

IISEE lecture for Global Seismological Observation Course

Introduction to UNIX

Lecturer
Yushiro Fujii

(This lecture was originally written by Y. Yagi
and modified by Y. Fujii)

Why UNIX?

- Weakness
 - UNIX is difficult to use; it has cryptic commands and its interface is non-intuitive.
- Advantage
 - UNIX is stable, flexible, and powerful for **multiple users** and **multitasking**. Many packages and libraries for seismology and geophysics have been developed in the UNIX system (e.g., *win*, *sac*, *GMT*, and *waveform inversion program*).

In this lecture, you will learn how to make use of UNIX's command and graphical tools, and familiarize yourself with the commands you can productively use.

Preparation for Lecture

Go to home directory

```
$ cd
```

Make "UNIX" directory

```
$ mkdir UNIX (We use this directory for today's lecture.)
```

Open editor

```
$ cd UNIX
```

```
$ nedit &
```

Make a simple FORTRAN code like

```
(7 spaces) real a, b, c
```

```
    a=1.0
```

```
    b=2.0
```

```
    c=a+b
```

```
    write(6,*) 'c=',c
```

```
    stop
```

```
    end
```

and save as "program.f"

Confirm the program can be compiled and run.

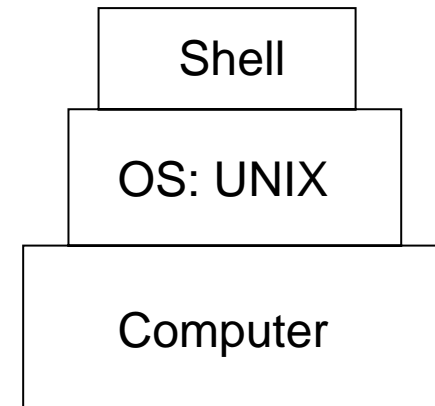
```
$ g77 program.f
```

```
$ ./a.out
```

```
    c= 3.
```

UNIX Shell

- Once you login, you are working with a program called a shell.
 - default shell for Cygwin, Linux (earth2, ocean): **bash**
 - default shell for SUN: **csh**
- The differences between the shells are slight. In this lecture, we will program the **csh** script.
- Shell prompt:
Shell prompts usually contain \$ or %.



Syntax of UNIX Commands

- UNIX commands are simple one-word entries such as the “ls” command. They can be more complex using various options. The general format for a UNIX command is

*\$ **command** (options) (file-names)*

(e.g., *\$ g77 -o program program.f*: you can find the program.exe file in the working directory).

Options

- Options modify the way in which a command works.
 - Syntax
 - Options are often single letters prefixed with a dash (-).
 - Multiple options in a single command line can be set off individually (-l -a) (in some cases, you can combine them after a single dash (like -la)).

```
$ ls -l -a
```

```
$ ls -la
```

Unresponsive Terminal

- In case your terminal does not respond to a command (hung or frozen), please type (Ctrl)-c. The process is killed, and you will get a new command prompt.

UNIX File System

- Like other systems, a file is the unit of storage in UNIX. Files are organized into directories (folders). A directory is a special kind of file where the system stores information about other files.
- A directory is a place where files are said to be contained, and you are said to be working inside a directory.
 - If you want to check the working directory (Where am I?),

Type

\$ *pwd*

P.56, 57, Unix in 24 Hours

Cygwin;

/cygdrive/c/Documents and Settings/(user ID)/My Documents

or

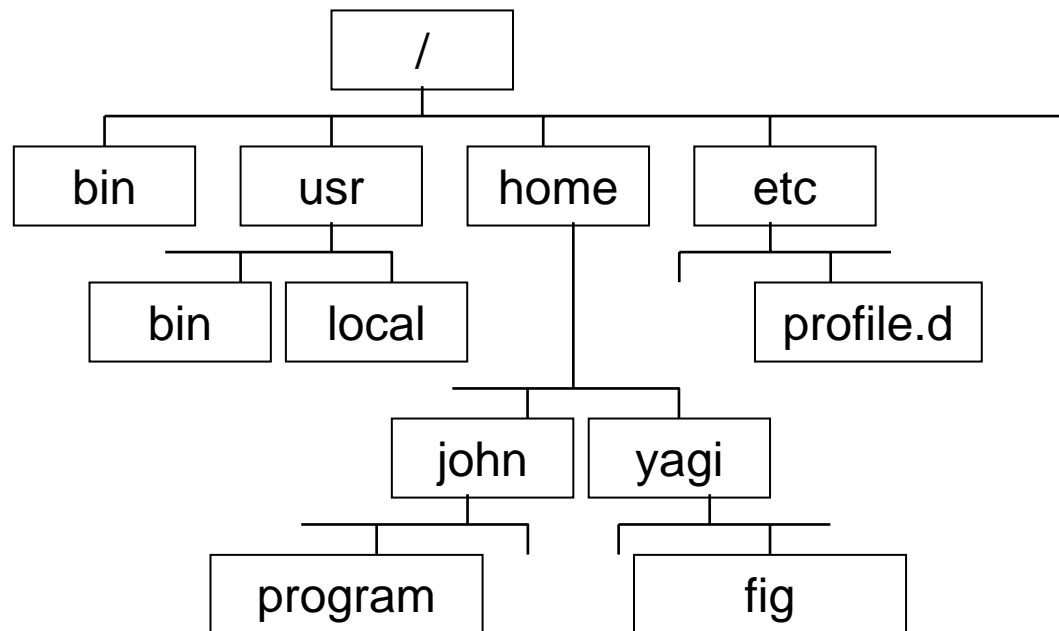
/home/(user ID)

Linux (earth2 or ocean):

/home/(user ID)

Directory Tree

- All directories in a UNIX system are organized into a hierarchical structure, which is similar to a family tree. The parent directory is the root directory and is written as a forward slash (/).



