- Log into the Moodle site
- Enter the "Lecture 4" area
- At 14:00, choose "Daily Quiz 3"
- Answer the multiple choice quiz (you have **until 14:10 to finish**)

Let's try!

- In the **remote machine**, inside directory `~jmalves/PE8/`
- Try commands `ls -l` and `cd` on directories:

`another_dir`

`also_a_dir`

`some_dir`

`third_dir`

Now you do it!

Go to the course site and enter **Practical Exercise 8**

Follow the instructions to answer the questions in the exercise

Remember: in a PE, you should do things in practice before answering the question!

How to change permissions

- The `chmod` command allows one to change permissions for a file or a directory (if one has permission to do that, of course!)
- This command can use a symbolic (i.e., letters) or octal (numbers) mode
- The symbolic mode uses letters u, g, o, a, r, w, and x
  - `u`, `g`, `o`, `a` are for user (owner), group, others, and all, respectively
  - `r`, `w`, `x` are read, write, execute, as usual
- The octal mode uses numbers `0` to `7` (eight digits, thus octal)

Description	Symbol	Octal code
Read	r	4
Write	w	2
Execute	x	1
Read and Execute	rx	5 (4 + 1)
Read and Write	rw	6 (4 + 2)
Read, Write and Execute	rwx	7 (4 + 2 + 1)

```
-rwxr-xr-x 1   joe bmp0260 12456   Feb 31 13:37      file.txt
```

Remove read and execute permissions for others (i.e., not user or group):

```
chmod o-rx file.txt
```

symbolic

or

```
chmod 750 file.txt
```

octal

```
-rwxr-x--- 1   joe bmp0260 12456   Feb 31 13:37      file.txt
```

Description	Symbol	Octal code
Read	r	4
Write	w	2
Execute	x	1
Read and Execute	rx	5 (4 + 1)
Read and Write	rw	6 (4 + 2)
Read, Write and Execute	rwx	7 (4 + 2 + 1)

```
-rwxr-xr-x 1   joe bmp0260 12456   Feb 31 13:37      file.txt
```

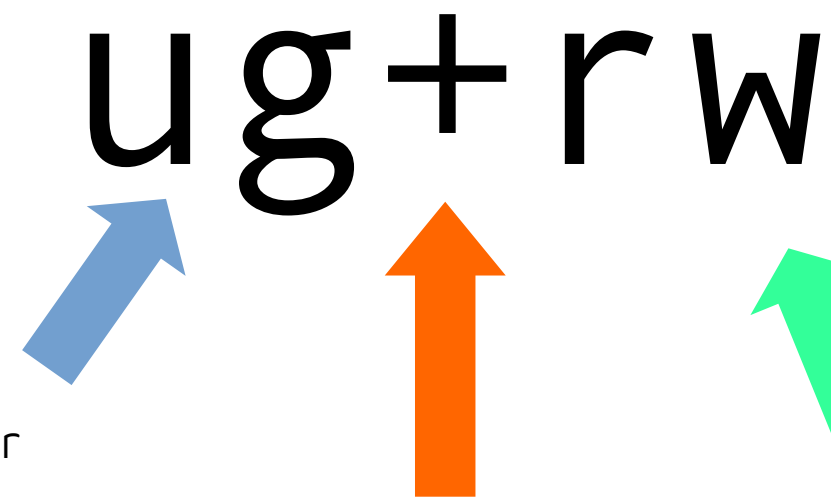
Add write permission for all (i.e., user, group and others):

```
chmod a+w file.txt
```

or

```
chmod 777 file.txt
```

```
-rwxrwxrwx 1   joe bmp0260 12456   Feb 31 13:37      file.txt
```



(So: adding **rw** to user and group, but changing nothing else)

Who should have their permissions changed (more than one, or **a** for all of them, which is the default)

Whether to add (+) or remove (-) the permission(s)

Which permission(s) to add or remove (one or more of r, w, x)

Note: the symbolic mode **only** changes what was explicitly mentioned; so, in the example above, permissions for others are not touched. The octal mode **always changes everything if you are not careful**.

754

Permissions for user (owner of file)

Permissions for users in same group as file

Permissions for others (who are not owner or in file's group)

