Linux Programming: Assignment-4

Samriddhi Guha | ENG24CY0157 | 3-C (25)

1. A system has a file /etc/passwd. How would you use grep + tee to extract usernames and save them to a file while also displaying them on screen? (CO4)

The *grep* and *tee* commands can be used together to extract usernames of users and save them in a file while also displaying them on the screen. The command *`grep "^[^:]*" /etc/passwd | cut -d: -f1 | tee usernames.txt`* gets the usernames from *`/etc/passwd`*, shows them on the screen, and saves them into a file at the same time.

samriddhi@samriddhi:~$ grep "^[^:]" /etc/passwd | cut -d: -f1 | tee usernames.txt*

This command can be broken down as:

1. `grep "^[^:]*" /etc/passwd` → gets all lines from /etc/passwd.
   Each username is the part before the first `:`.

2. `cut -d: -f1` → cuts (splits the line) when it sees the `:` character and pulls out the first field (username).

3. `tee username.txt` → saves all usernames to a file called `usernames.txt` as it shows them to the screen.

These usernames will be on the screen and saved in the file `usernames.txt`.

2. A binary isn't found in $PATH. How would you use commands (which, find, locate) to troubleshoot and fix the issue? (CO3)

When the shell can't find a binary in `$PATH`, that means there is no directory in the current `$PATH` search path, where the binary can be found. A missing binary most often occurs when a user installs software manually, or has environmental variables misconfigured. The issue can be fixed by the troubleshooting steps mentioned below:

1. Check with `which`:
   - *`which binary_name`*
   - If nothing is returned it is not in a directory that is present in `$PATH`.

2. Search with `find`:
   - *`sudo find / -type f -name "binary_name" 2>/dev/null`*
   - This command searches the entire file system for the file you are searching for.

3. Search with `locate`:
   - *`locate binary_name`*
   - If the `mlocate` database is updated, this command is most likely to return results.

4. Once you have found the binary you need, fix it by adding the binaries directory to `$PATH`, typically running:

- export PATH=$PATH:/path/to/binary
- If you want the change to persist you'll likely need to append this line to `~/.bashrc` or `~/.zshrc`.

3. Write a command pipeline that finds all .log files modified in the last 24 hours in /var/log and saves results into log_report.txt. (CO4)

To find all `.log` files within `/var/log` that were modified in the past 24 hours and outputting the results in a file, we can make use of a command pipeline that utilizes `find` and redirection.

This command pipeline can be executed as:
samriddhi@samriddhi:~$find /var/log -name ".log" -mtime -1 | tee log_report.txt*

- 1. 'find /var/log' will search `/var/log` directory.
- 2. -name "*.log"` will restrict results to files that end in `.log`.
- 3. `-mtime -1` selects any files with modification date within the past 24 hours.
- 4. `| tee log_report.txt` displays the results to screen *and* saves the output to `log_report.txt`.

4. What is the difference between shutdown -r now and reboot? (CO1)

shutdown -r now:

- The `shutdown` command is a safer, system-safe shutdown utility
- `-r` → tells the system to restart after stopping.
- `now` → does so immediately.
- It first notifies logged in users before it takes action.
- It safely stops running processes before it reboots.
- Example:
  samriddhi@samriddhi:~$shutdown -r now

reboot:

- The `reboot` command is a direct system-call to restart.
- It does not do a complete shutdown process first, so may be just a bit quicker.
- However, it does not send the same comments or warnings like `shutdown` does.
- Example:
  samriddhi@samriddhi:~$reboot

The main difference between both would be:

- `shutdown -r now` = a safer, better controlled reboot that has cleaned up processes and has warned all logged in users beforehand.
- `reboot` = it restarts immediately, a faster process but not as graceful process for the action.

5. How can you use the tee command to debug a script that generates both standard output and error messages? (CO4)

During the script debugging process, you may wish to record both standard output (stdout) and error messages (stderr) but still view them on the screen. The `*tee*` command is useful here since it can log output into a file and display it.

samriddhi@samriddhi:~$./script.sh 2>&1 | tee debug.log

In the command,

- `./script.sh` runs the intended script.
- `2>&1` redirects stderr (2) to stdout (1) so the outputs are mixed together.
- `| tee debug.log`takes the now mixed outputs and sends it to the terminal to view, and logs it to `debug.log`.

Some benefits of this command include:

- You can monitor the script in real-time on your terminal.
- A copy of the errors and normal output are also written into `debug.log`.
- Lets you follow along with logs later when deciding what errors in the output were significant.

