Unix(Linux) shell programming

1-1. UNIX System 소개

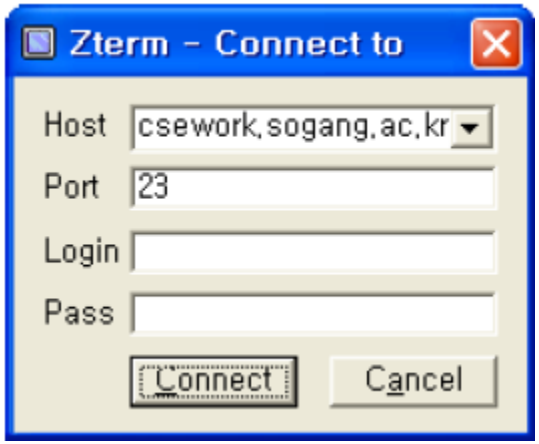
- Unix란
  - 주로 중대형 컴퓨터및 워크스테이션에서 많이 사용하는 OS
  - Linux – 일반 PC에서 주로 사용되는 Unix
- Unix 의 역사
  - 60년대 AT&T의 Bell 연구소에서 개발한 Multics 운영체제가 너무 복잡하여 개발하게 되었다.
  - 1971년 Bell 연구소에서 유닉스 시스템의 최초의 버전이 나오게 되었다.
  - C 언어가 유닉스에서 쓰이기 위해 개발되고, 유닉스 운영체제도 C로 다시 프로그램 되었다.
  - 인터랙티브 시스템사가 1977년 유닉스를 사무자동화를 위한 상업용으로 팔기 시작하였다.
  - System V vs. BSD
  - Solaris (Sun), IRIX (IBM), HP-UX (HP), ...

Unix System 기초

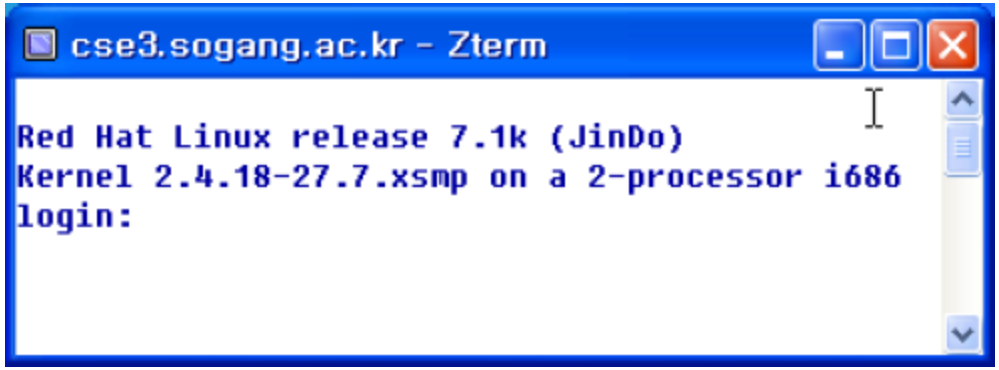
- **접속 방법**
  - 원격 접속 방법
    - telnet
    - ssh
  - 터미널 접속 방법
    - 서버에 직접 연결되어 있는 모니터와 키보드, 마우스를 통해 연결
  - 일반 사용자들은 Unix System 에 접속하기 위해서 보통 원격 접속 프로그램을 사용한다.
    - zterm, putty, SSH Secure Shell Client, SecureCRT, Xmanager 등

Unix System 기초

- zterm을 이용한 실행
  - zterm은 콘솔용 프로그램이다.



