

UE Programmation Unix

TPO Commandes Processus

Commandes de base de gestion de processus

1

Visualisation des caractéristiques des processus

ps, top

PID

Numéro du processus

La commande `ps` permet de visualiser des informations sur les processus.

```
[ev000000@saphyr ~]$ ps
```

PID	TTY	TIME	CMD
31903	pts/4	00:00:00	bash
31929	pts/4	00:00:00	ps

cmd	simple name of executable
cputime	TIME cumulative CPU time, "[DD-]hh:mm:ss" format. (alias time).

TTY identification de son terminal de contrôle

Utilisée sans option, la commande ne traite que les processus associés au même terminal que la commande `ps`.

TTY : le nom de la console dans laquelle se trouve l'utilisateur. Souvenez-vous que sous Linux il y a en général six consoles (`tty1` à `tty6`) et qu'en plus de ça, on peut en ouvrir une infinité grâce aux consoles graphiques (leur nom commence par `pts` , en général),

1.a) Consulter les pages man de la commande `ps` pour prendre connaissance des différentes options offertes.

```
[ij04115@saphyr:~]:jeu. sept. 03$ ps
```

PID	TTY	TIME	CMD
23248	pts/0	00:00:00	bash2
23272	pts/0	00:00:00	ps

```
[ij04115@saphyr:~]:mar. sept. 01$ man ps
```

```
PS(1)                                User Commands                                PS(1)

NAME
    ps - report a snapshot of the current processes.

SYNOPSIS
    ps [options]

DESCRIPTION
    ps displays information about a selection of the active processes.  If
    you want a repetitive update of the selection and the displayed
    information, use top(1) instead.
```

By default, `ps` selects all processes with the same effective user ID (`euid=EUID`) as the current user and associated with the same terminal as the invoker. It displays the process ID (`pid=PID`), the terminal associated with the process (`tname=TTY`), the cumulated CPU time in `[DD-]hh:mm:ss` format (`time=TIME`), and the executable name (`ucmd=CMD`). Output is unsorted by default.

OTHER INFORMATION

`--help` section

Print a help message. The section argument can be one of simple, list, output, threads, misc or all. The argument can be shortened to one of the underlined letters as in: `s|l|o|t|m|a`.

Usage:

`ps [options]`

Essayiez '`ps --aide <simple|liste|sortie|threads|divers|tous>`'
ou '`ps --aide <s|l|o|h|d|t>`'
pour plus d'aide.

```
[ij04115@saphyr:~]:mar. sept. 01$ ps --aide sortie
```

Usage:

`ps [options]`

Formats de sortie:

<code>-F</code>	vraiment complet
<code>-f</code>	format complet y compris les lignes de commande
<code>f, --forest</code>	arbre des processus en art ascii
<code>-H</code>	montre la hiérarchie des processus
<code>-j</code>	format de la tâche
<code>j</code>	format de contrôle de la tâche BSD
<code>-l</code>	format long
<code>l</code>	format long BSD
<code>-M, Z</code>	ajoute les données de sécurité (pour SELinux)
<code>-O <format></code>	pré-remplir avec les colonnes par défaut
<code>O <format></code>	comme <code>-O</code> avec la personnalité BSD
<code>-o, o, --format <format></code>	format défini par l'utilisateur
<code>s</code>	format de signal
<code>u</code>	format orienté utilisateur
<code>v</code>	format de mémoire virtuelle
<code>X</code>	format de registre
<code>-Y</code>	ne montre pas les fanions, montre rss/adr (utilisé avec <code>-l</code>)
<code>--context</code>	affiche le contexte de sécurité (pour SELinux)
<code>--headers</code>	répète les lignes d'en-tête, une par page
<code>--no-headers</code>	ne pas afficher du tout d'en-tête
<code>--cols, --columns, --width <num></code>	change la largeur de l'écran
<code>--rows, --lines <num></code>	change la hauteur de l'écran

User-defined format. format is a single argument in the form of a blank-separated or comma-separated list, which offers a way to specify individual output columns.

Headers may be renamed (`ps -o pid,ruser=RealUser -o comm=Command`) as desired.

1.b) Afficher les informations suivantes concernant le processus associé à votre commande `ps` :

- PID : numéro du processus,
- PPID: numéro du processus parent,
- TTY : identification de son terminal de contrôle,
- UID : identité du propriétaire réel du processus,
- PRI : priorité courante du processus,
- ADDR: adresse du processus en mémoire s'il est en mémoire et sur disque sinon,
- SZ : taille du processus exprimée en nombre de blocs.

Méthode simple mais pas jolie :

<code>pid</code>	PID	a number representing the process ID (alias <code>tgid</code>).
<code>uid</code>	user ID number	

ps -o pid -o ppid -o tty -o uid -o pri -o addr -o sz

```
[ij04115@saphyr:~]:jeu. sept. 03$ ps -o pid -o ppid -o tty -o uid -o pri -o addr -o sz
  PID  PPID TT      UID PRI ADDR  SZ
23248 23247 pts/0   11978 19  - 5246
23273 23248 pts/0   11978 19  - 1906
```

PID	PPID	TT	UID	PRI	ADDR	SZ
23248	23247	pts/0	11978	19	-	5246
23271	23248	pts/0	11978	19	-	1906

`pri` PRI priority of the process. Higher number means lower priority.

`tty` TT controlling tty (terminal) (alias `tname`, `tt`).

`sz` SZ size in physical pages of the core image of the process. This includes text, data, and stack space. Device mappings are currently excluded; this is subject to change. See `vsz` and `rss`.

