Shell

Shell

- 개요
  - ✓ 쉘은 사용자가 입력한 명령을 해석해 커널로 전달하거나 커널의 처리 결과를 사용자에게 전달하는 역할을 수행
  - ✓ Server의 텍스트 모드 나 X 윈도의 터미널처럼 명령을 입력하는 인터페이스
  - ✓ 쉘의 기능
    - 명령어 해석기
      - 사용자가 입력한 명령이나 파일에서 읽어 들인 명령을 해석하고 적절한 프로그램을 찾아서 실행
    - 프로그래밍
      - 쉘은 자체 내에 프로그래밍 기능이 있어서 프로그래밍 가능
      - 쉘의 프로그래밍 기능을 이용하면 여러 명령을 사용하여 반복적으로 수행하는 작업을 하나의 프로그램으로 만들 수 있는데 이렇게 작성된 쉘 프로그램을 쉘 스크립트라고 부름
    - 사용자 환경 설정
      - 쉘은 사용자 환경을 설정할 수 있도록 초기화 파일 기능을 제공
      - 초기화 파일에는 명령을 찾아오는 경로를 설정하거나 파일과 디렉토리를 새로 생성할 때 적용하는 기본 권한을 설정하거나 다양한 환경 변수 등을 설정할 수 있음
      - 사용자가 로그인할 때 이 초기화 파일이 실행되어 사용자 별로 특성에 맞게 초기 환경이 설정됨

Shell

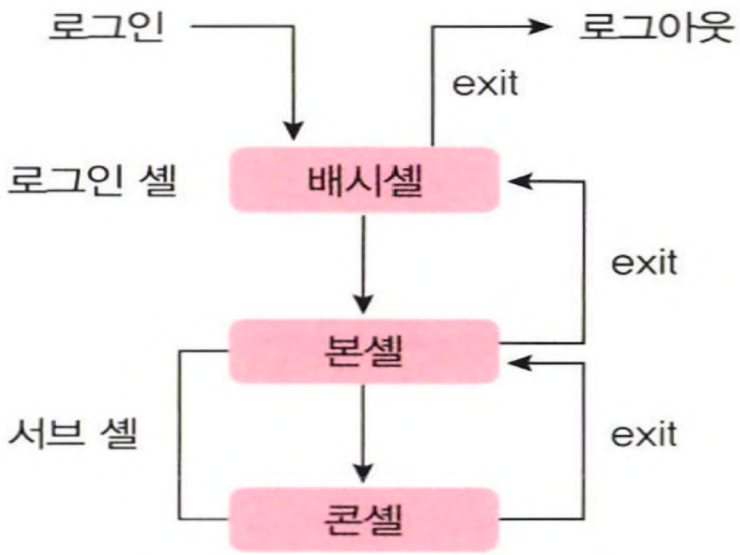
- 종류
  - ✓ 본쉘(sh): 최초의 쉘, 유닉스 v7에서 처음 등장했는데 현재는 배시쉘 등 다른 쉘로 대체
  - ✓ C쉘(csh): 캘리포니아 대학교에서 빌 조이가 개발한 쉘로 2BSD 유닉스에서 발표
  - ✓ 콘쉘(ksh): 1980년대 중반 AT&T 벨연구소에서 개발한 쉘로 SVR4 유닉스에서 발표
  - ✓ bash
    - 1988년 브레인 폭스가 개발
    - 본쉘과 호환성 유지, C쉘/콘쉘의 편리한 기능 모두 포함
    - 리눅스의 기본 쉘로 제공
  - ✓ tsch: csh에 이어서 개발된 C쉘 계열의 쉘
  - ✓ dash shell
    - 1997년 허버스 슈가 리눅스에 이식
    - 본쉘을 기반으로 개발, POSIX 표준 준수하며 작은 크기로 개발
  - ✓ zsh
    - 비교적 최근에 개발된 쉘로 bash 와 tsch의 기능에 독자적인 기능을 추가
    - 다양한 기능을 갖추고 있어서 메뉴얼만 17개 섹션에 달함

Shell

❖ 로그인 쉘 과 서브 쉘

✓ 개요

- 사용자는 프롬프트에서 다른 쉘을 실행할 수 있는데 이렇게 새로 생성된 쉘이 서브 쉘
- 서브 쉘은 또 다른 서브 쉘을 생성할 수 있으며 여러 개의 쉘이 사슬처럼 연결될 수 있음
- 서브 쉘을 종료하는 명령은 CTRL + d 또는 exit 등으로 일반 쉘의 로그아웃 명령과 동일
- 서브 쉘이 종료되면 서브 쉘을 실행했던 이전 쉘 환경으로 돌아가며 로그인 쉘에서 로그아웃을 하면 접속이 해제



