

주요 미리 정의된 변수의 의미

■ PS1 – 1차 프롬프트 문자열

- 명령어 입력을 기다리고 있음을 알려줌
- 특수한 내용 표시 방법은 p108 표 참조
- PS1의 기본 값: "[\u@\h \W]\\$ "

[gdhong@magics dir]\$

\$ PS1="[\W] \$ " ... 디렉토리 이름만 나타나게 변경

■ PS2 – 2차 프롬프트 문자열

- 명령어 입력 후 Enter를 입력했을 때에 추가 입력을 기다림을 알려줌
- \$ echo "Linux ... "로 시작하면 "로 끝나야 함"
- > shell ... 2차 프롬프트 출력, 입력 요청
- > prompt" ... 2차 프롬프트 출력, 입력 요청
- PS2의 기본 값: "> "
- \$ PS2="2> " ... 변경

33

주요 미리 정의된 변수의 의미(2)

■ HOME – 사용자 홈 디렉토리

- HOME 값이 변경되면 다음 cd 명령어 동작에 영향을 줌
- \$ cd ; cd \$HOME 동작

■ CDPATH – 이동할 디렉토리 위치를 찾는 디렉토리 경로로 사용

- "cd 절대경로" 또는 "cd 상대 경로" – 해당 디렉토리로 이동
- "cd 디렉토리명"
 - 현재 디렉토리와 CDPATH에 등록된 디렉토리를 검색하여 이동할 디렉토리를 찾아서 해당 디렉토리로 이동

\$ CDPATH=/lib:/usr/lib ... CDPATH에 2개의 디렉토리 추가

\$ cd java

/usr/lib/java ... /usr/lib에 있는 java 디렉토리로 이동

34

5.13 명령어 대치

■ 명령어 대치 : `command`

- `command`가 command의 실행 결과로 대치됨
- 이 기능을 사용하면 명령어의 출력을 인수로 사용할 수 있다.
- \$ echo Today is `date`
- Today is 2015. 09. 24. (목) 13:21:35 KST
- \$ echo 현재 `who | wc -l`명이 사용중입니다.
- 현재 4명이 사용중입니다.

■ bash의 명령어 대치 : \$(command)

- \$(command)은 `command`와 같지만, 중첩하여 사용하기가 쉽다.
 - \$ date +%Y ... 오늘의 연도를 출력
 - 2016
 - \$ echo 올해는 \$(date +%Y)년이다. ... 명령어 대치 사용
 - 올해는 2016년입니다.
 - \$ echo 내년은 \$(expr \$(date +%Y) + 1)년 ... 명령어 대치 중첩 사용
 - 내년은 2017년입니다.
- expr : 수식 연산 명령어

35

(요약) 셸의 3가지 대치 기능

■ 파일이름 대치 – 와일드카드(*, ?, [..])

- *.c → 파일이름 패턴에 매칭되는 파일이름 목록

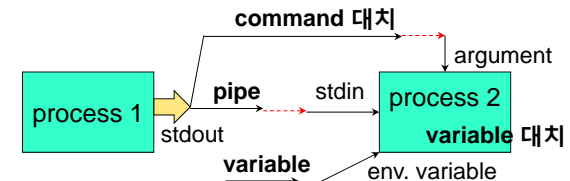
■ 변수 대치 – 변수값 접근

- \$var → 변수 var 값

■ 명령어 대치

- `command`, \$(command) → 명령어 command의 실행결과

■ 프로그램(명령어)에 값을 전달하는 경로



36

5.14 인용(quote)과 대치 금지

- 메타 문자 기능 금지
 - 역슬래시 문자 \ (탈출문자)와 함께 사용, 또는
 - 인용 부호(따옴표) 안에서 사용
- 작은 따옴표 - '...'
 - 모든 대치 금지: 파일이름 대치, 변수 대치, 명령어 대치, 메타문자
- 큰 따옴표 - "..."
 - 파일이름 대치 금지
 - \$ myvar=Korea
 - \$ echo f* \$myvar `expr 1 + 2` \\$... 모든 대치 가능
 - \$ echo 'f* \$myvar `expr 1 + 2` \\$' ... 모든 대치, 탈출문자기능 금지
 - \$ echo "f* \$myvar `expr 1 + 2` \\$" ... 파일이름 대치 금지
- 중첩된 인용부호는 바깥쪽 인용부호가 유효함
 - \$ echo "f* '\$myvar' `expr 1 + 2` \\$"
 - 'Korea' ... 변수대치 동작

