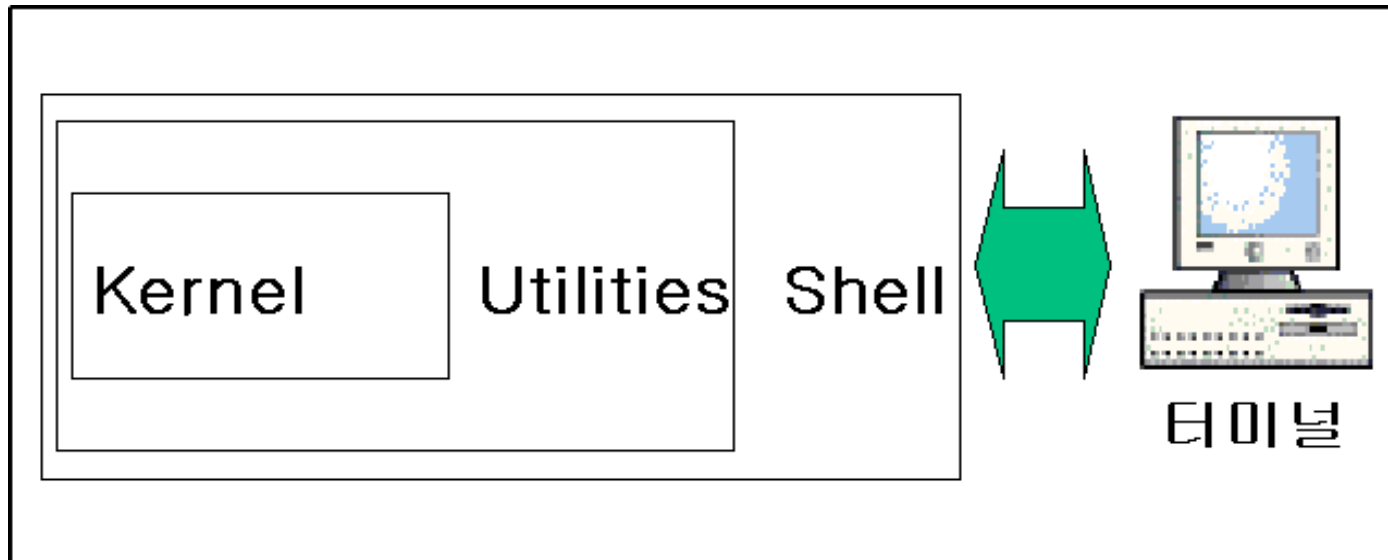

4장. 셸(shell)을 이용한 환경설정

개요

■ 셸(shell)

- ◆ 명령어 해석기(Command Interpreter) - 사용자가 입력한 명령어를 읽어서 해석하는 프로그램으로
- ◆ 커널(Kernel) 주위를 둘러싸며 사용자와의 인터페이스 담당



UNIX/LINUX 의 셸

■ 셸의 종류

◆ Bourne Shell

- 가장 오래된 유닉스 셸
- 개발자 이름을 따서 지었음

◆ C Shell

- C 언어와 유사한 형태
- 대학이나 연구 단체에 있는 유닉스 시스템의 기본 셸로 제공
- 대부분의 유닉스 사용자들이 사용

◆ Korn Shell

- Bourne Shell 을 확장
- Bourne Shell 의 모든 명령어들 인식
- 특히 명령어 기억, 별명(Alias)기능, 제어기능 을 가지고 있음

◆ Bash

- Bourne Shell 을 확장
- 리눅스의 기본 셸로서 가장 많이 사용

사용자 셸 바꾸기

- 로그인 셸(login shell) – 리눅스에 로그인 시 사용되는 셸
- 자신의 셸 확인

```
[cprog2@seps5 cprog2]$ echo $SHELL  
/bin/bash  
[cprog2@seps5 cprog2]$
```

```
[cprog2@seps5 cprog2]$ cat /etc/passwd  
:  
cprog2:x:1003:1003:./home/cprog2:/bin/bash  
:
```

셸의 종류	이름
bourne shell	sh
bourne again shell	bash
korn shell	ksh
C shell	csch
TC shell	tcsh

사용자 셸 바꾸기

■ 셸 변경

1. 로그인 셸을 그대로 둔 채 잠깐 사용하기 위해 셸을 변경

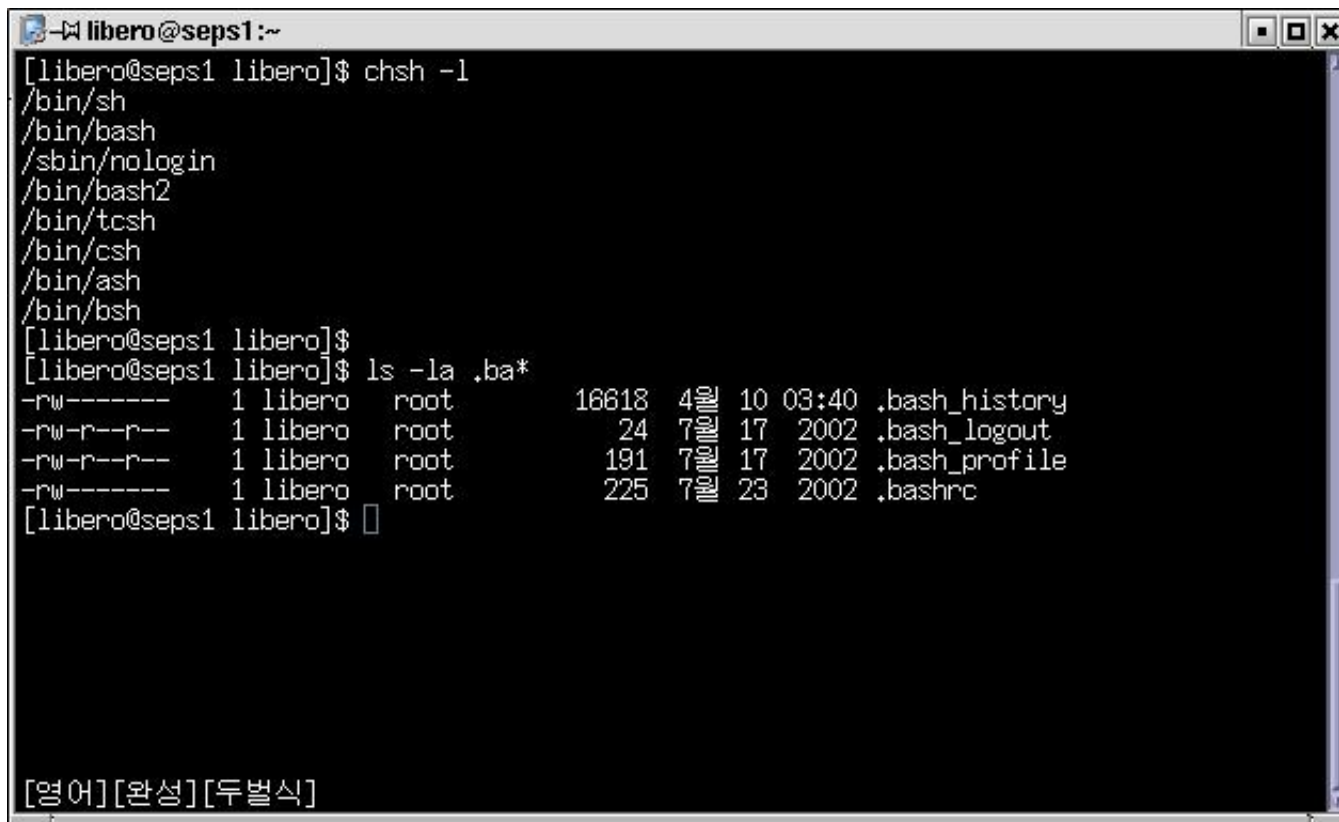
```
[cprog2@seps5 cprog2]$ sh  
[cprog2@seps5 ~]$ exit  
exit  
[cprog2@seps5 cprog2]$
```

