CS Lab Survival Guide
Computer Science Department
SUNY Potsdam
September 2013

1 CS Lab

1.1 Location and hours

The CS Lab is located in Dunn Hall 358. The Lab is open during building hours, which is generally from 7am to 11pm on weekdays and from 8am to 11pm on weekends.

1.2 Lab access

The CS Lab is open to all students taking CS classes, and all CS majors. Your Ranger Card will allow you access to the lab when the door is closed. Swipe your card in the slot with the stripe facing to the right and wait for the green light to indicate that the door handle is operational.

If you are a CS major taking classes above CIS 300, you also have access to the Dunn 356 High Performance Computing Lab.

You may use a CS Lab workstation even when there is a scheduled lab session in Dunn 358 for CIS 201 or 203, provided that there is an unusued workstation available.

1.3 Lab deportment

Food and/or beverages are not allowed in the vicinity of the Lab workstations, either on the tables or on the floor. Food and/or beverages are allowed on the CS Lab table near the printer.

Keep the CS Lab neat and tidy. Dispose of paper and other recyclables in the blue recycling containers and trash in the wastebasket. The CS Lab is more welcoming to others when it appears well cared for.

Push your chairs under the table when you are done working at a workstation. There should be two chairs at each workstation location; you are responsible for stowing both of them when you are done.

Turn out the lights and close the door when you are last to leave.

Contact the Lab administrator (`fossumtv@potsdam.edu`) if you find anything amiss in the lab such as non-functioning or missing equipment including workstations, monitors, keyboards, mice, chairs, or printers.

Disruptive or abusive behavior will not be tolerated in the lab. If you experience such behavior directed at you or anyone else, contact University Police immediately (phone 2222).

1.4 Logging in

To log in, go to a free Lab workstation and enter your campus computer account (CCA) username and password in the login area of the welcome screen: Enter your username first, followed by the `Enter` key, and then enter your password. Your password will not display on your screen.

Your workstation username and password are the same as your Campus Computer Account (CCA) username and password that you can use to access other computing resources on campus.

If the screen is locked because someone else has left without logging out, please report this to the lab system administrator and find another workstation to use.

1.5 Logging out

Be sure to log out once you are done. Click on the "logout" icon at the top of your screen. If you do not log out, you may prevent someone else from using the workstation later, and you may receive an email reminding you that you have not logged out.

Never power down any of the Lab workstations. Your workstation may be in use by someone who has remotely logged in, and powering down the workstation may result in their work being corrupted or lost.

You may leave your workstation unattended for short periods of time, but do not do so if you expect to be gone for more than half an hour.

Again, **push your chairs under the table when you are done working at a workstation**.

2 Workstation environment

When you log in to one of the Lab workstations, you should see a window environment with a mostly empty screen and small panels on the top and bottom. On the left-hand side of the top panel you should see an Applications menu and icons for a terminal shell and a browser. On the middle of the top panel you should see a Logout icon.

You will normally begin using the workstation by opening a terminal shell. This gives you a *command line environment* where you can enter commands to carry out activities such as managing your home directory; creating, examining, and editing files; and compiling and running your programs. We give some example commands later in this document.

2.1 Your home directory

When you log in, you will be placed in your *home directory*, which has the following path name:

```
/home/student/<your login>
```

Here, `<your login>` is your campus computer account ID, the one that you use to log in to the workstation. Don't confuse *your* home directory with the system-wide directory named `/home` – they are not the same!

Your home directory is stored on the CS Lab server, not on the workstation itself, so you always have access to your same home directory no matter what workstation you are using.

You have private access to the contents of your home directory. You are responsible for keeping your home directory private by not divulging your password to anyone and by maintaining control of your workstation when you are logged in.

In the rest of this document, we will refer to the ficticious username `user000` and the Lab workstation named `algonquin` for the purposes of giving examples.

2.2 Directory structure

A *directory* is an area of physical storage containing files and other directories. On a Microsoft Windows system, a directory is called a `folder`.

When you log in to a Lab workstation, there is a *top-level directory* called the *root* of the directory structure. The symbol used to refer to this top-level directory is '`/`', but we call it "root".

