

Outline

- Introduction - history
- Command line basics – getting help
- File system
- Working with files and directories
- More file handling
- The shell revisited
- Monitoring resources

1

Check disk space

2

Measuring disk usage

- Disk usage: `du`
returns the raw number of disk blocks used
- Human readable
 - `-h`: returns size on disk of the given file, in human readable format: K (kilobytes), M (megabytes) or G (gigabytes)
 - `$ du -h <file>`
- Summary
 - `-s`: returns the sum of disk usage of all the files in the given directory.
 - `$ du -sh <dir>`

3

Measuring disk usage

- All
 - lists the sizes of all files and directories in the given file path.
 - `$ du -ah <dir>`
- Time
 - shows the time of the last modification to any file in the directory or subdirectory
 - `$ du -h --time <dir>`

4

Measuring disk space

- Disk filesystem: `df`
full summary of available and used disk space usage of the file system on the Linux system.
- `$ df -h <dir>`
Returns disk usage and free space for the filesystem containing the given directory.
- `$ df -h`
Returns disk space information for all filesystems available in the system. When errors happen, useful to look for full filesystems.

5

Disk usage commands summary

- `du`: display the disk space used by files and directories within a specific directory. It can be used to get a summary of the disk space used by a particular directory and all of its subdirectories.
- `df`: display the amount of disk space available on a file system, as well as the total size and the amount of space used. `df` does not give information on specific files or directories, but rather on the file system as a whole.

6

Process management

9

jobs

- A **process** is an instance of a running program, managed by the Linux kernel. A **job** is a process or group of processes, managed by the shell. The shell provides commands like `jobs`, `fg` and `bg` to interact with jobs. The `ps` command is used to interact with processes.
- A running program launched from the shell is known as a **job**.
 - is started from the command line
 - runs until the program completes its task.
- Each job is always in one of three states:
 - Foreground: Running, with control of the terminal. (default)
 - Background: Running, but not able to read from the terminal.
 - Stopped: Waiting to be resumed.

10

jobs

- Run a command, if you need to free up the terminal, you can stop the process. Ctrl-z stops a process

```
sleep 10000
```

```
^Z
```

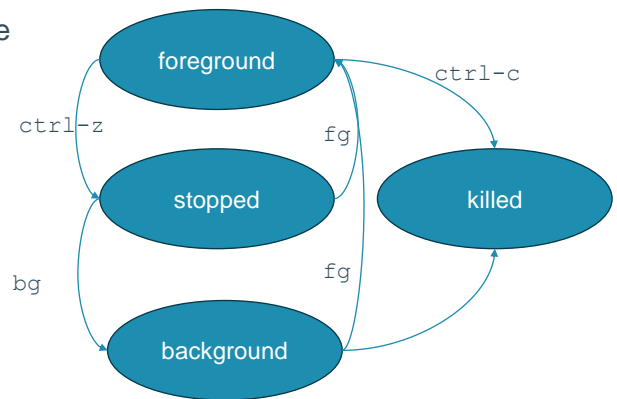
- Get a list of jobs in the background: `jobs`

```
jobs -l
```

Option `-l` shows info on the process id

- Bring a job back to the foreground: `fg`
multiple stopped jobs, specify the job ID

```
fg 2
```

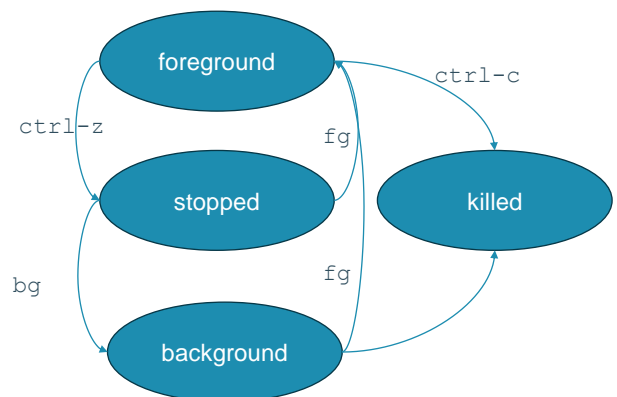


<https://www.baeldung.com/linux/foreground-background-process#:~:text=A%20process%20that%20connects%20to,is%20called%20a%20background%20job.>

12

jobs

- Run a stopped command in the background
 - Get a list of jobs in the background: `jobs`
- ```
jobs -l
```
- Bring a job to the background: `bg`  
multiple stopped jobs, specify the job ID
  - `bg 2`
  - Check with `jobs`: the process is in the background, but running instead of being stopped.



