도스/윈도우 에서 리눅스로

저자 Guido Gonzato, *guido@ibogeo.df.unibo.it* `<mailto:guido@ibogeo.df.unibo.it>` 버전 1.3.2 1999년 2월 22일

역자 : 김 성 수 (*songsookim@yahoo.com* `<mailto:songsookim@yahoo.com>`)

이 하우투는 자유 유닉스 클론(복제품)인 리눅스로 전환하려고 작정한 모든 도스/윈도우 사용자들(조금 있으면 '과거의'란 수식어가 붙을)을 위한 글이다. 이 문서의 목적은 독자가 갖고 있는 도스나 윈도우 지식을 리눅스 환경으로 변환할 뿐만 아니라 두 OS간에 파일과 그 밖의 리소스를 주고 받는데 힌트를 제공하는 것이다.

목 차

제 1 절 소개

1.1 리눅스가 당신에게 적합한가?

도스(윈도우 포함) 세계에서 리눅스로 바꾸고 싶은가? 잘 생각했다: 리눅스는 기술적으로 도스, 윈도우9x 심지어는 윈도우NT 보다 훨씬 훌륭하다. 하지만 당신에게는 유용하지 않을 수도 있다는 사실을 잊지 말라. 도스/윈도우와 리눅스 사이에는 커다란 차이점이 있다:

- 윈도우에서는 마이크로소프트 오피스와 많은 게임을 실행시킬 수 있다. 설치와 설정이 쉬운 것으로 인식된다. 아주 불안정하다. 성능이 형편없다. 심심하면 시스템이 죽는다.

- 리눅스는 스타오피스와 많은 테크니컬한 소프트웨어 그리고 몇 개 안 되는 게임을 실행시킬 수 있다. 설치와 설정이 어려울 수도 있다. 바위처럼 안정적이다. 나무랄 데 없는 성능을 갖고 있다; 시스템이 죽는 일은 매우 매우 드물다.

무엇이 필요한 지는 자신에게 달려있다. 더욱이 리눅스는 힘을 준다. 하지만 그 파워를 마음대로 쓸 수 있게 되기 까지는 시간을 들여서 공부를 해야 한다. 그러므로 필요한 프로그램이 대체로 상업 프로그램인 경우 또는 새로운 명령어와 개념을 배우고 싶지 않으면 딴데 가서 알아 볼 일이다. 많은 리눅스 입문자들이 초기의 어려움 때문에 포기한다는 사실을 잊지 말라.

리눅스가 사용하기에 편리해지려면 아직은 손질이 필요하다. 하지만 많은 문서들을 읽고 또 몇 달 동안 사용하기 전에는 능숙해질 기대를 하지 말라. 리눅스에서는 순식간에 좋은 결과를 얻지 못한다. 하지만 당신이 올바른 사용자라면 리눅스를 통해서 당신 컴퓨터의 열반의 세계를 발견하게 될 것이다. 그런데, 리눅스 + 도스/윈도우 가 같은 컴퓨터에 행복하게 공존할 수도 있다.

다음과 같은 것들을 이미 갖추고 있어야 한다.

- 기본적인 도스 명령어와 개념들을 이해한다.
- 리눅스 그리고 가능하면 X 윈도우 시스템 (줄여서 X11)이 PC에 제대로 설치되어 있다;
- 쉘(command.com에 해당하는 것)로 bash를 사용한다.
- 특별히 지적하지 않으면 이 문서에 있는 모든 정보는 낡고 후진 도스를 겨냥한 것이다. 윈도우에 대한 정보는 사방에 널려있다. 하지만 유닉스를 엉터리로 흉내낸 도스와는 달리 윈도우는 리눅스와 완전히 다르다. 그리고 이 문서는 완전한 입문서도 아니고 한 권만 떼면 만사 OK인 가이드북도 아니라는 점을 명심하라! 이 문서의 최신 버전은 다음의 장소에서 여러 가지 포맷으로 구할 수 있다. *sunsite.unc.edu:/pub/Linux/docs/howto* `<ftp://sunsite.unc.edu:/pub/Linux/docs/howto>` 한글판은 *www.kldp.org* `<http://www.kldp.org>`

1.2 좋다. 좀 더 이야기 해다오.

당신은 PC에 리눅스와 필요한 프로그램들을 설치하였다. 자신에게 계정을 만들어 부여했을 테고 리눅스가 운영 중일 것이다. 이제 막 이름과 비밀번호를 입력하고 화면을 쳐다보면서: '흠 이젠 뭘 하지?' 하고 생각하고 있을 것이다. 자, 실망하지 말라. 이제 거의 도스/윈도우 에서 하던 것들은 물론이고 그 보다 훨씬 더 많은 일들을 할 준비가 되어 있다. 리눅스 대신 도스/윈도우를 띄웠다면 다음과 같은 일들을 할 것이다.

- 프로그램 실행과 파일 생성, 복사, 보기, 지우기, 인쇄, 파일 이름 변경;
- 디렉토리에서 CD, MD, RD, DIR 명령 실행
- 플로피 포맷과 파일을 이리저리 복사하기;
- 시스템 다듬기;
- 인터넷 서핑;
- 제일 좋아하는 언어로 .BAT 파일과 프로그램 짜기
- 그 밖의 1%

반갑게도 이런 작업들이 도스에서와 비슷한 방법으로 리눅스에서도 이루어질 수 있다. 도스에서는 보통의 사용자들이 100개가 넘는 명령어 중 극히 제한된 숫자만 사용한다: 어찌 말하면 리눅스에서도 마찬가지라고 할 수 있다.

1.2.1 기본 개념

뭔가 배우려면 발을 담그는 것이 제일 좋다. 나는 리눅스를 실험하고 갖고 놀 것을 적극 권장한다. 'root'로 로그인 하지 않는 한 시스템을 아주 망치지는 않을 것이다.

몇 가지 알아 둘 것은:

- 우선 리눅스를 안전하게 종료할 줄 알아야 한다. 텍스트 모드 스크린이 보이면 <CTRL-ALT-DEL>을 눌러라. 그리고 시스템이 재시동에 들어가길 기다렸다가 PC의 스위치를 끈다. X 윈도우 시스템을 사용하고 있다면 먼저 <CTRL-ALT-BACKSPACE>를 누른 다음 <CTRL-ALT-DEL>을 누른다. 그냥 스위치를 끄거나 리셋 단추를 누르는 일은 절대로 없어야 한다. 그랬다가는 파일 시스템을 망칠 수도 있다.

- 도스 또는 윈도우와는 달리 리눅스는 내장 보안 매커니즘을 갖고 있다. 파일과 디렉토리들은 Permission(permission)이 달려 있다. 결과적으로 파일이나 디렉토리 중에서 일반 사용자는 접근할 수 없는 것들이 있다. ('Permission과 소유권' 부분 참조). 반면 도스와 윈도우는 하드디스크에 있는 것을 몽땅 지우는 것이 가능하다.

- 'root'라고 불리는 특별한 사용자가 있다. 그가 바로 컴퓨터의 삶과 죽음을 마음대로 할 수 있는 모든 권한을 가진 시스템 관리자다. 당신의 PC에서는 바로 당신이 root가 될 것이다. root 자격으로 작업하는 것은 위험하다. 도스/윈에서와 마찬가지로 어떤 실수로 시스템에 치명적인 손상을 주거나 파괴할 수 있다. 꼭 필요한 경우를 빼고는 root로서 작업하지 말라.

- 리눅스가 복잡한 것은 바로 모든 것을 설정할 수 있기 때문이다. 사실상 모든 기능과 모든 응용프로그램을 하나 또는 몇 개의 설정 파일로 자신에게 맞게 다듬을 수 있다. 파워에 대한 대가로 복잡함을 지불하는 것이다.

- Redirection과 Piping은 도스에서는 부수적인 기능이지만 리눅스에서는 훨씬 더 강력하고 중요하다. 간단한 명령들을 합쳐서 복잡한 작업을 수행해 낼 수 있기 때문이다. Redirection과 Piping을 잘 배워 둘 것을 적극 권장한다.

1.2.2 도움 얻기

리눅스에 대한 도움은 다양한 방법으로 얻을 수 있다. 가장 중요한 것은:

- 도큐멘테이션을 읽는다 – 이것은 필수다. 비록 지금 읽고 있는 하우투가 리눅스에 대한 소개가 될 수는 있겠지만 꼭 읽어야 될 책이 몇 권 있다. 맷 웰쉬(Matt Welsh)의 리눅스 설치와 익숙해지기(Linux Installation and Getting Started) *sunsite.unc.edu/mdw/LDP/gs/gs.html* `<http://sunsite.unc.edu/mdw/LDP/gs/gs.html>`), 래리 그린필드(Larry Greenfield)의 리눅스 사용자 가이드(Linux User Guide) *sunsite.unc.edu/pub/Linux/docs/linux-doc-project/users-guide* `<ftp://sunsite.unc.edu/pub/Linux/docs/linux-doc-project/users-guide>` 그리고 리눅스 FAQ *sunsite.unc.edu/mdw/FAQ/Linux-FAQ.html* `<http://sunsite.unc.edu/mdw/FAQ/Linux-FAQ.html>`. 이들 중 적어도 한 권을 읽기 전에는 죄책감을 가져라.

- 컴퓨터에 설치된 패키지에 대한 Documentation은 /usr/doc/ 아래에 있는 경우가 많다.

- Shell의 '내부 명령들'에 대한 도움이 필요하면 'help' 라고 입력하거나 또는 좀 더 좋은 방법으로 'man bash' 또는 'info bash' 라고 입력한다.

- 어떤 명령에 대한 도움말은 'man 명령어' 라고 입력하면 그 명령어의 매뉴얼 페이지를 불러온다. 다른 방법으로는 'info 명령어' 와 같이 입력할 수도 있는데 해당 명령어에 연관된 info 페이지가 있으면 불러온다. info는 하이퍼텍스트 기반 Documentation 시스템으로, 처음에는 자유 자재로 사용하기가 쉽지 않다. 마지막으로 'apropos 명령어' 또는 'whatis 명령어' 를 사용할 수도 있다. 이 모든 명령어를 종료할 때는 'q'를 누르면 된다.