Shell

- bash shell
  - ✓ 우분투에서 기본적으로 사용하는 쉘은 bash(Bourne Again Shell)
  - ✓ bourne Shell(sh)을 기반으로 Korn Shell과 C Shell의 장점을 합한 것
  - ✓ bash의 특징
    - alias 기능(명령 단축 가능)
    - history 기능
    - 연산 기능
    - Job Control 기능
    - 자동 이름 완성 기능
    - 프롬프트 제어 기능
    - 명령 편집 기능

Shell

- 지원하는 Shell 확인
  - ✓ cat /etc/shells

```
# /etc/shells: valid login shells
/bin/sh
/bin/bash
/usr/bin/bash
/bin/rbash
/usr/bin/rbash
/usr/bin/sh
/bin/dash
/usr/bin/dash
/usr/bin/tmux
/usr/bin/screen
```

Shell

- ❖ 쉘 변경
  - ✓ shell 확인: 특정 사용자의 정보를 찾아서 확인
    - grep adam /etc/passwd

      adam:x:1000:1000:adam:/home/adam:/bin/bash
      - ◆ 가장 앞의 정보가 로그인 ID이고 가장 마지막에 나온 /bin/bash가 사용자의 기본 쉘
      - ◆ adam 사용자는 로그인할 때 사용자의 기본 쉘로 배시쉘을 사용
      - ◆ 사용자가 로그인하면 자동으로 실행되는 기본 쉘을 로그인 쉘이라고 부름

Shell

- ❖ 쉘 변경
  - ✓ C shell 설치
    - 설치 명령: sudo apt-get install csh
    - 확인: cat /etc/shells

Shell

- ❖ 쉘 변경
  - ✓ shell 변경
    - ● chsh [옵션] [사용자명]
      - ◆ 옵션
        - ▪ -s shell경로: 지정하는 쉘(절대 경로)로 로그인 쉘을 변경
        - ▪ -l: /etc/shells. 파일에 지정된 쉘을 출력
      - ◆ 실습
        - ▪ adam@adam:~$ adam@adam:~$ chsh -s csh adam
          chsh: csh is an invalid shell #절대 경로가 아니라서 에러
        - ▪ adam@adam:~$ chsh -s /bin/csh adam
        - ▪ adam@adam:~$ exit
          logout
          Connection to 192.168.64.14 closed.
        - ▪ 다시 접속
          %
        - ▪ adam@adam:~$ chsh -s /bin/bash adam
        - ▪ adam@adam:~$ exit

Shell

- ❖ 쉘 변경
  - ✓ shell 변경
    - 쉘이름 -s sh 사용자명
    - 실습
      - adam@adam:~$ csh -s sh adam
      - % ls

        Desktop Downloads Pictures Public Videos
        Documents Music practice Templates
      - % bash -s sh adam

        adam@adam:~$

Shell

- Shell 내장 명령
  - ✓ 쉘은 자체적으로 내장 명령을 가지고 있는데 일반적인 리눅스 명령들이 /bin이나 /usr/bin 디렉토리에 별도의 실행 파일로 있는 것과 달리 쉘 명령은 별도의 실행 파일이 없이 쉘 안에 포함되어 있는데 대표적인 쉘 내장 명령은 cd
  - ✓ 실습 – 파일 확인
    - adam@adam:~$ file /usr/bin/pwd

      /usr/bin/pwd: ELF 64-bit LSB pie executable, ARM aarch64, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux-aarch64.so.1, BuildID[sha1]=797b5efcab530219e8f43f43bcf5c02105418720, for GNU/Linux 3.7.0, stripped
    - adam@adam:~$ file cd

      cd: cannot open `cd' (No such file or directory)
    - adam@adam:~$ file /usr/bin/cd

      /usr/bin/cd: cannot open `/usr/bin/cd' (No such file or directory)

