Introduction to Linux – Part 2

Anita Orendt and Brett Milash

Center for High Performance Computing

Editors

There are many choices – a few are:

- nano
- vi
- emacs

Nano Editor

To start either

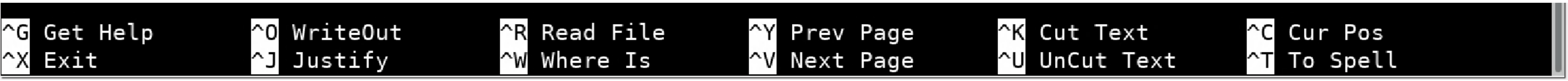
`nano`

OR

`nano` filename

-- if filename exists, it will open file in editor; if it does not, this will be the name used when you save the file.

-- if you start nano without a filename it will prompt you for a name when you “WriteOut” using ^O (Cntrl O)



```
^G Get Help    ^O WriteOut    ^R Read File   ^Y Prev Page   ^K Cut Text    ^C Cur Pos
^X Exit        ^J Justify     ^W Where Is    ^V Next Page   ^U UnCut Text  ^T To Spell
```

Vi editor

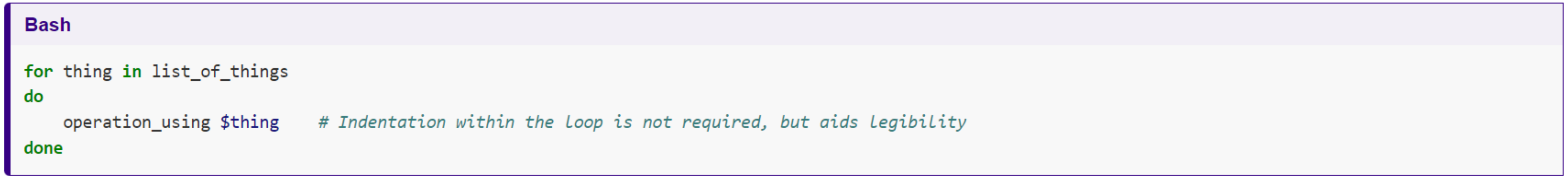
- Another common choice
  - Start with the command **`vi`** or **`vi`** filename
  - **`vi`** at CHPC is actually **`vim`**, which is an improved version of **`vi`**
  - More feature rich, takes more time to learn
- Not going into detail but we do provide a vi cheat sheet and a vi graphical cheat sheet – linked on the presentation page https://www.chpc.utah.edu/presentations/IntroLinux3parts.php
- There is also a tutor program – start with command **`vimtutor`** which is a great tool to learn to use the program

Loops

- Used when you want to preform the same action many times, such as on multiple files
- There are a number of ways you can do this
- One option
    - List multiple arguments for a command to act upon
    - Example (go to the `LinuxClass/shell-lesson-data/exercise-data/creatures` directory):
        - **`head -n 3`** basilisk.dat minotaur.dat unicorn.dat
- Another option – do a loop with a for/do statement

Loops

- ❑ in bash syntax a loop looks like:



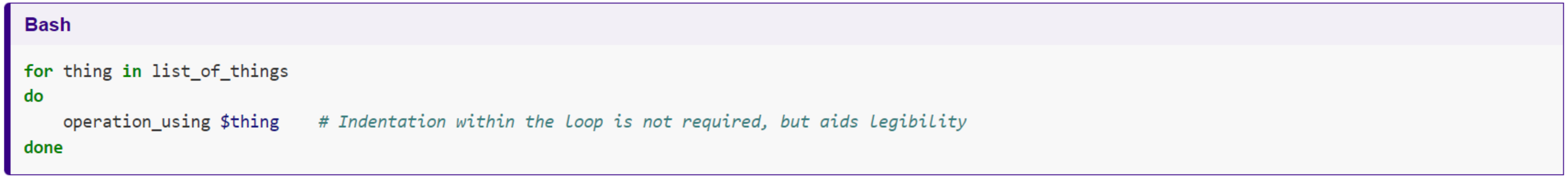
Bash

```bash
for thing in list_of_things
do
    operation_using $thing    # Indentation within the loop is not required, but aids legibility
done
```