6. Explain any three real-world applications of Linux in industries. (CO2)

Organizations from all sectors utilize Linux extensively due to its value while also maintaining its security and operational stability. Three applications of Linux in industry are:

1. Servers and Web Hosting:

- Most web servers (Apache, Nginx) use Linux as it is efficient, reliable, and well-protected by security updates and patches.
- Without Linux, global companies like Google, Facebook, and Amazon could not host thousands/millions of websites & applications nearly as effectively.

2. Cybersecurity/Ethical Hacking:

- Linux variants such as Kali Linux and Parrot OS are the standard practice for penetration testing and security audits as acknowledged by ethics committees.
- Linux is widely utilized for applications such as Wireshark, Metasploit, and Nmap; keeping everything open-source and standard beneficial at global scale.

3. Embedded Systems / IoT:

- Selected devices may be powered by Linux such as smart televisions, routers, drones and industrial IoT devices.
- Linux can be modularized for specific hardware, which is a beneficial option for industry standards to customize kernels.

7. Differentiate application, system and utility software in the context of Linux environment. (CO2)

1. System Software:
   - System software is the Linux kernel and operating system components that manage hardware and core processes of the system.
   - Example: Ubuntu, Fedora, or Debian distributions.
   - System software includes the platform software upon which other software runs.

2. Application Software:
   - Application software is software that will allow the users to perform specific tasks or activities.
   - Examples: LibreOffice (for document editing), Firefox (for web browsing), and VLC (for playing audio and video files).
   - Application software will run on top of the system software.

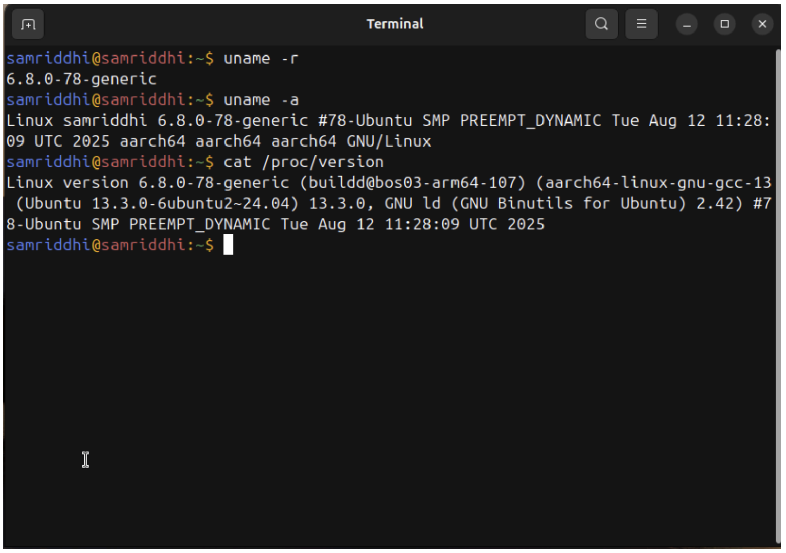
3. Utility Software:
   - Utility software is verification software (and others) that enables system maintenance and organization capabilities.
   - Examples: `top` (system processes), `df` (disk usage), `tar` (backup files).
   - Utility software performs functions in the middle of the different classes of software defined above, and sometimes provides verification capabilities for functions performed by application software.

8. What are the key differences between open-source and proprietary operating systems? (CO2)

Feature	Open-Source OS	Proprietary OS
Source Code Access	Fully accessible; can be modified & shared	Closed; cannot be viewed or modified
Cost	Usually free; optional paid support	Requires purchase or licensing fee
Customization	Highly flexible; can modify system & components	Limited; relies on vendor-provided features
Support	Community-driven, forums, documentation	Vendor-backed, official support
Updates	Frequent and user-implementable	Scheduled by vendor; sometimes paid
Security & Transparency	Open to audits; vulnerabilities quickly addressed	Hidden code; security controlled by vendor
Examples	Linux (Ubuntu, Fedora, Debian)	Windows, macOS

9. Write the command to display the system's kernel version. (CO3)

The command *uname -r* displays the current Linux kernel version.
For more detailed system info, *uname -a* or *cat /proc/version* can also be used.



10. What is the difference between head and tail commands in text processing? (CO1)

Feature	head	tail
Portion Shown	Beginning of the file	End of the file
Default Lines	First 10 lines	Last 10 lines
Syntax Example	head filename	tail filename
Custom Lines	head -n 5 filename	tail -n 5 filename
Common Use	Preview start of files	Monitor logs or view latest entries
Real-Time Monitor	Not applicable	tail -f filename to follow live updates