Bash scripting

processes practice for CSC420 Operating Systems Spring 2022 Lyon College

README

- This file accompanies lectures on the shell and `bash(1)`. To gain practice, you should type along in your own Org-mode file. You have to have Emacs and my `.emacs` file installed on your PC or the Pi you're working with.
- This section is based on chapter 24 of Shotts, The Linux Command Line (2e), NoStarch Press (2019), and on the DataCamp course "Introduction to Bash Scripting".
- To make this easier, use the auto expansion (`<s`). This will only work if you have my `.emacs` file (from GDrive) installed.

- Add the following two lines at the top of your file, and activate each line with `C-c C-c` (this is confirmed in the echo area as `Local setup has been refreshed`)):

```
#+PROPERTY: header-args:bash :results output
```

- Remember that `C-M-\` inside a code block indents syntactically (on Windows, this may only work if you have a marked region - set the mark with `C-SPC`).

Overview

- A shell script is a file containing a series of commands.
- The shell is both a **command line interface** to the OS and a **scripting language interpreter**.
- The shell reads the file, interprets and carries them out as if they had been entered on the command line.

How to write a shell script

- Write the script in a text editor (Emacs or vi or nano)
- Make the script executable by setting the file permissions
- Put the script somewhere the shell can find it

Script file format

- `[X]` Fire up an editor and create a "Hello World" program `hello.sh`. You can use `vi` or `nano` if you like!
  - In Emacs, `C-x C-f hello.sh` to create the file, and `C-x C-s` to save it
  - In vi, write `vi hello.sh` in the terminal, insert with `i`, save with `:w` and exit with `:q`
- `[X]` Put a comment in after the command, using `#`
- `[X]`

  You got to get the location of your `bash` program right.

```
which bash  # likely in /usr/bin/bash
```

  First line of your script should look like this:

```
#!/usr/bin/bash
```

- `[X]` If successful, run the same command in the terminal (including the comment.
- The first line of the script is the *shebang* to tell the kernel the name of the interpreter that should be used to execute the script.
- `[X]` Make a copy of the file, find a different interpreter (e.g. `csh`, the C shell) and modify the file accordingly.

Executable permissions

- `[X]` Check file permissions with the command `chmod`
- `[X]` Make your file executable. Check the permissions.

Script file location

- `[X]` Save and run the file on the shell (you can do that inside Emacs with `M-x shell`).
- For the file to run, an *explicit* path has to be provided, otherwise you get the `Command not found` error
- The 'source' operator `.` executes bash on the current location. It is a shell builtin that reads a specified file of shell commands and treats it like input from the keyboard.
- `[X]`

  Print the path that the OS searches when looking for a command:

  ```
  echo $PATH
  ```

- `[X]` Make a directory `/bin` in your home directory and add it to the `PATH` using the syntax `PATH=$HOME/bin:$PATH`
- `[X]` Check that `PATH` was altered as you wanted. The new directory should be the first in the list.
- `[X]` Copy `hello.sh` to that new directory and run the file again from your current location.
- `[ ]`

  To apply this change of `PATH` whenever `bash` is called, you need to include this line in your initialization file `$HOME/.bashrc`:

  To find the file:

  ```
  ls -la .bashrc
  ```

  ```
  export PATH=$HOME/bin:$PATH
  ```

  To append this line to `.bashrc` do:

  ```
  echo "export PATH=$HOME/bin:$PATH" >> .bashrc
  ```

  To check if the appending was successful (`cat` works, too):

  ```
  tail -1 .bashrc
  ```

- [ ]

  To make the change, you have to source the `$HOME/.bashrc` file using the source operator `.`:

  ```
  . .bashrc
  ```