- ❑Can do on a single line, separating by ';'
  - ❑for filename in basilisk.dat minotaur.dat unicorn.dat; do head -n 3 $filename; done
- ❑Note when you do the loop in a multi line format, there is a shell prompt ">" that is used to ask for the rest of the command
- ❑ Can use wildcards – try using a * instead of listing each file separately

Loop Terminology

- in bash syntax a loop looks like:



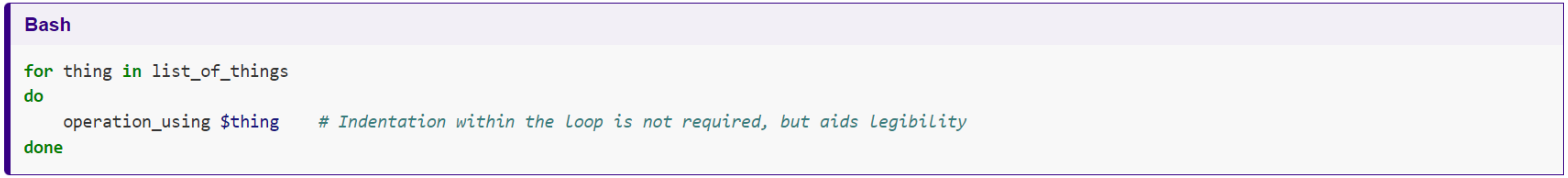
Bash

```
for thing in list_of_things
do
    operation_using $thing    # Indentation within the loop is not required, but aids legibility
done
```

- In this loop **thing** is a **variable**. During execution **$thing** is set to the first item in the list, the operation(s) is done, then it goes to the second and repeats the operation(s), etc until it reaches the last item in the list. Then the loop is exited.
- You can choose anything for **thing** – however, your choice of the what to use for thing should help a person reading the file understand what the loop is doing and what it is acting upon

Exercise

- ❑ in bash syntax a loop looks like:



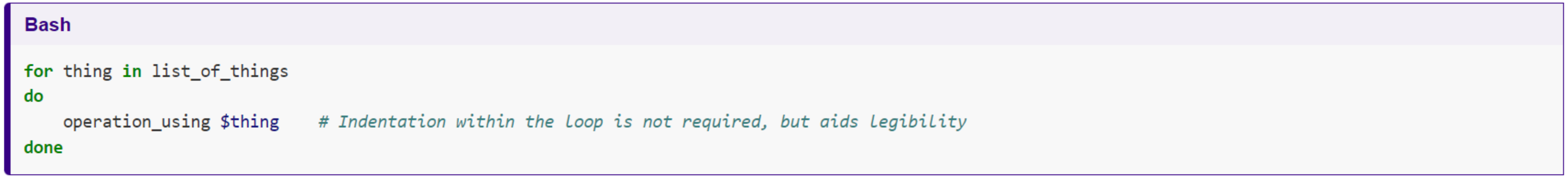
Bash

```
for thing in list_of_things
do
    operation_using $thing    # Indentation within the loop is not required, but aids legibility
done
```

- ❑ Go to the directory `shell-lesson-data/exercise-data/proteins` and write a loop that lists all of the pdb files
- ❑ Write a loop that lists all of the files

Exercise

- in bash syntax a loop looks like:



Bash

```
for thing in list_of_things
do
    operation_using $thing    # Indentation within the loop is not required, but aids legibility
done
```

- Go to the directory `shell-lesson-data/exercise-data/proteins` and write a loop that lists all of the pdb files
- Write a loop that lists all of the files – can use **echo** command
- Build upon this look by adding another command
    - Copy each file to a new name based on the existing name, file.pdb to orig-file.pdb
- Add to the loop again a loop, using pipe or redirect to a file
    - Use this to make a single file that has the content of all of the pdb files in this directory

```
for file in *.pdb
 do
   echo $file
    cp $file orig-$file
    cat $file >> all.pdb
 done
```

Exercise – Nelle's Data

- ❑ To process her data files Nelle will need to run an analysis on each of her sample files in **north-pacific-gyre**. The files to be processed have the consistent names of **NENExxxxA.txt** and **NENExxxxB.txt**
- ❑The analysis (a script – more on this next time – written by her supervisor is called **goostats.sh** and requires two arguments – the input file name and the output file name.
- ❑Nelle decides to call the output file **stats-NENExxxA.txt** – prepending the filename with stats-
- ❑She is being careful so she wants to test (using echo instead of running the script
- ❑ Hint – start with 1 file and run test to create the two arguments. You can prepend a v**ariable** with additional information: **stats-$variable**