<https://www.baeldung.com/linux/foreground-background-process#:~:text=A%20process%20that%20connects%20to,is%20called%20a%20background%20job.>

13

## &

- `&` is a command line operator that instructs the shell to start the specified program in the background.
  - This allows you to have more than one program running at the same time without having to start multiple terminal sessions.
  - Starting a process in background: add `&` at the end of your line:  
`$ sleep 10000 &`

15

## commands

- There are several commands that are used to control processes:
  - `jobs` - an alternate way of listing your own processes in background or suspended
  - `bg` - put a process in the background
  - `fg` - put a process in the foreground
  - `kill` - send a signal to one or more processes (usually to "kill" a process)
  - `ps` - list the processes running on the system

16

# Process

- Processes carry out tasks within the operating system.
- Several instances of the same program can run at the same time
- Processes are assigned a unique identifier which is used to monitor and control the process (PID)

17

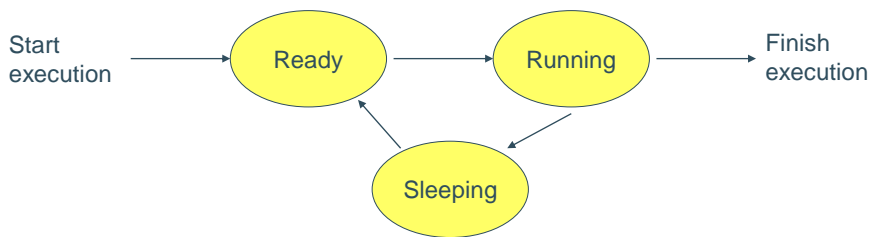
# Process

1. Running (R): When a new process is started, it's placed into the running or runnable state. In the running state, the process takes up a CPU core to execute its code.
2. Interruptible Sleep (S): During process execution, it might come across a portion of its code where it needs to request external resources. Since the process couldn't proceed without the resources, it would stall and do nothing. In events like these, they should give up their CPU cycles to other tasks that are ready to run, and hence they go into an interruptible sleep state.
3. Uninterruptible Sleep (D): The uninterruptible sleeping state waits for the resources to be available before it moves into a runnable state and doesn't react to any signals
4. Stopped (T): suspend a running process and put it into the stopped state. Usually, this is done by sending the SIGSTOP signal to the process. A process in this state will continue to exist until it is killed or resumed with SIGCONT.
5. Zombie (Z): the process completes its lifecycle when it's terminated and placed into a zombie state until its parent process clears it off the process table.

18

# Process

- A process in Linux is a running instance of a program, identified by a unique PID
- A Linux process can be in one of the following states:



19

## ps

- **process status:** display running processes (cfr. Windows Task Manager ctrl-shift-esc)
- `$ps` Display the current user's processes
- `$ps -e` Display all processes running on the system
- `$ps -ef` Display detailed information about running processes
- `$ps -u [USER]` Display processes owned by the specified user
- `$ps -aux`
  - a = show processes for all users
  - u = display the process's user/owner
  - x = also show processes not attached to a terminal

21



# ps

| PID   | TTY   | STAT | TIME | COMMAND      |
|-------|-------|------|------|--------------|
| 14748 | pts/1 | S    | 0:00 | -bash        |
| 14795 | pts/0 | S    | 0:00 | -bash        |
| 14974 | pts/0 | S    | 0:00 | vi test1.txt |
| 14876 | pts/1 | R    | 0:00 | ps ...       |

Process ID      Controlling Terminal name      State:  
S – Sleeping (waiting for input)  
R – Running

Total CPU usage      Name of executable/command

- For the example above, both bash processes, which are the shell of both terminals, are waiting for the input of user. They must be in the sleeping state
- The vi process, which is an editor, is also waiting for the input of user. Hence it is also in sleeping state
- When ps reporting the processes in the system, it is the only process that is running. Hence it is in running state

