COMP 3511
Operating Systems

Lab 01

Outline

- Welcome
- UNIX basics and Vi editor
- Using SSH to remote access Lab2(4214)
- Compiling a C Program
- Makefile
- Basic C/C++ programming
- (Optional) GDB – text base debugger

Welcome

- TA Self-introduction
  - Room number
  - Office hour

UNIX basic

- Basic command
  - ls
    - ls –a : show hidden files or directories
    - ls –l : list in long listing format
    - ls –al
  - cd
    - cd ~ / cd ; : change to home directory
  - mkdir, rmdir, mv
  - rm
    - rm –r directory : remove the contents of directories recursively
    - rm –f file : remove file without prompt
  - cp: copy a file

UNIX basic

- Practice
  - Create a comp3511 directory for lab1 under home directory

    cd ~

    mkdir comp3511

    cd comp3511

    mkdir lab01

    cd lab01
  - Copy a file from comp3511 directory

wget http://course.cse.ust.hk/comp3511/lab/lab01/Makefile/numprint.c

wget http://course.cse.ust.hk/comp3511/lab/lab01/Makefile/main.h

wget http://course.cse.ust.hk/comp3511/lab/lab01/Makefile/main.c

UNIX basic

- Cat
  - cat “filename” : display content of a file
  - cat > “filename” : create and append content to a file
  - cat >> “filename” : append content at the end of a file
- Practice
  - cat main.c // show you the content of main.c
  - cat > helloworld.txt (enter)
    - type “Hello World” (Enter) **(Ctrl + D)**
  - cat helloworld.txt

UNIX basic

- Get help information to see how to use UNIX command
  - rm --help
  - cp --help | more
  - cat --help
- find command
  - search for files in a directory hierarchy
  - try “find --help | more”
  - Search main.c file in your home directory
    - find . -name main.c

UNIX basic

- Useful links
  - http://course.cse.ust.hk/comp3511/Reference.html (References)

Vi – Starting vi

- You can use editors under X-windows like “kate”, you can also use a text base editor like “Vi”
- Starting vi
  - vi “filename” – Start at line 1 of file
  - vi +n “filename” – start at line n of file
  - vi + “filename” – start at last line of file
  - vi –r “filename” – recover file after a system crash
- Two modes in vi
  - Insertion mode : press “i” or “I” enter this mode
  - Command mode : press “Esc” enter this mode

Vi – saving files and leaving vi

- Saving files
  - :e “filename” – save current and edit other file
  - :w – save current editing file
  - :w “filename” – save as file
  - :w! “filename” – save as existing file
- Leaving vi
  - :q – quit vi
  - :wq – save file and quit vi
  - :q! – quit vi without saving
- Copy : ‘yy’
- Paste: ‘p’
- Cut: ‘cc’

Vi – commands

- Moving cursor
- Inserting text
- Changing and replacing text
- Deleting text
- Markers
- Search and replace
- ……
- (Go to here)
  - http://course.cse.ust.hk/comp3511/references/vi-ref.pdf

