RISC-V

System-on-Chip Design

Harris, Stine, Thompson & Harris

Appendix B: Hitchiker’s Guide to Linux

Appendix B :: Topics

Introduction to Linux

- Linux is a powerful, free, and open operating system (OS).
- It is the only environment supported by commercial logic synthesis tools.

Linus Torvalds

- Primary developer of the Linux kernel.
- He started the project for fun as a student and it became his master’s thesis at the University of Helsinki.
- He released version 1.0 in 1994 and remains the guiding force for the kernel development.

Photo:
https://en.wikipedia.org/wiki/Linus_Torvalds#/media/File:Lc3_2018_(263682303)_(cropped).jpeg

Appendix B: Linux

Connecting to a Linux Server

Linux Server Introduction

- **Linux server**
  - Typically has all the computer-aided design (CAD) tools installed – so users don't have to install them locally.
  - Servers are typically powerful – lots of RAM, hard drive space, processors, and speed
  - System administrator for server provides username and password
- **X11 (also simply X): Linux graphical user interface (GUI)**
  - To use GUI remotely, user may wish to install one of:
    - VNC (virtual Network Computing): RealVNC's VNC Connect is widely used but has a monthly subscription fee
    - X2Go: Not as fast as VNC but free & fairly easy to use
- **ssh** is sufficient for terminal-based work that doesn't require a GUI

X2Go

- **x2goserver** must be running on Linux server (check with system administrator)
- User downloads client (x2goclient) from **x2go.org**
  - Mac users – must also install X windows server Xquartz from xquartz.org
  - Mac users may also have to go to System Preferences → Security and Privacy → General and under "Allow apps downloaded from" choose Open Anyway
- **Start x2goclient** on user's laptop.
- Choose **Session → New Session**. Give the session a name.
- Enter the **Linux server host name** in Host, and your username in Login.
- Set Session type to **XFCE**
- Click on this session, enter your password, and choose OK.
- The session on the server will open.
- You can **suspend** your session and reopen it later and come back to the same windows and programs you had been using.
- Long jobs will continue to run on the server even when your session is suspended.

Change password

- Change your password by typing the following at a terminal on the server:

```
passwd
```

- The server will ask for your old password and then you will enter your new password twice.

Appendix B: Linux

Working with Files

Working with Files

- Typically navigate around file system using a terminal.

Command	Description	Example
`mkdir`	Make (create) a directory/folder	`$ mkdir tutorial`
`ls`	List files in current directory	`$ ls tutorial`
`cat`	"Concatenate": put text into a file. Press Ctrl-D to end the file	`$ cat > README` `Hi!` `<Ctrl-d>`
`less`	Display a file's contents one screenful at a time. (Older version was "more" but… less is more.)	`$ less README` `Hi!` `Press q to quit.`
`cp`	Copy a file. For example, copy README to README2.	`$ cp README README2` `$ ls` `README README2 tutorial`
`mv`	Move or rename a file. Example: -rename README2 to README3 -move README to tutorial directory.	`$ mv README2 README3` `$ mv README tutorial`
`cd`	Change directories. cd by itself changes to your home directory.	`$ cd tutorial` `$ ls` `README`
`pwd`	Print working directory: print path of your current directory.	`$ pwd` `/home/lynn/tutorial` `$ cd` `$ pwd` `/home/lynn`

Working with Files, cont'd

Command	Description	Example

`..`	Indicate the directory above your current directory. So cd .. moves up a level.	`$ cd ..`
		`$ pwd`
		`/home`

`~`	Indicate your home directory.
	~lynn is the home directory of user lynn.
	(You could indicate full path: cd /home/lynn/tutorial)
		`$ cd ~/tutorial`
		`$ pwd`
		`/home/lynn/tutorial`

`ls –l`	Give long listing with file permissions, owner, group, size, and modification date for each file. Permissions: 10 characters: directory, readable/writable/executable for user/group/all. Next is the number of files in a directory. The third and fourth columns indicate the user and group that own the file. A user can add other users to their private group with /usr/sbin/groupadd. The sixth column lists the date and time the file was last modified.	$ ls -l -rw-rw-r-- 1 ben users 4 Nov 26 06:24 README3 drwxrwxr-x 2 ben users 3 Nov 26 06:29 tutorial README3 can be read and written by the user and anyone in the group, but it can only be read by those outside of the group.

