

# Lab 1: Thoth, SSH, and C

Due by midnight, Saturday 9/8

## What is thoth?

**thoth** is a computer in the CS building. It's hooked up to Pitt's network, but the CS department manages it and puts software on it useful for this and other courses.

By logging into thoth remotely, you don't have to worry about setting up a C compiler on your own computer.

**ssh** is a way of running commands on a remote machine. It's like your command line, but the console is connected across the internet instead of to your own computer.

Later projects will also use features only available on thoth. **So please don't try to compile your projects on your local machine.**

---

## 1. Getting connected and set up

Nothing will show up when you type your password. That's normal.

When you log in, don't worry about the "unauthorized access" message. You *are* supposed to be logging in. *You're authorized ;)*

### Windows users

You need to get an ssh client, **like PuTTY** (use the 64-bit installer). Run it, and use `thoth.cs.pitt.edu` as the address.

Say "Yes" to the certificate, give your username (lowercase) and password, and you're in!

## Mac users

Open up Terminal (⌘+Space, type "terminal", hit enter). Then run:

```
$ ssh yourusername@thoth.cs.pitt.edu
```

Say "Yes" to the certificate, give your username (lowercase) and password, and you're in!

If you can't log in, please ask the TA for help. If you are sure you're putting in your username in lowercase, and the right password, please email me and CC Dr. Khattab ([skhattab@cs.pitt.edu](mailto:skhattab@cs.pitt.edu)). He's the thoth administrator.

---

## 2. Common UNIX commands

Here's a quick reference guide to refer back to.

- `pwd` – display the current directory
- `cd dirname` – change current directory to `dirname`
  - `cd ..` moves up one directory
  - `cd ~` goes to your home directory
  - `cd -` toggles back and forth between the last two directories you were in
- `ls` – list all files/folders in current directory
  - `ls dirname` will list files/folders in the directory `dirname`
- `mv source dest` – move or rename a file
  - `source` is the file you want to move/rename
  - `dest` is the new place/name
- `cp source dest` - copy a file from `source` to `dest`
- `mkdir name` - make a new directory named `name`
- `touch filename` - make an empty file named `filename`
- `cat filename` - display contents of text file `filename`
- `less filename` - view contents of text file `filename` - good for longer files
  - press `q` to exit!

## Where are we?

When you log in, you are placed in your **home directory**.

1. Try using `pwd`; it'll show you the full path of your home directory.
  - Your home directory can be referred to as `~` in many commands as a typing shortcut.
2. Try using `ls`. It will list the files and directories.
  - Your `private` directory is what you want to do your work in. No one else can see it.
3. Do `cd private` and you'll move into that directory.
  - `pwd` again, and you'll see that your directory changed.
  - You can use `cd ..` to move up one directory.
  - You can use `cd ~` to go to your home directory.

---

### 3. Setting up your `~/ .bash_profile` a little bit

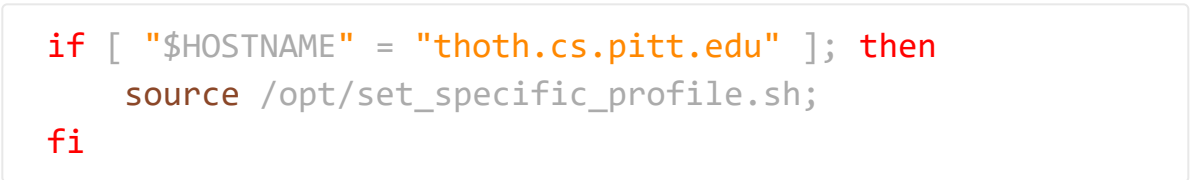
`bash` is what you're using right now - the thing you type commands into. `.bash_profile` is the configuration file for `bash`.

1. Do `cd ~`.
  - This goes back to your home directory.
2. Do `chmod u+rw .bash_profile`
3. Now let's edit it: `nano .bash_profile`
  - `nano` is a very simplistic text editor that runs inside the terminal. The controls are at the bottom of the screen - something like `^X` means press `Ctrl+X`. (Mac users, use the actual control key.)
4. Scroll down to the bottom of the file. There, you'll see:



```
# Define your own private shell functions and other command
```

5. Use the arrow keys to put your cursor after that line. Now, copy and paste this **exactly**:
  - In putty, you can right click and paste
  - In Terminal on a mac, `⌘V` will paste



```
if [ "$HOSTNAME" = "thoth.cs.pitt.edu" ]; then
    source /opt/set_specific_profile.sh;
fi
```

6. If you want a nice-looking terminal prompt like I have, find the line that starts with `export PS1`, and replace it with this:

```
export PS1="[ \[\033[1;32m\]\h\[\033[0m\] \[\033[1;34m\]\w\["
```