Vi – Practice

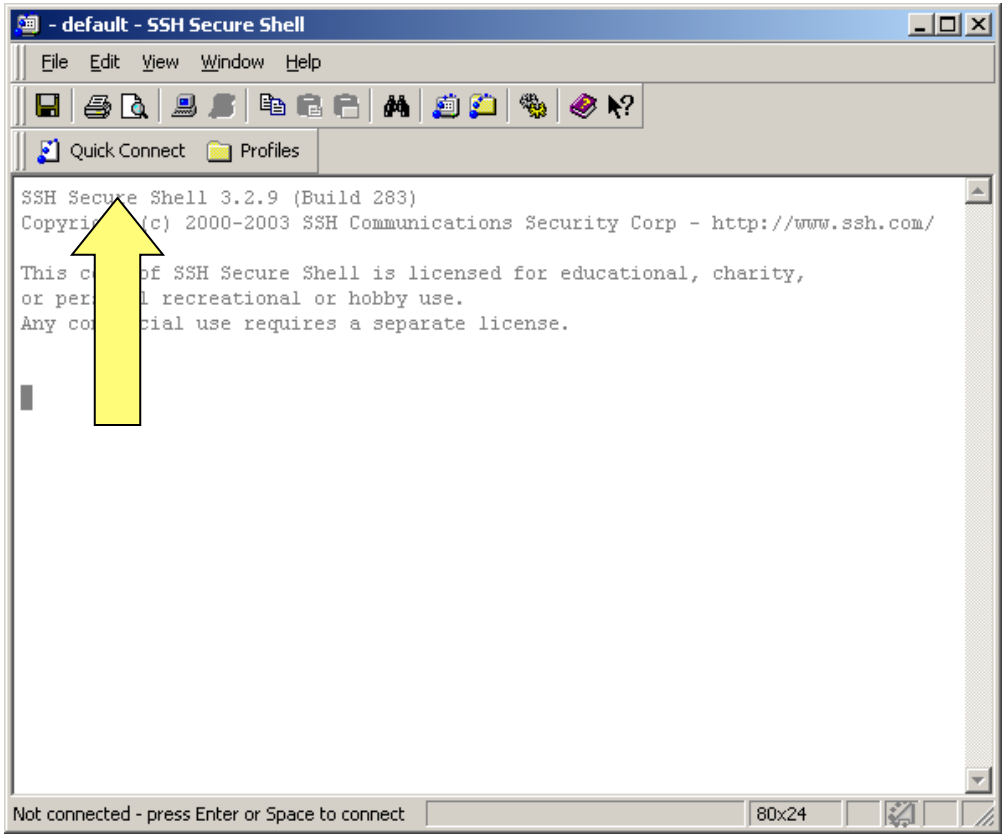
- Practice
  - Open helloworld.txt
    - vi helloworld.txt (In command mode at the beginning)
    - (press “i” to enter insertion mode, you can edit the file now)
    - (press ‘Esc’ to enter command mode)
    - (press ‘dd’ to delete current line)
    - (press ‘ndd’ to delete n line below current line)
    - (press ‘:wq’ in command mode to leave vi and save the file)

Remote access Lab2(4214)

- Software
  - SSH secure Shell
  - http://www.ssh.com
  - http://cssystem.cse.ust.hk/home.php?docbase=UGuides/RemoteAccess&req_url=UGuides/RemoteAccess/ssh.html
  - Putty
  - http://www.putty.nl
- UNIX ssh command
  - ssh –l “username” “hostname”
  - eg. ssh –l lfxad csl2wk35.cse.ust.hk

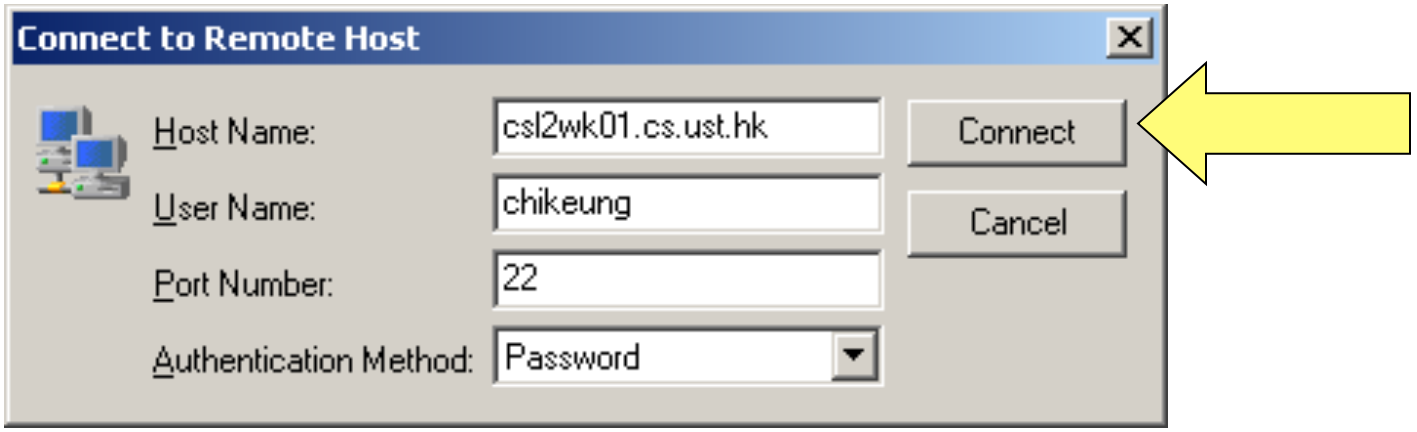
How to use SSH secure shell

- After you install SSH and start it, click “Quick Connect”