- 마지막으로 인터넷이 있다. 도움을 얻는 적절한 장소는 news*comp.os.linux.setup* `<http://comp.os.linux.setup>`>와 같은 유즈넷이다. 나는 일이 너무 많으니 제발 내게 이메일을 보내는 일은 삼가 했으면 좋겠다.

1.3 일러두기

이 문서 전체에서 예제는 다음과 같은 형식을 따른다. <...>는 필수 아규먼트고 [...]는 옵션 아규먼트다. 예를 들어:$ tar -tf <file.tar> [> redir_file]file.tar는 반드시 명시되어야 하며 redir_file로의 리다이렉션은 옵션이다. "RMP"는 더 알고 싶으면 Man 페이지를 읽어보란 뜻이다. 도큐멘테이션을 읽는 것에 대한 중요성은 아무리 강조해도 지나치지 않다. "DOSWin"은 "DOS/Winodws"를 의미한다.프람프트가 #로 나타나는 명령어는 루트에 의해서만 수행될 수 있다. (역자 추가 : 문장 중에 입력할 명령어가 나오는 경우가 있습니다. 원문에는 아무런 표시가 없으나 읽는 분들의 편의를 위하여 홑 따옴표 ' ' 로 묶었습니다. ' ' 속에 들어가는 것은 명령행에서 직접 입력할 내용을 표시합니다.)

제 2 절 마음이 급한 사람들을 위해서

스트라익 아웃 당하고 싶은가? 아래 테이블을 보라:

DOS	Linux	Notes
ATTRIB (+-)attr file	chmod <mode> file	매우 다름
BACKUP	tar -Mcvf device dir/	상동
CD dirname\	cd dirname/	거의 동일한 syntax 사용
COPY file1 file2	cp file1 file2	상동
DEL file	rm file	지운 파일 복구가 없음에 주의
DELTREE dirname	rm -R dirname/	상동
DIR	ls	syntax가 조금 다름
DIR file /S	find . -name file	완전히 다름
EDIT file	vi file	별로 좋아하지 않을 것임
	jstar file	도스의 Edit과 비슷함
EDLIN file	ed file	신경 쓰지 말 것
FORMAT	fdformat, mount, umount	아주 다른 syntax
HELP command	man command, info command	동일한 철학
MD dirname	mkdir dirname/	거의 동일한 syntax
MORE < file	less file	훨씬 좋다
MOVE file1 file2	mv file1 file2	상동
NUL	/dev/null	상동
PRINT file	lpr file	상동
PRN	/dev/lp0, /dev/lp1	상동
RD dirname	rmdir dirname/	거의 동일한 syntax
REN file1 file2	mv file1 file2	복수의 파일을 사용할 수 없음
RESTORE	tar -Mxpvf device	다른 syntax
TYPE file	less file	훨씬 좋다
WIN	startx	비고 가치도 없다!

명령어를 테이블에 나열한 것으로 충분하지 않은 사람들은 다음 섹션을 읽으라.

제 3 절 bash를 만나자

희소식 : 리눅스에서는 bash 쉘이 필요한 것을 알아서 타이프 해주고 멋진 라인 에디팅 기능들이 있기 때문에 타이핑이 적어진다. 우선, 화살표를 사용하면 이전의 명령행을 불러온다; 하지만 그 이상의 기능도 있다. <TAB> 키를 누르면 파일 이름과 디렉토리 이름을 완성시켜준다. 그래서 다음과 같이 타이핑하면

```
$ ls /uTABloTABbTAB
```

아래처럼 타이핑 하는 것이 된다.

```
$ ls /usr/local/bin
```

만약 같은 알파벳으로 시작하는 파일이나 디렉토리가 여럿 있으면

```
$ ls /uTABloTABiTAB
```

라고 입력하면 bash는 /usr/local/info를 의미하는지 /usr/local/include를 의미하는지 모르기 때문에 멈춘다. 그럴 때는 그 다음 글자를 치고 <TAB>을 누르면된다. 또 다른 편리한 키 누르기로 <ESC-BACKSPACE>가 있다. 이것은 왼쪽에 있는 단어를 통째로 지운다. 반면에 <ESC-D>를 누르면 오른쪽에 있는 단어를 지운다. <ESC-F>는 커서를 오른쪽으로 한 단어만큼 건너 뛰게 하고 <ESC-B>는 왼쪽으로 한 단어 만큼 건너 뛰게 한다. <ALT>키는 <ESC> 키와 동일하게 작동한다. 지금으로선 이 정도면 충분하다. 일단 이 단축키들에 익숙해지면 DOS 프람프트가 아주 불편하게 느껴질 것이다.

3.1 파일과 프로그램

3.2 파일 : 사전 지식

리눅스의 파일과 디렉토리 구조는 DOSWin의 구조와 매우 비슷하다. 파일들은 일정한 규칙에 따르는 파일 이름을 갖고 있으며 디렉토리 속에 저장돠고, 일부는 실행 가능하며 실행 가능한 파일들은 대부분 명령 스위치를 갖고 있다. 또한 와일드카드 문자, 리다이렉션, 파이핑을 사용할 수 있다.

차이점은 아주 적은데 다음과 같다:

- DOS에서는 파일 이름들이 소위 8.3 형태를 따른다. 예 NOTENOUG.TXT . 리눅스에서는 좀 더 잘 쓸 수 있다. ext2나 umsdos와 같은 파일 시스템을 사용해서 리눅스를 설치했다면 긴 파일 이름(최대 255 글자까지)을 사용할 수 있다. 그리고 점도 하나 이상 사용할 수 있다. 예 This_is.a.VERY_long.filename . 대문자와 소문자를 사용한 점을 눈 여겨 두자. 사실...
- 파일이름이나 명령에서 대문자와 소문자는 구별된다. FILENAME.tar.gz와 filename.tar.gz는 서로 다른 파일이다. ls는 명령어인데 LS와 같은 명령어는 없다.
- 윈도우 사용자들은 리눅스에서 긴 파일 이름을 사용할 때 주의할 점이 있다. 파일 이름에 여백(스페이스)이 들어가면 (불가능한 것은 아니지만 사용하지 않는 것이 좋다) 파일 이름 언급할 때 을 겹 따옴표로 둘러싸야 한다.

  예를 들어:

```
$ # 다음의 명령은 "My old files"라는 디렉토리를 만든다
$ mkdir "My old files"
$ ls    My old files    bin     tmp
```

그리고 !*$&# 문자들은 사용할 수 없다.

- 특정 확장자를 의무적으로 사용하지 않아도 된다. 예를 들어 실행 프로그램이라고 반드시 .COM이나 .EXE 또는 배치 파일이라고 .BAT를 사용해야만 하는 것은 아니다. 실행 가능한 파일들은 ls -F 명령을 내려면 이름 끝에 아스테리스크 즉 ’*’로 표시되어 나타난다.

  예를 들면:

```
$ ls -F
I_am_a_dir/   cindy.jpg    cjpg*   letter_to_Joe    my_1st_script*  old~
```

  cjpg* 파일과 my_1st_script* 파일은 실행 가능파일들 즉 프로그램이다. DOS에서는 백업 파일들은 .BAK로 끝나는 반면 리눅스에서는 틸드 ’ ’로 끝난다. 또 점으로 시작하는 파일은 숨김 파일로 간주된다. 예를 들어 .I.am.a.hidden.file 파일은 ls 명령으로 보이지 않는다.

- 도스 프로그램의 스위치(옵션)는 ’/스위치’ 형식으로 사용한다. 리눅스에서는 ’-스위치’ 또는 ’–스위치’ 와 같이 사용한다. 예를 들어: dir /s는 ls -R 이 된다. PKZIP이나 ARJ 등 많은 도스 프로그램들이 유닉스 스타일의 스위치를 사용한다.

  이제는 ” DOS의 명령어들을 리눅스 명령어로 번역” 섹션으로 넘어가도 괜찮을 것 같다. 하지만 나라면 계속 읽겠다.

3.3 Symbolic Links

유닉스는 DOS에는 없는 파일 종류가 있다: 바로 Symbolic Link다. 이것은 파일이나 디렉토리를 가리키는 포인터로 생각할 수 있는데 그것이 가리키는 파일이나 디렉토리 대신 사용할 수 있다. 윈도우의 ’바로 가기’와 비슷하다. symbolic link의 예로 /usr/X11R6를 가리키는 /usr/X11; /dev/ttyS0 또는 /dev/ttyS1 둘 중 하나를 가리키는 /dev/modem 등이 있다.

symbolic link 만들기:

```
$ ln -s <file_or_dir> <linkname>
```

예:

```
$ ln -s /usr/doc/g77/DOC g77manual.txt
```

이렇게 하면 /usr/doc/g77/DOC 대신 g77manual.txt를 사용할 수 있다. 링크는 디렉토리 리스팅을 하면 다음과 같이 나타

```
$ls -F
g77manual.txt@
$ ls -l       (several things...)           g77manual.txt -> /usr/doc/g77/DOC
```

3.4 Permission(permission)과 소유권

DOS 파일과 디렉토리들은 다음과 같은 속성을 갖는다. A(압축), H(숨김), R(읽기 전용), S(시스템). 리눅스에서는 이들 중 H와 R만 의미를 갖는다. 숨김 파일은 점으로 시작하고 R 속성에 대해서는 다음을 계속 읽어보자.

유닉스에서는 파일에 ”Permission(permission)”과 소유자가 있다. 소유자는 또 ”그룹”에 속한다. 다음의 예제를 보라.

```
$ ls -l /bin/ls
-rwxr-xr-x  1  root  bin  27281 Aug 15 1995 /bin/ls*
```

제일 처음 필드는 /bin/ls 파일의 permission을 나타내는 것으로 root가 소유하며 bin이란 그룹에 속한다. 다른 정보는 차치하고 -rwxr-xr-x 가 왼쪽에서 오른쪽 방향으로 무엇을 의미하는지 기억해두자:

- 는 파일 종류의 일종이다. (- = 보통 파일, d = 디렉토리, l = 링크 등) rwx는 파일 소유자가 갖는 permission으로 읽기, 쓰기, 실행 을 나타낸다. r-x는 파일 소유자가 속한 그룹에 주어진 Permission으로 읽기, 실행을 나타낸다. (초보자라면 그룹을 사용할 일이 별로 없을 테니 그룹의 개념에 대해서는 설명하지 않겠다 ;-) r-x는 다른 모든 사용자들이 갖는 permission으로 읽기, 실행을 나타낸다.

