Name: _____

Lab 1 Introduction to UNIX and C

This first lab is meant to be an introduction to computer environments we will be using this term. You must have a Pitt username to complete this lab.

NOTE: Text in UNIX is case-sensitive. CS449 is different from cs449 is different from Cs449. All filenames of concern in this lab are lowercase.

Please follow the instructions as listed in this document.

Here are a few common UNIX commands:

- cd – change directory
- ls – list all files in current directory
- pwd – display the current directory
- mv – move or rename a file
- cp - copy a file
- gcc – compile a c file
- nano – edit a text file

Part I

1) To login to the computers, you will need to use an SSH client. The SSH client that we will be using is **Putty** (at home, download from http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html)

We will connect to the machine (host) **thoth.cs.pitt.edu**

(or **unixs.cssd.pitt.edu** until you have access to thoth).

1) When you login first, you are placed in your home directory. The command

```
ls
```

will **LiSt** all of the files and directories there. The one we are most concerned with is the *private* directory. It is special in that only you can access files inside this directory. It will keep your work safe from other people.

3) Let's move into the *private* directory so we can work there:

```
cd private
```

Changes **D**irectory to the *private* directory
4) For this class, we'll keep all of our files organized into a *cs449* directory. Make it by typing:

```
mkdir cs449
```

If you want to double check that it worked, type `ls` to list.

5) We now want to move into the *cs449* directory to do our actual work.

```
cd cs449
```

Part II.

1) While still in the *cs449* directory type:

```
mkdir lab1
cd lab1
```

to make a directory for today's lab. Now type:

```
nano lab1.c
```

`nano` is a very simple text editor, a lot like Notepad on windows. It is one option for creating and editing code under UNIX/Linux.

2) Type the following text in exactly as it is shown:

```
#include <stdio.h>

int main() {
     printf("Hello World!\n");
     return 0;
}
```

3) Save the file by hitting **Ctrl + O** and then enter. Exit `nano` by typing **Ctrl + X**. At the bottom of the `nano` window, it shows what keys do special things. The ^ means to hold **Ctrl** while pressing the key

4) Back at the prompt type:

```
gcc -o lab1 lab1.c
```

which will make our program. A file named `lab1` will be in the directory if we type `ls`

5) Run it by typing:

```
./lab1
```

Part III: Archives and Project Submission

Whenever you turn in a project for this course, you will need to submit a copy of your code and executable to be graded. We will try this now.

You are probably familiar with a ZIP file, which does two things: it *archives* a bunch of files into a single file, and also *compresses* it to save space. In UNIX, we do this in two steps, we create a Tape Archive (tar) and then compress it (gzip).

First, let us go back up to our cs449 directory:

```
cd ..
```

Now, let us first make the archive. Type your username for the USERNAME part of the filename:

```
tar cvf USERNAME_lab1.tar lab1
```

And then we can compress it:

```
gzip USERNAME_lab1.tar
```

Which will produce a USERNAME_lab1.tar.gz file. We will then copy that file to the submission directory:

```
cp USERNAME_lab1.tar.gz ~aus4/submit/449
```

Once a file is copied into that directory, you cannot change it, rename it, or delete it. If you make a mistake, resubmit a new file with a new name, being sure to include your username.

Part IV: Manual Pages

If you ever want to see how a command works or you forget the various options you could use, you can consult the "man pages" on the command by typing:

```
man COMMAND
```

for example:

```
man ls
```

This will let you scroll through the online help about the `ls` command. The **SPACE BAR** will scroll the document one screenful at a time and the **ENTER** key will move one line at a time. At any time you can quit by pressing "**q**"

In the space below, explain the purpose of the -S switch (capital S, not lowercase.)

1.)

Part V: Recording your work with `script`

Sometimes something will go awry in your program and you may not know the source of an error message from the compiler. For us to help you, we need to see the output of the compiler. We can capture that with a program called `script`.

Type:

```
man script
```

now to read up on how it works.

Now open the `lab1.c` file from part I by navigating to it and then opening it in `nano`. Remove the semicolon after the printf() statement and save.

Now issue the `script` command.

```
script
```

Then compile the modified program.

```
gcc lab1.c
```

This should result in an error message.

Type `exit` or hit **CTRL** + **D** to stop `script` from recording. If you do an `ls` now, you should see a file named typescript.

We can use the program called `more` to display the contents of this file. `more` is also used on the man pages, and can be operated in the same way. If the file is longer than the screen, it will allow you to scroll or to quit at anytime. Type:

```
more typescript
```

to see the output you saw when you compiled the first lab.

Helpful Hints

- Keep this sheet as a guide until you get comfortable in UNIX
- Every time you want to write a program you will do the following after logging into your account:

```
cd private
cd cs449
```

- If you want to use vim or emacs instead of nano, that is fine. You may prefer to edit files under a GUI and use ftp to upload files, the choice is yours, however we will only officially support the steps described here.
- Remember: Case matters!