22

# ps -aux

```
(base) frankvp@CRD-L-08004:~$ ps -aux
```

| USER    | PID | %CPU | %MEM | VSZ   | RSS  | TTY   | STAT | START | TIME | COMMAND                        |
|---------|-----|------|------|-------|------|-------|------|-------|------|--------------------------------|
| root    | 1   | 0.0  | 0.0  | 1520  | 1132 | ?     | S1   | 07:52 | 0:00 | /init                          |
| root    | 19  | 0.0  | 0.0  | 1184  | 360  | ?     | Ss   | 08:25 | 0:00 | /init                          |
| root    | 20  | 0.0  | 0.0  | 1184  | 368  | ?     | R    | 08:25 | 0:00 | /init                          |
| frankvp | 21  | 0.0  | 0.0  | 10832 | 6036 | pts/0 | Ss   | 08:25 | 0:01 | -bash                          |
| root    | 170 | 0.0  | 0.0  | 1184  | 360  | ?     | Ss   | 11:54 | 0:00 | /init                          |
| root    | 171 | 0.0  | 0.0  | 1184  | 368  | ?     | S    | 11:54 | 0:00 | /init                          |
| frankvp | 172 | 0.0  | 0.0  | 10188 | 5140 | pts/1 | Ss+  | 11:54 | 0:00 | -bash                          |
| frankvp | 371 | 0.0  | 0.0  | 8364  | 3204 | pts/0 | T    | 15:49 | 0:00 | nano                           |
| frankvp | 372 | 0.0  | 0.0  | 8624  | 3200 | pts/0 | S    | 15:50 | 0:00 | /bin/bash ./run_hello_world.sh |
| frankvp | 373 | 0.0  | 0.0  | 7236  | 584  | pts/0 | S    | 15:50 | 0:00 | sleep 10                       |
| frankvp | 374 | 0.0  | 0.0  | 10860 | 3352 | pts/0 | R+   | 15:50 | 0:00 | ps -aux                        |

```
(base) frankvp@CRD-L-08004:~$
```

- More fields:
- USER: The effective user
- PID: Process ID
- %CPU: CPU time used divided by the time the process has been running
- %MEM: Ratio of the process's resident set size to the physical memory on the machine
- VSZ: Virtual memory usage of the entire process
- RSS: Resident set size, the non-swapped physical memory that a task has used
- TTY: Controlling terminal associated with the process
- STAT: Process status code
- START: Start time of the process
- TIME: Total CPU usage time
- COMMAND: Name of executable/command

23

## pstree

- Shows the running processes as a tree, making the output more visually appealing.
- The root of the tree is either init or the process with the given pid.
- Options
  - -a: To include command line arguments in output
  - -p: display PIDs for each process name
  - -u: who is the owner/user of a process

```
(base) frankvp@CRD-L-08804:~$ pstree -p
init(1)─┬─init(15)─┬─init(16)─┬─bash(17)─┬─pstree(137)
 │ │ │ │
 │ │ │ └─(init)(10)
 │ │ └─init(106)─┬─init(107)─┬─bash(108)
 │ │ │ │
 │ │ │ └─(init)(10)
 │ └─(init)(10)
 └─(init)(10)

(base) frankvp@CRD-L-08804:~$ pstree -a
init
├─init
│ └─bash
│ └─pstree -a
├─init
│ └─bash
└─(init)

(base) frankvp@CRD-L-08804:~$ pstree -ap
init,1
├─init,15
│ └─init,16
│ └─bash,17
│ └─pstree,139 -ap
├─init,106
│ └─init,107
│ └─bash,108
└─(init),10
```

24

## Signals

- Signals are used for communication between processes
- Use the `kill` command to send various signals to a running process.
- `kill -l`: get a list of all the available signals you can send to a process
- Some common use cases:
  - Interrupting a Process (SIGINT): to interrupt a process, typically via Ctrl+C. Often used to stop a command that's taking too long to complete or to stop a process that's running in the foreground.
  - Terminating a Process (SIGTERM): a generic signal used to terminate a program, can be ignored by the program.
  - Forcefully Terminating a Process (SIGKILL): forcefully terminates a process and cannot be ignored.
  - Pausing and Resuming a Process (SIGSTOP and SIGCONT): SIGSTOP pauses a process, and the SIGCONT signal resumes the execution of a paused process, to move processes between the foreground and background.

