 할 수 있습니다.

```
# tar --create --newer '8 Sep 1995' --file /dev/ftape /usr/src --verbose
tar: Removing leading / from absolute path names in the archive
usr/src/
usr/src/linux-1.2.10-includes/
usr/src/linux-1.2.10-includes/include/
usr/src/linux-1.2.10-includes/include/linux/
usr/src/linux-1.2.10-includes/include/linux/modules/
usr/src/linux-1.2.10-includes/include/asm-generic/
usr/src/linux-1.2.10-includes/include/asm-i386/
usr/src/linux-1.2.10-includes/include/asm-mips/
usr/src/linux-1.2.10-includes/include/asm-alpha/
usr/src/linux-1.2.10-includes/include/asm-m68k/
usr/src/linux-1.2.10-includes/include/asm-sparc/
usr/src/patch-1.2.11.gz
#
```

아쉽게도, tar는 파일의 inode 정보(파일의 이름과 퍼미션의 변경같은 정보)가 변경된 것을 알아내지 못합니다. 이 문제는 find를 사용해 지난번 백업의 파일리스트와 현재 파일시스템을 비교해 보는 방법으로 해결할 수 있습니다.

② tar를 사용해 파일 복원하기

tar의 —extract (-x) 옵션을 사용하면 파일들을 추출해 낼 수 있습니다.

```
# tar --extract --same-permissions --verbose --file /dev/fd0H1440
usr/src/
usr/src/linux
usr/src/linux-1.2.10-includes/
usr/src/linux-1.2.10-includes/include/
usr/src/linux-1.2.10-includes/include/linux/
usr/src/linux-1.2.10-includes/include/linux/hdreg.h
usr/src/linux-1.2.10-includes/include/linux/kernel.h
...
#
```


또한 커맨드 라인 상에서 이름을 명시해 주면, 특정 파일들과 디렉토리들을(그 안의 파일들과 하위 디렉토리를 포함해서) 빼낼 수가 있습니다.

```
# tar xpvf /dev/fd0H1440 usr/src/linux-1.2.10-includes/include/linux/hdreg.h
usr/src/linux-1.2.10-includes/include/linux/hdreg.h
#
```

백업본에 어떤 파일이 들어있는지 보기만 하려면 —list (-t) 옵션을 쓰면 됩니다.

```
# tar --list --file /dev/fd0H1440
usr/src/
usr/src/linux
usr/src/linux-1.2.10-includes/
usr/src/linux-1.2.10-includes/include/
usr/src/linux-1.2.10-includes/include/linux/
usr/src/linux-1.2.10-includes/include/linux/hdreg.h
usr/src/linux-1.2.10-includes/include/linux/kernel.h
...
#
```

tar는 백업본들을 순서대로만 읽기 때문에, 큰 백업본을 다루기엔 상당히 느립니다. 더구나 테이프 드라이브 같은 순차적 저장 장치들은, 원천적으로 랜덤 엑세스 데이터베이스 테크닉을 사용할 수가 없습니다.
또한 tar는 지워버린 파일들을 제대로 다루지 못합니다. 만약 풀 백업본 하나와 변경분 백업본 하나를 가지고 복원 작업을 한다고 했을 때, 두 백업본 사이에 지워버린 파일이 있다면 그 파일은 다시 복원되어 나타나게 됩니다. 이렇게 꼭 지워져야만 하는 민감한 파일까지도 다시 복원된다는 사실은 큰 문제라고 할 수 있습니다.

(4) 압축을 사용한 백업

백업은 큰 저장 용량을 필요로 하므로 압축할 수 있다면 훨씬 비용이 싸게 먹힐 것입니다. 이렇게 하는 데는 몇 가지 방법이 있습니다. 어떤 프로그램은 압축 백업 지원을 내장하고 있기도 한데, 예를 들면 GNU tar의 —gzip (-z) 옵션은 백업 받는 내용을 gzip 압축 프로그램으로 파이프 연결시켜 준다. 그리고 gzip을 통해 압축된 내용이 백업 매체에 기록됩니다.
그러나 안타깝게도, 압축된 백업은 문제를 일으킬 소지가 있습니다. 압축이 이루어지는 근본 원리에 비춰보면, 전체 압축 데이터 중에서 단 하나의 비트만 손상되어도 다른 모든 데이터들이 쓸모 없게 되고 만다는 것을 알 수 있습니다.
어떤 백업 프로그램은 이런 문제에 대체하기 위한 자체 에러 수정 기능을 갖고 있기도 하지만, 그마저도

에러가 많이 발생하면 속수무책일 수 밖에 없습니다. 예를 들어, GNU tar를 써서 하나의 파일로 압축된 백업본을 만들었을 경우 만일 여기서 딱 하나의 비트가 에러를 일으킨다면, 이 백업은 모두 쓸모없게 되므로 이래서는 곤란합니다.
한가지 대안은 각각의 파일을 따로 압축하는 것입니다. 이렇게 하면, 파일 하나가 손상되었다고 해서 전체 백업을 모두 못쓰게 되는 일은 없을 것입니다. 결국 손상된 파일은 포기할 수 밖에 없지만, 그렇다고 해서 모든 파일을 압축하지 않는 것보다는 이 방법이 좀 낫습니다. afio 프로그램(cpio의 개정판)을 쓰면 이렇게 할 수 있습니다.

