우분투 리눅스

시스템 & 네트워크

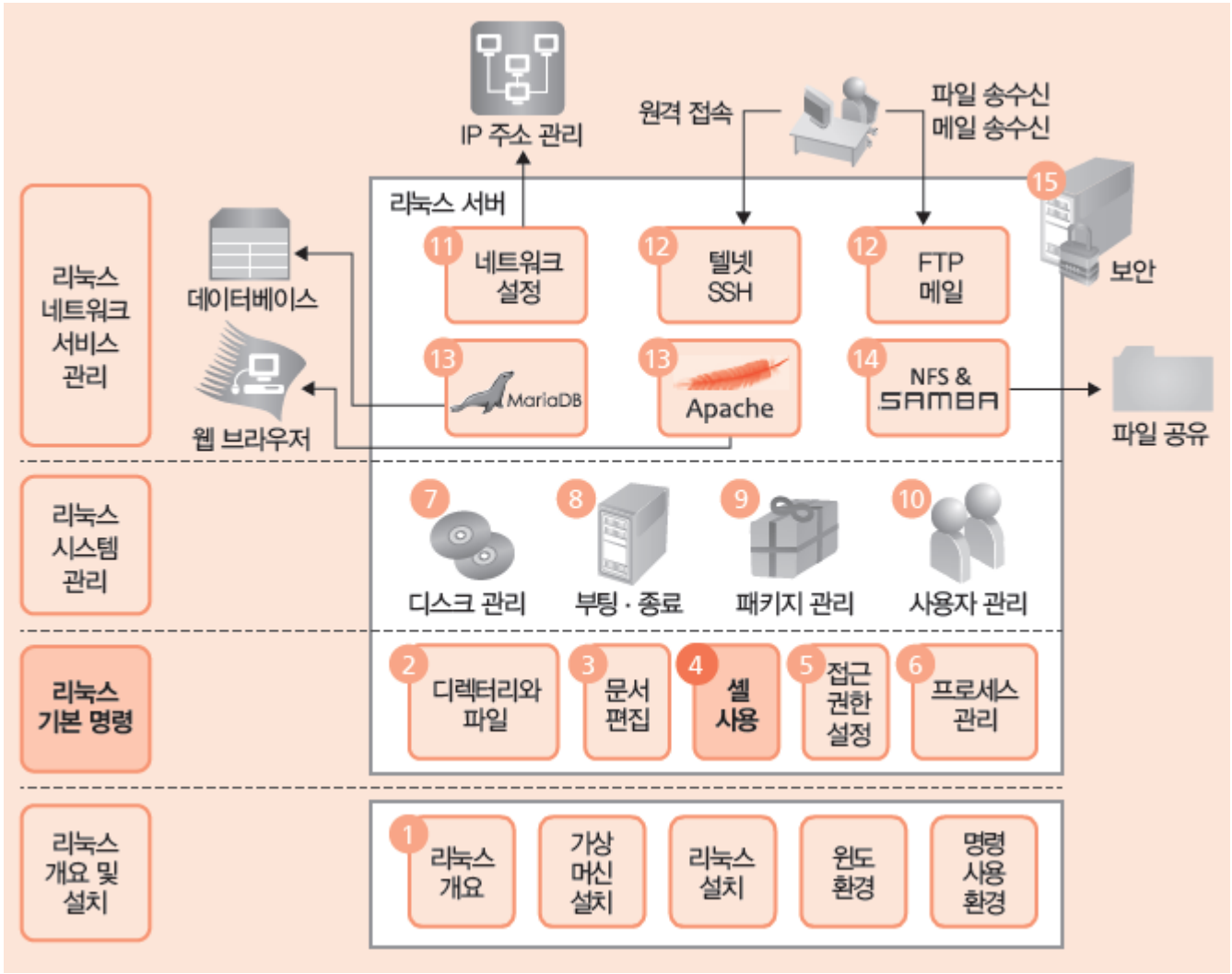
Chapter 04. 셸 사용하기

목차

학습목표

- 셸의 기능을 설명하고 주요 셸의 종류를 나열할 수 있다.
- 로그인 셸을 다른 셸로 바꿀 수 있다.
- 셸 특수문자의 종류를 알고 필요에 따라 적절하게 사용할 수 있다.
- 표준 입출력 장치를 이해하고 입출력 방향 바꾸기를 할 수 있다.
- 셸 변수와 환경 변수의 차이를 이해하고 변수를 정의하여 사용할 수 있다.
- 새로운 에일리어스를 만들거나 필요 없는 에일리어스를 해제할 수 있다.
- 히스토리 기능으로 명령을 재실행할 수 있다.
- 이스케이프 문자를 이해하고 프롬프트를 원하는 형태로 바꿀 수 있다.
- 시스템 환경 설정 파일과 사용자 환경 설정 파일을 구분하고 사용자 환경을 설정할 수 있다.

리눅스 실습 스터디 맵


IP 주소 관리
원격 접속
파일 송수신
메일 송수신
리눅스 서버
15 보안
리눅스 네트워크 서비스 관리
데이터베이스
웹 브라우저
11 네트워크 설정
12 텔넷 SSH
12 FTP 메일
13 MariaDB
13 Apache
14 NFS & SAMBA
파일 공유
리눅스 시스템 관리
7 디스크 관리
8 부팅 · 종료
9 패키지 관리
10 사용자 관리
리눅스 기본 명령
2 디렉터리와 파일
3 문서 편집
4 셸 사용
5 접근 권한 설정
6 프로세스 관리
리눅스 개요 및 설치
1 리눅스 개요
가상 머신 설치
리눅스 설치
윈도 환경
명령 사용 환경