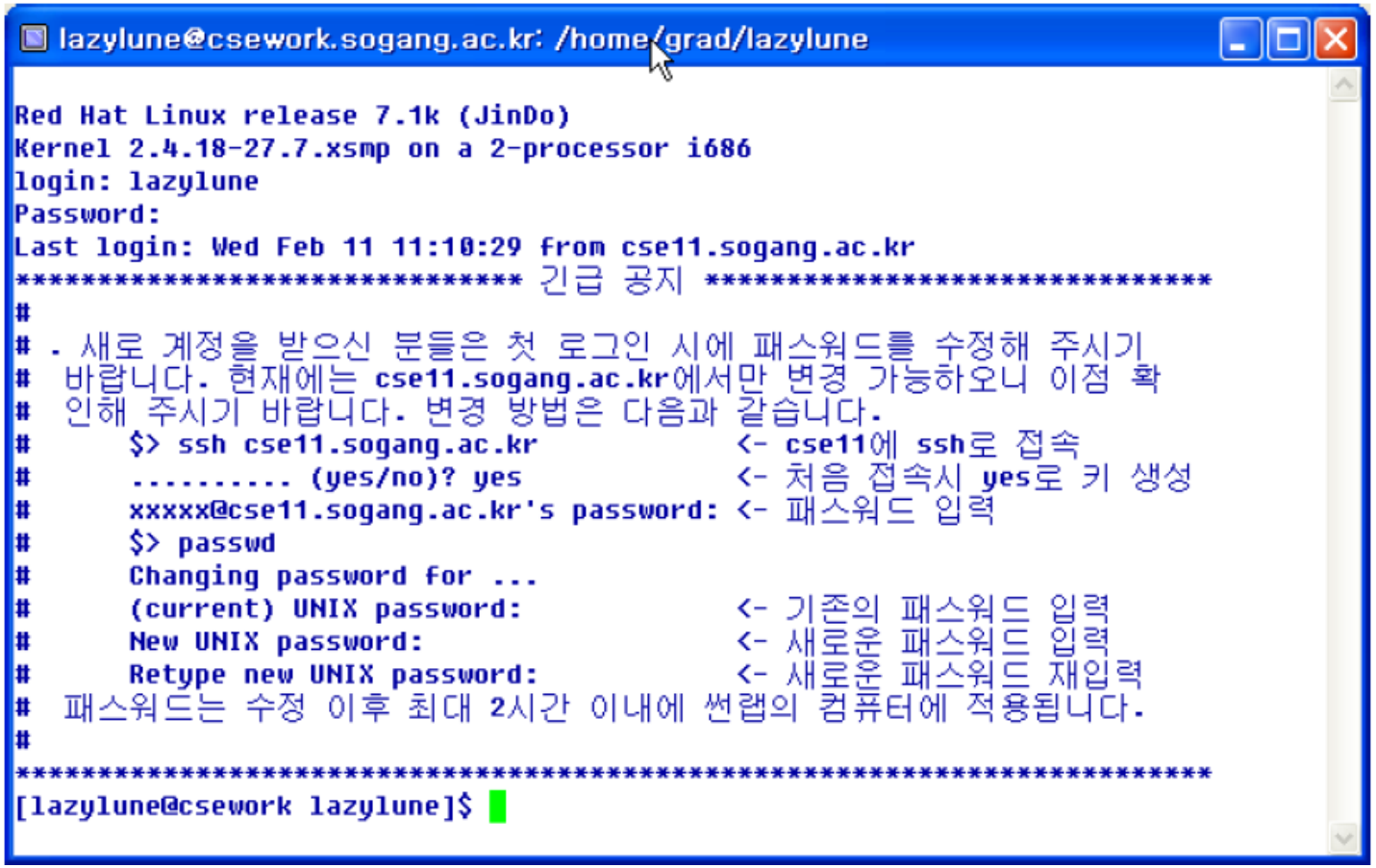
zterm 실행화면



zterm 로그인 화면

Unix System 기초

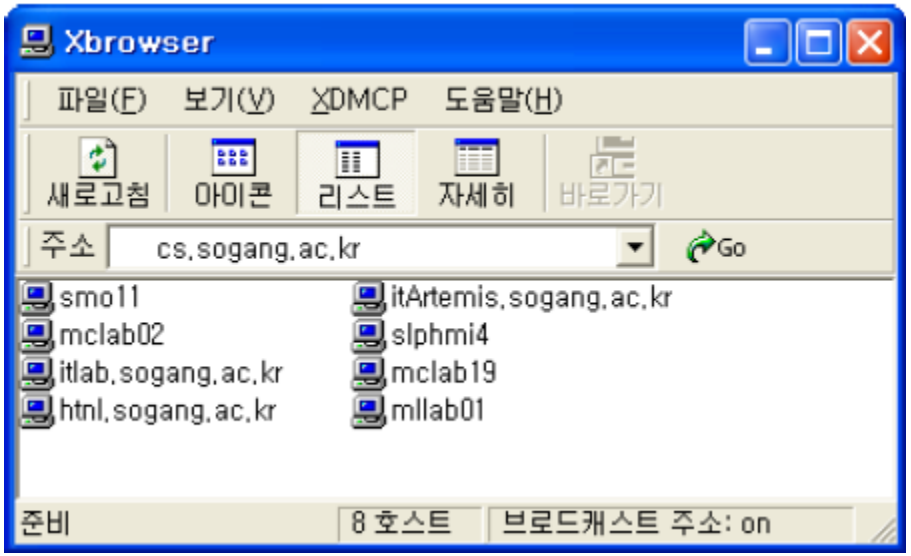
- zterm을 이용한 실행



zterm
작업화면

Unix System 기초

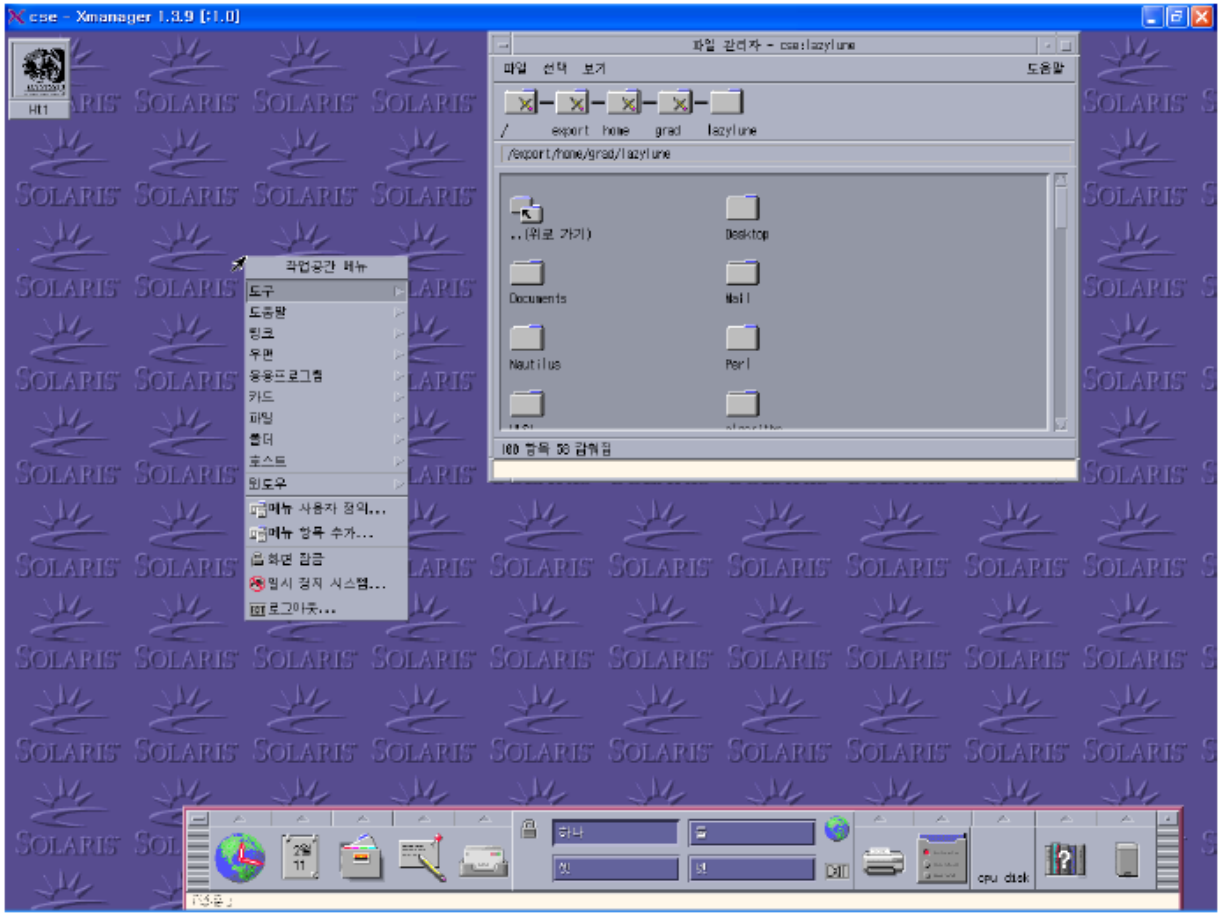
- Xmanager를 이용한 접속
  - Xmanager는 원격에서 Unix System에 GUI모드로 접속할 수 있게 해주는 프로그램이다.



Xbrouser 실행 결과.

Unix System 기초

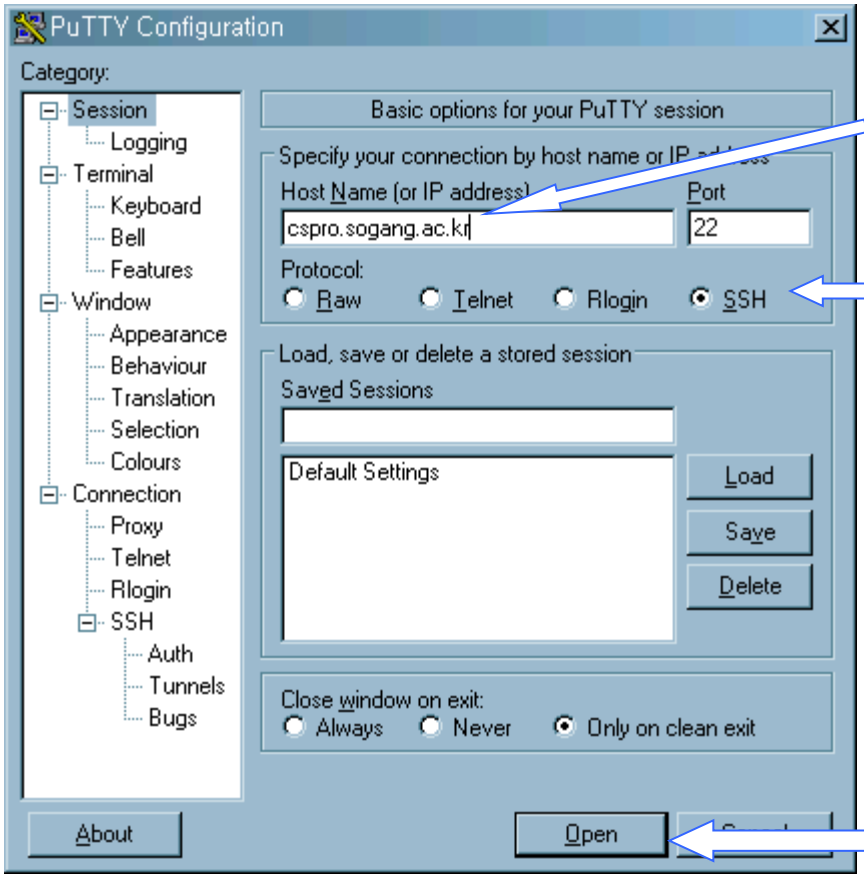
- Xmanager를 이용한 접속



Xmanager
실행화면

Unix System 기초

- Putty를 이용한 접속 Session Save 기능 활용하기
  - Unix 서버에 접속 할 때 주로 이 프로그램을 많이 이용한다.



