

Chapter 04 셸 사용법

목차

- 00 개요
- 01 셀의 기능과 종류
- 02 셀 기본 사용법
- 03 입출력 방향 변경
- 04 배시셀 환경 설정
- 05 앨리어스와 히스토리
- 06 프롬프트 설정
- 07 환경 설정 파일

학습목표



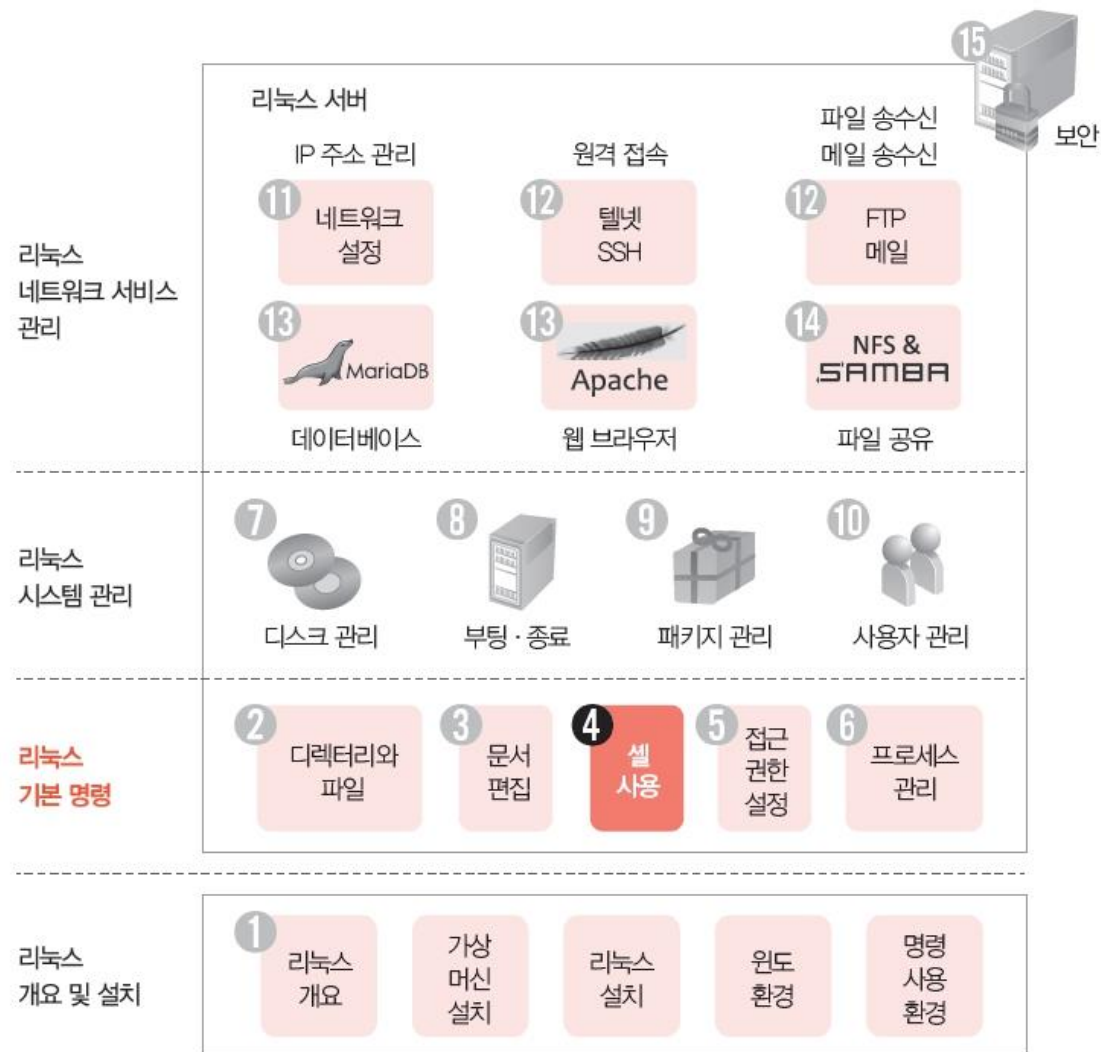
- 셀의 기능과 종류를 설명할 수 있다.
- 셀 특수문자의 종류를 이해하고 필요에 따라 적절하게 사용할 수 있다.
- 표준 입출력 장치를 이해하고 입출력 방향을 설정할 수 있다.
- 새로운 앨리어스를 만들거나 필요 없는 앨리어스를 삭제할 수 있다.
- 이스케이프 문자를 이해하고 프롬프트를 원하는 형태로 바꿀 수 있다.
- 시스템과 사용자 환경 설정 파일을 구분하고 사용자 환경을 직접 설정할 수 있다..