2. 로그인 셸을 바꾸는 경우

```
[cprog2@seps5 cprog2]$ chsh  
Changing shell for cprog2.  
Password:  
New shell [/bin/bash]: /bin/csh  
Shell changed.  
[cprog2@seps5 cprog2]$
```

사용자 셸 바꾸기

■ 시스템 내의 셸 확인



A terminal window titled "libero@seps1:~" showing the process of changing the user's shell. The user runs "chsh -l" and lists available shells: /bin/sh, /bin/bash, /sbin/nologin, /bin/bash2, /bin/tcsh, /bin/csh, /bin/ash, and /bin/bsh. Then, the user runs "ls -la .ba*" to list bash-related files in their home directory. The output shows four files: .bash_history (16618 bytes, 4월 10 03:40), .bash_logout (24 bytes, 7월 17 2002), .bash_profile (191 bytes, 7월 17 2002), and .bashrc (225 bytes, 7월 23 2002). The terminal also shows the prompt "[영어][완성][두벌식]" at the bottom.

```
libero@seps1:~  
[libero@seps1 libero]$ chsh -l  
/bin/sh  
/bin/bash  
/sbin/nologin  
/bin/bash2  
/bin/tcsh  
/bin/csh  
/bin/ash  
/bin/bsh  
[libero@seps1 libero]$  
[libero@seps1 libero]$ ls -la .ba*  
-rw----- 1 libero root 16618 4월 10 03:40 .bash_history  
-rw-r--r-- 1 libero root 24 7월 17 2002 .bash_logout  
-rw-r--r-- 1 libero root 191 7월 17 2002 .bash_profile  
-rw----- 1 libero root 225 7월 23 2002 .bashrc  
[libero@seps1 libero]$  
[영어][완성][두벌식]
```

셸에서 사용하는 주요 명령어

■ 정보 표시(echo)

◆ 유닉스에서 주어지는 인수(Parameter)를 표준출력에 표시하는 명령

인수	기능
\a	경고문자
\b	백스페이스
\c	새로운 라인 없이 프린터라인 사용
\f	폼 피드(form feed)
\n	새로운 라인
\r	캐리지 리턴(carriage return)
\t	탭(tab)

셸에서 사용하는 주요 명령어

■ echo 사용 예

```
libero@seps1:~ <2>  
[libero@seps1 libero]$ echo $SHELL  
/bin/bash  
[libero@seps1 libero]$ echo 표준 출력으로 출력한다.  
표준 출력으로 출력한다.  
[libero@seps1 libero]$ echo -e -n "출력에 NewLine를 추가하지 않습니다"  
[libero@seps1 libero]$ 지 않습니다  
[libero@seps1 libero]$ echo -e 새로운 라인 \\n NewLine # 새로운 라인에서 시작  
새로운 라인  
NewLine  
[libero@seps1 libero]$ echo -e 탭을 외움 \\t tab # 탭을 이용하여 외움  
탭을 외움 tab  
[libero@seps1 libero]$
```


셸에서 사용하는 주요 명령어

■ 메타(Meta Character) 문자

- ◆ 셸에서 특수하게 인식하는 문자들
- ◆ 셸 문장 해석 시 메타문자를 확인하고, 존재하면 이 문자에 대한 특별한 기능 수행
- ◆ 메타문자 기능을 없애는 방법 – 바로 앞에 “\” 삽입

■ 메타문자 사용 예

```
libero@seps1:~ <2>
[libero@seps1 libero]$ echo 메타문자
메타문자
[libero@seps1 libero]$ echo 메타문자 > imsi.txt      # 파일로 저장
[libero@seps1 libero]$ ls -la imsi.txt
-rw-r--r--  1 libero  root          9  8월 10 05:33 imsi.txt
[libero@seps1 libero]$ more imsi.txt
메타문자
[libero@seps1 libero]$ echo -e 메타문자 \> imsi.txt # 메타문자 특성 지움
메타문자 > imsi.txt
[libero@seps1 libero]$
```

셸에서 사용하는 주요 명령어

■ 표준 입·출력 제어(Redirection)

