Please Note : Some Exercises are intentionally added which are not discussed in class, this is to extend your knowledge and promote self learning skills.

**practice: working with files**

1. List the files in the /bin directory

2. Display the type of file of /bin/cat, /etc/passwd and /usr/bin/passwd.

3a. Create 2 blank files wolf.jpg , and book.pdf(use touch for that)

3b. Display the type of file of wolf.jpg and book.pdf

3c. Rename wolf.jpg to wolf.pdf (use mv).

3d. Display the type of file of wolf.pdf and book.pdf.

4. Create a directory ~/touched and enter it.

5. Create the files today.txt and yesterday.txt in touched.

6. Change the date on yesterday.txt to match yesterday's date.

7. Copy yesterday.txt to copy.yesterday.txt

8. Rename copy.yesterday.txt to kim

9. Create a directory called ~/testbackup and copy all files from ~/touched in it.

10. Use one command to remove the directory ~/testbackup and all files in it.

11. Create a directory ~/etcbackup and copy all \*.conf files from /etc in it. Did you include all subdirectories of /etc ?

12. Use rename command to rename all \*.conf files to \*.backup .

**practice: file contents**

1. Display the first 12 lines of **/etc/services**. 2. Display the last line of **/etc/passwd**.

3. Use cat to create a file named **count.txt** that looks like this:

One

Two

Three

Four

Five

4. Use **cp** to make a backup of this file to **cnt.txt**.

5. Use **cat** to make a backup of this file to **catcnt.txt**.

6. Display **catcnt.txt**, but with all lines in reverse order (the last line first).

7. Use more to display **/var/log/messages**.

8. Display the readable character strings from the **/usr/bin/passwd** command.

9. Use **ls** to find the biggest file in **/etc**.

10. Open two terminal windows (or tabs) and make sure you are in the same directory in both. Type **echo this is the first line > tailing.txt** in the first terminal, then issue **tail -f tailing.txt** in the second terminal. Now go back to the first terminal and type **echo This is another line >> tailing.txt** (note the double >>), verify that the **tail -f** in the second terminal shows both lines. Stop the **tail -f** with **Ctrl-C**.

11. Use **cat** to create a file named **tailing.txt** that contains the contents of **tailing.txt** followed by the contents of **/etc/passwd**.

12. Use **cat** to create a file named **tailing.txt** that contains the contents of **tailing.txt** preceded by the contents of **/etc/passwd**.

13. Does the file **/bin/cat** exist ? What about **/bin/dd** and **/bin/echo**. What is the type of these files ?

14. What is the size of the Linux kernel file(s) (vmlinu\*) in **/boot** ?

15. Are there any files in **/etc/skel/** ? Check also for hidden files.

16. Display **/proc/cpuinfo**. On what architecture is your Linux running ?

17. Can you enter the **/root** directory ? Are there (hidden) files ?

**practice: commands and arguments**

1. How many **arguments** are in this line (not counting the command itself). touch '/etc/cron/cron.allow' 'file 42.txt' "file 33.txt"

2. Is **tac** a shell builtin command ? 3. Is there an existing alias for **rm** ?

4. Read the man page of **rm**, make sure you understand the **-i** option of rm. Create and remove a file to test the **-i** option.

5. Execute: **alias rm='rm -i'** . Test your alias with a test file. Does this work as expected ?

6. List all current aliases.

7a. Create an alias called 'city' that echoes your hometown.

7b. Use your alias to test that it works.

8. Execute **set -x** to display shell expansion for every command.

9. Test the functionality of **set -x** by executing your **city** and **rm** aliases.

10 Execute **set +x** to stop displaying shell expansion.

11. Remove your city alias.

12. What is the location of the **cat** and the **passwd** commands ?

13. Explain the difference between the following commands:

echo

/bin/echo

14. Explain the difference between the following commands:

echo Hello

echo -n Hello

15. Display **A B C** with two spaces between B and C.

(optional)

16. Complete the following command to display exactly

the following output:

4+4 =8

10+14 =24

18. Use **echo** to display the following exactly: ??\\

19. Use one **echo** command to display three words on three lines.

Suggested READ

**What is the difference between UNIX and LINUX?**

Some key differences include:

| LINUX | UNIX |
| --- | --- |
| Open source | Not open source, proprietary |
| Created by Linus Torvalds in 1991 | Created by AT&T Bell Labs in the late 1960s |
| Clone of UNIX |  |
| Default shell is BASH | Default shell is Bourne Shell |
| Threat detection and solutions are quick | Longer wait times in new bug fixing patches |
| Distros of LINUX include Red Hat, Ubuntu, Solaris etc. | Distros of UNIX include HP-UX, AIS, BSD etc. |

**What is a swap space?**

Swap space is space on the hard disk which is a substitute of physical memory. It is used as virtual memory which contains process memory image. Whenever our computer runs short of physical memory it uses its virtual memory and stores information in memory on disk.