디렉토리인 /bin 에도 permission이 붙어있다: "디렉토리와 permission" 섹션에 좀 더 자세한 설명이 있다. 바로 이러한 이유 때문에 루트가 아니고서는 /bin/ls 를 지울 수가 없는 것이다: 즉 지울 permission이 없는 것이다. 파일의 permission을 바꾸려면 다음과 같은 명령을 사용한다.

```
$ chmod <whoXperm> <file>
```

여기에서 who는 u(사용자, 즉 소유자), g (그룹), o (그 외), X는 + 또는 -, perm은 r(읽기), w(쓰기), x(실행) 중 하나. 다음은 chmod를 사용하는 전형적인 예다.

```
$ chmod +x file
```

이렇게 하면 해당 파일의 실행 permission이 설정된다.

```
$ chmod go-rw file
```

이렇게 하면 소유자 이외의 모든 사용자로부터 읽기와 쓰기 permission을 제거한다.

```
$ chmod ugo+rwx file
```

이것은 모든 이에게 읽기, 쓰기, 실행 permission을 부여한다.

```
# chmod +s file
```

이것은 소위 "setuid" 또는 "suid" 파일로 소유자의 특권과 함께 모든 사람이 실행시킬 수 있다. 전형적인 예로 루트 suid 파일을 볼 수 있다. 이 파일들은 보통 X 서버와 같이 중요한 시스템 파일인 경우가 많다.

permission을 지시하는 좀 더 간단한 방법은 숫자로 표현하는 것이다. rwxr-xr-x는 755로 표현할 수 있다(모든 문자에 해당하는 비트가 있다. —-는 0이고 –x는 1, -w-는 2, -wx는 3...). 어려워 보이지만 조금 연습하면 개념을 이해할 수 있을 것이다. 루트는 수퍼유저로 모든 이들의 파일 permission을 바꿀 수 있다. RMP.

3.5 파일 : 명령어 번역

왼쪽에는 도스 명령; 오른쪽에는 그에 해당하는 리눅스 명령어이다.

```
ATTRIB:        chmod
COPY:          cp
DEL:           rm
MOVE:          mv
REN:           mv
TYPE:          more, less, cat
```

`Redirection and plumbing operators:  < > >> |`

```
Wildcards: * ?

nul:        /dev/null

prn, lpt1:  /dev/lp0 or /dev/lp1; lpr
```

3.5.1 Examples

```
DOS                                                             Linux
---------------------------------------------------------------------
C:\GUIDO>ATTRIB +R FILE.TXT             $ chmod 400 file.txt
C:\GUIDO>COPY JOE.TXT JOE.DOC                   $ cp joe.txt joe.doc
C:\GUIDO>COPY *.* TOTAL                                 $ cat * > total
C:\GUIDO>COPY FRACTALS.DOC PRN                  $ lpr fractals.doc
C:\GUIDO>DEL TEMP                                       $ rm temp
C:\GUIDO>DEL *.BAK                                      $ rm *~
C:\GUIDO>MOVE PAPER.TXT TMP\                    $ mv paper.txt tmp/
C:\GUIDO>REN PAPER.TXT PAPER.ASC                $ mv paper.txt paper.asc
C:\GUIDO>PRINT LETTER.TXT                       $ lpr letter.txt
C:\GUIDO>TYPE LETTER.TXT                        $ more letter.txt
C:\GUIDO>TYPE LETTER.TXT                        $ less letter.txt
C:\GUIDO>TYPE LETTER.TXT > NUL                  $ cat letter.txt > /dev/null
                n/a                                             $ more *.txt *.asc
                n/a                                             $ cat section*.txt | less
```

알아둘 점:

- 리눅스에서 *는 좀 더 다양한 기능을 갖고 있다. *는 숨김 파일을 제외한 모든 파일을 의미한다. .*는 모든 숨김 파일들을 의미한다 (또한 현재 디렉토리는 '.'이고 상위 디렉토리는 '..'라는 점에 주의하라!) *.*는 .가 중간이 들어가 있는 파일들 또는 점으로 끝나는 파일들만을 의미한다. p*r은 'peter' 또는 'piper'를 의미한다. *c*는 'picked' 그리고 'peck'를 의미한다.

- more를 사용할 때는 <SPACE>를 누르면 계속 읽을 수 있으며 'q'를 누르면 종료한다. less는 화살표를 사용할 수 있어서 명령을 잘 몰라도 사용하기 쉽다.

- UNDELETE가 없으므로 어떤 것이든 지울 때는 한 번 더 생각해보도록.

- DOS의 < > >>에 더하여 리눅스에는 2>를 사용하여 에러 메시지를 리다이렉트 시킬 수 있다(stderr). 추가로 2>&1은 stderr을 stdout으로 리다이렉트하고 1>&2는 stdout을 stderr로 리다이렉트한다.

- 리눅스에는 와일드 카드가 하나 더 있다. 바로 [ ]이다. 사용법: [abc]는 a, b, c로 시작하는 파일들을 의미한다. *[I-N2-3]은 I, J, K, L, M, N, 1, 2, 3;으로 끝나는 파일들을 의미한다.

- lpr <file>은 해당 파일을 백그라운드로 인쇄한다. 프린트 큐를 확인할 때는 lpq를 사용한다. 프린터 큐에서 파일을 삭제하려면 lprm을 사용하면 된다.

- 도스의 RENAME과 같은 것은 없다. 즉 mv *.xxx *.yyy 처럼 사용할 수 없다. REN-과 같은 명령어는 *sunsite.unc.edu/pub/Linux/utils/file* `<ftp://sunsite.unc.edu/pub/Linux/utils/file>` 에서 구할 수 있다.

- cp -i와 mv -i를 사용하면 파일이 겹쳐 쓰기 될 때 경고를 볼 수 있다.

3.6 프로그램 실행: 멀티 태스킹과 세션

프로그램을 실행할 때는 DOS에서와 마찬가지로 그 프로그램의 이름을 입력하면 된다. 만약 프로그램이 저장되어 있는 디렉토리(”디렉토리 사용” 참조)가 PATH(”시스템 초기화 파일” 섹션 참조)에 속해있다면 프로그램이 실행될 것이다.

예외: DOS에서와는 달리 리눅스에서는 해당 프로그램이 현재 디렉토리에 있다 하더라도 PATH에 속해 있지 않는 한 실행되지 않는다. 에스카모타지: 프로그램이 prog이라면 ’./prog’이라고 입력한다. 전형적인 명령행은 다음과 같은 형태다.

```
$ command [-s1 [-s2] ... [-sn]] [par1 [par2] ... [parn]] [< input] [> output]
```

여기에서 -s1, ..., -sn은 프로그램 스위치, par1, ..., parn은 프로그램 매개변수(parameter)다. 하나의 명령행에 여러 개의 명령을 내릴 수 있다.

```
$ command1 ; command2 ; ... ; commandn
```

이상이 프로그램 실행과 관련된 모든 것이다. 하지만 이보다 한 차원 높게 사용하는 것도 어렵지 않다. 리눅스를 사용하는 중요한 이유 중 하나가 멀티태스킹 OS 이기 때문이다 —즉 여러 개의 프로그램(앞으로는 프로세스라고 부름)을 동시에 실행할 수 있다. 여러 개의 프로세스를 백그라운드에서 실행시키고도 또 계속해서 다른 작업을 하고 있어도 된다. 게다가 리눅스는 복수의 세션도 가능하다: 마치 여러 대의 컴퓨터를 한 곳에 놓고 사용하는 것과 같다!

- 가상 콘솔에서 세션 1..6으로 전환하려면 <ALT-F1> ... <ALT-F6>을 누르면 된다.
- 같은 가상 콘솔에서 현재의 세션을 그대로 두고 새로운 세션을 시작하려면 su - <loginname>이라고 입력하면 된다. 예: su - root. 이는 루트 permission만이 가능한 작업을 수행해야 하는 경우와 같은 때 정말 편리하다.
- 세션을 멈추려면 exit을 입력한다. 중지된 작업이 있을 때는 경고가 나온다(이에 대해서는 조금 있다가 다시 다룬다)
- 백그라운드에서 프로세스를 하나 시작하려면 명령행의 끝에 ’&’를 붙인다. $ progname [-switches] [parameters] [< input] [> output] & [1] 123 쉘은 해당 프로세스를 작업 번호(예 [1]; 아래 참조)와 PID(Process Identification Number - 프로세스 식별 번호 예제에서는 123)로 식별한다.
- 현재 실행중인 프로세스가 몇 개인지 확인하려면 ps ax라고 입력한다. 이는 현재 실행되고 있는 프로세스의 리스트를 출력해준다.
- 프로세스를 하나 죽이려면(종료) kill <PID> 를 입력한다. 어떤 프로세스를 제대로 종료하는 방법을 모를 때에는 그 프로세스를 죽일 필요가 있다.... 루트가 아닌 이상 다른 사용자의 프로세스를 죽일 수는 없다. 때론 -SIGKILL <PID>를 사용해야만 죽는 프로세스도 있다. 추가로, 프로세스를 멈추게 하거나 임시로 작동 중지시키거나 백그라운드로 보내거나 백그라운드에서 포오그라운드로 불러올 수 있다. 이런 문맥에서 프로세스는 "job - 작업"이라고 불린다.
- 현재 작업이 몇 개인지 확인하려면 jobs라고 입력한다. 여기에서 작업들은 PID가 아니라 작업 번호로 식별된다.
- 포오그라운드에서 실행중인 프로세스를 종료하려면 <CTRL-C>를 누른다(항상 작동하는 것은 아니다).
- 포오그라운드에서 실행중인 프로세스를 일시 중단하려면 <CTRL-Z>를 누른다(tkdehd).
- 어떤 작업을 포오그라운드로 불러오려면 fg <%job>이라고 입력한다. 마지막으로 백그라운드로 보내졌던 작업을 포오그라운드로 불러오려면 그냥 fg만 입력하면 된다.