기호	의미
>	표준출력을 파일로 기록
>>	표준출력을 파일의 끝에 추가
<	파일로부터 입력을 읽음

```
libero@seps1:~ <2>
[libero@seps1 libero]$ ls
Desktop Desktop1 bashrc hello.c image imsi.txt mask test
[libero@seps1 libero]$ cat >imsi.txt          # 표준 출력으로 imsi.txt를 생성
새로운 파일을 만듭니다.
^D
[libero@seps1 libero]$ cat imsi.txt
새로운 파일을 만듭니다.
[libero@seps1 libero]$ cat >> imsi.txt        # 표준 출력으로 imsi.txt에 덧붙임
새로운 파일을 만들어서 기존 파일에 덧붙입니다.
^D
[libero@seps1 libero]$ ls
Desktop Desktop1 bashrc hello.c image imsi.txt mask test
[libero@seps1 libero]$ cat imsi.txt
새로운 파일을 만듭니다.
새로운 파일을 만들어서 기존 파일에 덧붙입니다.
[libero@seps1 libero]$
```

셸에서 사용하는 주요 명령어

■ 와일드 문자 사용

대표문자	의미
*	모든 문자열
?	한 문자와 일치
[..]	적어도 [] 안의 한 문자와 일치, “-” 를 이용하여 범위 지정 가능

셸에서 사용하는 주요 명령어

■ 사용 예

```
libero@seps1:~/test
[libero@seps1 test]$ ls
5      for.sh      integer.sh  special.sh  test1      text1.c
abc    hello.txt      reverse.sh  special4741.sh test2      text2.c
case.sh if.sh          sample.sh  test.c      text.c     zzz
[libero@seps1 test]$ ls *.txt      # 확장자가 "txt" 로 끝나는 파일 출력
hello.txt
[libero@seps1 test]$ ls ????.*     # 파일명이 4개의 문자로 구성된 파일 출력
case.sh test.c text.c
[libero@seps1 test]$ ls ?a*. *     # 두 번째 문자가 "a" 인 파일 출력
case.sh sample.sh
[libero@seps1 test]$ ls [cd]*      # 파일의 첫문자가 "c", "d" 로 시작하는 파일
case.sh
[libero@seps1 test]$ ls [a-c]*     # 파일 첫문자가 "a", "b", "c" 로 시작하는 파일
abc case.sh
[libero@seps1 test]$ ls test/*     # "test" 로 시작하는 디렉토리의 모든 파일
test2/test.c test2/text.c test2/text.tar.gz test2/text1.c test2/text2.c
[libero@seps1 test]$ ls [A-Za-z]* # 영문자로 시작하는 모든 파일
abc    hello.txt      reverse.sh  special4741.sh text.c     zzz
case.sh if.sh          sample.sh  test.c      text1.c
for.sh  integer.sh  special.sh test1      text2.c

test2:
test.c text.c text.tar.gz text1.c text2.c
[libero@seps1 test]$
```

[영어][완성][두벌식]

셸에서 사용하는 주요 명령어

■ 파이프 라인(Pipe Line)

- ◆ 한 명령의 출력을 다른 명령이나 셸의 입력으로 연결하여 사용
- ◆ 여러 명령을 연결하여 복잡하거나 큰 작업을 빠르고 쉽게 구성 가능
- ◆ 두 명령은 “|” 를 사용하여 연결



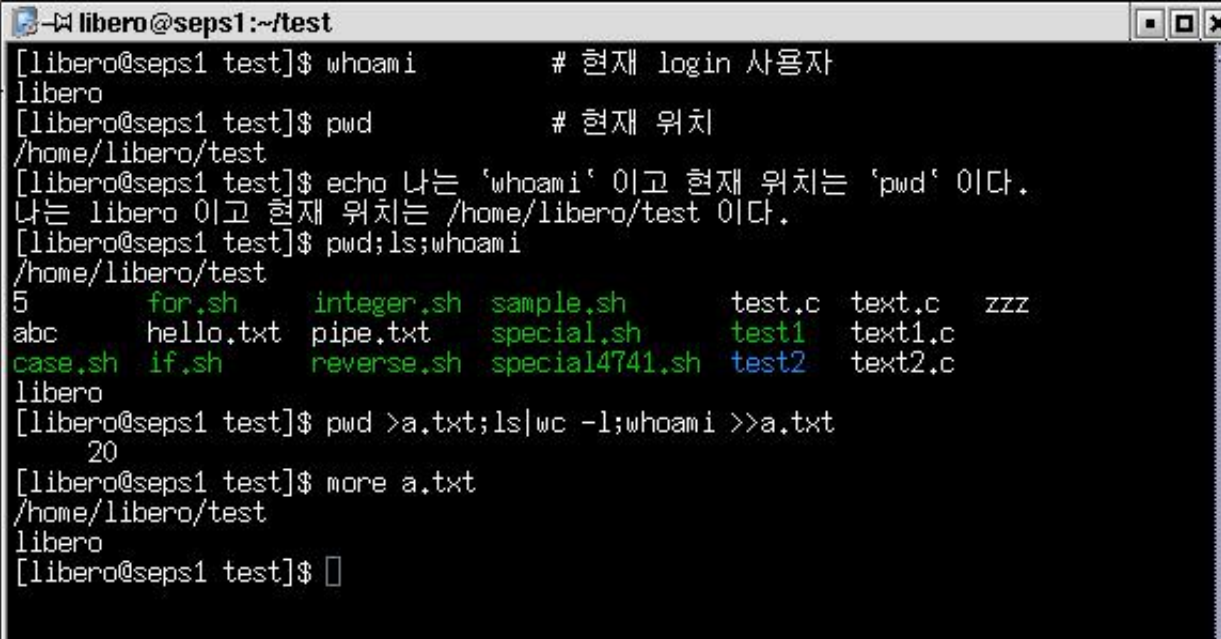
◆ 사용 예

```
libero@seps1:~/test
[libero@seps1 test]$ cat /etc/passwd | grep libero
libero:x:500:0::/home/libero:/bin/bash
[libero@seps1 test]$ cat /etc/passwd | grep f98* | tee pipe.txt | sort # 결과 저장
f97rw222:x:501:500::/home/f97rw222:/bin/bash
f98ri048:x:506:500::/home/f98ri048:/bin/bash
f98rw060:x:511:500::/home/f98rw060:/bin/bash
f99ru333:x:507:500::/home/f99ru333:/bin/bash
[libero@seps1 test]$ cat pipe.txt
f97rw222:x:501:500::/home/f97rw222:/bin/bash
f98ri048:x:506:500::/home/f98ri048:/bin/bash
f99ru333:x:507:500::/home/f99ru333:/bin/bash
f98rw060:x:511:500::/home/f98rw060:/bin/bash
[libero@seps1 test]$
```