Each directory may contain files (which can be ordinary text, executable code, or other stuff) and other directories, called *subdirectories*. It is entirely possible for a directory to be empty, which is the case when one is initially created.

For example, the home directory of our student `user000` is

```
/home/student/user000
```

The directory `user000` is a subdirectory of `/home/student`, and the directory `student` is a subdirectory of `/home`. The directory `home` is a subdirectory of the root directory `/`.

When you are logged in to a Lab workstation, the symbol '`~`' (called *tilde* or sometimes *wiggle*) refers to your home directory. So if you are logged in to a Lab workstation as a student with username `user000`, the symbol ~ means the same as `/home/student/user000`.

2.3 Directory navigation

When you get a shell (see above), you are automatically put into your home directory. Your shell will print a *command prompt* that appears like this:

```
user0000@algonquin:~$
```

This means that user `user000` is logged in to the `algonquin` Lab workstation and is currently in the user's home directory.

You can always see what directory you are currently in by entering the command `pwd` – which means Print Working Directory:

```
user000@algonquin:~$ pwd
/home/student/user000 <- this is what is printed by the pwd command
user000@algonquin:~$
```

The directory structure forms a *tree*, with root at the top of the tree (You may think that this is upside-down, but it will appear to be OK if you are standing on your head) and with files and subdirectories at the next level down. The subdirectories themselves may have subdirectories, and this progression may go on for many levels.

Your current working directory (the one reported by the `pwd` command) is also called *dot* and has the (appropriate) name '`.`'. The parent of your current working directory – the directory just above it – is called *dot-dot* and has the name '`..`'.

The `cd` (change directory) command changes to the specified directory given on the command line. Here are some annotated examples (given without the `user000@algonquin:` part of the prompt):

```
~$ cd /home/student   <- change to the /home/student directory
/home/student$ cd ..  <- change to the parent of /home/student
/home$ cd .           <- change to the current directory (no change!)
/home$ cd ~           <- change back to your home directory
~$ cd                 <- shorthand for cd ~
~$
```

You can use the `mkdir` command to create a directory (which starts out being empty) in the current working directory:

```
~$ mkdir foo
~$ cd foo
~/foo$ cd ..
~$ cd ~/foo
~/foo$
```

Notice that the last prompt says `~/foo`; this is just a shorthand for `/home/student/user000/foo`.

In summary,

- to move *up* one level in the directory structure, use '`cd ..`'
- to move *down* one level in the directory structure, use '`cd dirname`' where `dirname` is the name of a directory that is located in your current working directory
- to move up two levels, use '`cd ../..`'
- to move to a particular directory given its *relative path name* (relative to your current working directory), use '`cd relpath`', where `relpath` does not start with a leading '`/`'
- to move to a particular directory given its *absolute path name* (starting at the root), use '`cd abspath`', where `abspath` starts with a leading '`/`'

In the above examples, the command

```
~$ cd /home/student
```

used an absolute path name, and

```
~$ cd foo
```

used a relative path name.

Notice that if you start out in the directory `/home/student/user000`, the following command

```
~$ cd ../user000
```

would bring you back to your home directory again. The name `../user000` is a relative path name.

2.4 Your directory organization

You may want to have separate subdirectories of your home directory for the different classes you are taking. In each of these directories, you may want to have subdirectories for individual assignments or for playing around. To get an idea of how you might want to organize your home directory (like organizing your *real* home), take a look at the directory structure in `/home/student/Classes`.

2.5 Class-related directories

Your instructor will let you know where specifically to find material you will need to carry out your class-related assignments on the Lab systems. In particular, you may be required to copy files from a class-specific directory into a suitable working subdirectory of your home directory. Most class-related materials can be found in subdirectories of `/home/student/Classes`. For example, the directory

```
/home/student/Classes/201
```

will contain materials specific to CIS 201. Carefully follow the instructions in your assignment to be sure that you have all the files in your working directory necessary to carry out the assignment.

2.6 Your Desktop

The “desktop” that appears on your screen when you log in shows the contents of a directory named `Desktop` in your home directory. You can change to this directory in a command shell using the following:

```
cd ~/Desktop
```

Everything in this directory will appear as an icon on your desktop screen. Adding, renaming, or deleting files in this directory will result in those changes appearing on your desktop.