7. Now hit `Ctrl+O` and hit enter to save. Then hit `Ctrl+X` to exit.
8. Do `source .bash_profile`
9. Try doing `man open`. If you see this, yay! Hit `q` to exit.

```
OPEN(2)                                Linux Programmer's Manual                                OPEN(2)

NAME
    open, creat - open and possibly create a file or device

SYNOPSIS
    #include <sys/types.h>
    #include <sys/stat.h>
    #include <fcntl.h>

    int open(const char *pathname, int flags);
    int open(const char *pathname, int flags, mode_t mode);

    int creat(const char *pathname, mode_t mode);

DESCRIPTION
    Given a pathname for a file, open() returns a file descriptor, a sma
    (read(2), write(2), lseek(2), fcntl(2), etc.). The file descriptor returns
    descriptor not currently open for the process.

    By default, the new file descriptor is set to remain open across an execve
    fcntl(2) is initially disabled; the Linux-specific O_CLOEXEC flag, describe
    offset is set to the beginning of the file (see lseek(2)).

    A call to open() creates a new open file description, an entry in the sy
    offset and the file status flags (modifiable via the fcntl(2) F_SETFL opera
    entries; this reference is unaffected if pathname is subsequently removed o
    description is initially not shared with any other process, but sharing may
```

If you see `No manual entry for open` then you messed up somewhere.  
**Ask for help!**

## 4. Making a “hello world” program

Organization is good. Don’t just do all your work in `private`. Make a directory for your 449 work!

1. Make sure you are in your `private` directory. Then do `mkdir cs449`.

2. Do `ls`, and you should now see `cs449` listed. `cd` into `cs449`.
3. Now make a `lab1` directory inside your `cs449` directory, and `cd` into *that*.
4. Let's make a C file. Do `nano lab1.c`. This will create a new file and open it in `nano`.
5. Now **type the following into the editor**.

```
// Your Name (username)
#include <stdio.h>

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

6. Save the file and exit `nano`.
7. Now, compile it like so:

```
gcc --std=c99 -Wall -Werror -o lab1 lab1.c
```

If you did it right, it should print nothing. With UNIX, "no news is good news." Successful commands will usually be quiet. But if you `ls`, you should now see a new file, `lab1`. This is your executable!

8. Type `./lab1` to run your program. It should say "Hello, World!"

---

## 5. Archives and Submission

Your labs and projects will be submitted by copying (`cp`) files to a special directory.

Please follow these instructions for this and every lab/project you do. Bookmark this page!

1. Make sure your name and username are in a comment at the top of your `.c` file.
2. `cd` into your `~/private/cs449` directory.

3. Make a copy of the `lab1` directory like so: `cp -r lab1/ lab1_copy/`
  - Making a copy reduces the chances of you losing all your work.
4. Make a `tar` archive. `tar` lets you bundle several files together.

This is the step everyone messes up and loses their work on. Please be careful.

- Do this, ***using your username, not abc123.***

```
$ tar cvf abc123_lab1.tar lab1/
```

5. Check that the `tar` archive has the right files.
  - Do `tar tvf abc123_lab1.tar`. It should look something like...

```
$ tar tvf abc123_lab1.tar
```

```
drwxr-xr-x abc123/UNKNOWN1    0 2018-01-19 01:06 lab1/  
-rw-r--r-- abc123/UNKNOWN1   78 2018-01-19 00:54 lab1/  
-rwxr-xr-x abc123/UNKNOWN1 6424 2018-01-19 01:06 lab1/
```

6. `gzip` the tar file to compress it. It's as easy as:

```
$ gzip abc123_lab1.tar
```

- Now it should have transformed into `abc123_lab1.tar.gz`.

7. **Finally, copy it to the submission directory.**

```
$ cp abc123_lab1.tar.gz ~jfb42/submit/449
```

It will say nothing if it worked correctly. No news is good news! But if you're unsure, you can double-check that you submitted it properly like so:

```
$ ls ~jfb42/submit/449/abc123_lab1.tar.gz  
/afs/pitt.edu/home/j/f/jfb42/submit/449/abc123_lab1.tar.gz
```

If you messed up and need to submit again, then fix your mistake, repeat these steps, but **rename your tar file to `abc123_lab1_2.tar.gz` before copying it to the submission directory.**

If you mess up again, repeat, name it `abc123_lab1_3.tar.gz` , copy.

If you mess up again, repeat, name it `abc123_lab1_4.tar.gz` , copy.

If you mess up again, repeat, name it `abc123_lab1_5.tar.gz` , copy.

If you.... ok I think you get the idea.

© 2016-2018 Jarrett Billingsley