CSCI4113 Lab 0

Austin Glaser (810-92-4163, austin.glaser@colorado.edu)

1. Once the virtual machine has finished booting, use the command `pwd` to print the current (or **p**resent) **w**orking **d**irectory.

   ```
   $ pwd
   /home/austin
   ```

2. How many files does the *home directory* contain? A simple way to find out is to use the `ls` command.

   ```
   $ ls -l | wc -l
   3
   ```

3. How many *hidden* files does the *home directory* contain? With no arguments, the `ls` command doesn't show hidden files. Look at the man page for `ls` by running `man ls`. You can navigate the man page by using the up and down keys.

   HINT: *Hidden* files in Unix/Linux have names starting with "`.`". For example, `.bash_history` is a *hidden* file.

   ```
   $ ls -l .* | wc -l
   45
   ```

4. In what directory would you expect to find the `cp` command?

   I'd expect to find it (on a more modern linux distro) in the `/usr/bin` directory, though the canonical answer would be `/bin`

5. Where is the command to make a directory (`mkdir`) located on the filesystem? What command did you use to find `mkdir`? Give an alternative to the command you initially used to find `mkdir`.

   ```
   $ which mkdir
   /usr/bin/mkdir
   ```

   Or, alternatively (basically doing the exact same thing):

   ```
   $ find $(for path in ${PATH//:/ }; do if [ -d $path ]; then echo $path; fi; done) -name
   /usr/bin/mkdir
   ```

6. Use the `mkdir` command to create a new directory under the root user's home directory (i.e. `/root/`). Name it anything you'd like. Use the `touch` command to create a file under that directory. What does the new file contain?

   ```
   $ sudo mkdir /root/a_dir
   $ sudo touch /root/a_dir/a_file
   ```

`touch` does nothing more than create a file, so at this point the file is empty. We can verify this by:

```
$ sudo cat /root/a_dir/a_file
$
```

which drops us right back at the prompt, indicating there is indeed nothing in that file.

7. By default, the `rm` command will not remove directoriess. You can use the flag `-r` to tell the `rm` command to remove recursively; i.e., remove all files & directories under the target directory (and the target directory itself). What happens when you run the command `rm` without `-rf` to remove the directory you created in #6? What happens when you run the command `rm -rf` to remove the directory you created in #6?

   ```
   $ sudo rm /root/a_dir
   rm: cannot remove ‘/root/a_dir’: Is a directory
   $ sudo rm -rf /root/a_dir
   $
   ```

8. Print the contents of `/etc/passwd`, which contains the list of all users on the system in a very specific format. This format is:

   ```
   username:password_hash:user_id_number:group_id_number:full_name:home_directory:default_
   ```

   Write a *command pipeline* to print a list of just usernames here:

   ```
   $ cat /etc/passwd
   root:x:0:0:root:/root:/bin/bash
   bin:x:1:1:bin:/bin:/sbin/nologin
   [snip]
   sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
   austin:x:1000:1000:Austin Glaser:/home/austin:/bin/bash
   $ cut -d: -f1 /etc/passwd
   root
   bin
   [snip]
   sshd
   austin
   ```

   Or, to obtain a UUCA:

   ```
   $ cat /etc/passwd | cut -d: -f1
   ```

9. Write a *command pipeline* of the `cat`, `cut`, and `tail` commands to print only the username of the last user in /etc/passwd here:

   ```
   $ tail -n1 /etc/passwd | cut -d: -f1
   austin
   ```

10. Combine the `cat`, `cut`, and `sort` commands to print only the usernames, sorted alphabetically, in descending order. Write the *command pipeline* here:

```
$ cut -d: -f1 /etc/passwd | sort -r
tss
systemd-network
[snip]
austin
adm
```

11. Is the Debian Almquish Shell (`dash`) available on this virtual machine? Is the Fish shell (`fish`) available? List two ways below to check the availability of a shell.

    I'll almost always just use `which`:

```
$ which dash fish
/usr/bin/which: no dash in (/home/austin/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/u
/usr/bin/which: no fish in (/home/austin/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/u
```

    But we can also just try to run each command:

```
$ dash
-bash: dash: command not found
$ fish
-bash: fish: command not found
```

    Or, making the simplifying assumption that (as is the norm) shells are stored in `/bin` (which on our install links to `/usr/bin`):

```
$ find -L /bin -name dash -or -name fish
$
```

    which returns no results. With this last, the `-L` option is important. It means "follow sym*l*inks", and since (as noted above) `/bin` is itself a symlink, the command will produce false negatives without it:

```
$ find /bin -name bash
$ find -L /bin -name bash
/bin/bash
```

    This can be bypassed by adding a trailing slash:

```
$ find /bin/ -name bash
/bin/bash
```

    or by (some would claim more directly) just searching `/usr/bin`:

```
$ find /usr/bin -name bash
/usr/bin/bash
```

However, forcing a trailing slash is a brittle solution, and in this *particular* case I would argue for searching `/bin` since it's a convention to write `/bin/sh`, `/bin/bash`, etc in a shebang. It's really all pointless, though, since `which` automatically handles all of the details.

12. What is the current value of the `$PATH` environment variable? How would you append the directory `/usr/local/bin`?

```
$ echo $PATH
/home/austin/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/opt/bin
$ export PATH="$PATH:/usr/local/bin"
$ echo $PATH
/home/austin/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/opt/bin:/usr/local/
```

The `export` exposes this variable change to any parent processes. It's not strictly necessary here, but is the normal convention for scripts which might be run as subshells.

13. Issue this command and explain the result: `>time; date >> time; cat < time`

```
$ >time; date >> time; cat < time
Tue Aug 30 21:29:08 MDT 2016
```

This is not a single command, but really three commands executed sequentially. The first truncates the file `time`, if it exists, or creates it if it doesn't. The second appends the output of the `date` command to the file `time`. The third command prints the contents of the time file, which we just wrote.

This is obviously an exercise in input and output redirection, but in a real setting I would accomplish the same result using:

```
$ date | tee time
Tue Aug 30 21:32:30 MDT 2016
```

This has the obvious advantage of being far more readable.

14. Take a snapshot of the virtual machine, then run the command `rm / -rf` on your virtual machine. What happened? Restart the virtual machine (you may have to click Machine, then Reset). Does it boot?

```
$ rm -rf /
rm: it is dangerous to operate recursively on '/'
rm: use --no-preserve-root to override this failsafe
$ sudo rm -rf /
rm: it is dangerous to operate recursively on '/'
rm: use --no-preserve-root to override this failsafe
$ rm --no-preserve-root -rf /
[permission denied for days]
^C
```

```
$ sudo rm --no-preserve-root -rf /
[same thing, but only warnings on /proc, /sys]
-bash: wc: command not found
-bash: ((: r =  : syntax error: operand expected (error token is "=  ")
-bash: wc: command not found
-bash: ((: r =  : syntax error: operand expected (error token is "=  ")
```

`rm` tried *so hard* to keep me from doing that. I feel kind of bad...

These last are likely from my custom prompt script, but the fact that `wc` can't be found is a symptom of the horrible things we've just done to our system.

I can't even shutdown from the command prompt, or examine the damage (since neither `shutdown` nor `ls` are left on disk). Shutting down gracefully through the VM's interface is likewise no longer an option.

It's a foregone conclusion by now, but no. No it does not boot. Instead, we're dropped into grub:

```
.
error: file `/grub2/i386-pc/normal.mod' not found.
Entering rescue mode...
grub rescue>
```

Poor machine =(