

소프트웨어프로젝트 I

리눅스 (Linux) - 셸 환경 변수
- vi 에디터

2022학년도 1학기

국민대학교 소프트웨어학부

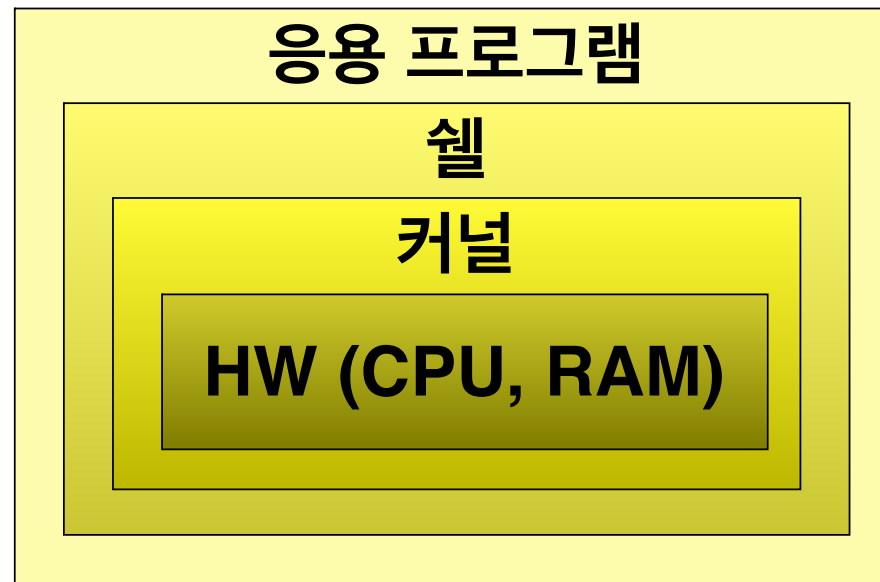
셸 (Shell)

UNIX 스타일의 운영체제에서 사용자가 입력하는 명령을 해석, 실행하는 프로그램

여러 가지 서로 다른 셸 이용 가능

- Bourne shell (sh), C shell (csh), Korn shell (ksh) 등

우리가 주로 이용할 셸은: Bourne Again shell (bash)



우선 셸 프로그램들을 확인해볼까?

```
sheayun@sheayun-ubuntu: ~  
sheayun@sheayun-ubuntu:~$ ls /bin/*sh  
/bin/bash /bin/dash /bin/rbash /bin/sh /bin/static-sh  
sheayun@sheayun-ubuntu:~$
```

```
sheayun@sheayun-ubuntu: ~  
sheayun@sheayun-ubuntu:~$ ls -l /usr/bin/tclsh  
lrwxrwxrwx 1 root root 8 11월 21 2015 /usr/bin/tclsh -> tclsh8.6  
sheayun@sheayun-ubuntu:~$
```

나의 로그인 셸은?

(로그인 셸: 로그인할 때 실행되는 셸) → /etc/passwd 에 지정되어 있음

```
sshd:x:115:65534:./var/run/sshd:/usr/sbin/nologin  
sheayun:x:1000:1000:sheayun,,,:/home/sheayun:/bin/bash  
lightdm:x:116:125:Light Display Manager:/var/lib/lightdm:/bin/false
```

셸 환경변수

- 환경 변수 (environment variables)
 - 셸이 프로그램들 사이에 미리 설정된 값들을 전달해 주는 역할을 하는 변수
 - 기본적으로 환경 변수는 대문자를 사용
- 설정 및 확인
 - 환경 변수의 값을 설정하기 위해서는 export 명령을 사용
\$ export variable_name=value
 - 환경 변수의 값을 확인하고 싶은 경우 printenv 명령 또는 env 명령을 사용
\$ printenv [variable_name]
\$ env