- 작업을 죽이려면 kill <%job>이라고 입력한다. 여기에서 <job>은 1, 2, 3,....과 같은 것들이다. 이런 명령들을 사용하여 동시에 디스크를 포맷하고, 몇 개의 파일들을 압축하여 묶고, 프로그램을 컴파일하고, 압축 파일을 풀면서도 명령을 기다리고 있는 프람프트에 원하는 또 다른 명령을 내릴 수 있다. 윈도우를 이와 같이 사용하려고 해보면 그 성능이 얼마나 차이가 나는 지 금방 알 수 있다 (그것도 시스템이 죽지 않고 살아 있을 경우에만 말이다).

3.7 리모트 컴퓨터에서 프로그램 실행하기

remote.machine.edu라는 리모트 머신(원격 컴퓨터)에서 프로그램을 실행하려면 다음과 같이 입력한다:

```
$ telnet remote.machine.edu
```

로그인을 한 다음 원하는 프로그램을 시작하면 된다. 물론 그 리모트 머신에 쉘 어카운트가 있어야 한다. X11을 갖고 있다면 리모트 컴퓨터에서 X 애플리케이션도 실행시켜 자신의 X 화면에 나타나게 할 수도 있다. remote.machine.edu는 리모트 X 컴퓨터, local.linux.box는 자신이 사용하고 있는 리눅스 머신이라고 가정하자. remote.machine.edu에 존재하고 있는 X 프로그램을 local.linux.box에서 실행하려면 다음과 같이 하면 된다.

o X11, xterm 또는 그와 동격인 터미널 에뮬레이터를 실행한 다음 다음과 같이 입력한다:

```
$ xhost +remote.machine.edu
$ telnet remote.machine.eduo
```

로그인을 한 다음 다음과 같이 입력한다.

```
remote:$ DISPLAY=local.linux.box:0.0
remote:$ progname &
```

(DISPLAY..., 대신 setenv DISPLAY local.linux.box:0.0 을 입력해야 할 수도 있다. 이는 모두 리모트 쉘에 달려있다.)

자 새롭지않은가! 이제 progname은 은 remote.machine.edu에서 시작하여 당신의 컴퓨터 화면에 나타날 것이다. 하지만 모뎀은 너무 느리니 모뎀을 통해서 이 같은 작업을 하려는 것은 쓸모가 없을 것이다. 또한 이 같은 작업 방식은 불완전하고 불안정하다. *sunsite.unc.edu/LDP/HOWTO/mini/Remote-X-Apps* `<http://sunsite.unc.edu/LDP/HOWTO/mini/Remote-X-Apps>` 에 있는 “Remote X Apps mini-HOWTO” 를 읽어보라.

제 4 절 디렉토리 사용

4.1 디렉토리: 사전 지식

DOSWin에서와 리눅스에서 파일의 차이점에 대해서 알아보았다. 이번엔 디렉토리에 대해서 알아보자. DOSWin에서는 루트 디렉토리가 \, 리눅스에서는 / 이다. 비슷하게, 하위 디렉토리는 DOSWin에서는 \ 로 구분되고 리눅스에서는 / 로 구분된다. 다음 파일 경로의 예를 보자:

```
DOS:    C:\PAPERS\GEOLOGY\MID_EOC.TEX
Linux:  /home/guido/papers/geology/middle_eocene.tex
```

일반적으로 사용되는 것처럼 .. 는 상위 디렉토리, . 는 현재 디렉토리를 나타낸다. 리눅스 시스템에서는 cd, rd, md를 원한다고 해서 아무데서나 할 수 있는 것은 아니라는 점을 명심하라. 사용자마다 시스템 관리자가 나누어준 'home'이라고 불리는 디렉토리에 자기 '것'들을 갖고 있다. 예를 들어 내 PC에서 나의 홈 디렉토리는 /home/guido 이다.

4.2 디렉토리 permission

디렉토리 또한 permission이 있다. ”permission과 소유권” 섹션에서 본 것들은 디렉토리에도 해당된다(사용자, 그룹, 기타). 디렉토리에 대해서 rx는 그 디렉토리로 cd 할 수 있으며 w는 그 디렉토리에 있는 파일(물론 그 파일의 permission에 따라서)이나 그 디렉토리 자체를 지울 수 있다는 것을 의미한다. 예를 들어, 다른 사용자가 /home/guido/text에 들어오지 못하게 하려면 다음과 같이 하면 된다:

```
$ chmod o-rwx /home/guido/text
```

4.3 디렉토리

명령어 번역

```
DIR:            ls, find, du
CD:             cd, pwd
MD:             mkdir
RD:             rmdir
DELTREE:        rm -rf
MOVE:           mv
```

4.3.1 예제

```
DOS                                     Linux
---------------------------------------------------------------------

C:\GUIDO>DIR                                    $ ls
C:\GUIDO>DIR FILE.TXT                   $ ls file.txt
C:\GUIDO>DIR *.H *.C                    $ ls *.h *.c
C:\GUIDO>DIR/P                                  $ ls | more
C:\GUIDO>DIR/A                                  $ ls -l
C:\GUIDO>DIR *.TMP /S                           $ find / -name "*.tmp"
C:\GUIDO>CD                                     $ pwd
        n/a - see note                          $ cd
        ditto                                   $ cd ~
        ditto                                   $ cd ~/temp
C:\GUIDO>CD \OTHER                              $ cd /other
C:\GUIDO>CD ..\TEMP\TRASH               $ cd ../temp/trash
C:\GUIDO>MD NEWPROGS                    $ mkdir newprogs
C:\GUIDO>MOVE PROG ..                           $ mv prog ..
C:\GUIDO>MD \PROGS\TURBO                $ mkdir /progs/turbo
C:\GUIDO>DELTREE TEMP\TRASH      $ rm -rf temp/trash
C:\GUIDO>RD NEWPROGS                    $ rmdir newprogs
C:\GUIDO>RD \PROGS\TURBO                $ rmdir /progs/turbo
```

주의 사항:

- rmdir을 사용할 때 삭제할 디렉토리는 비어있어야 한다. 디렉토리를 그 내용물과 함께 한꺼번에 삭제하려면 rm -rf를 사용한다(위험 감수는 자신이 알아서 할 것)

- '~' 문자는 자신의 홈 디렉토리에 대한 단축 문자다. cd 또는 cd ~ 명령은 자신이 어디에 있든 자신의 홈 디렉토리로 이동시켜준다; cd ~/tmp는 /home/your_home/tmp로 이동시켜준다.
- cd - 는 마지막 cd 명령을 취소 시킨다.

제 5 절 플로피, 하드디스크, 기타

리눅스에서 장치를 관리하는 방법은 DOS 식과 리눅스 식 두 가지가 있다. 좋은 대로 골라서 쓰면 된다.

5.1 DOS 식 디바이스 관리

대부분의 리눅스 배포 판에는 Mtools 수트가 들어 있다. 이는 도스의 해당 명령어를 사용할 수 있게 해주는 것인데 다만 앞에 'm'이 붙는다. 즉 mformat, mdir, mdel, mmd, 기타등등. 이들은 긴 파일 이름도 보존할 수 있는데 파일 permission은 보존되지 않는다. /etc/mtools.conf 라는 파일을 편집하여 Mtools을 설정할 수 있는데(배포 판에 샘플이 들어 있다) DOS/Win 파티션과 CD–ROM, Zip 드라이브도 액세스할 수 있다. 하지만 새 디스크를 포맷하는 것은 mformat 명령으로는 안 된다. 먼저 루트 권한으로 다음의 명령을 선언해야 한다. fdformat /dev/fd0H1440.

less a:file.txt와 같은 방식의 명령으로 플로피에 있는 파일을 액세스 할 수 없다. 이는 DOS 방식으로 디스크를 액세스할 때 약점이다.

5.2 유닉스 방식의 디바이스 관리

유닉스에서는 다른 방식으로 디바이스를 관리한다. 유닉스에서는 A: 또는 C:와 같이 볼륨이 분리되어 있지 않다. 플로피든 뭐든 하나의 디스크는 "mounting"이란 오퍼레이션을 통해서 로컬 파일 시스템의 일부가 된다. 디스크 사용이 끝나면 제거하기 전에 먼저 "unmount" 시켜야 한다. 디스크를 물리적으로 포맷하는 것과 그 위에 파일 시스템을 만드는 것은 별개의 일이다. 도스 명령어인 'FORMAT A:'는 두 가지를 한꺼번에 한다. 하지만 리눅스에서는 개별적인 명령으로 처리한다. 플로피 포맷에 대해서는 위를 참고하고; 파일 시스템을 만드는 것은 다음과 같이 하면 된다.

```
# mkfs -t ext2 -c /dev/fd0H1440
```

ext2 대신 도스, vfat(추천) 또는 그 밖의 포맷을 사용할 수 있다. 디스크가 준비된 다음에는 다음의 명령을 통해서 마운트 한다.

```
# mount -t ext2 /dev/fd0 /mnt
```

위에서 ext2를 사용하지 않을 때에는 해당 파일 시스템을 지정하면 된다. 이제 A:나 B: 대신 /mnt를 사용해서 플로피에 있는 파일들을 어드레스 할 수 있게 되었다.

예:

```
DOS                                Linux
---------------------------------------------------------------------

C:\GUIDO>DIR A:                    $ ls /mnt
C:\GUIDO>COPY A:*.*                $ cp /mnt/* .
C:\GUIDO>COPY *.ZIP A:             $ cp *.zip /mnt
C:\GUIDO>EDIT A:FILE.TXT           $ jstar /mnt/file.txt
C:\GUIDO>A:                        $ cd /mnt
A:> _                              /mnt/$ _
```

작업을 마친 다음 디스크를 꺼내기 전에 다음의 명령을 사용해서 언마운트 시켜야 한다.

```
# umount /mnt
```

이미 사용하던 디스크가 아니라 포맷되지 않은 디스크만 fdformat과 mkfs 해야 됨은 물론이다. B: 드라이브를 사용하고 싶으면 위 예제에서 fd0H1440과 fd0 대신 fd1H1440과 fd1을 사용하면 된다.

