

SEEM3460 Tutorial I: Lab practice on Unix introduction

SHUAIYI, LI

sli@se.cuhk.edu.hk

2025.09.12

Before we start, pls ensure the followings:

1. You have your SEEM account & password.
 - ▶ Each SEEM student should have an account when you first joined the department.
 - ▶ If you have problems with the account, you can send email to sli@se.cuhk.edu.hk or other TAs email address.
 - ▶ This is important since you need to use it for the assignments and project.
2. Connect to CUHK/SEEM via VPN if you are out of the campus
 - ▶ Otherwise you won't be able to connect to the servers.

Unix Shell

1. Shell: a program that acts as a middleman between you and the UNIX OS
2. Terms similar (but different) to shell:
 - ▶ Terminal / Terminal Emulator
 - ▶ Console
3. A shell allows users to
 - ▶ Run programs
 - ▶ Manage I/O of processes easily

SSH and X Window System

1. SSH (Secure Shell)

- ▶ connect to remote machines and execute commands remotely

2. X Window System

- ▶ commonly used in Unix-like systems to provide GUI environment
- ▶ To run GUI program remotely, you need to provide a local X Server
- ▶ We use VcXsrv to start an X Server on Windows

Required Software (Windows)

1. SSH client (required)
 - ▶ PuTTY (FREE)
 - ▶ MobaXterm
2. X server (optional, but needed for GUI)
 - ▶ VcXsrv: an open-source display server for Microsoft Windows. It enables a user of the Windows operating system to run GUI programs designed for the X Window System.

Required Software (OS X)

1. SSH client

- ▶ No need, OS X have native support
- ▶ Open Terminal and type:
- ▶ `ssh [your UNIX name]@[your host name]`
- ▶ `ssh -X [your UNIX name]@[your host name]` with GUI support

2. X server

- ▶ XQuartz
- ▶ <http://xquartz.macosforge.org>

Step-by-step demonstration

1. We have 2 available servers for this class
 - ▶ linux03.se.cuhk.edu.hk
 - ▶ linux04.se.cuhk.edu.hk
2. Inside SE intranet, you may omit “.se.cuhk.edu.hk” part
3. To connect these servers outside CUHK, you need to connect the CUHK/SEEM VPN.



Windows: Using Putty

1. Download and install putty
2. `https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html`

Package files

You probably want one of these. They include versions of all the PuTTY utilities (except the new and slightly experimental Windows pterm).

(Not sure whether you want the 32-bit or the 64-bit version? Read the [FAQ entry](#).)

We also publish the latest PuTTY installers for all Windows architectures as a free-of-charge download at the [Microsoft Store](#); they usually take a few days to appear there after we release them.

MSI ('Windows Installer')

64-bit x86: [putty-64bit-0.78-installer.msi](#) [\(signature\)](#)

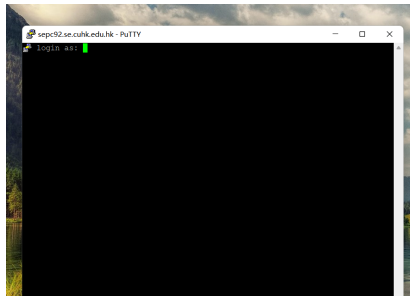
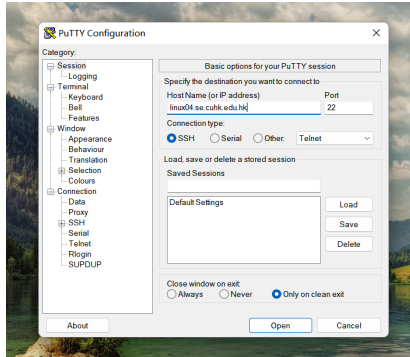
64-bit Arm: [putty-arm64-0.78-installer.msi](#) [\(signature\)](#)

32-bit x86: [putty-0.78-installer.msi](#) [\(signature\)](#)

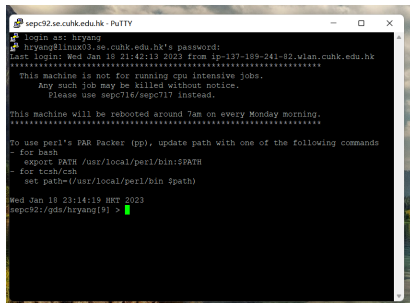
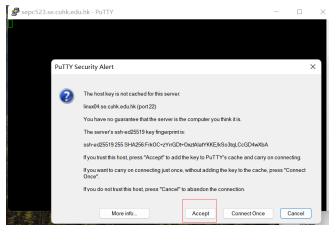
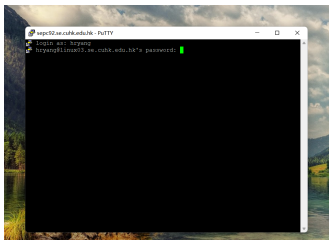
Unix source archive