S -- Process Status
The status of the task which can be one of:
D = uninterruptible sleep
R = running
S = sleeping
T = stopped by job control signal
t = stopped by debugger during trace
Z = zombie

TTY -- Controlling Tty
The name of the controlling terminal. This is usually the device (serial port, pty, etc.) from which the process was started, and which it uses for input or output. However, a task need not be associated with a terminal, in which case you'll see '?' displayed.

La commande `ps` présente un cliché instantané des processus en cours. Pour obtenir un affichage remis à jour régulièrement, utilisez la commande `top`.

```
0 R 60933 32023 31903 0 77 0 - 1035 - pts/4 00:00:00 ps
```

TTY

TIME+

The `top` program provides a dynamic real-time view of a running system. It can display system summary information as well as a list of processes or threads currently being managed by the Linux kernel.

COM. COMMAND

1.c) Tester la commande top.

%MEM -- Memory Usage (RES)

A task's currently used share of available physical memory.

%CPU -- CPU Usage

The task's share of the elapsed CPU time since the last screen update, expressed as a percentage of total CPU time.

%MEM -- Memory Usage (RES)

A task's currently used share of available physical memory.

TIME -- CPU Time

Total CPU time the task has used since it started.

TIME+ -- CPU Time, hundredths

The same as TIME, but reflecting more granularity through hundredths of a second.

COMMAND -- Command Name or Command Line

Display the command line used to start a task or the name of the associated program.

```
[ij04115@saphyr:~]:mer. sept. 02$ top
top - 11:12:21 up 6 days, 9:21, 1 user, load average: 0,00, 0,00, 0,00
Tâches: 116 total, 1 en cours, 115 en veille, 0 arrêté, 0 zombie
%Cpu(s): 0,0 ut, 0,0 sy, 0,0 ni,100,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
KiB Mem : 6214852 total, 4877644 libr, 71068 util, 1266140 tampon/cache
KiB Éch: 2093052 total, 2093052 libr, 0 util. 5532272 dispo Mem
```

PID	UTIL.	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TEMPS+	COM.
15648	root	20	0	0	0	0	S	0,3	0,0	0:02.32	kworker/1:2
1	root	20	0	6620	4828	3660	S	0,0	0,1	0:03.05	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.03	kthreadd
3	root	20	0	0	0	0	S	0,0	0,0	0:00.02	ksoftirqd/0
5	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	kworker/0:0H
7	root	20	0	0	0	0	S	0,0	0,0	1:19.54	rcu_sched
8	root	20	0	0	0	0	S	0,0	0,0	0:00.00	rcu_bh
9	root	rt	0	0	0	0	S	0,0	0,0	0:00.02	migration/0
10	root	rt	0	0	0	0	S	0,0	0,0	0:01.48	watchdog/0
11	root	rt	0	0	0	0	S	0,0	0,0	0:01.18	watchdog/1
12	root	rt	0	0	0	0	S	0,0	0,0	0:00.02	migration/1
13	root	20	0	0	0	0	S	0,0	0,0	0:00.06	ksoftirqd/1
15	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	kworker/1:0H
16	root	rt	0	0	0	0	S	0,0	0,0	0:01.13	watchdog/2
17	root	rt	0	0	0	0	S	0,0	0,0	0:00.02	migration/2
18	root	20	0	0	0	0	S	0,0	0,0	0:00.24	ksoftirqd/2
20	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	kworker/2:0H

PID -- Process Id

USER -- User Name

The effective user name of the task's owner.

PR -- Priority

The scheduling priority of the task. If you see 'rt' in this field, it means the task is running under real time scheduling priority.

NI -- Nice Value

The nice value of the task. A negative nice value means higher priority, whereas a positive nice value means lower priority. Zero in this field simply means priority will not be adjusted in determining a task's dispatchability.

KiB = kibibyte = 1024 bytes

VIRT -- Virtual Memory Size (KiB)

The total amount of virtual memory used by the task. It includes all code, data and shared libraries plus pages that have been swapped out and pages that have been mapped but not used.

RES -- Resident Memory Size (KiB)

The non-swapped physical memory a task is using.

KiB = kibibyte = 1024 bytes

SHR -- Shared Memory Size (KiB)

The amount of shared memory available to a task, not all of which is typically resident. It simply reflects memory that could be potentially shared with other processes.

SUMMARY Display

UPTIME and LOAD Averages

This portion consists of a single line containing:
program or window name, depending on display mode
current time and length of time since last boot
total number of users
system load avg over the last 1, 5 and 15 minutes

1

```
top - 11:12:21 up 6 days, 9:21, 1 user, load average: 0,00, 0,00, 0,00
Tâches: 116 total, 1 en cours, 115 en veille, 0 arrêté, 0 zombie
%Cpu(s): 0,0 ut, 0,0 sy, 0,0 ni, 100,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
KiB Mem : 6214852 total, 4877644 libr, 71068 util, 1266140 tamp/cache
KiB Éch: 2093052 total, 2093052 libr, 0 util. 5532272 dispo Mem
```

TASK and CPU States

This portion consists of a minimum of two lines. In an SMP environment, additional lines can reflect individual CPU state percentages.

Line 1 shows total tasks or threads, depending on the state of the Threads-mode toggle. That total is further classified as:

running; sleeping; stopped; zombie

Line 2 shows CPU state percentages based on the interval since the last refresh.