Example of a directory tree.

In this case, the user directories are “/home/yagi” or “/home/john”.

The *directory* named “bin” contain some programs.

In some cases, you cannot change the file without accessing your home directory.

Absolute and Relative Paths

Absolute path

An absolute pathname signifies the path of the directories you need to travel to get from the root to the required directory or file. In the path name, put slashes (/) between the directory names.

e.g., */home/yagi/tmp.ps*

- the root is the first “/”
- the directory “home” is a subdirectory of root “/”
- the directory “yagi” is a subdirectory of “home”
- the file of tmp.ps is located in the directory “/home/yagi/”

Type

\$ pwd

/home/yagi

You can get the absolute pathname of the working directory.

Absolute and Relative Paths

Relative path

A *relative pathname* gives the location relative to your working directory. Unless you use an *absolute pathname* (starting with a slash), UNIX assumes that you are using a relative pathname.

e.g.,

go to home directory

```
$ cd
```

P.56, 57, Unix in 24 Hours

go to "UNIX" directory

```
$ cd UNIX
```

Make "work" directory

```
$ mkdir work
```

P.110 -112

go to "work" directory

```
$ cd work
```

You can go up the tree by using the shorthand ".." for the parent directory. Evidently, you can also go down the tree by using the subdirectory name.

go up the tree

```
$ cd ..
```

go up the tree and go down the directory

```
$ cd ../UNIX/work
```

Listing Files

The “**ls**” command lists the entries in the directory tree. The syntax is

```
$ ls (options) (directory-and-filename)
```

Type

```
$ ls
```

P.50 - 54, 62 - 77, 86 - 93, Unix in 24 Hours

You can get the filenames in the working directory.

To get more information about each file, type

```
$ ls -al
```

```
total 104
```

```
d rwxr-xr-x 6 yagi none 0 Sep 19 10:28 .
d rwxr-xr-x 4 yagi none 0 Sep 19 10:28 ..
-rwxr-xr-x 1 yagi none 1999 Sep 23 15:40 a.out
-rw-r--r-- 1 yagi none 256 Sep 23 15:40 program.f
d rw-r-xr-x 1 yagi none 0 Sep 23 15:40 work
```

type access modes # of links owner group size modification date and time name

Listing Files

Type

Tells whether the file is a directory “d” or a plain file “-”

Access modes

Specifies three types of users (yourself “u”, your group “g”, and all others “o”) who are allowed to read “r”, write “w”, or execute “x” files.

If you want to edit the access modes, the “**chmod**” command is useful.

(e.g., type

```
$ chmod u-x (file-name)
```

You cannot execute the “*file-name*”

```
$ chmod u+x (file-name)
```

You can again execute “*file-name*”

P.94 - 102, Unix in 24 Hours

Size (in bytes)

Size of the file

Modification date

Date when the file was last modified.

Listing Files

If you give the pathname to a directory, “ls” will list the directory, but it will not change your working directory.

```
$ ls /usr/local/
```

```
bin doc etc include lib man share
```

-F and --color options are useful for detecting the file type.

```
$ ls -F
```

```
a.out* program.f work/
```

“/” at the end of each directory name. Files with an execute status “x”, like programs, are marked with “*”.

```
$ ls --color
```

```
a.out program.f work
```

A green filename signifies the execute status, and a blue filename signifies the directory name.

Looking Inside Files

The “**cat**” command lets you move forward in the files by any number of pages or lines. The syntax is

\$ cat (files)

P.134 - 136, Unix in 24 Hours

.....
.....*Inside file*.....
.....

“cat” works for short files containing characters that can be displayed on a single screen or less. You cannot go back to view the previous screens.

Looking Inside File

The “**less**” and “**more**” commands let you move forward in the files by any number of pages or lines. The syntax is

\$ more (files)

.....

(file-name)

P.136 - 140, Unix in 24 Hours

.....

.....Inside file.....

.....

(file-name)

.....

.....Inside file.....

-----More----(50%)

The prompt says that you are 50% of your way through the file.

Looking Inside Files

If you type “h” at the “more” prompt, you can get the useful “more” commands on your system.