셸 환경변수 값 확인

```
sheayun@sheayun-ubuntu: ~  
sheayun@sheayun-ubuntu:~$ printenv PATH  
/home/sheayun/bin:/home/sheayun/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin  
sheayun@sheayun-ubuntu:~$
```

```
sheayun@sheayun-ubuntu: ~  
sheayun@sheayun-ubuntu:~$ env | grep PATH  
XDG_SESSION_PATH=/org/freedesktop/DisplayManager/Session0  
XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0  
DEFAULTS_PATH=/usr/share/gconf/ubuntu.default.path  
PATH=/home/sheayun/bin:/home/sheayun/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin  
MANDATORY_PATH=/usr/share/gconf/ubuntu.mandatory.path  
sheayun@sheayun-ubuntu:~$
```

grep - 표준 입력으로부터 특정 문자열 (또는 정규표현식) 이 포함된 행들을 골라 표준 출력으로

다른 환경변수는 잘 모른다 하더라도

- 셸에서 사용되는 대표적인 환경 변수는 PATH가 있음
 - PATH의 값은 디렉토리 이름의 연속
 - 탐색 경로를 정의하는 명령:
`$ export PATH="$PATH:/my_bin"`
- 환경 변수 파일
 - bin 디렉토리에는 주로 실행 프로그램을 보관하고 있으므로 기본 프로그램을 실행하기 위해서는 반드시 경로에 지정해 주어야 함
 - 일반적으로 ~/.bashrc 파일에 탐색 경로를 지정해 놓음

셸 스크립트 (Shell Scripts)

- 셸에 요청할 수 있는 명령들을 프로그래밍 형태로 파일에 저장하여 실행할 수 있음
 - 기본적인 프로그래밍 구조를 지원
- 지금 까지 실습했던 내용들을 모두 셸 스크립트로 작성 할 수 있음
 - 짧고 간단한 프로그램을 작성할 경우 매우 유용함
 - 많은 소프트웨어 패키지가 셸 스크립트를 설정, 설치 등에 이용하도록 구성되어 있음

셸 스크립트 (Shell Scripts) 예시

~/bashrc

→ 지금 이 내용을 이해할 필요는 없으나...

```
sheayun@sheayun-ubuntu: ~  
# ~/.bashrc: executed by bash(1) for non-login shells.  
# see /usr/share/doc/bash/examples/startup-files (in the package bash-doc)  
# for examples  
  
# If not running interactively, don't do anything  
case $- in  
    *i*) ;;  
    *) return;;  
esac  
  
# don't put duplicate lines or lines starting with space in the history.  
# See bash(1) for more options  
HISTCONTROL=ignoreboth  
  
# append to the history file, don't overwrite it  
shopt -s histappend  
  
# for setting history length see HISTSIZE and HISTFILESIZE in bash(1)  
HISTSIZE=1000  
HISTFILESIZE=2000  
  
# check the window size after each command and, if necessary,  
# update the values of LINES and COLUMNS.
```


간단한 셸 스크립트를 만들어보자

```
sheayun@sheayun-ubuntu: ~/testdir
sheayun@sheayun-ubuntu:~/testdir$ cat > myscript
#!/bin/bash
ls -l
id
sheayun@sheayun-ubuntu:~/testdir$
```

표준 입력을 myscript 라는 파일로
(redirection)

첫 줄에, "#!" 로 시작하여
어느 셸이 이 스크립트를
실행할지를 지정

정성들여 (틀리면 고치기 어려움, 처음부터 다시 해야)
입력한 후에 ctrl-D 를 눌러서 끝냄

그런데, 막상
실행하려 해 보니

```
sheayun@sheayun-ubuntu: ~/testdir
sheayun@sheayun-ubuntu:~/testdir$ ls -l
합계 4
-rw-rw-r-- 1 sheayun sheayun 19  4월 12 17:02 myscript
sheayun@sheayun-ubuntu:~/testdir$ ./myscript
bash: ./myscript: 허가 거부
sheayun@sheayun-ubuntu:~/testdir$
```