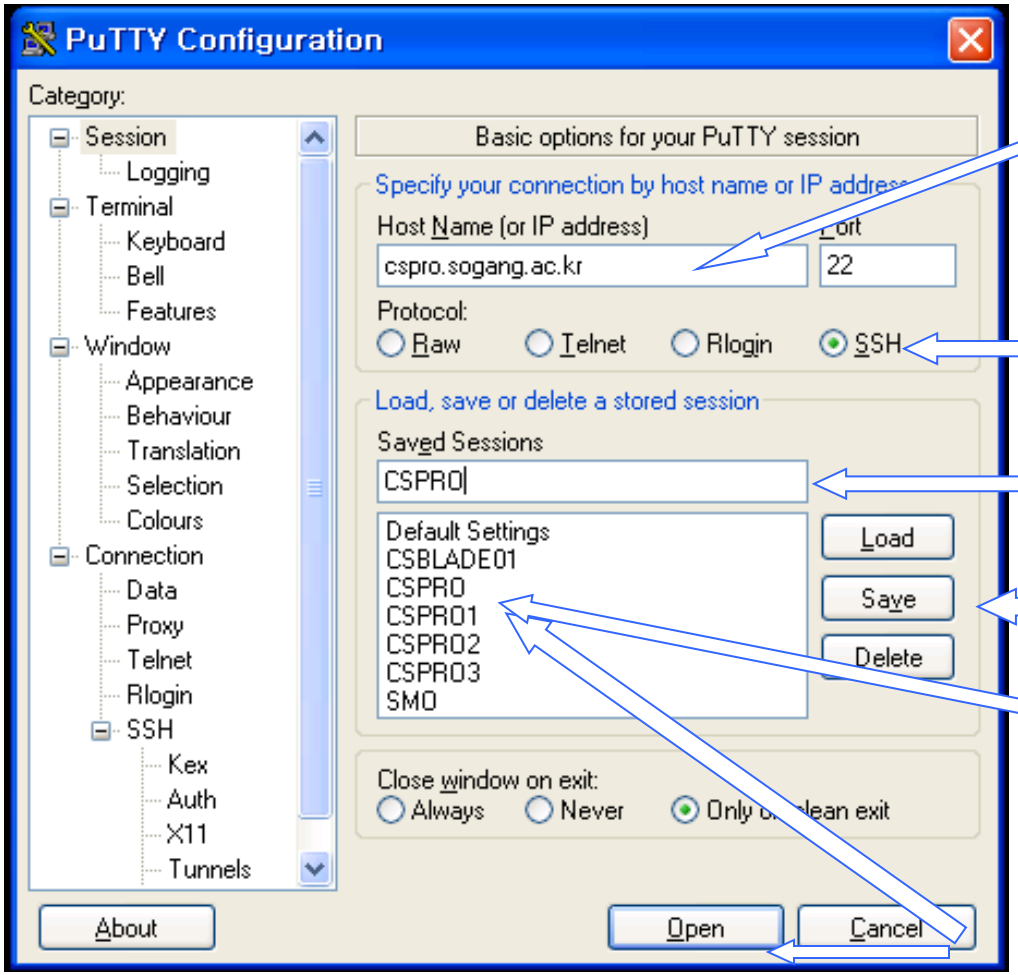
1. 주소창에 cspro.sogang.ac.kr 을 입력한다.

2. ssh 프로토콜을 선택한다.

3. Open 버튼을 누른다.

Unix System 기초

- Putty의 Session Save 기능 활용하기



1. 주소창에 cspro.sogang.ac.kr 을 입력한다.

2. ssh 프로토콜을 선택한다.

3. Session 명을 적는다.

4. Save 버튼을 누른다.

접속방법 1.

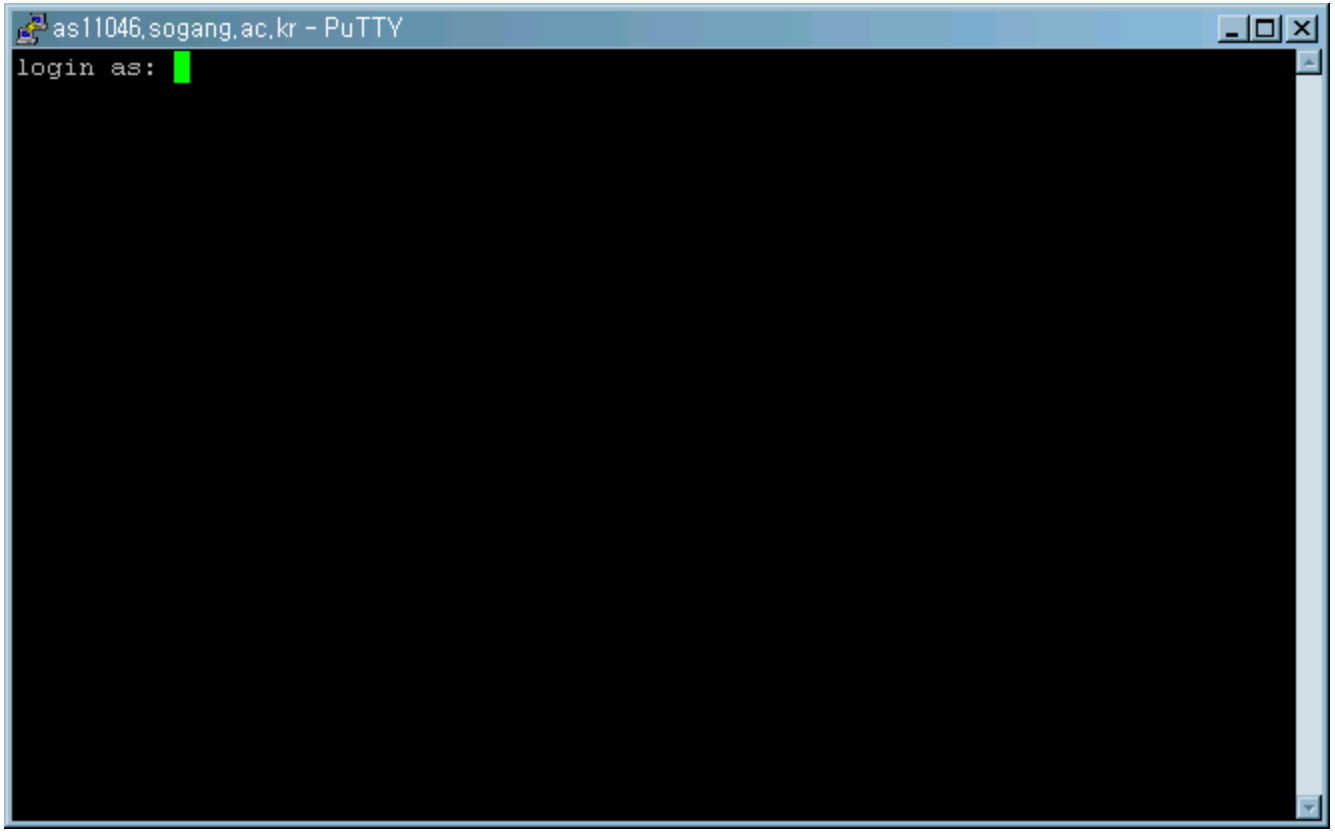
저장된 Session명을 더블클릭 한다.

접속방법 2.

저장된 Session명을 클릭 후 open을 클릭한다.

Unix System 기초

- 로그인하기
  - 아래와 같이 login창이 뜨면 자신의 id와 password를 입력한다.



Unix System 기초

- Putty에서 한글보이게 하는 방법

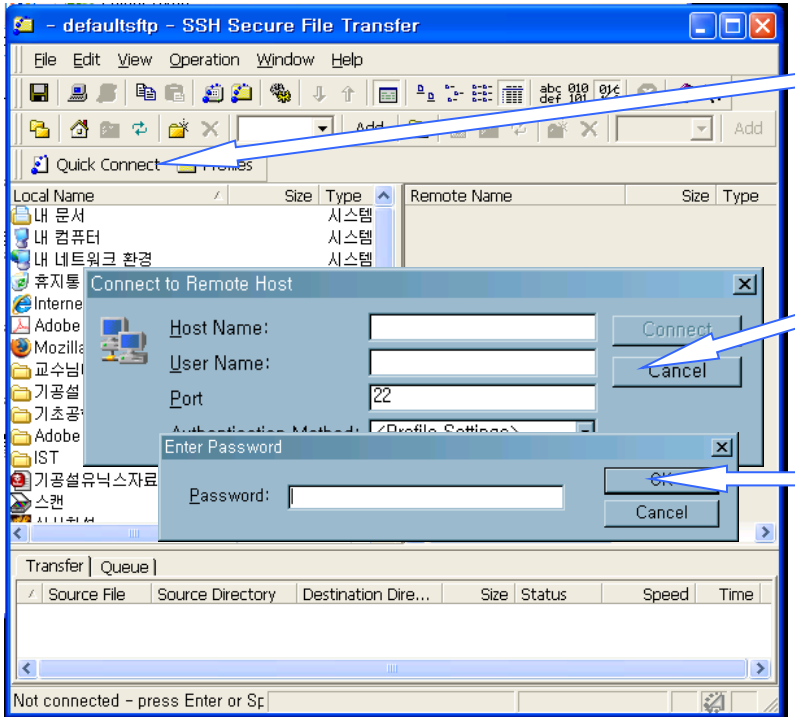
1. 한글 putty를 다운로드 받아 사용

2. 윈도우 시작메뉴에서 '실행' 메뉴 클릭 → 'regedit'를 적고 확인을 누름 → regedit → HKEY_CURRENT_USER → Software → SimonTatham → PuTTY → Sessions → 설정을 원하는 세션선택 → FontCharSet 16 진수 0 을 16 진수 81 로 변경