Shell

- ❖ Shell 내장 명령
  - ✓ 출력 명령
    - ● echo
      - ◆ 기능: 화면에 한 줄의 문자열을 출력
      - ◆ 형식: echo [-n] [문자열 또는 변수]
      - ◆ 옵션 -n 마지막에 줄 바꿈을 하지 않음
      - ◆ 실습
        - ▪ adam@adam:~$ echo linux

          linux
        - ▪ adam@adam:~$ echo "linux ubuntu"

          linux ubuntu

Shell

- ❖ Shell 내장 명령
  - ✓ 출력 명령
    - ● printf
      - ◆ 기능: 데이터를 형식화 해서 화면에 출력
      - ◆ 형식: printf [형식] [인수]
      - ◆ 옵션: 형식은 %d, ₩n 등 C 언어의 printf 함수의 형식을 지정
      - ◆ 실습
        - ▪ adam@adam:~$ printf linux
        - ▪ linuxadam@adam:~$ printf "ubuntu linux ₩n"

          ubuntu linux
        - ▪ adam@adam:~$ printf "%d + %d = %d₩n" 1 3 4

          1 + 3 = 4

Shell

- ❖ 특수 문자
  - ✓ 개요
    - 쉘은 사용자가 더욱 편리하게 명령을 입력하고 실행할 수 있도록 다양한 특수 문자를 제공
    - 특수 문자의 종류와 사용법은 모든 쉘에서 거의 비슷
    - 특수 문자는 쉘에서 특별한 의미를 가진 문자로 각 특수 문자에 따라 특수 기능을 수행
    - 주요 특수 문자로는 *, ?, |,;, [ ] , ~ 등이 있음
    - 사용자가 명령을 입력하면 쉘은 먼저 입력한 내용 중에 특수 문자가 있는지 확인하고 이를 해독하여 적절한 형태로 변경한 후 명령을 실행

Shell

❖ 특수 문자

✓ *

- 임의의 문자열을 의미 – 글자 수 상관없음

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.

Shell

❖ 특수 문자

✓ ? 와 [ ]

- 하나의 문자를 의미

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.

- 명령어 수행 후 echo $? 는 이전 명령어의 수행 결과로 성공하면 1 실패하면 0

Shell

- ❖ 특수 문자
  - ✓ ~ 와 –
    - ~은 현재 사용자의 홈 디렉토리를 의미하는 문자
    - -는 현재 디렉토리로 이동하기 직전의 디렉토리

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.

Shell

❖ 특수 문자

✓ `

- 문자열에서 백틱으로 묶으면 명령을 수행해서 명령의 결과를 대체

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.

Shell

- ❖ 특수 문자
  - ✓ ; 와 |, &&
    - ; -> 여러 개의 명령을 순차적으로 실행할 때 사용하는데 앞의 명령어가 실패해도 다음 명령어를 수행
    - && -> 앞의 명령어가 성공했을 때 뒤의 명령어 수행
    - | -> 앞 쪽의 명령의 실행 결과를 뒤에 전달, Pipe

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.

Shell

❖ 특수 문자

✓ ₩

● 특수 문자의 효과 무력화

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.

Shell

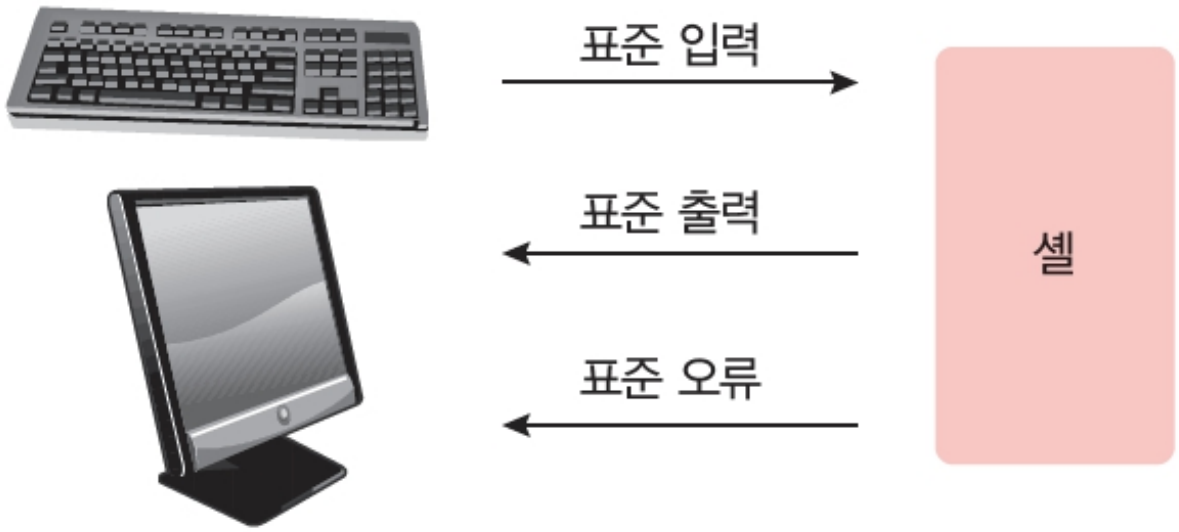
❖ 특수 문자

✓ ' '(모든 특수 문자의 기능 무력화) 와 " "($, ₩, ${ }, { } 은 원래대로 처리)

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.