셸에서 사용하는 주요 명령어

■ 명령어 관리(Command)

- ◆ “” (악센트기호) : 문장 내에서 명령어 해석하여 결과를 표준출력
- ◆ “grep” 명령 : 특정 문자열을 담은 파일 검색
- ◆ “;” : 한 행에서 실행순서(Sequences) 대로 명령어 실행
- ◆ 사용 예



```
libero@seps1:~/test
[libero@seps1 test]$ whoami          # 현재 login 사용자
libero
[libero@seps1 test]$ pwd              # 현재 위치
/home/libero/test
[libero@seps1 test]$ echo 나는 'whoami' 이고 현재 위치는 'pwd' 이다.
나는 libero 이고 현재 위치는 /home/libero/test 이다.
[libero@seps1 test]$ pwd;ls;whoami
/home/libero/test
5      for.sh      integer.sh  sample.sh  test.c    text.c    zzz
abc     hello.txt    pipe.txt   special.sh test1      text1.c
case.sh if.sh        reverse.sh special4741.sh test2     text2.c
libero
[libero@seps1 test]$ pwd >a.txt;ls|wc -l;whoami >>a.txt
20
[libero@seps1 test]$ more a.txt
/home/libero/test
libero
[libero@seps1 test]$
```

셸에서 사용하는 주요 명령어

■ 조건부 실행

◆ 반환값 : "0"의 값을 반환하면 명령어 성공, "1"의 값을 반환하면 명령어 실패를 나타냄

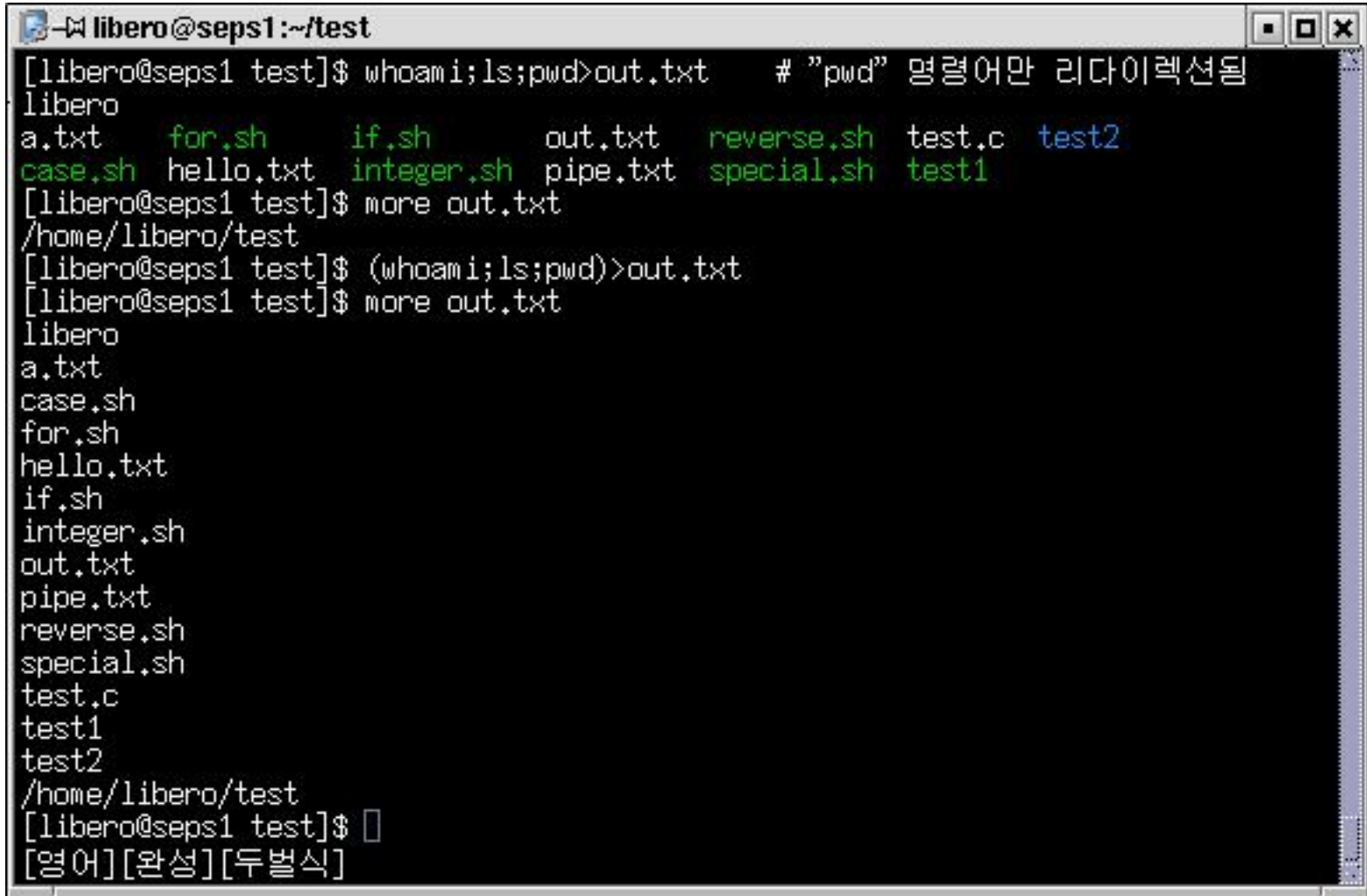
문자	의미
&&	이전 명령이 정상 종료인 경우에만 다음 명령 실행
	이전 명령이 비정상 종료인 경우에만 다음 명령 실행