37

5.15 별명(alias)

- alias - 자주 사용하는 옵션을 포함한 명령어에 별명을 부여
 - \$ alias del='rm -i' ; = 전후에 빈칸 없음
 - \$ alias cp='cp -i' ; 같은 이름의 별명
 - 별명 내용 출력
 - \$ alias del ; 별명의 정의 내용 출력
 - \$ alias ; 현재 부여된 별명 출력
 - 명령어 대신에 별명을 사용하여 실행 (csh부터 제공)
 - \$ del file1 ; rm -i file1
 - \$ cp file1 file2 ; cp -i file1 file2
- alias 제거
 - \$ unalias cp
- alias 사용 금지
 - \를 앞에 붙이거나 작은 따옴표 사용
 - \$ \cp file1 file2 ; 원래의 명령어 실행 - cp file1 file2
 - \$ 'cp' file1 file2

38

5.16 셸 시작 파일(startup file)

- 셸 시작(초기화) 파일
 - 셸이 시작되기 전에 먼저 실행되는 명령어를 포함한 파일
 - 자주 사용하는 alias 정의, 셸 변수 값 초기화 작업에 주로 사용됨
- 대화형(interactive) 셸의 시작 파일 (bash)
 - 로그인 셸의 시작 파일 탐색 및 실행 순서
 - /etc/profile → ~/.bash_profile, ~/.bash_login, ~/.profile
 - 비로그인 셸의 시작파일 탐색 및 실행 순서
 - ~/.bashrc (이 파일은 대개 .bash_profile에 포함되어 있음)
 - .bash_profile - 로그인할 때만 실행하는 명령어 포함
 - .bashrc - 대화형 셸을 시작할 때마다 수행하는 명령어 포함
- 비대화형 셸의 시작 파일
 - 시작 파일 = 환경변수 BASH_ENV에 저장된 파일이름의 파일
 - \$ export BASH_ENV=~/.bashrc"; 대화형 셸의 시작파일 그대로 사용

39

시작파일과 source 명령어

- (예) 시작 파일
 - ".bash_profile" - 환경변수
 - export PATH=\$PATH:\$HOME/bin
 - export HISTSIZE=20
 - export HISTFILESIZE=10
 - ".bashrc" - alias, 지역변수
 - alias dir='ls -F'
 - alias rm='rm -i'
 - PS1="\$ "
- source 명령어 - 현재 셸에서 셸 프로그램 실행(주로 시작파일 사용)
 - \$ source .bash_profile
 - \$. .bash_profile

40

5.17 히스토리(history)

- history
 - 최근에 실행한 명령어들의 목록을 출력


```
$ history
```
 - csh에서부터 제공된 기능으로 재실행할 때에 사용
- history와 관련된 변수
 - HISTSIZE history 목록에 저장할 명령어 개수(기본 500)
 - HISTFILE history file 이름 (~/.bash_history)
 - HISTFILESIZE 다음 세션을 위해 history file에 저장할 명령어 개수
- history를 포함한 프롬프트 -\!


```
$ PS1="\!$ "
36$
```

history 참조

- history 참조 내용은 명령어 재실행 또는 인수로 사용할 수 있음

형식	의미
!!	이전 명령어 (!-1과 같음)
!n	히스토리 번호 n번 명령어
!-n	n번 이전의 명령어
!str	문자열 str로 시작하는 가장 최근 명령어
!str?	문자열 str을 포함하는 가장 최근 명령어
^old^new	이전 명령어에서 문자열 old를 문자열 new로 치환

```
$ !! ; 이전 명령어 실행
$ !ec ; 최근의 ec로 시작하는 명령어 실행
$ echo bash
$ ^bash^tssh ; echo tssh 실행
```

history 일부 접근

- history의 일부 인수를 선택하여 사용 가능

형식	의미
:0	명령어 행의 첫 번째 단어 (명령어)
:n	명령어 행의 n+1 번째 단어 (n번째 인수)
:m-n	명령어 행의 m+1 번째 단어부터 n+1 번째까지의 단어들
:^ 또는 ^	:0과 같음
:\$ 또는 \$	명령어 행의 마지막 단어
:* 또는 *	명령어 행의 명령어를 제외한 모든 인수. :1-\$과 같음
:n*	:n-\$와 같음
:n-	:n*에서 마지막 단어를 제외한 것