3. Category에서 Window 선택 → Appearance → Font Setting → Change → 굴림 → 스크립트 : 한글 → 다시 session tab 으로 와서 cspro로 접속

Unix System 기초

- SSH Secure Shell을 이용한 접속
  - SSH Secure Shell프로그램을 설치한 뒤 Secure Shell Client프로그램을 실행한다.
  - 아래와 같은 창이 뜨면 Quick Connect 버튼을 누른 뒤 Host name에 cspro.sogang.ac.kr을 입력하고, User Name에 자신의 login id를 입력한 뒤 Connect버튼을 누른다.



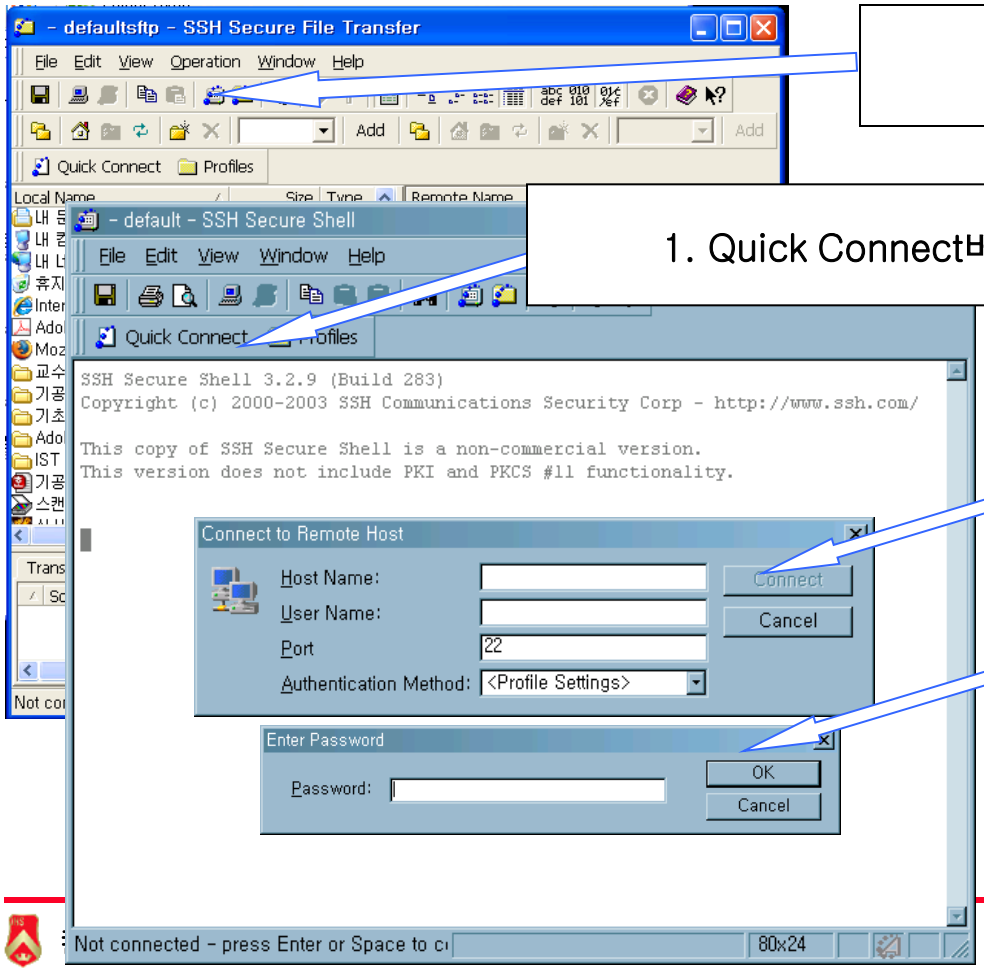
1. Quick Connect버튼을 누른다.

2. **Host name에 cspro.sogang.ac.kr을 입력하고, User Name에 자신의 login id를 입력한 뒤 Connect버튼을 누른다.**

3. 자신의 password를 입력한 뒤 OK버튼을 누른다.

SSH Secure Shell 이용

- 아래와 같은 창이 뜨면 Quick Connect 버튼을 누른 뒤 Host name에 cspro.sogang.ac.kr을 입력하고, User Name에 자신의 login id를 입력한 뒤 Connect버튼을 누른다.



0. 이 버튼을 클릭한다.

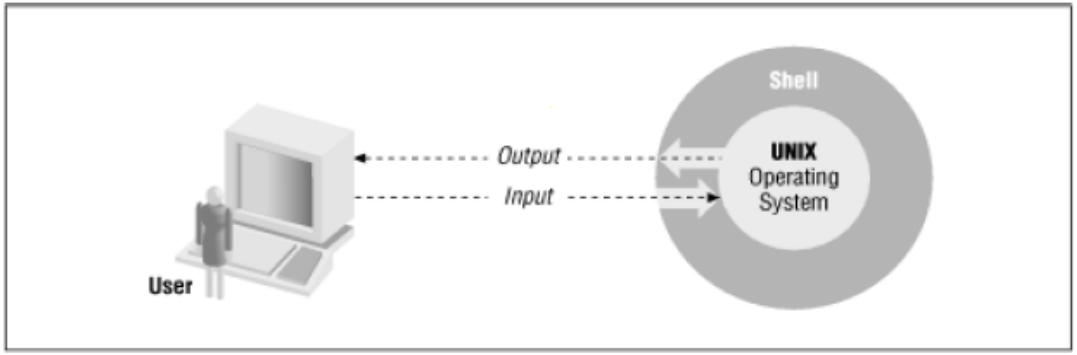
1. Quick Connect버튼을 누른다.

2. **Host name에 cspro.sogang.ac.kr을 입력하고, User Name에 자신의 login id를 입력한 뒤 Connect버튼을 누른다.**

3. 자신의 password를 입력한 뒤 OK버튼을 누른다.

Unix System 기초

- Shell
  - Shell은 unix OS와 사용자 간에 대화 소통을 원활하게 해주는 역할을 한다.



shell의 역활

Unix System 기초

- Shell (Cont')
  - *sort –n phonelist > phonelist.sorted*
    - 명령행을 sort, –n, phonelist, >, phonelist.sorted와 같은 조각으로 나눈다. 이러한 조각들을 워드(word) 라 한다.
    - 각 워드가 의미하는 바를 파악한다.
    - > phonelist.sorted에 따라 I/O를 설정한다.
    - sort 명령을 –n 옵션으로 하고 phonelist를 인자로 하여 실행한다.
  - 종류
    - Bourne Shell (sh), C Shell (csh), Korn Shell (ksh), Bourne Again Shell (bash), tcsh등

Unix System 기초

- 명령어 실행
  - Shell 은 입력 받은 명령어를 해석, 실행하고 그 결과를 돌려주는 역할을 수행한다.