■ 사용 예

```
libero@seps1:~/test
[libero@seps1 test]$ whoo||echo Ok      # 비정상 종료이므로 우측 실행
-bash: whoo: command not found
Ok
[libero@seps1 test]$ whoo&&echo Ok      # 비정상 종료이므로 우측 미실행
-bash: whoo: command not found
[libero@seps1 test]$ whoami|echo Ok     # 정상 종료이므로 우측 미실행
libero
[libero@seps1 test]$ whoami&&echo Ok     # 정상 종료이므로 우측 실행
libero
Ok
[libero@seps1 test]$
```


셸에서 사용하는 주요 명령어

■ 사용 예



```
libero@seps1:~/test
[libero@seps1 test]$ whoami;ls;pwd>out.txt    # "pwd" 명령어만 리다이렉션됨
libero
a.txt    for.sh    if.sh    out.txt  reverse.sh  test.c  test2
case.sh  hello.txt  integer.sh  pipe.txt  special.sh  test1
[libero@seps1 test]$ more out.txt
/home/libero/test
[libero@seps1 test]$ (whoami;ls;pwd)>out.txt
[libero@seps1 test]$ more out.txt
libero
a.txt
case.sh
for.sh
hello.txt
if.sh
integer.sh
out.txt
pipe.txt
reverse.sh
special.sh
test.c
test1
test2
/home/libero/test
[libero@seps1 test]$
```

[영어][완성][두벌식]

셸에서 사용하는 주요 명령어

■ 유닉스 작업 처리

◆ 포그라운드(Foreground) 작업 처리

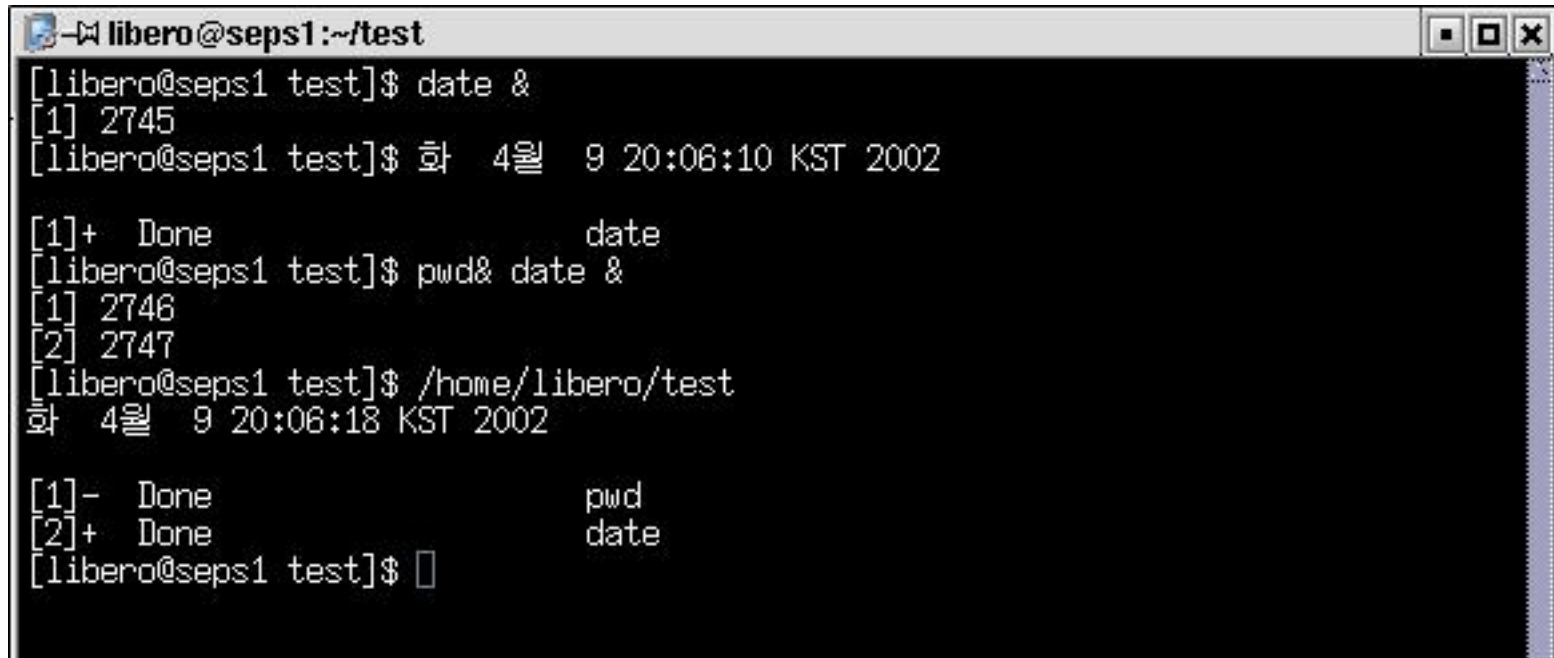
- 사용자가 명령어를 입력하면 바로 그 결과를 표준출력으로 보여주고 새로운 명령어를 받아들일 수 있도록 프롬프트가 표시됨

◆ 백그라운드(Background) 작업 처리

- 작업 결과를 중간에 볼 필요없고 결과만을 원할 때, 또는 시간이 많이 걸리는 작업은 백그라운드로 작업하여 프로세서 사용을 극대화
- 해당 명령어나 셸에 "&"를 붙여 백그라운드 작업 실행
- 현재 포그라운드 실행중인 작업 : **Ctrl-Z** 하여 중단한 다음 **bg** 명령 실행
- 백그라운드 작업이 시작되면 시스템은 별도의 프로세스를 할당하여 그 프로세스 **ID**를 사용자에게 보여주고 바로 명령 수행 상태로 변경함