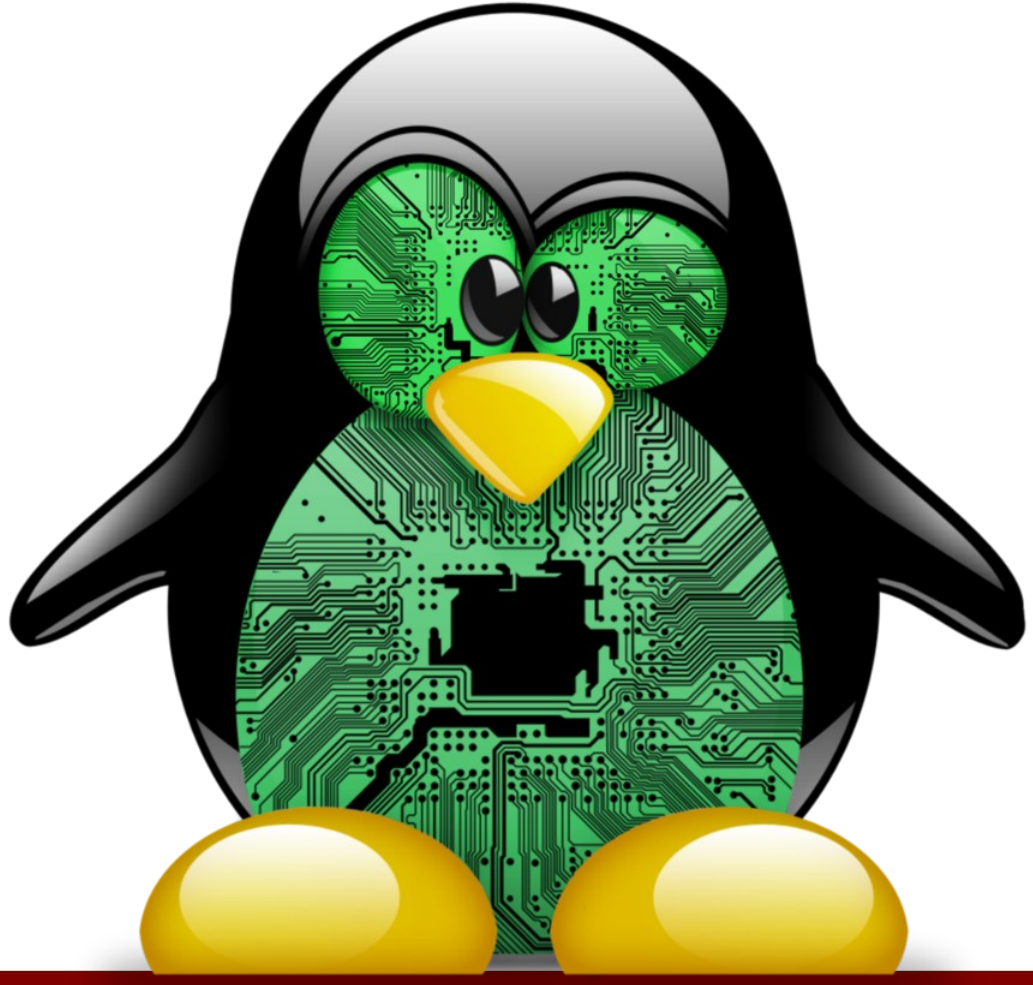
(So: user can **rwx**, group can **rx**, and others can only **r**)

Note: As mentioned above, the octal mode **always changes everything**; there is no way to just add or remove permissions to selected categories of users like it can be done using symbolic mode.

Quiz time!

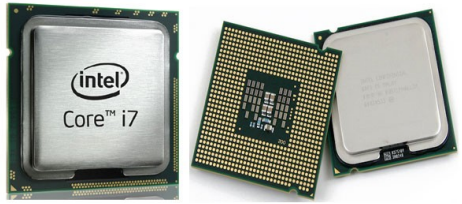
Go to the course page and choose **Quiz 6**

Hardware and system resources



A computer is made of many parts

- The main processor, a.k.a. **CPU** (central processing unit), is a microchip that is the "brain" or "engine" of the computer
- The CPU **gets commands and data** from the working memory, **processes them**, and **returns the results** to working memory
- The speed of a CPU is measured in clock frequency, in cycles per second (in GHz, gigahertz, one billion Hz): the **higher**, the **faster**
- A single CPU can have **more than one** processing **core** (a "mini-CPU" within): that is why you hear about dual-core, quad-core, octo-core etc.



A detour into units and multipliers

- The **bit** is a binary digit: **`0`** or **`1`**
- A **byte** is a set of **8 bits**, e.g., **`01001010`**

Base 10 is not the only way to count

Base 10
(decimal)

74

7×10^1 4×10^0

Digits: 0 1 2 3 4 5 6 7 8 9

Base 2
(binary)

0×2^5 1×2^3 1×2^1

1001010

1×2^6 0×2^4 0×2^2 0×2^0

Digits: 0 1

Base 10 is not the only way to count

Base 8
(octal)

112

1×8^2 1×8^1 2×8^0

Digits: 0 1 2 3 4 5 6 7

Base 16
(hexadecimal)

4A

4×16^1 10×16^0

Digits: 0 1 2 3 4 5 6 7 8 9 A B C D E F

A detour into units and multipliers

- The **bit** is a binary digit: **0** or **1**
- A **byte** is a set of **8 bits**, e.g., **01001010**

- There are **two** kinds of (very similar and often confused) multipliers: those that are **base-10** and those that are **base-2**

2^0	1	1 bit
2^3	8	1 byte
2^{10}	1,024	1 kibibit
2^{20}	1,048,576	1 mebibit
2^{30}	1,073,741,824	1 gibibit
2^{40}	1,099,511,627,776	1 tebibit