Typical useful “more” commands are as follows:

Command	Description
(SPACE)	Display next page
z	Display next page
(ENTER)	Display next page
d or (Ctrl)-D	Scroll k lines (k depends on your system)
b or (Ctrl)-B	Skip backward k screens of text
q	End reading file

Managing Your Files

The previous lecture dealt with formulating FORTRAN programs; for separating the programming files with the others files, you can create a programming directory using the “**mkdir**” command.

```
$ mkdir (new-directory-names)
```

P.110 - 112, Unix in 24 Hours

Before editing a file, you can save a copy using the “**cp**” command.

```
$ cp (old-file-name) (new-file-name)
```

```
e.g., $ cp program.f program2.f
```

```
      $ cp program2.f program3.f
```

If you want to put a copy of the file into a subdirectory, use

```
$ cp (file-name) (subdirectory-name)
```

```
e.g., $ cp program.f work
```

P.112 -114

Managing Your Files

You can change the filename using the “**mv**” command.

Type

```
$ mv (old-file-name) (new-file-name)
```

If you want to put a file into a subdirectory, use

```
$ mv (file-name) (subdirectory-name)
```

If you do not want to overwrite any old files, use “-i” option for safety. The syntax is

```
$ mv -i (old-file-name) (new-file-name)
```

P.114 - 116, Unix in 24 Hours

Managing Your Files

You can remove files using the “**rm**” command. Type

```
$ rm (file-name)
```

Please use “-i” option for safety.

```
$ rm -i (file-name)
```

P.117 - 121, Unix in 24 Hours

e.g.,

```
$ rm -i program2.f
```

(press “n” for not deleting a file)

or use alias (like a shortcut in Windows)

```
$ alias rm='rm -i'
```

```
$ alias mv='mv -i'
```

P.294 - 296

```
$ alias cp='cp -i'
```

You can remove directories using the “**rmdir**” command. Type

```
$ rmdir (directory-name)
```

The directory has to be empty before it is deleted.

P.116 -117

Changing your Environment

On a UNIX system, you can change the environment. When you log into a UNIX computer, csh refers to the “/home/(user-name)/.cshrc” file that contains information about the environment, and bash refers to the “/home/(user-name)/.bashrc” file. For changing the environment, type

```
$ pwd  
/home/(user-name)  
$ cp /etc/bash.bashrc .bashrc  
$ nedit .bashrc &
```

In the nedit window, insert the following lines:

```
alias rm='rm -i'  
alias mv='mv -i'  
alias cp='cp -i'
```

```
$ source .bashrc
```

refer to new environment file

```
% rehash
```

refresh environment (for csh)

Useful Tip 1

If you mistyped on the command line, you do not need to type the same command again. Push the up arrow key and modify the command.

e.g.,

```
$ cpy file-name1 file-name2 (wrong command for "cp")
```

```
$ (push up-arrow key)
```

```
$ cpy file-name1 file-name2 (edit command using left- or  
right-arrow key)
```

```
$ cp file-name1 file-name2
```

This tip is also useful in case you want to repeat the same command.

Useful Tip 2

Bash or tcsh has a function to interpolate a file or command name. You do not need to type all the characters of a long filename.

Push the tab key after you type a few beginning characters of a file or command name.

e.g.,

There is a file called program.f in the “work” directory.

```
$ cd work
```

```
$ less prog (push the Tab key)
```

```
$ less program.f (filename is interpolated!)
```

Managing Your Files

You can create a backup file using the “**tar**” command. The syntax is

Create a backup file from files or directory

```
$ tar -cvf (backup-file-name) (files or directory)
```

Extract all the files from backup file

```
$ tar -xvf (backup-file-name)
```

P.376 - 381, Unix in 24 Hours

Exercise

You will create “tmp1” and “tmp2” directories in your “UNIX” directory; copy the FORTRAN programs into “tmp1”; go to “tmp1”; backup the Fortran programs in “tmp1”; copy the backup files into “tmp2”; and extract all the files in “tmp2”.

Exercise

Go to home directory; create “tmp1” and “tmp2” directories

```
$ cd UNIX
```

```
$ mkdir tmp1 tmp2
```

Copy the FORTRAN programs into “tmp1”; move “tmp1”

```
$ cp *.f tmp1/
```

```
$ cd tmp1
```

Backup the Fortran programs in “tmp1”; copy backup file into “tmp2”

```
$ tar -cvf program.tar *.f
```

```
$ cp program.tar ../tmp2
```

Go to “tmp2”; extract all files in “tmp2”

```
$ cd ../tmp2
```

```
$ tar -xvf program.tar
```

- “*” (Asterisk) is replaced by any character in a filename (e.g., *.f representing all the FORTRAN files in the working directory)
- “?” (question) represents any single character in a filename

Redirecting I_nput/O_utput

Input

The shell takes whatever you type on your keyboard as the input to the command (after you press (*Return*) to start the command)

Input redirection

You can use a given file as the input to a command and/or a program that does not accept filenames by using the “<” operator. The syntax is

\$ command < (input-file)

P.143 - 146, Unix in 24 Hours

Redirecting Input/Output

Sample program to use redirecting input (<)

Make a simple FORTRAN code and save it as "program_in.f"

```
(7 spaces) real a, b, c
      write(6,*) 'a=?'
      read(5,*) a
      write(6,*) 'b=?'
      read(5,*) b
      c=a+b
      write(6,*) 'c=',c
      stop
      end
```

Make an input file and save it as "input.dat"

```
1.0
2.0
```

Confirm that the program can be compiled and run.

```
$ g77 program_in.f
$ ./a.exe < input.dat
a=?
1.0
b=?
2.0
c=3.
```

Redirecting Input/Output

Output

As the command runs, the results are usually displayed on your terminal. The terminal is the command's standard output.