00 개요

00. 개요

■ 리눅스 학습 맵에서 4장의 위치

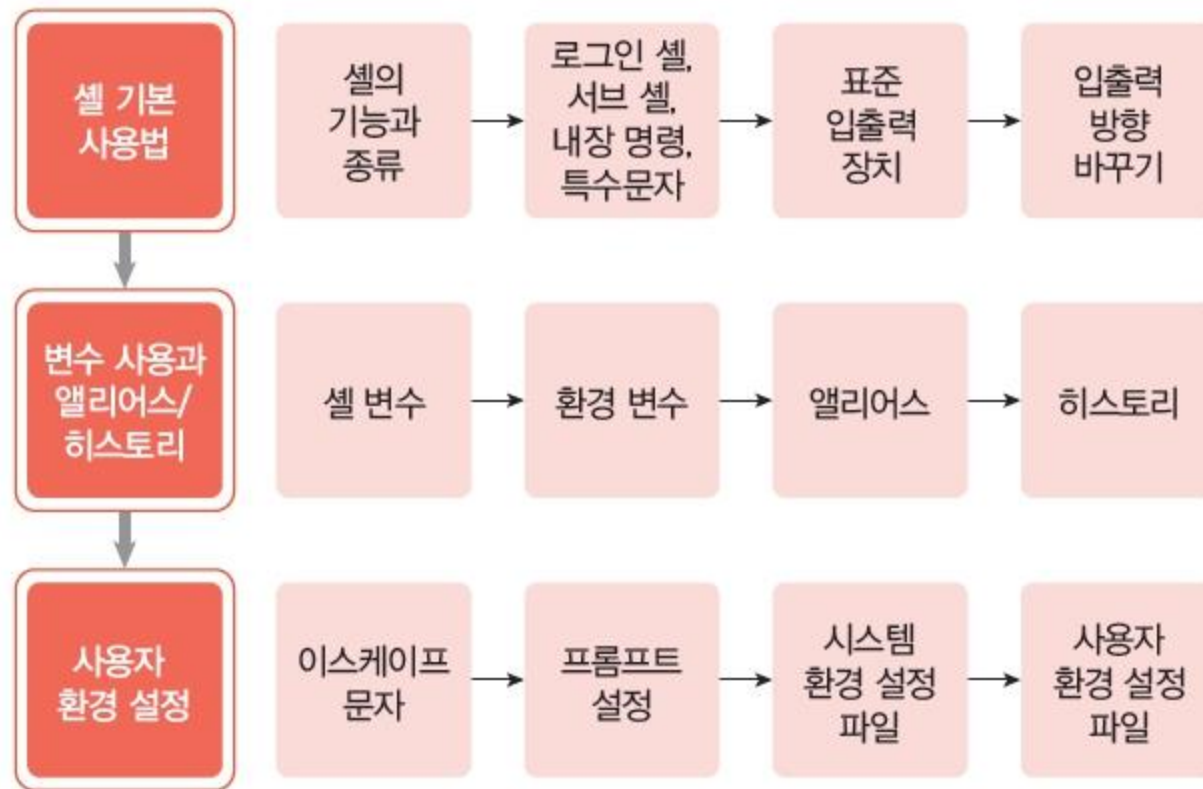
- 4장은 제2단계 중 세 번째 항목으로 셀이 무엇인지 이해하고 사용하는 방법을 익힌다.