A detour into units and multipliers

2^0	1	1 bit
2^3	8	1 byte
2^{10}	1,024	1 kibibit
2^{20}	1,048,576	1 mebibit
2^{30}	1,073,741,824	1 gibibit
2^{40}	1,099,511,627,776	1 tebibit

10^0	1	1 bit
10^3	1,000	1 kilobit
10^6	1,000,000	1 megabit
10^9	1,000,000,000	1 gigabit
10^{12}	1,000,000,000,000	1 terabit

A detour into units and multipliers

Powers of 1,024 (or 2)

bit	b	byte	B
kibibit	Kib	kibibyte	KiB
mebibit	Mib	mebibyte	MiB
gibibit	Gib	gibibyte	GiB
tebibit	Tib	tebibyte	TiB

Powers of 1,000 (or 10)

kilobit	kb	kilobyte	kB
megabit	Mb	megabyte	MB
gigabit	Gb	gigabyte	GB
terabit	Tb	terabyte	TB

A computer is made of many parts

- The **RAM** (random access memory) is a set of chips that work as the **working memory** of the computer

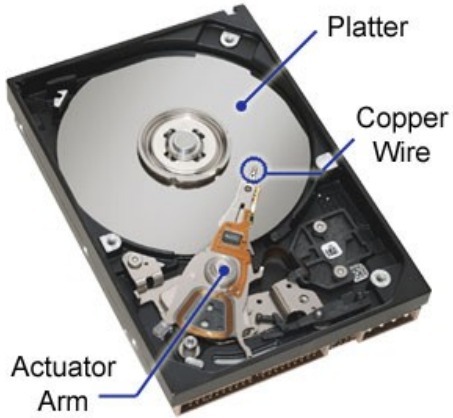
- The RAM stores commands and data that are **being actively used**: everything you see on the screen is in a form of RAM (either the main system memory or a video card's memory)

- The amount of data that can go into RAM is measured in multiples of **bits or bytes** (Gb or GB for gigabits or gigabytes, respectively)

- Data in RAM **disappears** when the computer is turned off

A computer is made of many parts

- The **hard-disk drive** (HDD) is the component that stores data in the long term
- A form of virtual RAM (called **swap memory** in Linux or virtual memory in Windows) can be stored in the HDD – it is **VERY slow** compared to RAM
- The amount of data that can go into an HDD is measured in multiples of **bits or bytes** (Gb or GB for gigabits or gigabytes, respectively, or T for terabits and terabytes)
- Data in the HDD does **not disappear** (we hope) when the computer is turned off



What's in YOUR penguin?

- It is important, specially when running analysis programs that demand **lots of computational power**, to be aware of how much you have available of each kind of system resource
- For example:
    - The program you are going to run generates very **large files**; how much hard drive space do you have? How many drives do you have and where are they in the file system?
    - And/or the program needs **a lot of working memory**; how much total (and free) RAM do you have?
    - And/or the program needs **a lot of processing power**; how many CPUs do you have, and how many of them are idle?

Remember!

- Linux is a time-sharing operating system, so if you are in a large server, chances are that you are **not the only user** of the system; someone else might be logged in (and possibly running commands) at the same time as you!

- So it is necessary to be a **good neighbor** and not hog system resources

What's in YOUR penguin?

- There are several commands that let you **check your system resources**
- Let's start by looking at disk space with the **df** (disk free) command
- First, run **df** by itself and see what you get...

```
Filesystem     1K-blocks    Used Available Use% Mounted on
/dev/xvda1      19620732 3305824  15295160  18% /
udev               10240       0     10240   0% /dev
tmpfs             810700   82340    728360  11% /run
tmpfs            2026744       0   2026744   0% /dev/shm
tmpfs               5120       0      5120   0% /run/lock
tmpfs            2026744       0   2026744   0% /sys/fs/cgroup
/dev/xvdb1      82437808   57092  78170080   1% /data
```

- Of course you can tell **df** **which place**, like a directory, that you want to examine (e.g.: **df .** will look at free disk space in the disk where you are)

Quiz time!

Go to the course page and choose **Quiz 7**

df -h

```
Filesystem      Size  Used Avail Use% Mounted on
/dev/xvda1       19G  3.2G   15G  18% /
udev             10M     0   10M   0% /dev
tmpfs           792M   81M  712M  11% /run
tmpfs           2.0G     0  2.0G   0% /dev/shm
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           2.0G     0  2.0G   0% /sys/fs/cgroup
/dev/xvdb1       79G   56M   75G   1% /data
```

-h, --human-readable
print sizes in powers of 1024 (e.g., 1023M)

df

```
Filesystem     1K-blocks    Used Available Use% Mounted on
/dev/xvda1      19620732 3305824  15295160  18% /
udev               10240       0     10240   0% /dev
tmpfs             810700   82340    728360  11% /run
tmpfs            2026744       0   2026744   0% /dev/shm
tmpfs               5120       0      5120   0% /run/lock
tmpfs            2026744       0   2026744   0% /sys/fs/cgroup
/dev/xvdb1      82437808   57092  78170080   1% /data
```

Identifying disks

- As you saw in the output above, it is not immediately obvious which lines are **actual disks** and which aren't

- First of all, they are **partitions**, not really disks (a disk can have more than one partition; they would all have similar names, with different numbers, like **`/dev/sda1`**, **`/dev/sda2`** etc.)