00 개요

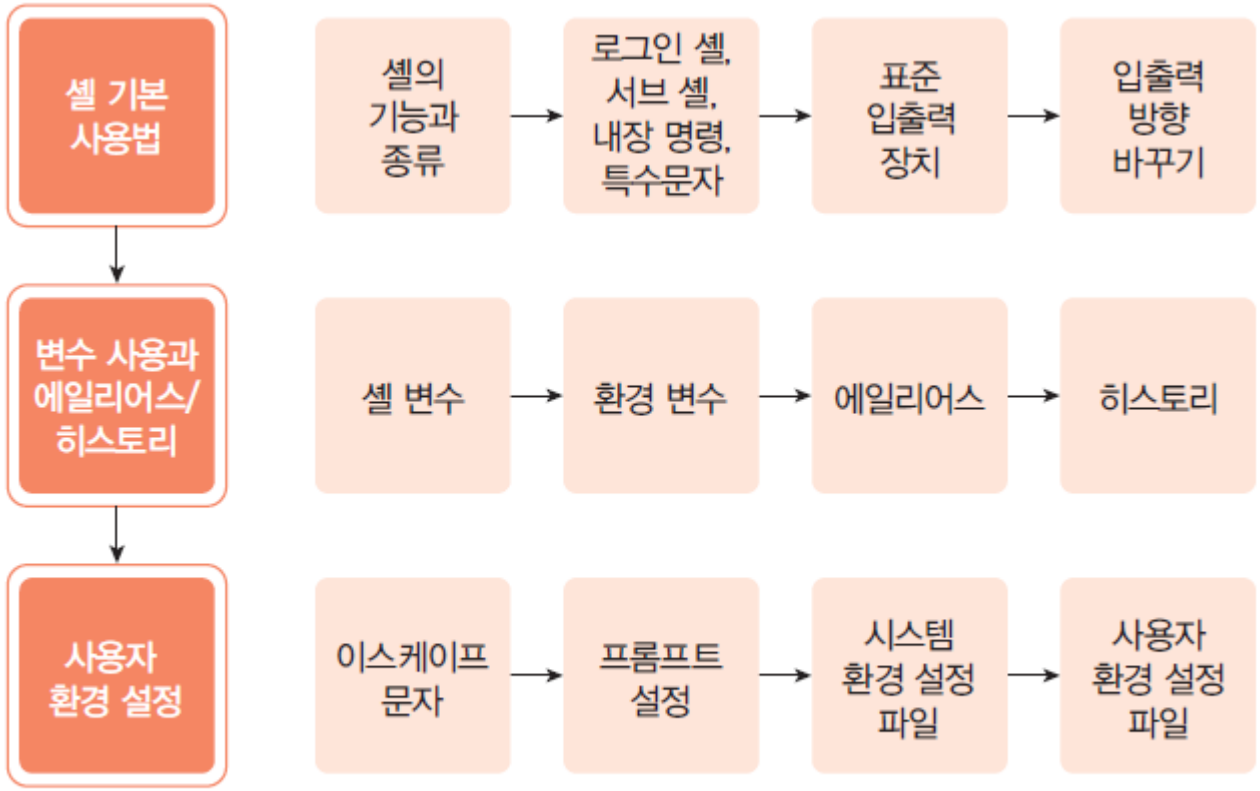


그림 4-1 4장의 내용 구성

01 셸의 기능과 종류

셸의 기능

- 명령어 해석기 기능, 프로그래밍 기능, 사용자 환경 설정 기능

명령어 해석기 기능

- 사용자와 커널 사이에서 명령을 해석하여 전달하는 해석기(interpreter)와 번역기(translator) 기능
- 사용자가 로그인하면 셸이 자동으로 실행되어 사용자가 명령을 입력하기를 기다림 -> 로그인 셸
- 로그인 셸은 /etc/passwd 파일에 사용자별로 지정
- 프롬프트: 셸이 사용자의 명령을 기다리고 있음을 나타내는 표시

프로그래밍 기능

- 셸은 자체 내에 프로그래밍 기능이 있어 반복적으로 수행하는 작업을 하나의 프로그램으로 작성 가능
- 셸 프로그램을 셸 스크립트

사용자 환경 설정 기능

- 사용자 환경을 설정할 수 있도록 초기화 파일 기능을 제공
- 초기화 파일에는 명령을 찾아오는 경로를 설정하거나, 파일과 디렉터리를 새로 생성할 때 기본 권한을 설정하거나, 다양한 환경 변수 등을 설정

01 셸의 기능과 종류

셸의 종류

- 본 셸, 콘 셸, C 셸, 배시 셸, 대시 셸

본 셸(Bourne shell)

- 유닉스 V7에 처음 등장한 최초의 셸
- 개발자의 이름인 스티븐 본(Stephen Bourne)의 이름을 따서 본 셸이라고 함
- 본 셸의 명령 이름은 sh임
- 초기에 본 셸은 단순하고 처리 속도가 빨라서 많이 사용되었고, 지금도 시스템 관리 작업을 수행하는 많은 셸 스크립트는 본 셸을 기반으로 하고 있음
- 히스토리, 에일리어스, 작업 제어 등 사용자의 편의를 위한 기능을 제공하지 못해 이후에 다른 셸들이 등장

C 셸(C shell)

- 캘리포니아대학교(버클리)에서 빌 조이(Bill Joy)가 개발
- 2BSD 유닉스에 포함되어 발표
- 본 셸에는 없던 에일리어스나 히스토리 같은 사용자 편의 기능을 포함
- 셸 스크립트 작성을 위한 구문 형식이 C 언어와 같아 C 셸이라는 이름을 가지게 되었음
- C 셸의 명령 이름은 csh

01 셸의 기능과 종류

콘 셸(Korn shell)

- 1980년대 중반 AT&T 벨연구소의 데이비드 콘(David Korn)이 콘 셸을 개발
- 유닉스 SVR 4에 포함되어 발표
- C 셸과 달리 본 셸과의 호환성을 유지하고 히스토리, 에일리어스 기능 등 C 셸의 특징도 모두 제공하면서 처리 속도도 빠름
- 콘 셸의 명령 이름은 ksh

배시 셸(bash shell)

- 본 셸을 기반으로 개발된 셸로서 1988년 브레인 폭스(Brain Fox)가 개발
- 본 셸과 호환성을 유지하면서 C 셸, 콘 셸의 편리한 기능도 포함
- 배시 셸의 명령 이름은 bash
- 배시 셸의 모든 버전은 GPL 라이선스에 의거하여 자유롭게 사용 가능
- 리눅스의 기본 셸로 제공되고 있어 리눅스 셸로도 많이 알려짐

대시 셸(dash shell)

- 본 셸을 기반으로 개발된 셸로 포직스(POSIX) 표준을 준수하면서 보다 작은 크기로 개발
- 암키스트 셸(ash, Almquist Shell)의 NetBSD 버전으로 1997년 초에 허버트 슈가 리눅스에 이식
- 우분투 6.10부터 본 셸 대신 대시 셸을 사용

```
user1@myubuntu:~$ ls -l /bin/sh
lrwxrwxrwx 1 root root 4 11월  8 23:10 /bin/sh -> dash
user1@myubuntu:~$
```

02 셸 기본 사용법

기본 셸 확인

- 프롬프트 모양 참조
  - 본 셸, 배시 셸, 콘 셸의 기본 프롬프트: $
  - C 셸의 기본 프롬프트: %
- 사용자 정보 확인: /etc/passwd 파일
  - 사용자 정보의 가장 마지막에 나온 /bin/bash가 기본 셸

```
user1@myubuntu:~$ grep user1 /etc/passwd
user1:x:1000:1000:user1,,,:/home/user1:/bin/bash
user1@myubuntu:~$
```

02 셸 기본 사용법

■ 기본 셸 바꾸기

chsh

- **기능** 사용자 로그인 셸을 바꾼다.
- **형식** chsh [옵션] [사용자명]
- **옵션** -s shell: 지정한 셸(절대 경로)로 로그인 셸을 바꾼다.
  -l: /etc/shells 파일에 지정된 셸을 출력한다.
- **사용 예** chsh
  chsh -l
  chsh -s /bin/sh user1

▪ 바꿀 수 있는 셸의 종류: /etc/shells 파일에 지정

```
user1@myubuntu:~$ cat /etc/shells
# /etc/shells: valid login shells
/bin/sh
/bin/dash
/bin/bash
/bin/rbash
user1@myubuntu:~$
```