00. 개요

■ 4장의 내용 구성

- 리눅스 커널과 사용자를 연결하는 역할을 수행하는 셸의 기능과 종류 이해
- 리눅스의 기본 셸인 배시셸의 기능 학습
- 사용자 환경 설정 방법 학습



01 셀의 기능과 종류

01. 셀의 기능과 종류

■ 셀의 기능

- 명령어 해석기 기능
- 프로그래밍 기능: 셀 스크립트
- 사용자 환경 설정 기능

01. 셸의 기능과 종류

■ 셸의 종류

- 본셸(sh): 최초의 셸, 유닉스 v7에서 처음 등장, 현재는 배시셸 등 다른 셸로 대체
- C셸(csh): 캘리포니아대학교에서 빌 조이가 개발, 2BSD 유닉스에서 발표
 - 앨리어스, 히스토리 기능 포함, 셸 스크립트 구문 형식이 C언어와 동일
- 콘셸(ksh): 1980년대 중반 AT&T 벨연구소에서 개발, SVR4 유닉스에서 발표
 - 본셸과 호환성 유지, 앨리어스, 히스토리 기능 제공
- 배시셸(bash): 1988년 브레인 폭스가 개발
 - 본셸과 호환성 유지, C셸/콘셸의 편리한 기능 모두 포함
 - 리눅스의 기본 셸로 제공
- 대시셸(dash): 1997년 허버스 슈가 리눅스에 이식
 - 본셸을 기반으로 개발, POSIX 표준 준수하며 작은 크기로 개발
 - 우분투는 6.10버전부터 본셸 대신 대시셸을 사용

02 셀 기본 사용법

02. 셸 기본 사용법

■ 셸 지정 및 변경

- 사용자의 기본 셸은 /etc/passwd 파일에 저장

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

- 기본 셸 변경하기

chsh

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

```
user1@myubuntu:~$ chsh -s /bin/sh user1
암호:
user1@myubuntu:~$
```

바꾸려는 셸의
절대경로 지정

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

02. 셸 기본 사용법

■ 로그인 셸과 서브 셸

- 서브 셸: 사용자가 프롬프트에서 다른 셸을 실행하여 생성한 셸
- 여러 개의 서브 셸을 사슬처럼 연결 가능
- ctrl+d 또는 exit로 서브 셸 종료

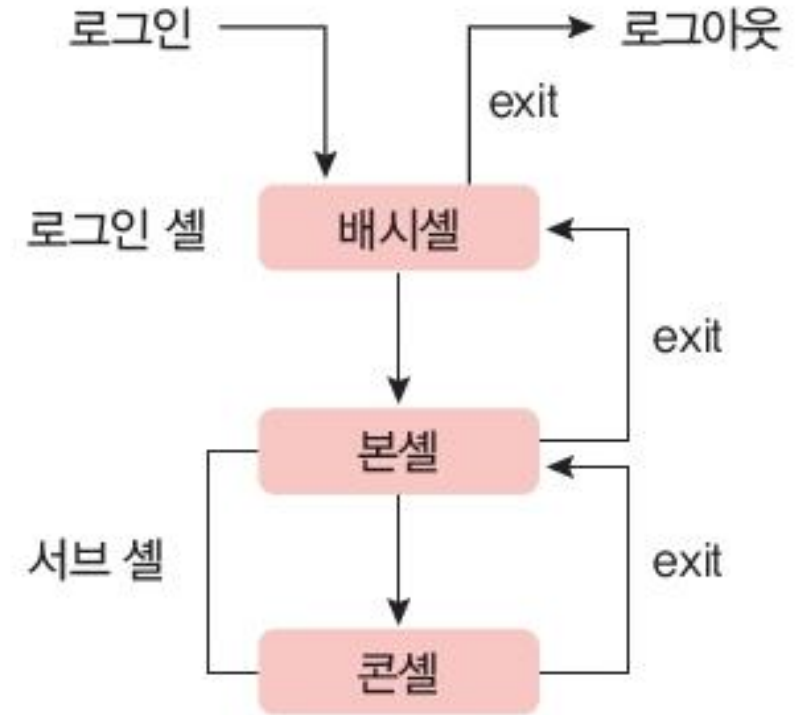


그림 4-1 로그인 셸과 서브 셸

02. 셸 기본 사용법

■ 셸 내장 명령

- 셸 자체적으로 가지고 있는 명령
- 대표적인 내장 명령: cd
- 일반적인 리눅스 명령들을 /bin이나 /usr/bin 등 다른 디렉터리에 실행파일이 있음
- file 명령으로 실행 파일 확인 예: /usr/bin/pwd

```
user1@myubuntu:~$ file /usr/bin/pwd
/usr/bin/pwd: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically
linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=401d991746e6ee4db275a4
fd4a860eab43145610, for GNU/Linux 3.2.0, stripped
```