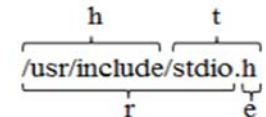
```
20 $ echo I like horseback riding
21 $ echo !:3 // horseback
22 $ echo !20:2-$ // like horseback riding
```

파일 이름 일부 접근/수정

- 인수로 사용된 파일 이름을 수정하여 사용 가능

형식	의미
:h	(head) 경로 이름의 앞 부분 (마지막 파일 또는 디렉토리 이름 제거)
:t	(tail) 경로 이름의 뒷 부분 (마지막 파일 또는 디렉토리 이름)
:r	(root) 경로 이름의 확장자를 제외한 부분
:e	(extension) 경로 이름의 . 으로 시작하는 마지막 부분(확장자)
:s/old/new/	명령어 행의 문자열 old를 문자열 new로 대치 (s앞에 g를 사용하면 모든 문자열 old를 문자열 new로 치환)
:p	참조된 사항을 실행하지 않고 출력만 함

```
$ ls /usr/include/stdio.h
$ echo !:1:h → /usr/include
$ echo !-2:1:s/stdio/stdlib/ → /usr/include/stdlib.h
```



fc 명령어 와 자동완성 기능

■ fc – fixed command

- 히스토리 편집 후 실행 가능
 - \$ fc -l ; 히스토리 목록
 - \$ fc -s 42 ; 42번 명령어 재실행
 - \$ fc -s ; 직전 명령어 재실행
 - \$ fc 42 ; 42번 명령어 **편집후** 재실행 - vi 편집

■ 자동완성 기능

- 명령어 완성 가능한 명령어가
 - \$ bz<Tab><Tab> ; 여러 개이면 목록 출력
 - \$ bz<Tab> ; 한 개이면 명령어 자동 완성
- 파일 이름 완성/변수 완성 - 명령어 완성과 유사
 - ... /usr/include/st<Tab><Tab> \$HIST<Tab><Tab>
 - ... /usr/include/stdl<Tab> \$HISTS<Tab>

45

5.18 디렉토리 스택

■ pushd

- 인수 디렉토리를 디렉토리 스택에 저장하고, 인수 디렉토리로 이동
 - \$ pushd /usr

■ popd

- 디렉토리 스택 위의 디렉토리를 제거하고, 그 다음 디렉토리로 이동
 - \$ popd

■ dirs

- 디렉토리 스택 내용 출력 (~ 부터 저장되어 있음)
 - \$ dirs
 - /etc /usr ~

46

5.19 간단한 셸 프로그램

■ shell script

- 셸에서 실행될 수 있는 명령어들을 포함하는 파일
- 일반 프로그램에서 사용하는 제어구문도 사용할 수 있음
- shell program이라고도 하며, 실행 허가권이 부여되면 실행가능
- 반복적으로 수행할 필요가 있는 다양한 작업에 유용

■ 간단한 shell script의 작성 및 실행

\$ vi script1 ; 편집기로 작성

```
echo Hi! Shell Script
echo HOME=$HOME
echo Today is `date`
```

\$ chmod +x script1 ; 실행허가권 부여

\$./script1 ; shell script 실행

47

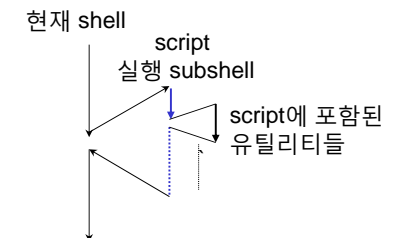
Script 실행 셸 지정

■ Shell Script를 실행하는 셸 지정

- 첫 줄 내용에 따라서 command를 처리하는 셸이 결정됨
 - 1) # 없음 : /bin/sh
 - 2) # 로 시작 : 현재 shell
 - 3) #! pathname : pathname (프로그램의 절대경로명)
- 나머지 줄의 #은 comment로 사용됨

```
#
echo Hi! Shell Script
echo HOME=$HOME
echo Today is `date`
```

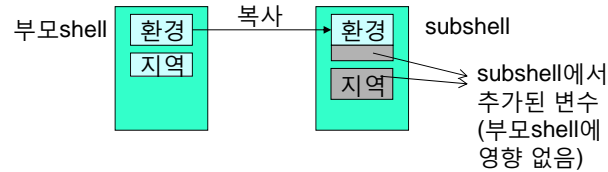
```
#!/bin/bash
echo Hi! Shell Script
echo HOME=$HOME
echo Today is `date`
# comment
```