셸에서 사용하는 주요 명령어

■ 백그라운드 작업 사용 예

A terminal window titled 'libero@seps1:~/test' showing a sequence of commands and their outputs. The user runs 'date &', which starts a background process. The prompt changes to '[1] 2745'. Then the user runs '화 4월 9 20:06:10 KST 2002', which prints the date. The prompt changes to '[1]+ Done'. Then the user runs 'date', which prints the date. The prompt changes to '[libero@seps1 test]\$'. Then the user runs 'pwd& date &', which starts two background processes. The prompt changes to '[1] 2746'. Then the user runs 'date', which prints the date. The prompt changes to '[2] 2747'. Then the user runs '/home/libero/test', which prints the path. The prompt changes to '화 4월 9 20:06:18 KST 2002'. Then the user runs 'pwd', which prints the path. The prompt changes to '[1]- Done'. Then the user runs 'date', which prints the date. The prompt changes to '[2]+ Done'. Finally, the user runs '[libero@seps1 test]\$' and the prompt changes to '[libero@seps1 test]\$'.

```
libero@seps1:~/test
[libero@seps1 test]$ date &
[1] 2745
[libero@seps1 test]$ 화 4월 9 20:06:10 KST 2002

[1]+  Done                  date
[libero@seps1 test]$ pwd& date &
[1] 2746
[2] 2747
[libero@seps1 test]$ /home/libero/test
화 4월 9 20:06:18 KST 2002

[1]-  Done                  pwd
[2]+  Done                  date
[libero@seps1 test]$
```

사용자 환경 설정

■ 환경변수

- ◆ 사용자가 각자 원하는 형태로 자신의 컴퓨터 사용 환경을 설정해 두기 위한 변수들
- ◆ 주요 공통 환경 변수들

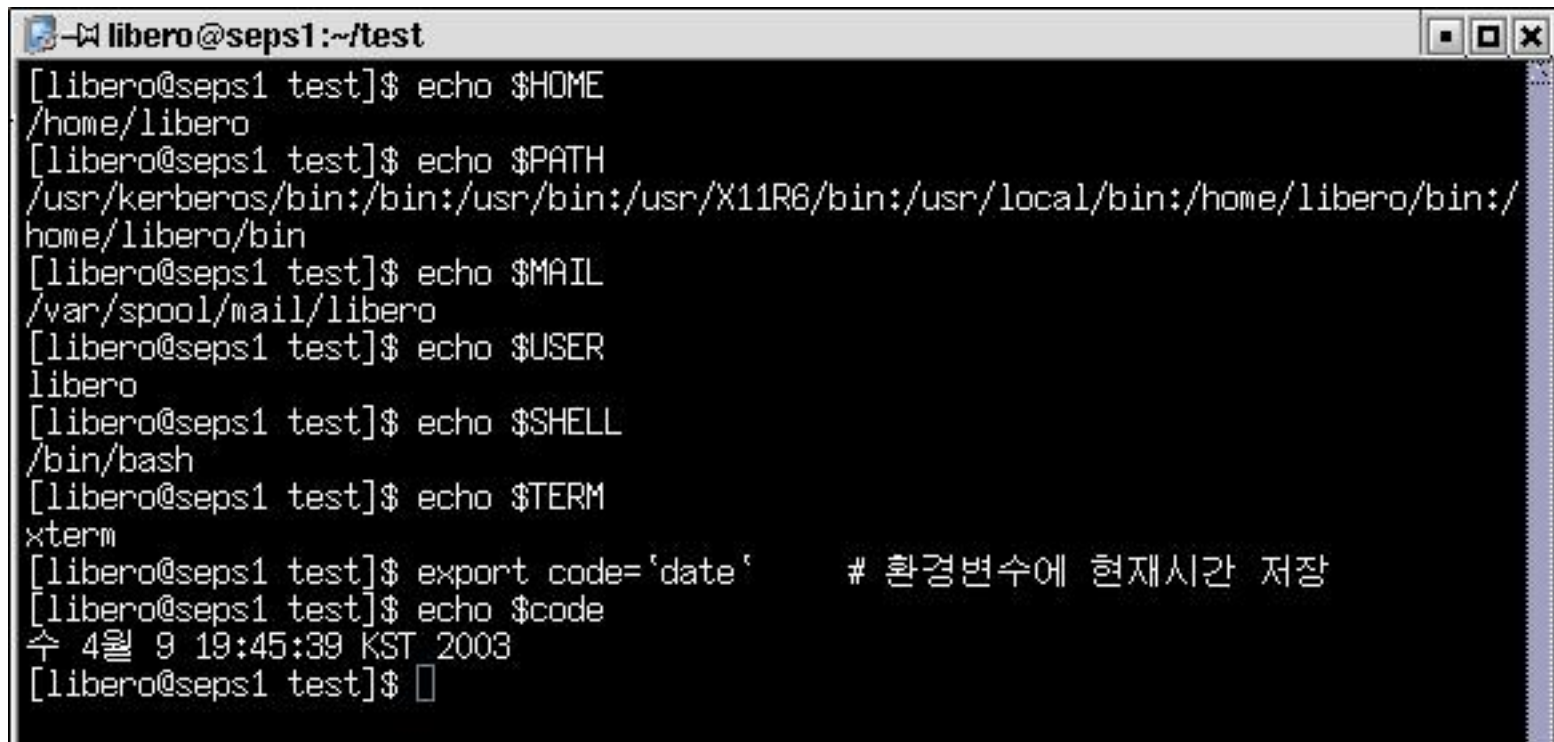
변수	의미
HOME	홈 디렉토리의 절대경로
PATH	명령어 탐색 경로
USER	사용자 이름(login name)
SHELL	로그인 쉘의 절대 경로
TERM	터미널 유형

- ◆ “echo” 명령 : 환경변수 내용 확인
- ◆ "set" 명령 : 환경 변수 설정 및 현재 환경변수의 목록 출력

사용자 환경 설정

■ 환경 변수 출력 예

```
[cprog2@seps5 cprog2]$ echo $PATH
./usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin:/opt/IBMJava2-13/bin:/usr/local/bin
[cprog2@seps5 cprog2]$
```



```
libero@seps1:~/test
[libero@seps1 test]$ echo $HOME
/home/libero
[libero@seps1 test]$ echo $PATH
/usr/kerberos/bin:/bin:/usr/bin:/usr/X11R6/bin:/usr/local/bin:/home/libero/bin:/home/libero/bin
[libero@seps1 test]$ echo $MAIL
/var/spool/mail/libero
[libero@seps1 test]$ echo $USER
libero
[libero@seps1 test]$ echo $SHELL
/bin/bash
[libero@seps1 test]$ echo $TERM
xterm
[libero@seps1 test]$ export code='date'      # 환경변수에 현재시간 저장
[libero@seps1 test]$ echo $code
수 4월 9 19:45:39 KST 2003
[libero@seps1 test]$
```

