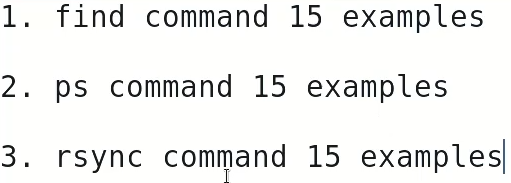
Lecture 22

**Process-daemons-Performance Monitoring-P3**

**Process deamons**

* Interview related and a best knowledge as Linux Admin
* 

15 examples of ps command

1. Display all running processes: **ps -e**
2. Display processes for the current terminal session: **ps -t tty**
3. Display process information for a specific user: **ps -U username**
4. Display process information for a specific process ID: **ps -p pid**
5. Display all processes in a tree-like format: **ps -ejH**
6. Display all processes with the full command line: **ps -ef**
7. Display processes sorted by memory usage: **ps -eo pid,%mem,cmd**
8. Display processes sorted by CPU usage: **ps -eo pid,%cpu,cmd**
9. Display process statistics for a specific process: **ps -o pid,%cpu,%mem,cmd -p pid**
10. Display all processes running on a specific terminal: **ps -t pts/0**
11. Display all processes running on a specific terminal, sorted by memory usage: **ps -o pid,%mem,cmd -t pts/0**
12. Display all processes running on a specific terminal, sorted by CPU usage: **ps -o pid,%cpu,cmd -t pts/0**
13. Display all processes running on a specific terminal, sorted by memory usage and grouped by user: **ps -eo user,pid,%mem,cmd -t pts/0**
14. Display all processes running on a specific terminal, sorted by CPU usage and grouped by user: **ps -eo user,pid,%cpu,cmd -t pts/0**
15. Display all running processes with a real-time update: **watch -n 1 ps -e**

15 examples of find command

1. Find all files in the current directory: **find .**
2. Find all files with a specific name: **find . -name "file.txt"**
3. Find all files with a specific extension: **find . -name "\*.txt"**
4. Find all files that were modified in the last 24 hours: **find . -mtime 0**
5. Find all files that are larger than 100MB: **find . -size +100M**
6. Find all files that are owned by a specific user: **find . -user username**
7. Find all files that are owned by a specific group: **find . -group groupname**
8. Find all files that are executable: **find . -perm /u+x**
9. Find all files that are symbolic links: **find . -type l**
10. Find all files that are directories: **find . -type d**
11. Find all files that have the word "example" in their name: **find . -name "\*example\*"**
12. Find all files that have the word "example" in their content: **find . -exec grep -H "example" {} \;**
13. Find all files and directories that are empty: **find . -empty**
14. Find all files and directories that match a specific permissions: **find . -perm 644**
15. Find all files and directories older than a specific date: **find . -mtime +30**

15 examples of ps command

1. Display all running processes: **ps -e**
2. Display processes for the current terminal session: **ps -t tty**
3. Display process information for a specific user: **ps -U username**
4. Display process information for a specific process ID: **ps -p pid**
5. Display all processes in a tree-like format: **ps -ejH**
6. Display all processes with the full command line: **ps -ef**
7. Display processes sorted by memory usage: **ps -eo pid,%mem,cmd**
8. Display processes sorted by CPU usage: **ps -eo pid,%cpu,cmd**
9. Display process statistics for a specific process: **ps -o pid,%cpu,%mem,cmd -p pid**
10. Display all processes running on a specific terminal: **ps -t pts/0**
11. Display all processes running on a specific terminal, sorted by memory usage: **ps -o pid,%mem,cmd -t pts/0**
12. Display all processes running on a specific terminal, sorted by CPU usage: **ps -o pid,%cpu,cmd -t pts/0**
13. Display all processes running on a specific terminal, sorted by memory usage and grouped by user: **ps -eo user,pid,%mem,cmd -t pts/0**
14. Display all processes running on a specific terminal, sorted by CPU usage and grouped by user: **ps -eo user,pid,%cpu,cmd -t pts/0**
15. Display all running processes with a real-time update: **watch -n 1 ps -e**

15 examples of find command

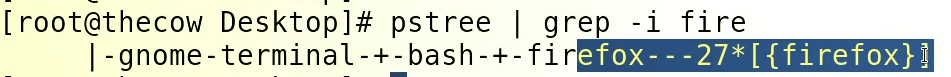
1. Find all files in the current directory: **find .**
2. Find all files with a specific name: **find . -name "file.txt"**
3. Find all files with a specific extension: **find . -name "\*.txt"**
4. Find all files that were modified in the last 24 hours: **find . -mtime 0**
5. Find all files that are larger than 100MB: **find . -size +100M**
6. Find all files that are owned by a specific user: **find . -user username**
7. Find all files that are owned by a specific group: **find . -group groupname**
8. Find all files that are executable: **find . -perm /u+x**
9. Find all files that are symbolic links: **find . -type l**
10. Find all files that are directories: **find . -type d**
11. Find all files that have the word "example" in their name: **find . -name "\*example\*"**
12. Find all files that have the word "example" in their content: **find . -exec grep -H "example" {} \;**
13. Find all files and directories that are empty: **find . -empty**
14. Find all files and directories that match a specific permissions: **find . -perm 644**
15. Find all files and directories older than a specific date: **find . -mtime +30**