You can open a file manager application that will display the contents of other directories in a file manager window. From this window you can “drag and drop” files between the file manager window and your desktop, or even between two file manager windows. Other operations such as “copy and paste” or “delete” are possible on your desktop or in a file manager window. All of these sorts of operations can be done using commands from a shell window, which is what most Linux programmers do.

3 Useful commands

command	example	description
`pwd`	`pwd`	display the current working directory
`cd`	`cd /home`	change directory to `/home`
`rm`	`rm bar`	remove the file `bar`
`rmdir`	`rmdir foo`	remove the directory `foo`
		the directory must be empty for this to work
`mv`	`mv bar car`	rename the file `bar` to `car`
`mv`	`mv bar foo`	move the file `bar` into directory `foo`
		`foo` is assumed to be a directory in the above
`cp`	`cp bar far`	copy the file `bar` onto the file `far`
	`cp ../bar .`	copy the file `../bar` to a file
		of the same name in the current directory
`cat`	`cat bar`	display the contents of the file `bar`
`ls`	`ls foo`	list the contents of directory `foo`
	`ls -l foo`	do a long list (with details) of `foo`
	`ls`	list the contents of the current directory
`lpr`	`lpr bar`	print the file `bar` on the Lab printer
`javac`	`javac Prog.java`	compile `Prog.java`
`java`	`java Prog`	run the `Prog.class` program

3.1 Wildcards

When a command requires one or more filename arguments (such as `javac`), you can use wildcards to refer to all of the filenames that match a particular pattern. The most common wildcard is '`*`' which matches any (zero or more) filename characters. For example, the following command

```
javac *.java
```

will compile all of the Java files in the current directory.

3.2 Man pages

Most commands have on-line documentation that you can read to find out the various *switches* (options) that are allowed for the command. The `man` (for *manual*) program will display this on-line command documentation, as the following example shows:

```
man cat
```

3.3 PATH

Your `PATH` is a colon-separated list of directories that the shell uses to determine the location of a command that you enter. You can see your `PATH` by using the following command:

```
echo $PATH
```

If you want to run a program that is not in your `PATH`, you can do so by naming the program with an absolute or relative path name. For example, if you compile a program `hello.c` using the `gcc` compiler, the compiler will produce an executable named `a.out` in your current directory. But your current directory is almost certainly not in your `PATH`, so to execute this program you would need to do

```
./a.out
```

where the '`./`' part (pronounced "dot-slash") says to look in your current directory for the program to run instead of looking in the directories in your `PATH`.

4 Redirection

Many programs take input from *standard input* and send their output to *standard output*. Unless directed otherwise, standard input comes from your keyboard, and standard output goes to your shell window. In Java programs, `System.in` refers to standard input and `System.out` refers to standard output.

Programs may also send their output to *standard error*, which also defaults to your shell window but is used principally to report error messages. In Java programs, `System.err` refers to standard error.

When running a program that writes to standard output, you can *redirect* the output to go to a file instead using the '>' redirection operator: For example, the following command

```
ls > DIR
```

would send the output of the `ls` command to the file `DIR` instead of sending it to your shell window. Similarly, if `Prog` is a Java program that reads from `System.in`, the following command

```
java Prog < DIR
```

would read its input from the file `DIR` instead of from the keyboard.

5 Pipes

A *pipeline* is a concatenation of commands where the output of one command becomes the input to the next command. In this case we say that the one command *pipes* its output to the next command. The vertical bar operator '|' is used to denote a pipe. In the following example

```
ls | java Prog
```

the output of the `ls` command becomes the input to the Java program `Prog`. Notice that this amounts to the same as the two commands

```
ls > DIR
java Prog < DIR
```

without the need for the intermediate file `DIR`.

6 Remote login

You can log in to the CS Lab from off campus using a *remote shell*. On a Microsoft Windows system, you can install and use `putty`. However, logging in remotely from a Windows system will generally not allow you to execute programs on the Lab systems that are graphics-oriented, such as `gedit` or `emacs`.