26

# kill

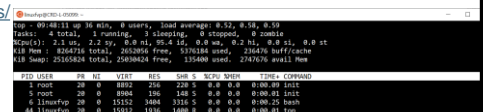
- Send (default) a terminate signal to the given processes.
  - `kill <pid>`
- Send an immediate termination signal. Useful when a process is really stuck.
  - `kill -9 <pid>`
- Suspend a process by sending STOP signal
  - `kill -STOP <PID> or %<jobID>`
  - `kill -19 <PID> or %<jobID>`
- Resume the process, use the CONT flag
  - `kill -CONT <PID> or %<jobID>`
  - `kill -18 <PID> or %<jobID>`

27

# top

- Displays a real-time system status summary. The output displays the amount of system memory(RAM) used for different purposes, percentage of CPU being utilized, swap memory, and other information.
- Press 'z' option will display the running process in color which may help you to identify the running process easily.
- Press 'f' to edit the columns, press space bar to select/deselect

<http://www.thegeekstuff.com/2010/01/15-practical-unix-linux-top-command-examples/>



```
top: 07:42:11 up 36 min, 0 users, load average: 0.52, 0.10, 0.09
tasks: 6 total, 1 running, 3 sleeping, 0 stopped, 0 zombie
procs: 2.1 us, 2.2 sy, 0.0 ni, 95.4 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
Mem: 8264736 total, 2652096 free, 5170164 used, 216476 buff/cache
Mem Swap: 25105824 total, 25030424 free, 135400 used, 2747076 avail Mem

 PID USER NI BT VSZ RSS DM S STAT WCHAN CPUTA COMMAND
 1 root 0 0 4832 256 0 S sleep 0.0% 0.0% init
 5 root 0 8804 180 148 0 S sleep 0.0% 0.0% init
 6 linuxfp 0 15312 3680 316 5 0 S sleep 0.0% 0.0% bash
44 linuxfp 0 15912 1936 1488 8 0 S sleep 0.0% 0.0% top
```

28

# htop

- Displays the data in a more informative and interactive manner.
- The process names are more descriptive and the mouse integration is an extra feature that is not present with the 'top' command.
- Use the mouse to select various columns displayed on the terminal output.

The screenshot shows the htop interface in a terminal window. At the top, system statistics are displayed, including CPU usage (0.0%), memory usage (0.0%), and swap usage (0.0%). Below this, a table of running processes is shown. The processes are sorted by CPU usage, with 'top' being the most active process. The table columns include PID, PPID, USER, CPU, MEM, VSZ, RSS, T, and COMMAND. The bottom of the screen shows a navigation bar with various keys for interacting with the interface.

| PID | PPID | USER | CPU | MEM | VSZ | RSS | T | COMMAND |
|-----|------|------|-----|-----|-----|-----|---|---------|
| 1   | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 2   | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 3   | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 4   | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 5   | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 6   | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 7   | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 8   | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 9   | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 10  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 11  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 12  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 13  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 14  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 15  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 16  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 17  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 18  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 19  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 20  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 21  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 22  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 23  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 24  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 25  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 26  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 27  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 28  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 29  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 30  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 31  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 32  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 33  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 34  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 35  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 36  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 37  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 38  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 39  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 40  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 41  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 42  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 43  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 44  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 45  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 46  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 47  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 48  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 49  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 50  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 51  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 52  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 53  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 54  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 55  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 56  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 57  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 58  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 59  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 60  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 61  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 62  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 63  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 64  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 65  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 66  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 67  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 68  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 69  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 70  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 71  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 72  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 73  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 74  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 75  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 76  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 77  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 78  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 79  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 80  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 81  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 82  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 83  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 84  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 85  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 86  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 87  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 88  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 89  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 90  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 91  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 92  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 93  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 94  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 95  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 96  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 97  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 98  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 99  | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |
| 100 | 0    | root | 0.0 | 0.0 | 0   | 0   | 0 | init    |