간단한 셸 스크립트를 실행해보자

```
sheayun@sheayun-ubuntu: ~/testdir
sheayun@sheayun-ubuntu:~/testdir$ ls -l
합계 4
-rw-rw-r-- 1 sheayun sheayun 19  4월 12 17:02 mvscript
sheayun@sheayun-ubuntu:~/testdir$ chmod u+x mvscript
sheayun@sheayun-ubuntu:~/testdir$ ls -l
합계 4
-rwxrw-r-- 1 sheayun sheayun 19  4월 12 17:02 myscript
sheayun@sheayun-ubuntu:~/testdir$
```

맞아, 실행 권한이 있어야겠지!

```
sheayun@sheayun-ubuntu: ~/testdir
sheayun@sheayun-ubuntu:~/testdir$ ./myscript
합계 4
-rwxrw-r-- 1 sheayun sheayun 22  4월 12 17:11 mvscript
uid=1000(sheayun) gid=1000(sheayun) 그룹들=1000(sheayun),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),110(lxd),114(sambashare),124(lpadmin)
sheayun@sheayun-ubuntu:~/testdir$
```

그런데, 맨 앞의 "./" 는 무엇인가요?
꼭 필요한가요?

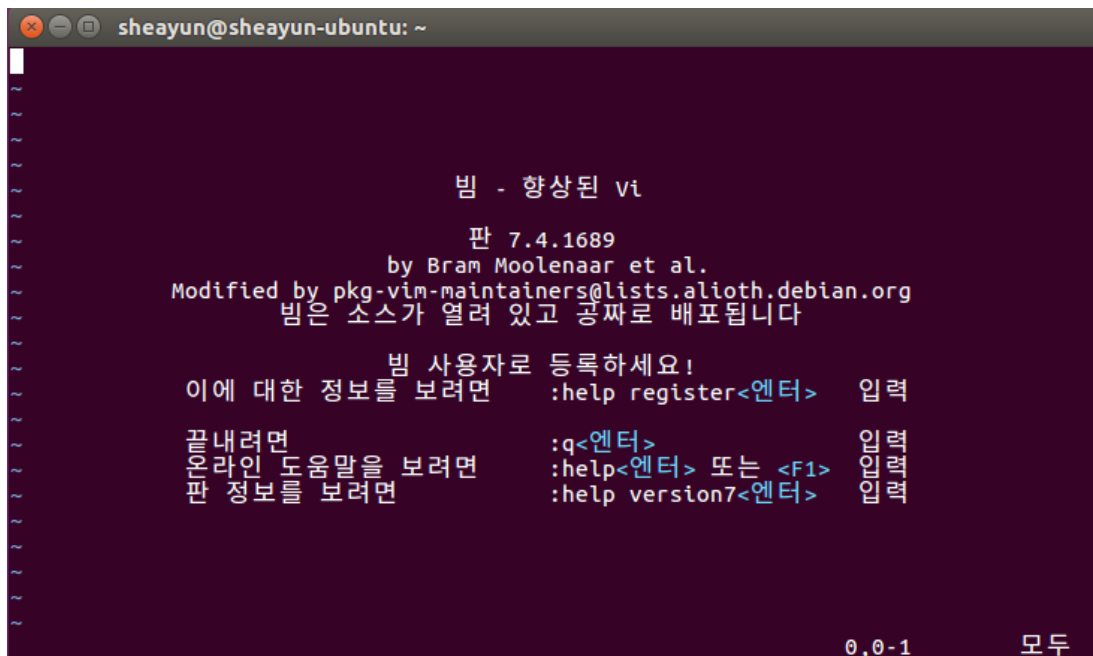
텍스트 편집 - vi

- vi 특징
 - 터미널 기반 텍스트 편집기
 - vi는 마우스와 방향키 없이도 사용가능
 - 익숙해지고 나면 (그러기가 쉽지 않지만) 텍스트 편집을 매우 효율적으로 할 수 있음
 - UNIX 스타일 시스템에는 (거의) 항상 있음
- vim 특징
 - vi보다 편의성을 더 향상시킨 형태
 - 최근에는 vim을 더 많이 사용