TAR, GZ: [putty-0.78.tar.gz](#) [\(signature\)](#)

Login server



Login server

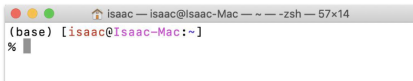


Windows: Using Putty with GUI support (Optional)

1. If you need to run GUI programs like gedit, you need to start a local X Server such as VcXsrv
2. Run VcXsrv
 - ▶ run/start Xlaunch which will ask for some configuration
 - ▶ just take all the default options and click Next until you see the Finish button
 - ▶ click finish and a small X icon will appear in the system tray
3. Run putty.exe
 - ▶ Connection-SSH-X11: Enable X11 forwarding
 - ▶ Session: Host Name: (e.g.) linux03

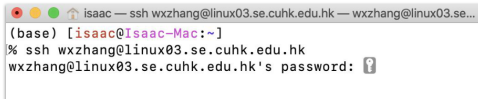
Login server with Mac

- ❏ Open terminal in your Mac



```
isaac — isaac@isaac-Mac — ~ — zsh — 57x14
(base) [isaac@isaac-Mac:~]
% 
```

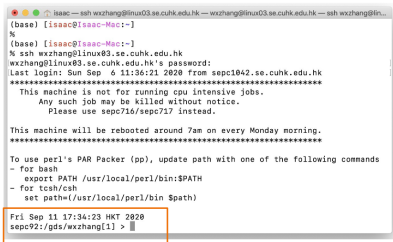
- ❏ Type: `ssh [your UNIX name]@[your host name]`



```
isaac — ssh wxzhang@linux03.se.cuhk.edu.hk — wxzhang@linux03.se...
(base) [isaac@isaac-Mac:~]
% ssh wxzhang@linux03.se.cuhk.edu.hk
wxzhang@linux03.se.cuhk.edu.hk's password: 
```

Login server with Mac

- ❑ After typing in the password, you should see something like this:



```
isaac — ssh wxzhang@linux03.se.cuhk.edu.hk — wxzhang@linux03.se.cuhk.edu.hk — ssh wxzhang@lin...
(base) [isaac@Isaac-Mac:~]$ %
(base) [isaac@Isaac-Mac:~]$ % ssh wxzhang@linux03.se.cuhk.edu.hk
wxzhang@linux03.se.cuhk.edu.hk's password:
Last login: Sun Sep  6 11:36:21 2020 from sepc1042.se.cuhk.edu.hk
*****
This machine is not for running cpu intensive jobs.
Any such job may be killed without notice.
Please use sepc716/sepc717 instead.

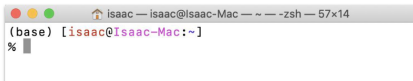
This machine will be rebooted around 7am on every Monday morning.
*****

To use perl's PAR Packer (pp), update path with one of the following commands
- for bash
  export PATH /usr/local/perl/bin:$PATH
- for tcsh/csh
  set path=(/usr/local/perl/bin $path)

Fri Sep 11 17:34:23 HKT 2020
sepc92:/gds/wxzhang[1] >
```

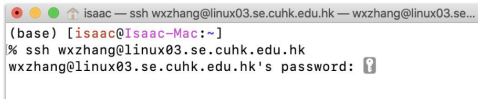
Login server with Mac

- ❏ Open terminal in your Mac



```
isaac — isaac@isaac-Mac — ~ — zsh — 57x14
(base) [isaac@isaac-Mac:~]
% 
```

- ❏ Type: `ssh [your UNIX name]@[your host name]`



```
isaac — ssh wxzhang@linux03.se.cuhk.edu.hk — wxzhang@linux03.se...
(base) [isaac@isaac-Mac:~]
% ssh wxzhang@linux03.se.cuhk.edu.hk
wxzhang@linux03.se.cuhk.edu.hk's password: 
```