As a default, percentages for these individual categories are displayed. Where two labels are shown below, those for more recent kernel versions are shown first.

us, user : time running un-niced user processes
sy, system : time running kernel processes
ni, nice : time running niced user processes
id, idle : time spent in the kernel idle handler
wa, IO-wait : time waiting for I/O completion
hi : time spent servicing hardware interrupts
si : time spent servicing software interrupts
st : time stolen from this vm by the hypervisor

In the alternate cpu states display modes, beyond the first tasks/threads line, an abbreviated summary is shown consisting of these elements:

a b c d
%Cpu(s): 75.0/25.0 100[...

Where: a) is the combined us and ni percentage; b) is the sy percentage; c) is the total; and d) is one of two visual graphs of those representations. See topic 4b. SUMMARY AREA Commands and the 't' command for additional information on that special 4-way toggle.

2

MEMORY Usage

This portion consists of two lines which may express values in kibibytes (KiB) through exbibytes (EiB) depending on the scaling factor enforced with the 'E' interactive command.

As a default, Line 1 reflects physical memory, classified as:
total, free, used and buff/cache

Line 2 reflects mostly virtual memory, classified as:
total, free, used and avail (which is physical memory)

The avail number on line 2 is an estimation of physical memory available for starting new applications, without swapping. Unlike the free field, it attempts to account for readily reclaimable page cache and memory slabs. It is available on kernels 3.14, emulated on kernels 2.6.27+, otherwise the same as free.

In the alternate memory display modes, two abbreviated summary lines are shown consisting of these elements:

a b c
GiB Mem : 18.7/15.738 [...
GiB Swap: 0.0/7.999 [...

3

RES -- Resident Memory Size (KiB)
The non-swapped physical memory a task is using.

Where: a) is the percentage used; b) is the total available; and c) is one of two visual graphs of those representations.

In the case of physical memory, the percentage represents the total minus the estimated avail noted above. The 'Mem' graph itself is divided between used and any remaining memory not otherwise accounted for by avail. See topic 4b. SUMMARY AREA Commands and the 'm' command for additional information on that special 4-way toggle.

```
KiB = kibibyte = 1024 bytes
MiB = mebibyte = 1024 KiB = 1,048,576 bytes
GiB = gibibyte = 1024 MiB = 1,073,741,824 bytes
TiB = tebibyte = 1024 GiB = 1,099,511,627,776 bytes
PiB = pebibyte = 1024 TiB = 1,125,899,906,842,624 bytes
EiB = exbibyte = 1024 PiB = 1,152,921,504,606,846,976 bytes
```

1.d) Préciser quand il est préférable d'utiliser la commande `top` plutôt que la commande `ps`.

```
[i]04115@saphyr:/:mar. sept. 01$ man ps
PS(1)                                User Commands                                PS(1)

NAME
    ps - report a snapshot of the current processes.

SYNOPSIS
    ps [options]

DESCRIPTION
    ps displays information about a selection of the active processes.  If
    you want a repetitive update of the selection and the displayed
    information, use top(1) instead.
```

« **ps** donne un aperçu des processus actifs.

Si vous souhaitez une mise à jour répétitive de cette aperçu, utilisez **top**. »

Donc,

Si vous voulez connaître les statistiques de processus en temps réel d'un système, par exemple : la liste des processus en cours d'exécution, quel processus prend le plus de mémoire ou de CPU, quel processus est exécuté à un moment donné, etc. Vous utilisez **top**.

Source : <https://www.quora.com/What-is-the-difference-between-top-and-ps-command>

`top` enables you to see your processes ordered by the amount of processor power they use. `ps` enables you to see all your processes, or just the processes used by certain users, for example `root` or yourself.

`top` should be used to see which processes are most active, `ps` could be used to see which processes you (or any other user) are running currently.

2

Exécution différée at, batch

La commande at

La commande at permet de différer, à une date spécifiée, l'exécution d'une suite de commandes. Sauf demande de redirection explicite, la sortie standard et la sortie erreur sont redirigées sur la boîte à lettres de l'utilisateur.

```
[ev00000@saphyr ~]$ at 17:24
```

```
at> ls >fls
```

```
at> <EOT> ← frappe de ctrl d
```

```
job 13 at 2007-02-05 17:24
```

Pour lancer l'exécution dans un fichier nommé « fls »

Taper CTRL + D quand on a fini d'indiquer les command à exécuter plus tard

La table ASCII

EOT	Ctrl+D
-----	--------

Fin de communication

examine or delete jobs for later execution

DESCRIPTION

at and batch read commands from standard input or a specified file which are to be executed at a later time, using /bin/sh.

at executes commands at a specified time.

At accepts times of the form HH:MM to run a job at a specific time of day.
(If that time is already past, the next day is assumed.)

You may also specify midnight, noon, or teatime (4pm) and you can have a time-of-day suffixed with AM or PM for running in the morning or the evening. You can also say what day the job will be run, by giving a date in the form month-name day with an optional year, or giving a date of the form MMDD[CC]YY, MM/DD/[CC]YY, DD.MM.[CC]YY or [CC]YY-MM-DD. The specification of a date must follow the specification of the time of day. You can also give times like now + count time-units, where the time-units can be minutes, hours, days, or weeks and you can tell at to run the job today by suffixing the time with today and to run the job tomorrow by suffixing the time with tomorrow.

For example, to run a job at 4pm three days from now, you would do at 4pm + 3 days, to run a job at 10:00am on July 31, you would do at 10am Jul 31 and to run a job at 1am tomorrow, you would do at 1am tomorrow.

If you specify a job to absolutely run at a specific time and date in the past, the job will run as soon as possible. For example, if it is 8pm and you do a at 6pm today, it will run more likely at 8:05pm.