**What is the advantage of open source?**

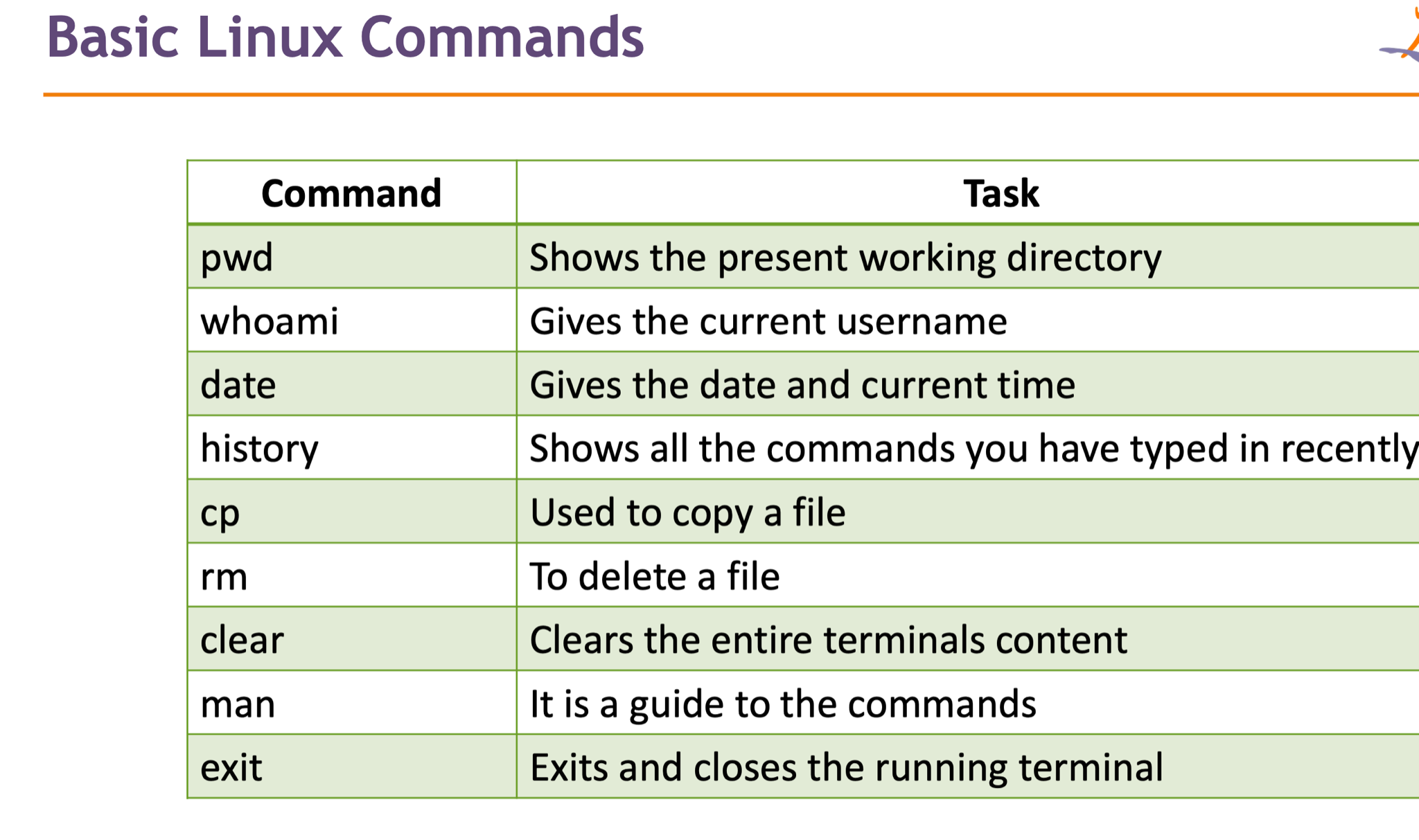
Some advantages of open source include:

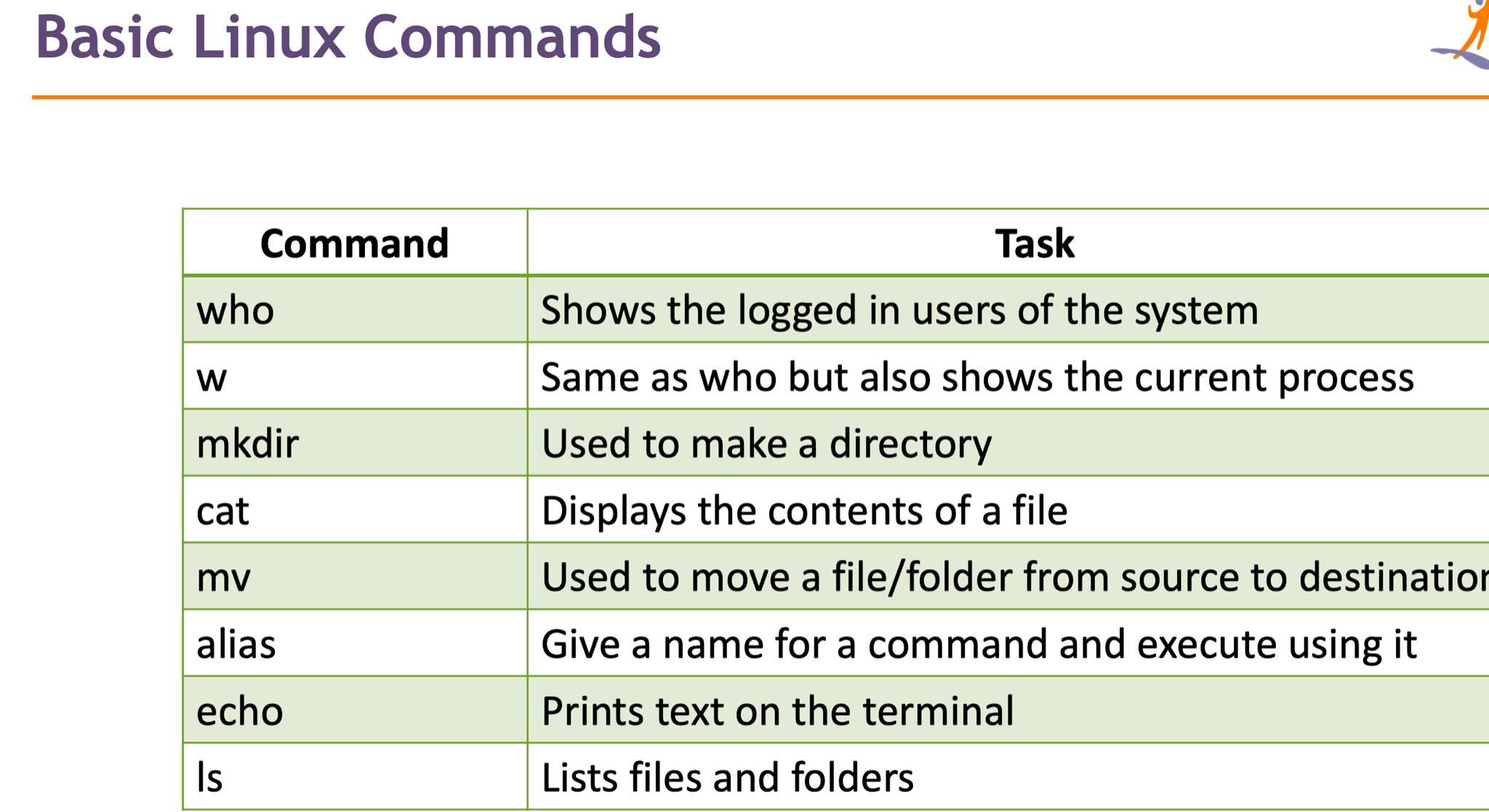
* Community driven - new concepts and securities are implemented faster, better and more effectively
* Transparency - full visibility to the code base so you know exactly what you’re getting
* Reliability - with a worldwide user base targeting a single code base the output tends to be robust, tried and tested thoroughly
* Security - code is much more thoroughly reviewed and vetted by the community resulting in much more secure code
* Freedom lock-in - proprietary software increases the chance of being locked-in by the vendor or technology
* Cost - there are no licensing fees associated with using open source software

