

## CS 2213, Spring 2009

### Assignment 1: The Linux Command-Line Environment

Due: 5pm, Friday, January 30, 2009 (email 1/27, see below)

**Goal:** The purpose of this assignment is to familiarize you with the command-line Linux environment that you will be using this class.

This assignment covers only the most basic usages of Linux/UNIX command-line. For more information, please read the appropriate sections of Sobell. There is also an online UNIX tutorial from the Canadian Research Council at <http://rcsg-gsir.imsb-dsgi.nrc-cnrc.gc.ca/documents/basic/node1.html> that might be useful.

**Learning objectives:** be able to login to the CS department machines; be able to execute programs from a bash or tcsh shell; be able to use basic UNIX commands to navigate the file-system (pwd, cd, ls), manage files and directories (cp, mv, rm, mkdir, rmdir), and view the contents of text files (cat); be able to use vi to edit text files; be able to use svn to download updates from a server repository (svn update), and to record your additions/modifications in the server repository (svn update, svn status, svn add, svn commit); be able to use gcc to compile single source file C programs.

**Other objectives:** setup account password; prepare to receive email on your CS account; checkout the svn repository used in this course.

**Directions:** Please read the entire assignment, and then follow the steps enumerated below.

You will need to turn in the answers to the bolded questions in the text file `assgn1/questions.txt`. A template for `questions.txt` will be retrieved from subversion as part of the section on subversion. Instructions on how to format/submit the `questions.txt` are found in the final section. It is suggested that you simply record your answers to the bolded questions on a piece of paper until you get to the final section of this assignment.

Except as noted with regard to the bonus question, you are allowed to use outside resources and get help from your classmates and other individuals without penalty *on this assignment* as long as the help received is (1) completely disclosed in the README file and (2) does not involve directly copying answers without doing the prescribed activities or understanding what those answers mean.

**Login information.** Every registered CS student is given a Windows XP and a UNIX account. The usernames are the same,<sup>1</sup> and they share space (users' UNIX home directory is available as Z: under windows).

#### **What is your username?**

---

<sup>1</sup>Usually, first initial + last name, truncated to 8 characters. If there's already an account with that name, the username will be up to 7 letters of the first name and the first letter of last name. The username's of new users should be posted in the labs. See also: <http://www.cs.utsa.edu/~javalab/lab/accountLogin.html>

1. Use one of the machines in the Main Lab or one of the department's teaching laboratories on the third floor of the science building.
2. If the machine is currently booted in Windows XP, shut it down. As you restart the computer, select Ubuntu from the Grub boot menu.
3. Login using your department username.
  - (a) The UNIX and Windows XP account passwords are not synchronized, so you actually have two different passwords. It is recommended that you set them both to the same thing. See <http://www.cs.utsa.edu/~javalab/lab/accountLogin.html>
  - (b) If you have not previously logged in to one of the department UNIX hosts (Solaris or Linux) and changed your password, your password will be the first 8 digits of your student id number.
4. Once you have logged in, open up a terminal. In the GNOME desktop environment (which is the default graphical user interface for Ubuntu) this can be accomplished by selecting the program "Terminal" under the Applications/Accessories drop-down menu.

The terminal will contain a prompt at which you can type commands.
5. If you have not previously set your UNIX password, enter the command `passwd` followed by an enter into the Terminal and follow the prompts for entering your old and new passwords.

**Directory Navigation.** One important concept in Linux/UNIX is the file-system hierarchy, which organizes the permanent storage of a UNIX system. (For more information, see Sobell, Chapter 4.) There are two items in the UNIX file-system: directories (a.k.a., folders) and files. Files contain arbitrary streams of text or binary data. Files are named through entries in directories. Directories may also contain other directories. The outermost directory, known as the root director (written `/`) is the only directory not contained in another directory. Thus, the directory hierarchy consists of a tree rooted at `/`.<sup>2</sup>

1. The prompt you see when you first open up terminal is called the UNIX shell. There are actually a couple different shells that have slightly different features and operate in different ways. The machines in our department normally default to `tsh`, a descendant of `cs` (the C shell); another common shell is Bourne shell (`sh`) and its descendant `bash`. Both of these are described in Sobell (Chapters 8 and 9). For our purposes, it shouldn't matter which one you are using.