- **If you know file system types**, you can tell which ones are "real" file systems where data lives and which aren't by using the **`--print-type`** option:
  `df --print-type`

- A rule-of-thumb that usually works: lines that **start with `/dev/`** are probably "real" partitions; in our remote machine: **`/dev/xvda1`** and **`/dev/xvdb1`** (two different disks, see the **a** and **b** before the **1**)

More fine-grained disk usage info

- Frequently, you want to see **how much space each directory** is taking (say, your disk is full and you want to find the culprits)
- In this case, looking at disk space with the **du** (disk usage) command is the best bet on the command-line
- First, run **du** by itself and see what you get...
- ...potentially a lot of lines, ending in something like this:

```
… … …
8    ./.config/htop
12   ./.config
168  ./store/BLING/htdocs/aux/bling
172  ./store/BLING/htdocs/aux
180  ./store/BLING/htdocs
100  ./store/BLING/cgi
656  ./store/BLING
824  ./store
4    ./.emacs.d
853564   .
```

- Without any arguments, **du** will give you file size information for the directory where it was run, **recursively** (i.e., it will enter directories inside directories and examine them too)
- One can also tell it where to examine, e.g.:

```
du /etc/
```

- One can also use the `-h` option to get more "human" output:

```
du -h /etc/
```

- If one is interested only in the grand total size of the directory, in other words, how much total space it is taking, one can use the **`-s`** switch:

```
du -h -s
```

```
du -hs
```

Now you do it!

Go to the course site and enter **Practical Exercise 9**

Follow the instructions to answer the questions in the exercise

Remember: in a PE, you should do things in practice before answering the question!

RAM: Working memory

- RAM is usually the system component that is most crucial to computer speed: if you have little RAM, things get **very slow, very quickly**

- The working memory keeps needed parts of the operating system in itself while it is running, plus programs and user data that are needed at the time

- If your RAM gets full, a part of the hard-disk drive (or SSD, solid state drive, in more modern computers) is used as auxiliary working memory

- To see how much RAM you have, and how much is free, use the command called... `free`

```
             total       used       free     shared    buffers     cached
Mem:       4053488     573424    3480064      82772     150896     323688
-/+ buffers/cache:      98840    3954648
Swap:       901116          0     901116
```

Quiz time!

Go to the course page and choose **Quiz 8**

RAM: Working memory

- The info displayed by **`free`** is parsed from a **system file**:

  `/proc/meminfo`

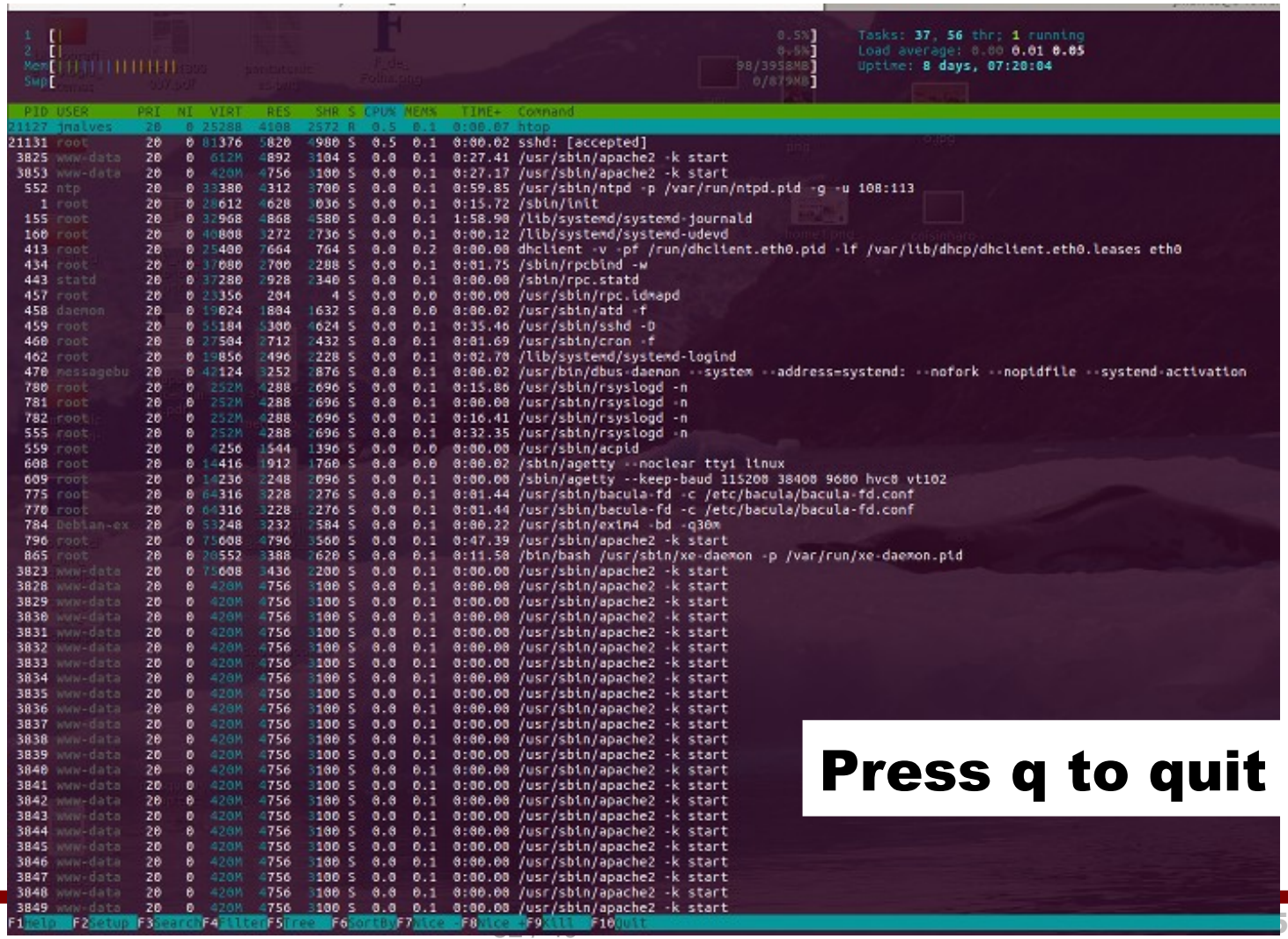
- Take a look at it:

```
more /proc/meminfo
```