Summary

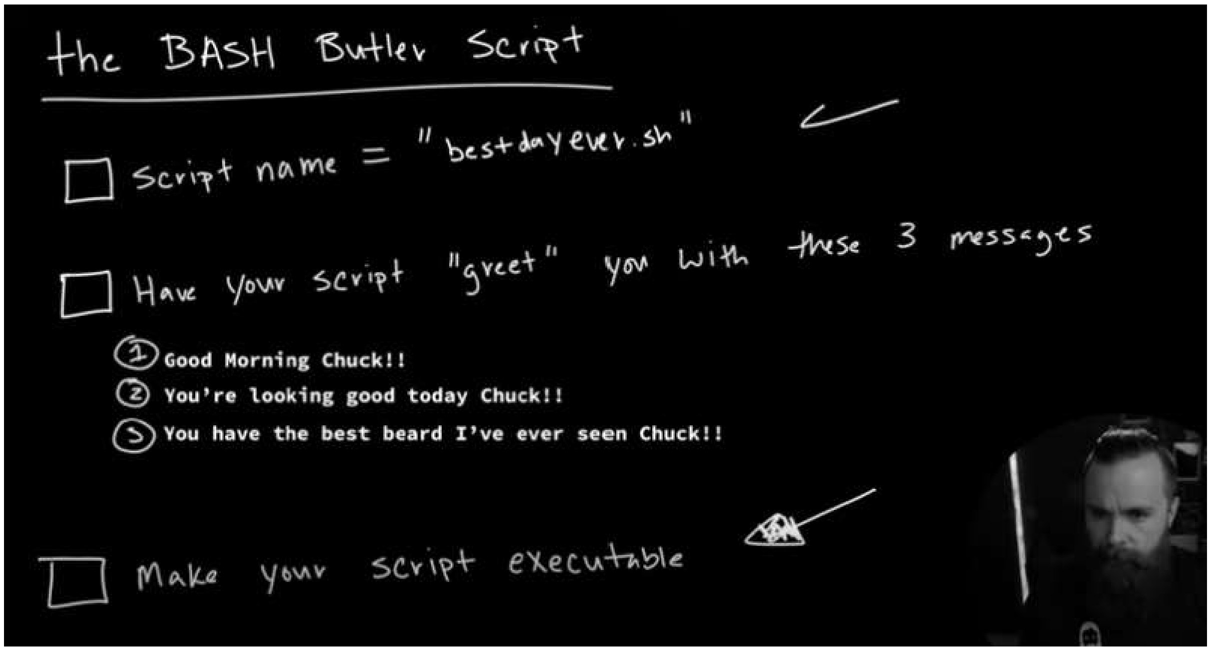
- `[X]` How to write a shell script in 3 steps
- `[X]` Script file format with *shebang*
- `[X]` Permission to execute with `chmod`
- `[X]` Location with `$PATH`

Assignment: "BASH scripting will change your life"

What is this?

- Assignment for Tuesday, 19 April (no physical session because of Honors Convocation session).
- These notes are based on NetworkChuck's video (Apr 12, 2022).
- This guy is a wildly successful Linux aficionado, who seems to make quite a bit of money with the recent resurgence of the command line[1].
- Do not sign up for "Linode" by the way. You're better than that: I told you seven different ways of getting hold of Linux. If you're totally lazy, just use bash on replit.com.

- Note: you're already on this journey while others are still only thinking about it or have no clue on how to begin!


the BASH Butler Script

☐ Script name = "bestdayever.sh"

☐ Have your script "greet" you with these 3 messages

1. Good Morning Chuck!!
2. You're looking good today Chuck!!
3. You have the best beard I've ever seen Chuck!!

☐ Make your script executable

Figure 1: Network Chuck's bash scripting project

What you should do

- Watch the 14 min video and follow Chuck's instructions!
- If you're not on Linux, use `replit.com` as you know it from class.
- Check my notes below if you like.