For both at and batch, commands are read from standard input or the file specified with the -f option and executed. The working directory, the environment (except for the variables BASH_VERSION, DISPLAY, EUID, GROUPS, SHELL_OPTS, TERM, UID, and _) and the umask are retained from the time of invocation.

```
[ij04115@saphyr:~]:jeu. sept. 03$ at 12:30
warning: commands will be executed using /bin/sh
at> cd /
at> ls
at> <EOT>
```

```
job 44 at Thu Sep 3 12:30:00 2020
```

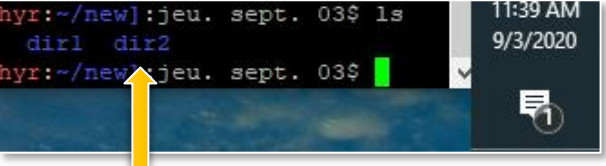
```
[ij04115@saphyr:~]:jeu. sept. 03$
```

There are many different Unix shells, but all derive many of their features from the Bourne shell, or /bin/sh. Every Unix system needs the Bourne shell to function correctly, as you will see throughout this book.

a) Tester la commande dans le cas où les commandes à exécuter sont lues sur l'entrée standard puis le cas où elles sont lues dans un fichier.

Les commandes à exécuter sont lues sur l'entrée standard

```
[ij04115@saphyr:~/new]:jeu. sept. 03$ ls
dir1
[ij04115@saphyr:~/new]:jeu. sept. 03$ at 11:35
warning: commands will be executed using /bin/sh
at> mkdir dir2
at> <EOT>
job 46 at Thu Sep  3 11:35:00 2020
[ij04115@saphyr:~/new]:jeu. sept. 03$ ls
dir1
[ij04115@saphyr:~/new]:jeu. sept. 03$
```



Les commandes à exécuter sont lues dans un fichier

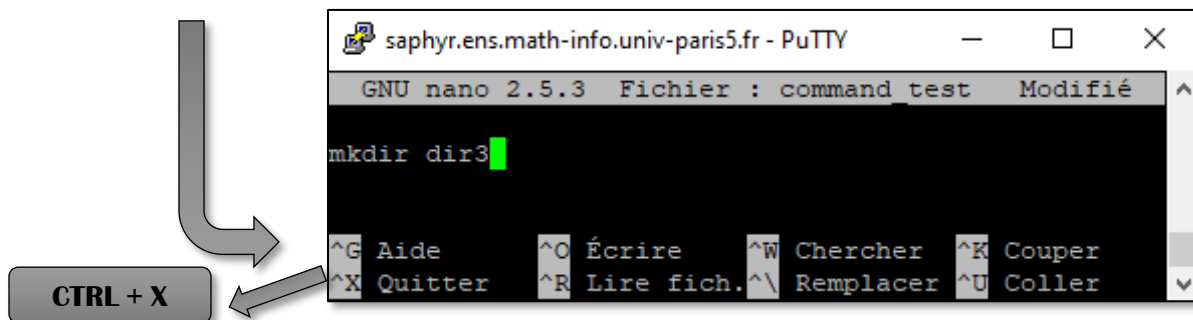
1.3 Créer un fichier

En reprenant la liste des commandes de manipulation de fichiers et de répertoires, on remarque qu'il n'existe pas de commande pour créer un fichier comme il en existe une pour créer un répertoire (mkdir).

Pour créer un fichier sous Unix, on est alors obligé de détourner le fonctionnement normal d'autres commandes :

	commande	remarque
1	touch fichier	méthode recommandée si fichier vide
2	nano fichier	puis quitter en sauvegardant /sinon
3	cat > fichier	
4	cp /dev/null fichier	
5	mv /dev/null fichier	
6	echo "" > fichier	

```
[ij04115@saphyr:~/new]:jeu. sept. 03$ nano command_test
```




```
[ij04115@saphyr:~/new]:jeu. sept. 03$ ls
command_test dirl dir2
[ij04115@saphyr:~/new]:jeu. sept. 03$ at -f command_test 11:50
warning: commands will be executed using /bin/sh
job 47 at Thu Sep  3 11:50:00 2020
[ij04115@saphyr:~/new]:jeu. sept. 03$ ls
command_test dirl dir2
[ij04115@saphyr:~/new]:jeu. sept. 03$
```

ENG
INTL

11:46 AM
9/3/2020



SYNOPSIS

```
at [-V] [-q queue] [-f file] [-mMlv] timespec...
at [-V] [-q queue] [-f file] [-mMkv] [-t time]
```

```
[ij04115@saphyr:~/new]:jeu. sept. 03$ ls
command_test dirl dir2 dir3
[ij04115@saphyr:~/new]:jeu. sept. 03$
```

11:50 AM
9/3/2020



b) Comment êtes-vous avertis de la terminaison de la commande ?

Pour lancer l'exécution dans un fichier rajouter >file.