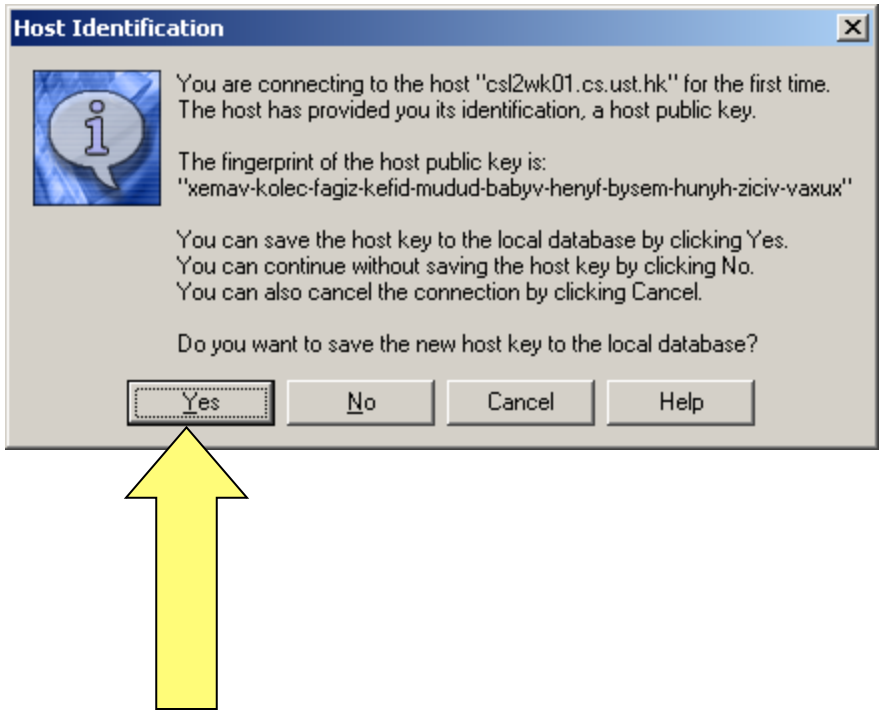
Logging Page

- Host Name: csl2wk***XX***.cse.ust.hk
  - ***XX*** can be 01 to 40
- User Name: your CSD logging account
- Port Number: 22 (SSH default port)
- Authentication method: Password



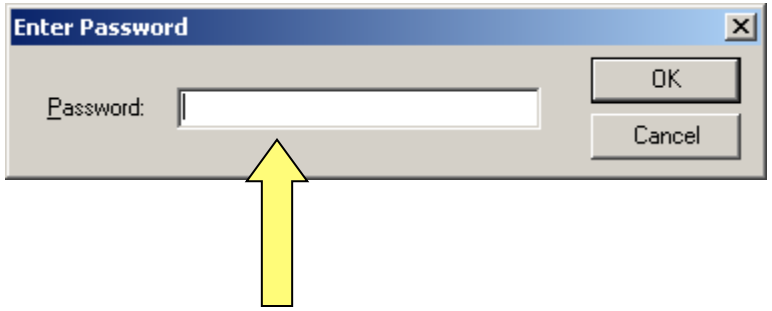
First time logging to a machine

- Press “Yes” to save a host key
  - If you save it in your local computer, this windows will not popup when you logging to same machine again

