Northeastern University

Bioinformatics Program

Introduction to UNIX/Linux and HPC/remote computing

Module 04

Contents

Task I: Introduction to output redirection on Unix

Use the command **echo** to display a text message:

```
$ echo “Hello”
```

Use the command **ls** to display a list the files and directories available in your current directory:

```
$ ls
```

Use the sign **>** (*also called the redirection operator*) to save (or redirect) your output to the file **p01.txt**

```
$ echo “Hello” > p01.txt
```

Use the **ls** command to display a list the files and directories available in your current directory (notice that p01.txt is created):

```
$ ls
```

Use the **cat** command (short for "concatenate") to display the contents of one or more files in the terminal. Let’s use it to display the content of **p01.txt**:

```
$ cat p01.txt
```

Use the sign **>** save the output of **echo “World”** in **p01.txt**:

```
$ echo “World” > p01.txt
```

Display the content of **p01.txt**:

```
$ cat p01.txt
```

Q01- What do you observe? How can you fix this output in order to display both Hello and World in the same file p01.txt? Create a Bash script including the command(s) you used. (Hint [redacted] Copy and paste in txt or command interface to reveal.)

Task II: Use the top command to view system’s processes and resource usage

The command **top** provides real-time information about the system's processes, system resource usage (CPU, Memory), and system performance. The command **top** is a valuable tool in pipeline deployment for monitoring system activity and identifying processes that may be consuming excessive resources.

When you run the top command in your terminal, it presents a table with various columns, including:

PID: Process ID, a unique identifier for each running process.
USER: The user who started the process.
PR: Priority of the process.
NI: Nice value, which affects the process's priority.
VIRT: The total virtual memory used by the process.
RES: Resident memory size (physical RAM) used by the process.
SHR: Shared memory size.
S: Process status (e.g., R for running, S for sleeping, Z for zombie).
%CPU: Percentage of CPU usage by the process.
%MEM: Percentage of physical memory (RAM) usage.
TIME+: Total CPU time used by the process.
COMMAND: The command or program associated with the process.

The top command provides a variety of keyboard shortcuts within its interface to help you customize the display, sort processes, and interact with the live system performance data. Here are some of the most common top keyboard shortcuts:

Sorting Processes:
P: Sort processes by CPU usage (default).
M: Sort processes by memory (RAM) usage.
T: Sort processes by running time.
Changing the Update Interval:
Filtering by User:
u: Prompt for a username. Enter the username to filter processes for that user.
Filtering by Process Name:
o: Prompt for a process name. Enter the process name to filter processes with that name.
Highlighting Field and Column Headers:
y: Toggle highlighting field and column headers.
Displaying Full Command Lines:
c: Toggle between showing short or full command lines for processes.
Killing a Process:
k: Prompt for the PID (Process ID) of the process you want to kill. Enter the PID and press Enter. Be cautious when killing processes.
Changing the Number of Processes Displayed:
n: Prompt for the number of processes to display.
Renicing a Process (Change Priority):
r: Prompt for the PID and a new **nice** value to renice a process. Be cautious when changing process priorities.

Searching for a Process:
/: Prompt for a string to search for in the process list. Use arrow keys to navigate through search results.
Viewing CPU States:
1: Toggle the display of individual CPU core statistics.
Exiting top:
q: Quit the top command and return to the shell prompt.

These keyboard shortcuts make the top command a flexible and powerful tool for monitoring system performance and managing processes in real-time. Remember to use these shortcuts carefully, especially when **killing** or **renicing** processes, as it can impact system stability and performance.

Use the command **top** to display the queue of all running processes:

```
$ top
```

Try all the different keyboard shortcuts mentioned above and observe the results.

Use the command **clear** to clear the content on your screen or use **CTRL/SHIFT/L**.

```
$ clear
```

Use the sign **>** (also called the redirection operator) to save (or redirect) your output to the file **log01.txt** for future inspection:

```
$ top > log01.txt
```

To ouput the content of **top** command into a file sorted by memory usage:

```
$ top -o %MEM -b -n 1 > log01.txt
```

Explanation of the command:

top: Starts the top command.

-o %MEM: Sorts the processes by memory usage.

-b: Runs top in batch mode, which means it runs once and outputs the result to standard output.

-n 1: Specifies that top should run for one iteration and then exit.

> log01.txt: Redirects the standard output to a file named log.txt.

Q02- Create a Bash script that includes the system's processes, system resource usage (CPU, Memory), and system performance report for 3 iterations separated with 1 min each and sorted by process name. (Hint [redacted] Copy and paste in txt or command interface to reveal.)

Task III: I/O Redirection for DevOps

The < symbol is called the "input redirection operator" in Unix-like operating systems, including Linux. It's used to redirect the standard input (stdin) of a command from a file rather than typing input directly from the keyboard**.**

This exercise allows you to practice various redirection operators in a Unix-like shell environment, including `<<` for creating custom messages or text interactively.

1. Create a new text file named `source.txt` with the following content:

```
This is the first line.
This is the second line.
```

2. Using input redirection (`<`), create a new file named `copy.txt` that contains the same content as `source.txt`. Do this without manually typing the content. Use `cat` to display the content of `copy.txt` to verify that it's a copy of `source.txt`.

3. Append a new line to the end of `copy.txt` with the text: "This is the third line."

4. Display the content of `copy.txt` using `cat`.

5. Redirect the standard error (stderr) from the `ls` command to a file named `error.txt`. Run `ls` with an incorrect directory to generate an error message.

6. Append the output of the `date` command to a file named `dates.log` without overwriting its content.

7. Display the contents of `dates.log` to check if the date was appended correctly.

8. Redirect the output of the `ls` command to a file named `files.txt`.

9. Using input redirection (`<<`), create a new file named `message.txt` and enter the following text interactively:

```
This is a custom message.
Please press Ctrl+D to finish.
```

10. Using `cat`, display the content of `message.txt` to verify that your message was entered correctly.

11. Using input redirection (`<`), count the number of lines in the `files.txt` file and save the count to a file named `line_count.txt`.

12. Display the content of `line_count.txt` to view the line count.

13. Finally, clean up your directory by removing the temporary files (`copy.txt`, `error.txt`, `dates.log`, `files.txt`, `message.txt`, and `line_count.txt`) using the `rm` command.