On a Linux system such as Ubuntu, you can use `ssh` to log in remotely. Doing so will give you a command shell that is the same as if you were running the shell on the lab system. If you give the '`-X`' command-line switch to the `ssh` command, this will route graphics-oriented output from the Lab system to your remote system. If you do not have a Linux-based system, you can install `virtualbox` to set up a Linux virtual machine that will behave appropriately.

To use `ssh` from a remote Linux system, run the following command:

```
ssh -X user000@lab.cs.potsdam.edu
```

where you should replace the `user000` with your username. You will be asked to enter your CCA password, after which you will get a shell on one of the Lab workstations that you can use as if you were sitting at the Lab workstation.

If you want simply to copy a file between a remote Linux system and the Lab server, you can use the `scp` (for "secure copy") command. Here's an example of how to use `scp` from your remote Linux system:

```
scp user000@lab.cs.potsdam.edu:foo .
```

This will copy the file named `foo` from your Lab home directory into the file with the same name on your remote Linux system. You can also copy in the other direction, from your remote Linux system to your Lab home directory; see the `scp` documentation for details.

If you do your work in the Lab instead of remotely, you have the advantage of being able to discuss your homework with your peers and faculty who are on campus, and you will also get to know your fellow students better. One of the best ways to become a successful CS student is to become a part of our CS family by working and playing in our CS Lab environment. We recommend that you use remote login rarely.

7 USB drives

You should be able to plug a USB device into one of the USB slots in the front of a Lab workstation to access the files in the device. You will typically get a file manager window showing the contents of your device, and you should be able to copy files and directories between the file manager window and your desktop by using conventional drag-and-drop or copy-and-paste operations that are familiar to Windows users.

The directory structure of each USB device that you plug in will appear as a subdirectory of the `/media` directory, with a name that comes from the device "disk label". You might see, for example, a directory named

```
/media/XXX
```

if the USB you plug in has `XXX` as its disk label. You can cd to such a directory to perform operations on the device as you would any other directory. If you leave your file manager window open during the use of the device, you can "Eject" the device from this file manager window, which will commit any changes to the device that are pending. Unplugging a USB device without ejecting it can result in device corruption, although this is rare if you have not been accessing the device for a few minutes.

A particular USB device may have a filesystem that is formatted in such a way that a Lab system doesn't understand its organization. In this case, the Lab system may not be able to give you access to the content of the device. If this happens, and if you need to access the contents of the device, you may get help from the system administrator.

8 Editors

To modify a file such as a Java source file, a document, or a test file, you need to use an *editor*. An editor is a program that allows you to make changes to a file.

Everyone has their favorite editor, and we will not play favorites here. However, one of the following editors (listed in alphabetical order) should serve you well:

`emacs` (pronounced "ee-macks")
`gedit` (pronounced "gee-edit")
`jedit` (pronounced "jay-edit")
`nano`
`vi` (pronounced "vee-eye")

9 Miscellany

The names of the CS Lab systems come from the Adirondack High Peaks. Mount Marcy, the tallest, has an elevation of 5,343 feet. The artwork on the walls of the Lab are on loan from the Gibson Art Gallery and are intended to be suggestive of the Adirondack mountain region.

The CS Lab workstations and server were replaced in Summer 2013. Each workstation is a Hewlett-Packard Z620 with a Xeon processor (3+GHz), 12Gb memory, 500 GB disk space, and an Nvidia GTX480 GPU graphics card. The Lab server is a Hewlett-Packard rack server with 1.5TB storage, dual Xeon processors, and 32Gb memory.

Home directories on the Lab server are scheduled daily for backup, at two separate locations, in the wee hours of the morning. If you have files or directories that you have mistakenly overwritten or deleted, you can request that the system administrator restore your backed up versions. Any changes that were made after the backup will be lost. Even though backups are scheduled daily, their successes are dependent on factors such as network connectivity, power availability, and hardware reliability; *the system administrator cannot guarantee that files will be restored to their prior contents.*

The CS Lab system administrator is Dr. Fossum (fossumtv@potsdam.edu).

10 This Document

The LaTeXsource file for this document is named `survival.tex` and is located in the directory

```
/home/student/Classes/Survival
```

on the CS lab server `cs.potsdam.edu`. The PDF version of this document is

```
/home/student/Classes/Survival/survival.pdf
```