CS 3307-01 Operating Systems 2

Instructor: Professor Olawale Omoyeni

Name: Ryohei Hayashi

Learning Journal 5

**Summary of Readings and Comparison of Commands for Mac, Windows, and Unix**

This week’s readings focused on basic file manipulation commands and shell scripting. Ahmed (2020) introduced examples of Bash scripts, providing a foundation for file operations and automation in Linux environments. Similarly, Shotts (2019) focused on file and directory manipulation, explaining the flexibility and efficiency of operations in Linux. Brookes (2021) highlighted 16 essential terminal commands for Mac users, while KomalSrivastava (2021) explained file management techniques using the Windows Command Prompt (CMD).

These resources enable us to compare file manipulation commands across Unix/Linux, Mac, and Windows environments:

**Unix/Linux:**

Unix/Linux provides simple yet powerful commands that make file operations efficient. Common commands include:

* `ls`: List directory contents
* `cd`: Change directories
* `cp`: Copy files or directories
* `mv`: Move or rename files and directories
* `rm`: Delete files and directories

(Shotts, 2019)

**Mac:**

Since macOS is Unix-based, it supports most Linux commands (Brookes, 2021). For example, commands like `ls`, `cd`, `cp`, `mv`, and `rm` can be used. Additionally, Mac offers specific utilities such as the `open` command to open files with related applications and `pbcopy`/`pbpaste` for clipboard operations.

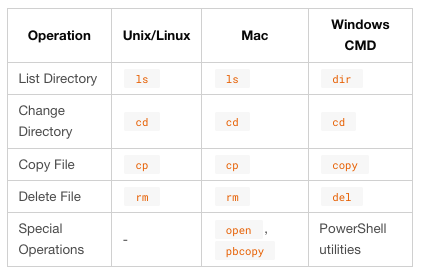
**Windows:**

Windows uses CMD or PowerShell for file manipulation. While the commands differ from Unix/Linux, their functionality is similar (KomalSrivastava, 2021):

* `dir`: List directory contents
* `cd`: Change directories
* `copy`: Copy files or directories
* `move`: Move or rename files and directories
* `del`: Delete files

By using PowerShell, Windows can execute commands similar to those in Unix/Linux, increasing flexibility for system management.

The following table summarizes the differences in file manipulation commands across the three environments:



While Unix/Linux and Mac share almost identical command structures, Windows relies on its own CMD commands. However, the adoption of PowerShell and WSL (Windows Subsystem for Linux) is narrowing the gap between these environments. While Unix/Linux commands are praised for their simplicity and efficiency, Windows has improved its usability and flexibility.

**Preferred Schema and Reasons**

Among the three schemas (Unix/Linux, Mac, and Windows), I prefer Unix/Linux shell scripting for its efficiency, flexibility, and ease of automation.

**1. Efficiency and Simplicity**

Unix/Linux shell scripting is highly efficient for file operations and system management. By combining simple commands like `ls`, `cd`, `cp`, and `mv`, complex tasks can be completed in a short script (Shotts, 2019). Additionally, powerful tools like `grep` and `awk` enable rapid processing of large datasets.

For example, to search for error logs within a file, a single command is sufficient:

grep "ERROR" /var/log/syslog

In Windows, performing the same operation would require PowerShell and multiple steps, highlighting the simplicity of Unix/Linux commands.

**2. Automation and Script Flexibility**

Shell scripts excel at automating repetitive tasks, making them essential for system management and server operations (Ahmed, 2020). For instance, the following script automates daily backup tasks:

```bash

#!/bin/bash

tar -czvf backup\_$(date +%F).tar.gz /home/user/data

```

By scheduling this script using `cron`, backups can run automatically, demonstrating the flexibility of shell scripting (Shell Scripting Tutorial, 2023).

**3. Learning Cost and Compatibility**

Unix/Linux shell scripting skills are highly transferable to other systems, including Mac and Windows (Brookes, 2021). Since macOS is Unix-based, commands are nearly identical, while Windows can emulate Unix/Linux environments through WSL (KomalSrivastava, 2021). This compatibility makes learning Unix/Linux commands highly cost-effective and widely applicable.

**Conclusion**

Unix/Linux shell scripting stands out for its efficiency, flexibility, and automation capabilities. While Windows CMD supports basic operations, it lacks the flexibility of Unix/Linux, and PowerShell requires additional learning. Similarly, although macOS shares the Unix environment, Linux shell scripting offers broader utility and proven use cases. For these reasons, I choose Unix/Linux shell scripting.

**Workplace Use Case**

At my workplace, Unix/Linux command-line tools are the most commonly used for server management, environment configuration, and file manipulation tasks. Below are two specific use cases:

**1. System Monitoring and Log File Analysis**

Monitoring system health requires analyzing logs to detect errors or anomalies. The following commands streamline this process:

* `tail`: Monitors the end of a file in real-time, useful for watching error logs:

tail -f /var/log/nginx/error.log

* `grep`: Extracts specific keywords from log files, such as “404 Not Found”:

grep "404 Not Found" /var/log/nginx/access.log

* `awk`: Formats and extracts specific data, such as IP addresses from logs:

awk '{print $1}' /var/log/nginx/access.log | sort | uniq -c | sort -nr

This command provides a sorted count of access attempts by IP address.

**2. Disk Space Monitoring and Management**

Managing disk space is essential to avoid system downtime. The following commands help monitor and clean up storage:

* `df`: Displays disk usage across file systems:

df -h

* `du`: Checks the size of directories:

du -sh /home/user/data

* `find`: Searches for and removes files older than 30 days:

find /home/user/logs -type f -mtime +30

**3. File Synchronization and Automation**

Automating file backups ensures data integrity. For instance, the `rsync` command synchronizes files to a backup server:

rsync -avz /home/user/data/ user@backupserver:/backup/data/

Combined with `cron`, this command can be scheduled to run automatically, streamlining backup tasks.

**Conclusion**

The Unix/Linux command line is integral to workplace operations, providing powerful tools for monitoring, file management, and automation. While similar tasks can be performed on Windows or Mac, Unix/Linux excels in efficiency, simplicity, and flexibility, making it the preferred choice for my work environment.

Word Count: 793

References

1. Ahmed, H. (2020, September 2). *25 bash script examples.* FOSS Linux. https://www.fosslinux.com/42541/bash-script-examples.htm
2. Brookes, T. (2021, July 24). *16 terminal commands that every Mac user should know.* How-To Geek. https://www.howtogeek.com/729038/16-terminal-commands-every-mac-user-should-know/
3. KomalSrivastava@TWC. (2021, June 8). *Useful commands to manage files and folders through CMD in Windows 11/10*. TheWindowsClub. https://www.thewindowsclub.com/commands-to-manage-files-and-folders-through-cmd
4. Shell scripting tutorial. (2023, January 14). *Shell Scripting.* https://www.shellscript.sh/
5. Shotts, W. (2019). *The linux command line* (5th ed.). No Starch Press.