Shell

❖ 표준 입출력 장치



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

Shell

- 출력 redirection – 출력 방향을 변경하는 것
  - ✓ >
    - 출력 방향을 변경하는 명령
    - 출력 방향의 내용을 전부 변경
    - 형식

      명령 1 > 경로

      명령 > 경로
    - date 1 > a.txt

Shell

- ❖ 출력 redirection
  - ✓ >>
    - 파일에 내용을 추가
    - 형식

      명령 >> 파일 경로
    - date >> b.txt

Shell

- ❖ 에러 redirection
  - ✓ 2>
    - 표준 오류 메시지를 파일에 저장
    - 형식

      명령 2>파일 경로
    - 에러 redirection
      - ◆ ls /abc 2>ls.err
      - ◆ cat ls.err

        ls: cannot access '/abc': No such file or directory
    - 오류 메시지 버리기
      - ◆ ls /abc 2>/dev/null
    - 표준 출력 과 에러를 하나의 파일에 redirection
      - ◆ ls . /abc > ls.out 2>&1
      - ◆ cat ls.out

        ls: cannot access '/abc': No such file or directory

        .:

        data

        data1

Shell

- 입력 redirection
  - ✓ 표준 입력을 바꾸는 기능
  - ✓ 형식
    - 명령 0< 파일 경로
    - 명령 < 파일 경로
  - ✓ 매개변수가 없는 명령어에 매개변수를 설정할 수 있음
    - tr a-z A-Z < today.txt

환경 설정

- 변수
  - ✓ 종류
    - 쉘 변수: 현재 쉘(터미널)에서만 사용 가능한 변수
    - 환경 변수: 시스템 전체에 적용되는 변수
  - ✓ 변수 확인
    - echo $변수
    - env: 환경 변수 전부 출력
    - set: 모든 변수 와 함수를 출력
  - ✓ 변수 수정
    - 쉘 변수 생성 및 수정: 변수명=값
    - 환경 변수 생성 및 수정: export 변수명=값
    - 환경 변수를 로컬 변수로 수정: export -n 변수명
  - ✓ 변수 삭제
    - unset 변수명

환경 설정

❖ 변수

✓ 주요 환경 변수

환경 변수	설명	환경 변수	설명
HOME	현재 사용자의 홈 디렉터리	PATH	실행 파일을 찾는 디렉터리 경로
LANG	기본 지원되는 언어	PWD	사용자의 현재 작업 디렉터리
TERM	로그인 터미널 타입	SHELL	로그인해서 사용하는 셸
USER	현재 사용자의 이름	DISPLAY	X 디스플레이 이름
COLUMNS	현재 터미널의 컬럼 수	LINES	현재 터미널 라인 수
PS1	1차 명령 프롬프트 변수	PS2	2차 명령 프롬프트(대개는 '>')
BASH	bash 셸의 경로	BASH_VERSION	bash 버전
HISTFILE	히스토리 파일의 경로	HISTSIZE	히스토리 파일에 저장되는 개수
HOSTNAME	호스트의 이름	USERNAME	현재 사용자 이름
LOGNAME	로그인 이름	LS_COLORS	ls 명령의 확장자 색상 옵션
MAIL	메일을 보관하는 경로	OSTYPE	운영체제 타입

환경 설정

- ❖ 변수
  - ✓ 실습
    - set
    - env
    - echo $SHELL
    - SOME=test
    - echo SOME

      SOME
    - set | grep SOME

      SOME=test

      _=SOME
    - env | grep SOME
    - export SOME
    - env | grep SOME

      SOME=test
    - export SOME1=test1
    - echo $SOME1

      test1

환경 설정

- ❖ 변수
  - ✓ 실습
    - env | grep SOME
      SOME=test
      SOME1=test1
    - export -n SOME1
    - env | grep SOME
      SOME=test
    - set | grep SOME
      SOME=test
      SOME1=test1
      _=SOME1
    - unset SOME
    - unset SOME1
    - echo $SOME
    - echo $SOME1