Output redirection

You can write the results of a command or/and program to a named file using the ">" operator. The syntax is

```
$ command > (output-filename)
```

e.g.,

```
$ ls -al > list.dat
```

```
$ cat list.dat
```

```
total 104
```

```
d rwxr-xr-x 6 yagi none 0 Sep 19 10:28 .
```

```
.....
```

```
$ echo "Ohayou-gozaimasu" > greeting.dat
```

```
$ cat greeting.dat
```

```
Ohayou-gozaimasu
```

echo: display a line of text

If you created a *new-file* consisting of *file1* followed by *file2*, type

```
$ cat (file1) (file2) > (new-file)
```

or

```
$ cat (file1) > (new-file)
```

```
$ cat (file1) >> (file2)
```

(if you want to add the contents of *file1* to the end of *file2*)

Comparing Text Files

“diff” command is useful to check the differences between text file A and text file B.

e.g.,

```
$ diff program.f program_in.f
```

```
2,3c2,5
```

```
<  a=1.0          ← program.f  
<  b=2.0
```

```
---
```

```
>  write(6,*) 'a=?'  
>  read(5,*) a    ← program_in.f  
>  write(6,*) 'b=?'  
>  read(5,*) b
```

Pipes and Filters

You can connect two commands together so that the output from one command (program) becomes the input to the next command (program) using the “|” (vertical bar) operator. Any two commands (programs) can form a pipe as long as the first program writes to the standard output and the second program reads from the standard input. The syntax is

\$ command (options) (filename) | command (options)

If you use a filtering program, the pipe is very useful.

Filtering program: grep, sort, more, less

P.146 - 149, Unix in 24 Hours

Pipes and Filters

The “**grep**” program searches for files with lines that have a certain pattern. The syntax is

```
$ grep (pattern) (files)
```

e.g.,

```
$ grep real *.f
```

```
program.f: real a, b, c
```

```
program2.f: real a, b, c
```

```
program3.f: real a, b, c
```

```
filename: lines that have the specified pattern
```

P.153 - 156, Unix in 24 Hours

e.g.,

```
$ ls -al | grep Sep
```

```
drwxr-xr-x 6 yagi none 0 Sep 23 10:28 .
```

```
drwxr-xr-x 4 yagi none 0 Sep 19 14:07 ..
```

```
-rw-r--r-- 1 yagi none 595 Sep 23 10:47 .bash_history
```

First, our example runs “**ls -al**” to list your directory; the standard output of “**ls -al**” is piped to **grep**, which only outputs the lines that contain the string “Sep”.

Pipes and Filters

The “**sort**” program arranges the lines of text alphabetically or numerically. The syntax is

\$ grep (options) (file)

option Descriptions

-n Sort numerically and ignore blanks and tabs.

-r Reverse the order of sort.

+x Ignore the first x fields when sorting.

e.g.,

\$ ls -al | grep “Oct” | sort +4 -n -r

P.147 - 149, Unix in 24 Hours

-rwxr-xr-x 1 fujii none 80493 Oct 20 19:19 program.exe

-rwxr-xr-x 1 fujii none 80493 Oct 20 19:19 a.exe

-rw-r--r-- 1 fujii none 100 Oct 20 19:19 program.f

drwxr-xr-x+ 3 fujii none 0 Oct 20 19:19 .

drwxr-xr-x+ 2 fujii none 0 Oct 20 19:17 work

dr-x-----+ 40 fujii none 0 Oct 20 19:17 ..

Multitasking

Running a command as a background process

To run a command in the background, add the “&” character at the end of the command line. The syntax is

```
$ program < (input-file) > (output-file) &
```

```
[1] 12222
```

The process ID for this command is 12222. To check on a process, the “**ps**” command is useful. The syntax is

```
$ ps (options)
```

P.273, 274, Unix in 24 Hours

To cancel a process, the “kill” command is useful. The syntax is

```
$ kill (ID)
```

P.312, 317 - 319

Multitasking

e.g.,

```
$ nedit &
```

```
[1] 3292
```

```
$ ps
```

<i>PID</i>	<i>PPID</i>	<i>PGID</i>	<i>WINPID</i>	<i>TTY</i>	<i>UID</i>	<i>STIME</i>	<i>COMMAND</i>
2380	1	2380	2380	con	1006	18:05:18	/usr/bin/bash
432	2380	432	2436	con	1006	19:17:56	/usr/bin/sh
3764	432	432	168	con	1006	19:17:56	/usr/X11R6/bin/xinit
3544	3764	3544	240	con	1006	19:17:56	/usr/X11R6/bin/XWin
2808	3764	2808	1672	con	1006	19:17:59	/usr/bin/xterm
2136	2808	2136	3904	0	1006	19:18:00	/usr/bin/bash
3292	2136	3292	1960	0	1006	19:41:49	/usr/X11R6/bin/nedit
984	2136	984	3040	0	1006	19:41:50	/usr/bin/ps

```
$ kill 3292
```

```
[1]+ Terminated
```

```
nedit
```

You can enter an entire sequence of commands separated using semicolons (;).