02. 셸 기본 사용법

■ 출력 명령

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
```

printf

- **기능** 자료를 형식화하여 화면에 출력한다.
- **형식** printf [형식] [인수]
- **옵션** 형식: %d, \n 등 C 언어의 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 *

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 []

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 -

표 4-3 특수문자 ~와 -

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

02. 셸 기본 사용법

■ 특수문자 ; 과 |

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 " "

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.

02. 셸 기본 사용법

■ 특수문자 ``

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 2021. 12. 12. (일) 15:42:40 KST
ls /usr/bin/`uname -m`	uname -m 명령의 실행 결과를 문자열로 바꾸어 파일 이름으로 사용한다.

02. 셸 기본 사용법

■ 특수문자 \

표 4-7 특수문자 \

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

02. 셸 기본 사용법

■ 특수문자 >, <, >>

표 4-8 특수문자 >, <, >>

사용 예	의미
ls -l > res	ls -l 명령의 실행 결과를 화면이 아닌 res 파일에 저장한다.
ls -l >> res	ls -l의 명령의 결과를 res파일의 끝부분에 추가한다.
cat < text	cat 명령의 입력을 text 파일에서 받는다.

03 입출력 방향 변경

03. 입출력 방향 변경

■ 표준 입출력 장치



그림 4-2 표준 입출력 장치

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 변경

■ 출력 리다이렉션

- 명령의 결과를 화면 출력이 아닌 파일에 저장
- 파일 덮어쓰기: >
 - 명령의 실행결과를 기존 파일의 내용에 덮어쓴다.

>

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

03. 입출력 방향 변경

```
user1@myubuntu:~$ mkdir linux_ex/ch4
user1@myubuntu:~$ cd linux_ex/ch4
user1@myubuntu:~/linux_ex/ch4$ ls out1           → out1이 있는지 확인한다.
ls: 'out1'에 접근할 수 없습니다: 그런 파일이나 디렉터리가 없습니다
user1@myubuntu:~/linux_ex/ch4$ ls -al           → 명령의 결과가 화면(표준 출력)으로 출력된다.
합계 8
drwxrwxr-x 2 user1 user1 4096 12월 12 15:43 .
drwxrwxr-x 5 user1 user1 4096 12월 12 15:43 ..
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 12월 12 15:45 .
drwxrwxr-x 5 user1 user1 4096 12월 12 15:43 ..
-rw-rw-r-- 1 user1 user1    0 12월 12 15:45 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.
2021. 12. 12. (일) 15:45:45 KST
```

03. 입출력 방향 변경

- 실수로 중요한 파일에 덮어쓰기를 방지하려면: `set -o noclobber`

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

- 설정을 해제하려면 + 옵션 사용

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

03. 입출력 방향 변경

- 파일에 내용 추가하기: >>

>>

- **기능** 파일에 내용을 추가한다.
- **형식** 명령 >> 파일명

```
user1@myubuntu:~/linux_ex/ch4$ cat out1
```

```
Ubuntu Linux
```

```
I love Linux.
```

```
user1@myubuntu:~/linux_ex/ch4$ date >> out1
```

```
user1@myubuntu:~/linux_ex/ch4$ cat out1
```

```
Ubuntu Linux
```

```
I love Linux.
```

```
2021. 12. 12. (일) 15:50:18 KST
```

→ 기존 파일 내용을 확인한다.

→ 리다이렉션한다(내용 추가).

→ 파일 내용을 확인한다.

→ 추가된 내용

03. 입출력 방향 변경

■ 오류 리다이렉션

- 오류는 기본적으로 화면으로 출력

```
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$ 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
```

03. 입출력 방향 변경

- 오류를 파일에 저장하기

2>

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

```
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.