예제

```
$ who ; ps
lazylune pts/0        Feb 11 20:20
  PID TTY          TIME CMD
17281 pts/0    00:00:00 bash
17332 pts/0    00:00:00 ps
$
```

```
$ ps
  PID TTY          TIME CMD
17281 pts/0    00:00:00 bash
17388 pts/0    00:00:00 ps
$
```

Unix System 기초

- Standard Input/Output/Error, Redirection, Pipe
  - Unix 에서 명령어는 Standard Input을 통해서 입력받고, 처리된 결과는 Standard Output을 통해서 출력된다.
  - 발생한 error 는 Standard Error를 통해 출력된다.
  - 파일을 통해서 입력이 되거나 출력을 파일로 저장하고자 할 때 Redirection(<, >)을 이용한다.

```
$ ps > test
$
```

```
$ ps >> test
$
```

```
$ cat test
  PID TTY          TIME CMD
17281 pts/0    00:00:00 bash
17422 pts/0    00:00:00 ps
  PID TTY          TIME CMD
17281 pts/0    00:00:00 bash
17433 pts/0    00:00:00 ps
$
```

Unix System 기초

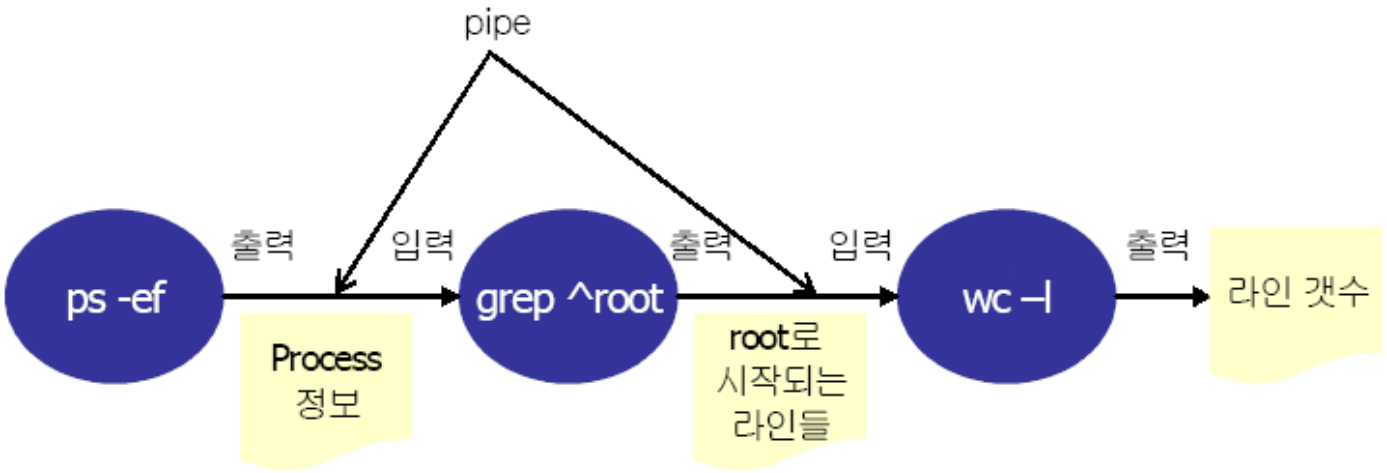
- Standard Input/Output/Error, Redirection, Pipe

```
$ wall
Hello
^d
Broadcast message from lazylune (pts/0) (Wed Feb 11 21:50:49 2004):

Hello
$ cat > test
hello
^d
$ wall < test
$
Broadcast message from lazylune (Wed Feb 11 21:51:10 2004):

hello

$
```

Unix System 기초

- Standard Input/Output/Error, Redirection, Pipe
  - Pipe(|, shift + ₩)를 이용하면 한 명령의 표준 출력을 다른 명령의 표준 입력으로 보낼 수 있다.

```
$ ps -ef | grep ^root | wc -l
    41
$
```



Pipe 처리 과정

Unix System 기초

- Meta Character

특수문자	용도
cmd;	Command terminator cd; ls execute sequentially
cmd &	Run preceding command in the background
> file	Output redirection
>> file	Appending output redirection
< file	Input redirection
<< word	Read standard input up to a line identical word
cmd | cmd	From a pipeline between preceding command and the following command
*	The * is used in file name generating to the match any sequence of characters ls new*
?	The ? ls used in file name generation to match any single character in a file name ls new?

Unix System 기초

- Meta Character

[set]	The [introduces a character set for file name generation and]closes the set ls [D-R]*
-	Indicates a character range in a character class
$word	The word following the $ will be treated as a parameter and will be replaced by its value
\c	The character following the backslash will be quoted
'text'	No substitutions will occur
"text"	Parameter and command substitution occur
(list)	Execute a command list in a subshell (date; who; pwd) > file
{list}	Execute a command list in the current shell
cmd && cmd	Execute the second command only if the first completes with zero exit status
cmd || cmd	Execute the second command only if the first completes with a non-zero exit status

Unix System 기초

- 파일 시스템
  - Path name
    - Unix에는 디렉토리를 나타내는 특수한 문자가 존재한다. “.”은 현재 디렉토리를 나타내며 “..”은 현재 디렉토리의 상위 디렉토리를 나타내고, “~”는 home 디렉토리를 나타낸다.
    - 절대 경로 (root로부터 시작)
      - /home/lazylune/project/os/hw3/
    - 상대 경로
      - ../../../../os/hw3
      - ~/project/os/hw3 (현재 로긴되있는 계정의 홈디렉토리)
      - ~lazylune/project/os/hw3 (~username 해당유저의 홈디렉토리를 나타내는 상대경로)

Unix System 기초

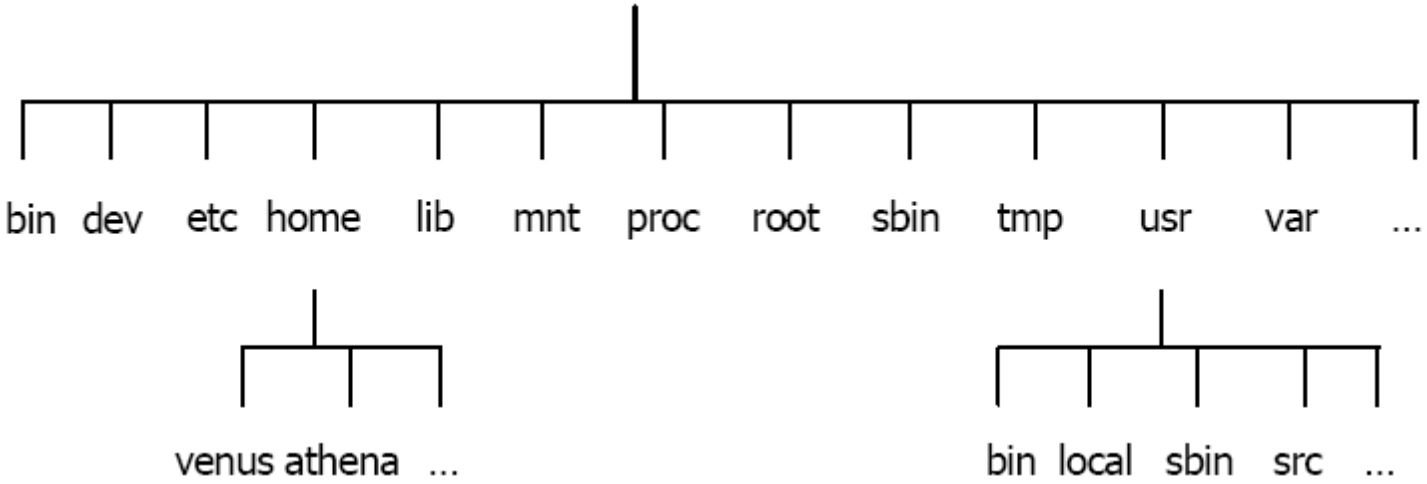
파일 시스템

Permission

- Unix는 multi-user System 이다. 따라서 사용자가 자신의 Directory와 File에 접근하는 위험이 항시 존재한다.
- Permission 이란 해당 파일에 대한 접근 권한을 뜻한다.
- 파일에 점근하는 사용자를 크게 소유자, 그룹, 다른 사용자의 3가지로 분유하여 각각에 대하여 읽기, 쓰기, 실행 권한을 줄 수 있다.
- 예 > drwxr-xr-x

Unix System 기초

- Unix System Directory 구조



Unix 디렉토리 구조

Unix System 기초

- Unix System Directory 구조

디렉토리	내 용
/	최상위 디렉토리로 "/"로 표시하고 root(루트)로 불린다. 루트 디렉토리를 중심으로 유닉스 시스템의 중요한 파일들이 계층 구조를 이루고 있다.
/bin	유닉스 시스템의 명령어 중에서 중요하고도 기본적인 실행 파일들이 모여 있는 곳이다.
/dev	시스템에 관련된 장치들을 모아놓은 곳이다. 유닉스는 모든 장치를 다 파일로 인식한다.
/etc	암호 파일과 시스템 설정에 관련된 중요한 파일들이 모여 있는 곳이다. 일반 유저 보다는 시스템 관리자가 더 잘 알아야 하는 곳이다.
/home	사용자들의 홈 디렉토리가 위치하는 곳이다. 가령 계정명이 sogang인 사용자의 홈 디렉토리는 /home/sogang/으로 설정된다.
/lib	UNIX 시스템에서 사용되는 라이브러리 파일들을 모아놓은 곳이다.
/mnt	기본적인 파티션 외의 다른 저장 장치가 마운트 되는 곳이다. UNIX에서는 기본적으로 파티션(디스크의 일정 공간)을 사용하기 위해서는 마운트 작업을 수행해야 하는데, cdrom이나 Windows의 파티션이 여기에 마운트 될 수 있다.
/proc	UNIX 시스템의 정보가 들어있는 곳이다. 이곳에는 현재 실행되고 있는 프로세스의 정보라든지 사용되고 있는 장치명 등 UNIX 커널에 관련된 정보들이 저장되어 있다.