Sinon la sortie (d'un **ls** par exemple) sera envoyé par mail.

```
[ij04115@saphyr:~/new]:jeu. sept. 03$ at 12:05
warning: commands will be executed using /bin/sh
at> ls > file
at> <EOT>
job 48 at Thu Sep  3 12:05:00 2020
[ij04115@saphyr:~/new]:jeu. sept. 03$ ls
command_test dirl dir2 dir3
[ij04115@saphyr:~/new]:jeu. sept. 03$
```

ENG
INTL

11:58 AM
9/3/2020



```
[ij04115@saphyr:~/new]:jeu. sept. 03$ ls
command_test dirl dir2 dir3 file
[ij04115@saphyr:~/new]:jeu. sept. 03$
```

12:07 PM
9/3/2020

```
[ij04115@saphyr:~/new]:jeu. sept. 03$ nano file
```

saphyr.ens.math-info.univ-paris5.fr - PuTTY

GNU nano 2.5.3

Fichier : file

```
command_test
dirl
dir2
dir3
file
```

Ça marche aussi

```
[ij04115@saphyr:~/new]:jeu. sept. 03$ at 12:17
warning: commands will be executed using /bin/sh
at> ls > bla_bla_bla
at> <EOT>
job 49 at Thu Sep  3 12:17:00 2020
[ij04115@saphyr:~/new]:jeu. sept. 03$ ls
command_test  dirl  dir2  dir3  file
[ij04115@saphyr:~/new]:jeu. sept. 03$
```

INTL
12:15 PM
9/3/2020

```
[ij04115@saphyr:~/new]:jeu. sept. 03$ ls
bla_bla_bla  command_test  dirl  dir2  dir3  file
[ij04115@saphyr:~/new]:jeu. sept. 03$
```

INTL
12:17 PM
9/3/2020

L'option `-l` (respectivement `-r`) permet de lister (respectivement de supprimer) les commandes enregistrées. On peut également utiliser à la place les deux commandes `atq` et `atrm`.

c) Tester les options précédentes.

`-l` Is an alias for `atq`.

`-r` Is an alias for `atrm`.

`atq` lists the user's pending jobs, unless the user is the superuser; in that case, everybody's jobs are listed. The format of the output lines (one for each job) is: Job number, date, hour, queue, and username.

`atrm` deletes jobs, identified by their job number.

```
[ij04115@saphyr:~]:jeu. sept. 03$ at 18:00
warning: commands will be executed using /bin/sh
at> ps
at> <EOT>
job 51 at Thu Sep  3 18:00:00 2020
[ij04115@saphyr:~]:jeu. sept. 03$ at 19:00
warning: commands will be executed using /bin/sh
at> ls
at> <EOT>
job 52 at Thu Sep  3 19:00:00 2020
```

Lister les commandes enregistrées

```
[ij04115@saphyr:~]:jeu. sept. 03$ at -l  
52      Thu Sep  3 19:00:00 2020 a ij04115  
51      Thu Sep  3 18:00:00 2020 a ij04115
```

```
[ij04115@saphyr:~]:jeu. sept. 03$ atq  
52      Thu Sep  3 19:00:00 2020 a ij04115  
51      Thu Sep  3 18:00:00 2020 a ij04115
```

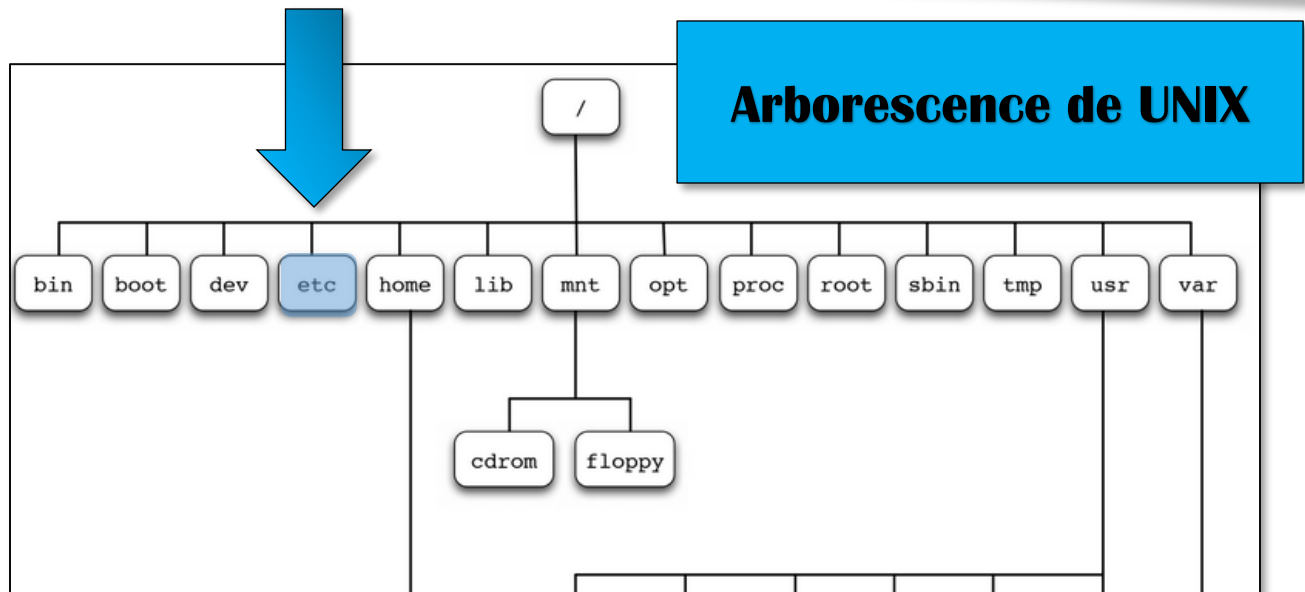
Supprimer les commandes enregistrées

```
atrm [-V] job [job...]
```

```
[ij04115@saphyr:~]:jeu. sept. 03$ at -r 52  
[ij04115@saphyr:~]:jeu. sept. 03$ at -l  
51      Thu Sep  3 18:00:00 2020 a ij04115  
[ij04115@saphyr:~]:jeu. sept. 03$
```

```
[ij04115@saphyr:~]:jeu. sept. 03$ atrm 51  
[ij04115@saphyr:~]:jeu. sept. 03$ atq  
[ij04115@saphyr:~]:jeu. sept. 03$
```

Le fichier `/etc/at.allow` (respectivement `/etc/at.deny`) contient la liste des utilisateurs autorisés (respectivement non autorisés) à utiliser la commande `at`.