02 셸 기본 사용법

기본 셸 바꾸기 예

- 바꾸려는 셸은 절대 경로로 지정

```
user1@myubuntu:~$ chsh -s sh user1
암호:                                    → user1 계정의 암호를 입력한다.
chsh: sh is an invalid shell             → 적합한 셸이 아니라는 메시지가 출력된다.
user1@myubuntu:~$ chsh -s /bin/sh user1
암호:
user1@myubuntu:~$
```

로그인 셸과 서브 셸

- 프롬프트에서 다른 셸을 실행할 수 있는데 이를 서브 셸이라 함
- 서브 셸은 또 다른 서브 셸 생성 가능
- 서브 셸을 종료하는 명령은 ^d(+d), exit 등 사용
- 서브 셸이 종료되면 서브 셸을 실행했던 이전 셸 환경으로 복귀
- 로그인 셸에서 로그아웃하면 접속 해제

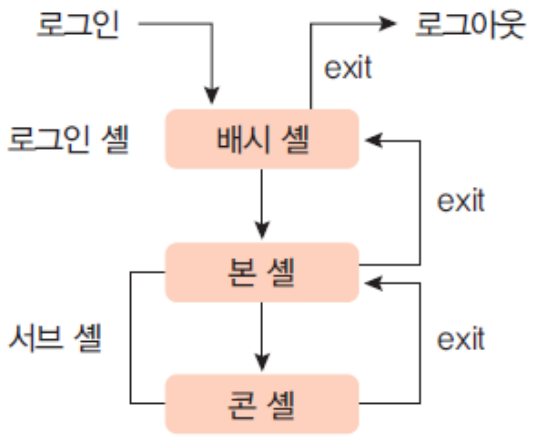


그림 4-2 로그인 셸과 서브 셸

02 셸 기본 사용법

셸 내장 명령

- 셸은 자체적으로 내장 명령을 가지고 있음
- 셸 내장 명령은 별도의 실행 파일이 없고 셸 안에 포함
  - 셸 명령 예: cd

- 일반 명령(실행 파일)의 경우
  - 실행 파일은 바이너리 파일이므로 cat 명령으로 파일의 내용을 확인할 수 없음

```
user1@myubuntu:~$ file /bin/pwd
/bin/pwd: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically
linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0, BuildID[sha1
]=4fa7f0113eb324b15d32009b852ae66b071d4902, stripped
user1@myubuntu:~$ file /usr/bin/mawk
/usr/bin/mawk: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically
linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.24, BuildID[sha
1]=4e43393facf439efde422d29048db342528974ad, stripped
user1@myubuntu:~$
```

02 셸 기본 사용법

배시 셸의 출력 명령

- echo

echo

- 기능 화면에 한 줄의 문자열을 출력한다.
- 형식 echo [-n] [문자열]
- 옵션 -n: 마지막에 줄 바꿈을 하지 않는다.
- 사용 예 echo
  echo text
  echo -n text

```
user1@myubuntu:~$ echo linux
linux
user1@myubuntu:~$ echo "linux ubuntu"
linux ubuntu
user1@myubuntu:~$
```

02 셸 기본 사용법

배시 셸의 출력 명령

- printf
  - % 지시자와 ₩ 문자를 이용하여 출력 형식을 지정 가능

printf

- 기능 자료를 형식화하여 화면에 출력한다.
- 형식 printf [옵션] [인수]
- 옵션 %d, \n 등 C 언어 printf 함수의 형식을 지정한다.
- 사용 예 printf text
  printf "text\n"
  printf "%d\n" 100

```
user1@myubuntu:~$ printf linux
linuxuser1@myubuntu:~$ printf "linux ubuntu \n"
linux ubuntu
user1@myubuntu:~$ printf "%d + %d = %d\n" 10 10 20
10 + 10 = 20
```

02 셸 기본 사용법

특수문자 사용하기

- 사용자가 더욱 편리하게 명령을 입력하고 실행할 수 있도록 다양한 특수문자를 제공
- 주요 특수문자는 *, ?, |, ;, [ ], ~, ' ', " ", ` `` 등
- 명령을 입력하면 셸은 먼저 특수문자가 있는지 확인하고 이를 적절한 형태로 변경한 후 명령을 실행

특수문자 *(별표)

- 임의의 문자열을 나타내는 특수문자로 0개 이상의 문자로 대체

표 4-1 특수문자 *

사용 예	의미
ls *	현재 디렉터리의 모든 파일과 서브 디렉터리를 나열한다. 서브 디렉터리의 내용도 출력한다.
cp * /tmp	현재 디렉터리의 모든 파일을 /tmp 디렉터리 아래로 복사한다.
ls –F t*	t, tmp, temp와 같이 파일명이 t로 시작하는 모든 파일의 이름과 파일 종류를 출력한다. t도 해당한다는 데 주의한다.
cp *.txt ../ch3	확장자가 txt인 모든 파일을 상위 디렉터리 아래의 ch3 디렉터리로 복사한다.
ls –l h*d	파일명이 h로 시작하고 d로 끝나는 모든 파일의 상세 정보를 출력한다. hd, had, hard, h12345d 등 이 조건에 맞는 모든 파일의 정보를 볼 수 있다.

02 셸 기본 사용법

특수문자 ?와 [ ]

- 하나의 문자를 나타내는 데 사용
- ?는 길이가 1인 임의의 한 문자를, [ ]는 괄호 안에 포함된 문자 중 하나를 나타냄

표 4-2 특수문자 ?와 [ ]

사용 예	의미
ls t?.txt	t 다음에 임의의 한 문자가 오고 파일의 확장자가 txt인 모든 파일의 이름을 출력한다. t1.txt, t2.txt, ta.txt 등이 해당된다. 단, t.txt는 제외한다.
ls -l tmp[135].txt	tmp 다음에 1, 3, 5 중 하나가 오고 파일의 확장자가 txt인 모든 파일의 이름을 출력한다. tmp1.txt, tmp3.txt, tmp5.txt 파일이 있으면 해당 파일의 상세 정보를 출력한다. 단, tmp.txt는 제외한다.
ls -l tmp[1-3].txt	[1-3]은 1부터 3까지의 범위를 의미한다. 따라서 ls -l tmp[123].txt와 결과가 같다. 즉 tmp1.txt, tmp2.txt, tmp3.txt 파일이 있으면 해당 파일의 상세 정보를 출력한다.
ls [0-9]*	파일명이 숫자로 시작하는 모든 파일의 목록을 출력한다.
ls [A-Za-z]*[0-9]	파일명이 영문자로 시작하고 숫자로 끝나는 모든 파일의 목록을 출력한다.

02 셸 기본 사용법

특수문자 ~와 –

- ~(물결표)와 -(붙임표)는 디렉터리를 나타내는 특수문자
- ~만 사용하면 현재 작업 중인 사용자의 홈 디렉터리를 표시하고 다른 사용자의 로그인 ID와 함께 사용하면(~로그인 ID) 해당 사용자의 홈 디렉터리 표시
- -는 cd 명령으로 디렉터리를 이전하기 직전의 작업 디렉터리 표시

표 4-3 특수문자 ~와 –

사용 예	의미
cp *.txt ~/ch3	확장자가 txt인 모든 파일을 현재 작업 중인 사용자의 홈 디렉터리 아래 tmp 디렉터리로 복사한다.
cp ~user2/linux.txt .	user2라는 사용자의 홈 디렉터리 아래에서 linux.txt 파일을 찾아 현재 디렉터리로 복사한다.
cd –	이전 작업 디렉터리로 이동한다.