Unix System 기초

- Unix System Directory 구조

/root	슈퍼유저(root)의 홈 디렉토리이다. 유닉스에서 슈퍼유저는 권한을 무시하고 무엇이든지 할 수 있는 유저라고 보면 된다. 따라서 많은 해커들이 슈퍼유저의 권한을 얻기 위해 애를 쓴다.
/sbin	bin이 일반 사용자들이 사용하는 명령들을 저장하고 있다면, 이곳에서는 시스템 관리를 위한 명령들을 저장하고 있다.
/tmp	임시 저장공간으로 사용자라면 누구나 읽고 쓸 수 있는 디렉토리이다.
/usr	/usr 파일 시스템은 실제 사용자들과의 관계가 없어도 user file system이라고 부른다. /usr 디렉토리는 실행가능한 명령, 프로그램에 사용되는 헤더 파일, 라이브러리 그리고 시스템 관련 유틸리티 등이 들어 있는 곳이다. /bin과 /sbin의 전형적인 UNIX 프로그램들이 저장되는 공간이라면 /usr은 기본 프로그램들 외에 추가적으로 설치되는 프로그램들이 저장되는 공간이라고 보면 된다.
/var	시스템 접속 상황, 프린터 기록, 인스톨 결과, 패키지 정보, 패치 정보 등 시스템에 관한 모든 기록들이 남겨지는 곳이다.

Unix 기본 명령어

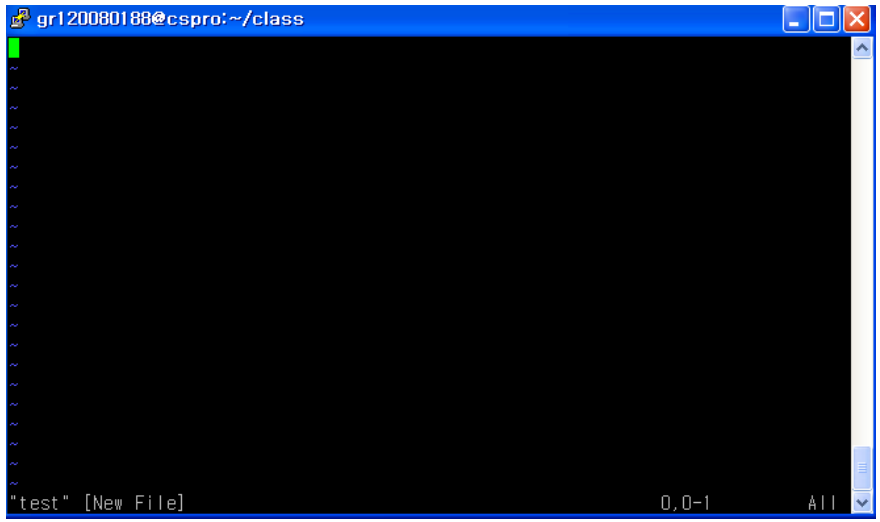
- ls : dir과 동일한 명령어. 현재 폴더에 들어있는 파일들의 목록을 보여줌.
  - ls -al : 숨김 파일, 권한 설정 등 파일의 자세한 정보와 함께 파일들의 목록을 보여줌
- passwd : 자신의 비밀번호를 변경. (cspro에서만 가능, cspro1, cspro2.. 는 불가)
- pwd : 현재 위치를 확인
- who : 현재 접속한 user의 정보 확인
- cp : 파일 복사
- mv : 파일 이동
- rm : 파일 삭제 (하위 디렉토리까지 지우려면 rm -rf)
- mkdir : 디렉토리 생성
- rmdir : 디렉토리 삭제 (해당 폴더에 파일이 들어있으면 지울 수 없다. 이때는 rm -rf 사용)
- cd : 작업 위치를 변경
  - "cd" 만 칠 경우 자신의 홈디렉토리로 이동.
  - "cd 디렉토리"를 치면 해당 디렉토리로 이동.
  - "cd .."을 입력할 경우 한 단계 상위 디렉토리로 이동
- exit : 현재 세션 종료
- ps (ps -aux) : 현재 수행중인 프로세스 목록
- kill : 해당 프로세스를 강제 종료
- ./파일명 : 파일 실행

vi 설정 및 사용법

vi 기본 사용법

- vi는 Visual display editor로 유닉스 상에서 가장 널리 쓰이는 텍스트 편집기임.
- Unix의 vi가 소스코드가 공개되지 않은 관계로 Linux 상에서는 vi 역할을 하는 Vim (Vi Improved)이라는 텍스트 편집기를 사용. linux에서 vi 명령어를 치면 vim이 실행됨.
- 보통 "vi 파일이름"으로 실행시킨 후 저장 후 종료하면 해당 파일이 생성됨.

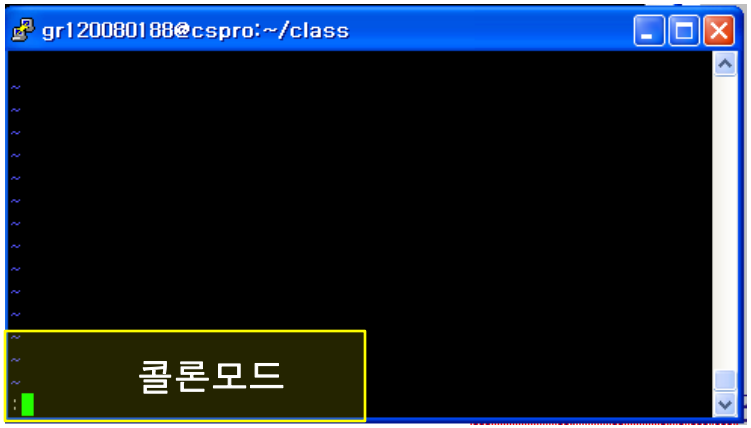
```
gr120080188@cspro ~/class $ vi test
```

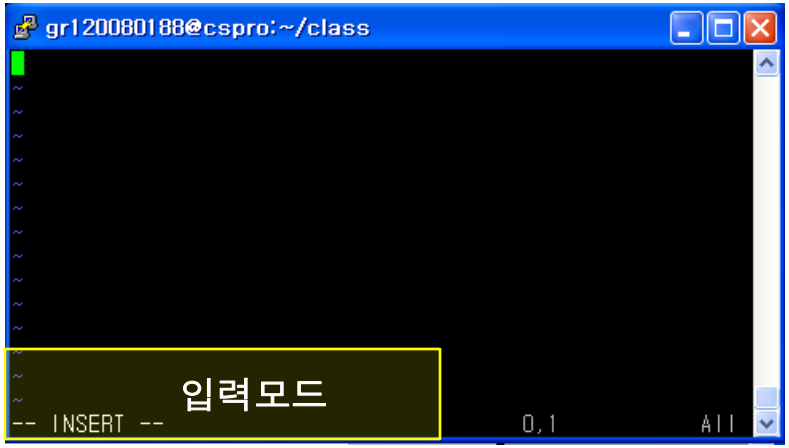


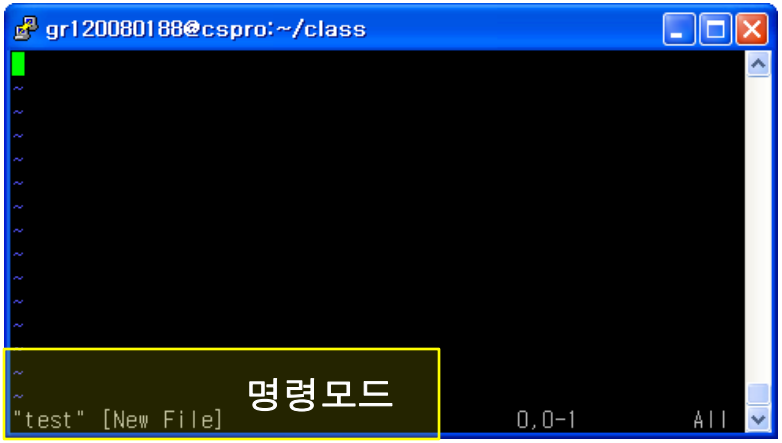
실행화면

vi editor

- 기본 명령어
  - vi에는 명령 모드, 입력 모드, 콜론 모드 이렇게 3가지 명령어 모드가 있다.