플로피에 적용되는 것이 다른 디바이스에도 적용됨은 말할 필요도 없다. 예를 들어, 하드디스크나 CD–ROM 드라이브를 추가로 마운트 하고 싶다고 가정하자. CD–ROM은 다음과 같이 마운트 한다.

```
# mount -t iso9660 /dev/cdrom /mnt
```

이 방법이 디스크를 마운트 하는 "공식적인" 방법이었다. 하지만 흔히 사용하는 트릭이 있다. 플로피나 CD–ROM을 마운트 하기 위해서 루트가 되어야 하는 것은 좀 귀찮은 일이므로 다음과 같이 하면 모든 사용자들이 그것들을 마운트 할 수 있다.

o 루트로 로그인 하여 다음과 같이 한다:

```
# mkdir /mnt/floppy ; mkdir /mnt/cdrom
# chmod 777 /mnt/floppy /mnt/cd*
# # make sure that the CD-ROM device is right
# chmod 666 /dev/hdb ; chmod 666 /dev/fd*
```

o /etc/fstab에 다음의 행을 추가한다.

```
/dev/cdrom      /mnt/cdrom  iso9660 ro,user,noauto          0       0
/dev/fd0        /mnt/floppy vfat    user,noauto             0       0
```

이제 도스 플로피나 CD–ROM을 다음과 같은 방법으로 마운트 한다:

```
$ mount /mnt/floppy
$ mount /mnt/cdrom
```

이제 모든 사용자가 /mnt/floppy와 /mnt/cdrom을 사용할 수 있게 되었다. 이런 방법을 통해서 모든 사용자에게 디스크 마운트를 허락하는 것은 보안 체계에 구멍을 내는 것이라는 사실을 잊지 말도록.

마운트 된 파일 시스템에 대한 정보를 얻는 df와 해당 디렉토리가 차지한 용량에 대해서 알려주는 du dirname 두 개의 명령을 알아두면 편리하다.

5.3 백업

여러 가지 패키지가 있지만 멀티 볼륨 백업을 하는데 최소한의 것은(루트로서):

```
# tar -M -cvf /dev/fd0H1440 dir_to_backup/
```

반드시 드라이브에 포맷된 플로피가 들어 있어야 하며 추가로 여러 장 대기시켜 두어야 한다. 복구할 때에는 첫번째 플로피를 드라이브에 넣고 다음을 입력한다.

```
# tar -M -xpvf /dev/fd0H1440
```

제 6 절 윈도우 없어요?

윈도우에 해당되는 것이 바로 그래픽 시스템인 X 윈도우 시스템이다. MS윈도우나 맥과는 달리 X11은 사용의 편리함이나 멋을 염두에 둔 것이 아니라 단지 유닉스 웍스테이션에 그래픽 기능을 제공하도록 디자인되어있다.

주요 차이점은 다음과 같다.

- MS윈도우는 세계 어디를 가나 모양과 사용 방법이 똑 같은 반면, X11은 그렇지 않다: 많은 것을 사용자가 설정할 수 있다. X11의 전체 적인 모습은 "윈도우 매니저"라는 핵심 콤포넌트에 의해서 정해지는데 여기에는 선택할 수 있는 것들이 아주 많이 있다. 기본적이지만 아주 메모리 효율적인 fvwm을 비롯해서 fvwm2-95, Afterstep, WindowMaker, Enlightenment, 이 밖에도 여러 개가 있다. w.m.은 일반적으로 .xinitrc 에서 불러올 수 있다.
- w.m.은 창이 윈도우에서 작동하는 것처럼 설정할 수도 있다. 즉 마우스로 클릭 했을 때 포오그라운드가 되도록. 다른 선택할 수 있는 것으로는 마우스가 해당 창 위로("포커스") 되었을 때 포오그라운드가 되게 하는 것이다. 또, 화면에서 창의 위치를 자동 또는 대화형으로 설정할 수 있다. 프로그램 대신 이상한 프레임이 나타나면 그것이 나타나게 만들고 싶은 장소에 왼쪽 클릭하면 된다.
- 하나 또는 몇 개의 설정 파일을 편집함으로써 대부분의 기능들을 자신이 원하는 방식으로 맞출 수 있다. w.m.의 도큐멘테이션을 읽어보라. 설정 파일은 .fvwmrc, .fvwm2rc95, .steprc, 등이다. 보통 샘플 설정 파일이 /etc/X11/window-manager-name/system.window-manager-name; 에 들어 있다.
- X11 애플리케이션들은 특정 라이브러리("widget sets")를 사용하여 만든다. 위젯 세트가 여러 개 있으므로 애플리케이션들의 모양이 서로 다르다. 가장 기본적인 것은 Athena widget(2–D 모양; xdvi, xman, xcalc)을 사용하는 것들이다. 그 밖에 Motif (netscape)를 사용하는 것도 있고 아직도 Tcl/Tk, XForms, Qt, Gtk 를 사용하는 것도 있다. 그래도 이 라이브러리들은 윈도우즈 처럼 거의 같은 모양과 느낌을 제공한다.
- 산만한 느낌이 들 수도 있다. 예를 들어, 마우스를 사용하여 텍스트 한 줄을 선택한 다음 <BACKSPACE>를 누르면 아마 그 줄 전체가 없어지기를 기대할 것이다. Athena 기반 애플에서는 그렇게 작동하지 않는다. 하지만 다른 위젯 세트로 만들어진 것에서는 대체로 그렇게 작동한다.
- 스크롤바와 창의 크기 조절 방식은 w.m.과 위젯 세트에 달려있다. 팁: 스크롤바가 기대하는 것처럼 행동하지 않으면 가운데 단추나 두 개의 단추를 눌러서 움직여보라.
- 애플리케이션의 아이콘은 디폴트로 주어지는 것이 아니다. 하지만 있는 것들도 많다. 대부분의 w.m.은 데스크탑("루트 창")을 클릭함으로써 호출되는 메뉴 기능을 갖고 있다. 이 메뉴를 사용자 설정할 수 있음은 물론이다. 루트 창의 모양을 바꾸려면 xsetroot또는 xloadimage를 사용하면 된다.
- 클립보드는 텍스트에만 해당되며 이상하게 작동한다. 일단 텍스트를 선택하면, 이미 자동으로 클립보드에 복사된다. 아무데로나 움직여서 가운데 버튼을 누르면 붙여 넣기가 된다. xclipboard라 불리는 애플리케이션이 있는데 복수의 클립보드 버퍼를 제공한다.
- 드랙앤드롭은 옵션으로 X11 애플리케이션과 드랙앤드롭을 지원하는 w.m.을 사용해야만 된다.

메모리를 절약하기 위해서 동일한 라이브러리를 사용하는 애플리케이션을 사용하는 것이 좋은데 현실적으로는 조금 어렵다.

X11을 윈도우즈처럼 같은 모양과 느낌으로 작동하게 만드는 것을 목표로 하는 프로젝트가 있다. Gnome, *www.gnome.org* `<http://www.gnome.org>` and KDE, *www.kde.org* `<http://www.kde.org>`,는 정말 훌륭하다. 한번 사용해보라. 윈도우즈 데스크탑을 더 이상 그리워하지 않을 것이다.

제 7 절 취향에 따라 시스템 맞추기

7.1 시스템 설치 파일들

도스에는 중요한 2개의 파일로 AUTOEXEC.BAT과 CONFIG.SYS가 있다. 이들은 부트 할 때 시스템을 초기 설정하고, PAHT나 FILES와 같은 몇 개의 환경변수를 정하며 프로그램이나 배치 파일을 시동하기도 한다. 리눅스에서 초기 설정 파일이 아주 많이 있으며 그 중 어떤 것들은 아주 잘 알기 전에는 건드리지 않는 것이 좋다. 어쨌거나 제일 중요한 것들에 대해서 알려주겠다.

FILES	NOTES
/etc/inittab	지금은 건드리지 말 것!
/etc/rc.d/*	상동

PATH와 그 밖의 환경 변수를 설정하거나 로그인 메시지를 변경하거나 로그인 뒤에 자동으로 프로그램이 시동되는 정도만이 필요할 때는 다음의 파일들을 살펴본다.

FILES	NOTES	
/etc/issue	로그인 이전 메시지 설정	
/etc/motd	로그인 이후 메시지 설정	
/etc/profile	$PATH와 그 밖의 변수 등 설정	
/etc/bashrc	alias와 함수 등 설정	
/home/your_home/.bashrc	자신의 alias와 함수 설정	/home/your_home/.bash_profile 또는
/home/your_home/.profile	환경 설정과 프로그램 시동	

후자의 파일(숨김 파일이 아닐 것)이 존재하면 로그인 후에 읽혀진 다음 명령이 실행될 것이다.

예—다음의 .bash_profile을 보라.

```
# 나는 코멘트이다.
echo Environment;  printenv | less      # 도스의 명령어 세트에 해당된다
alias d='ls -l' # alias가 무엇인지 쉽게 이해할 수 있을 것이다
alias up='cd ..'
echo "경로는"$PATH  echo "이다"
echo "오늘은 'date'이다" # 'date' 명령의 출력을 사용한다
echo " 좋은 하루, " $LOGHAME
# 다음은 "쉘 기능"이다
ctgz() # .tar.gz 아카이브의 내용을 보여준다
{
        for file in $*
        do
                gzip -dc $(file) | tar tf -
        done
}
  # .profile의 끝
```

$PATH와 $LOGNAME는 예상 할 수 있듯이 환경 변수이다. 이 밖에도 건드릴 수 있는 환경변수 들이 많이 있다. less나 bash 같은 애플리케이션을 위한 RMP가 좋은 예이다.

다음의 행을 /etc/profile에 추가하면 PROMPT PG와 거의 같은 효과를 제공한다.

```
export PS1="\w\\$"
```

7.2 프로그램 초기 설정 파일

리눅스에서는 거의 모든 것들을 필요한 대로 원하는 대로 설정할 수 있다. 대부분의 프로그램이 하나 또는 그 이상의 변경 가능한 초기 설정 파일을 갖고 있다. 이들은 홈 디렉토리의 .prognamerc 형태로 존재하는 경우가 많다. 제일 먼저 수정하고 싶어할 만한 것들은 다음과 같다.