The syntax is

```
$ command1 ; command2
```

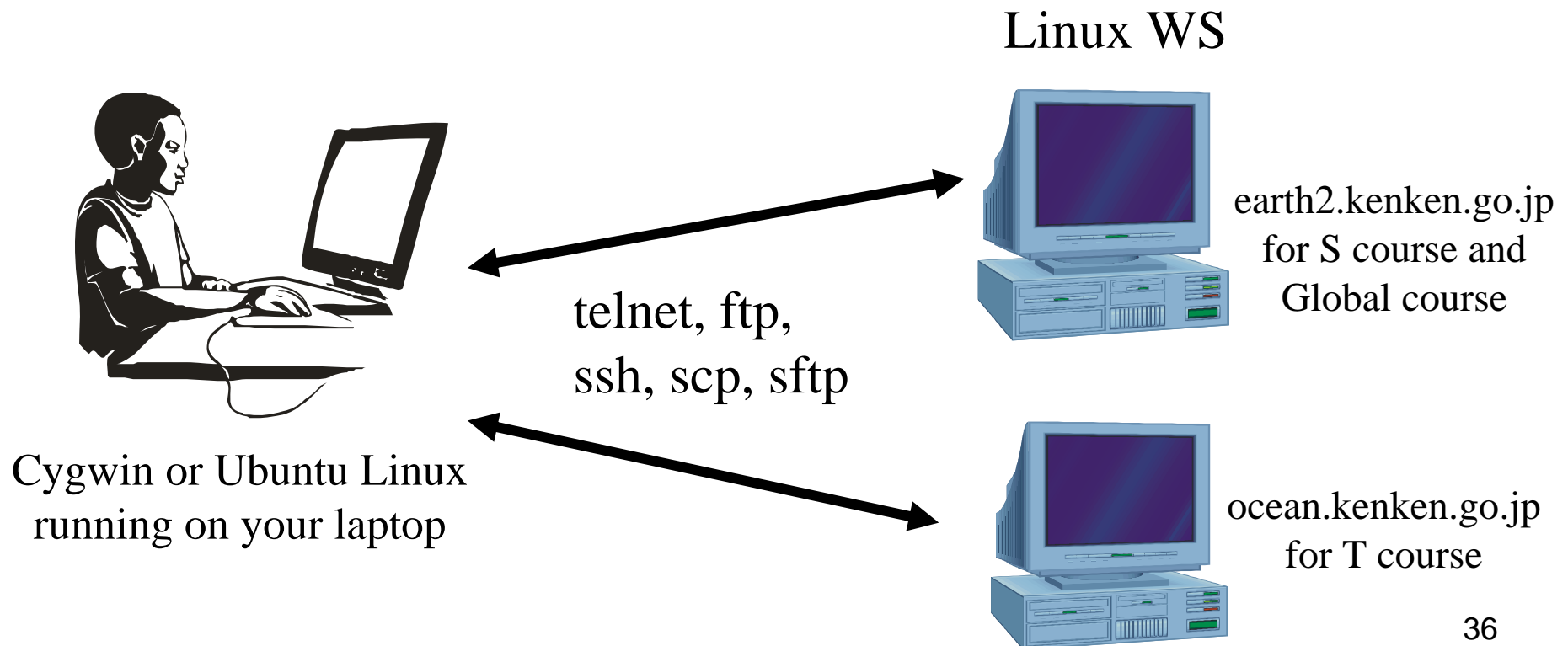
If you want to run a command line in the background, type

```
$ command &
```

```
$ (command1 ; command2) &
```

Multuser

UNIX offers a “multiuser system” environment.
We can use the same UNIX computer (Linux WS)
simultaneously from any computer (Windows PC).



Connecting to a UNIX Computer

- When you might have a workstation on your desk, but you need to work from the main computer at another location, remote login programs can be useful and powerful.
- We use “**ssh**” (secure shell) or “telnet” to log into another computer.
 - The syntax for most remote login programs is
\$ (program-name) (remote-hostname)
 - Type
\$ ssh earth2.kenken.go.jp -l (user ID)
or
\$ ssh (user ID)@earth2.kenken.go.jp

Connecting to a UNIX Computer

IP address

Almost all machines on a LAN have an Internet Protocol (IP) address and host name, which are unique to each machine. IP address is a 32-bit number. The syntax is ????.????.????.??? (four sets of numbers separated by ".").

Domain host name

Domains and host names are usually used instead of IP addresses.

e.g.,

earth2.kenken.go.jp



host name



domain name

Connecting to a UNIX Computer

You can confirm whether another computer is alive or not using the “**ping**” command. The syntax is

```
$ ping (host-name or IP-address)
```

e.g.,

```
$ ping earth2.kenken.go.jp
```

```
Pinging earth2.kenken.go.jp [172.16.21.40] with 32 bytes of data:
```

```
Reply from 172.16.21.40 : bytes=32 time=4ms TTL=253
```

```
Reply from 172.16.21.40 : bytes=32 time=4ms TTL=253
```

```
Reply from 172.16.21.40 : bytes=32 time=4ms TTL=253
```

(type (Ctrl)-c to stop)

Connecting to a UNIX Computer

```
$ ssh earth2.kenken.go.jp -l (your ID)
```

or

```
$ ssh (your ID)@earth2.kenken.go.jp
```

```
Connected to earth2.kenken.go.jp (172.16.21.40).
```

```
(user ID)@earth2's password:
```

```
[(user ID)@earth2 (user ID)]$
```

e.g.,

```
[fujii@earth2 fujii]$:
```

```
[fujii@earth2 fujii]$: ls -a
```

```
(file names in earth2) .....
```

```
[fujii@earth2 fujii]$: mkdir UNIX
```

```
[fujii@earth2 fujii]$: exit
```

```
Connection closed by foreign host.
```

You can edit, compile, and run in “earth2” or “ocean” using ssh through the network.