사용자 환경 설정

■ 환경변수 출력 예

```
[cprog2@seps5 cprog2]$ set
BASH=/bin/bash
BASH_VERSINFO=([0]="2" [1]="05b" [2]="0" [3]="1" [4]="release" [5]="i686-pc-linux-gnu")
BASH_VERSION='2.05b.0(1)-release'
COLORS=/etc/DIR_COLORS
COLUMNS=80
DIRSTACK=()
EUID=1003
GROUPS=()
G_BROKEN_FILENAMES=1
HISTFILE=/home/cprog2/.bash_history
HISTFILESIZE=1000
HISTSIZ=1000
HOME=/home/cprog2
HOSTNAME=seps5
HOSTTYPE=i686
IFS=$' \t\n'
INPUTRC=/etc/inputrc
JAVA_HOME=/opt/IBMJava2-13
JLESSCHARSET=ko
KDEDIR=/usr
LANG=C
LC_ALL=ko_KR.euckr
:
```

셸 매개변수

■ 셸 매개변수(parameters)

- ◆ 셸이 수행될 때 인자들이 매개변수로 전달됨
- ◆ 매개변수 관련 기호들

이름	의미
\$\$	셸의 프로세스 ID
\$0	셸 스크립트의 이름(실행파일)
\$n	명령 행의 n 번째 인수
\$*	모든 인수의 목록
\$#	인수의 갯수

셸 매개변수

■ 셸 매개변수 사용 예

```
libero@seps1:~/test
[libero@seps1 test]$ cat >reverse.sh          # 셸 스크립트 작성
echo $3 $2 $1^D
[libero@seps1 test]$ chmod +x reverse.sh      # 실행 허가 여부
[libero@seps1 test]$ ./reverse.sh first second third
third second first
[libero@seps1 test]$ cat special.sh           # 셸 스크립트 작성
#!/bin/sh
# 이것은 셸 프로그램입니다.
echo "첫번째 인수" $1
echo "두번째 인수" $2
echo "모든 인수" $*
echo "프로세서 ID" $1.$$                      # 명령어 리다이렉션
uname>$1.$$                                   # 파일 목록 표시
rm $1.$$                                       # 파일 삭제
[libero@seps1 test]$ ./special.sh first second third # 3개의 인수를 가정
첫번째 인수 first
두번째 인수 second
모든 인수 first second third
프로세서 ID first.2852
[libero@seps1 test]$
```

셸에서의 인용부호 사용(Quoting)

■ 셸의 대표문자를 금지할 필요가 있을 경우

◆ 보통 인용부호인 “ 또는 ‘ 를 사용

◆ 인용부호 기호들

인용부호	의미
“ ”	대표문자의 대치와 변수대치, 명령어 대치 금지
“ ”	대표문자의 대치만을 금지
중첩	인용부호가 중첩될 때에는 바깥쪽의 인용부호만 효력을 가짐

◆ 인용부호 사용 예

```
libero@seps1:~/test
[libero@seps1 test]$ echo *                # 현재 디렉토리의 모든 파일 출력
a.txt case.sh for.sh hello.txt if.sh integer.sh out.txt pipe.txt reverse.sh spec
ial.sh test.c test1 test2
[libero@seps1 test]$ echo "*"              # 특수문자 기능 금지
*
[libero@seps1 test]$ echo '*'
*
[libero@seps1 test]$ name=libero            # 새로운 변수에 이름을 할당
[libero@seps1 test]$ echo "나의 이름 $name & 날짜 `date`"
나의 이름 libero & 날짜 수 4월 9 19:56:17 KST 2003
[libero@seps1 test]$
```


사용자 환경 파일

■ 사용자가 처음 로그인할 때, 셸을 수행할 때, 로그아웃 할 때 시스템이 자동으로 수행하는 환경 파일

◆ 각 주요 셸의 사용자 환경 파일

셸 이름	환경파일 이름		
	로그인	셸 수행	로그아웃
csh(tcsh)	.login	.cshrc	.logout
bash	.bash_profile	.bashrc	.bash_logout

C 셸 환경 파일

■ .login – 로그인 시 수행 파일

◆ .login 파일 예

```
# .login 파일의 예  
stty erase ^H  
date
```

◆ 수행 결과

```
login: cprog2  
Password:  
Last login: Wed Feb  4 20:34:08  
Wed Feb 4 22:19:59 KST 2004  
[cprog2@seps5 ~]$
```

C 셸 환경 파일

■ .login 파일의 예

```
echo "-----"  
echo "  안녕하세요 리눅스 시스템에 오신걸 환영합니다."  
echo "-----"  
echo "오늘날짜는" `date` "입니다."
```

◆ 수행 결과

```
login: cprog2  
Password:  
Last login: Wed Feb  4 20:34:08  
  
-----  
  안녕하세요 리눅스 시스템에 오신걸 환영합니다.  
-----  
  
오늘날짜는 Wed Sep 4 22:19:59 KST 2004 입니다.  
[cprog2@seps5 ~]$
```

C 셸 환경 파일

■ **.logout** – 로그아웃 시 수행 파일

■ **.logout** 파일의 예

```
# .logout 예제
echo "*****"
echo $USER "님 안녕히 가십시오."
echo " "
echo "로그아웃 시간은" `date` "입니다."
echo '*****'
```

◆ 수행 결과

```
[cprog2@seps5 ~]$ logout
logout
*****
cprog2 님 안녕히 가십시오.
로그아웃 시간은 금 9월 13 18:34:09 KST 2004 입니다.
*****
Connection closed by foreign host.
```

C 셸 환경 파일

■ .cshrc – C 셸이 수행할 때마다 수행

■ .cshrc 파일의 예

1 # 간단한 .cshrc 예제 파일	←	주석문
2 umask 077	←	파일 생성 시 파일의 퍼미션 지정
3 set history=10	←	명령 history의 화면 출력 개수
4 set savehist = 100	←	명령 history 저장 개수
5 set filec	←	파일명 완성 기능
6 stty erase ^H	←	tty 설정 명령(erase키를 ^H로 지정)
7 set path=(. /bin /usr/sbin /usr/bin /usr/lib /usr/local/bin)		
8 #----- Aliases -----		
9 alias md 'mkdir'		
10 alias rd 'rmdir'		
11 alias ls 'ls -l'		
12 alias cls 'clear'		
13 alias setprompt 'set prompt="(`pwd`)"'		
14 alias cd 'cd W!*; setprompt'		
15 setprompt		