02 셸 기본 사용법

특수문자 ;과 |

- ;(쌍반점)과 |(파이프)는 명령과 명령을 연결
- ;은 연결된 명령을 왼쪽부터 차례로 실행
- |는 왼쪽 명령의 실행 결과를 오른쪽 명령의 입력으로 전달

표 4-4 특수문자 ;과 |

사용 예	의미
date; ls; pwd	왼쪽부터 차례대로 명령을 실행한다. 즉 날짜를 출력한 후 현재 디렉터리의 파일 목록을 출력하고, 마지막으로 현재 작업 디렉터리의 절대 경로를 보여준다.
ls –al / | more	루트 디렉터리에 있는 모든 파일의 상세 정보를 한 화면씩 출력한다. ls –al / 명령의 결과가 more 명령의 입력으로 전달되어 페이지 단위로 출력되는 것이다.

02 셸 기본 사용법

■ 특수문자 ' '와 " "

- ' '(작은따옴표)와 " "(큰따옴표)는 문자를 감싸서 문자열로 만들어주고, 문자열 안에 사용된 특수문자의 기능을 없앰
- ' '는 모든 특수문자를, " "는 $, `, ₩을 제외한 모든 특수문자를 일반 문자로 간주하여 처리

표 4-5 특수문자 ' '와 " "

사용 예	의미
echo '$SHELL'	$SHELL 문자열이 화면에 출력된다.
echo "$SHELL"	셸 환경 변수인 SHELL에 저장된 값인 현재 셸의 종류가 화면에 출력된다. /bin/bash를 예로 들 수 있다.

■ 특수문자 ` `

- 셸은 ``` `로 감싸인 문자열을 명령으로 해석하여 명령의 실행 결과로 전환

표 4-6 특수문자 ` `

사용 예	의미
echo "Today is `date`"	`date`가 명령으로 해석되어 date 명령의 실행 결과로 바뀐다. 결과적으로 다음과 같이 출력된다. Today is 2018. 01. 20. (토) 18:32:45 KST
ls /usr/bin/`uname -m`	uname -m 명령의 실행 결과를 문자열로 바꾸어 파일명으로 사용한다.

02 셸 기본 사용법

■ 특수문자 ₩

- ₩(역빗금, ₩와 동일함)은 특수문자 바로 앞에 사용되는데 해당 특수문자의 효과를 없애고 일반 문자처럼 처리

표 4-7 특수문자 \

사용 예	의미
ls -l t*	t*라는 이름을 가진 파일의 상세 정보를 출력한다. \ 없이 t*를 사용하면 t로 시작하는 모든 파일의 상세 정보를 출력한다.
echo \$SHELL	$SHELL을 화면에 출력한다. echo '$SHELL'과 결과가 같다.

■ 특수문자 >, <, >>

- 입출력의 방향을 바꾸는 특수문자

표 4-8 특수문자 〉, 〈, 〉〉

사용 예	의미
ls -l 〉res	ls -l 명령의 실행 결과를 화면이 아닌 res 파일에 저장한다.

03 입출력 방향 바꾸기

표준 입출력 장치

- 표준 입력 장치: 셸이 작업을 수행하는 데 필요한 정보를 받아들이는 장치 -> 키보드
- 표준 출력 장치: 실행 결과를 내보내는 장치 -> 모니터
- 표준 오류 장치: 오류 메시지를 내보내는 장치 -> 모니터

그림 4-3 표준 입출력 장치

파일 디스크립터

- 파일 관리를 위해 붙이는 일련 번호
- 입출력 장치를 변경할 때는 이 파일 디스크립터를 사용
- 표준 입출력 장치를 파일로 바꾸는 것을 '리다이렉션(redirection)'이라고 함

표 4-9 표준 입출력 장치의 파일 디스크립터

파일 디스크립터	파일 디스크립터 대신 사용하는 이름	정의
0	stdin	명령의 표준 입력
1	stdout	명령의 표준 출력
2	stderr	명령의 표준 오류

03 입출력 방향 바꾸기

출력 리다이렉션

- > : 기존 파일의 내용을 삭제하고 새로 결과를 저장
- >> : 기존 파일의 내용 뒤에 결과를 추가

파일 덮어쓰기 : >

>

- 기능 파일 리다이렉션(덮어쓰기)을 한다.
- 형식 명령 1> 파일
  명령 > 파일

- 1: 파일 디스크립터 1번(표준 출력, 화면)
- 셸은 >를 사용한 리다이렉션에서 지정한 이름의 파일이 없으면 파일을 생성해서 명령의 수행 결과를 저장
- 파일이 있으면 이전의 내용이 없어지고 명령의 수행 결과로 대체

03 입출력 방향 바꾸기

- **파일 덮어쓰기 : > 예**

```
user1@myubuntu:~$ mkdir linux_ex/ch4
user1@myubuntu:~$ cd linux_ex/ch4
user1@myubuntu:~/linux_ex/ch4$ ls out1          → out1 파일이 있는지 확인한다.
ls: 'out1'에 접근할 수 없습니다: 그런 파일이나 디렉터리가 없습니다
user1@myubuntu:~/linux_ex/ch4$ ls -al           → 명령의 결과가 화면으로 출력된다(표준 출력).
합계 8
drwxrwxr-x 2  user1 user1 4096     01월 15 23:38 .
drwxrwxr-x 5  user1 user1 4096     01월 15 23:38 ..
user1@myubuntu:~/linux_ex/ch4$ ls -al > out1    → 명령의 결과를 out1 파일에 저장한다.
user1@myubuntu:~/linux_ex/ch4$ cat out1         → 파일 내용을 확인한다.
합계 8
drwxrwxr-x 2  user1 user1 4096     01월 15 23:41 .
drwxrwxr-x 5  user1 user1 4096     01월 15 23:38 ..
-rw-rw-r--  1  user1 user1    0     01월 15 23:41 out1
user1@myubuntu:~/linux_ex/ch4$ date > out1      → 명령의 결과를 out1 파일에 저장한다.
user1@myubuntu:~/linux_ex/ch4$ cat out1         → ls 명령의 실행 결과가 없어졌다.
2018. 01. 15. (월) 23:41:18 KST
user1@myubuntu:~/linux_ex/ch4$
```

03 입출력 방향 바꾸기

예상치 않게 파일의 내용이 겹쳐 쓰이는 상황을 예방하기

```
user1@myubuntu:~/linux_ex/ch4$ set -o noclobber
user1@myubuntu:~/linux_ex/ch4$ ls > out1
-bash: out1: cannot overwrite existing file
user1@myubuntu:~/linux_ex/ch4$
```