`chmod`	Change file permissions. It takes 3 octal digits indicating the permission for user, group, and world. Each digit has three bits indicating read (4), write (2), and execute (1). For example, to make README3 read and writable by lynn, readable by users, and inaccessible by others.	$ chmod 640 README3
		$ ls -l README3
		-rw-r----- 1 lynn users 4 Nov 26 06:24 README3

Appendix B: Linux

More Handy Commands

Frequently Used Commands

- Working with characters & text
- Other common commands
- Working with processes
- Compressing, bundling, and transferring files

Cast of Characters

Command	Description	Example

`*`	A wildcard that matches any set of 0 or more characters	`$ ls *.sv`
		Lists all files ending in .sv
		`$ rm -rf *`
		Removes all files in current directory and any subdirectories. Use with caution!

`.`	The current directory	`$ mv tutorial/README .` Moves README from the tutorial subdirectory to the current directory.
`&`	Run a command in the background so user can continue using the terminal.	`$ vsim &` Runs the ModelSim simulator, opening the window in the background.
`>`	Redirect the output of a command to a file	`$ ls *.sv > svfilenames` Puts a list of all the files ending in .sv into a file named svfilenames rather than displaying the list on the screen.
|	Pipe output of one command to input of another	$ ls -l | less Lists all files in a directory, 1 page at a time.

Other Common Commands

Command	Description	Example
`man`	Print a manual page for a Linux command, including all the options.	`$ man ls` LS(1) User Commands LS(1) NAME ls - list directory contents SYNOPSIS ls [OPTION]... [FILE]... DESCRIPTION List information ... (Mandatory) arguments -a, --all do not ignore entries starting with . ...

`which`	Print the path to where a command is found.	`$ which ls`
		`/usr/bin/ls`

`grep`
 	
(global regular expression print)
	Search specified files for a string and print where found. The string may be simple text, or a *regular expression* (see The Linux Command Line for examples).	`$ grep XLEN *.sv *.vh`
		Prints all places XLEN appears in file ending with .sv or .vh.
		`$ grep XLEN *`
		Prints all the places XLEN appears in any file in current directory
		`$ grep -r XLEN`
		Prints all the places XLEN appears in any file in the current directory or recursive subdirectories

Other Common Commands, cont'd

Command	Description	Example
`diff`	Compare two files and print the differences.	`$ diff file1 file2` Compares file1 and file2 and prints the differences.
`sort`	Print file with the lines sorted.	`$ sort file > file1_sorted`
`wc`	Print # of lines, words, bytes.	`$ wc file1.txt`
`find`	Find all the files in specified directory (& its subdirectories) matching a particular criterion.	`$ find . –name *.sv` Prints the paths to all the files in the current directory and below, whose names end with .sv.

`du`	Report disk usage (file size). Use –h option to print in units of KB, MB, or GB.	`$ du –h`
		`1.0K     ./tutorial`
		`2.0K     .`

`ln –s`	Create a symbolic link to a file. Symbolic link: same as shortcut in Windows or alias on macOS.	`$ ln –s tutorial/README README4`
		`-rw-r----- 1 lynn users  7 Nov 26 08:26 README3`
		`lrwxrwxrwx 1 lynn users 15 Nov 28 10:15 README4 –> tutorial/README`
		`drwxrwxr-x 2 lynn users  3 Nov 26 06:29 tutorial`

Working with Processes

Command	Description	Example
`up/down arrow on keyboard`	Bring back previous commands to cursor.	`$ ls -als` Recalls previous commands to save typing.

`history`	Print the previous commands you entered.	`$ history`
		…
		`102 ls *.sv > svfilenames`
		`103 ls -l | less`

`!!`	Repeat previous command.
	`!`<string> repeats the last command starting with string.
	`!`<number> repeats a numbered command reported by history.
		`$ !!`
		runs `history` again (if it was the last command).
		`$ !c`
		runs `chmod` again (if it was the last command starting with c).
		`$ !103`
		runs `ls -l | less` (see history above).

`top`	Display processes with greatest load on the computer. `htop` gives more info for multiprocessors.	`$ top`
		Press 1 to view the load on each CPU.
		`$ htop`

Working with Processes, cont’d

Command	Description	Example
`ps`	Display the processes currently running, including their process ID (PID) & fraction of CPU and memory they are consuming.	`$ ps –u` Prints all the processes you are running.

`kill`	Terminate process specified by its PID. Especially useful if you have a runaway process taking up CPU time. Add the –9 flag to forcefully kill a process if the regular kill command doesn’t work.	`$ kill <PID>`
		`$ kill –9 <PID>`

`killall`	Terminate all processes running the specified command.	`$ killall vsim` Kills all vsim jobs.

