# 목차

# 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 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

# 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

# 02 셸 기본 사용법

## 기본 셸 바꾸기

**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:~$ 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                   절대 경로로 입력하라는 메시지가 출력된다.
user1@myubuntu:~$ chsh -s /bin/sh user1
암호:
user1@myubuntu:~$ tail /etc/passwd
(생략)
user1:x:1000:1000:user1,,,:/home/user1:/bin/sh
sshd:x:116:65534::/var/run/sshd:/usr/sbin/nologin
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

# 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 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically
linked (uses shared libs), for GNU/Linux 2.6.24, BuildID[sha1]=0x5bc8aca164f8696
1368a41a06a0f7487f853d9d2, 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<br>echo text<br>echo -n text |

```
user1@myubuntu:~$ echo linux
linux
user1@myubuntu:~$ echo "ubuntu linux"
ubuntu linux
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 형식 [인수...] |
| **옵션** | %d, \n 등 C 언어의 printf 함수 형식을 지정한다. |
| **사용 예** | printf text<br>printf "text\n"<br>printf "%d\n" 100 |

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

# 02 셸 기본 사용법

## ■특수 문자 사용하기

- ▪사용자가 더욱 편리하게 명령을 입력하고 실행할 수 있도록 다양한 특수 문자를 제공
- ▪주요 특수 문자는 *, ?, |, ;, [ ], ~, ‘ ’, “ ”, ` `` 등
- ▪명령을 입력하면 셸은 먼저 특수 문자가 있는지 확인하고 이를 적절한 형태로 변경한 후 명령을 실행

## ■특수 문자 *(별표)

- ▪ 임의의 문자열을 나타내는 특수 문자로 0개 이상의 문자로 대체

특수 문자 *

| 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 밑의 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

## ■특수 문자 ?와 [ ]

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

특수 문자 ?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 제외된다. |
| ls –l tmp[135].txt | tmp 다음에 1, 3, 5 중 한 글자가 오고 파일의 확장자가 txt인 모든 파일의 이름을 출력한다. tmp1.txt, tmp3.txt, tmp5.txt 파일이 있으면 해당 파일의 상세 정보를 출력한다. tmp.txt는 제외된다. |
| ls –l tmp[1–3].txt | [1–3]은 1부터 3까지의 범위를 의미한다. 따라서 ls –l tmp[123].txt와 결과가 같다. tmp1.txt, tmp2.txt, tmp3.txt 파일이 있으면 해당 파일의 상세 정보를 출력한다. |
| ls [0–9]* | 파일명이 숫자로 시작하는 모든 파일 목록을 출력한다. |
| ls [A–Za–z]*[0–9] | 파일명이 영문자로 시작하고 숫자로 끝나는 모든 파일 목록을 출력한다. |

# 02 셸 기본 사용법

## ■특수 문자 ~와 –

- ~(물결표)와 -(붙임표)는 디렉터리를 나타내는 특수 문자
- ~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

특수 문자 ~와 –

| 사용 예 | 의미 |
|---|---|
| cp *.txt ~/ch3 | 확장자가 txt인 모든 파일을 현재 작업 중인 사용자의 홈 디렉터리 아래 ch3 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

## ■특수 문자 ;과 |

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

특수 문자 ;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

## ■특수 문자 ‘ ’와 “ ”

- ‘ ’(작은따옴표)와 “ ”(큰따옴표)는 문자를 감싸서 문자열로 만들어주고, 문자열 안에 사용된 특수 문자의 기능을 없앰
- ‘ ’는 모든 특수 문자를, “ ”는 $, ``, ₩을 제외한 모든 특수 문자를 일반 문자로 간주하여 처리

특수 문자 ‘ ’와 “ ”

| 사용 예 | 의미 |
|---|---|
| echo ‘$SHELL’ | $SHELL 문자열이 화면에 출력된다. |
| echo “$SHELL” | 셸 환경 변수인 SHELL에 저장된 값인 현재 셸의 종류가 화면에 출력된다. 예를 들면 /bin/sh이다. |

## ■특수 문자 ``

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

특수 문자 ``

| 사용 예 | 의미 |
|---|---|
| echo "Today is `date`" | `date` 는 명령으로 해석되어 date 명령의 실행 결과로 바뀐다. 결과적으로 다음과 같이 화면에 출력된다.<br>Today is 2014. 02. 23. (일) 15:46:48 KST |
| ls /platform/`uname -m` | uname -m 명령의 실행 결과를 문자열로 바꿔 디렉터리 이름으로 사용한다. |

# 02 셸 기본 사용법

## ■특수 문자 ₩

▪₩(역빗금, ₩와 동일함)은 특수 문자 바로 앞에 사용되는데 해당 특수 문자의 효과를 없애고 일반 문자처럼 처리

특수 문자 \

| 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'의 결과와 같다. |

## ■특수 문자 >, <, >>

▪입출력의 방향을 바꾸는 특수 문자

특수 문자 〉, 〈, 〉〉

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