03. 입출력 방향 변경

- 표준 출력과 표준 오류를 한 번에 리다이렉션하기

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

- 오류 메시지 버리기

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

- 표준 출력과 표준 오류를 한 파일로 리다이렉션하기

```
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 내용

```
ls.err
```

```
ls.out
```

```
out1
```

03. 입출력 방향 변경

■ 입력 리다이렉션

<

- **기능** 표준 입력을 바꾼다.
- **형식** 명령 0< 파일명
명령 < 파일명

```
user1@myubuntu:~/linux_ex/ch4$ cat out1
```

```
Ubuntu Linux
```

```
I love Linux.
```

```
2021. 12. 12. (일) 15:50:18 KST
```

```
user1@myubuntu:~/linux_ex/ch4$ cat < out1
```

```
Ubuntu Linux
```

```
I love Linux.
```

```
2021. 12. 12. (일) 15:50:18 KST
```

```
user1@myubuntu:~/linux_ex/ch4$ cat 0< out1
```

```
Ubuntu Linux
```

```
I love Linux.
```

```
2021. 12. 12. (일) 15:50:18 KST
```

→ 파일 내용을 출력한다(< 생략).

→ 표준 입력을 리다이렉션한다(< 사용).

→ 표준 입력을 리다이렉션한다(0< 사용).

201p. 따라해보기 : vi로 입력,수정,삭제,복구하기

- ① 현재 디렉터리에서 l(소문자 L)로 시작하는 모든 파일의 상세 정보를 확인: `ls -l l*`
- ② 임시로 사용할 temp 디렉터리 생성: `mkdir tmp`
- ③ l로 시작하는 모든 파일을 temp 디렉터리로 이동: `mv l* temp`
- ④ uname 명령은 운영체제의 이름을 출력: `echo "This is `uname` System."`
- ⑤ 출력 방향 바꾸기를 통해 ④번 명령의 실행 결과를 u.out 파일에 저장: `echo "This is `uname` System." > u.out`
- ⑥ date 명령의 실행 결과를 u.out 파일에 저장: `date > u.out`
- ⑦ u.out 파일에는 어떤 내용이 있을까?

04. 배시셀 환경 설정

■ 셀 변수와 환경 변수

- 셀 변수: 현재 셀에서만 사용할 수 있고 서브 셀로는 전달되지 않는 변수
- 환경 변수: 현재 셀뿐만 아니라 서브 셀로도 전달되는 변수

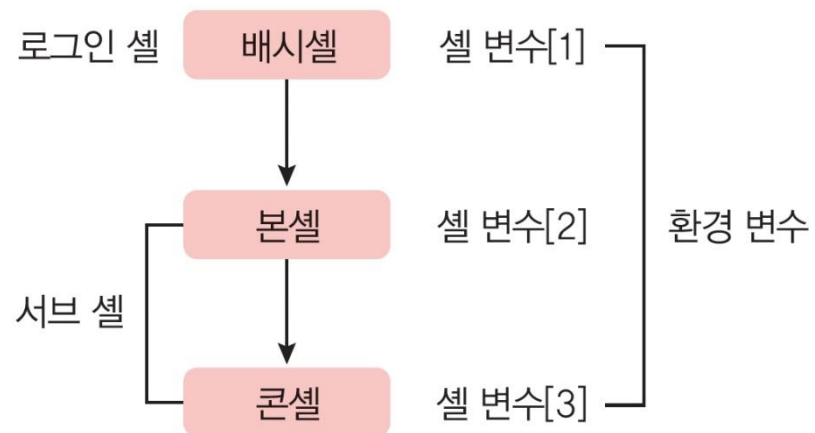


그림 4-3 셀 변수와 환경 변수

표 4-10 주요 셀 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 셀
LOGNAME	사용자 계정 이름		

04. 배시셀 환경 설정

■ 전체 변수 출력하기: set, env

- set: 셸 변수와 환경 변수 모두 출력
- env: 환경 변수만 출력

set 명령 실행

```
user1@myubuntu:~/linux_ex/ch4$ set
BASH=/bin/bash
BASHOPTS=checkwinsize:cmdhist:complete_fullquote:expand_aliases:extglob:extquote
:force_fignore
:globasciiranges:histappend:interactive_comments:login_shell:progcomp:promptvars:sou
rcepath
BASH_ALIASES=()
BASH_ARGC=([0]="0")
BASH_ARGV=()
(중략)
quote_readline ()
{
    local ret;
    _quote_readline_by_ref "$1" ret;
    printf %s "$ret"
}
```

04. 배시셸 환경 설정

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