- 설정 해제

```
user1@myubuntu:~/linux_ex/ch4$ set +o noclobber
user1@myubuntu:~/linux_ex/ch4$ ls > out1
user1@myubuntu:~/linux_ex/ch4$
```

cat 명령으로 파일 생성하기

```
user1@myubuntu:~/linux_ex/ch4$ cat > out1      → 표준 입력을 받아 out1 파일에 저장한다.
Ubuntu Linux                                   → 내용을 입력한다.
I love Linux.
^D                                             → 입력을 종료한다.
user1@myubuntu:~/linux_ex/ch4$ cat out1        → 파일 내용을 확인한다.
Ubuntu Linux
I love Linux.
user1@myubuntu:~/linux_ex/ch4$
```

03 입출력 방향 바꾸기

■ 파일에 내용 추가하기 : >>

>>

- 기능 파일에 내용을 추가한다.
- 형식 명령 >> 파일

▪ 지정한 파일이 없으면 파일을 생성하고, 파일이 있으면 기존 파일의 끝에 명령의 실행 결과를 추가

```
user1@myubuntu:~/linux_ex/ch4$ cat out1        → 파일 내용을 확인한다.
Ubuntu Linux
I love Linux.
user1@myubuntu:~/linux_ex/ch4$ date >> out1    → 리다이렉션한다(내용 추가).
user1@myubuntu:~/linux_ex/ch4$ cat out1        → 파일 내용을 확인한다.
Ubuntu Linux
I love Linux.
2018. 01. 15. (월) 23:44:58 KST                        → 추가된 내용이다.
user1@myubuntu:~/linux_ex/ch4$
```

03 입출력 방향 바꾸기

■ 오류 리다이렉션

- 표준 오류도 기본적으로 화면으로 출력되며 표준 출력처럼 리다이렉션 가능
- 표준 출력과 표준 오류 예

```
user1@myubuntu:~/linux_ex/ch4$ ls
out1                                                    → 정상 실행(표준 출력)
user1@myubuntu:~/linux_ex/ch4$ ls /abc
ls: '/abc'에 접근할 수 없습니다: 그런 파일이나 디렉터리가 없습니다   → 오류 메시지(표준 오류)
user1@myubuntu:~/linux_ex/ch4$
```

- 표준출력 리다이렉션: 오류 메시지는 리다이렉션 안됨

```
user1@myubuntu:~/linux_ex/ch4$ ls > ls.out              → 표준 출력 리다이렉션
user1@myubuntu:~/linux_ex/ch4$ ls /abc > ls.err         → 표준 출력 리다이렉션
ls: '/abc'에 접근할 수 없습니다: 그런 파일이나 디렉터리가 없습니다 → 오류 메시지가 화면에 출력된다.
user1@myubuntu:~/linux_ex/ch4$ cat ls.err               → 오류 메시지가 저장되지 않았다.
user1@myubuntu:~/linux_ex/ch4$ cat ls.out               → 표준 출력 내용이 출력된다.
ls.out
out1
user1@myubuntu:~/linux_ex/ch4$
```

03 입출력 방향 바꾸기

■ 오류 리다이렉션

- 오류 리다이렉션에서는 파일 디스크립터 번호를 생략 불가

2>

- **기능** 표준 오류 메시지를 파일에 저장한다.
- **형식** 명령 2> 파일

```
user1@myubuntu:~/linux_ex/ch4$ ls /abc 2> ls.err          → 표준 오류를 리다이렉션한다.
user1@myubuntu:~/linux_ex/ch4$ cat ls.err
ls: '/abc'에 접근할 수 없습니다: 그런 파일이나 디렉터리가 없습니다  → 파일에 저장된 메시지이다.
user1@myubuntu:~/linux_ex/ch4$
```

- 표준 출력과 표준 오류를 한 번에 리다이렉션하기

```
user1@myubuntu:~/linux_ex/ch4$ ls . /abc > ls.out 2> ls.err
user1@myubuntu:~/linux_ex/ch4$
```

03 입출력 방향 바꾸기

오류 리다이렉션

- 오류 메시지 버리기

```
user1@myubuntu:~/linux_ex/ch4$ ls /abc 2> /dev/null
user1@myubuntu:~/linux_ex/ch4$
```

표준 출력과 표준 오류를 한 파일로 리다이렉션하기

- 명령의 정상 실행 결과를 파일로 리다이렉션(>).
- 그 명령 전체의 오류 메시지를 1번 파일(표준 출력 파일, &1이라고 표현함)로 리다이렉션(2>).

```
user1@myubuntu:~/linux_ex/ch4$ ls . /abc > ls.out 2>&1
user1@myubuntu:~/linux_ex/ch4$ cat ls.out
ls: '/abc'에 접근할 수 없습니다: 그런 파일이나 디렉터리가 없습니다   → 오류 메시지를 저장한다.
.:                                                                    → 현재 디렉터리 내용이다.
ls.err
ls.out
out1
user1@myubuntu:~/linux_ex/ch4$
```

03 입출력 방향 바꾸기

■ 입력 리다이렉션

<

- 기능 표준 입력을 바꾼다.
- 형식 명령 0< 파일
  명령 < 파일

▪ 입력 리다이렉션 예: cat 명령

```
user1@myubuntu:~/linux_ex/ch4$ cat out1      → 파일 내용을 출력한다(< 생략).
Ubuntu Linux
I love Linux.
2018. 01. 15. (월) 23:44:58 KST
user1@myubuntu:~/linux_ex/ch4$ cat < out1    → 표준 입력을 리다이렉션한다(< 사용).
Ubuntu Linux
I love Linux.
2018. 01. 15. (월) 23:44:58 KST
user1@myubuntu:~/linux_ex/ch4$ cat 0< out1   → 표준 입력을 리다이렉션한다(0< 사용).
Ubuntu Linux
I love Linux.
2018. 01. 15. (월) 23:44:58 KST
user1@myubuntu:~/linux_ex/ch4$
```

04 배시 셸 환경 설정

셸 변수와 환경변수

- 셸의 환경을 설정하기 위한 값을 저장할 수 있도록 셸 변수와 환경 변수를 제공
- 셸 변수: 현재 셸에서만 사용이 가능하고 서브 셸로는 전달되지 않음(지역변수)
- 환경 변수: 현재 셸뿐만 아니라 서브 셸로도 전달(전역변수)

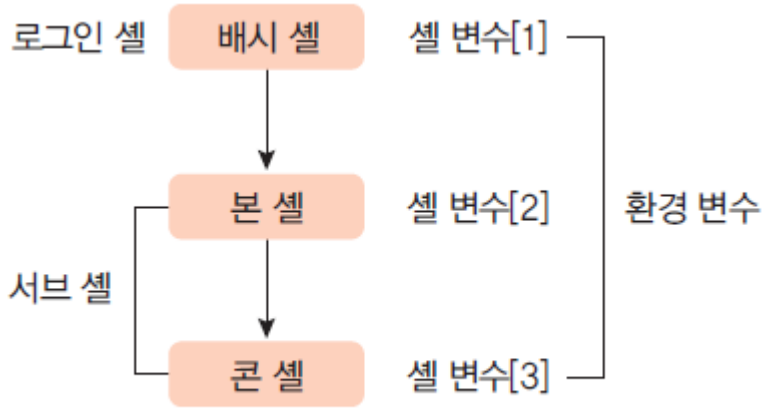


그림 4-4 셸 변수와 환경 변수

04 배시 셸 환경 설정

전체 변수 출력: set, env

- set: 셸 변수와 환경변수 모두 출력

```
user1@myubuntu:~/linux_ex/ch4$ set
BASH=/bin/bash
BASHOPTS=checkwinsize:cmdhist:complete_fullquote:expand_aliases:extglob:extquote
:force_fignore:histappend:interactive_comments:login_shell:progcomp:promptvars:
sourcepath
BASH_ALIASES=()
BASH_ARGC=()
(생략)
quote_readline ()
{
    local quoted;
    _quote_readline_by_ref "$1" ret;
    printf %s "$ret"
}
user1@myubuntu:~/linux_ex/ch4$
```