- But the output of **`free`** (and **`/proc/meminfo`**) is **ugly, and static**

- Let's use a nicer program... **`htop`**

htop (or top)


Tasks: 37, 56 thr; 1 running
Load average: 0.00 0.01 0.05
Uptime: 8 days, 07:20:04
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
21127 jmalves 20 0 25288 4108 2572 R 0.5 0.1 0:00.07 htop
21131 root 20 0 81376 5820 4980 S 0.5 0.1 0:00.02 sshd: [accepted]
3825 www-data 20 0 612M 4892 3104 S 0.0 0.1 0:27.41 /usr/sbin/apache2 -k start
3853 www-data 20 0 420M 4756 3100 S 0.0 0.1 0:27.17 /usr/sbin/apache2 -k start
552 ntp 20 0 33380 4312 3700 S 0.0 0.1 0:59.85 /usr/sbin/ntpd -p /var/run/ntpd.pid -g -u 108:113
1 root 20 0 28612 4628 3036 S 0.0 0.1 0:15.72 /sbin/init
155 root 20 0 32968 4868 4580 S 0.0 0.1 1:58.90 /lib/systemd/systemd-journald
160 root 20 0 40808 3272 2736 S 0.0 0.1 0:00.12 /lib/systemd/systemd-udevd
413 root 20 0 25400 7664 764 S 0.0 0.2 0:00.00 dhclient -v -pf /run/dhclient.eth0.pid -lf /var/lib/dhcp/dhclient.eth0.leases eth0
434 root 20 0 37680 2700 2288 S 0.0 0.1 0:01.75 /sbin/rpcbind -w
443 statd 20 0 37280 2928 3340 S 0.0 0.1 0:00.00 /sbin/rpc.statd
457 root 20 0 23356 204 4 S 0.0 0.0 0:00.00 /usr/sbin/rpc.idmapd
458 daemon 20 0 19024 1804 1632 S 0.0 0.0 0:00.02 /usr/sbin/atd -f
459 root 20 0 55184 5300 4624 S 0.0 0.1 0:35.46 /usr/sbin/sshd -D
460 root 20 0 27504 2712 2432 S 0.0 0.1 0:01.69 /usr/sbin/cron -f
462 root 20 0 19856 2496 2228 S 0.0 0.1 0:02.70 /lib/systemd/systemd-logind
470 messagebu 20 0 42124 3252 3876 S 0.0 0.1 0:00.02 /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation
780 root 20 0 252M 4288 2696 S 0.0 0.1 0:15.86 /usr/sbin/rsyslogd -n
781 root 20 0 252M 4288 2696 S 0.0 0.1 0:00.00 /usr/sbin/rsyslogd -n
782 root 20 0 252M 4288 2696 S 0.0 0.1 0:16.41 /usr/sbin/rsyslogd -n
555 root 20 0 252M 4288 2696 S 0.0 0.1 0:32.35 /usr/sbin/rsyslogd -n
559 root 20 0 4256 1544 1396 S 0.0 0.0 0:00.00 /usr/sbin/acpid
608 root 20 0 14416 1912 1760 S 0.0 0.0 0:00.02 /sbin/agetty --noclear tty1 linux
609 root 20 0 14236 2248 2096 S 0.0 0.1 0:00.00 /sbin/agetty --keep-baud 115200 38400 9600 hvc0 vt102
775 root 20 0 64316 3228 2276 S 0.0 0.1 0:01.44 /usr/sbin/bacula-fd -c /etc/bacula/bacula-fd.conf
770 root 20 0 64316 3228 2276 S 0.0 0.1 0:01.44 /usr/sbin/bacula-fd -c /etc/bacula/bacula-fd.conf
784 Debian-ex 20 0 53248 3232 3584 S 0.0 0.1 0:00.22 /usr/sbin/exim4 -bd -q30m
796 root 20 0 75608 4796 3560 S 0.0 0.1 0:47.39 /usr/sbin/apache2 -k start
865 root 20 0 20552 3388 2620 S 0.0 0.1 0:11.58 /bin/bash /usr/sbin/xe-daemon -p /var/run/xe-daemon.pid
3823 www-data 20 0 75608 3436 2200 S 0.0 0.1 0:00.00 /usr/sbin/apache2 -k start
3828 www-data 20 0 420M 4756 3100 S 0.0 0.1 0:00.00 /usr/sbin/apache2 -k start
3829 www-data 20 0 420M 4756 3100 S 0.0 0.1 0:00.00 /usr/sbin/apache2 -k start