Changing your Password

On a UNIX system, everyone can find your username. If you type “who”, you can get all the usernames. For your own safety, please change your password using the “**passwd**” command. Type

```
$ passwd
```

```
passwd: changing password for yagi
```

```
Enter login(NIS) password: *****
```

```
New password: *****
```

- In general, a password should be something that is easy for you to remember but difficult for other people to guess. Please use a combination of alphabets, numbers, and symbols.

Copying Files between Two Computers (1)

The command “**sftp**” or “**ftp**” (file transfer protocol) is a flexible way to copy files between two computers. The syntax is

```
$ sftp (your ID)@(host-name)
```

P.424 - 429, Unix in 24 Hours

Type

```
$ sftp fujii@earth2.kenken.go.jp
```

```
Connecting to earth2.kenken.go.jp...
```

```
fujii@earth2.kenken.go.jp's password:
```

```
password: *****
```

```
sftp>
```

```
sftp> type “quit”, “bye”, or “exit” to quit
```

```
$
```

Copying Files between Two Computers (1)

Some **sftp** commands

command	Description
<i>sftp> put (file-name)</i>	Copies the file from your local computer to the remote computer.
<i>sftp> get (file-name)</i>	Copies the file from the remote computer to your local computer.
<i>sftp> mput (file-names)</i>	Copies the files from your local computer to the remote computer.
<i>sftp> mget (file-name)</i>	Copies the file from the remote computer to your local computer.
<i>sftp> cd (path-name)</i>	Changes the working directory on the remote machine to <i>path-name</i> .
<i>sftp > ls</i>	Lists the remote directory.
<i>sftp > !ls</i>	Lists the local directory

Some **ftp** commands

<i>ftp> bin</i>	Tells the ftp to copy the following file without translation.
<i>ftp> asc</i>	Transfers plain text file data if needed.

Copying Files between Two Computers (2)

The command “**scp**” is a simple way to copy files between two computers.

You have to know in advance where the file you want is located in the remote computer.

The syntax is

```
$ scp (your ID)@(host-name):(file name with remote directory path)
(local directory)
```

Type

```
$ scp fujii@ocean:/home/fujii/work/program.f .
```

If you want to transfer the entire directory to the current directory, type

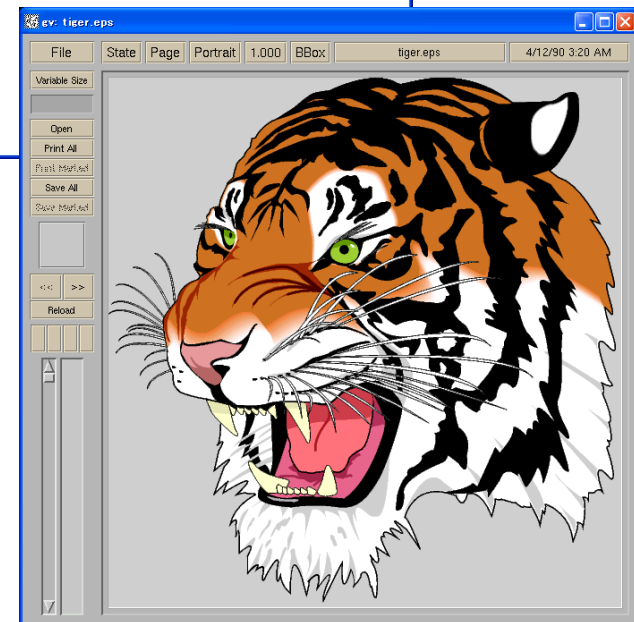
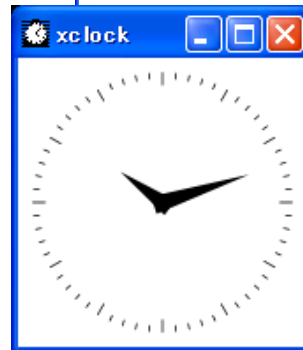
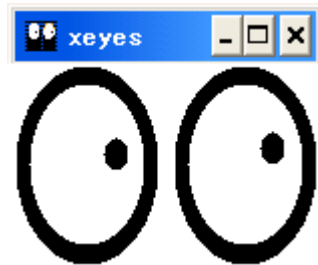
```
$ scp -r fujii@ocean:/home/fujii/work .
```

If you want to preserve the time stamp of the file, type

```
$ scp -p fujii@ocean:/home/fujii/work/program.f .
```