Part of the shell's state is the current 'working directory' (a.k.a., current directory), which is where many commands default to operating. Print the working directory of your shell by typing the command `pwd` (which stands

---

<sup>2</sup>As a side note, UNIX users do not usually need to deal with individual volumes or drive letters as in Windows. The system administrator can just mount a disk partition or network share as a replacement for some subdirectory within the single unified hierarchy.

for print working directory), followed by an enter to find out what directory you are in. When you first login, you will normally be placed in your home directory. (Normally, in a UNIX or Linux system, each user has a home directory, which is a place for that user to place their personal files. In the CS Department at UTSA, this directory is usually exported to Windows XP as your Z:.)

Notice that the directory printed out by `pwd` starts with a `/` (slash); this means that it is an *absolute path* from the root directory. Other slashes separate each containing directory (parent directory) from the directory or file that it contains (child directory). For example, `/usr/local/bin` refers to the directory named `bin` inside the directory named `local` inside the directory named `usr` inside the root directory.<sup>3</sup>

**What is the absolute path of your home directory, as reported by `pwd`?  
What is the absolute path of the parent directory of your home directory?**

`~` and `~username` can also be used as aliases for the current user's home directory and *username's* home directory, respectively. They can be used by themselves or as part of larger path names any place where the absolute path to the home directory would otherwise be used.

2. Two other important UNIX shell commands are `cd` and `ls`: You can change the shell's current working directory with the command `cd new-directory`. You can list the contents of the current directory using the command `ls`.

Change to the directory `/var/log` and list its contents. **What commands did you use to accomplish this?**

3. **BONUS. Most UNIX commands are just external programs in `/bin`, `/usr/bin`, etc., but `cd` is not. Instead it is part of the shell program. Can you think of a reason why `cd` is built in to the shell? Answer this question without googling for the answer or referencing other materials not provided by the instructor.**

4. It is also possible to specify a directory<sup>4</sup> with `ls` to list the contents of that directory without needing to change the current working directory: `ls directory`.

List the contents of `/bin` without changing your current directory.

**What command did you use?**

5. It is not necessary to specify directories for `cd`, `ls`, and other programs using absolute paths. If the path doesn't start with a slash, it is interpreted as being relative to the current directory, so for example, if you are still in `/var/log`, you could list the contents of `/var/log/news` using the command `ls news`. Paths can also refer to the parent directory of a directory using `..`, so from the current directory of `/var/log`, you can use `ls ..` to view the contents of `/var`.

---

<sup>3</sup>Technically, the names of files and directories are stored in the containing directory along with a pointer to something called an inode that contains other information (meta-data) about the file along with the location of the files contents (data).

<sup>4</sup>actually, a list of files and directories

From the directory `/var/log`, list the contents of `/var/run` using a relative path. **What command did you use?**

6. What happens if you try to pass `ls` a non-existent name, such as `ls /bin/transmogrify`?

**Managing files and directories.** The following commands can be used to copy, move/rename, remove, or create files and directories.

Command	Meaning
<code>cp old-file-name<sup>5</sup> new-file-name</code>	copies the contents of the file <i>old-file-name</i> to a new file named <i>new-file-name</i>
<code>cp list-of-files<sup>6</sup> destination-directory</code>	copies each of the files in <i>list-of-files</i> to the <i>destination-directory</i>
<code>mv old-file-name new-file-name</code>	renames the file <i>old-file-name</i> to <i>new-file-name</i> (which can be in a different directory from <i>old-file-name</i> )
<code>mv old-directory-name new-directory-name</code>	renames the directory <i>old-directory-name</i> to <i>new-directory-name</i>
<code>mv list-of-files existing-directory</code>	moves each of the files in <i>list-of-files</i> to some <i>existing-directory</i>
<code>rm list-of-files</code>	removes (deletes) <sup>7</sup> each of the files in <i>list-of-files</i>
<code>mkdir new-directory</code>	creates a new directory called <i>new-directory</i> , which must be in a previously existing directory, but must not previously exist
<code>rmdir empty-directory-name</code>	removes an empty directory called <i>empty-directory-name</i>