3830 www-data 20 0 420M 4756 3100 S 0.0 0.1 0:00.00 /usr/sbin/apache2 -k start
3831 www-data 20 0 420M 4756 3100 S 0.0 0.1 0:00.00 /usr/sbin/apache2 -k start
3832 www-data 20 0 420M 4756 3100 S 0.0 0.1 0:00.00 /usr/sbin/apache2 -k start
3833 www-data 20 0 420M 4756 3100 S 0.0 0.1 0:00.00 /usr/sbin/apache2 -k start
3834 www-data 20 0 420M 4756 3100 S 0.0 0.1 0:00.00 /usr/sbin/apache2 -k start
3835 www-data 20 0 420M 4756 3100 S 0.0 0.1 0:00.00 /usr/sbin/apache2 -k start
3836 www-data 20 0 420M 4756 3100 S 0.0 0.1 0:00.00 /usr/sbin/apache2 -k start
3837 www-data 20 0 420M 4756 3100 S 0.0 0.1 0:00.00 /usr/sbin/apache2 -k start
3838 www-data 20 0 420M 4756 3100 S 0.0 0.1 0:00.00 /usr/sbin/apache2 -k start
3839 www-data 20 0 420M 4756 3100 S 0.0 0.1 0:00.00 /usr/sbin/apache2 -k start
3840 www-data 20 0 420M 4756 3100 S 0.0 0.1 0:00.00 /usr/sbin/apache2 -k start
3841 www-data 20 0 420M 4756 3100 S 0.0 0.1 0:00.00 /usr/sbin/apache2 -k start
3842 www-data 20 0 420M 4756 3100 S 0.0 0.1 0:00.00 /usr/sbin/apache2 -k start
3843 www-data 20 0 420M 4756 3100 S 0.0 0.1 0:00.00 /usr/sbin/apache2 -k start
3844 www-data 20 0 420M 4756 3100 S 0.0 0.1 0:00.00 /usr/sbin/apache2 -k start
3845 www-data 20 0 420M 4756 3100 S 0.0 0.1 0:00.00 /usr/sbin/apache2 -k start
3846 www-data 20 0 420M 4756 3100 S 0.0 0.1 0:00.00 /usr/sbin/apache2 -k start
3847 www-data 20 0 420M 4756 3100 S 0.0 0.1 0:00.00 /usr/sbin/apache2 -k start
3848 www-data 20 0 420M 4756 3100 S 0.0 0.1 0:00.00 /usr/sbin/apache2 -k start
3849 www-data 20 0 420M 4756 3100 S 0.0 0.1 0:00.00 /usr/sbin/apache2 -k start
F1Help F2Setup F3SearchF4FilterF5Tree F6SortByF7Nice -F8Nice +F9Kill F10Quit
Press q to quit

Processing power: CPUs

- **The more** CPU cores you have available, **the faster** your system will be – each core will be able to run a different program simultaneously with the other cores

- The information in file **/proc/cpuinfo** tells you what is installed:

```
more /proc/cpuinfo
```

```
processor : 0
vendor_id : GenuineIntel
cpu family: 6
model     : 47
model name: Intel(R) Xeon(R) CPU E7- 2870  @ 2.40GHz
stepping  : 2
microcode : 0x37
cpu MHz       : 2396.884
cache size    : 30720 KB
… … …
```

Processing power: CPUs

- **A nicer way** to ask the system for the CPU information is the **`lscpu`** program
- It reads the information in file **`/proc/cpuinfo`** and formats it in a (slightly) more human-friendly:

`lscpu`

```
Architecture:            x86_64
CPU op-mode(s):          32-bit, 64-bit
Byte Order:              Little Endian
CPU(s):                  2
On-line CPU(s) list:     0,1
Thread(s) per core:      1
Core(s) per socket:      1
Socket(s):               2
NUMA node(s):            1
Vendor ID:               GenuineIntel
… … …
```

Now you do it!