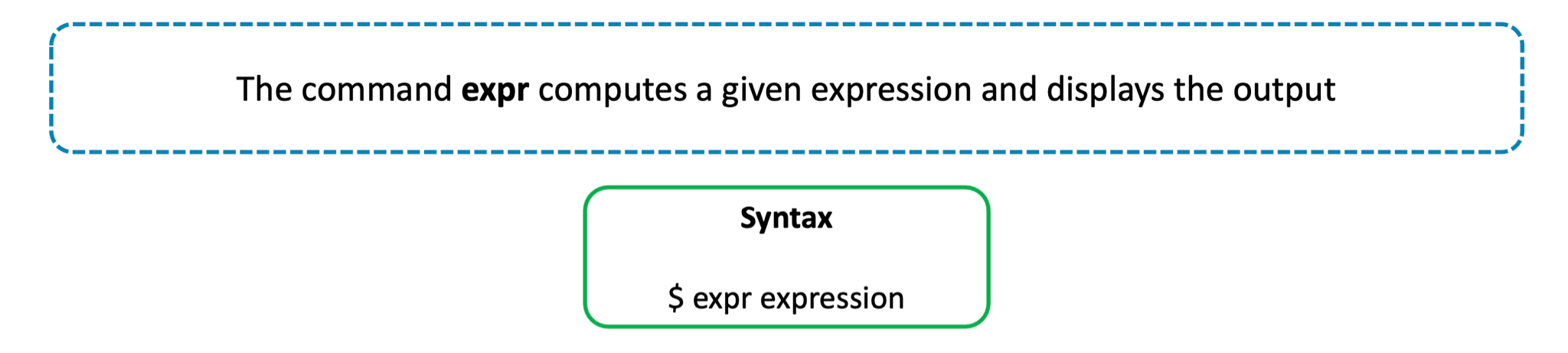
**What are daemons?**

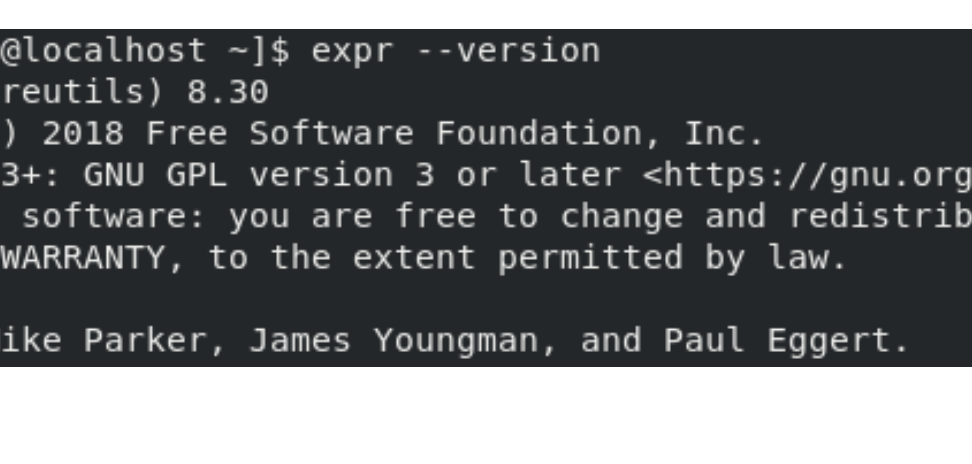
A daemon (also known as background processes) is a Linux or UNIX program that runs in the background, rather than being under the direct control of an interactive user.