Exercise – Nelle’s Data Answer

- To make sure getting all of the files

```
$ for datafile in NENE*A.txt NENE*B.txt
> do
>     echo $datafile
> done
```

- Test getting the output file name

```
$ for datafile in NENE*A.txt NENE*B.txt
> do
>     echo $datafile stats-$datafile
> done
```

- Executing the script across all files

```
$ for datafile in NENE*A.txt NENE*B.txt
> do
>     echo $datafile
>     bash goostats.sh $datafile stats-$datafile
> done
```

Some other useful commands

- **`cut`** – e.g. **`cut -f 2 -d : file.txt`**
  - Prints selected parts of lines from file to standard output (screen)
- **`du`** – e.g. **`du –hs`**
  - Reports file space usage; -s give summary of total usage, -h gives it in “human readable” format of K, M, G
- **`df`** – e.g. **`df –h`**
  - Reports file system disk space usage
- **`ln`** – e.g. **`ln -s ~/bin/prog.exe prog1.exe`**
  - create a link between files (-s symbolic)

On your own – Use and explore options of these commands

File Permissions

- Shown with **`ls -l`**
- User (u), group (g), other (o)
  - -rw-rw-r-- 1 u0028729 chpc 86 Jul 30 02:41 notes.txt
- Can also use a for all to set u, g, and o to same settings
- Permissions are read (r), write (w), execute or search for a directory (x)
- **`chmod`** – to change permissions of file or directory, can set
- Examples:
  - **`chmod g=rwx file`**
  - **`chmod g+x file`**
  - **`chmod o-rwx *.c`**
- Executable files (programs and scripts) must have executable permissions; directories must be executable in order to be able to cd into them
  - **`chmod +x *.sh`**

Login Scripts & Environment Variables

- In your home directory are a number of dot files - `.bashrc` and `.custom.sh`, `.tcshrc` and `.custom.csh` Depending on your shell choice, the appropriate pair of these are executed during login.
- These set the environment (as environment variables) needed for you to work on CHPC resources
- Commands to check your environment: `env` or `printenv`
- Some important variables
  - $USER
  - $HOME
  - $PATH – paths to search for commands
  - $LD_LIBRARY_PATH – paths to search for libraries when linking a program (more on that later)

Processes

- A Process is a running Linux program
  - Each process has a PID (Process ID)
- **`ps`** reports a snapshot of current processes

`ps, ps x`	Display ALL of your processes
`ps ax`	Display ALL processes
`ps aux`	Display ALL processes (more detailed)
`ps auxw`	Display ALL processes (more detailed & unlimited width)
`ps –eFwww`	Also displays ALL processes

- **`kill PID`** kills the process with the specified PID
- **`killall processname`** kills all process with the processname
- **`kill -9 PID`** kills the process with the specified PID if a kill does not work

Other Job Controls

- Ctrl+C (^C) terminate the currently running process
- Ctrl-Z (^Z) suspends the currently running process
- & runs the job in the background
- Jobs: lists all jobs, with their number
- bg %n: puts current or specified job (%n) in the background and
- fg %n: bring suspended program back to the foreground, e.g., so that it occupies the shell until done

Monitoring processes/usage

- uptime
- free
- top
- atop
- htop
- sar

Moving files to/from CHPC

https://www.chpc.utah.edu/documentation/data_services.php

- Can mount CHPC file systems on your local machine (Windows, Mac or Linux), must be on campus or using the campus VPN
- Windows – there are graphical tools such as WinSCP
- Mac, Windows, cloud options – cyberduck, another graphical tool
- Linux
  - **`scp`** command (secure shell copy) – to copy files between linux systems
- **`wget`** – to download from web with URL
  - **`curl`** is another option
- For larger data sets – look into the Data Transfer Nodes (DTNs) and transfer tools such as globus, see
  - https://www.globus.org/quickstart
  - https://www.chpc.utah.edu/documentation/data_services.php