My notes while watching the video

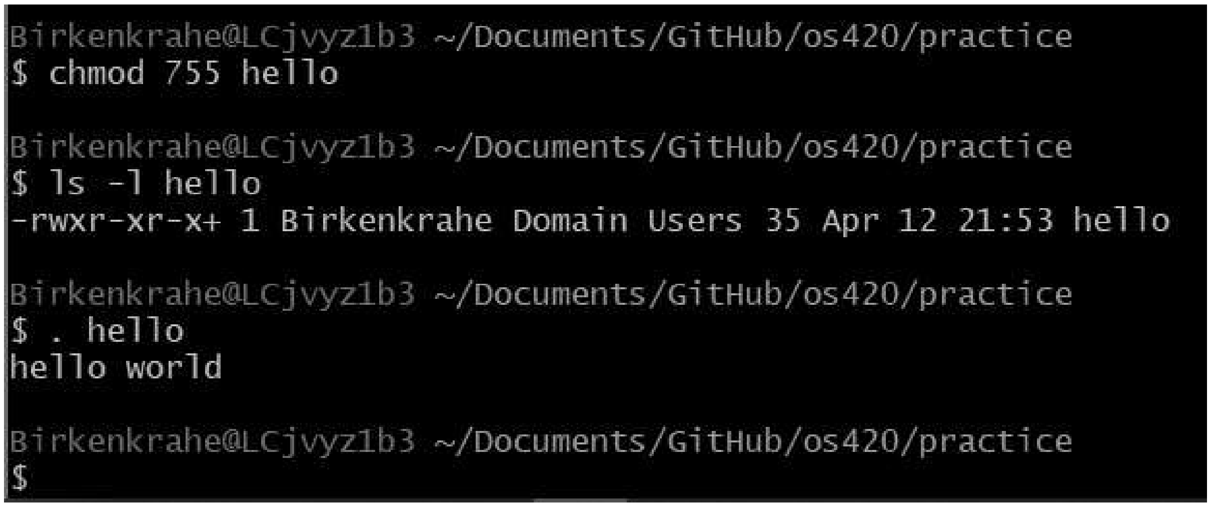
- In Emacs, of course. My Windows box recognizes `bash` because of Cygwin:

  ```
  which bash
  ```

  ```
  /usr/bin/bash
  ```

- This means that I can use this *shebang* for my files: `#!/usr/bin/bash`
- When `bash` finds this expression at the top of a file (any file), it'll run the script. Really any file? Let's test that:
  - I created a shell script with the *shebang* but called it `hello` (without the `.sh` file extension). Will it run? Try it!
  - The following script worked in Cygwin with the command `. hello` after changing the permissions to `755`.

    ```
    #!/usr/bin/bash
    echo 'hello world'
    ```



```
Birkenkrahe@LCjvyz1b3 ~/Documents/GitHub/os420/practice
$ chmod 755 hello

Birkenkrahe@LCjvyz1b3 ~/Documents/GitHub/os420/practice
$ ls -l hello
-rwxr-xr-x+ 1 Birkenkrahe Domain Users 35 Apr 12 21:53 hello

Birkenkrahe@LCjvyz1b3 ~/Documents/GitHub/os420/practice
$ . hello
hello world

Birkenkrahe@LCjvyz1b3 ~/Documents/GitHub/os420/practice
$
```

Figure 2: Running hello world with shebang

  - As you can see you don't need the `bash` command before the file, or the file type `.sh` (remember: Linux doesn't need file types).

- I notice that he uses the `nano` editor: you can install this editor with Cygwin if you like. To do this, re-run the setup program and when you're in the installation dashboard, choose `nano`.