Répertoire	Sous-répertoires	Contenu
/etc	/etc.rc.d	Contient les fichiers et répertoires de configuration du système

```

[ij04115@saphyr:~]:jeu. sept. 03$ cd /
[ij04115@saphyr:/]:jeu. sept. 03$ ls
automnt  home      mnt       srv        var
backup   initrd.img  opt       sys        vmlinuz
bin       initrd.img.old  proc      tmp        vmlinuz.old
boot      lib        root      users
dev       lost+found run        userssl
etc       media     sbin      usr
[ij04115@saphyr:/]:jeu. sept. 03$ cd etc
  
```


Action	Fichier	Répertoire
Afficher le contenu	cat OU more	ls

```
[ij04115@saphyr:/etc]:jeu. sept. 03$ cat at.deny
cat: at.deny: Permission non accordée
[ij04115@saphyr:/etc]:jeu. sept. 03$ cat at.allow
cat: at.allow: Aucun fichier ou dossier de ce type
```

La commande batch

La commande `batch` permet de différer, à une date déterminée par le système en fonction de la charge du système, l'exécution de commandes lues sur l'entrée standard. Les mécanismes de redirection sont les mêmes que ceux de la commande `at`.

```
[ev00000@saphyr ~]$ batch
at> ls >fls2
at> <EOT>← frappe de ctrl d
```

Pour lancer l'exécution dans un fichier nommer « fls2 »

Taper CTRL + D quand on a fini d'indiquer les command à exécuter plus tard

La table ASCII

EOT

Ctrl+D

Fin de communication

d) Tester la commande batch.

```
[ij04115@saphyr:~/new]:jeu. sept. 03$ batch
warning: commands will be executed using /bin/sh
at> mkdir dir4
at> <EOT>
job 58 at Thu Sep  3 18:36:00 2020
```

batch accepts no parameters !

e) Donner des exemples où il est préférable d'utiliser l'une des deux commandes (at et batch) plutôt que l'autre ?

Il est préférable de lancer batch pour ne pas obliger la machine à exécuter la tâche à l'heure précise en cas de surcharge.

3

Emission d'un signal

kill

La commande `kill` permet d'envoyer au processus (ou groupe de processus) d'identification donnée le signal désigné.

```
[ev000000@saphyr ~]$ ps
  PID TTY          TIME CMD
 1059 pts/3        00:00:00 bash
 1173 pts/3        00:00:02 a.out
 1174 pts/3        00:00:00 ps
[ev000000@saphyr ~]$ kill -9 1173
```

