

# Outline

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

Check disk space



## 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>`

## 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>`



## 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.

## 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.



# How much space do I have?

- `quota`: command to see all quotas for your directories are, if any

```
3.10.0-957.27.2.el7.x86_64
bash-4.2$ echo $SHELL
/bin/bash
bash-4.2$ ls
Desktop          Videos          mytest.m
Documents        core.23219       openmp
Downloads        inbox            output.txt
MATLABDesktopCreateError.log  intel            pbsnodes-list
Matlab_and_Worker_p2.pdf  java.log.14915  result10hpc.txt
Music            matlab           simple_script_1
Pictures         matlabtest       test
Public           mex_sum_openmp.c test_mex_openmp.m
Templates        mex_sum_openmp.mxa64  testtwo
bash-4.2$ pwd
/vsc-hard-mounts/leuven-user/300/vsc30051
bash-4.2$ quota
quota: error while getting quota from nfs home.usr.hydra.brussel.vsc:/apps/brussel for vsc30051 (id 2530051): Operation not permitted
quota: error while getting quota from nfs data.usr.hydra.brussel.vsc:/data/brussel for vsc30051 (id 2530051): Operation not permitted
quota: error while getting quota from nfs home.usr.hydra.brussel.vsc:/user/brussel for vsc30051 (id 2530051): Operation not permitted
quota: error while getting quota from nfs data.gastly.gent.vsc:/user/data/gent for vsc30051 (id 2530051): Operation not permitted
quota: error while getting quota from nfs apps.gastly.gent.vsc:/apps/gent for vsc30051 (id 2530051): Operation not permitted
quota: error while getting quota from nfs home.gastly.gent.vsc:/user/home/gent for vsc30051 (id 2530051): Operation not permitted
Disk quotas for user vsc30051 (uid 2530051):
Filesystem blocks quota limit grace files quota limit grace
10.118.240.67:/user 2650468 2831156 3145728 61675 90000 1000000
10.118.240.67:/data 34919020 76546048 78643200 4930 9000000 10000000
bash-4.2$
```

## Process management

# jobs

- 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.
- A Linux **job** refers to a task that is executed in the foreground or background of a shell session, while a Linux **process** refers to an instance of a running program in the operating system. A job can consist of one or multiple processes.

# 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
```

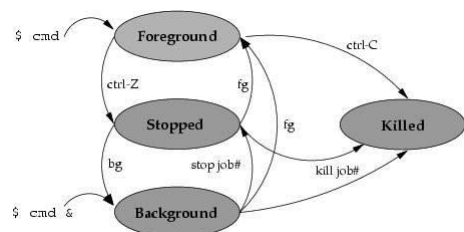
- Get a list of jobs in the background: `jobs`

```
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

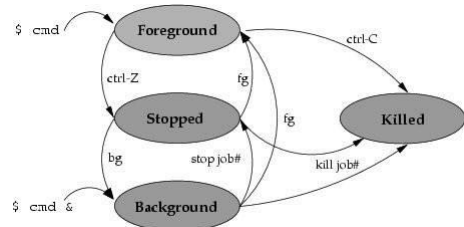
```
fg 2
```



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

# 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.>

# jobs

- Run a command, 2 ways to execute
  - Foreground Processes
    - depend on the user for input
    - also referred to as interactive processes
    - A process that connects to the terminal is called a foreground job. A job is said to be in the foreground because it can communicate with the user via the screen and the keyboard.
  - Background Processes
    - If the background job requires interaction with the user, it will stop and wait until establishing a connection to the terminal. Referred to as non-interactive or automatic processes
    - Daemons: special type of background processes that start at system startup and keep running forever as a service; they don't die.

## &

- `&` 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 &`

## commands

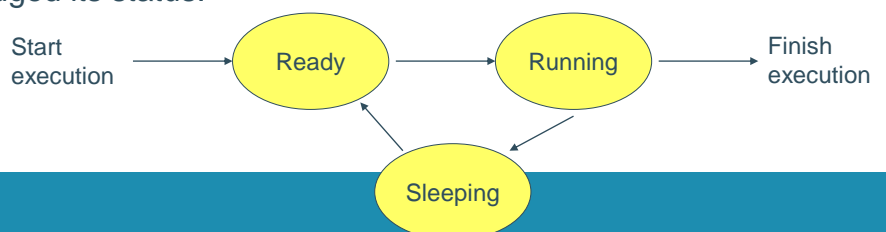
- There are several commands that are used to control processes:
  - `jobs` - an alternate way of listing your own processes in background or suspended
  - `bg` - put a process in the background
  - `fg` - put a process in the foreground
  - `ps` - list the processes running on the system
  - `kill`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

# 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)

# Process

- A program that is claimed to be executing is called a process
- A Linux process can be in one of the following states:
  - Running: the process is currently executing
  - Sleeping: the process is waiting for an event or resource
  - Stopped: the process has been stopped by a signal or command
  - Zombied: the process has completed execution but its parent process has not yet acknowledged its status.





## 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

## 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

## ps -aux

- More fields:

```
(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:~$
```

- USER: The effective user (the one whose access we are using)
- 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

## Process state codes

- R: running or runnable (waiting for the CPU to process it)
- S: Interruptible sleep, waiting for an event to complete, such as input from the terminal
- D: Uninterruptible sleep, processes that cannot be killed or interrupted with a signal, usually to make them go away you have to reboot or fix the issue
- Z: Zombie, are terminated processes that are waiting to have their statuses collected
- T: Stopped, a process that has been suspended/stopped

# kill

- Sends an abort signal to the given processes. Lets processes save data and exit by themselves. Should be used first.

- `$ kill <pid>`

Example:

```
$ kill 3039 3134 3190 3416
```

- `$ kill -9 <pid>`

Sends an immediate termination signal. The system itself terminates the processes. Useful when a process is really stuck.

# 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: 09:42:11 up 36 min, 0 users, load average: 0.52, 0.10, 0.09
tasks: 6 total, 1 running, 3 sleeping, 0 stopped, 0 zombie
MiB Mem: 2.110G total, 252.0M free, 85.41M used, 0.0M sw; 0.0% hdd, 0.0% ss, 0.0% st
MiB Mem: 8264736 total, 2652096 free, 5593184 used, 216476 buff/cache
MiB Swap: 25165824 total, 25030424 free, 135400 used, 2747076 avail Mem

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM     time+  COMMAND
  1 root        20   0   8192   256   228  S   0.0   0.0   0:00.01  init
  5 root        20   0 89804  180   148  S   0.0   0.0   0:00.01  init
  6 linuxfp    20   0 15312  3680  3116  S   0.0   0.0   0:00.25  bash
 44 linuxfp    20   0 15912  1936  1488  R   0.0   0.0   0:00.01  top
```

## 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.

[illegible]