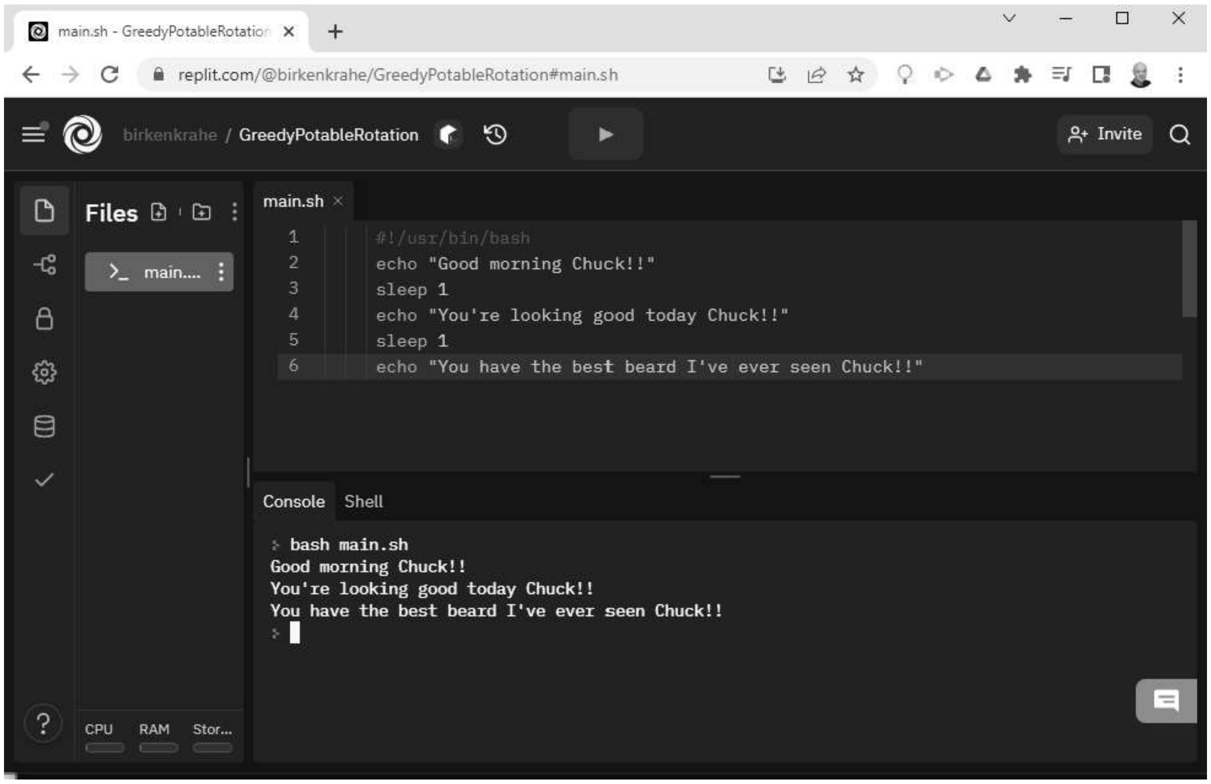
- I simply create this file in Emacs and tangle it to a shell script (just in case I decide to run it on the command line, too). Then I run it in here.

```bash
#!/usr/bin/bash
echo "Good morning Chuck!!"
sleep 1
echo "You're looking good today Chuck!!"
sleep 1
echo "You have the best beard I've ever seen Chuck!!"
```

```
Good morning Chuck!!
You're looking good today Chuck!!
You have the best beard I've ever seen Chuck!!
```

```
Good morning Chuck!!
You're looking good today Chuck!!
You have the best beard I've ever seen Chuck!!
```

- Though this works alright in Emacs, the `sleep` command is not recognized. So I'm going to move over to `replit.com`. Here, it looks like this:


main.sh - GreedyPotableRotation
replit.com/@birkenkrahe/GreedyPotableRotation#main.sh
birkenkrahe / GreedyPotableRotation
Invite
Files
main....
main.sh
Console
Shell
CPU
RAM
Stor...

```
#!/usr/bin/bash
echo "Good morning Chuck!!"
sleep 1
echo "You're looking good today Chuck!!"
sleep 1
echo "You have the best beard I've ever seen Chuck!!"
```

```
> bash main.sh
Good morning Chuck!!
You're looking good today Chuck!!
You have the best beard I've ever seen Chuck!!
>
```