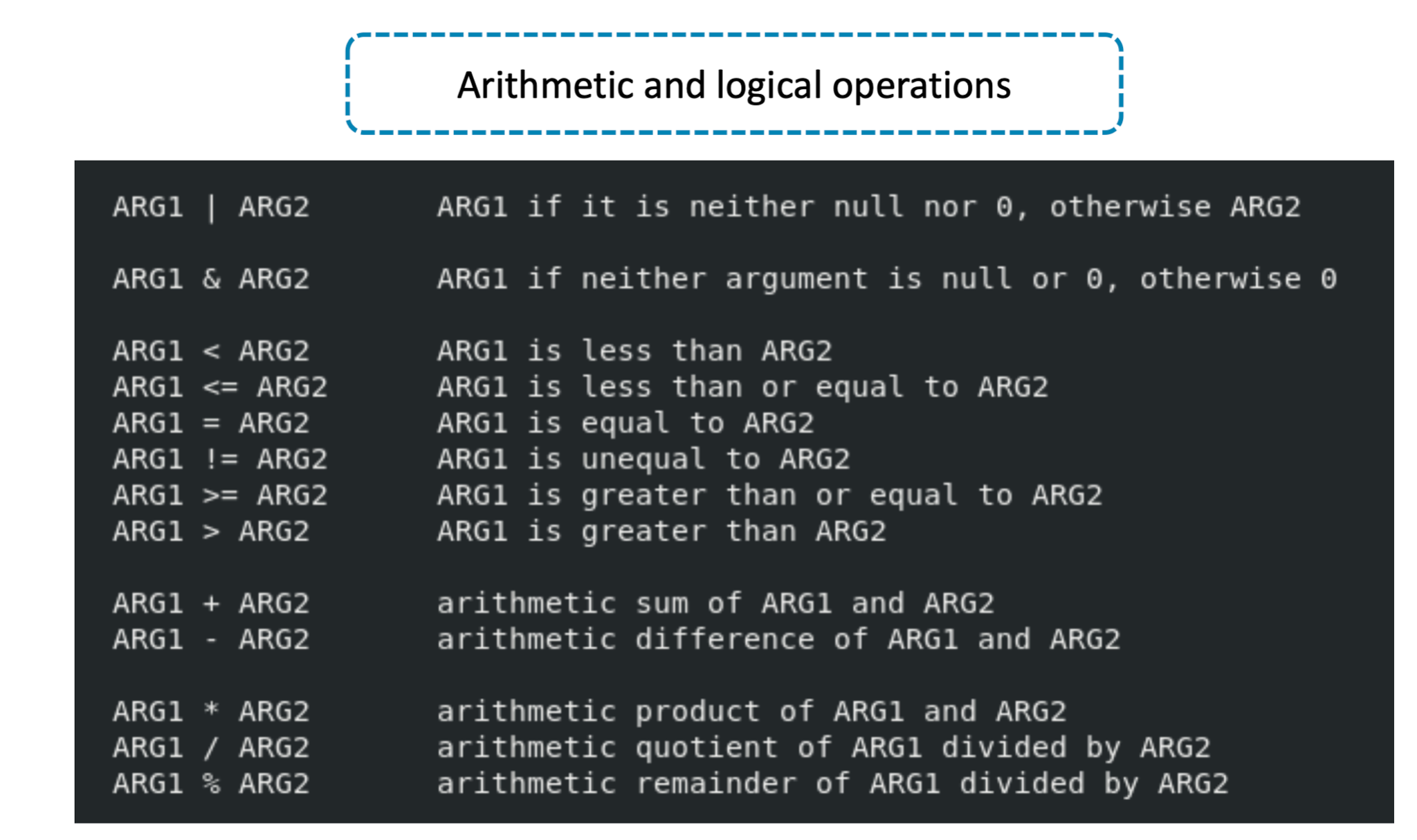
**# REFRENCES**

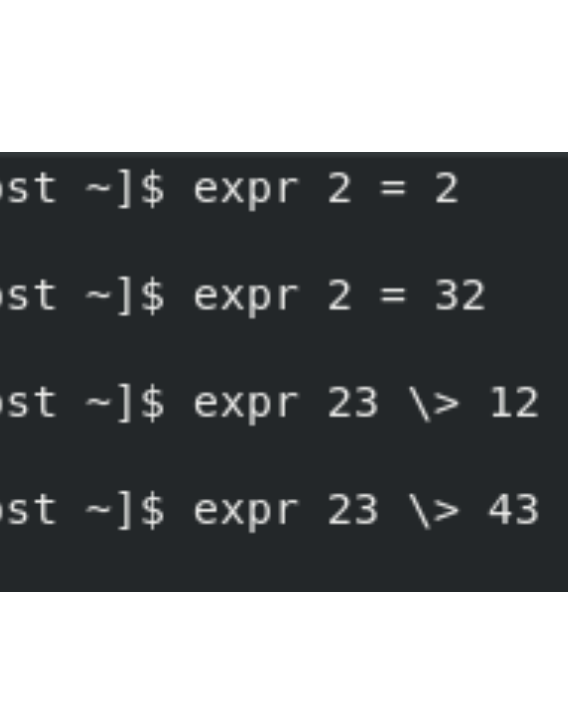
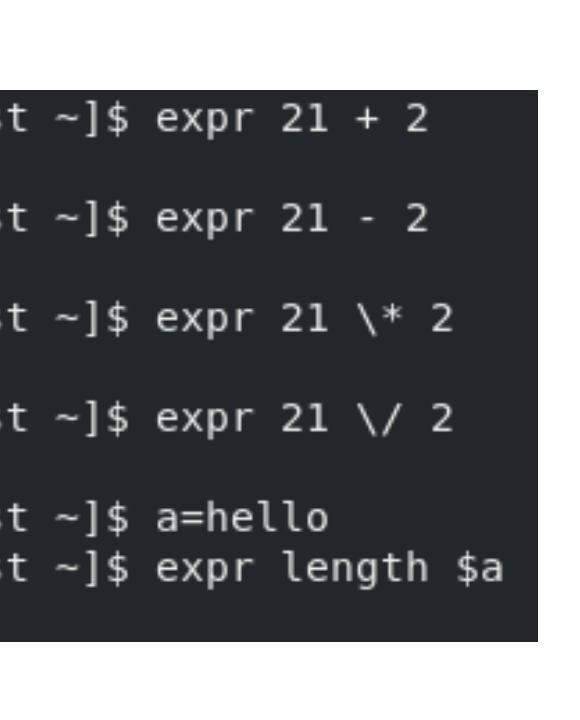