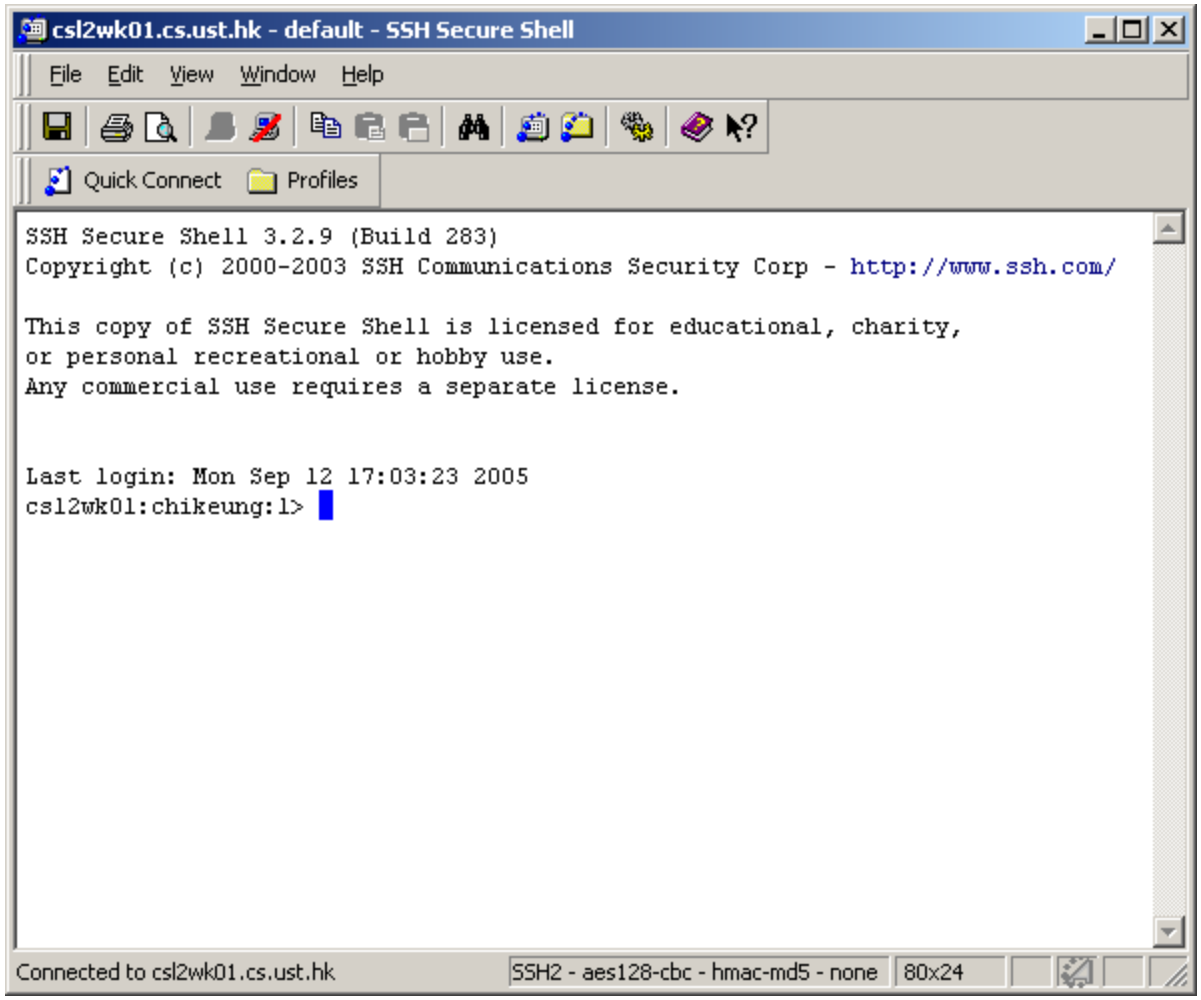


Enter your password

- Enter your password and click OK



You logging to a machine!!!



Notes

- A machine can be accessed for more than one user at the same time, use “who” command to know who is using the machine
- You can click “Window -> New Terminal” to start another terminal to the same machine
- SSH secure Shell is available in the CS lab 1, lab 3, lab 4 and all computer barn.

Compiling a C Program

- GNU's C Compiler (gcc)
  - The main compiler that will be used in this course
  - For compiling C++ programs, you can use g++
  - Linux/UNIX does not have a pretty program like Microsoft Visual Studio for managing C programming projects

- Compiling a C program

  gcc –g –m32 –ansi –Wall –c main.c –o main.o

  gcc –g –m32 –ansi –Wall –c numprint.c –o numprint.o

  -g: include symbols for gdb debugger.

  -m32: compile and link 32bit i386 binaries

  -ansi: check code against ansi C standard.

  -Wall: display all warnings.

  - Use –ansi and -Wall for user programs.
  - Use –Wall for kernel programs.

- Linking

  gcc –g –m32 –ansi –Wall main.o numprint.o –o lab01

Makefile

- You need to type quite a bit for compiling a simple program
- For large projects
  - You may have many .c and .h files
  - You may use many library calls
  - You need to specify them at compilation time.
  - Things start to get tedious and messy
- One way to manage this complexity is to use a Makefile
  - Automates the compilation process
  - Easy to declare all the compilation options and flags

Makefile

- Example Makefile

```
SRCS = main.c numprint.c
HDRS = main.h
OBJECTS = main.o numprint.o
INCLUDE = -I/usr/local/include
LIBS = -lm
CC = gcc
CFLAGS = -g –ansi –Wall
EXEC = lab01
all:     $(EXEC)
$(EXEC): $(OBJECTS)
    $(CC) $(CFLAGS) $(INCLUDE) $(LIBS) $(OBJECTS) –o $(EXEC)
clean:
    rm –f $(OBJECTS) $(EXEC) core *~
depend:
    makedepend -- $(CFLAGS) $(INCLUDES) $(SRCS) $(HDRS) –

# DO NOT DELETE THIS LINE – make depend depends on it.
```