1. Change to your home directory (a.k.a., `~`).
2. Copy `/usr/bin/man` to the current directory (which can be represented using a single dot: `.`)? **What command did you use?**
3. Rename `~/man` to `~/my-man` without using dot `.`, slash `/`, or tilde `~`. **What command did you use?**
4. Remove `my-man` from you home directory? **What command did you use?**
5. Create a subdirectory within your home directory called `courses`. **What command did you use?**
6. In order to make sure, you classmates can't look at your course files, please run the command: `chmod g-rwx,o-rwx ~/courses`

<sup>5</sup> The filenames/directory names used in these commands may be specified using relative or absolute paths. Most often, at least one of the arguments is in the current directory and specified just as a file or directory name without any slashes; this is just a special case of a relative path.

<sup>6</sup> For all of the commands that take lists of files or directories; it is permissible (and quite common) to just give a list consisting of only one file or directory. If there are more than one file or one directory in the list, they should be separated with spaces.

<sup>7</sup> UNIX/Linux `rm` does not normally implement any kind of Trash Basket or Recycle Bin like most graphical user interfaces (GUI) do. Once you remove a file with `rm`, there is no guaranteed way to recover it.

**Viewing and Editing Text Files.** Most files in UNIX are text files, which contain lines of readable characters and white-spaces characters separated by newlines '\n'. Many UNIX/Linux command operate on text files, including cat and more. vi can be used to edit text files.

1. The command, `cat file-list`, can be used to display the contents of one or more files. Use `cat` to look at `/etc/issue`.

**What is the contents of the file /etc/issue?**

2. Some files, however, take up more than one page, and are inconvenient to view with `cat`. For example, try to view `/etc/services` with `cat`. This file more than fills up one screen.

The UNIX utility `more`<sup>8</sup> provides a solution to this problem, by displaying only one screen-full at a time, and waiting until you press the space bar to see one "more," screen-full. In addition, one can press `enter` to see one more line at a time, and 'b' to go up one screen-full at a time.

Use `more` to view `/etc/services` and find out which service operates on port 22. **Which service is this?**

3. You can create and edit text-files using the editor `vi`.<sup>9</sup> `vi` is a bit of an acquired taste. At first it is quite difficult to use, because (1) it has a few distinct modes that operate in different ways, and (2) you have to memorize the commands to get it to work.

Run the program `vimtutor` to learn how to navigate `vi`. You may also find it helpful to browse through the tutorial at <http://rcsg-gsir.imsb-dsgi.nrc-cnrc.gc.ca/documents/basic/node166.html> which has a fairly concise listing of some of the most important `vi` commands, or to read Chapter 6 in Sobell.

**Email.** Your UNIX account is capable of receiving email at the address `username@cs.utsa.edu`. An email will be sent to all students in this class on **January 27**. Reply to this email by January 30, 2009. This email can be read through a web mail system <http://mail.cs.utsa.edu>; connecting your favorite email client to the department IMAP server `imaps://mail.cs.utsa.edu`; using a text-mode client, such as `mutt`; or by forwarding future email to a different email account (see the instructions for using `vi` to create a `.forward` file below).

Now that you know how to use `vi`, you can use it to setup your CS department email to be forwarded to another account.

1. In your home directory, use the command `vi .forward` to launch the `vi` editor with a blank document buffer that you will save to `.forward`.
  - (a) Insert a single line that contains the email address of the account you normally use for email (e.g., `xyz123@myutsa.edu`).

---

<sup>8</sup>The utility `less`, which is available on Linux systems, also does about the same things, but let's you go backwards in some situations `more` does not.