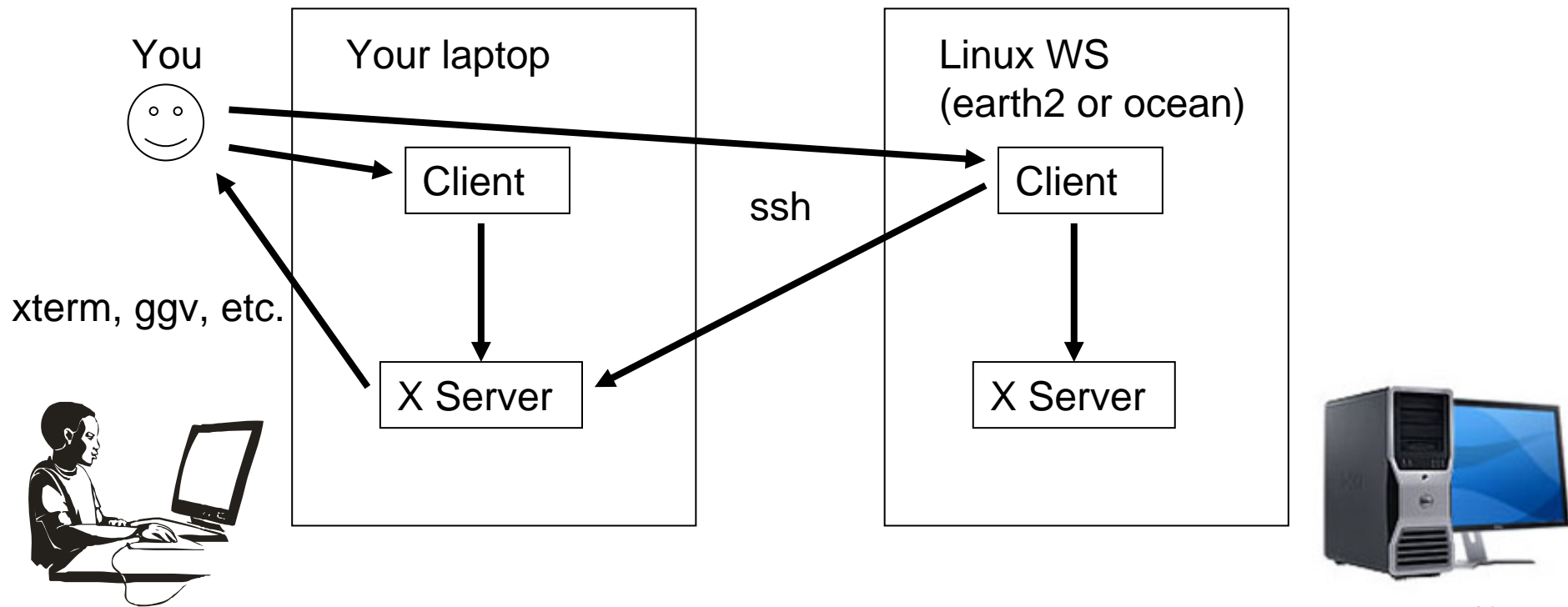
Connecting to a UNIX Computer Using X Window System

If you want to plot a graphic file into another computer, you can plot it using the “X Window System”.



X Window System

X Window System is a client-server system. The “client” asks the “server” to offer graphics, and the “server” can yield a graphical environment.



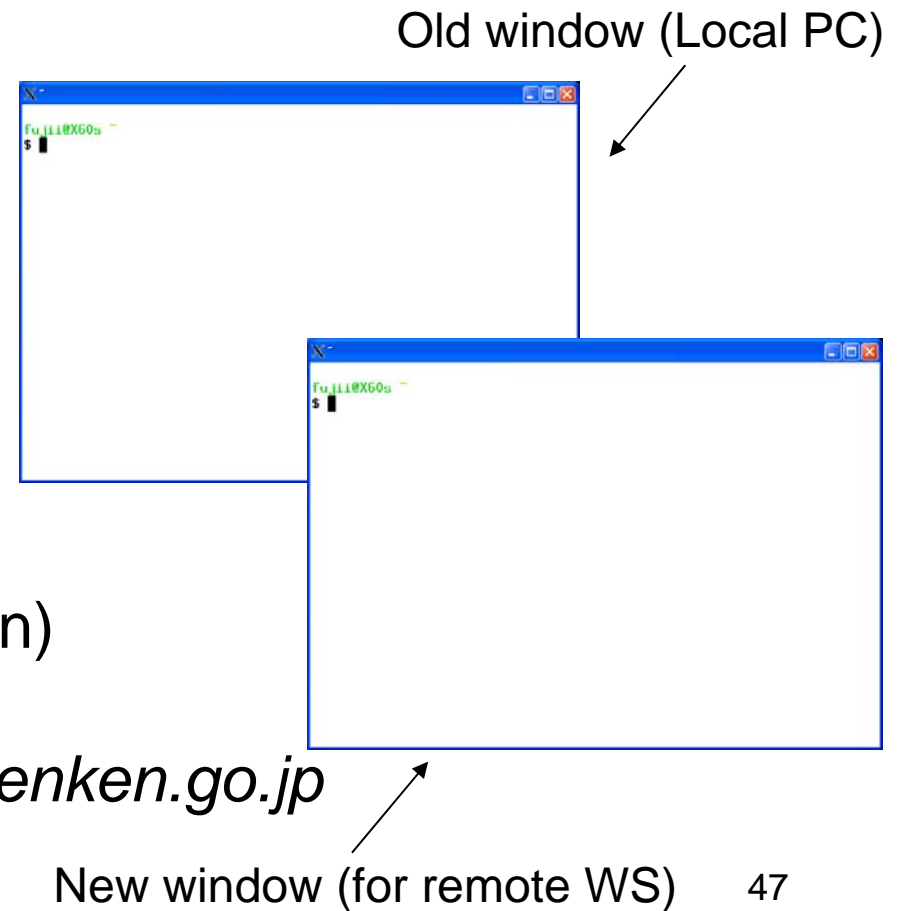
Connecting to a UNIX Computer Using X Window System