- .inputrc: bash가 사용하는 파일로 키 바인딩을 정의한다.
- .xinitrc: startx가 사용하는 파일로 X 윈도우 시스템을 초기 설정한다.
- .fvwmrc: 윈도우 매니저인 fvwm이 사용한다.
- .joerc, .jstarrc: 에디터인 joe가 사용한다.
- .jedrc: 에디터인 jed가 사용한다.
- .pinerc: 메일 리더인 pine이 사용한다.
- .Xdefault: 여러 X 프로그램들이 사용한다.

조만간 이 모든 것들과 그 밖의 것들을 만날 것이다. RMP. 설정 하우투인 *sunsite.unc.edu/mdw/HOWTO/Config-HOWTO.html* `<http://sunsite.unc.edu/mdw/HOWTO/Config-HOWTO.html>`? 를 읽어보면 보다 잘 알 수 있다.

제 8 절 네트워킹 개념

리눅스에서는 "전화 접속 네트워킹"이 가능하기만 한 것이 아니라 훨씬 더 안정적이고 빠르다. 이름은 바로 "PPP"로서 모뎀을 이용해서 인터넷에 연결되는데 사용하는 프로토콜이다. 이를 위해선 PPP를 지원하는 커널과 전화를 걸고 연결하는 툴이 필요하다.

인터넷 서비스 제공자의 서버에서 메일을 추출하기 위해서는 POP 프로토콜을 사용하며 "email fetcher 이메일 수신기"라고 불리는 툴이 필요하다. 메일을 가져오면 리눅스 박스에 전달된 형태 그대로 나타난다. 그러면 pine이나 mutt, elm과 그 밖의 여러 MUA(Mail User Agent)를 사용해서 관리하면 된다.

윈도우즈에서는 인터넷 애플리케이션을 시동하면 자동으로 다이얼러가 불려지는 반면 리눅스에서는 다른 순서를 사용해야 한다. 먼저 전화를 건 다음 애플리케이션을 시동한다. diald라 불리는 것의 행동 양식은 보편적이다. 전화 접속 네트워킹을 설치하고 설정하는 것은 리눅스에서 가장 어려운 작업 중 하나였지만 이제는 그렇지 않다. Configuration HOWTO를 참고하라.

마지막으로 "Network neighborhoo"에 대해서 한 마디 하겠다. 리눅스 웍스테이션을 Windows NT/9X의 로컬 네트웍에 있는 윈도우즈 머신처럼 보이게 만들 수 있다! 바로 Samba를 사용하면 된다. 하하 Samba란 브라질 사람들의 자유 분방한 춤이 아니라 리눅스용 SMB 프로토콜이다. *samba.anu.edu.au/samba* `<http://samba.anu.edu.au/samba>`에 가보도록.

제 9 절 약간의 프로그래밍

9.1 쉘 스크립트: 흥분제 먹은 .BAT 파일

긴 명령줄을 간단하게 쓰기 위해서 .BAT를 사용해본 적이 있으면(나는 아주 많다), profile이나 .bash_profile에 적절한 alias 행을 삽입함으로써 같은 목적이 얻어질 수 있다. 하지만 좀 더 복잡한 .BAT에 대해서는 쉘에서 사용 가능한 스크립트 언어를 아주 좋아할 것이다. 바로 강력한 QBasic이 그 중 하나다. 여기엔 변수와 while, for, case if... then... else같은 구조를 비롯하여 여러 가지 기능들이 있다: '진짜' 프로그래밍 언어의 좋은 대안이 될 수 있는 것이다.

스크립트—도스의 .BAT 파일에 해당하는 것—를 작성하는 것은 단지 인스트럭션들로 된 표준 ASCII 파일을 작성하한 다음 저장하고 chmod +x <스크립트파일>을 사용하여 실행 가능하게 만들어주면 된다. 실행하는 것은 이름을 입력하면 된다.

주의 할 점 한가지. vi라 불리는 시스템 편집기가 있는데 내 경험에 의하면 처음 사용하는 사람들은 매우 어려워한다. vi 편집기의 사용법에 대해서는 설명하지 않겠다. Matt Welsh의 책이나 인터넷에서 설명서를 찾아볼 것을 권한다. 여기서는 다음과 같은 것들을 언급하는 것으로 충분할 것 같다.

- 텍스트를 삽입하려면 i를 누르고 텍스트를 입력한다.
- 문자를 지우려면 <ESC>를 누른 다음 x를 누른다;
- 저장하지 않고 vi를 종료하려면 <ESC>를 누른 다음 :q!를 누른다.
- 저장하고 종료하려면<ESC>를 누른 다음 :wq.를 누른다.

초보자를 위한 좋은 편집기는 joe이다: jstar를 입력해서 joe를 시작하면 DOSWin의 편집기와 똑 같은 키 바인딩을 사용할 수 있다. WordStar의 jed 또는 IDE는 더욱 좋다. "Where to Find Applicaions"섹션을 참조하면 어디에서 이들 편집기를 구할 수 있는지 나와있다.

bash의 스크립트를 작성하는 것은 매우 큰 주제이기 때문에 그 자체만으로도 책 한 권이 된다. 그래서 더 깊이 설명하지는 않겠다. 단지 기본적인 규칙을 깨달을 수 있는 쉘 스크립트 예제 몇 개를 보여주겠다:

```
#!/bin/sh
# sample.sh
# 나는 코멘트(주석)
# 첫번째 줄은 바꾸지 말 것 거기에 있어야 함
echo "이 시스템은: 'uname -a'" # 명령의 출력을 사용함
echo "내 이름은 $0" # 내장 변수
echo "당신은 다음을 주었음 $# 매개변수: "$*
echo "첫 번째 매개변수는: "$1
echo -n "이름은?" ; read your_name
echo 차이점을 잘 보자: "안녕 $your_name" # "로 인용문 넣기
echo 차이점을 잘 보자: '안녕 $your_name' # '로 인용문 넣기
DIRS=0 ; FILES=0
for file in 'ls .' ; do
        if [ -d ${file} ] ; then # file이 디렉토리라면
                DIRS='exr $DIRS + 1' # DIRS = DIRS + 1
        elif [ -f ${file} ] ; then
                FILES='expr $FILES +1'
        fi      case ${file} in
                *.gif|*jpg) echo "${file}: graphic file" ;;
                *.txt|*.tex) echo "${file}: text file" ;;
                *.c|*.f|*.for) echo "${file}: source file" ;;
                *) echo "${file}: genric file" ;;
        esac
done
echo " ${DIRS} 디렉토리와 ${FILES}가 존재한다"
ls | grep "ZxY--!!!WKW"
if [ $? != 0 ] ; then # 마지막 코드 종료
echo "ZxY--!!!WKW 못 찾음"
fi
echo "이상... 추가 정보가 필요하면 'man bash'를 입력할 것."
```

9.2 C 혼자 배우기

유닉스에서는 좋건 싫건 시스템 언어가 C다. 여러 다른 언어들(Java, FORTRAN, Pascal, Lisp, Basic, Perl awk...)도 사용할 수 있다.

C를 안다는 것을 가정하고 터보 C++ 등 도스용에 버릇이 나빠진 사람들을 위한 가이드를 몇 개 알려주겠다. 리눅스의 C 컴파일러는 gcc라고 불리며 DOS 용에 있는 수 많은 벨과 휘슬들이 없다. 다시 말해서 IDE, 온라인 도움말, 통합된 디버거 등이 없다. 그냥 거친 명령행 컴파일러일 뿐이며 아주 강력하고 효율적이다. 누구나 다 배우는 표준 hello.c를 컴파일 하려면 다음과 같이 입력한다.

```
$ gcc hello.c
```

그러면 실행 가능 파일인 a.out이 만들어진다. 그 실행 파일에 다른 이름을 주려면 다음과 같이 입력한다.

```
$ gcc -o hola hello.c
```

프로그램에 라이브러리를 링크하려면 -l<libname>를 스위치로 붙이면 된다. 예를 들어, math 라이브러리를 링크하려면 다음과 같이 한다:

```
$ gcc -o mathprog mathprog.c -lm
```

(-l<libname> 스위치는 gcc로 하여금 라이브러리인 /usr/lib/lib<libname> 를 링크하게 만든다. so; so -lm은 /usr/lib/libm.so를 링크한다).

이제까지는 순조로웠다. 하지만 프로그램이 여러 개의 소스 파일로 만들어졌을 때는 유틸리티인 make를 사용해야 된다. 표현 해석기를 작성했다고 가정하자. 그 소스 파일을 parser.c 라고 부르고 parser.h와 calc.c 두 개의 헤더 파일을 #include 시킨다. 그런 다음 calc.c라고 하는 프로그램에서 parser.c에 있는 루틴을 사용하고 싶다고 하자. 그러면 순차적으로 parser.h가 #include 된다. 얼마나 복잡한가! calc.c를 컴파일 하려면 어떻게 해야 한단 말인가?

소위 Makefile이라고 풀리는 것을 작성해야 한다. 이는 컴파일러에게 소스들과 객체 파일들의 종속성을 알려준다.