<sup>9</sup>Actually, our department machines default to an enhanced variant of `vi` called `vim`.

- (b) Return to normal mode, and issue the command to write the buffer to the file.
  - (c) Issue the command to quit vi.
2. Use cat to view `~/ .forward` to make sure it contains only a single line, which contains your preferred email account.
  3. Test your `.forward` file by using your favorite email program to send an `username@cs.utsa.edu`.
  4. If you would in fact, prefer to use the CS email account directly using web-mail, IMAP, or a UNIX text-mode email application, and know how to do so, you can remove the forwarding by deleting the `.forward` file you just created.
  5. *Please complete this portion of the assignment by 1/27*, so that you can receive and reply to an email that will be sent out on that date.

**Using man pages.** Most UNIX/Linux commands have online documentation called man pages. These can be accessed by using the command: `man command` Which will format the appropriate manual page into an enhanced text file paged by `more`.<sup>10</sup>

Read the man page for `cat`. **What is one thing `cat` can do, which was not mentioned above?**

**Using subversion.** Subversion is a version control system. It is usually used by teams of programmers who are working together on large software projects. It helps manage the changes made by individual members of the team, by keeping a record of old software snapshots ‘versions’ in a centralized repository and helping programmers merge their local versions into the central versions.

For this class, each student has been set up with a subversion repository, in which they are to store (commit) copies of their work on the assignments. Each student has access only to their repository. The instructor and TA will also have access to view and edit the work in all of the repositories.

The repository can/will be used in the following ways:

- The instructor/TA will use the subversion repository to include provided code for assignments.
- Students can commit to the repository as many times as they want. In addition to the current version, subversion will continue to keep track of all of the previously committed, and students can use subversion to compare with and revert back to old versions of their work.
- Students can commit to and update from the repository on multiple machines (e.g., at UTSA and at home), and use subversion to help move/synchronize their work between the different machines.

---

<sup>10</sup>or less

- The TA/instructor will use subversion to access the most recently committed version of each student's assignment as of 5pm on the due date for that assignment. This version will be graded. (Students can continue to commit additional changes after the deadline, but these will be ignored during grading.)

Subversion works mostly the same as an older system called Concurrent Versions System (CVS), so most of what Sobell says about CVS on page 420 ff. applies to Subversion (just replace `cv`s with `sv`n in the commands. There's also an online book at <http://svnbook.red-bean.com/>. Here we will cover just the minimum needed to use `sv`n to submit your assignments in this class. Further reading in `sv`nbook or Sobell is recommended.

The repositories that will be used in this class are stored on the machine `nougat.cs.utsa.edu` and accessed using your department UNIX account names and passwords. The URL for each student's repository is: `https://nougat.cs.utsa.edu/cs2213/cs2213-username`

1. The first step in using subversion is to `cd` into your `~/courses` directory and check out a working copy of your repository using the command: `svn co https://nougat.cs.utsa.edu/cs2213/cs2213-username`

This will download the current version and setup a working copy of your repository `~/courses/cs2213-username`.

2. Use `ls` to see what files are in your working copy.
3. Now, make some modifications in the working directory as follows:
  - (a) Change into `~/courses/cs2213-username/assgn1` directory.
  - (b) Use `vi` to edit the file `questions.txt` and put your name at the top.
  - (c) Use `vi` to create a new text file called `README` containing your name.
4. Now, that there have been changes made to the local copy it is necessary to commit them before they are uploaded to the repository and become visible to other users (i.e., the instructor and TA). It is recommended, that you do this before you leave the computer each time you've been working on your assignment.

Until one becomes an `sv`n expert, the following procedure is recommended. Use it to commit your changes to `questions.txt` and `README`.