Go to the course site and enter **Practical Exercise 10**

Follow the instructions to answer the questions in the exercise

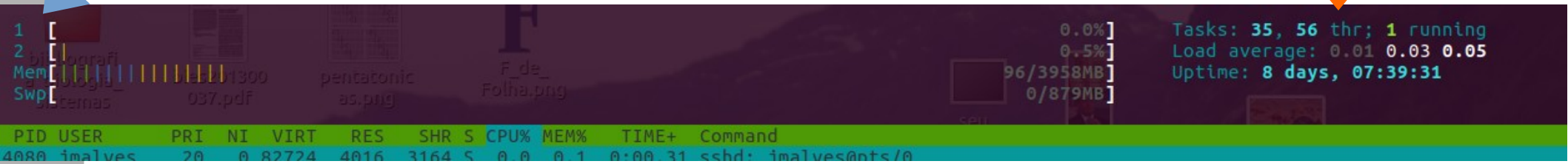
Remember: in a PE, you should do things in practice before answering the question!

Processing power: how much is in use or not?

- One could have many CPU cores installed, but many programs running and occupying them all!
- To check, one can use two different ways: `htop` (or **`top`**, if **`htop`** is not installed) or `uptime`
- `uptime` is static, and shows for how long the system has been running since the last time it was rebooted, and the **system load** (average number of programs running) over the past **1, 5, and 15 minutes**
- **`htop`** is dynamic and shows the different CPU cores being used in real time; it also displays average system load information anyway

Average load and uptime information

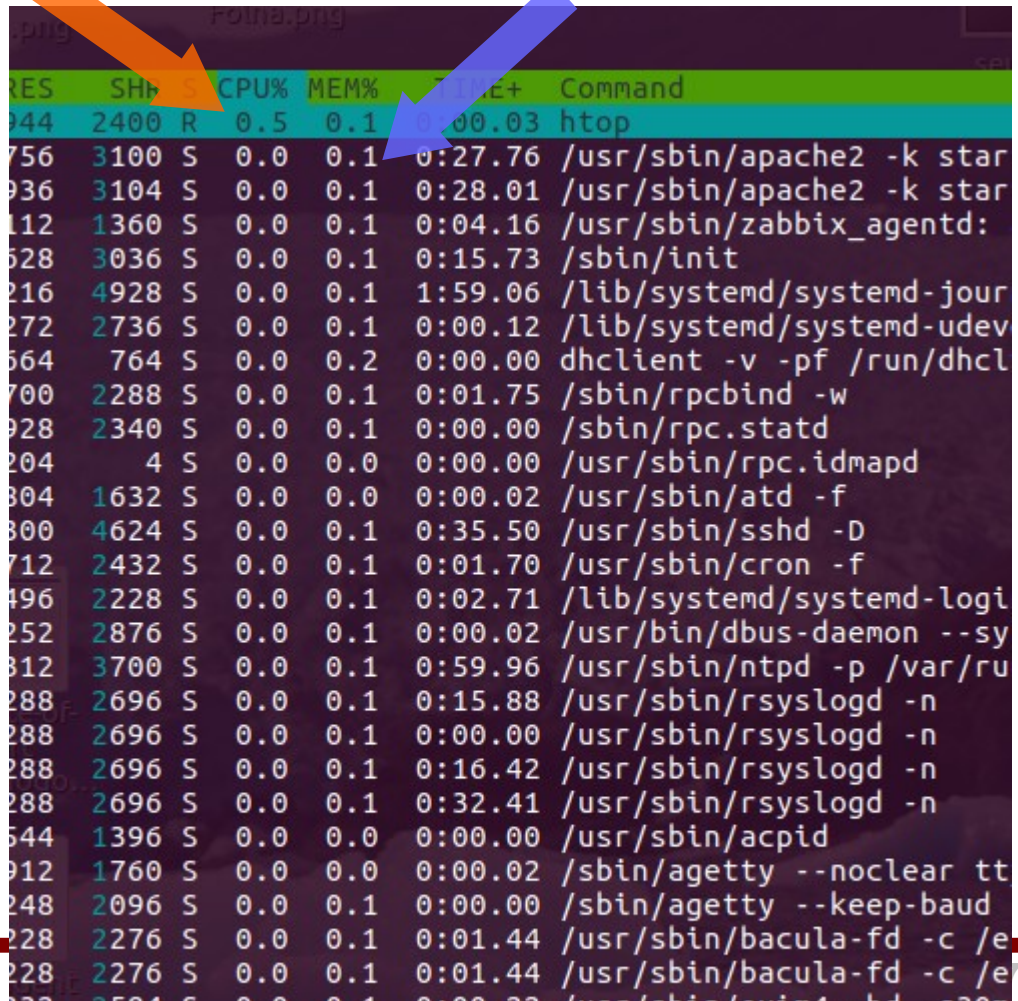
CPU cores 1 and 2



Percentage of CPU cores being used by each program

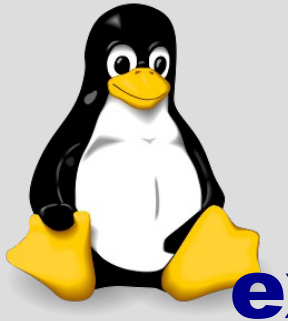
(can be > 100% for programs that use multiple cores)