Utilities

1. Unix utilities are the basic programs supporting the user to control the system
2. Examples:
 - ▶ `date`: shows the system date
 - ▶ `man -s 1 ls`: shows help on `ls` from the built-in manual section 1
 - ▶ `pwd`: prints the working director
 - ▶ `echo`: prints a message

A shell command

1. ls -lp
2. ls: program name, “ls” is the utility to list files
3. -lp: options/switches, starting with a hyphen
 - ▶ “l” means list in long format
 - ▶ “p” means putting a slash after directory names
4. : remaining parameters
 - ▶ actual meaning depends on the program used
 - ▶ for ls, the remaining parameters are the files or directories to be listed

Unix file system in brief

1. A hierarchy of directories
2. To locate a file in the system, a pathname is needed
3. Command: `pwd`
 - ▶ print your current working directory
4. Pathnames
 - ▶ Absolute pathname, Starting from the root (with a beginning `"/"`)
 - ▶ Relative pathname, Starting from the current working director

Directory: ls, cd, mkdir

1. List files in a directory or fits the pattern
 - ▶ `ls < directories/filenamepatterns >`
2. Change working directory
 - ▶ `cd < aDirName >`
3. Creating (making) a directory
 - ▶ `mkdir < newDirName >`
4. Creating (making) a file
 - ▶ `nano < filename >`

Files operations

1. View the content of a file

- ▶ `cat < filepaths >`
- ▶ `less < filepaths >`
- ▶ `head < filepaths >`
- ▶ `tail < filepaths >`

Files operations

1. Copying a file

▶ `cp < oldFileName > < newFileName >`

2. Remove a file

▶ `rm < FileName >`

3. Remove a non-empty directory

▶ `rm -r < DirName >`

4. Remove a whole directory without prompt

▶ `rm -rf < DirName >`

5. Moving (renaming) a file

▶ `mv < aFileName > < aDirectoryName >`

▶ `mv < oldFileName > < newFileName >`

Editing a file

1. nano
 - ▶ simple, easy to learn and use but no GUI
2. emacs
 - ▶ Adv: has GUI, easy for beginners, but slow
3. vi/vim (vim stands for Vi Improved)
 - ▶ fast for advance users
 - ▶ text-version is quite difficult to learn

Editing a file

1. Compiler

- ▶ `cc` – the native C compiler under Unix
- ▶ `gcc` – C compiler under GNU project
- ▶ Usage is the same, `gcc` is more popular

2. To compile a C source code file, say `example.c`, type in:

- ▶ `gcc example.c`
- ▶ The output of both compilers (i.e. the executable) would be “`a.out`” by default

3. To override the default executable name, use “`-o`” flag

- ▶ `gcc example.c -o example`

4. You can name the executable as `.exe` or `.bin` file to remind yourself the nature of the file

Detailed Procedures

Four Steps of Compilation: preprocessing, compiling, assembly, linking.

| Preprocessing:

Preprocessing is the first step. The preprocessor obeys commands that begin with # (known as directives) by:

- removing comments
- expanding macros
- expanding included files

If you included a header file such as `#include <stdio.h>`, it will look for the `stdio.h` file and copy the header file into the source code file.

The preprocessor also generates macro code and replaces symbolic constants defined using `#define` with their values.

Detailed Procedures

| Compiling:

Compiling is the second step. It takes the output of the preprocessor and generates assembly language, an intermediate human readable language, specific to the target processor.

| Assembly:

Assembly is the third step of compilation. The assembler will convert the assembly code into pure binary code or machine code (zeros and ones). This code is also known as object code.

Detailed Procedures

| Linking:

Linking is the final step of compilation. The linker merges all the object code from multiple modules into a single one. If we are using a function from libraries, linker will link our code with that library function code.

In static linking, the linker makes a copy of all used library functions to the executable file. In dynamic linking, the code is not copied, it is done by just placing the name of the library in the binary file.

Commands

1. `gcc -E hello.c -o hello.i`
2. `gcc -S hello.i -o hello.s`
3. `gcc -c hello.s -o hello.o`
4. `gcc hello.o -o hello`

Practice

- ❑ Get familiar with file/directory-related commands
- ❑ Create a seem3460 folder in your home directory
- ❑ In the tutorial-01 subfolder, create a **hello.c** file with the following content, compile with gcc and run the compiled program to see the output

```
#include <stdio.h>
int main() {
    printf("Hello World\n");
}
```

- ❑ **Note:** Copy and Paste may produce strange characters in your editor, so try to type the code by yourself.