HIGH PERFORMANCE PROGRAMMING
UPPSALA UNIVERSITY
SPRING 2019
LINUX COMMAND LINE BASICS

The aim of this tutorial is to familiarize students with the Linux/Unix environment.

Before you start! Search on the web for a Linux/Unix quick reference. There are many to choose from — find one you like!

Another thing before you start! It may be tempting to copy+paste some of the commands from these tutorial instructions into your terminal window. However, sometimes it happens that some characters are not copied correctly when copying from a pdf. Make sure that you learn how to type the special characters needed, like the at-sign (`@`) and the tilde (`~`), directly using the keyboard.

1. Getting Started

Open a terminal. You'll see a bash prompt (a.k.a. shell) like

```
username@hostname:~$
```

where `username` is your username and `hostname` is the name of the computer you are logged in to.

You can type commands at the prompt, and bash will attempt to execute them. It does this by looking through your PATH until it finds the right program. PATH is an *environment variable* that lets the system find the things you need, no matter where they might be found.

Important: Linux is case-sensitive. That means that there's a difference between PATH, Path, and path.

Type `echo HEJ hej` to see how the `echo` command works — simply echoing back the thing you gave it as input.

Type `echo $PATH` to see what PATH expands into — a colon-delimited list of directories.

Type `which echo` to see where the `echo` command is located.

A single dot, "`.`", denotes the current working directory, where you're located right now (which should be your home directory). Similarly, "`..`" denotes the parent directory.

Type `pwd` to see your current working directory.

Type `ls` to see the contents of your current working directory.

Date: January 21, 2019.

Type `cd ..` to change directory to the parent directory.

Type `cd -` to return to the previous directory.

Create a directory for the course: Type `mkdir HPP` and cd into it (`cd HPP`).

Create a directory for this tutorial, cd into it.

Move the tutorial file to this directory (note that ~ always denotes your home directory):

```
mv ~/Desktop/Tutorial_Linux.tar.gz ./
```

Decompress the tar-ball with the following command:

`tar -xvzf L<TAB>`, where `<TAB>` is the "tab" button. After you hit TAB, the command should read `tar -xvzf Tutorial_Linux.tar.gz`.

That's right, hitting the TAB button autocompletes directories and filenames, so you don't have to type everything and, *even more importantly*, avoid typos. *Always use tab completion as you navigate directories and write file names. This will save your life!*

`tar` is an archiving program. The cryptic-looking string of letters `-xvzf` are known as *flags*. Flags control what the program is going to do.

Use the command `man tar` to look at the manual page for tar and read about what each flag does. Note that 'q' quits the man, space scrolls down a page, and typing e.g. `/Word` searches for 'Word' in the text. You can also use arrow keys to scroll up/down, or pageup/pagedown keys to scroll a whole page at a time.

Many programs also take the `--help` option, which usually prints out a shorter, more understandable summary of the command than man.

The `tar` command can also be used to create a tar-ball (this is how the `Tutorial_Linux.tar.gz` file was created). Use `man tar` to figure out how this is done, and create your own tar-ball package from some files or directories. Hint: the `-c` and `-f` options can be useful (and `-z` if you want to use gzip compression).

The `tar` command will be useful later when you are going to submit code and/or reports for the assignments in this course. Then you will be asked to package your submission into a single .tar.gz and submit that in the Student Portal.

By convention, a tar file (a.k.a. tarball) created without compression is given the extension .tar while a file created with compression is given the extension .tar.gz or .tgz.

2. Using ssh and scp

The `ssh` command can be used to login to another computer. The `scp` command allows us to transfer files to/from another computer.

In this course, ssh and scp are useful for us since using those commands we can copy our code to another computer and login and to test runs there, making it easy to test our code on different kinds of computers, with different CPU models etc.

At http://www.it.uu.se/datordrift/maskinpark/linux/ you can find a list of Linux computers that are available for students to login to and use. You can use any of the computers listed under "**Students**" there, i.e. from `arrhenius.it.uu.se` down to `vitsippa.it.uu.se`.

To login to one of them, use the `ssh` command like this:

```
ssh user123@arrhenius.it.uu.se
```

where `user123` is your username (the same username that you use to login to the computers in the lab rooms).

The first time you login to a computer, you will probably see a message about "authenticity of host" and a question "Are you sure you want to continue connecting (yes/no)?" – you need to type "yes" to continue.

Pick one of the computers in the list at http://www.it.uu.se/datordrift/maskinpark/linux/ and login there using ssh as described above. To verify that you are really logged in to a different computer, you can use the `hostname` command, that simply prints the name of the computer you are currently logged in to:

```
hostname
```

To logout from the remote system, type "`exit`". Then run the `hostname` command again to verify that you have really logged out.

Two other commands that are useful in this context are `who` and `whoami` ("who am I"). The `who` command shows the list of users currently logged in to a computer, and the `whoami` command shows your own username; the username you used to login. Note that your username on a remote system could be different from the username on your local computer. Try the `who` and `whoami` commands both locally and when logged in to a remote computer.

Note that it can be convenient to use several terminal windows when using `ssh` to login to different remote computers. For example, you could have one terminal window for your local computer, another window where you login to remote computer A and a third window where you login to remote computer B. Then, if you want to do something locally you do not need to do "`exit`" from the remote computer, you can simply use another terminal window instead.

Cd into the `Task-1` directory, and use `ls` to look at the contents of that directory. There should be a file called `hej.txt` with a little text inside it. Next, we will look at how you can copy that file to a remote computer using the `scp` command.

The `scp` command (secure copy) works in a similar way as the regular `cp` command (copy), with the difference that `scp` allows the source file or the destination file to be on a remote system.

To copy the `hej.txt` file to the other computer, you can use `scp` like this:

```
scp hej.txt user123@arrhenius.it.uu.se:/tmp/hejuser123.txt
```

Do not use exactly the filename `hejuser123.txt` but instead modify it to reflect your own name, so that you can distinguish it from other students' files that may also appear in the `/tmp/` directory of the remote computer.

The second argument in the above command, `user123@arrhenius.it.uu.se:/tmp/hejuser123.txt`, has the following form: username, then "@", then hostname, then ":", then file path. So, the example above tells `scp` that the local file `hej.txt` should be copied to `arrhenius.it.uu.se` and that the new file created (the destination) should be `/tmp/hejuser123.txt`.

To check that your copying worked, use `ssh` to login to the other computer, then do "`cd /tmp/`" and then "`ls -lrt`" to check that your file is really there. Then do "`exit`" to logout again.

You can also use `scp` to copy a file in the other direction, from a remote computer to the local computer:

```
scp user123@arrhenius.it.uu.se:/tmp/hejuser123.txt hej2.txt
```

Try that, and then use "`ls`" to check that the file was really copied. Does it work?

Do some experiments of your own using ssh and scp, to make sure you understand how those commands work. You should be able to copy files from your local computer to a remote system, login to that system and work with the files there, and also copy files back from the remote system to your local computer.