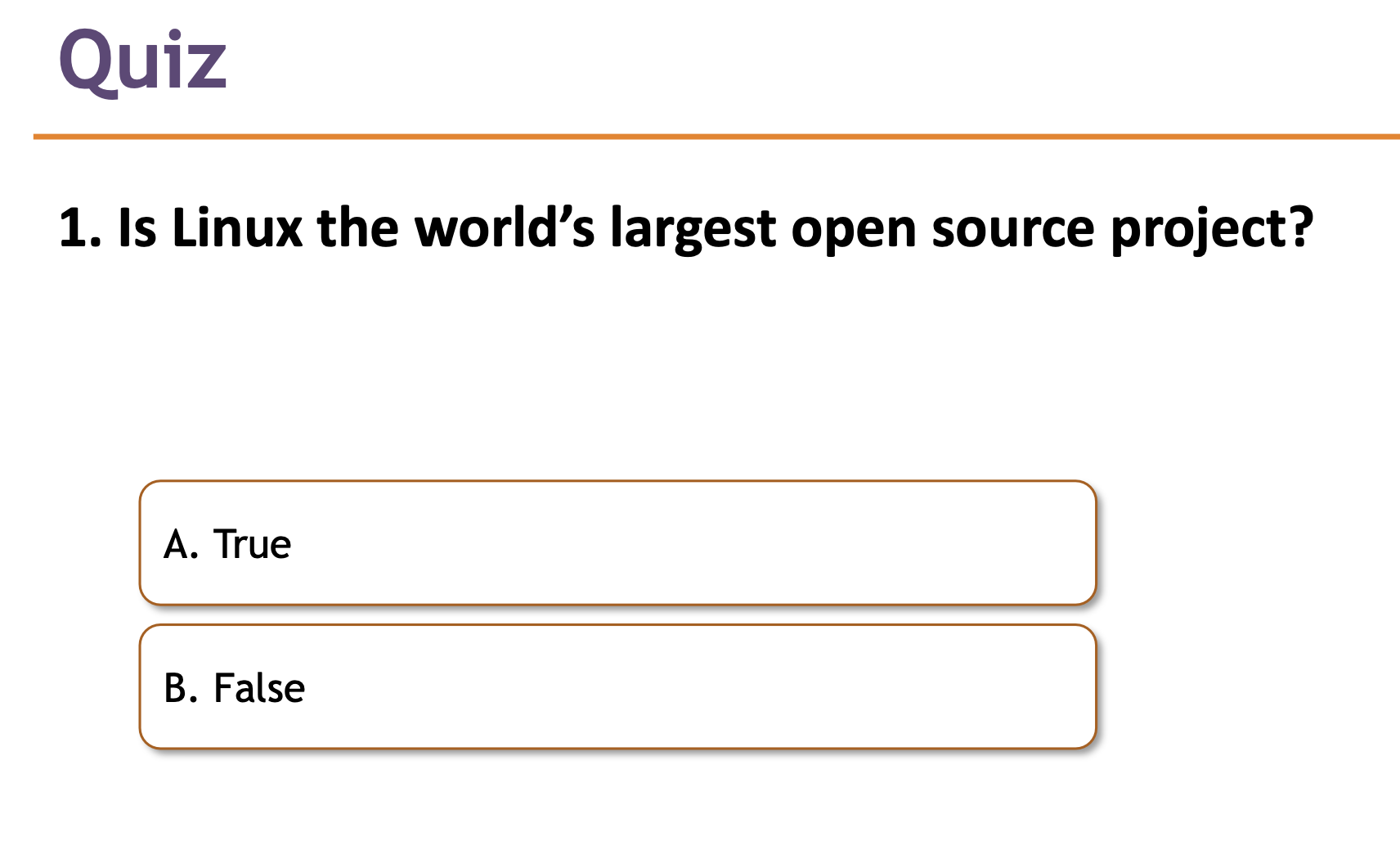


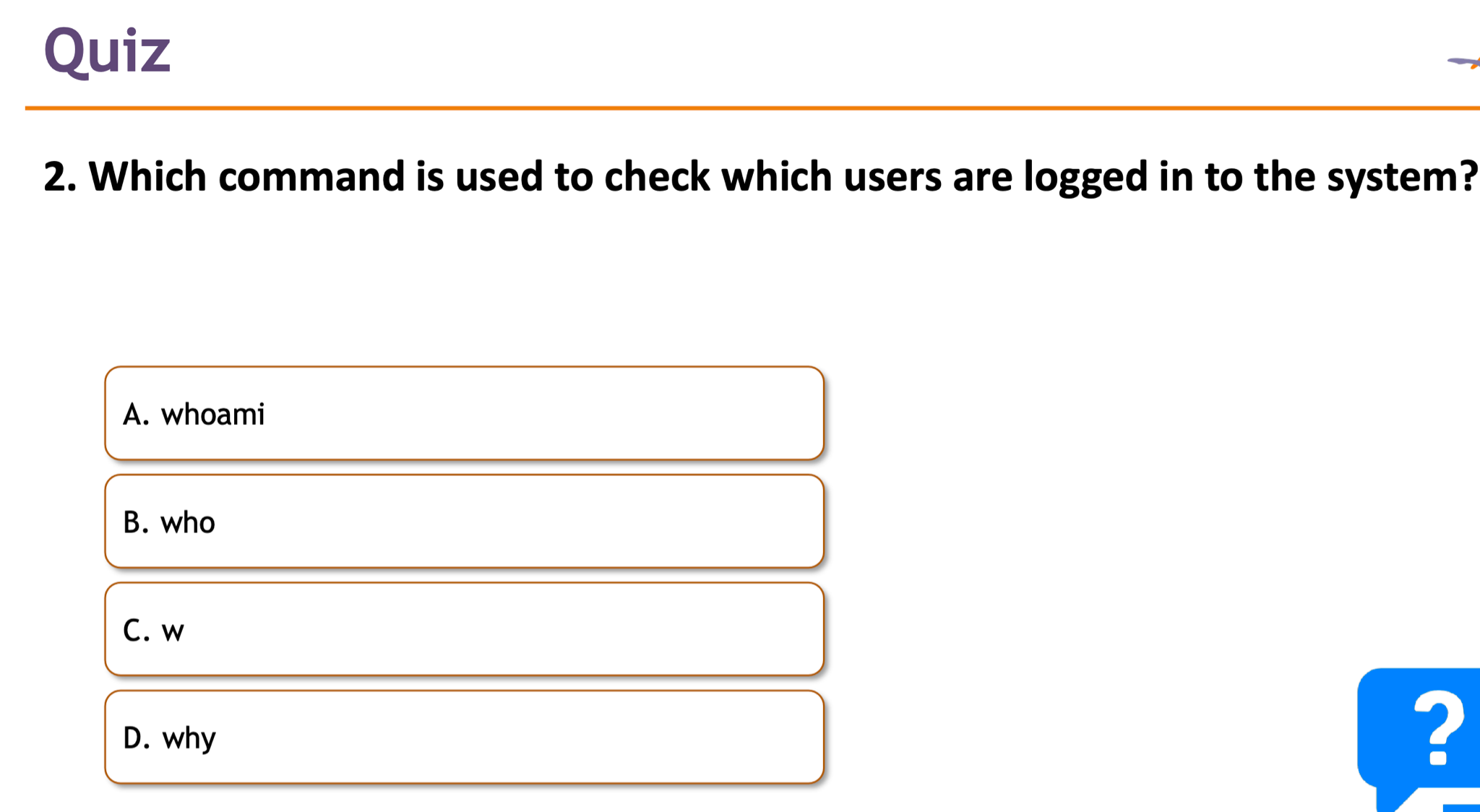


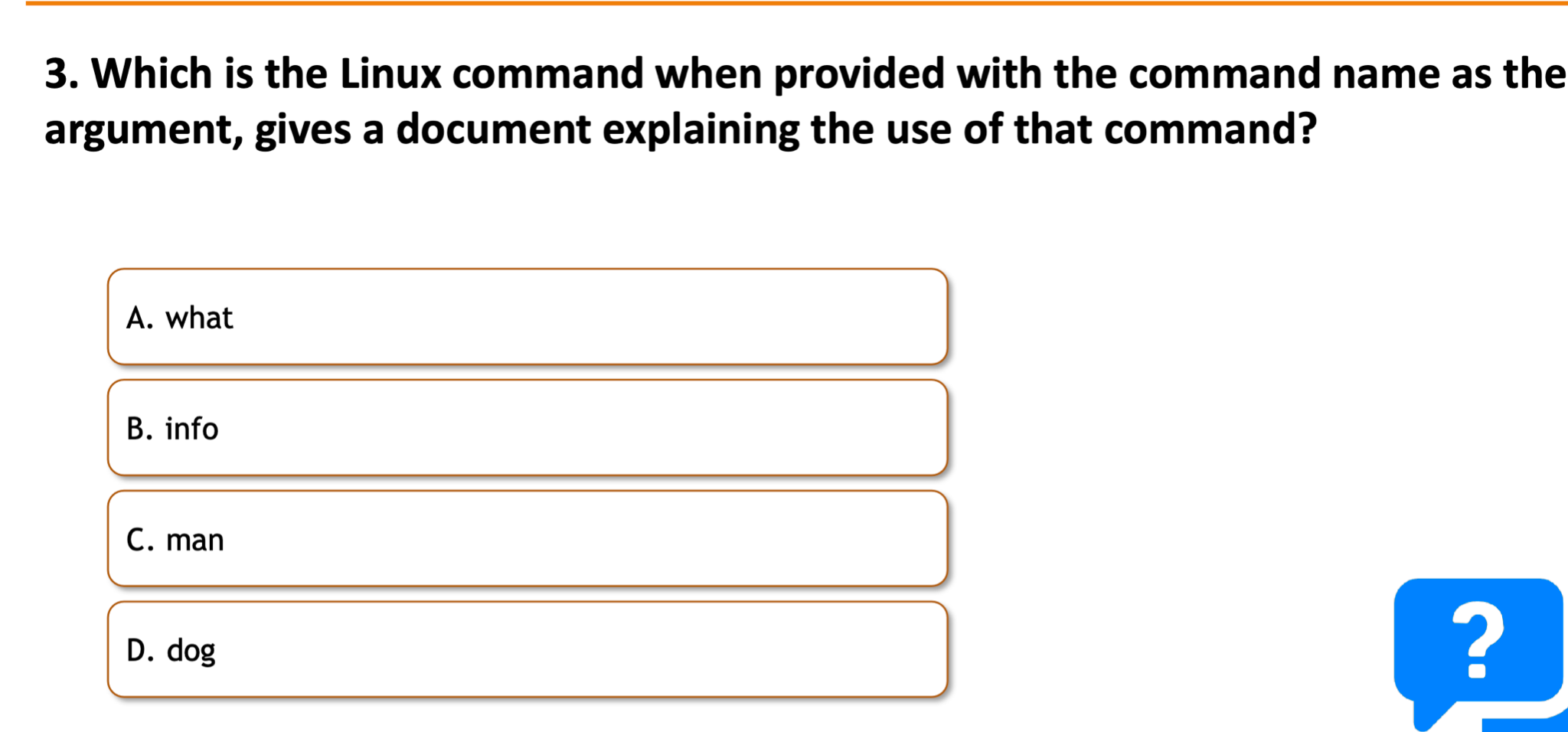