예제에서:

```
# 이것은 Makefile로 calc.c를 컴파일 하는데 사용된다.
# 지시된 곳에서 <TAB> 키를 누른다!

calc: calc.o parser.o
<TAB>gcc -o calc calc.o parser.o -lm
# calc는 다음 두 개의 객체 파일에 의존한다:

calc.o, parser.ocalc.o: calc.c parser.h
<TAB>gcc -c calc.c
# calc.o는 두 소스 파일에 의존한다

parser.o: parser.c parser.h xh.h
<TAB>gcc -c parser.c
# parser.o는 세 개의 소스 파일에 의존한다.

# Makefile의 끝
```

이 파일을 Makefile로 저장하고 make를 입력하여 프로그램을 컴파일 한다. 다른 방법으로 calc.mak로 저장한 다음 make -f calc.mak 라고 입력한다. RMP는 두말 할 필요도 없다. 다음과 같이 하면 man 페이지 섹션 3에 있는 C 기능에 대한 약간의 도움말을 불러올 수 있다.

```
$ man 3 printf
```

프로그램을 디버깅하려면 gdb를 사용한다. gdb의 사용법을 알고 싶으면 info gdb라고 입력한다. 사용 가능한 라이브러리는 많이 있다. 그 중 제일 먼저 사용하고 싶은 것은 ncurse(텍스트 모드 효과)와 svgalib(콘솔 그래픽스)일 것이다. 한번 과감하게 X11 프로그래밍을 해보고 싶으면(그렇게 어려운 것은 아니다) X11 프로그래밍을 아주 쉽게 하도록 도와주는 라이브러리가 여럿 있다. Gtk가 리눅스의 표준이 되어가고 있다는 사실을 염두에 두고 *www.xnet.com/˜blatura/linapp6.html* `<http://www.xnet.com/~blatura/linapp6.html>`를 살펴보라.

많은 에디터들이 IDE 역할을 할 수 있다. 그 예로 emacs와 jed가 있는데 신택스 하이라이트, 자동 들여쓰기 등의 기능이 있다. 다른 것으로는 *sunsite.unc.edu:/pub/Linux/devel/debuggers/* `<ftp://sunsite.unc.edu:/pub/Linux/devel/debuggers/>` 에서 rhide 패키지를 가져온다. 이것은 Borland IDE 클론으로 아마 좋아할 것이다.

제 10 절 나머지 1%

사실 1% 이상이다.

10.1 tar와 gzip의 사용

유닉스에서는 파일들을 한데 모으고 저장하는데 널리 사용되는 애플리케이션이 있다. tar는 파일들을 한데 모으는데 사용한다—tar는 마치 PKZIP이나 Winzip 에서 압축 기능이 빠진 것과 같다, 파일들을 한데 모으는 기능만 한다. 새로운 저장 파일을 만들려면 다음과 같이 입력한다.

```
$ tar =cvf <archivve_name.tar> <file> [file...]
```

저장 파일로부터 파일을 추출하려면:

```
$ tar -xpvf <archive_name.tar> [file...]
```

저장 파일의 내용을 보려면:

```
$ tar -tf <archive_name.tar> | less
```

compress를 사용하면 파일을 압축할 수 있다. 하지만 compress는 이제 한물 가서 더 이상 쓰이지 않으므로 qzip을 사용해야 한다.

```
$ compress <file>
$ gzip <file>
```

이렇게 하면 확장자가 .Z인 파일(compress를 썼을 때)이나 .gz인 파일(gzip을 썼을 때)이 만들어진다. 이 프로그램들은 한번에 하나의 파일만 압축할 수 있다. 압축을 해제 할 때는:

```
$ compress -d <file.Z>
$ gzip -d <file.gz>
```

RMP.

10.2 애플리케이션 설치

제일 먼저 : 패키지를 설치하는 것은 루트의 작업이다. 대부분의 리눅스 애플리케이션은 .tar.gz 아카이브 형태로 배포된다. 이 아카이브에는 보통 적당한 이름의 디렉토리가 들어 있으며 그 속에는 또 적절한 이름의 파일과 서브디렉토리가 들어 있다. 이런 패키지를 설치하는 방법은 대체로 /usr/local 에서 다음의 명령을 사용한 다음

```
# tar -zxf <archive.tar.gz>
```

그 다음 README나 INSTALL 파일을 읽으면 된다. 대부분의 경우 애플리케이션은 소스로 배포되어 컴파일 시켜야 한다. make를 입력한 다음 make install을 입력하면 충분한 경우가 흔하다. gcc나 g++ 컴파일러가 있어야 하는 것은 물론이다.

/ 에서 풀어야 하는 아카이브도 있다. Slackware의 .tgz 아카이브가 이런 경우다. 서브디렉토리는 없이 파일만 들어 이는 아카이브도 있다 - 이런 때는 파일들이 뒤 섞이지 않도록 주의해야 한다! 설치하기 전에 항상 내용물을 확인해보는 것이 좋다.

데비안과 레드햇은 각각 .deb와 .rpm의 자체 아카이브 포맷을 갖고 있다. .rpm은 점점 인기를 얻어가고 있는데 rpm 패키지를 설치하려면 다음과 같이 입력하면 된다.

```
# rpm -i package.rpm
```

10.3 꼭 알아 두어야 할 팁

백스크롤:

<SHIFT + PAG UP>를 누르면 위쪽 페이지들로 백스크롤 할 수 있다. 이 때 뒤로 갈 수 있는 페이지양은 비디오 메모리에 따라 결정된다.

화면 재정리:

실수로 이진 파일에 more나 cat 명령을 사용하면 화면에 필요 없는 것들로 가득찬다. 이를 고치려면 화면에 나타나지 않더라도 그냥 'reset'을 입력하거나 echo CTRL-V ESC c RETURN을 차례대로 누르면 된다.

텍스트 붙여 넣기:

콘솔에서 아래를 보라;X 에서는 x터미널 창에 있는 텍스트를 선택하려면 클릭한 다음 드래그하고, 붙여 넣기는 가운데 버튼을(또는 2버튼 마우스를 사용하고 있다면 두 개의 버튼을 동시에) 누르면 된다. xclipboard도 있다(불행히도 텍스트에만 작동되지만); 반응이 느린 것 때문에 혼동하지 않도록.

마우스 사용:

콘솔용 마우스 드라이버인 gpm을 설치했다면 클릭+드래그로 텍스트를 선택하고 오른쪽 클릭으로 선택된 텍스트를 붙여 넣기 할 수 있다. 서로 다른 VC 들 간에 사용할 수 있다.

10.4 애플리케이션 찾기

오랫동안 사용해온 그 믿음직한 DOS/Win 애플리케이션들을 리눅스용으로 대체할 수 있는지 궁금할 것이다. 먼저 리눅스 소프트웨어 저장소들을 뒤져볼 것을 권한다. *sunsite.unc.edu/pub/Linux* `<ftp://sunsite.unc.edu/pub/Linux>`, *tsx-11.mit.edu/pub/linux* `<ftp://tsx-11.mit.edu/pub/linux>`, *ftp.funet.fi/pub/Linux* `<ftp://ftp.funet.fi/pub/Linux>`. 그 밖에 "Linux Applications and Utilities Page" 인 *www.xnet.com/˜blatura/linapps.shtml* `<http://www.xnet.com/~blatura/linapps.shtml>`과 리눅스 "공식" 페이지인 *www.linux.org* `<http://www.linux.org>`를 찾아보는 것도 좋다.

10.5 할 수 없는 몇 가지 것들

DOS/Windows에서는 작업이 귀찮거나 어렵거나 심지어는 불가능한 많은 작업들이 리눅스에서는 가능하다. 자 여기 식욕을 돋울만한 몇 개를 소개해보겠다.

at

는 특정 시간에 지정된 프로그램이 실행되도록 한다.

awk

는 데이터 파일을 조작하는 간단하지만 강력한 언어다. 예를 들어 data.dat를 멀티 필드 데이터 파일이라고 하자.

```
$ awk ’$2   "abc" {print $1, "\t", $4}’ data.dat
```

이 명령은 data.dat에 있는 모든 행 중 두 번째 필드에 "abc"가 들어 있는 행들의 1과 4번 필드를 프린트한다.

cron

은 특정 요일의 특정 시간에 주기적으로 필요한 작업에 쓸모가 있다. man 5 crontab을 입력해보라.

file

<filename>은 filename이 무엇인지 알려준다(ASCII 텍스트, 실행, 아카이브 등);

find

는 가장 강력하고 유용한 명령 중 하나이다("Directories: Translating Commands" 섹션을 참조하라). 이는 해당 문자를 포함한 파일을 찾거나 그 파일에 어떤 명령을 수행하는데 사용한다. find의 일반적인 사용은 다음과 같다.

```
$ find <directory> <expression>
```

여기에서 <expression>에는 검색 규범과 행동이 포함된다. 예:

```
$ find . -type l -exec ls -l {} \;
```

이 명령은 모든 symbolic link를 찾아서 그것이 지시하는 것이 무엇인지 보여준다.

```
$ find / -name ”*.old” -ok rm {} \;
```

이 명령은 지정된 패턴에 해당되는 파일을 모두 찾아서 당신에게 허락을 요구한 다음 지운다.

```
$ find . -perm +111
```

이 명령은 permission이 111인(실행파일) 모든 파일을 찾는다.

```
$ find . -user root
```

이 명령은 root에 속하는 모든 파일을 찾는다. 이 경우 여러 가지 가능성이 있다—RMP.

grep

는 파일 내에서 지정 텍스트 패턴을 찾는다. 예를 들면,

```
$ grep -l "geology" *.tex
```

이 명령은 ’geology’란 단어가 들어가 있는 *.tex 파일들을 나열한다. 이의 변형인 zgrep는 gzip으로 압축된 파일에 실행된다. RMP;

일반 표현들은 매우 복잡하지만 텍스트에 검색 작업을 수행하는 방식이 아주 강력하다. 예를 들어 ˆa[ˆa-m]X{4,}txt$는 ’a’로 시작하고 그 뒤에 a-m 사이에 들어가지 않는 문자로 시작하고 그 뒤에는 ’X’가 4개 이상 들어가고 txt로 끝나는 행을 찾는다. less와 같은 고급 편집기를 비롯하여 다른 많은 애플리케이션에서 일반 표현을 사용한다. man grep 하여 설명을 읽어보라.

script

<script_file>은 종료 명령을 내릴 때까지 화면 내용을 script_file에 덤프 한다. 디버깅을 할 때 유용하다.

sudo

는 root의 작업 중 일부를 사용자들이 수행할 수 있게 해준다(예, 디스크의 포맷과 마운팅; RMP)

uname -a

는 시스템에 대한 정보를 알려준다.

zcat과 zless

는 gzip으로 압축된 파일들을 압축해제하지 않고 브라우징하고 파이핑하는데 유용하다.