환경 설정

- ❖ 환경 변수
  - ✓ 이스케이프 문자와 프롬프트 설정
    - ● 배시쉘에서는 환경 변수 PS1에 프롬프트로 사용할 문자열을 저장
    - ● PS1의 현재 설정 값
      - ◆ adam@adam:~$ echo $PS1

        ₩[₩e]0;₩u@₩h: ₩w₩a₩]${debian_chroot:+($debian_chroot)}₩[₩033[01;32m₩]₩u@₩h₩[₩033[00m₩]:₩[₩033[01;34m₩]₩w₩[₩033[00m₩]₩$

        - ◆ PS1의 설정 값을 보면 ₩e, ₩u, ₩h 등 의미를 알 수 없는 문자로 구성되어 있는데 이렇게 ₩로 시작하는 특별한 문자들을 이스케이프 문자라고 하며 ₩u와 같이 ₩로 시작하는 이스케이프 문자는 두 글자가 아닌 한 글자로 처리
        - ◆ 이스케이프 문자는 화면에 문자 그대로 출력되지 않고 쉘이 문자의 의미를 해석하여 실행

환경 설정

- ❖ 환경 변수
  - ✓ 이스케이프 문자와 프롬프트 설정
    - ● 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.

환경 설정

- ❖ 환경 변수
  - ✓ 이스케이프 문자와 프롬프트 설정
    - 환경 변수 PS1에 새로운 형태의 문자열을 지정하면 프롬프트를 변경할 수 있음
    - PS1의 값을 바꾸기 전에 먼저 현재 PS1의 값을 임시 변수에 저장 해 두어야 하는데 원래 프롬프트로 돌아갈 때 사용하기 위함
    - 변수를 설정할 수 있고 기호를 이용할 수 있음
    - 프롬프트에서 사용할 수 있는 기호
      - [: [자체를 그대로 나타냄.
      - ₩u : 현재 사용자를 의미함.
      - @: @ 기호를 자체를 나타냄
      - ₩h: 현재 시스템의 호스트 이름을 의미
      - ₩W: 현재 위치의 절대 경로 가운데 현재 디렉토리명만을 나타냄
      -] :] 자체를 그대로 나타냄
      - ₩₩$: root(UID가 0이면)이면 #을 표시하고 일반 사용자면 $을 표시함.
      - ₩t, ₩T, ₩@, ₩d: 현재 시간이나 날짜

환경 설정

- ❖ 환경 변수
  - ✓ 이스케이프 문자와 프롬프트 설정
    - ● 실습
      - ◆ adam@adam:~$ PROMPT=$PS1
      - ◆ adam@adam:~$ PS1='LINUX] '

        LINUX]
      - ◆ LINUX] PS1='[$PWD]'
      - ◆ [/home/adam] cd ..
      - ◆ [/home]PS1='`uname -n` $ '
      - ◆ itstudy $ PS1='[₩u ₩T] ₩!$ '
      - ◆ [adam 10:45:20] 53$
      - ◆ [adam 05:16:19] 131$ PS1=$PROMPT

환경 설정

- ❖ 환경 변수
  - ✓ PATH
    - 명령어의 위치를 찾는 디렉토리를 저장하고 있는 변수
    - 명령어를 찾을 디렉토리를 ; 으로 구분해서 설정
    - PATH 확인: echo $PATH

      /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
    - PATH에 경로 추가: PATH="$PATH:추가할경로"
      - PATH="$PATH:~/bin"
      - echo $PATH

        /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:~/bin

환경 설정

- ❖ 환경 변수
  - ✓ LANG
    - ● 로케일 정보를 저장하고 있는 변수
    - ● 현재 사용 중 인 로케일 확인
      - ◆ echo $LANG

        en_US.UTF-8
    - ● 시스템에서 지원하는 로케일 확인
      - ◆ locale -a

        C

        C.utf8

        en_US.utf8

        POSIX
    - ● 로케일 설정
      - ◆ LANG=로케일

환경 설정

- 환경 변수
  - ✓ HISTFILE: 커맨드 라인 이력을 저장할 파일 이름으로 기본값은 ~/bash_history
  - ✓ HISTFILESIZE: 파일에 저장할 커맨드 라인 이력의 최대 개수
  - ✓ HISTSIZE: 메모리에 저장할 커맨드 라인 이력의 최대 개수
  - ✓ HOME: 홈 디렉토리
  - ✓ SHELL: 로그인 쉘의 경로
  - ✓ PWD: 현재 작업 디렉토리

환경 설정

- 환경 설정 파일
  - ✓ 환경 설정 파일은 시스템을 사용하는 사용자의 환경을 설정하는 파일로 로그인할 때마다 무조건 실행되는 파일

환경 설정

- ❖ 시스템 환경 설정 파일
  - ✓ 시스템 환경 설정 파일

파일	기능
/etc/profile	• 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	• 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.