# did you try ‘who’ command (please try out)

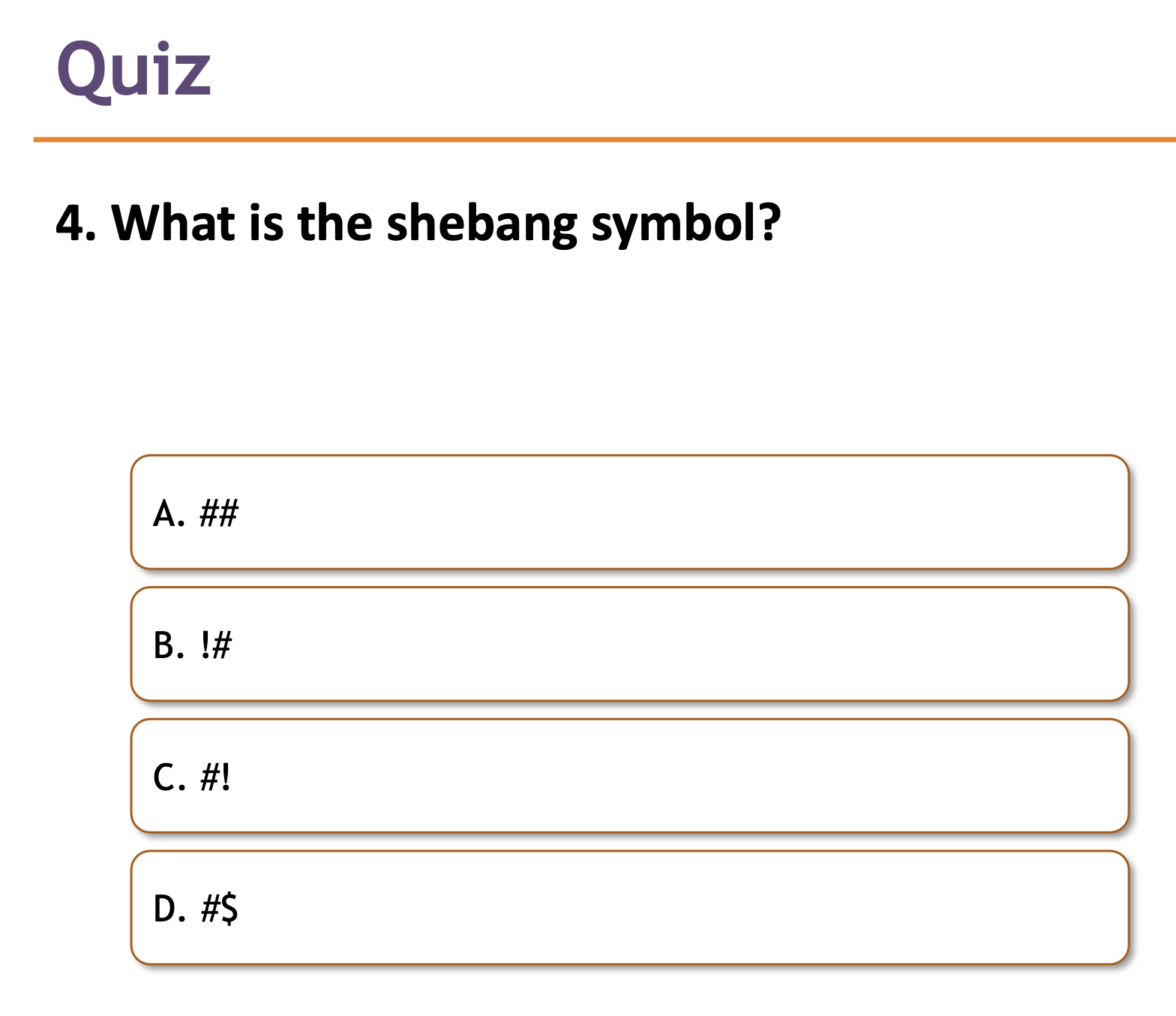
# it shows list of users current logged into system.