15 examples of rsync command

1. Synchronize a local directory with a remote directory: **rsync -avz /local/dir/ user@remote:/remote/dir/**
2. Synchronize a remote directory with a local directory: **rsync -avz user@remote:/remote/dir/ /local/dir/**
3. Only copy new and modified files: **rsync -avz --ignore-existing /local/dir/ user@remote:/remote/dir/**
4. Exclude specific files or directories from synchronization: **rsync -avz --exclude "file1" --exclude "dir1" /local/dir/ user@remote:/remote/dir/**
5. Synchronize a directory and its subdirectories: **rsync -avz --recursive /local/dir/ user@remote:/remote/dir/**
6. Synchronize a directory and keep the same permissions: **rsync -avz --perms /local/dir/ user@remote:/remote/dir/**
7. Synchronize a directory and keep the same timestamps: **rsync -avz --times /local/dir/ user@remote:/remote/dir/**
8. Synchronize a directory with compression: **rsync -avz --compress /local/dir/ user@remote:/remote/dir/**
9. Synchronize a directory over SSH: **rsync -avz -e ssh /local/dir/ user@remote:/remote/dir/**
10. Synchronize a directory with a dry run: **rsync -avzn /local/dir/ user@remote:/remote/dir/**
11. Synchronize a directory with a progress bar: **rsync -avz --progress /local/dir/ user@remote:/remote/dir/**
12. Synchronize a directory and delete files in destination that don't exist in source: **rsync -avz --delete /local/dir/ user@remote:/remote/dir/**
13. Synchronize a directory and only copy files that have changed: **rsync -avz --ignore-times /local/dir/ user@remote:/remote/dir/**
14. Synchronize a directory and limit the transfer rate: **rsync -avz --bwlimit=200 /local/dir/ user@remote:/remote/dir/**
15. Synchronize a directory and keep the same ownership: **rsync -avz --owner /local/dir/ user@remote:/remote/dir/**

* “&”is used to put a process in backgroup within shell 🡪 if terminal is closed the process will also stop
* E.g
* $ gdite&
* 
* $ fg Stands for “foreground” displays a process in foreground
* $ bg
* $ jobs 🡪 to check background process
* “nohup” wil put the process in background but out of the shell 🡪 if terminal is closed the process will not stop
* $ firefoxnohup
* nohup is a POSIX command which means "no hang up". Its purpose is to execute a command such that it ignores the HUP signal and therefore does not stop when the user logs out.
* 

**Tip:-**

* To check process tree of a specific process,
* $ pstree | grep -I <process\_name>
* $ pstree | grep -I firefox
* 
* It shows us that “gnome terminal ” loadeded “bash” 🡪 which loaded “firefox”
* Tip:- if a process can’t be killed or terminated, we need to kill it forcefully with flag “-9” 🡪 for example the bash terminal can’t be terminated with simple “pkill” command so,
* $ pkill -9 <PID> will kill that specific process.
* Tip:- by default “kill” commands uses “-15 no” signals in background.
* Forcefully kill uses “-9” signal followed by <PID>
* # kill 15 (SIGTERM) 🡪 “SIGTERM” is name of that signal
* # kill -8 (SIGKILL)
* How Kill works?
* 🡪 kill signals Kernel and the job is done
* List all kill command signals
* $ kill -l
* SIGTERM , SIGKILL and SIGHUP (1) are commonly used.
* Text

  Description automatically generated
* $ echo && 🡪 alternative of “ps” command
* To kill a process , but it’s title or name is not known, and thousands processes are working.
* Steps
* $ ps -el | grep <process\_name>
* $ ps aux 🡪 also displays additional information, better than “-el” flag
* **How to avoid “fork bombing” is today’s query**
* “daemons” 🡪 services which run in background and never stops until killed or terminated 🡪 a process
* E.g “fdisk” a process which terminates after its job is done
* Firefox is a daemon 🡪 it works as a service, and it has to be stopped.
* Two types of daemons
  + Application daemon 🡪 if killed no impact on system, e.g, firefox, gedit, terminal
  + System daemon 🡪 if killed, system crashed, e.g, init,
* How to capture “top” command in a file for analysis later on
* $ top -n1 <file\_name>