10. 주기적으로 프로그램 실행하기

배치작업의 등록 및 관리는 시스템 관리자로서의 필수적인 영역중의 하나입니다.
배치작업의 사전적 의미는 "프로그램을 적재하고 작업을 실행할 때, 작업의 중단없이 연속적으로 컴퓨터를 점유하여 처리하는 작업"입니다.

(1) batch 와 at

배치작업을 실행시키기 위하여 시스템 사용자가 사용할 수 있는 명령어는 많습니다.
보통 배치작업 등록이란 시스템 관리자가 cron table에 crontab이라는 명령을 이용하여 특정 명령을 정해진시간에 실행하도록 환경을 설정하는 것을 말합니다.
먼저 batch, at, cron 명령의 차이점은 아래와 같습니다.

명령어	시작시간 지정	우선순위	주기
batch	불가능	낮은	1회
at	가능	보통	1회
cron	가능	보통	반복가능

batch는 작업이 더 낮은 우선순위로 백그라운드에서 실행되도록 계획합니다. 작업을 실행할 시기는 시스템이 결정합니다. at은 지정된 시간에 작업이 실행되도록 합니다. cron은 사용자가 지정한 주기로 하나 이상의 작업을 반복 작업합니다.

(2) cron

cron과 관련된 명령어는 다음과 같습니다.
crontab -l : cron table에 등록되어 있는 내용을 출력
crontab -r : cron table의 내용을 삭제
crontab -e : cron table의 내용을 수정(vi 에디터를 이용하여 /tmp/에 임시화일 생성)
crontab cronfile : cronfile 의 내용을 cron table에 등록

1 cron table 형식

먼저 cron table의 형식(cronfile)에 대해 알아봅시다.

cron table은 모두 6가지 필드로 이루어져 있습니다.
각 필드는 하나이상의 space로 구분되며 하나의 배치작업은 한 라인에 정의됩니다.
첫 필드에 #이 나타나면 그 라인은 주석 처리됩니다.
매시마다 시스템의 부하량(load average)를 /tmp/load.log라는 파일에 저장한다고 가정해 봅시다.
시스템의 부하량 점검은 uptime(1) 명령으로 알 수 있습니다.
이러한 배치작업을 실행하기 위한 cron table의 내용은 아래와 같습니다.

```
0 * * * * W 〉〉 /tmp/load.log
```

즉, cron table 형식은 다음과 같습니다.
minite hour day month week command
예를 들어 다음과 같은 라인이 있다고 합시다.
1 2 3 4 5 uptime 〉〉 /tmp/load.log
위cron table은 금요일(5) 이면서 4월(4) 3일(3) 02시(2) 01분(1) 에
"uptime 〉〉 /tmp/load.log" 라는 명령을 실행시키라는 것입니다. (일요일은 0으로 표현됩니다.)
만약 13일 금요일 13시 정각마다 위 프로그램을 실행 시킬려면 다음처럼 합니다.
0 13 13 * 5 uptime 〉〉 /tmp/load.log
10-2-2. cron table 예제
예 1) 만약 uptime 〉〉 /tmp/load.log를 loveme이라는 사용자가 실행시키게 하고 싶다면?
0 13 13 * 5 su - loveme -c "uptime 〉〉 /tmp/load.log"
즉 command 필드에서 su 명령을 사용하여 프로그램을 loveme이라는 사용자가 실행하게 하면 됩니다.
-c 옵션은 su 명령에 있는 옵션인데 -c 옵션뒤에 command 가 나온다는 의미입니다.
예 2) 만약 w 〉〉 /tmp/load.log를 09시부터 18시동안 30분 마다 실행하고 싶다면?
0,30 09-18 * * * uptime 〉〉 /tmp/load.log

(3) /var/spool/cron/ 디렉토리

사용자가 "crontab cronfile" 명령을 사용하든, crontab -e 명령을 사용하든 시스템은 실제 cron table 의 내용을 /var/spool/cron/ 디렉토리에 사용자 이름으로 복사하여 관리합니다.
즉, crontab -e 라는 명령을 사용하면 /var/spool/cron/ 디렉토리내의 파일을 /tmp/ 디렉토리에 복사하여 임시로 사용한 후, 내부적으로 crontab 명령을 실행하는 것입니다.
따라서 /var/spool/cron/ 디렉토리내의 파일을 편집만 한다고 해서 cron table이 변경이 되는 것은 아닙니다. crontab 이라는 명령어로 해당 파일을 실행해야 합니다.
/var/spool/cron/ 디렉토리내의 파일을 인위적으로 손대는 것은 바람직하지 않습니다.
그러므로 /var/spool/cron/에 위치한 파일을 수정하지 말고, crontab -e, crontab cronfile 등의 명령을 사용해야 합니다.