**Linux QUIZ**

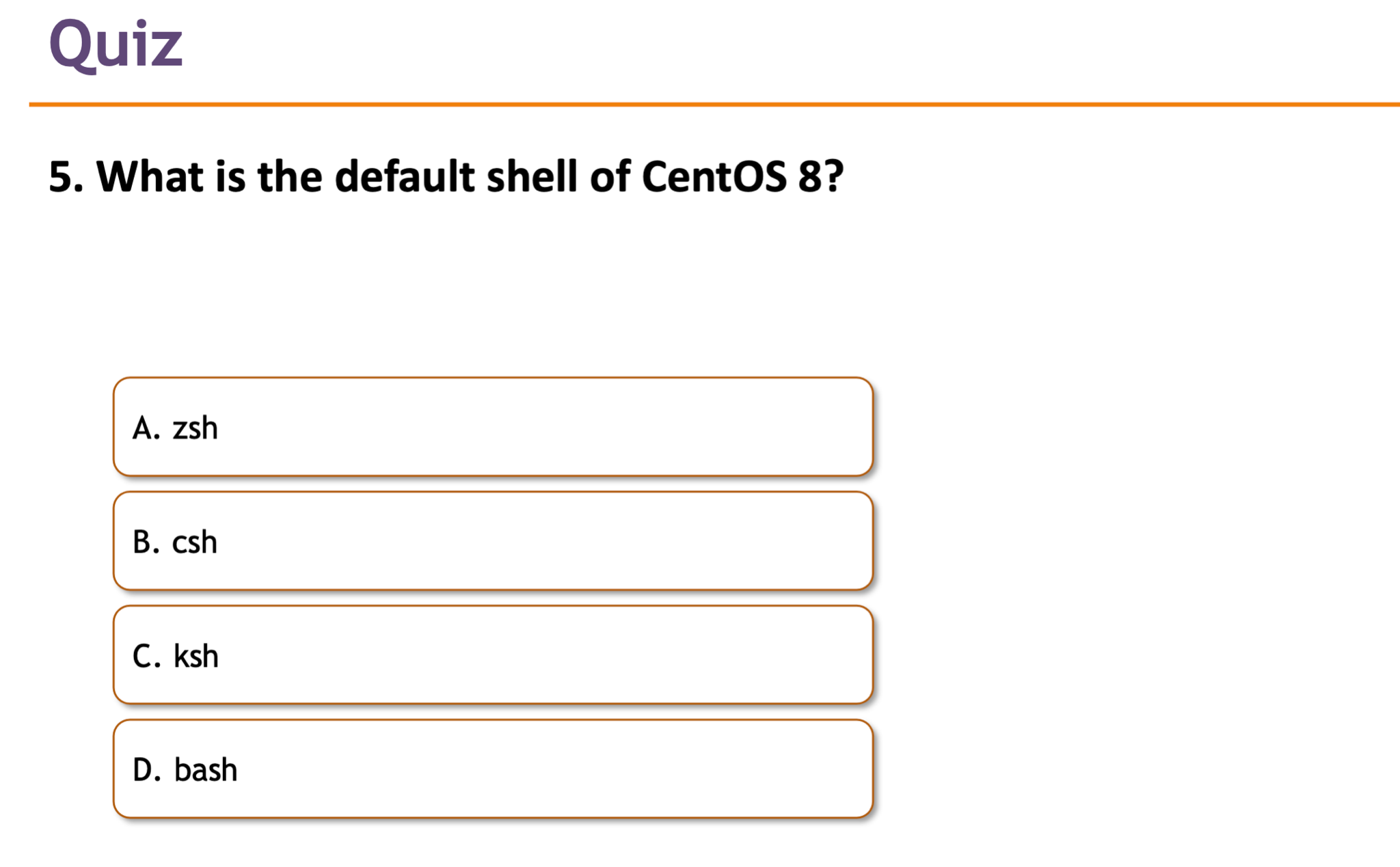




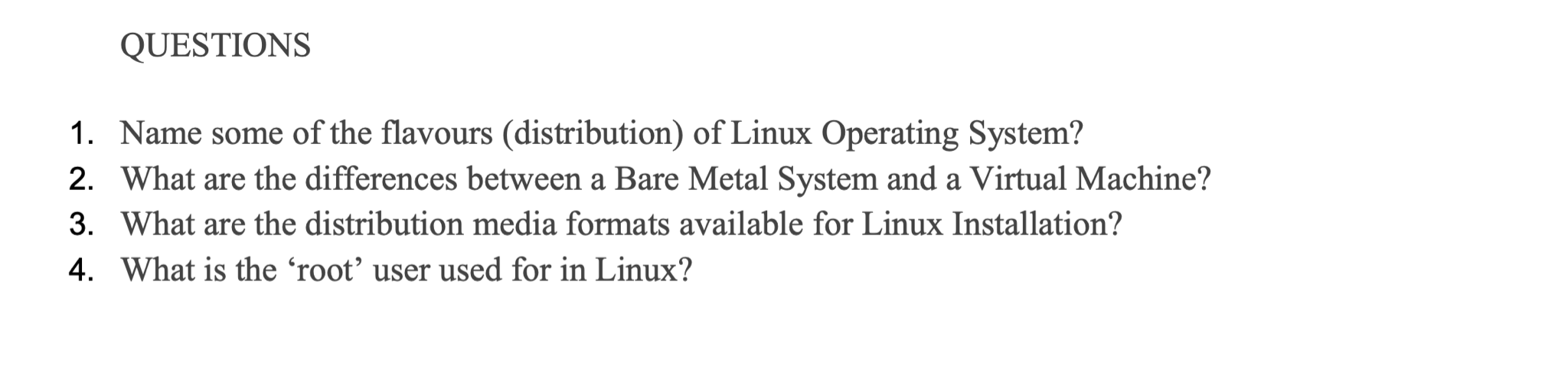




cc



Interview Questions Answers to be Added in your Google Doc.