이제부터 vi

- vi <파일명>
 - 파일명을 명시하면 해당 파일이 열리고, 그렇지 않으면 빈 파일이 생성됨
 - cat 으로 이미 만들어진 파일을 열어보자
- 빈 화면이 나온다
 - 텍스트를 입력할 수 있는 편집 화면이 나옴
 - 처음에는 명령어 모드 상태임
 - 텍스트를 입력할 수 있는가
- vi의 3가지 모드
 - **명령어 모드**: vi에서 제공하는 명령어들을 사용하는 모드, 이 때 입력한 텍스트들은 문서에 삽입되지 않음
 - **입력 모드**: 텍스트를 입력하여 문서를 수정할 수 있는 모드
 - **ex 모드**: vi에서 제공하는 저장, 종료와 같은 기능을 이용할 수 있는 모드



```
sheayun@sheayun-ubuntu: ~  
  
      vim - 향상된 vi  
  
      판 7.4.1689  
      by Bram Moolenaar et al.  
Modified by pkg-vim-maintainers@lists.alioth.debian.org  
      vim은 소스가 열려 있고 공짜로 배포됩니다  
  
      vim 사용자로 등록하세요!  
이에 대한 정보를 보려면      :help register<엔터>      입력  
  
끝내려면                      :q<엔터>              입력  
온라인 도움말을 보려면      :help<엔터> 또는 <F1>      입력  
판 정보를 보려면            :help version7<엔터>      입력  
  
0,0-1      모두
```

vi 빠져나오기

```
빔 - 향상된 vi

판 7.4.1689
by Bram Moolenaar et al.
Modified by pkg-vim-maintainers@lists.alioth.debian.org
빔은 소스가 열려 있고 공짜로 배포됩니다

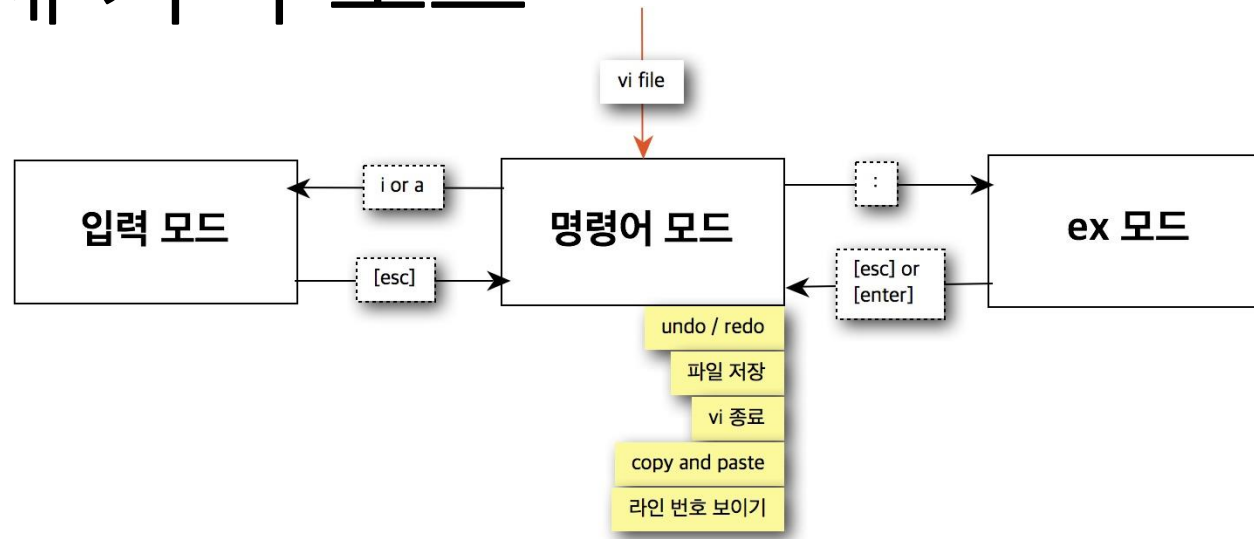
빔 사용자로 등록하세요!
이에 대한 정보를 보려면      :help register<엔터>    입력
끝내려면                      :q<엔터>                입력
온라인 도움말을 보려면       :help<엔터> 또는 <F1>    입력
판 정보를 보려면             :help version7<엔터>      입력
```