Compress, Bundle, Transfer Files

Command	Description	Example

`gzip`	Compresses files. Works best for text files with repetitive or structured content, achieving compression of around 50% for a plain text novel to 90+% for more repetitive files. `gzip -d` or `gunzip` decompresses.	`$ ls -l`
		`-rw-rw-r-- 1 ben users 1232922 moby.txt`
		`$ gzip moby.txt`
		`$ ls -l`
		`-rw-rw-r-- 1 lynn users 505050 moby.txt.gz`
		`$ gunzip moby.txt.gz`

`tar` (tape archive)	Create a *tarball*, a single file containing directory contents. Easier to move directories between systems.
	Options:
	`-c` Create the archive
	`-x` eXtract the archive
	`-v` Verbose: display which files are archived
	`-f` Filename of archive
	`-z` gZip compress the archive
		`$ tar -czvf archive.tar.gz tutorial`
		Archives and compresses the tutorial directory and all its contents
		`$ tar -xzvf archive.tar.gz`
		Uncompress and expand the archive

Compress, Bundle, Transfer Files

Command	Description	Example

`rsync`	Flexible tool to copy/synchronize files or directories. Directories can be remote or local, and rsync can efficiently sync on only modified files or files that do not exist at the destination.	`$ rsync lynn@vlsi.hmc.edu:archive.tar.gz .`
		Copies archive.tar.gz from lynn's home directory on vlsi.hmc.edu to the current working directory where you issued the command. If the files already exist, only copy the modified ones.
		`$ rsync -auv --info=progress2 src/mmu lynn@vlsi.hmc.edu:src/new`
		Verbosely copies src/mmu directory from local machine into src/new directory on remote computer. Preserves permissions, copies only files which are newer on the local machine, and provides a progress bar.

Appendix B: Linux

Linux Productivity

Multiple Terminal Windows

- Keep terminal open in each directory you're working in.
- Desktop environment usually has a good terminal, including:
  - Scroll bars
  - Copy/paste
  - Multiple tabs
- If not, type following at command:

```
xterm –sb –sl 5000
```

- Close terminal by typing following in terminal:

```
exit
```

Text Editors

- **Most widely used text editors:**
  - **vim**
  - **emacs**
- vim (newer version of vi – if vim not available, try vi)
  - Fast, universally available, but cryptic
  - At command, to open mmu.sv file, navigate to directory & type:

    ```
    vim mmu.sv
    ```
  - In current mode, typing keys invokes commands:

dd	Delete current line
i	Insert mode (now can type text instead of command)
Esc key	Return to command mode
u	Undoes last command
/text	Searches for "text" in document
p	Pastes last line(s) that were deleted
:w	Writes (saves) file
ZZ	Saves file & quits vim
:q!	Quits without saving

Free vim user manual: http://www.eandem.co.uk/mrw/vim/usr_doc/index.html

Text Editors: VSCode

- **Visual Studio Code (also called VSCode)**
  - GUI-based
  - Install yourself (code.visualstudio.com)
- **Preferred editing method, as long as good network connection**
  - Not responsive over x2go connection
- **After VSCode installed, install extensions (click on Extensions icon: or click on gear icon in lower-left and select Extensions):**
  - Remote – SSH
  - Verilog-HDL/SystemVerilog/Bluespec (for syntax highlighting)
  - GitLens (helps track modifications to lines of files in a Git repository)
- Click on green status bar icon with "><" to set up ssh connection to server.
  - Choose Remote-SSH: Connect to Host… enter server name (e.g., tera.eng.hmc.edu)
  - Choose File → Open Folder & enter path to repository (e.g., home/lynn/riscv-wally)
  - Now editing files on server directly

Linux Filesystem

Directory	Description
/	The root of the file system
/home	User home directories.
/bin /sbin /lib	Binary executable files and shared libraries needed to boot the system. /bin: holds most common commands like ls and cd. /sbin: holds system binaries used by sysadmins. /lib: holds libraries such as the standard C library libc.so.
/boot	The bootloader files such as the Grand Unified Bootloader GRUB, the compressed Linux kernel vmlinuz, and the root filesystem image initramfs.
/etc	Configuration files, such as /etc/passwd
/usr /usr/bin, /usr/lib /usr/local/bin, /usr/local/lib /opt	Additional software not required for system boot.
/dev	I/O devices: accessed using Linux file commands from /dev
/mnt	A place for extra filesystems to be mounted (USB, etc.)
/tmp	A place for temporary files, could be deleted when rebooting.