vi 명령어 - 입력모드

- 입력 모드 전환
  - i : 커서 바로 앞부터 입력 모드로 전환
  - I : 커서가 있는 줄의 맨 앞부터 입력 모드로 전환
  - a : 커서 바로 뒤부터 입력 모드로 전환
  - A : 커서가 있는 줄 맨 뒤부터 입력 모드로 전환
  - o : 현재의 줄 아래에 새로운 줄을 만들어 입력 모드로 전환
  - O : 현재의 줄 위에 새로운 줄을 만들어 입력 모드로 전환

※ vim에서는 입력모드로 전환 후 backspace나 delete key를 사용 가능 (vi에서는 불가능)

vi 명령어 - 콜론모드

- 콜론모드로 전환
  - ‘Esc’key를 누르고 ‘:’(콜론)을 입력
- 저장
  - wq : 저장 후 종료 (or shift + zz)
  - w : 저장 (종료하지 않음)
  - w + 파일명 : 새이름으로 저장
- 종료
  - qw : 저장 후 종료
  - q! : 저장하지 않고 강제 종료
- 찾기
  - / + 텍스트 : 텍스트를 본문에서 검색
  - n : 다음 찾는 곳으로 이동

vi 명령어 - 명령모드

- **페이지 이동**
  - 행번호 + G : 행번호를 입력하고 Shift +g를 입력하면 해당 라인으로 이동
  - ctrl + F : 다음 화면으로 이동, Page Down (혹은 Page Down key)
  - ctrl + B : 이전 화면으로 이동, Page UP (혹은 Page Up key)
- **실행 취소**
  - u : 실행 취소
  - U : 전체 실행 취소
- **블록 선택**
  - v : 현재 위치부터 커서의 이동에 따라 영역을 블록 선택
  - V : v와 동일하나 라인 단위로만 선택 가능
  - (선택 후 선택한 영역만 삭제하거나 복사할 수 있음)
  - (c 언어 프로그래밍 시 블록 선택 후 “=”key를 눌러 indentation을 맞출 수 있다.)

vi 명령어 - 명령모드

- 삭제 (잘라내기)
  - dd : 한줄 삭제
  - 숫자 + dd : 현재 줄부터 숫자만큼의 줄을 삭제
  - d + 방향키 (아래,위) : 현재 줄과 아래(혹은 위)줄을 같이 지움
  - dw : 한 단어 삭제 (숫자 + dw로 숫자만큼의 단어 삭제)
  - x : 커서가 있는 문자 삭제 (Delete key)
  - X : 커서 앞문자 삭제 (Back Space key)
  - (삭제 명령어는 윈도우의 잘라내기와 동일. 붙여넣기로 다른 곳에 붙일 수 있음)
- 복사
  - yy : 현재 라인 복사
  - 숫자 + yy : 현재부터 아래로 숫자만큼의 라인 복사
  - yw : 현재 단어 복사
  - 숫자 + yw : 현재부터 뒤로 숫자만큼의 단어 복사
- 붙여넣기
  - p : 현재 커서 다음 줄에 붙여넣기
  - P : 현재 커서 이전 줄에 붙여넣기

.vimrc 설정법

1. 기본 자신의 홈 디렉토리로 이동 (cd)
2. ls –al을 쳐보면 .vimrc 는 기본으로 폴더에 들어있지 않음
3. vi .vimrc를 입력하여 .vimrc 생성

- .vimrc 설정
  - set shiftwidth=4 – 자동 들여쓰기를 할 때 4칸 들여 쓰도록 한다.
  - set number – 행 번호를 사용한다.
  - set nobackup – 백업파일을 생성하지 않는다.
  - set fileencoding=euc-kr – 실제로 파일을 저장할 때 사용되는 인코딩은 euc-kr
  - set background=light – 하이라이팅 옵션 lihgt or dark
  - set expandtab – 탭을 입력하면 공백문자로 변환하는 기능을 설정
  - set hlsearch – 검색어를 구문강조해주는 기능
  - set ignorecase – 검색할 때 대소문자 무시하도록 하는 것.

3-2. Shell 프로그래밍

- 개요
  - 쉘은 인터프리터 언어이다. 즉 컴파일을 필요로 하지 않는다.
  - 쉘 프로그래밍의 필요성
    - Unix 는 기본적으로 작고 간단한 역할을 수행하는 프로그램으로 이루어져 있다.
    - 어떠한 작업을 수행하기 위해선 기존의 Unix 프로그램들을 구조적으로 연결하여 조립할 필요가 있다.
    - 복잡한 작업을 위해 프로그램의 제어와 사용자와의 상호작용 등이 필요로 하고 이를 위해 명령어들을 스크립트로 작성하여 실행하도록 한다.

Shell Programming - 문법

- 변수
  - 선언과 할당
    - Bourene shell 에서는 변수를 선언하지 않고 사용할 수 있다.
    - 변수의 이름은 암묵적으로 대문자를 많이 사용한다.

```
$ name=sogang
$ echo $name
sogang
$ name=computer
$ echo $name
computer
```

Shell Programming - 문법

- 변수
  - 선언과 할당

```
$ name=sogang computer
bash: computer: command not found
$ name="sogang computer"
$ echo $name
sogang computer
$ ux=UNIX
$ echo ${ux}tm
UNIXtm
$echo "$ux"tm
UNIXtm
$ echo $uxtm

$ set -u
$ echo $uxtm
bash: uxtm: unbound variable
$ set +u
$ echo $uxtm

$
```

```
$ flower=tulip
$ readonly flower
$ flower=rose
bash: flower: readonly variable
```

Shell Programming - 문법

- 변수
  - Exporting shell variables
    - 변수는 선언된 쉘 내에서만 사용된다. 하지만 export 명령을 사용하면 변수를 다른 곳에서도 사용할 수 있게 된다.

```
$ cat foodlike
echo $pn butter is yummy

$ pn=peanut
$ ./foodlike
butter is yummy
$ export pn
$ ./foodlike
peanut butter is yummy
```

Shell Programming - 문법

변수

Automatic shell variables

변수	설명
$?	직전에 실행한 명령에 대한 exit value를 나타낸다. 0값이 return 되면 직전의 명령이 정상적으로 수행되었음을 나타내고, 나머지 값이 return 되면 직전의 명령이 정상적으로 수행되지 못했음을 나타낸다.
$$	shell의 process id를 나타낸다.
$!	shell이 실행시킨 background process의 number를 나타낸다.
$#	script의 argument의 개수를 나타낸다. $1은 첫 번째, $2는 두 번째... $9는 9번째 argument를 나타낸다.
$*	argument의 수를 나타낸다. $과 다른 점은 $#은 9개까지 밖에 나타내지 못하지만, $*는 입력된 모든 argument를 나타낼 수 있다.
$@	$*과 비슷한 기능을 수행한다. 단, "$*"의 경우는 "$1 $2 ... $n"을 나타내지만 "%@"은 "$1" "$2" ... "$n"을 의미한다.