(1) ":" 키를 눌러서 명령어 모드 → ex 모드 전환

(2) ex 모드에서 "q" 다음 <Enter> 키를 눌러서 빠져나온다.

빠져나오지 못하고 있으면 "q!" 라고 입력하고 <Enter> 해보자.

vi 의 세 가지 모드



- 명령어 모드

- 파일을 열고 (vi <파일명>) 들어온 후 최초 상태
- 명령어모드에서 'i', 'a' 키 등 입력 모드로 진입하는 키들이 있음
- 명령어모드에서 ':' 를 입력하면 ex 모드로

- 입력 모드

- 파일의 내용을 텍스트로 편집할 수 있음
- <ESC> 키를 누르면 명령어 모드로

- ex 모드

- write, cancel, quit, help 와 같은 기능을 이용할 수 있는 실행 모드
- <ESC> 나 <Enter> 키를 누르면 명령어 모드로 전환

가장 간단한 vi 실습

- cat 을 이용해서 새로운 파일을 만든다.
- 이 파일을 vi 로 열고 (최초 상태는 명령어 모드)
- 입력 모드로 가서 입력한 후
- 명령어 모드로 가고 (어떻게?)
- ex Mode로 이동한 후 (어떻게?)
- 파일 저장함과 동시에 vi 종료 (어떻게?)

가장 간단한 vi 실습

```
sheayun@sheayun-ubuntu: ~/testdir  
sheayun@sheayun-ubuntu:~/testdir$ cat > new.txt  
This is a test file.  
sheayun@sheayun-ubuntu:~/testdir$ vi new.txt
```

```
sheayun@sheayun-ubuntu: ~/testdir  
This is a test file.  
~  
~  
~
```

```
sheayun@sheayun-ubuntu: ~/testdir  
This is a test file.  
I write another line here.  

```

<ESC> → ':' → 'wq' → <Enter>

익숙해지기 vi 실습

- 파일을 열고
- 간단한 내용 입력후
- copy and paste 로 복잡하게 만듦
- 저장 및 종료 후
- cat 으로 내용 확인
- 다시 들어가서 내용 일부 삭제
- undo 기능을 사용하고
- 저장 및 종료 후
- cat 으로 내용 확인

vi 설정 변경 (1)

- set number
 - line 표시를 해줍니다.
- set ai
 - auto indent
- set si
 - smart indent
- set cindent
 - c style indent
- set tabstop=4
 - tab을 4칸으로
- set ignorecase
 - 검색시 대소문자 구별하지않음
- set hlsearch
 - 검색시 하이라이트(색상 강조)
- set expandtab
 - tab 대신 띄어쓰기로
- set background=dark
 - 검정배경을 사용할 때, (이 색상에 맞춰 문법 하이라이트 색상이 달라집니다.)

vi 설정 변경 (2)

- set nocompatible
 - 방향키로 이동가능
- set fileencodings=utf-8,euc-kr
 - 파일인코딩 형식 지정
- set history=1000
 - 명령어에 대한 히스토리를 1000개까지
- set ruler
 - 상태표시줄에 커서의 위치 표시
- set nobackup
 - 백업파일을 만들지 않음
- set title
 - 제목을 표시
- set showmatch
 - 매칭되는 괄호를 보여줌
- set nowrap
 - 자동 줄바꿈 하지 않음
- set wmnu
 - tab 자동완성시 가능한 목록을 보여줌
- syntax on
 - 문법 하이라이트 기능