Percentage of memory used by each program

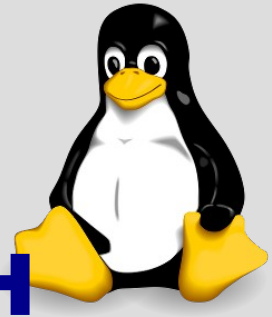


Who's here?

- Another, more indirect, way of telling how busy the system is consists of checking **how many other users** are logged in the system
- Commands **`who`** and **`w`** can tell you that, and more
- **`who`** tells you who is currently in the system, in which terminal, and from which original address (IP number or not)
- **`w`** also tell you what program each user is running at the time
- Overall, both programs are quite similar
- **Try them in the remote server!**



Basic file manipulation, environmental variables, executing programs out of $PATH



J. M. P. Alves

Laboratory of Genomics & Bioinformatics in Parasitology

Department of Parasitology, ICB, USP

Creating files and directories

- There is an easy way to create a new, empty **regular** file: the `touch` command (we will see other ways later)
- The main function of touch is to **change the timestamps** of files and directories but, if you try to change them for a file that does not exist, then by default it is **created (empty)**
- (Parenthesis: timestamps are the time information about files, i.e., creation, modification, and access time. You can use `stat` to see all of that)
- To create, in the current directory, a new file called `my_new_file` run:

```
touch my_new_file
```

- If you did everything correctly, when you list your directory contents you will see a file called `my_new_file`, with size of zero bytes. **Try it!**

Let's try!

- Log into the remote server, if you are not there already
- List (long format) the contents of your home area
- You should see a file called **`dummy_file`**
- **Take notice** of that file's modification time in your listing. What do you see?
- Now run touch on **`dummy_file`**:

```
touch dummy_file
```

- List your directory contents again
- **Did anything change? What?**

Creating files and directories

- How about directories? We cannot use **`touch`** to **create** directories (although we **CAN** use it to change their timestamps, of course!)

- The command to create new directories is **`mkdir`** (from **m**a**k**e **dir**ectory), which we have also seen before

- To create, in the current directory, a new directory called **`my_new_dir`** run:

```
mkdir my_new_dir
```

- If you did everything correctly, when you list your directory contents you will see a new directory called **`my_new_dir`**. **Try it!**

Creating files and directories

- How about creating **nested** directories? Say you want to create a directory inside another directory...

- You can use `mkdir` from anywhere
- After you have created **`my_new_dir`** in the previous try, run:

```
mkdir my_new_dir/subdir1
```

- If you did everything correctly, you should now see **`subdir1`** inside of **`my_new_dir`**. **Try it!**
- **Tip:** as with **`mkdir`** above, you do not need to enter (**`cd`** into) the directory to list its contents! Just do **`ls -l my_new_dir`** to see what's inside

Creating files and directories

- What if the **parent** directory does not exist? Say you want to create a directory called **`subdir2`** inside a directory called **`my_new_dir2`**, but **`my_new_dir2`** does not exist...

- **Try it**:

```
mkdir my_new_dir2/subdir2
```

Quiz time!

Go to the course page and choose **Quiz 9**

Creating files and directories

- What if the **parent** directory does not exist? Say you want to create a directory called **`subdir2`** inside a directory called **`my_new_dir2`**, but **`my_new_dir2`** does not exist...

- **Try it**:

```
mkdir my_new_dir2/subdir2
```

- The option **`-p`** creates any parent directory (or directories!) that does (or do) not exist. Try:

```
mkdir -p my_new_dir2/subdir2/testdir/
```

Now you do it!

Go to the course site and enter **Practical Exercise 11**

Follow the instructions to answer the questions in the exercise

Remember: in a PE, you should do things in practice before answering the question!

Recap

- Sharing is caring: when multiple users are in the system at the same time, it is important **not to abuse the resources** (working memory, disk space, processing power)
- Several useful commands (**`du`**, **`df`**, **`free`**, **`htop`**, **`uptime`**, etc.) allow one to **check** what resources are available in the system, and how heavily they are being used
- You can create new **regular files** with **`touch`** (a tool primarily intended for changing timestamps, but which creates an empty regular file is it does not exist)
- To create empty directories, use the **`mkdir`** command
- The **`-p`** option of **`mkdir`** allows us to, one command, create nested directories (e.g., **`one/two/nextone/finaldir`**)