C 셸에서 별명(alias) 사용하기

■ 별명 지정

◆ “alias” 명령 사용

alias	별명	'원래명령어'
-------	----	---------

◆ 지정된 별명 목록 출력

```
[/home/cprog2] alias
cd      cd !*; setprompt
cls     clear
gc      gcc !*.c -o !*
l.      ls -d .* --color=tty
ll      ls -l --color=tty
ls      ls -l
md      mkdir
rd      rmdir
setprompt set prompt="[\`pwd`]"
telnet  telnet -8
vi      vim
[/home/cprog2]
```

◆ 별명 지정 사용 예

alias	md	'mkdir'
-------	----	---------

C 셸에서 별명(alias) 사용하기

■ 여러 명령어를 하나의 별명으로 사용

◆ 형식

```
alias    별명    '명령어1;명령어2;명령어3;....'
```

◆ 사용 예 1

```
[/home/cprog2]alias dir 'pwd;ls'
[/home/cprog2]dir
/home/cprog2
Desktop bin cshrc.org hyomin mail myprog test
[/home/cprog2]
```

◆ 사용 예 2 (.cshrc 파일)

```
14 alias    cd    'cd W!*; setprompt'
```

!* 는 모든 명령 인자를 나타냄

C 셸 프롬프트 바꾸기

■ C 셸 프롬프트(prompt) 바꾸기

◆ 환경변수 **prompt** 변경

◆ 변경 예

```
%  
%set prompt="$USER>"  
cprog2>
```

```
%  
% set prompt="[`pwd`]"  
[/home/cprog2]
```

```
%  
% alias setprompt 'set prompt="[`pwd`]"'  
% alias cd 'cd W!*;setprompt'  
% setprompt  
[/home/cprog2]cd test  
[/home/cprog2/test]
```


BASH 환경파일

■ Bash

◆ Bourne 셸을 확장한 셸

◆ 강력한 셸 프로그래밍 - 다양한 작업 제어 기능과 연산 기능 제공

■ .bashrc – bash 셸이 수행할 때마다 실행

◆ .bashrc 파일의 예

```
1 # 간단한 .bashrc 예제 파일
2 umask 077
3 export HISTSIZE=10
4 export HISTFILESIZE=100
5 stty erase ^H
6 export PATH=./bin:/usr/sbin:/usr/bin:/usr/lib:/usr/local/bin
7 #----- Aliases -----
8 alias      md='mkdir'
9 alias      rd='rmdir'
10 alias      ls='ls -l'
11 alias      cls='clear'
12 #----- End Alias -----
13 export     PS1="[ $PWD ]"
```

BASH 환경파일

■ BASH 환경변수 설정

◆ 환경변수이름=값 : 환경변수 지정

◆ “export” 명령 : 환경변수를 BASH 에 알림

```
환경변수이름=값  
export 환경변수이름  
또는  
export 환경변수이름=값
```

■ 별명(alias) 지정

◆ 지정 형식

```
alias 별명='원래명령어'
```

◆ 사용 예

```
alias cls 'clear' #csh일 때  
alias cls='clear' #bash일 때
```

BASH 셸 프롬프트 바꾸기

■ BASH 셸 프롬프트(prompt) 바꾸기

◆ 환경변수 PS1 변경

◆ 변경 예

```
bash$ export PS1='Cprogram>'
Cprogram>
```

```
13 export PS1='[$PWD]'
```

```
bash$
bash$export PS1='[$PWD]'
```

```
[/home/cprog2]cd test
[/home/cprog2/test]
```

BASH 셸 프롬프트 바꾸기

■ BASH 셸 프롬프트에 사용하는 특수문자

특수문자	내용
<code>\h</code>	호스트 이름
<code>\u</code>	사용자 이름
<code>\w</code>	작업 디렉토리의 절대경로
<code>\W</code>	작업 디렉토리의 이름
<code>\d</code>	오늘 날짜
<code>\t</code>	현재 시간
<code>\#</code>	사건 번호
<code>!</code>	히스토리 번호

◆ 사용 예

```
bash$ export PS1='[\h:\Ww]'  
[seps5:~]  
[seps5:~]cd test  
[seps5:~/test]
```

로그인 쉘의 재수행

■ 로그인 쉘의 재수행

◆ “source” 명령 수행

```
[seps5:~]source .bashrc          //bash
또는
[seps5:~]source .cshrc           //csh
```

히스토리(history) 기능 사용

■ 히스토리 기능

◆ 사용자가 이전에 사용한 명령들을 기록하고 나중에 다시 불러 사용

◆ “**history**” 명령 : 명령 히스토리 출력

```
[cprog2@seps5 cprog2]$ history
1005 vi .ddd
1006 ls -al
1007 vi .exrc
1008 rm .exrc
1009 rm .exrc
1010 ls
1011 ls -al
1012 ls -al |more
1013 vi .bash_history
1014 history
[cprog2@seps5 cprog2]$
```

히스토리(history) 기능 사용

■ 히스토리 기능을 이용한 명령 수행 예

◆ “!1010” : 히스토리 목록 중 1010 번에 해당하는 명령 수행

```
[cprog2@seps5 cprog2]$ !1010  
ls  
Desktop Mail Test bashrc book2 cprog2 packages struct temp test.txt
```

◆ “!!” : 바로 직전 명령 수행

```
[cprog2@seps5 cprog2]$ !!  
ls  
Desktop Mail Test bashrc book2 cprog2 packages struct temp test.txt
```

◆ “!h” : 최근 명령 중 “h” 로 시작하는 명령 수행

```
[cprog2@seps5 cprog2]$ !h  
history  
1009 rm .exrc  
1010 ls  
1011 ls -al  
1012 ls -al |more  
1013 ls  
1014 history  
[cprog2@seps5 cprog2]$
```