이런 설정들을 ~/.vimrc 파일에 적어 두면,
vi(m) 이 실행할 때마다 설정 자동 적용됨

vi 에서 특정 문자열 검색하기

- **/**<검색어>
 - <검색어>를 찾아줌, 위에서 아래로 검색
- **?**<검색어>
 - <검색어>를 찾아줌, 아래에서 위로 검색
- 검색이 된 후 사용할 수 있는 명령어
 - **n**: 그 다음 검색 (검색 방향은 유지)
 - **N**: 그 다음 검색 (역방향으로 검색)
- 대/소문자 구분 없이 검색
 - set ignorecase (ex 모드에서 실행)

vi 에서 특정 문자열 치환하기

- : [범위]/[매칭문자열]/[치환문자열]/[행범위]
 - 예) %s /old /new /g
- 예
 - :s/old/new
 - 현재 행의 처음 old를 new로 교체
 - :s/old/new/g
 - 현재 행의 모든 old를 new로 교체
 - :10, 20s/old/new/g
 - 10번째 행부터 20번째 행까지 모든 old를 new로 교체
 - :-3, +4s /old/new/g
 - 현재 커서 위치에서 3행 위부터 4행 아래까지 old를 new로 교체
 - :%s/old/new/g
 - 문서 전체에서 old를 new로 교체
 - :%s/old/new/gc
 - 문서 전체에서 old를 new로 확인하며 교체

vi cheat sheet (Cheat Sheet)

version 1.1
April 1st, 06

vi / vim graphical cheat sheet

Esc
normal mode

Dvorak

~ toggle case	! external filter	@ play macro	# prev ident	\$ eol	% goto match	^ "soft" bol	& repeat :s	* next ident	(begin sentence) end sentence	{ begin parag.	} end parag.
· goto mark	1	2	3	4	5	6	7	8	9	0 "hard" bol	[misc] misc
" reg. spec	< un-indent	> indent	P paste before	Y yank line	F "back" find ch	G eof/ goto ln	C change to eol	R replace mode	L screen bottom	? find (rev.)	+ next line	
' goto mk. bol	reverse t/T/f/F	· repeat cmd	p paste after	y yank	f find char	g extra cmds	c change	r replace char	l →	/ find	= auto format	
A append at eol	O open above	E end WORD	U undo line	I insert at bol	D delete to eol	H screen top	T back 'till	N prev (find)	S subst line	"soft" bol down		
a append	o open below	e end word	u undo	i insert mode	d delete	h ←	t 'till	n next (find)	s subst char	- prev line		
· ex cmd line	Q ex mode	J join lines	K help	X back-space	B prev WORD	M screen mid'l	W next WORD	V visual lines	Z quit	bol/ goto col		
· repeat t/T/f/F	q record macro	j ↓	k ↑	x delete char	b prev word	m set mark	w next word	v visual mode	Z extra cmds	\ not used!		

- motion** moves the cursor, or defines the range for an operator
- command** direct action command, if **red**, it enters insert mode
- operator** requires a motion afterwards, operates between cursor & destination
- extra** special functions, requires extra input

q.

bol = beginning of line, eol = end of line, mk = mark, yank = copy

words: `quux(foo, bar, baz);`
WORDS: `quux(foo, bar, baz);`

Main command line commands ('ex'):

:w (save), :q (quit), :q! (quit w/o saving)
:e f (open file f),
:%s/x/y/g (replace 'x' by 'y' filewide),
:h (help in vim), :new (new file in vim),

Other important comands:

CTRL-R: redo (vim),
CTRL-F/-B: page up/down,
CTRL-E/-Y: scroll line up/down,
CTRL-V: block-visual mode (vim only)