Les signaux sont identifiés par des nombres entiers (numéros absolus dans le système tels que fournis par la commande `ps`) ou par des noms symboliques tels qu'ils apparaissent dans le fichier `signal.h` (privés du préfixe `SIG`). Les noms des signaux reconnus sont affichés par la commande `kill -l`.

```
KILL(1)                                User Commands                                KILL(1)

NAME
    kill - send a signal to a process

SYNOPSIS
    kill [options] <pid> [...]

DESCRIPTION
    The default signal for kill is TERM. Use -l or -L to list available
    signals. Particularly useful signals include HUP, INT, KILL, STOP,
    CONT, and 0. Alternate signals may be specified in three ways: -9,
    -SIGKILL or -KILL. Negative PID values may be used to choose whole
    process groups; see the PGID column in ps command output. A PID of -1
    is special; it indicates all processes except the kill process itself
    and init.
```

```
[ij04115@saphyr:~]:ven. sept. 04$ kill -l
1) SIGHUP      2) SIGINT      3) SIGQUIT      4) SIGILL      5) SIGTRAP
6) SIGABRT     7) SIGBUS      8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV    12) SIGUSR2    13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT  17) SIGCHLD   18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN    22) SIGTTOU   23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM  27) SIGPROF   28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS     34) SIGRTMIN   35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4 39) SIGRTMIN+5 40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9 44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5 60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```

La commande kill

La commande `kill` envoie un signal aux processus actifs. Le numéro du signal et le PID du processus sont indiqués sur la ligne de commande. Si aucun numéro de signal n'est précisé, c'est `SIGTERM` (15) qui est envoyé. Ce signal tue les processus, sauf ceux qui l'interceptent. Pour ces processus, il faut envoyer explicitement le signal `SIGKILL` (9), qui ne peut être intercepté. L'administrateur `root` peut arrêter tous les processus avec la commande `kill`. Les autres utilisateurs ne peuvent gérer que les processus dont ils sont propriétaires.

Les différentes syntaxes de cette commande sont :

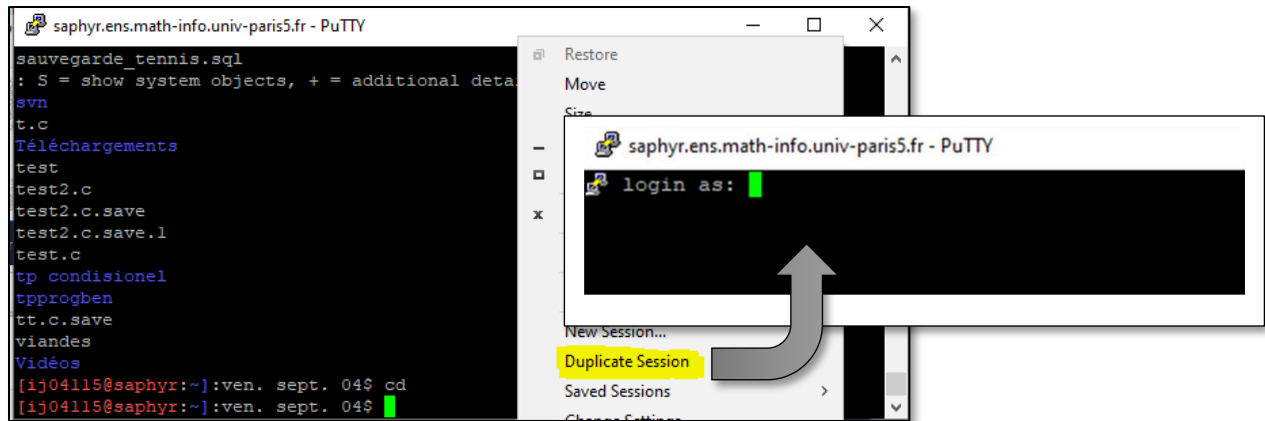
```
$ kill -num_signal pid1 pid2 etc.
$ kill -s num_signal pid1 pid2 etc.
$ kill -l
```

où `num_signal` est le numéro ou le nom de signal à envoyer, et `pid1`, `pid2`, etc., sont les numéros de processus vers lesquels envoyer le signal. Les PID des processus sont obtenus avec la commande `ps`. La commande `kill` avec l'option `-l` affiche la liste et le nom des signaux qui peuvent être administrés.

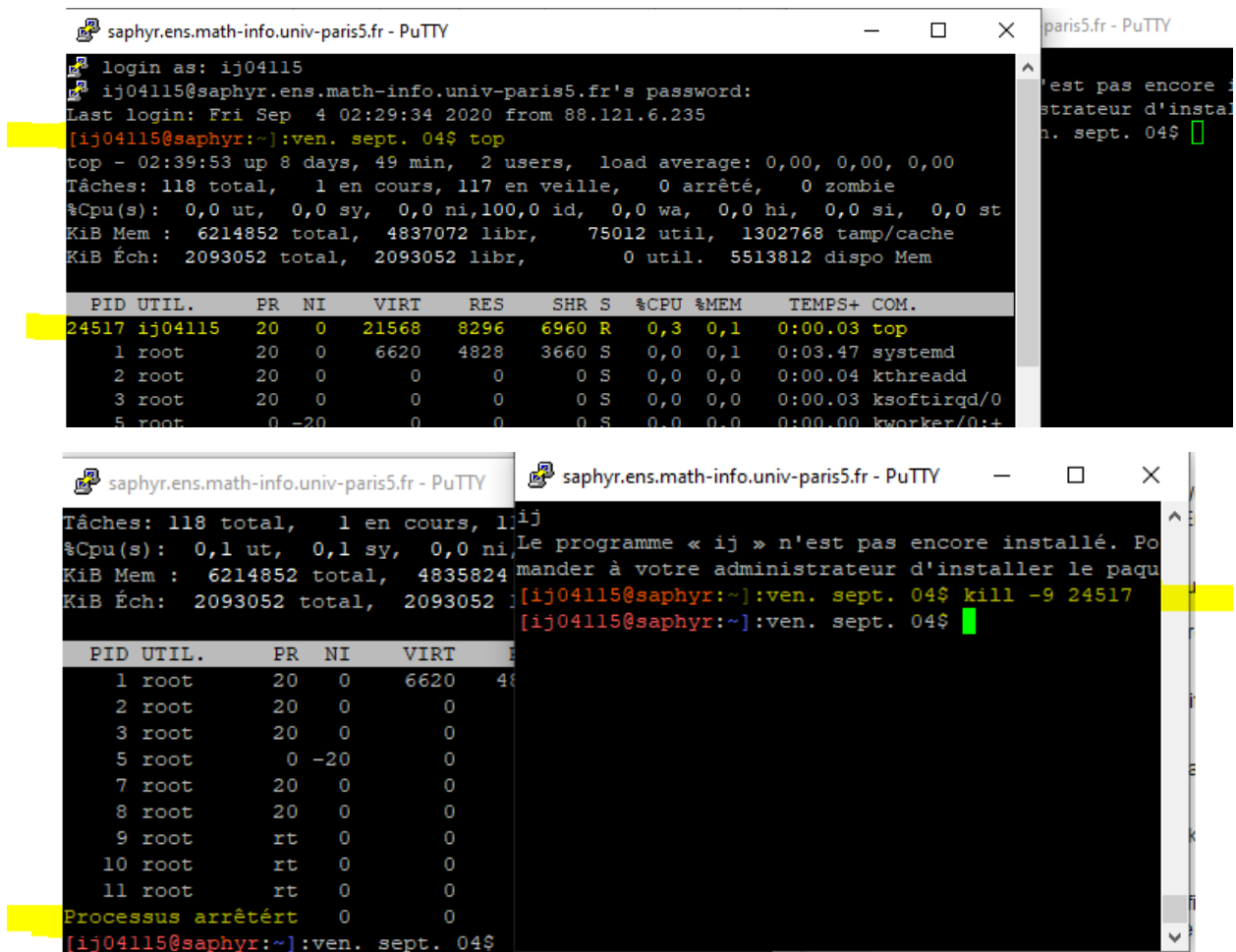
Comment arrêter un programme ?