```
SHELL=/bin/bash
```

```
PWD=/home/user1/linux_ex/ch4
```

```
LOGNAME=user1
```

```
XDG_SESSION_TYPE=tty
```

```
MOTD_SHOWN=pam
```

```
HOME=/home/user1
```

```
LANG=ko_KR.UTF-8
```

```
(생략)
```

```
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/  
games:/snap/bin
```

```
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
```

```
SSH_TTY=/dev/pts/1
```

```
_=/usr/bin/env
```

```
OLDPWD=/home/user1
```

env 명령 실행

- 특정 변수 출력하기: echo

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

04. 배시셀 환경 설정

■ 셸 변수 정의하기

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

env 명령으로
SOME 변수 검색안됨

04. 배시셀 환경 설정

■ 환경 변수 설정하기: export

환경 변수 정의

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

앞에서 정의한 SOME 변수를
export

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

변수 선언과 export를
한번에 가능

```
user1@myubuntu:~/linux_ex/ch4$ export SOME1=test1
user1@myubuntu:~/linux_ex/ch4$ echo $SOME1
```

env 명령으로
SOME1 변수검색됨

04. 배시셸 환경 설정

- 환경 변수를 다시 셸 변수로 변경하기: `export -n`

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

04. 배시셀 환경 설정

■ 변수 해제

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$
```

변수가 해제되어 echo 명령으로 출력되지 않음

05 앨리어스와 히스토리

05. 앨리어스와 히스토리

■ 앨리어스

- 기존 명령을 대신해서 다른 이름(별칭)을 붙일 수 있도록 하는 기능

alias

- **기능** 앨리어스를 생성한다.
- **형식** `alias 이름='명령'`
- **사용 예** `alias`: 현재 설정된 별칭 목록을 출력한다.
 `alias 이름='명령'`: 명령을 수정해 사용하는 경우다.
 `alias 이름='명령;명령2;...'`: 여러 명령을 하나의 이름으로 사용하는 경우다.

05. 앨리어스와 히스토리

- 앨리어스 출력하기

그냥 alias 입력하면 앨리어스 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'
```

05. 앨리어스와 히스토리

■ 앨리어스 설정: '앨리어스 이름=명령'

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

```
out1  temp  u.out
```

```
user1@myubuntu:~/linux_ex/ch4$ alias l.='ls -d .*'
```

→ 공백이 있으면 작은따옴표를 사용한다.

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

```
.  ..
```

```
user1@myubuntu:~/linux_ex/ch4$ alias rm='rm -i'
```

```
user1@myubuntu:~/linux_ex/ch4$ rm out1
```

```
rm: 일반 파일 'out1'를 제거할까요? n
```

05. 앨리어스와 히스토리

■ 앨리어스 삭제하기

unalias

- **기능** 앨리어스를 삭제한다.
- **형식** unalias 앨리어스

```
user1@myubuntu:~/linux_ex/ch4$ unalias l.  
user1@myubuntu:~/linux_ex/ch4$ unalias rm
```

05. 앨리어스와 히스토리

■ 앨리어스에 인자 전달하기

```
user1@myubuntu:~/linux_ex/ch4$ unalias cd
user1@myubuntu:~/linux_ex/ch4$ function cdpwd {
> cd $1;pwd
> }
user1@myubuntu:~/linux_ex/ch4$ cdpwd /tmp
/tmp
user1@myubuntu:/tmp$
```

→ 앨리어스를 해제한다.

→ 함수 입력을 시작한다.

→ 프롬프트가 >로 바뀐다. 내용을 입력한다.

→ 함수 입력을 종료한다.

05. 앨리어스와 히스토리

■ 히스토리

- 사용자가 이전에 입력한 명령을 다시 불러내서 사용하는 것

history

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

```
user1@myubuntu:/tmp$ history
```

(생략)

```
340 unalias cd
```

```
341 function cdpwd { cd $1;pwd; }
```

```
342 cdpwd /tmp
```

```
343 cdpwd
```

```
344 cdpwd /tmp
```

```
345 history
```

05. 엘리어스와 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.

05. 엘리어스와 히스토리

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

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

```
out1  temp  u.out
```

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

→ 바로 직전 명령을 재실행한다.