04 배시 셸 환경 설정

전체 변수 출력: set, env

- env: 환경변수만 출력

```
user1@myubuntu:~/linux_ex/ch4$ env
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd
=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;
(생략)
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/
local/games:/snap/bin
LESSOPEN=| /usr/bin/lesspipe %s
_=/usr/bin/env
OLDPWD=/home/user1
user1@myubuntu:~/linux_ex/ch4$
```

04 배시 셸 환경 설정

■ 주요 셸 환경변수

표 4-10 주요 셸 환경 변수

환경 변수	의미	환경 변수	의미
HISTSIZE	히스토리 저장 크기	PATH	명령을 탐색할 경로
HOME	사용자 홈 디렉터리의 절대 경로	PWD	작업 디렉터리의 절대 경로
LANG	사용하는 언어	SHELL	로그인 셸
LOGNAME	사용자 계정 이름		

■ 특정 변수 출력하기 : echo

- 변수의 값을 출력할 때는 변수 이름 앞에 특수문자 $를 붙임

```
user1@myubuntu:~/linux_ex/ch4$ echo $SHELL
/bin/bash
user1@myubuntu:~/linux_ex/ch4$
```

04 배시 셸 환경 설정

셸 변수 설정하기

- 변수 이름과 문자열 사이에 공백이 있으면 안됨

셸 변수 정의

- 형식 변수명=문자열
- 사용 예 SOME=test

```
user1@myubuntu:~/linux_ex/ch4$ SOME=test
user1@myubuntu:~/linux_ex/ch4$ echo $SOME
test
user1@myubuntu:~/linux_ex/ch4$
```

04 배시 셸 환경 설정

환경 변수 설정하기 : export

- 먼저 셸 변수를 정의하고, export 명령을 사용하여 이를 환경 변수로 변경

export

- 기능 지정한 셸 변수를 환경 변수로 바꾼다.
- 형식 export [옵션] [셸 변수]
- 옵션 -n: 환경 변수를 셸 변수로 바꾼다.
- 사용 예 export
  export SOME
  export SOME=test

```
user1@myubuntu:~/linux_ex/ch4$ export SOME
user1@myubuntu:~/linux_ex/ch4$ env | grep SOME
SOME=test
user1@myubuntu:~/linux_ex/ch4$
```

- 변수를 설정하면서 바로 export 명령을 사용하여 한 번에 환경 변수로 전환도 가능

```
user1@myubuntu:~/linux_ex/ch4$ export SOME1=test1
user1@myubuntu:~/linux_ex/ch4$ echo $SOME1
test1
user1@myubuntu:~/linux_ex/ch4$ env | grep SOME
SOME=test
SOME1=test1
user1@myubuntu:~/linux_ex/ch4$
```

04 배시 셸 환경 설정

환경 변수를 다시 셸 변수로 바꾸기 : export –n

- 예: SOME은 보이지만 SOME1은 보이지 않음

```
user1@myubuntu:~/linux_ex/ch4$ export -n SOME1
user1@myubuntu:~/linux_ex/ch4$  env | grep SOME
SOME=test
user1@myubuntu:~/linux_ex/ch4$
```

변수 해제하기

unset

- 기능 지정한 변수를 해제한다.
- 형식 unset 변수
- 사용 예 unset SOME

```
user1@myubuntu:~/linux_ex/ch4$ unset SOME
user1@myubuntu:~/linux_ex/ch4$ unset SOME1
user1@myubuntu:~/linux_ex/ch4$ echo $SOME

user1@myubuntu:~/linux_ex/ch4$ echo $SOME1

user1@myubuntu:~/linux_ex/ch4$
```

05 에일리어스와 히스토리

■ 에일리어스

- 에일리어스(alias)는 우리말로 '별명'을 의미
- 기존의 명령을 대신하여 다른 이름(별명)을 붙일 수 있도록 하는 기능
- 긴 명령 대신 짧은 명령을 만들어 사용 가능
- 여러 명령을 연결하여 하나의 명령으로 만들 수도 있음
- 자주 사용하는 옵션을 포함하여 새로운 이름을 붙여서 사용 가능

alias

- **기능** 에일리어스를 생성한다.
- **형식** alias 이름='명령'
- **사용 예** alias: 현재 설정된 별칭 목록을 출력한다.
  alias 이름='명령': 명령을 수정하여 사용하는 경우이다.
  alias 이름='명령;명령2;…': 여러 명령을 하나의 이름으로 사용하는 경우이다.

05 에일리어스와 히스토리

기존 에일리어스 확인: alias

- 아무것도 지정하지 않고 alias 명령을 실행하면 현재 설정되어 있는 에일리어스가 출력

```
user1@myubuntu:~/linux_ex/ch4$ alias
alias alert='notify-send --urgency=low -i "$([ $? = 0 ] && echo terminal || echo
error)" "$(history|tail -n1|sed -e '\''s/^\s*[0-9]\+\s*//;s/[;&|]\s*alert$//'\'')"'
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -alF'
alias ls='ls --color=auto'
user1@myubuntu:~/linux_ex/ch4$
```

05 에일리어스와 히스토리

기존 에일리어스 확인: alias

- 에일리어스 실행 예

```
user1@myubuntu:~/linux_ex/ch4$ ll
합계 28
drwxrwxr-x 3 user1 user1 4096 11월 16 00:01 ./
drwxrwxr-x 5 user1 user1 4096 11월 15 23:38 ../
-rw-rw-r--  1 user1 user1   91 11월 16 00:00 ls.err
-rw-rw-r--  1 user1 user1  113 11월 16 00:00 ls.out
-rw-rw-r--  1 user1 user1   60 11월 15 23:44 out1
drwxrwxr-x 2 user1 user1 4096 11월 15 23:59 temp/
-rw-rw-r--  1 user1 user1   33 11월 15 23:59 u.out
user1@myubuntu:~/linux_ex/ch4$
```

05 에일리어스와 히스토리

에일리어스 설정하기 : alias

- '에일리어스 이름=명령' 형식 사용
- 에일리어스 설정 예: ls

```
user1@myubuntu:~/linux_ex/ch4$ mkdir tmp
user1@myubuntu:~/linux_ex/ch4$ ls
ls.err  ls.out  out1  temp  tmp  u.out
user1@myubuntu:~/linux_ex/ch4$ alias ls='ls -F'   → 공백이 있으면 작은따옴표를 사용한다.
user1@myubuntu:~/linux_ex/ch4$ ls                 → 에일리어스의 ls를 실행한다.
ls.err  ls.out  out1  temp/  tmp/  u.out          → ls -F의 결과를 출력한다.
user1@myubuntu:~/linux_ex/ch4$
```

