Shell Scripting 2020 first week

Format of the learning diary and scripts

The tasks are returned in moodle.

Return the learning diary in **PDF format**. Preamble the document with the following information:

- Name and student number

The learning diary should contain the answers to questions made in the "Put in your answer" parts. Also example output can be put into the answer.

Include the code in week's directory as a tarball with scripts named with the task name as "task#.sh", i.e. for task 2 the name would be "task2.sh".

Deadline for this set of tasks is Monday 9th of November 2020 and I will release the next set of tasks on 2nd of November.

If you run into problems, try to read the man pages and try google. If you are still stuck with the task send me a msg in moodle and I will answer during normal office hours (9-17).

Directory setup (1)

The solutions are meant to be iterative in the sense that a student builds up on the previous program code to achieve more and more difficult tasks.

Open a terminal to your home directory, and create a *ShellScripting2019* folder using the terminal. Within that directory create a *Week1* folder.

Put in your answer:

- the command lines for creating the directories

Identity shift (2)

The alias command can make life simple by abbreviating very long command line argument lists into shorter commands.

Mask the standard ls command by creating your own alias, using your favourite command line arguments for ls.

Put in your answer:

- the alias of your favorite command line for ls
- an alias cman which uses a browser (chrome, chromium, firefox, lynx, or similar) to read man pages. (university's cubbli installation has problems with xhost so this can be currently solved with lynx).

(NON)-logins (3)

The key difference between login and non-login shells can be demonstrated using ssh.

You can try out the difference by removing the snippet invoking ~/.bash_aliases from one of the bash configuration files and then invoking ssh for a login or directly by using the ls command.

```
Hint: when you call a command like ssh melkki.cs.helsinki.fi, the host operating
system actually invokes ssh -t melkki.cs.helsinki.fi "bash -" if your default shell
is Bash. Check echo $SHELL on the target host to verify this.
```

Put in your answer:

- Example where the output of ls differs depending on how you have invoked ssh and **why**.

Enter RSYNC (4)

Between hosts which do not share their home directories, the best way to copy files is through rsync. It is a very powerful utility, specially for backups, since it can detect changes between a source and a destination file, and then send only the changed portions.

In the Dept. network, we need some content to actually see the speed improvement. Let's copy a sizeable chunk of files.

The Exactum-kamera keeps watch of your Dept's home building. It saves one image every hour, on the hour. These archives go back many, many years, and can be browsed if one knows where to look. I'll tell you, but let's keep this our secret:

https://www.cs.helsinki.fi/u/tkt_cam/2018/12/01/ (Exactum cam) is the web URL for Saturday, December 01st, 2018. You could go there and click on each link to view the pictures, or you could just copy the files over the NFS system.

rsync --archive /cs/home/tkt_cam/public_html/2018/12/01/
~/ShellScripting2019/Week1/Wednesday.2018.12.01

Please be very thorough with those commands.

Put in your answer:

- Experiment by running rsync on either a previously empty subdir Wednesday.2018.12.01 and one previously populated with previous rsync. Show how rsync's output changes. Explain what's going on. Try adding the --stats argument to rsync.

Time and Date (5)

The commands from the previous exercise are fine, but very static. They are good for that one day only.

Read man date and experiment with its format parameters. For example, try date +%d.%m.%Y

Put in your answer:

- Print today's date in the format equivalent to 'Wednesday.2017.09.30'. Show the command and it's output.

Inserting date (6)

You can use backquotes or $(command) to evaluate a command into a string

Example:

Echo today's date

```
#!/bin/bash
echo "Today is `date +%m.%d.%Y`"
echo "Today is $(date +%m.%d.%Y)"
```

Put in your answer:

- Create a script rsync-todays-Exactum-cam.sh that uses echo to print a rsync command with the correct paths for the current date. Paste the code below.

Two at once (7)

A program may produce both useful content to stdout and also error messages to stderr. This typically happens when a program is partially successful.

Figure out how to redirect stdout to one file and stderr to another file simultaneously. Use ls.

Put in your answer:

- Present the command.

```
Hint: you can ls multiple directories at once.
```

Hey! What about STDIN? (8)

By default, stdin is read from the keyboard connected to the terminal session. As you may have noticed, ls isn't exactly interested in keyboard input. In other words, it doesn't really read its stdin.

Let's take a look at a program which does read its stdin! Enter wc. Erm... Pun unintended. Anyhow. In this case, the name wc originates from "word count", at least based on my best guess. If you take a sneak peek into man wc, as you should do every time you learn a new command, you'll notice that wc prints not only characters, but also bytes and lines.

Bytes are not as useful to count as they used to be, but counting lines can be very handy. Let's try that now. If you enter wc -l, wc will count every line you enter until you press ctrl-d.

Put in your answer:

- Try the same thing with cat. What do you think cat is doing to your stdin?

```
Ctrl-d is actually more than a little special. It causes the terminal
to produce an end-of-transmission character, which is a signal to stop waiting for
more input from stdin.
```