```
ls
```

```
out1  temp  u.out
```

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

(생략)

```
346 cd ~/linux_ex/ch4
```

```
347 ls
```

```
348 history
```

```
user1@myubuntu:~/linux_ex/ch4$ !347
```

→ 히스토리 번호로 재실행한다.

```
ls
```

```
out1  temp  u.out
```

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

→ 명령의 앞 글자로 재실행한다.

```
ls
```

```
out1  temp  u.out
```

05. 엘리어스와 히스토리

- 명령 편집 및 재실행하기

↑, ↓로 이전 명령을 불러온 다음 ←, →를 사용하여 명령의 잘못된 부분을 수정

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

hisdory로 잘못 입력

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

방향키로 명령 다시 불러오기

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


명령 수정하기

05. 엘리어스와 히스토리

- 히스토리 저장하기: 홈 디렉터리 아래의 숨김 파일인 .bash_history에 저장

```
user1@myubuntu:~/linux_ex/ch4$ more ~/.bash_history
pwd
ls
ls -a
ls /tmp
(생략)
```

214p. 따라해보기 : 환경변수, 앨리어스, 히스토리

- ① 셀 변수 TESTA를 설정하고 출력
- ② 본셀(sh)을 실행하여 서브 셀로 이동
- ③ 셀 변수 TESTA가 출력되는지 확인한다. 출력되지 않는 이유는 무엇일까?
- ④ 서브 셀에서 로그인 셀로 복귀
- ⑤ pwd 명령과 ls 명령을 묶어서 앨리어스 pls 만들기
- ⑥  로 이전 명령을 불러서 clear 명령에 대한 앨리어스 c를 만들기

06 프롬프트 설정

06 프롬프트 설정

■ 프롬프트를 저장한 환경변수: PS1

- 환경 변수 PS1의 값 설정을 수정하면 프롬프트가 바뀜

```
user1@myubuntu:~/linux_ex/ch4$ echo $PS1
```

```
\\[\\e]0;\\u@\\h: \\w\\a\\}${debian_chroot:+($debian_chroot)}\\u@\\h:\\w\\$
```

→ PS1의 현재 설정값

06 프롬프트 설정

■ 이스케이프 문자

- 예: \u
- 셸이 문자의 의미를 해석하여 실행

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	배시셸의 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 끝 부분을 표시한다.

06 프롬프트 설정

■ 프롬프트 변경 예

```
user1@myubuntu:~/linux_ex/ch4$ PROMPT=$PS1
```

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

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

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

```
myubuntu $ PS1='[\u \T] \!$ '  
[user1 12:17:55] 363$
```

현재 프롬프트 정보 저장

프롬프트 변경 1

프롬프트 변경 2

프롬프트 변경 3

프롬프트 변경 4

06 프롬프트 설정

■ 컬러 프롬프트 정의하기

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
녹색	32	42
갈색	33	43
파란색	34	44
보라색	35	45
청록색	36	46
흰색	37	47

표 4-14 프롬프트 특수 기능 번호

번호	기능
0	기본 색
1	굵게
4	흑백에서 밀줄
5	반짝임
7	역상
10	기본 폰트
38	밀줄 사용 가능
39	밀줄 사용 불가능

06 프롬프트 설정

■ 컬러 프롬프트 설정 예

```
[user1 04:16:59] 138$ PS1='\e[34mLinux $\e[0;0m '
```

Linux \$ → 파란색

```
Linux $ PS1=' \e[34;1mLinux $\e[0;0m '
```

Linux \$ → 파란색, 볼드

```
Linux $ PS1=' \e[31;4mLinux $\e[0;0m '
```

Linux \$ → 빨간색, 밑줄

```
Linux $ PS1=' \e[35;43m\u@\h $\e[0;0m '
```

user1@myubuntu \$ → 갈색 배경, 보라색 글자

프롬프트를 파란색으로 변경

프롬프트를 파란색, 볼드로 변경

프롬프트를 빨간색, 밑줄로 변경

프롬프트를 갈색 배경, 보라색 글자색으로 변경

06 프롬프트 설정

- 출력하지 않을 문자열 설정 추가하기