To start X Window System,
select “Xwin Server” icon.

Open another terminal window
`$ xterm &`

Log into Linux WS (earth2 or ocean)
in a “new command window”

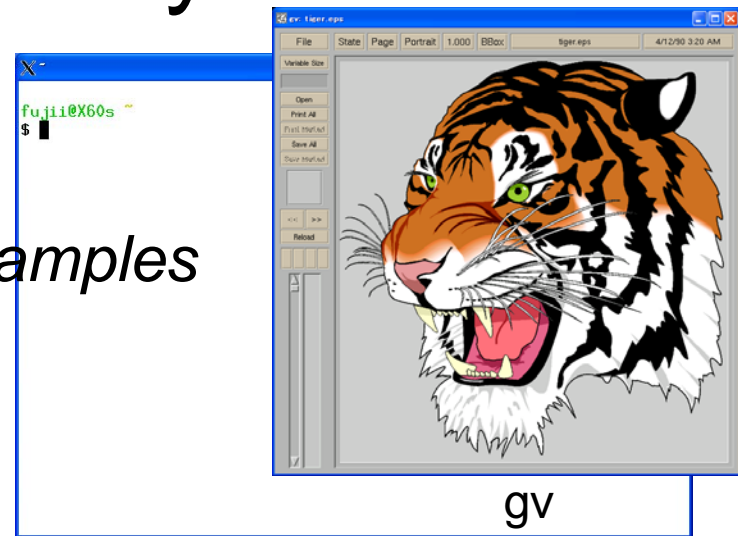
`$ ssh -XC (user ID)@earth2.kenken.go.jp`



Connecting to a UNIX Computer Using X Window System

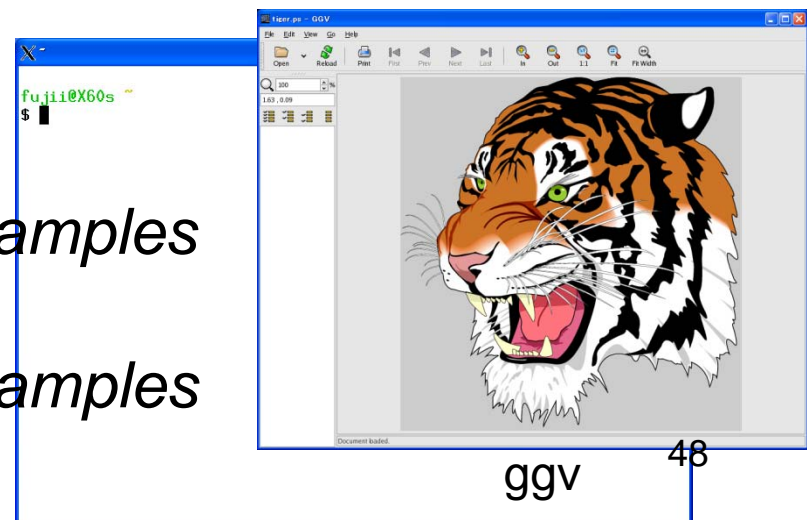
In the “old command window”
(local laptop)

```
$ cd /usr/share/ghostscript/8.50/examples  
$ gv tiger.eps &
```



In the “new command window”
(remote Linux WS)

```
(earth2)  
$ cd /usr/share/ghostscript/7.07/examples  
(ocean)  
$ cd /usr/share/ghostscript/7.07/examples  
$ ggv tiger.ps &
```



Exercise

You will put a *PS* file (e.g., *tiger.eps*) into the Linux WS (earth2 or ocean) using “sftp” or “scp” command, and plot the *PS* file in the Linux WS on your display.

Exercise

On the local computer window, type

```
$ cd /usr/share/ghostscript/8.50/examples
```

```
$ sftp (your ID)@earth2.kenken.go.jp
```

```
sftp> cd UNIX
```

```
sftp> put (PS-file)
```

```
sftp> quit
```

```
$ ssh -XC (your ID)@earth2.kenken.go.jp
```

On earth2 or ocean command line, type

```
$ cd UNIX
```

```
$ ggv (PS-file)
```

Useful Tip 3

Copy and paste the text in X Window, between X Window (e.g., xterm or nedit) and Windows software (e.g., Word, Excel, or IE).

	Copy	Paste
X Window	Just select text	Middle click
Windows software	Select area and (Ctrl)-c	(Ctrl)-v