- 에일리어스 설정 예: rm

```
user1@myubuntu:~/linux_ex/ch4$ alias rm='rm -i'
user1@myubuntu:~/linux_ex/ch4$ rm out1
rm: 일반 파일 'out1'를 제거할까요? n
user1@myubuntu:~/linux_ex/ch4$
```

05 에일리어스와 히스토리

에일리어스에 인자 전달하기

- 배시 셸에서는 에일리어스로 인자를 전달할 수 없음
- 배시 셸에서 인자를 전달하려면 프로그래밍 기능에서 함수를 사용
- 인자 전달 함수 예

```
user1@myubuntu:~$ unalias cd              → 에일리어스를 해제한다.
user1@myubuntu:~$ function cdpwd {        → 함수 입력을 시작한다.
> cd $1;pwd                               → 프롬프트가 >로 바뀐다. 내용을 입력한다.
> }                                       → 함수 입력을 종료한다.
user1@myubuntu:~$ cdpwd /tmp
/tmp
user1@myubuntu:/tmp$
```

에일리어스 해제하기 : unalias

unalias

- **기능** 에일리어스를 삭제한다.
- **형식** unalias 에일리어스

```
user1@myubuntu:/tmp$ unalias ls
user1@myubuntu:/tmp$ unalias rm
user1@myubuntu:/tmp$
```

05 에일리어스와 히스토리

히스토리

- 사용자가 이전에 입력한 명령을 다시 불러 사용하는 것

history

- **기능** 히스토리(명령 입력 기록)를 출력한다.
- **형식** history

```
user1@myubuntu:/tmp$ history
(생략)
   86  alias cd='cd;pwd'
   87  cd
   88  cd linux_ex/ch4
   89  unalias cd
   90  function cdpwd { cd $1;pwd; }
   91  cdpwd /tmp
   92  history
user1@myubuntu:/tmp$
```

05 에일리어스와 히스토리

■ 명령 재실행하기 : !

표 4-11 !를 사용한 명령 재실행 방법

사용법	기능
!!	바로 직전에 실행한 명령을 재실행한다.
!번호	히스토리에서 해당 번호의 명령을 재실행한다.
!문자열	히스토리에서 해당 문자열로 시작하는 마지막 명령을 재실행한다.

- 직전 명령 재실행 예

```
user1@myubuntu:/tmp$ cd ~/linux_ex/ch4
user1@myubuntu:~/linux_ex/ch4$ ls
ls.err  ls.out  out1  temp  tmp  u.out
user1@myubuntu:~/linux_ex/ch4$ !!          → 바로 직전에 실행한 명령을 재실행한다.
ls
ls.err  ls.out  out1  temp  tmp  u.out
user1@myubuntu:~/linux_ex/ch4$
```

05 에일리어스와 히스토리

명령 재실행하기 : !

- 이전에 수행한 명령을 재실행 예

```
user1@myubuntu:~/linux_ex/ch4$ history
(생략)
   93  cd ~/linux_ex/ch4
   94  ls
   95  history
user1@myubuntu:~/linux_ex/ch4$ !94          ➜ 히스토리 번호로 재실행한다.
ls
ls.err  ls.out  out1  temp  tmp  u.out
user1@myubuntu:~/linux_ex/ch4$ !l           ➜ 명령의 앞 글자로 재실행한다.
ls
ls.err  ls.out  out1  temp  tmp  u.out
user1@myubuntu:~/linux_ex/ch4$
```

05 에일리어스와 히스토리

■ 명령 편집하기와 재실행하기

- 화살표 키를 사용하여 오류가 난 명령을 다시 프롬프트로 불러내서 수정한 뒤 재실행 가능

① 편집과 재실행 예1 : 명령에 오타를 입력

```
user1@myubuntu:~/linux_ex/ch4$ man hisdory
No manual entry for hisdory
user1@myubuntu:~/linux_ex/ch4$
```

② 프롬프트에서 ↑키를 누르면 방금 실행한 명령이 다시 나타남

```
user1@myubuntu:~/linux_ex/ch4$ man hisdory
```

③ 좌우 화살표로 커서를 이동하여 백스페이스키로 삭제한 후 다시 글자를 입력하고 엔터키를 눌러서 실행

```
user1@myubuntu:~/linux_ex/ch4$ man history
```

◆ 히스토리 저장하기

- 로그아웃할 때 홈 디렉터리 아래의 숨김 파일인 .bash_history에 히스토리 저장

```
user1@myubuntu:~/linux_ex/ch4$ more ~/.bash_history
file .profile
file 다운로드
file /bin/ls
(생략)
user1@myubuntu:~/linux_ex/ch4$
```

06 프롬프트 설정

프롬프트 설정 변수: PS1

- 프롬프트를 바꾸는 것은 환경 변수 PS1에 새로운 형태의 문자열을 지정하는 것

```
user1@myubuntu:~/linux_ex/ch4$ echo $PS1
\[\e]0;\u@\h: \w\a\]${debian_chroot:+($debian_chroot)}\u@\h:\w\$
user1@myubuntu:~/linux_ex/ch4$
```

이스케이프 문자와 프롬프트 설정하기

- ₩으로 시작하는 특별한 문자가 이스케이프 문자
- ₩u와 같이 ₩으로 시작하는 이스케이프 문자는 두 글자가 아니라 한 글자로 처리
- 이스케이프 문자는 화면에 문자 그대로 출력되지 않고 셸이 문자의 의미를 해석하여 실행

06 프롬프트 설정

■ 프롬프트에서 사용할 수 있는 이스케이프 문자

표 4-12 이스케이프 문자

이스케이프 문자	기능
\a	ASCII 종소리 문자(07)
\d	'요일 월 일' 형식으로 날짜를 표시한다(예: Wed May 1).
\e	ASCII의 이스케이프 문자로 터미널에 고급 옵션을 전달한다.
\h	첫 번째 .(마침표)까지의 호스트 이름(예: server.co.kr에서 server)
\H	전체 호스트 이름
\n	줄 바꾸기
\s	셸 이름
\t	24시간 형식으로 현재 시간을 표시한다(HH:MM:SS 형식).
\T	12시간 형식으로 현재 시간을 표시한다(HH:MM:SS 형식).
\@	12시간 형식으로 현재 시간을 표시한다(오전/오후 형식).
\u	사용자 이름
\v	배시 셸의 버전
\w	현재 작업 디렉터리(절대 경로)
\W	현재 작업 디렉터리의 절대 경로에서 마지막 디렉터리명
\!	현재 명령의 히스토리 번호
\[	출력하지 않을 문자열의 시작 부분을 표시한다.
\]	출력하지 않을 문자열의 끝부분을 표시한다.