예를 들어: $ zless textfile.gz $ zcat textfile.gz | lpr

다음의 명령들도 가끔 편리할 때가 많다. bc, cal, chsh, cmp, cut, fmt, head, hxdump, nl, passwd, printf, sort, split, strings, tac, tail, tee touch, uniq, w, wall, wc, whereis, write, xargs, znew. RMP.

10.6 DOS/Windows에서 유닉스 연습

믿거나 말거나, DOS/Windows에서 유닉스 비슷한 환경을 제공하는 훌륭한 툴들이 많이 있다. 그 중 하나가 Djgpp 수트다 DOS용은 (*www.delorie.com/djgpp* `<http://www.delorie.com/djgpp>`) Win32용은 Cygnus (*www.cygnus.com* `<http://www.cygnus.com>`)다. Cygnus는 제법 복잡한 툴이다. 둘 다 리눅스처럼 GNU 개발툴과 유틸리티가 들어 있다. 하지만 동일한 안정성과 성능을 얻을 수는 없다.

리눅스 맛을 조금 보고 싶으면 Djgpp를 시험해보라. 다음의 파일들을 받아서 설치해본다. (이 글을 쓰고 있는 시점에서 가장 최신 버전은 2.02이다): djdev202.zip, bnu281b.zip, bsh1147b.zip, fil316b.zip, find41b.zip, grep22b.zip, gwk303b.zip, lss332b.zip, shl112b.zip.. 설치 방법이 제공되어 있으며 <news:comp.os.msdos.djgpp> 에서 도움을 얻을 수 있다.

특히 DOSWin에서 bash를 사용하면 상큼한 기분이 들 것이다. 설정을 하려면 함께 제공되는 BOOT.BAT를 편집하여 자신의 설치에 맞게 고친다. 그런 다음 이 파일들을 홈 디렉토리(Windows 파티션의)에 제공된 것들 대신 넣는다.

```
        # 이것은 _bashrc 다
        LS_OPTIONS="-F -s --color=yes"
        alias cp='cp -i'
        alias d='ls -l'
        alias l=less
        alias ls="ls $LS_OPTIONS"
        alias mv='mv -i'        alias rm='rm -i'
        alias u='cd ..'

# 이것은 _bprof 이다
if [ -f ~/_bashrc ]; then
        .~/_bashrc
fi
PS1='\w\$
'PS2='> 'CDPATH="$CDPATH:~"
# less(1)을 위한 것들
LESS="-M-Q"                     # 긴 프람프트, 침묵
LESSEDIT="%E ?lt+%lt. %f"       # 제일 첫 번 행을 편집
```

```
VISUAL="jed"                    # 에디터
LESSCHARSET=latin1              # 강조된 문자들을 시각화
export PS1 PS2 CDPATH LS_OPTIONS LESS LESSEDIT LESSOPEN VISUAL LESSCHARSET
```

10.7 일반적으로 사용하는 확장자와 관련 프로그램

여러 파일 확장자들을 보게 될 것이다. 좀 희귀한 것들(예, 폰트 등)을 제외하고 보편적으로 사용되는 것들이 나열해 보았다.

> 1 ... 8: man 페이지들. groff -Tascii -man <file.1>을 사용하여 읽어보라. arj로 만든 아카이브. TeX로 만든 출력 파일 (아래 참조). xdvi를 사용하면 시각화 할 수 있다. dvips는 PostScript 파일인 .ps 파일로 바꾸어준다. gzip으로 만든 아카이브 info 파일 (man 페이지의 일종의 대치품). 정보를 얻는다. Linux Software Map 파일. 보통의 ASCII 파일로 패키지에 대한 설명이 들어 있다. 포스트스크립트 파일. 이 파일을 보거나 인쇄하려면 gs에 꼭 필요한 것은 아니지만 ghostview나 gv를 사용한다. Red Hat 패키지. 패키지 매니저인 rpm을 사용하여 어느 시스템에나 설치할 수 있다. tar로 한데 모으고 compress로 압축한 아카이브 tar로 한데 모으고 gzip으로 압축한 아카이브 강력한 조판 시스템인 TeX에 보내지는 텍스트 파일. 여러 배포판이 나와 있는 tex 패키지를 구해서 쓴다. texinfo 파일. TeX와 info 파일을 모두 만들 수 있다.(info 참조). texinfo를 구해서 사용한다.o xbm, xpm, xwd: 그래픽 파일 compress로 만든 아카이브

10.8 파일 변환

DOS/Win과 리눅스 사이에 텍스트 파일들을 교환할 필요가 있으면 "end of line" 문제에 주의해야 한다. DOS에서는 텍스트의 모든 행이 CR/LF(즉 ASCII13 + ASCII 10)으로 끝나고 리눅스에서는 LF로 끝난다. DOS 텍스트 파일을 리눅스에서 편집하면 각 행은 이상한 M 모양 비슷한 문자로 끝날 가능성이 높다. 리눅스 텍스트 파일을 DOS에서 보면 문단 구분이 없는 아주 아주 긴 하나의 행으로 나타난다. 도스와 리눅스 사이의 파일 변환을 하는 툴로 dos2unix와 unix2dos 두 개가 있다.

텍스트로만 된 파일에 강조 문자가 들어 있다면 보통 DOS가 아니라 Windows에서(Notepad)로 만들어진 것인지 확인하라. 그렇지 않으면 모든 강조 문자들이 엉망이 될 것이다.

Word나 WordPerfec 파일을 일반 텍스트로 변환하는 것은 좀 다 까다롭지만 가능하다. 이를 위해서는 CTAN 싸이트에서 발견할 수 있는 툴이 필요하다. 그 중 하나가 *ftp.tex.ac.uk* `<ftp://ftp.tex.ac.uk>` 이다. /pub/tex/tools/ 디렉토리에서 word2x 프로그램을 받거나 /pub/tex/support/ 디렉토리에 있는 프로그램 중 하나를 사용해보라. 특히 word2x는 Word 6 파일을 변환 하는데, Word 97 파일의 변환을 위해선 HTML로 변환하는 mswordview (*www.csn.ul.ie/˜caolan/docs/MSWordView.html* `<http://www.csn.ul.ie/~caolan/docs/MSWordView.html>`)가 필요하다.

10.9 자유 오피스 수트

파일 변환만 갖고는 충분하지 않다면 마이크로소프트 오피스 비슷한 패키지를 선택할 수도 있다.

StarOffice 수트는 개인 사용은 자유이다. 이 수트는 크고 속도가 좀 느리긴 하지만 아주 좋다. 마이크로소프트 오피스에는 없는 기능들도 많이 제공된다. 비록 변환이 항상 완벽한 것은 아니지만 Word와 Excel 파일도 읽고 쓸 수 있다. 홈페이지는 : *www.stardivision.com* `<http://www.stardivision.com>`.

제 11 절 일단 여기까지

축하한다! 이제 유닉스를 조금 배웠으니 작업할 준비가 되었다. 리눅스 시스템에 대한 지식은 아직도 부족하고 편안하게 사용하려면 더 연습이 필요하다는 것을 잊지 말라. 하지만 필요한 것이 단지 애플리케이션을 잔뜩 설치하여 그것으로 작업하는 것뿐이라면 여기에 소개된 것만으로도 충분하다.

나는 여러분이 리눅스 사용을 즐기고 더 많은 것을 배울 것이라고 확신한다. 그렇지 않은 사람이 없으니까. 또한 DOSWin!으로 절대로 안 돌아갈 것을 장담한다. 이 글을 읽은 사람들 중 서너 명이라도 내 언어를 이해하고 그들에게 뭔가 좋은 서비스가 되었으면 나는 만족한다.

11.1 저작권

특별히 지시되지 않은 한 리눅스 하우투 문서는 글쓴이에게 저작권이 귀속된다. 리눅스 하우투 문서는 이 저작권 명시가 모든 복사판에 유지되는 한 어떠한 물리적 또는 전자적 매체 형태로 전체 또는 부분적으로 재생산되고 배포될 수 있다. 상업적 재배포도 허락되며 권장된다. 하지만 그러한 배포는 저자에게 알렸으면 좋겠다.

모든 번역, 파생 결과물 또는 다른 리눅스 하우투 문서와 합쳐진 집합 결과물은 이 저작권 명시에 의해서 보호 받아야 한다. 즉, 하우투 문서로부터 파생 결과물을 만들어서 그것의 배포에 대해서 추가의 제한을 할 수는 없다. 이러한 규칙에 대한 예외는 특별한 경우에 허락될 수도 있다. 아래 주소의 하우투 진행 책임자에게 연락하라. 간단히 말해서, 우리는 이 정보를 가능한 한 많은 경로를 통해서 보급되길 바란다. 하지만, 하우투 도큐먼트에 대한 저작권은 유지하고 HOWTO를 재 배포할 때는 통보 받았으면 좋겠다.

질문 할 사항이 있으면 리눅스 하우투 진행 책임자인 Tim Bynum에게 이메일*linux-howto@sunsite.unc.edu* `<mailto:linux-howto@sunsite.unc.edu>`로 연락하라.

11.2 성명서

"From DOS to Linux HOWTO"는 Guido Gonzato*REMOVE_MEguido@ibogeo.df.unibo.it* `<mailto:REMOVE_MEguido@ibogeo.df.unibo.it>`,가 썼다. ("REMOVE_ME"는 제거할 것.) "Linux Installation and Getting Started"의 저자인 Matt Welsh와, "Linux frequently asked questions with answers"의 저자인 Ian Jackson, "Linux"의 저자인 Giuseppe Zanetti, 나에게 이메일로 질문을 보낸 모든 사람들 그리고 특히 우리에게 리눅스를 선사한 Linus Torvalds와 GNU에게 감사한다.

이 문서는 "있는 그대로의 형태로" 제공된다. 능력이 닫는 한 정확하게 쓰려고 많은 노력을 기울였지만 담겨 있는 정보를 사용하는 것은 여러분의 책임 하에서 이다. 이 저작의 사용으로 발생하는 어떠한 손해에 대해서는 나는 책임이 없다.

Feedback은 항상 환영이다. 요구나 제안, 혹은 불만 등이 있으면 주저하지 말고 내게 연락을 하라. 리눅스와 삶을 즐기시길...Guido =8-0