Visual mode:

Move around and type operator to act on selected region (vim only)

Notes:

- (1) use "x before a yank/paste/del command to use that register ('clipboard') (x=a..z,*) (e.g.: "ay\$ to copy rest of line to reg 'a')
- (2) type in a number before any action to repeat it that number of times (e.g.: 2p, dzw, 5i, dj)
- (3) duplicate operator to act on current line (dd = delete line, >> = indent line)
- (4) ZZ to save & quit, ZQ to quit w/o saving
- (5) zt: scroll cursor to top, zb: bottom, zz: center
- (6) gg: top of file (vim only), gf: open file under cursor (vim only)

For a graphical vi/vim tutorial & more tips, go to www.viemu.com - home of ViEmu, vi/vim emulation for Microsoft Visual Studio

vi 종합 실습

- 파일 다운로드 (correct.txt, incorrect.txt)
- 파일의 내용을 서로 비교
- vi 로 incorrect.txt 를 편집하여 correct.txt 와 같도록
- 완전히 같아질 때까지 편집 (연습!)

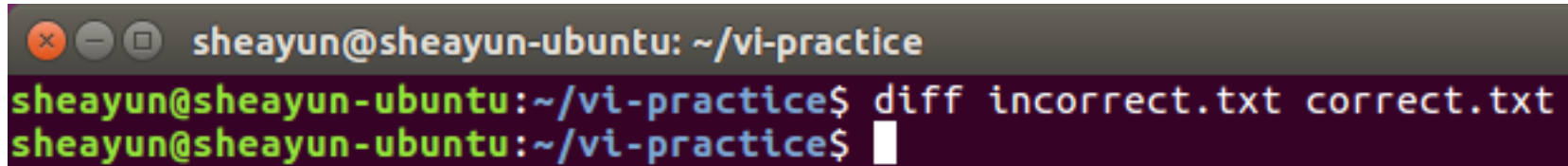
파일의 내용 비교

```
sheayun@sheayun-ubuntu: ~/vi-practice
sheayun@sheayun-ubuntu:~/vi-practice$ ls -l
합계 8
-rw-rw-r-- 1 sheayun sheayun 3876  4월 13 16:03 correct.txt
-rw-rw-r-- 1 sheayun sheayun 3840  4월 13 16:04 incorrect.txt
sheayun@sheayun-ubuntu:~/vi-practice$ diff incorrect.txt correct.txt | head -n 10
5c5
< All in the silver afternoon
---
> All in the golden afternoon
7,8c7,9
< For both our oares, with little skill,
< By little arms are plied, While little hands make vain pretence
---
> For both our oars, with little skill,
> By little arms are plied,
sheayun@sheayun-ubuntu:~/vi-practice$
```

diff - 파일의 내용을 서로 비교 (다른 곳을 알려줌)

head - 표준 입력으로부터 n 행 (여기에서는 10) 만큼만 표준 출력에 표시

파일의 내용 비교

A terminal window with a dark background. The title bar shows window control icons and the text 'sheayun@sheayun-ubuntu: ~/vi-practice'. The prompt is 'sheayun@sheayun-ubuntu:~/vi-practice\$'. The command 'diff incorrect.txt correct.txt' has been entered. The next line shows the prompt again with a cursor, indicating no output was produced.

```
sheayun@sheayun-ubuntu: ~/vi-practice
sheayun@sheayun-ubuntu:~/vi-practice$ diff incorrect.txt correct.txt
sheayun@sheayun-ubuntu:~/vi-practice$
```

파일의 내용이 완전히 동일 → diff 에 의하여 아무 것도 출력되지 않음

vi 활용 연습 이외에도 diff 의 출력을 해석하는 연습은 매우 중요

(이후 프로그램 소스 코드의 변경사항을 추적할 때 “늘” 보게 될 것임)