06 프롬프트 설정

■ 프롬프트 변경 예제

① 간단한 문자열로 변경: 프롬프트의 끝을 표시하기 위해 마지막에]나 $ 같은 표시를 하고 공백 문자를 둠

```
user1@myubuntu:~/linux_ex/ch4$ PS1='LINUX ] '
LINUX ]
```

② 환경 변수를 사용: 프롬프트에 현재 작업 디렉터리가 출력

```
LINUX ] PS1='[$PWD] '
[/home/user1/linux_ex/ch4] cd ..
[/home/user1/linux_ex]
```

③ 명령의 실행 결과를 사용: 특수문자 `` `를 이용, uname -n 명령은 호스트 이름을 출력

```
[/home/user1/linux_ex]  PS1='`uname -n` $ '
myubuntu $
```

④ 이스케이프 문자 ₩u, ₩T, ₩!를 사용

```
myubuntu $ PS1='[\u \T] \!$ '
[user1 10:54:50] 103$
```

06 프롬프트 설정

■ 컬러 프롬프트 설정하기

컬러 프롬프트

- **형식** `PS1= '\[\e[x;y;nm\] 프롬프트\[\e[x;y;0m\]'`

표 4-13 프롬프트 컬러 번호

컬러	글자색 번호	배경색 번호
검은색	30	40
빨간색	31	41
초록색	32	42
갈색	33	43
파란색	34	44
보라색	35	45
청록색	36	46
하얀색	37	47

표 4-14 프롬프트 특수 기능 번호

번호	기능
0	기본 색상
1	볼드
4	밑줄
5	반짝임
7	역상
10	기본 폰트
38	밑줄 사용 가능
39	밑줄 사용 불가능

06 프롬프트 설정

■ 컬러 프롬프트 설정 예

① 파란색으로 설정하기

```
[user1 10:54:50] 103$ PS1='\e[34mLinux $\e[0;0m'
Linux $      → 파란색
```

② 파란색의 볼드로 설정하기

```
Linux $ PS1='\e[34;1mLinux $\e[0;0m'
Linux $      → 파란색, 볼드
```

③ 밑줄 친 빨간색으로 설정하기

```
Linux $  PS1='\e[31;4mLinux $\e[0;0m'
Linux $      → 빨간색, 밑줄
```

④ 배경은 갈색, 글자는 보라색, 프롬프트는 '사용자 이름@호스트 이름 $ '로 설정하기

```
Linux $ PS1='\e[35;43m\u@\h $\e[0;0m'
user1@myubuntu $       → 갈색 배경, 보라색 글자
```

07 환경 설정 파일

■ 환경 설정 파일

- 사용자가 로그인할 때마다 자동으로 실행되는 명령을 저장한 것이 환경 설정 파일
- 시스템 환경 설정 파일과 사용자 환경 설정 파일이 있음
- 셸마다 다른 이름의 파일을 사용

■ 시스템 환경 설정 파일

- 시스템을 사용하는 전체 사용자의 공통 환경을 설정하는 파일

표 4-15 배시 셸의 시스템 환경 설정 파일

파일	기능
/etc/profile	• 본 셸이나 본 셸과 호환되는 모든 셸에 공통으로 적용되는 .profile 파일이다. • 배시 셸의 경우 /etc/bash.bashrc 파일을 실행한다. • 배시 셸이 아닌 경우 프롬프트를 #(root 사용자)나 $(일반 사용자)로 설정한다. • /etc/profile.d/*.sh 파일을 실행한다.
/etc/bash.bashrc	• 시스템 공통으로 적용되는 .bashrc 파일이다. • 기본 프롬프트를 설정한다. • sudo 명령과 관련된 힌트를 설정한다.
/etc/profile.d/*.sh	• 언어나 명령별로 각각 필요한 환경을 설정한다. • 필요시 설정 파일을 추가한다.

07 환경 설정 파일

■ 시스템 환경 설정 파일

- /etc/profile 파일

```
user1@myubuntu:~/linux_ex/ch4$ cd
user1@myubuntu:~$ more /etc/profile
# /etc/profile: system-wide .profile file for the Bourne shell (sh(1))
# and Bourne compatible shells (bash(1), ksh(1), ash(1), ...).

if [ "${PS1-}" ]; then
  if [ "${BASH-}" ] && [ "$BASH" != "/bin/sh" ]; then
    # The file bash.bashrc already sets the default PS1.
    # PS1='\h:\w\$ '
    if [ -f /etc/bash.bashrc ]; then
      . /etc/bash.bashrc
    fi
  else
    if [ "`id -u`" -eq 0 ]; then
      PS1='# '
    else
      PS1='$ '
    fi
  fi
fi
(생략)
```

07 환경 설정 파일

■ 사용자 환경 설정 파일

- 각 사용자의 홈 디렉터리에 숨김 파일로 생성
- 사용자가 내용을 수정하고 관리 가능

표 4-16 배시 셸의 사용자 환경 설정 파일

파일	기능
~/.profile	• 경로 추가 등 사용자가 정의하는 환경을 설정한다. • .bashrc 파일이 있으면 실행한다.
~/.bashrc	• 히스토리의 크기를 설정한다. • 기본 에일리어스나 함수 등을 설정한다.
~/.bash_aliases	• 사용자가 정의한 에일리어스를 별도 파일로 저장한다.
~/.bash_logout	• 로그아웃 시 실행할 필요가 있는 함수 등을 설정한다.

07 환경 설정 파일

- 사용자 환경 설정 파일 예

```
user1@myubuntu:~$ cat .profile
# ~/.profile: executed by the command interpreter for login shells.
# This file is not read by bash(1), if ~/.bash_profile or ~/.bash_login
# exists.
# see /usr/share/doc/bash/examples/startup-files for examples.
# the files are located in the bash-doc package.

# the default umask is set in /etc/profile; for setting the umask
# for ssh logins, install and configure the libpam-umask package.
#umask 022

# if running bash
if [ -n "$BASH_VERSION" ]; then
    # include .bashrc if it exists
    if [ -f "$HOME/.bashrc" ]; then
        . "$HOME/.bashrc"
    fi
fi

# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/bin" ] ; then
    PATH="$HOME/bin:$PATH"
fi
user1@myubuntu:~$
```

07 환경 설정 파일

사용자 환경 설정 파일 만들기

- vi로 .bash_aliases 파일 수정

```
user1@myubuntu:~$ vi .bash_aliases
alias rm='rm -i'
alias h=history
alias c=clear
~
:wq                    → 저장하고 종료한다.
user1@myubuntu:~
```

사용자 환경 설정 파일 적용하기

```
user1@myubuntu:~$ source .bash_aliases
```

```
user1@myubuntu:~$ . .bash_aliases
```

07 환경 설정 파일

■ 다른 셸의 환경 설정 파일

표 4-17 다른 셸의 환경 설정 파일

셸	시스템 초기화 파일	사용자 초기화 파일	실행 조건	실행 시기		
				로그인	서브 셸	로그아웃
본 셸	/etc/profile	$HOME/.profile		○		
콘 셸	/etc/profile	$HOME/.profile		○		
		$HOME/.kshrc	ENV 변수 설정	○	○	
C 셸	/etc/.login	$HOME/.login		○		
		$HOME/.cshrc		○	○	
		$HOME/.logout				○