Environment Variables

- Configure how the system operates.
- Environment variable names:
  - Start with $
  - Usually upper case
- **$PATH**
  - One of most important environment variables
  - Defines an ordered list of directories that Linux searches for commands.
- **Commands**
  - `echo` command prints path

    ```
    $ echo $PATH
    /usr/local/bin:/bin:/use/bin:/home/ben/bin:/sbin
    ```

  - `export` changes path (or by itself, lists all environment variable settings)

    ```
    $ export PATH=/cad/verilator/bin:$PATH
    $ echo $PATH
    /cad/verilator/bin:/usr/local/bin:/bin:/use/bin:/home
    /ben/bin:/sbin
    ```

Environment Variables, cont'd

- **Files: `.profile` or `.bash_profile`**
  - Sourced (run) every time you open a terminal
  - Add necessary export statements to these files so that your setup is correct, or (even better) source a shared configuration file:

```
source /cad/scripts/setups/S21/riscv-setup
```

- **Other common environment variables:**

  LD_LIBRARY_PATH: like PATH, but used when searching for shared library files
  LM_LICENSE_FILE: points to a license manager, commonly for paid CAD tools

Other Operations

- **Web Browsing**
  - Generally less responsive from Linux server
  - But Firefox is commonly available (type `firefox` at terminal)
  - Or download file to your local machine and rsync it to the server
- **Document Viewing & Printing**
  - `evince` and `ocular`: common tools for viewing PDF, Postscript, & other image files
  - **Printing:** best bet may be to generate plain text or print to PDF and then rsync the text/PDF file to your local computer for printing. Or do a screen capture with software from your local computer (Snipping Tool on Windows / Screenshot on MacOS).
- **Installing Packages**
  - `dnf` (formerly `yum`): package manager for RedHat
  - `apt-get`: package manager for Ubuntu
  - For new Linux users, ask sysadmin for help

Appendix B: Linux

Scripting & Programming

Shell Scripts

- **Shell script:** A text file with series of commands that executes as if typed at commend line
  - File typically ends with .sh suffix
- **Example:** Create file called hello.sh. hello.sh contents:

```
echo Hello World!
```

- Execute script:

```
$ source hello.sh
```

or make file executable for all (rwe: 755 = 111 101 101) and execute:

```
$ chmod 755 hello.sh
$ ./hello.sh
```

Example Shell Script

- **Script for rsync'ing files between computers**
- **Process:**

```
$ mkdir ~/bin              (only if necessary)
$ cat > ~/bin/mysync
#!/usr/bin/bash
rsync -auv --info=progress2 src/${1} ben@vlsi.hmc.edu/src
<Ctrl-d>
$ chmod 755 ~/bin/mysync
$ mysync foo.sv
```

(Linux will find the script if ~/bin is in your PATH)

The last line executes:

```
#!/usr/bin/bash
rsync -auv --info=progress2 src/${1} ben@vlsi.hmc.edu/src
```

`${1}` is the 1st argument (`foo.sv`)

Python

- Widely used scripting language
- Files end in .py
- **Example:** Create file called hello.py with following contents:

```
print("Hello World!")
```

- Execute script:

```
$ python hello.py
```

or add first line saying which program to use to run it. hello.py contents:

```
#! /usr/bin/python
print("Hello World!")
```

Make it executable, then run it:

```
$ chmod 755 hello.py
$ ./hello.py
```

Many good books/ classes on Python programming.
Python for Everyone is freely available.

C

- Good for system programming because pointers enable access to hardware explicitly.
- GCC pre-installed on most Linux distributions.
- **Example:** Create file called hello.c with following contents:

```
#include <stdio.h>

void main(void) {
  printf("Hello World!\n");
}
```

- **Compile & run:**

```
$ gcc -o hello hello.c
$ ./hello
```

Makefiles

- Facilitate compiling – especially for program with many files.
  - Don't need to manually type in (and remember) flags, etc.
- Format of rules:

```
target : dependencies
                command
```

Example Makefile

```
objects = a.o b.o c.o

example : $(objects)
        gcc -o example $(objects)

a.o: a.c x.h y.h
        gcc -c a.c
b.o: b.c x.h
        gcc -c b.c
c.o: c.c y.h
        gcc -c c.c

clean:
        rm example $(objects)
```

Example Makefile

Run makefile – change to directory:

```
$ make
gcc –c a.c
gcc –c b.c
gcc –c c.c
gcc –o example a.o b.o c.o
```

Touch a file (as if modified it) so that program recompiles:

```
$ touch y.h
$ make
gcc –c a.c
gcc –c c.c
gcc –o example a.o b.o c.o
```

Remove all executables (leave only source files):

```
$ make clean
rm example a.o b.o c.o
```

About these Notes