Shell Programming - 문법

- 변수
  - Automatic shell variables

```
$ cp abc cdf
cp: cannot stat `abc': No such file or directory
$ echo $?
1
$ echo $$
24085
$ ps
  PID TTY          TIME CMD
24085 pts/0    00:00:00 bash
24129 pts/0    00:00:00 ps
$ echo $?
0
$
```

Shell Programming - 문법

- 변수
  - Automatic shell variables

```
$ cat tt
echo $# $2 $3 $4 $5 $6 $7 $8 $9 $10 $11 $12

$ ./tt a b c d e f g h i j k
11 a b c d e f g h i a0 a1 a2
$
```

```
$ cat tt1
echo $# $*

$ ./tt1 a b c d e f g h i j k
11 a b c d e f g h i j k
$ cat tt2
echo $# $@

$ ./tt2 a b c d e f g h i j k
11 a b c d e f g h i j k
$
```

Shell Programming - 문법

■ 변수

♦ Standard shell variable

변 수	설 명
$PATH	명령어 실행을 위해서 검색하는 디렉토리 경로를 나타낸다. PATH 상에 나타나 있는 디렉토리 순서대로 명령어를 검색하여 먼저 찾게 되는 디렉토리의 명령어를 실행한다. 가령 ls 명령어가 /bin과 /usr/bin에 각각 존재한다고 하자. $PATH=/bin:/usr/bin:/usr/sbin 으로 설정되어 있다면 사용자가 쉘에서 ls 명령을 쳤을 때, /bin/ls 명령을 실행하게 된다.
$CDPATH	cd 명령을 사용할 때의 기준점. 만약 $CDPATH를 /home/abc로 설정한 경우 cd ttl을 하면, 현재 디렉토리가 /home/abc/ttl로 옮겨지게 된다.
$LOGNAME	로그인 이름(계정명)
$IFS	internal field seperator, 기본적으로 공백문자(스페이스, 탭)로 설정
$SHELL	로그인 하면 실행되는 기본 쉘의 이름
$PS1	로그인 하였을 때 나타나는 프롬프트 모양
$TERM	접속한 terminal
$HOME	user의 홈 디렉토리

Shell Programming - 문법

- 변수
  - Quoting special characters
    - Backslash(₩) – 한 문자에 대하여 meta character 에 관계없이 있는 그대로 하나의 문자로 인식한다.
    - Single quote(‘ ‘) – single quote 내에 있는 문자열을 있는 그대로 처리한다.
    - Double quote(“ “) – double quote 내에 있는 변수를 변수 값으로 치환하여 처리한다.

```
$ echo $HOME
/home/venus
$ echo '$HOME'
$HOME
$ echo \$home
$HOME
$ echo "$home"
/home/venus
$
```

Shell Programming - 문법

- 조건문
  - if

```
if condition
then
    statements
[elif condition
    then statements...]
[else
    statements]
fi
```

```
if [ "$FILE1" = "$FILE2" ]
then
...
fi
```

```
if test "$FILE1" = "$FILE2"
then
...
fi
```

Shell Programming - 문법

- 조건문
  - if (Cont')

```
$ cat if1
if [ $# -eq 4 ]
then
        echo $4 $3 $2 $1
else
        echo Usage : $0 arg1 arg2 arg3 arg4
fi

$ ./if1
Usage: ./if1 arg1 arg2 arg3 arg4
$ ./if1 1 2 3 4
4 3 2 1
$
```

Shell Programming - 문법

- 반복문
  - for

```
for name [in list]
do
    statements that can use $name...
done
```

```
$ cat for1
IFS=:

for dir in $PATH
do
    ls -ld $dir
done
$ echo $PATH
/bin:/usr/bin:/sbin:/usr/sbin:/usr/local/bin
$ ./for1
drwxr-xr-x    2 root     root         4096 Apr 11  2003 /bin
drwxr-xr-x    2 root     root        53248 Dec 16 21:56 /usr/bin
drwxr-xr-x    2 root     root         8192 Apr 15  2003 /sbin
drwxr-xr-x    2 root     root        12288 Apr 15  2003 /usr/sbin
drwxr-xr-x    2 root     root         4096 Feb  7  1996 /usr/local/bin
$
```

Shell Programming - 문법

- 반복문
  - While and until

```
while condition
do
     statements...
done
```

```
until condition
do
     statments...
done
```

```
$ cat square
# squares - prints the square of integers in succession
int=1
while [ $int -lt 5 ]
do
     sq=`expr $int \* $int`
     echo $sq
     int=`expr $int + 1`
done
echo "job complete"
$ ./square
1
4
9
16
job complete
$
```

Shell Programming - 문법

선택문

case

```
case expression in
     pattern1 )
          statements ;;
     pattern2 )
          statements ;;
     ...
     * )
          statements ;;
esac
```

```
$ cat hi
echo "Is it morning? Please answer yes or no"
read timeofday
case "$timeofday" in
     yes | y | Yes | YES ) echo "Good Morning" ;;
     n* | N* ) echo "Good Afternoon" ;;
     * ) echo "Sorry, answer not recognized" ;;
esac
$ ./hi
Is it morning? Please answer yes or no
yes
Good Morning
$
```

Shell Programming - 문법

- 함수

```
name()
{
     statements
}
```

```
$ cat yesno
yes_or_no () {
    echo "Is your name $* ?"
    while true
    do
        echo -n "Enter yes or no:"
        read x
        case "$x" in
            y | yes ) return 0;;
            n | no ) return 1;;
            * ) echo "Answer yes or no"
        esac
    done
}
echo "Original parameters are $*"

if yes_or_no "$1"
then
    echo "Hi $1, nice name"
else
    echo "Never mind"
fi
exit 0
$ ./yesno lazy lune
Original parameters are lazy lune
Is your name lazy ?
Enter yes or no: yes
Hi lazy, nice name
$
```

1주차 실습

- **온라인 전화/주소록**
  - 전화번호와 주소록이 기록되어 있는 데이터 파일이 있다. 사람의 이름, 주소, 전화번호의 일부분이 입력으로 주어지면 데이터 파일에서 입력과 맞는 데이터를 검색하여 포맷에 맞게 출력한다. 스크립트 파일명은 “phone”으로 한다.

- **입출력 형식**
  - 데이터 파일 형식: 각 줄마다 하나의 레코드가 “이름|주소|전화번호”로 입력된다.
  - 입력형식: 쉘프롬프트 상에서 한 개 이상의 인자(argument)로 주어진다.
  - 출력형식: 검색된 레코드를 아래 예와 같이 출력한다. 만약 검색된 레코드가 없다면 아무것도 출력하지 않는다.

1주차 실습

- 입출력 형식 (Cont')

데이터 파일예

```
홍길동|서울시 마포구 신수동 서강대학교 AS관 301호|02-705-2665
Andrew|경기도 의정부시 호원동 23-12번지|031-827-7842
```

입력예

```
./phone  홍길동  신수동
```

출력예

```
-------→
name: 홍길동
address: 서울시 마포구 신수동 서강대학교 AS관 301호
phone: 02-705-2665
←-------
```

1주차 실습

- **문제 해결 방법**
  - 에러처리: 반드시 하나이상의 인자를 필요로 하며, 해당 인자가 주어지지 않은 경우 다음과 같이 사용 예(usage)를 출력해 준다.

입력예

```
./phone
```

출력예

```
Usage: phone searchfor [… searchfor]
(You didn't tell me what you want to search for.)
```

  - 입력 인자 변경: egrep 을 사용하기 위해 모든 인자를 다 붙여서 하나의 문자열로 만들어야 한다.

```
egrep –i "(arg1|arg2|…|argn)" datafile
과 같이 사용되어야 하므로
./phone arg1 arg2 … argn
에서 (arg1|arg2|…|argn) 으로 고쳐줘야 한다
```

  - 해당 데이터 추출 및 출력: egrep을 통해 원하는 데이터를 추출하고 포맷에 맞게 출력해야 한다. 포맷을 맞추는 것은 awk 프로그램을 이용할 수 있다. Field의 구분자로 "|"를 사용해야 하며 이것은 BEGIN {} 내에 FS="|"; 를 넣어주면 된다.
  - Stripping off the head of a pathname: basename (e.g. name="`basename $0`")

awk

- Programming language designed to make many common information retrieval and text manipulation tasks easy to state and to perform

- Basic operation: scan a set of input lines in order, searching for lines which match any of a set of patterns which the user has specified

  예 **{print $1, $2, $3}**

- Usage
  - awk 'program' [datafile]
  - awk –f program [datafile]

1주차 실습

- 프로그래밍 문제 해결