Figure 3: Chuck's script in a replit.com Linux container

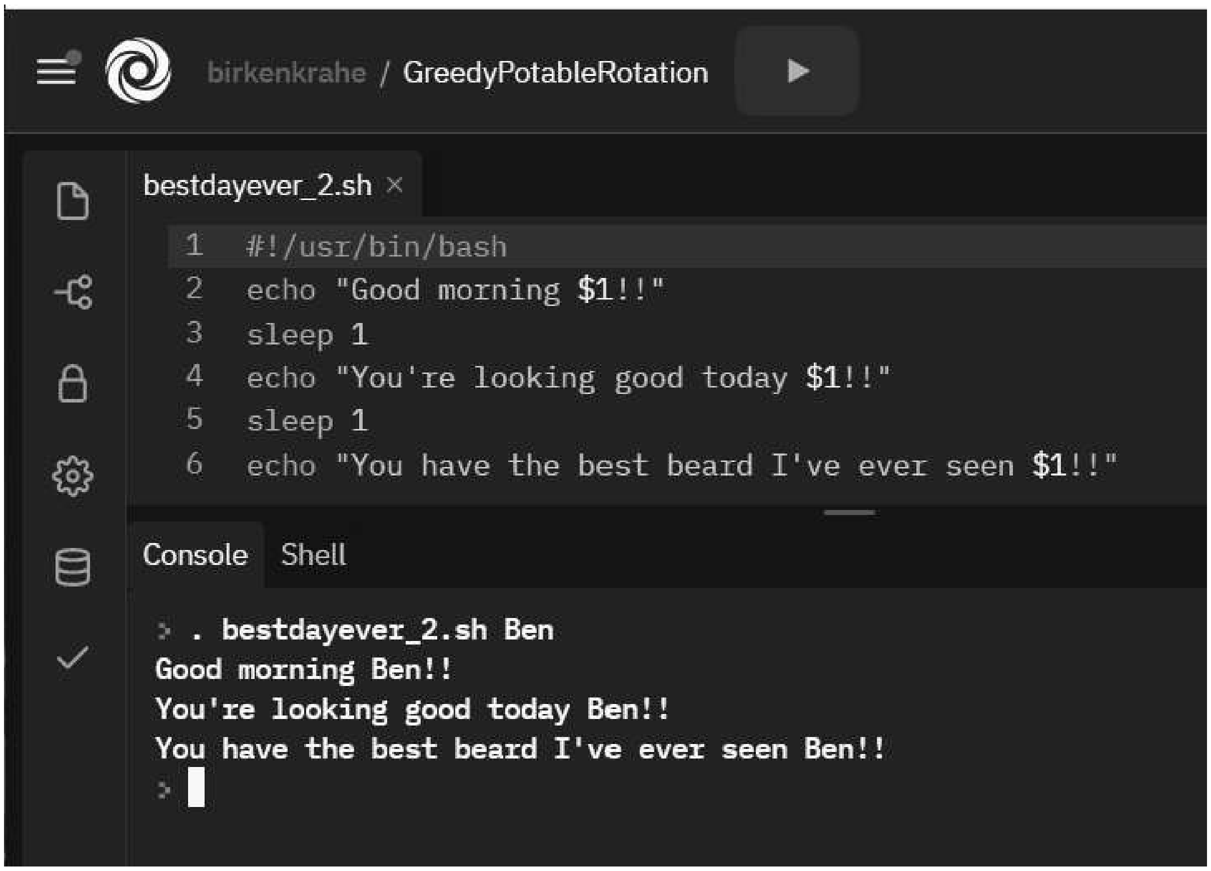
- Enable execution of the script with `chmod +x` - that's the same thing as `chmod ugo+x`.

- In the video, Chuck is getting started with **variables**, because

  "We're IT people and we're lazy."

- [ ] Assignments for you:
  1. Create Chuck's original script `bestdayever.sh`.
  2. Make the script executable.
  3. Copy the file to `bestdayever_1.sh` before you change it.
  4. Introduce a variable `$name` instead of `Chuck` and assign it to another name at the start of the file.
  5. Copy the file to `bestdayever_2.sh` before you change it.

  6. Comment out the assignment of `$name` and change it in all statements to `$1`. Then run the file with the name as the argument.