```
aaar1@myubuntu $ aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa → 중간에 이동
```

터미널의 화면 끝에 가기 전에 커서가 앞으로 돌아오고 같은 행을 겹쳐서 기록

```
user1@myubuntu $ PS1='\[\e[35;43m\]\u@h $\[\e[0;0m\]'  
user1@myubuntu $
```

설정 수정



```
user1@myubuntu $ ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff fff  
fffffffffffffffffffffffffffffffffffffffffffffffff^C  
user1@myubuntu $
```

```
user1@myubuntu $ PS1=$PROMPT  
user1@myubuntu:~/linux_ex/ch4$
```

프롬프트를 처음 설정으로 다시 지정하기

07 환경 설정 파일

07. 환경 설정 파일

■ 시스템 환경 설정 파일

표 4-15 배시셸의 시스템 환경 설정 파일

파일	기능
/etc/profile	<ul style="list-style-type: none">• 본셸이나 본셸과 호환되는 모든 셸에 공통으로 적용되는 환경 설정 파일이다.• 배시셸의 경우 /etc/bash.bashrc 파일을 실행한다.• 배시셸이 아닌 경우 프롬프트를 #(root 사용자)나 \$(일반 사용자)로 설정한다.• /etc/profile.d/*.sh 파일을 실행한다
/etc/bash.bashrc	<ul style="list-style-type: none">• 시스템 공통으로 적용되는 .bashrc 파일이다.• 기본 프롬프트를 설정한다.• sudo 명령과 관련된 힌트를 설정한다.
/etc/profile.d/*.sh	<ul style="list-style-type: none">• 언어나 명령별로 각각 필요한 환경을 설정한다.• 필요시 설정 파일을 추가한다.

07. 환경 설정 파일

```
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

표 4-16 배시셸의 사용자 환경 설정 파일

파일	기능
~/.profile	<ul style="list-style-type: none">• .bashrc 파일이 있으면 실행한다.• 경로 추가 등 사용자가 정의하는 환경 설정 파일이다.
~/.bashrc	<ul style="list-style-type: none">• 히스토리의 크기를 설정한다.• 기본 앨리어스나 함수 등을 설정한다.
~/.bash_logout	<ul style="list-style-type: none">• 로그아웃 시 실행할 필요가 있는 함수 등을 설정한다.
~/.bash_aliases	<ul style="list-style-type: none">• 사용자가 정의한 앨리어스를 별도 파일로 저장한다.

07. 환경 설정 파일

```
user1@myubuntu:~$ cat .profile      → .bash_profile 출력
# ~/.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"
```

```
user1@myubuntu:~$ cat .bash_logout  → .bash_logout 출력
# ~/.bash_logout: executed by bash(1) when login shell exits.
# when leaving the console clear the screen to increase privacy
if [ "$SHLVL" = 1 ]; then
    [ -x /usr/bin/clear_console ] && /usr/bin/clear_console -q
fi
```


07. 환경 설정 파일

■ 사용자 환경 설정 파일 만들기

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

■ 사용자 환경 설정 파일 적용하기: .(점) 또는 source 명령

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

```
user1@myubuntu:~$ source .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
본셸	/etc/profile	\$HOME/.profile	—	○		
콘셸	/etc/profile	\$HOME/.profile	—	○		
		\$HOME/.kshrc	ENV 변수 설정	○	○	
C셸	/etc/.login	\$HOME/.login	—	○		
		\$HOME/.cshrc	—	○	○	
		\$HOME/.logout	—			○

226p. 따라해보기 : 사용자 환경 설정 파일 수정하기

- ① .profile 파일을 vi로 연다.
- ② .profile 파일의 내용에 행 번호가 보이도록 한다 :set nu
- ③ 27행 다음에 경로를 추가하고 파일을 저장한 후 종료
- ④ .bashrc 파일을 vi로 열고 행 번호가 보이도록 한다.
- ⑤ 62행의 맨 앞에 #을 붙이고 62행의 다음에 프롬프트 설정을 추가한 후 저장
- ⑥ .profile 파일을 실행 -> .bashrc 파일은 .profile에 의해 자동으로 실행되어 다음과 같이 프롬프트가 바뀐다.
- ⑦ 경로에 /etc 디렉터리가 추가되었는지 확인

Thank You !