48

서브셸과 변수

- 서브셸의 작업디렉토리
 - 서브셸은 자신만의 작업 디렉토리를 가짐
 - 서브셸에서의 작업 디렉토리 변경이 부모셸에 영향을 주지 않음
- 셸 변수와 서브셸
 - 환경변수 : 서브셸(또는 자식 프로세스)에 복사됨
 - 유용한 정보 전달에 사용됨
 - 지역변수 : 서브셸에 복사되지 않음



49

Shell Script 주석

- #로 시작 (첫째 줄 제외)
- 콜론(:) 명령어 사용
 - 콜론 명령어 : 아무 동작도 하지 않음. null 명령
 - \$: 'colon command' argument

```

: '
comment line 1
comment line 2
'
    
```

- here document 사용

```

<<MARK
comment line 1
comment line 2
MARK
    
```

50

Argument 변수

- 인수(argument) 접근 셸 변수
 - \$0 : command name
 - \$n : n번째 command line argument (n=1-9)
 - * : 모든 argument의 list
 - # : argument 개수
 - \$\$: shell의 process id (임시파일 이름 생성에 유용)

- 예: script2

```

echo $0 and $1 and $2 and $3
echo $* : $# arguments
    
```

\$./script2 a b c d

51

5.20 셸의 기타 기능

- 표준입출력과 파일기술자 번호
 - 0 : 표준 입력
 - 1 : 표준 출력
 - 2 : **표준에러 출력**
 - 3 이상 : 추가로 open되는 파일들
- 표준 에러출력의 방향전환 : 2>, 2>>

(cf) >와 >>는 1>와 1>>이다.

 - \$ cat in1 in2 2> out2 ; in1 존재, in2 미존재
 - \$ cat in1 in2 > out1 2> out2 ; 두 출력 각각 방향전환
- 두 출력의 연계 : 2>&1 (표준에러출력을 표준출력 장치로)
 - \$ cat in1 in2 2>&1
 - \$ (cat in1 in2 2>&1) > out3 ; 두 출력 같은 파일로 방향전환
 - \$ cat in1 in2 > out3 2>&1 ; 두 출력 같은 파일로 방향전환
 - \$ cat in1 in2 2>&1 > out3 ; 에러출력(원래표준출력=화면) ; 표준출력(out3)

52

표준에러 출력의 방향전환

- 두 출력의 연계 : 1>&2 (표준출력을 표준에러출력으로)
\$ echo This is stderr output 1>&2 ; echo를 사용한 표준에러출력
- 표준출력과 표준에러 출력의 동시 방향전환 : >&, &>
\$ cat in1 in2 >& out3
- 표준에러 출력의 파이프 전달
\$ cat in1 in2 2>&1 | wc ; 두 출력 파이프에 동시 전달
\$ cat in1 in2 2>&1 > out4 | wc ; 표준에러출력만 파이프에 전달

53

set 명령어 사용한 셸 기능 변경 (bash)

- 셸 기능 설정 제어
 - noglob – 파일 이름 대치 금지
 - noclobber – 파일 덮어쓰기 금지
 - notify – 백그라운드 작업 종료 시 즉시 알림
 - ignoreeof – ^D 입력 무시
- 설정 방법
\$ set -o noglob ; 설정
\$ set +o noglob ; 설정 해제
\$ set -o noclobber
(cf) csh에서는 set과 unset을 설정과 해제에 사용함
- noclobber 설정 시에 덮어쓰기 : >|

54

5.21 내장 명령어

- help (bash)
\$ help ; 내장 명령어 목록을 출력
\$ help read ; 내장 명령어 사용법을 출력
- hash (sh)
 - 경로 디렉토리를 검색하지 않고 명령어 위치를 빠르게 찾으려면, 명령어 경로 이름을 해시 테이블에 저장
\$ hash ; 해시테이블 목록과 사용 횟수 출력
- type (bash)
 - 명령어 경로, 내장 명령어 여부, 별명 정보 제공
- command (bash)
 - 인수로 제공된 명령어 실행 시 별명(alias)이나 함수를 사용하지 않음
- enable (bash)
 - 내장 명령어 활성화/비활성화
\$ enable -n pushd ; 비활성화
\$ enable pushd ; 활성화

55