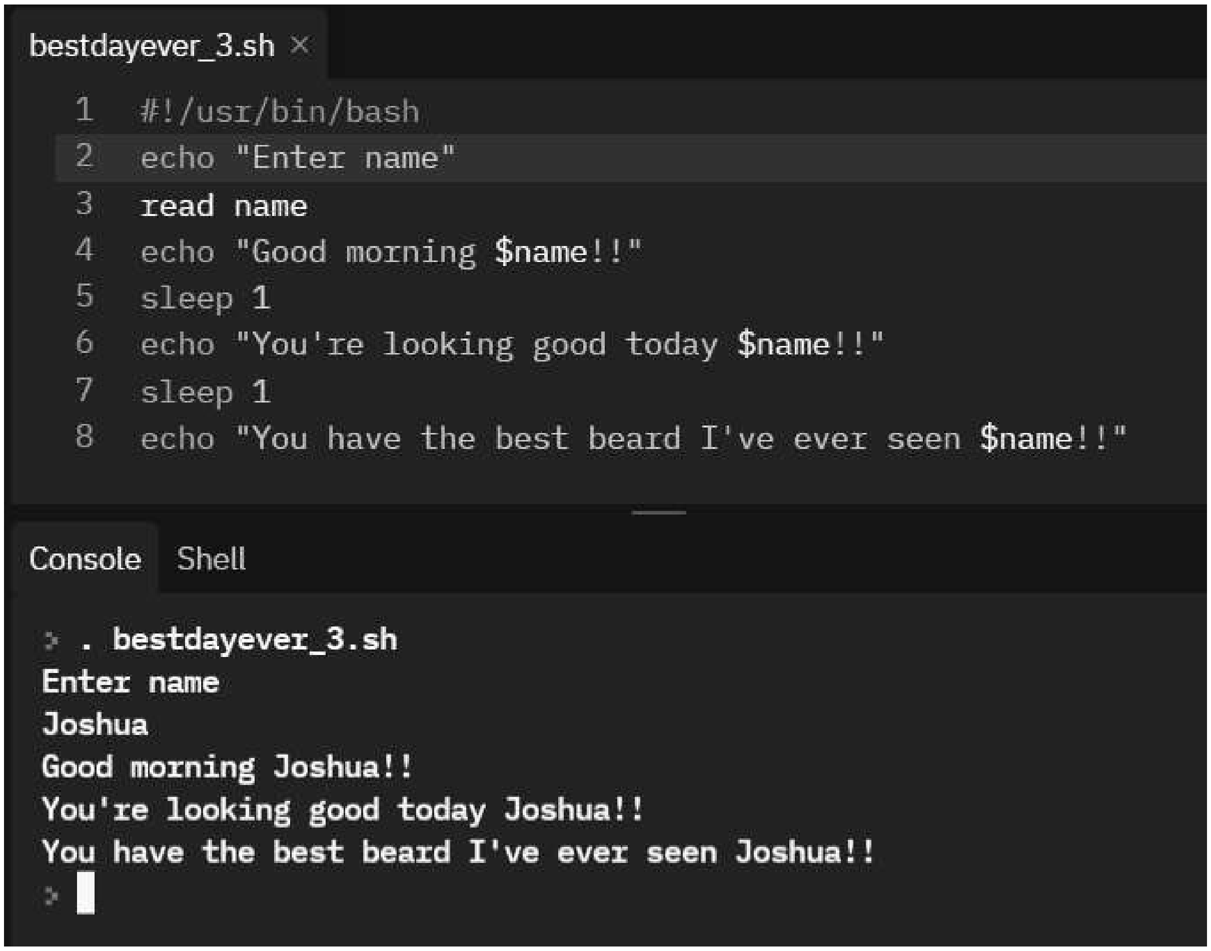
```
#!/usr/bin/bash
echo "Good morning $1!!"
sleep 1
echo "You're looking good today $1!!"
sleep 1
echo "You have the best beard I've ever seen $1!!"
```

```
> . bestdayever_2.sh Ben
Good morning Ben!!
You're looking good today Ben!!
You have the best beard I've ever seen Ben!!
>
```

Figure 4: Chuck's script reading input (1)

  7. Copy the file `bestdayever_1.sh` to `bestdayever_3.sh` before you change it.

  8. Chuck shows another way to get the `name` variable set: with the command `read`.



```
#!/usr/bin/bash
echo "Enter name"
read name
echo "Good morning $name!!"
sleep 1
echo "You're looking good today $name!!"
sleep 1
echo "You have the best beard I've ever seen $name!!"
```

```
> . bestdayever_3.sh
Enter name
Joshua
Good morning Joshua!!
You're looking good today Joshua!!
You have the best beard I've ever seen Joshua!!
>
```

Figure 5: Chuck's script reading input (2)

References

- Shotts, The Linux Command Line (2e), NoStarch Press (2019).
- DataCamp, Introduction to Bash Scripting (course).

Footnotes:

[1] In the wake of Microsoft and other vendors' decision to snuggle up to Linux, and the increased interest - partly politically motivated - to invest in cybersecurity.

Author: Marcus Birkenkrahe
Created: 2022-04-12 Tue 22:40