CS 241 Data Organization using C
Lab 1: Introduction to C, ASCII Art, and the Linux Command Line Environment

Fall 2014

```
  ,ad8888ba,   ad88888ba   ad888888b,      ,d8     88
 d8"'    `"8b d8"     "8b d8"     "88    ,d888   ,d88
d8'           Y8,                 a8P  ,d8" 88 888888
88            `Y8aaaaa,         ,d8P" ,d8"   88     88
88              `""""""8b,    a8P"  ,d8"     88     88
Y8,                   `8b  a8P'    8888888888888   88
 Y8a.    .a8P Y8a     a8P d8"                88     88
  `"Y8888Y"'   "Y88888P"  88888888888       88     88
```

1 Brief Introduction to Command Line Linux

This lab is designed for you to have a chance to familiarize yourself with the computing environment we will be using this semester, and in a number of upper level CS classes.

2 Logging in to Linux

Use SSH to connect to a CS machine.

3 Running some basic Linux Commands

Hopefully now you are logged in, and on your screen you have and empty window that has a text line in it that looks something like this:

```
bchenoweth@prospero:~$
```

This line is called the **command prompt** or simply **prompt**. In Linux, the equivalant to "My Documents" of Windows is called your **home directory**. When you first start out, your home directory will contain a number of default files and folders. To see what is present in your home directory, enter the command "ls" (without the quotes) on the command prompt and hit return. The "ls" command has a number of options:

ls -F appends a "*" to the end of every executable file, a "/" to the end of every folder, an "@" to end of every symbolic link, and a few other suffixes for other special file types.

ls -l creates a more verbose listing of the files and folders in the directory that includes permissions, size, date modified, the user name of the owner, and some other information.

ls -a includes hidden files: that is files with names whose first character is ".". These generally include various configuration files. Some of these files are automatically executed during the login process. Some are executed when a particular application starts. One file of particular note is the **.bashrc** file. This file is a plain text, batch file that runs automatically run during login. Unless you know what you are doing, you should not change anything in this file; however, it is generally safe to add new commands to the end of the file. For example, many people append the Linux command "`alias ls='ls -F'`" to the end of their .bashrc. This creates an alias for ls so that whenever "ls" is entered as a command, Linux will replace it with "ls -F".

ls -laF applies all three of the above effects.

Find out more about **ls** by reading the *manual pages*. The command "**man ls**" at the command prompt will display the manual pages (often called *man pages*).

4 Creating Directories

Create a new folder (called directory in Linux) with the **mkdir** command. For example, if you want to create a folder called **foo101** enter the command: **mkdir foo101**.

5 Navigating Directories

Next up, well try to descend into the newly created directory. This can be done by using the **cd** (change directory) command: **cd foo101**.

Now that the active command location has entered the newly created directory, enter the **ls -aF** command, and you will see that the directory is empty except for two directories: "./" and "../". The first of these is a pointer to the current directory and the second is a pointer to the directory one level up. Thus, "**cd ..**" will return the active command location back to your home directory. Additionally, no matter where you are in a directory structure, you can always return to your home directory with the command "**cd** ".

6 Editing text files

You can use emacs or vim, whichever you prefer. (You can even use one of the other editors available in Linux, but those are the two big ones.[1])

[1] The source of many wars among old-school hackers.

7 Copying, Renaming, Moving, and Removing Files

The Linux commands to copy, move and delete (remove) a file are:

- cp *existingFileName newfileName*
- mv *existingFileName newfileName*
- rm *filename*

For example: **cp helloWorld.c myBackupCopy.c** will create a copy of **helloWorld.c** with the name **myBackupCopy.c**.

8 Setting File and Directory Permissions

The **chmod** command (abbreviated from change mode) is a shell command used to change file system permissions (also called file mode bits) of files and directories.

The **ls -l** command will show a file's permissions in the following format:

The initial "-" will be a "d" if the file is a directory.

The syntax of the chmod command is:

chmod <*who*><+|-><*permission*>*file*

who	**u** for user, **g** for group, **o** for other, or **a** for all
+	add permission
-	remove permission
permission	**r** for read, **w** for write, or **x** for execute

For example, in order to make all files in a directory unreadable by anyone other than you, use the commands:

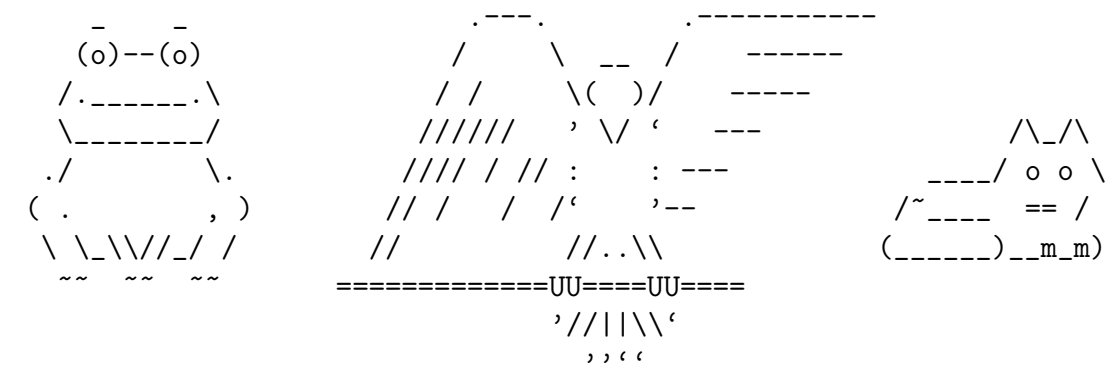
```
chmod o-r *
chmod g-r *
```

Do take care when changing file permissions. You don't want to accidentally make your home directory writable by the entire world!

9 ASCII Art

ASCII art is a graphic design technique that consists of pictures pieced together from the 95 printable (from a total of 128) characters defined by the ASCII Standard from 1963 and ASCII compliant character sets. ASCII art can be created with any text editor. Most examples of ASCII art require a fixed-width font (non-proportional fonts, as on a traditional typewriter) such as Courier for presentation.

Examples:



10 Problem Specification

Use the SSH application to connect to **moons.cs.unm.edu** or **trucks.cs.unm.edu** with your CS account.

1. Create a new working directory called **cs241**, and within that directory create another directory called **lab-01**.

2. Use the Linux shell **chmod** command to ensure that only the owner has read, write, and execute permissions for this new directory.

3. Within this directory, you are to create a C program, stored in the file **asciiArt.c**.

4. Your **asciiArt.c** must use the **printf** function to display to the standard output stream an ASCII art image that fits within 60x24 columns and rows. It is ok to be smaller than that size. Your image must be a stylized version of your first or last name. Each "letter" must be at least 3x3 characters: `printf("Brooke\n")` would NOT get credit as an ASCII art image of my name.

5. The first five lines of your **asciiArt.c** must be of the form:

```
/************************************************/
/* YourFirstName YourLastName                   */
/* Date                                         */
/* CS-241 Section #                             */
/************************************************/
```

6. Use the Linux shell **chmod** command to ensure that only the owner has read and write permissions for your **asciiArt.c**.

7. Compile the program with **/usr/bin/gcc**.

8. Compile and run your modified program with the output redirected from the shell to a text file called **yourfirstname-yourlastname.txt**. This is done using a shell redirection "**>**" command:

   ./a.out >yourfirstname-yourlastname.txt

9. Use a text editor to edit **yourfirstname-yourlastname.txt**. To the beginning of the file, add the following items:

   (a) Your name

   (b) Your major

   (c) Your cs login name

   (d) Why you enrolled in this class (this can span more than one line)

10. From the directory you created in step 1, run the Linux command:

    pwd >out1.txt

    This will create a file that shows the directory path you created. It must be the correct directory path on **cs.unm.edu** of your CS account home directory with the directory created in step 1.

11. From the directory you created in step 1, run the Linux command:

    ls -a -l >out2.txt

    This will create a file, **out2**, containing a listing of the current directory (.), the parent directory (..), your **asciiArt.c** file, the executable file **a.out**, and whatever other files you have in the directory. The **out2.txt** file you created must include the permissions, owner, group, size and date of each file. The permissions of the current directory and your C program must only be read/write (and execute for the directory) for the owner.

11 Turning in your assignment

You are now done with your assignment, and need to attach the files **asciiArt.c**, **yourfirstname-yourlastname.txt**, **out1.txt**, and **out2.txt** (not a.out) into Lab 1 assignment in UNM Learn.

12 Grading Rubric (total of 20 points)

[1 point]: The program starts with the specified comments. Note: the number of '*' can be any number less than or equal to 78 and large enough so that all the terminating '*/' are aligned.

[2 points]: The four files are named as specified.

[2 points]: The program compiles without errors or warnings on **moons.cs.unm.edu** using **/usr/bin/gcc** with no options.

[2 points]: All code in each block is indented exactly two spaces per block structure level. *Spaces* must be used for indenting, not *tabs*. Note: in this simple program, most likely your only block will be the **main** function.

[1 point]: No characters except spaces are on the same line as any open, {, or close, }, curly brackets.

[5 points]: When compiled and run, the program outputs a beautifully stylized, and readable, ASCII art version of your name.

[3 points]: The directory structure shown in **out1.txt** is as specified.

[4 points]: The file permissions of **asciiArt.c** and its parent directory are shown in **out2.txt** and are as specified.

```
 ____                  _
|  _ \                | |
| |_) |_ __ ___   ___ | | _____
|  _ <| '__/ _ \ / _ \| |/ / _ \
| |_) | | | (_) | (_) |   <  __/
|____/|_|  \___/ \___/|_|\_\___|
```