- (a) Use `sv`n `status` to get a list of any files or directories that you've created or modified, since the last time you did a commit. Files that are the same in the working directory and the server repository will not be listed.
- (b) If `sv`n `status` puts a `?` before a file, that file is 'unknown' to subversion and needs to be 'added' before it can be committed to the repository. This can be accomplished with the command: `sv`n `add list-of-files`

- (c) Use the command `svn update` to synchronize your local copy with the latest changes in the repository. (That is, download any new updates from the server and merge them into your local copy.) Normally, this should ‘just work,’ but if `svn` reports a conflict, please see <http://svnbook.red-bean.com/en/1.5/svn.tour.cycle.html#svn.tour.cycle.resolve>.
- (d) Use `svn commit` to actually commit your current working version to the server repository.
  - As part of the commit process, an editor will be opened and you will be given an opportunity to write a sentence or two about what changes you are committing. You can later view these as a log of what changes you’ve made at what times.

**Compiling with gcc.** GCC can be used to compile a C source code into an executable program using the following command: `gcc -o program-name list-of-source-files`. The program can then be executed with the command `./program-name`.

1. A program `hello.c` has been provided in the `assgn1` directory of your `svn` repository.
  - (a) `cat` it.
  - (b) Compile it to produce an executable `hello` in `assgn1`.
  - (c) Run the compiled program with the command `./hello`. **What does the program output?**
  - (d) Use `vi` to modify the program, so that the output is customized with your name, so that, for example, if your name were John Smith, you’d modify the program to output: `Hello, John Smith!`
  - (e) Recompile and rerun the program. If there are errors, fix them, and repeat as necessary.
  - (f) Commit your modified `hello.c` to the subversion repository for grading using the steps described above. The program file `hello` does not need to be added to `svn`. In general, it is desirable to have `svn` manage only those files created or edited by humans (i.e., source files and documentation), but not the files produced by the compiler or automatically by other tools from those files. Your `svn` repositories should already be set up to ignore the files `assgn1/hello` and `assgn1/fix-me`.
2. Another program `fix-me.c` has been provided in the `assgn1` directory of your subversion repository.
  - (a) Use `gcc` to try to compile `fix-me.c` into the executable program `fix-me`. What happens? **What error message does gcc report?**  
There are two errors: one on line 8 and one on line 11. **How can you tell this from gcc’s output?**
  - (b) Fix the first error by deleting the third `+` in line 8.

- (c) Fix the second error by adding a semicolon at the end of line 11.
- (d) Recompile `fix-me.c` into `fix-me`.
- (e) Test the program by running it and typing several characters followed by a `ctrl-d`. The program should tell you how many characters you typed.
- (f) Commit your changes to the repository.

### Answering the questions and committing.

1. Use `vi` to edit `question.txt`.
  - (a) Make sure your name is at the top of `questions.txt`
  - (b) Insert your answers to the bolded questions in this document below the corresponding question in `questions.txt`.
  - (c) Put two blank lines before each question, and one blank line between your question and your answer.
2. Add any notes, you wish the TA/instructor to consider while grading at the top of the text-file `README`.
3. In addition, below any notes, and above your name, please answer the following two questions in your `README` file:
  - (a) Did you use any materials other than those provided by or referenced by the instructor or TA in completing this assignment? If so, please cite the source (including its URL if it is online) and the nature and extent of your use of that material. (For example: I read <http://www.ee.surrey.ac.uk/Teaching/Unix/> to help understand how changing directories works. OR: I copied the answer to question 7 from <http://www.pixelbeat.org/docs/env.html>.)
  - (b) Did you get help in completing this assignment from anybody other than the instructor or TA? If so, please describe the nature and extent of the help received.
4. At the end of the `README`, in the line below your name, place the current date and time.
5. Use `svn` to commit your final submission by 5pm on the due date.
  - (a) Use `ls` and `svn status` to make sure that you have all of the required files and they've all been added to `svn`. The expected contents of `assgn1` directory in the repository includes: `assignment1.pdf`, `README`, `questions.txt`, `hello.c`, and `fix-me.c`
  - (b) You can continue to make changes, and re-commit as often as you want. The last version committed before 5pm on the due date will be graded.