-- Informational Content---

The Linux Boot Process

A Common Linux Boot Process

* The system is checked by the system BIOS which launches the first stage boot loader on the primary hard disk‟s MBR
* The first stage boot loader is loaded into memory by itself and it then launches the second stage boot loader from the /boot partition
* The kernel is loaded into memory by the second stage boot loader. Necessary modules are loaded by the second stage boot loader which also mounts the root partition read-only
* Control of the boot process is transferred by the kernel to the /sbin/init program. “init” runs with a process ID of 1 and spawns all other processes.
* “init” reads /etc/inittab to find the location of the “sysinit” script.
* Services and user-space tools are loaded by the /sbin/init program which also mounts all  
  partitions listed in /etc/fstab
* The login prompt appears

vim cheat sheet

Press the <ESC> (escape) key to ensure you‟re in normal mode, then: :q! quits without saving

:wq saves and quits (write quit)

x deletesindividualcharacters

i insertstext

dw deletes to the end of a word (d2w deletes two words, d3w deletes three words, etc.)

d$ deletes to the end of a line

dd deletes an entire line (2dd deletes two lines, 23dd deletes 23 lines, etc.)

u undoes the last command

U fixes an entire line

<CTRL>R redoes the command

p puts the last deletion after the cursor

r replaces the character under the cursor

cw is the “change word” command, that deletes the word (from the cursor to the right) and places you in “insert” mode

c$ is the “change line” command, that deletes the line (from the cursor to the right) and places you in “insert” mode

<CTRL>g shows your location in a file

<SHIFT>G moves to the end of the file, [number]<SHIFT>G moves to the line number specified in the command, for example 1<SHIFT>G moves to line #1.

/[search term] searches forward through a file for the search term. For example, “/apache” will search for the next instance of the word “apache” in the file

?[search term] searches backwards through a file for the search term. For example, “?apache” will search for the last instance before the cursor of the word “apache” in the file

:s/[old]/[new] will replace the next instance of “old” with “new”. For example, :s/blue/red will replace the next instance of “blue” with “red”.

:s/[old]/[new]/g will replace the every instance of “old” on the current line with “new”. For example, :s/blue/red will replace the every instance of “blue” with “red”.

:#,#s/[old]/[new]/g will replace every instance of “old” with “new” in the range of lines specified with the # sign.

:! allows you to execute external commands

:set nuturnsonlinenumbering

:nohlsearch turns off highlighting of search terms

**-- Practice Content--**

Exercise:

| Conditional Searching  In this exercise, you will search for a unique text string within a file buried deep within a directory tree. |
| --- |
| 1. Create a deep directory tree with the following command in a terminal window:  #mkdir –p /demo/demo1/demo2/demo3  (The “p” switch creates parent directories when they do not already exist.) |

2. Using “vi”, create a file called “deepfile” in the demo3 directory:

#vi /demo/demo1/demo2/demo3/deepfile

| 1. Enter five lines of text in the file as shown in the screen capture. 2. When you‟re finished, use the key combination of ESC, then :wq to save the file and close “vi”. |  |
| --- | --- |
|  | |
|  | |

| Archiving and compressing  In the following exercise, you will learn how to create a “tarball” and compress it using common archiving and compression tools. |
| --- |
| (please ignore this step)  1. Enter the following commands to switch user to root and change directory to /demo:  $su - Password:p@ss5678 #cd /demo |
| 1. Once again, use the touch command to create three files in /demo: #touch file1 file2 file3 2. Enter the “ls” command to confirm the presence of file1, file2, and file3. |
| 4. Create a tarball (tape archive) of the three files called files.tar with the following command:  #tar cvf files.tar file\* |
| 5. View the contents of the tarball with the following command: #tar tvf files.tar |
| 6. View the size of the tarball with the following command:  #ls –l |
| 7. What is the size of files.tar?  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ It should be about 10,240 bytes. |
| 8. Compress the tarball with the following command: #gzip files.tar |
| 9. Touch the up arrow twice to cycle back to the “ls –l” command and press Enter. |
| 10. Notice that files.tar has been renamed to files.tar.gz, indicating that it‟s a gzip compressed file. What is the size of files.tar.gz?  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ It should now be about 141 bytes. |
| 11. Now, uncompress files.tar.gz with the following command: #gunzip files.tar.gz |
| 12. Touch the up arrow twice to cycle back to the “ls –l” command and press Enter. |
| | 13. Notice that the .gz extension has been removed and files.tar is back to its original size. | | --- | | 14. Remove the tarball with the following command: #rm -rf files.tar | | You can perform similar operations with bzip2 using the commands bzip2 and bunzip2. tar also allows you create a tarball and compress it in a single operation using the “z” switch for gzip or the “j” switch for bzip2: | | 1. While in /demo, execute the following command: #tar cvfj files.tar.bz2 file\* |   2. Use the “ll” command to display the contents of /demo.  3. Notice that files.tar.bz2 is now compressed. What is its size? |
|  |

| 4. What was the size of files.tar.gz?  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  In this case, the tarball compressed with bzip2 should be slightly smaller than the tarball compressed with gzip. |
| --- |
| 5. Use the following command to view the contents of files.tar.bz2: #tar jtvf files.tar.bz2 |
| 6. Notice that, even though compression has been applied to the tarball, you can still view the contents using the “t” and “j” option with tar. |
| 7. You‟re finished with the compression and archiving exercises, so you can delete the tarball:  #rm –f files.tar.bz2 |

**practice: file system tree**

1. Does the file **/bin/cat** exist ? What about **/bin/dd** and **/bin/echo**. What is the type of these files ?

2. What is the size of the Linux kernel file(s) (vmlinu\*) in **/boot** ?

3. Issue the following two commands, and look at the first character of each output line.

ls -l /dev/sd\* /dev/hd\*

ls -l /dev/tty\* /dev/input/mou\*

The first ls will show block(b) devices, the second ls shows character(c) devices. Can you tell the difference between block and character devices ?

4. Use cat to display **/etc/hosts** and **/etc/resolv.conf**. What is your idea about the purpose of these files ?

5. Are there any files in **/etc/skel/** ? Check also for hidden files.

6. Display **/proc/cpuinfo**. On what architecture is your Linux running ?

7. Display **/proc/interrupts**. What is the size of this file ? Where is this file stored ?

8. Can you enter the **/root** directory ? Are there (hidden) files ?

9. Are ifconfig, fdisk, parted, shutdown and grub-install present in **/sbin** ? Why are these binaries in **/sbin** and not in **/bin** ?

10. Is **/var/log** a file or a directory ? What about **/var/spool** ?