Makefile

- The first few lines are fairly straightforward
  - SRCS, HDRS, and OBJECTS specify the source, header, and object files
  - INCLUDE, the directory for include files
  - LIBS, the library to be linked into the compilation
  - CC, the type of C compiler
  - CFLAGS, the compilation flags
  - EXEC, the name of the executable image.
- The line containing *all* specifies the final compilation targets
  - in this case, the content of EXEC, or $(EXEC).
- The creation of $(EXEC) depends on the $(OBJECTS), or object files.
- To create $(EXEC), the compiler needs to link the objects by running the $(CC) command.
- All the .c files are automatically converted to .o files without the need of specifications.
- Note that you need to **tab** the indentation of the $(CC) command. Makefile won't work if you use the space bar to create the indentation.
- Lines start with # denote comments. Let's ignore the remaining lines for now. With this Makefile in your current project directory, all you need to type to compile your project is ***gmake***.

Makefile

- Prepare a Makefile

  wget http://course.cse.ust.hk/comp3511/lab/lab01/Makefile/Makefile
  wget http://course.cse.ust.hk/comp3511/lab/lab01/Makefile/numprint.c
  wget http://course.cse.ust.hk/comp3511/lab/lab01/Makefile/main.h
  wget http://course.cse.ust.hk/comp3511/lab/lab01/Makefile/main.c
  cat Makefile
- Run the Makefile
  - gmake
- More about Makefile
  - http://www.gnu.org/software/make/manual/html_node/index.html

Basic C/C++ programming

- Please read a C tutorial on our course web
  - http://course.cs.ust.hk/comp3511/lab/lab01/Cbasics/c_tutorial.pdf

  And write a “HelloWorld” C program

- Understand command line arguments in C/C++ program
  - http://course.cs.ust.hk/comp3511/lab/lab01/Cbasics/command-line_color.pdf

  And run ./lab01 –h –n 5

Basic C/C++ programming

- Read and understand the source code that we just compiled
  - main.h, main.c, numprint.c
  - Run ./lab01 –h –n 5
- A quick introduction to C++
  - http://course.cs.ust.hk/comp3511/lab/lab01/Cbasics/c++.pdf
- Useful Links
  - http://course.cse.ust.hk/comp3511/Reference.html (References)

(Optional) Debugging

- Printf/Printk
  - Simple programs can often be debugged efficiently with well placed print statements.
    - See the example programs to see how we use print statements to verify the arguments supplied
  - Kernel programming requires the use of printk for debugging.
- gdb
  - The GNU debugger. An interactive debugger.
  - Useful for debugging complex logic.
- gdb name_of_executable
  - Starts GNU debugger (gdb) and loads the executable file into memory.

(Optional) GDB debugger

- Basic command is introduced here:
  - http://course.cs.ust.hk/comp3511/Others/GDB.htm
  - http://www.dirac.org/linux/gdb/

(Optional) GDB Practice

- Compilation
  - wget http://course.cse.ust.hk/comp3511/lab/lab01/Makefile/gdb.c
  - gcc **-g** -o gdb_demo gdb.c
  - gdb gdb_demo
- View the code
  - list (view 10 lines of code)
  - list (view next 10 lines of code)
  - list - (view previous 10 lines of code)
  - list 3,8 (view line 3 to line 8)
- Run it
  - run 1 2 (run the program with arguments)

(Optional) GDB Practice

- Insert / delete breakpoint
  - break 7 (inset a break point in line 7)
  - info breakpoint (view break point information)
  - delete 1 / clear 1 (delete break point # 1)
- Enable / disable breakpoint
  - break 7
  - break 16
  - run
  - disable 3 (disable break point # 3)
  - run (can you see the different?)
  - info breakpoint
  - enable 3 (enable break point #3)

(Optional) GDB Practice

- Print out the value of a variable
  - disable 3
  - run (program will stop at line 7)
  - print i (print out the current value of i)
  - next (go to next line, it is line 6)
  - step (go to next line, it is line 7)
    - Note: different between next and step
  - print i (i become 1)
  - continue (go to next break point, it is line 7 again)
- Quit GDB
  - quit