OS Tutorial 1:

Linux System & C Programming Basic

Huan Wang, Changli Zhang
huanwang, clzhang@uvic.ca

About the Course Tutorial

- Tutorial Instructor:
  - Huan Wang: huanwang@uvic.ca
  - Changli Zhang: clzhang@uvic.ca
- Time & Location:
  - Tutorial Hour:

Thursday	1:30 – 2:20 p.m.	Location: CLE A315
Friday	9:30 – 10:20 a.m.	Location: DSB C108
Friday	2:30 – 3:20 p.m.	Location: ECS 104

  - Office Hour:

Huan:	Friday 4:30 – 5:30 p.m.	ECS 317
Changli:	Wednesday 10:30 – 11:30 a.m.	ECS 330

About the Course Tutorial

* What I can do for you:
  * Help you understand the assigns.
  * Provide required knowledge to complete the assigns.
  * Give hints/tips at key points of assigns.

* It is **your** responsibility to:
  * Prepare solution codes of assigns.
  * Debug your programs.
  * Pay attention to the due time.

OS Tutorial 1:
Linux System & C Programming Basic

Outline

- **Linux System Basic**
- C Programming Basic

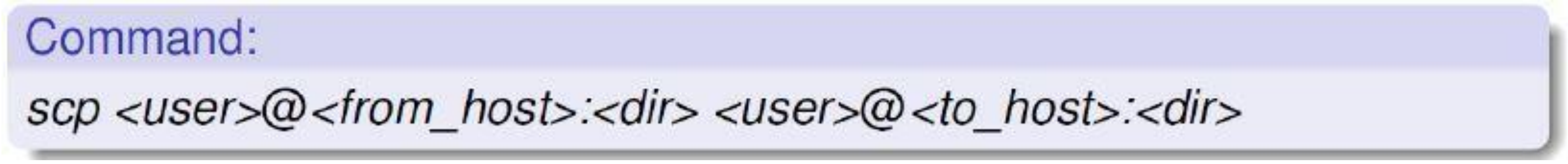
Linux Basic (1)

- Linux Distributions: Ubuntu, CentOS, Debian, etc.
- Use Linux
  - ***Local machine:***
    - your laptop with Linux OS or VM (e.g., VirtualBox) in Windows
    - Drop in ECS 242
  - ***Remote access via SSH***
    - Linux and Mac OS X:
      ssh NetlinkID@linux.csc.uvic.ca
      ssh -l netlinkID linux.csc.uvic.ca
    - Windows users: PuTTy, MobaXterm, etc.

Linux Basic (2)

Remote copy file:

* Use command : Linux and Mac OS X:



Command:

```
scp <user>@<from_host>:<dir> <user>@<to_host>:<dir>
```

* Use app: Windows/Mac users: WinSCP, FileZilla etc. (use port number 22 (sftp))
* ***Zip your assignments (code) as tarfile***
* tar -czvf (create archive) ; tar -zxvf (extract files from archive)

Linux Basic (3)

* Linux Shell
  * An interpreter between users and Linux kernel
* Basic operation commands
  * **man**: manual pages (IMPORTANT)
  * **ls**: list directory contents
  * **pwd**: print working directory
  * **cd**: change directory
  * **cp**: copy files from source to dest
  * **mv**: cut and move files from source to dest
  * **mkdir**: create a directory
  * **rmdir**: remove a directory
  * **rm**: remove files
  * **chmod**: change file mode bits, permissions
  * **exit (ctrl + d)**

Frequently used options:

-a, -d,
-l

-i, -r

-i

-i, -r,
-f
-R

Linux Basic (4)

- **style 1: <u>$ chmod</u> <u>xyz</u>   <u>filename :</u>**

  Use digits to represent the permission of file: r: 4, w: 2, x: 1.

  E.g., change a file's permission as [rwx r–x r–x]

  <u>$ chmod 755 filename</u>

  (owner=rwx=4+2+1=7, group=r–x=4+0+1=5, others=r–x=4+0+1= 5)

- **style 2: <u>$ chmod</u> <u>u/g/o/a</u>   <u>+/-/=</u>   <u>filename :</u>**

  E.g., change a file's permission as [rwx r— r—],
  <u>$ chmod u=rwx,go=r filename</u>

  Give the permission 'x' to group member:
  <u>$ chmod g+x filename</u>

Linux Basic (5)

- User commands are in Section 1 of the manual pages
  - **$ man 1 cp**
- Other sections of the man pages
  - Section 1: user commands (e.g., **$ man 1 man**)
  - Section 2: system calls (e.g., **$ man 2 kill**)
  - Section 3: library functions (e.g., **$ man 3 exec**)
  - …
  - Full list of sections info.: http://linux.die.net/man/

C Programming Basic (1)

- Why C language?
  - Better control of low-level operations
  - Better performance
  - Other languages, like Java and Python, hide many details for OS level interaction and coding
    - Process mgmt.
    - Memory mgmt.
    - Error detection

C Programming Basic (2)

* What you need:
  * text editor + compiler + C standard library
* Editor:
  * Command line editor: vi, vim
  * GUI editor: gedit (installed in ECS 242 machines)
* Compiler: GNU Compiler Collection (GCC)
  * **$ gcc example.c –o output**
  * **$ ./output**
* Debugger:
  * gdb

C Programming Basic (3)

- 1. Create and Edit Source Files
  - Using editors mentioned before: vim, gedit or emacs etc.
  - An example: **$ vim hello.c**
- 2. Compile Single Source File
  - **$ gcc hello.c -o hello**
  - Preprocess -> compile -> assemble -> link
  - Warning info.: **$ gcc –Wall hello.c –o hello**
- 3. Execute Output
  - **$ ./hello**

C Programming Basic (4)

- 4. Compile Multiple Source Files
  - **$gcc –c main.c –o main.o**
  - **$gcc –c add.c –o add.o**
  - **$gcc main.o add.o –o result**

  - What if you have more source files?

C Programming Basic (5)

- 5. Makefile for multiple source files
  - Basic Syntax:

```
Target: [dependencies]
 [TAB] <command>
        <command 2> …
```

  - Example: Makefile of the example in 4.

```
result: main.o add.o  gcc
        main.o add.o -o main
main.o: main.c add.h
        gcc -c main.c
add.o: add.c add.h
        gcc -c add.c
clean:
        rm *.o
```

  - Command:
    - **$ make**
    - use **–f** to specify a Makefile: **$ make –f myMakefile**
  - Tutorials:
    - http://mrbook.org/blog/tutorials/make/
    - http://www.cprogramming.com/tutorial/makefiles.htm
l

C Programming Basic (6)

- 6. Debug Programs
  - GDB:
    - **$gcc –g hello.c –o hello**
    - **$gdb hello**
  - Official doc.:http://www.gnu.org/software/gdb/documentation/

- Step-by-step tutorial: https://www.cprogramming.com/gdb.html

C Programming Basic (7)

- Available Online C Programming Tutorials
  - http://www.cprogramming.com
  - http://www.cprogrammingexpert.com/C/
  - http://einstein.drexel.edu/courses/Comp_Phys/General/C_basic_s/
  -
  - Of course, YouTube!

Outline

* Linux Basic
* **C Programming Basic (Questions?)**

Contributors:

* **Cheng Chen, Guoming Tang, Yongjun Xu**
* **Huan Wang, Changli Zhang**