“Hello World” of the Bash shell

If you need help at any time, put your **red** sticky note on the back of your laptop. When you’ve finished the steps on the *front* of this page, put your **green** sticky note on the back of your laptop. Then, you can turn the page over and try out some of the more advanced tricks on the back while you wait for the rest of the group to be ready.

Accessing our remote shell

For this workshop, we have prepared a Linux server on which to run all the tutorial exercises. Your first assignment is to gain access to this server by opening a terminal on your laptop and using SSH to log in to our Linux server.

First, open a terminal:

- On OS X, open Applications › Utilities › Terminal.
- If you are using Windows, you should have previously installed Git Bash. To open it, click the Windows or Start icon. In the Programs list, click on Git › Git Bash.
- If you are running Ubuntu Linux, you can open the Dash (Super Key) or Applications and type terminal. Or, use the keyboard shortcut by pressing Ctrl + Alt + T. For other Linux flavors, you may have to modify these instructions slightly.

This is your local shell. Make a note of what the prompt looks like. For example, mine looks like this:

```
ffund@ffund-laptop:~$
```

It shows my username (`ffund`), hostname (`ffund-laptop`), my current working directory (~), and then a $ to signify that I’m working as a normal (unprivileged) user. (If I was working as the privileged “root” user, the prompt would end with a # instead.)

Now you are going to log in to a remote shell using SSH.

Use your net ID as your username, and `server.camp.ch-geni-net.instageni.nysernet.org` as the hostname. For example, my net ID is `ff524`, so I will run

```
ssh ff524@server.camp.ch-geni-net.instageni.nysernet.org
```

When prompted, give the following password:

```
BOOK-FINISHED-WONDER-lost
```

and then hit Enter. Note that when you type the password, the cursor will not move, you won’t see the text of the password or any asterisks appear on the screen as you type characters, and there is no indicator the password is being entered at all. This is normal; it is a security mechanism. Just continue typing the password even if there is no output, and hit Enter when you are done.

This is a temporary server that will go away when this workshop is over, so don’t save anything important on it.

Now, if you look at your prompt, you should notice that the username and hostname are different from your local shell. When working on remote servers, the prompt is useful for determining *where* you are running a command.

“Hello world”

For the standard “hello world” exercise, we use the `echo` command to print a quoted string to the terminal output. At the terminal prompt, type:

```
echo "Hello world"
```

and then hit Enter to run the command you’ve just entered.

Shell tricks

Tab autocompletion

Many terminals have a feature called "tab autocompletion" where, when you type a partial command and then press the Tab key, it will finish the command for you. Let's try this with the `whoami` command. First write out the entire command:

```
whoami
```

When you hit Enter, you should see that this command returns your username. Now try typing just

```
whoa
```

and then hit Tab. At the prompt, the rest of the command `whoami` should be filled out, and you can then hit Enter to run it.

Tab will only fill out the entire command if only one command on the system matches what you've entered so far. If there are multiple matching commands, Tab will show you all of them. You'll have to continue typing in the one you want until there is only one match, and then Tab will autocomplete it for you. Try typing

```
who
```

and then hit Tab to see how this works.

Tab autocompletion also works for file and directory names, for arguments to many commands, and for variables. For example, suppose you save the string "hello world" in a new variable called "mymessage" like this:

```
mymessage="hello world"
```

(note that there is no space on either side of the =). You can then run

```
echo $mym
```

and hit Tab, and it will be autocompleted to `echo $mymessage` (which will print "hello world" to the terminal output).

History

It's often useful to be able to see and re-run commands you've previously run.

You can use the ↑ and ↓ keys to scroll through your previous commands. Or, to see your command history all at once, run

```
history
```

You'll note that each line in the output of the `history` command has a number next to it, with which you can re-run that command. To run a command that appears as number 1 in your history, run

```
!1
```

or, to quickly run your last command again (without having to specify the number), you can run

```
!!
```

Sometimes you want to run the same command again, but with different arguments; or run a different command on the same arguments (for example, if you are doing several operations on a file.) Here are some useful shortcuts you can try:

```
!:0 # command only of last command in history
!^  # first argument of last command in history
!*  # all arguments of last command in history
!$  # last argument of last command in history
```