환경 설정

❖ 사용자 환경 설정 파일

- ✓ 사용자 환경 설정 파일은 각 사용자의 홈 디렉토리에 숨김 파일로 존재하며 사용자가 내용을 수정하고 관리
- ✓ 사용자가 로그인하면 제일 먼저 시스템 환경 설정 파일이 실행되어 시스템 공통 환경을 만들고 이후 사용자 환경 설정 파일을 순서대로 실행하여 사용자 별 환경을 설정
- ✓ 사용자 환경 설정 파일

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

- ✓ 우분투에서 .profile은 .bashrc 파일이 있으면 실행시키고 환경 변수 PATH에 기본 경로를 설정
- ✓ .bash_logout 파일에는 콘솔 화면을 클리어하는 코드가 설정되어 있음
- ✓ 로그아웃을 하고 로그인을 다시 해야 적용이 되는데 . 이나 source 명령을 이용해서 현재 상태에서 적용 가능

환경 설정

- ❖ 사용자 환경 설정 파일
  - ✓ 별명 관련 환경 설정 파일 만들기 및 적용
    - sudo vi .bash_aliases
    - 내용 작성

      alias rm='rm-i'

      alias h=history

      alias c=clear
    - source .bash_aliases
    - h 또는 c 명령 수행

환경 설정

- bash 옵션
  - ✓ set 명령
    - -o 나 +o를 지정하여 옵션 기능을 활성화하거나 비활성화 할 수 있음
    - -o를 지정하면 기능이 활성화되고 +o를 지정하면 비활성화
    - ignoreeof(쉘을 빠져나가는 기능인 eof를 무시하는 기능) 활성화: CTRL + D를 눌러도 쉘을 빠져나가지 않음
      - adam@adam:~$ set -o ignoreeof
      - adam@adam:~$

        Use "logout" to leave the shell.
      - adam@adam:~$

환경 설정

- ❖ bash 옵션
  - ✓ set 명령
    - ● noclobber: 덮어쓰기 방지
      - ◆ touch file1
      - ◆ echo 1234 > file1
      - ◆ set -o noclobber
      - ◆ echo 1111 > file1

        -bash: file1: cannot overwrite existing file
      - ◆ set +o noclobber
      - ◆ cat file1

        1234
      - ◆ > file1
      - ◆ cat file1

환경 설정

- bash 옵션
  - ✓ shopt 명령
    - 옵션 기능을 활성화하거나 비활성화 할 수 있음
    - 명령어 사용
      - shopt — 전체 옵션 리스트
      - shopt -s — on 되어 있는 옵션 리스트
      - shopt -s option — 옵션 on
      - shopt -u — off 되어 있는 옵션 리스트
      - shopt -u option — 옵션 off
      - shopt -p option — 현재 옵션에 해당되는 명령어 출력
      - shopt -q option — 옵션의 on/off 여부를 exit code로 반환

환경 설정

- ❖ bash 옵션
  - ✓ shopt 명령
    - ● 옵션
      - ◆ autocd — 디렉토리 이름을 입력하면 해당 디렉토리로 이동
      - ◆ dotglob — * 나 ?를 사용한 경로 확장의 결과에 .으로 시작하는 파일도 포함
      - ◆ cdspell — cd 명령어 사용 시 디렉토리 이름의 오타를 자동 수정
      - ◆ globstar — 경로 확장으로 **을 사용하면 서브 디렉토리까지 포함한 모든 파일에 매칭
      - ◆ histappend — 배시를 종료할 때 히스토리 파일에 명령어 이력을 덮어쓰지 않고 추가

환경 설정

- ❖ bash 옵션
  - ✓ shopt 명령
    - ● 실습
      - ◆ ls

        Desktop Downloads Music practice Templates

        Documents file1 Pictures Public Videos
      - ◆ cd Desktopp

        -bash: cd: Desktopp: No such file or directory
      - ◆ shopt -s cdspell
      - ◆ cd Desktopp

        Desktop
      - ◆ shopt –s autocd
      - ◆ /

        cd -- /