```
$ kill -9 pid_du_programme
```

a) Lancer un processus puis lui émettre certains signaux. Que se passe-t-il ?



Test 1



Test 2

```
[ij04115@saphyr:~]:ven. sept. 04$ kill -l
 1) SIGHUP      2) SIGINT      3) SIGQUIT      4) SIGILL      5) SIGTRAP
 6) SIGABRT     7) SIGBUS      8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD    18) SIGCONT     19) SIGSTOP    20) SIGTSTP
21) SIGTTIN     22) SIGTTOU    23) SIGURG      24) SIGXCPU    25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF    28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```

19 - SIGSTOP (STOP) - Stop - Pause the process

saphyr.ens.math-info.univ-paris5.fr - PuTTY

PID	UTIL.	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TEMPS+	COM.
1	root	20	0	6620	4828	3660	S	0,0	0,1	0:03.47	system
2	root	20	0	0	0	0	S	0,0	0,0	0:00.04	kthrea
3	root	20	0	0	0	0	S	0,0	0,0	0:00.03	ksofti
5	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	kworke
7	root	20	0	0	0	0	S	0,0	0,0	1:45.99	rcu_sc
8	root	20	0	0	0	0	S	0,0	0,0	0:00.00	rcu_bh
9	root	rt	0	0	0	0	S	0,0	0,0	0:00.03	migrat
10	root	rt	0	0	0	0	S	0,0	0,0	0:01.86	watchd
11	root	rt	0	0	0	0	S	0,0	0,0	0:01.44	watchd
Processus arrêté											

[ij04115@saphyr:~]:ven. sept. 04\$ top

top - 02:47:37 up 8 days, 56 min, 2 users, load average: 0,00, 0,00, 0,0
Tâches: 118 total, 1 en cours, 117 en veille, 0 arrêté, 0 zombie
%Cpu(s): 0,0 ut, 0,1 sy, 0,0 ni, 99,9 id, 0,0 wa, 0,0 hi, 0,0 si, 0
KiB Mem : 6214852 total, 4833940 libr, 78100 util, 1302812 tamp/cach
KiB Éch: 2093052 total, 2093052 libr, 0 util. 5510728 dispo Mem

PID	UTIL.	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TEMPS+	COM.
24519	ij04115	20	0	21568	8228	6892	R	0,3	0,1	0:00.19	top
1	root	20	0	6620	4828	3660	S	0,0	0,1	0:03.47	syst+
2	root	20	0	0	0	0	S	0,0	0,0	0:00.04	kthr+

Is there any spec has suggested t North Carolina?

Official warning t

saphyr.ens.math-info.univ-paris5.fr - ...

saphyr.ens.math-info.univ-paris5.fr - PuTTY

```
[ij04115@saphyr:~]:ven. sept. 04$ kill -9 24517
[ij04115@saphyr:~]:ven. sept. 04$ kill -19 24519
[ij04115@saphyr:~]:ven. sept. 04$
```

PID	UTIL.	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TEMPS+	COM.
37	root	0	-20	0	0	0					
38	root	0	-20	0	0	0					
39	root	0	-20	0	0	0					
43	root	20	0	0	0	0					
44	root	0	-20	0	0	0					
45	root	20	0	0	0	0					
46	root	20	0	0	0	0					
62	root	0	-20	0	0	0					
66	root	0	-20	0	0	0					
67	root	0	-20	0	0	0					
68	root	0	-20	0	0	0					
69	root	0	-20	0	0	0					
70	root	0	-20	0	0	0					

[1]+ Arrêté top

[ij04115@saphyr:~]:ven. sept. 04\$

Quand on utilise la commande « ps » sur le terminal ou top est en pause, « top » est afficher sur la liste des processus courant.

4

Ignorer certains signaux

nohup

```
[ij04115@saphyr:~]:ven. sept. 04$ kill -l
```

```
1) SIGHUP      2) SIGINT      3) SIGQUIT      4) SIGILL      5) SIGTRAP
6) SIGABRT     7) SIGBUS      8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV    12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT  17) SIGCHLD    18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN    22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM  27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS     34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4 39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9 44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6 59) SIGRTMAX-5 60) SIGRTMAX-4 61) SIGRTMAX-3 62) SIGRTMAX-2
63) SIGRTMAX-1 64) SIGRTMAX
```

```
[ij04115@saphyr:~]:sam. sept. 05$ man 7 signal
```

Term Default action is to terminate the process.

Signal	Value	Action	Comment
SIGHUP	1	Term	Hangup detected on controlling terminal or death of controlling process
SIGINT	2	Term	Interrupt from keyboard
SIGQUIT	3	Core	Quit from keyboard

Core Default action is to terminate the process and dump core (see core(5)).

NAME

core - core dump file

DESCRIPTION

The default action of certain signals is to cause a process to terminate and produce a core dump file, a disk file containing an image of the process's memory at the time of termination. This image can be used in a debugger (e.g., `gdb(1)`) to inspect the state of the program at the time that it terminated. A list of the signals which cause a process to dump core can be found in `signal(7)`.

```
saphyr.ens.math-info.univ-paris5.fr - PuTTY
login as: ij04115
ij04115@saphyr.ens.math-info.univ-paris5.fr's password:
Last login: Sun Sep  6 16:28:00 2020 from 88.121.6.235
[ij04115@saphyr:~]:dim. sept. 06$ top
top - 16:31:22 up 10 days, 14:40,  2 users,  load average: 0,00, 0,00, 0,00
Tâches: 118 total,  1 en cours, 117 en veille,  0 arrêté,  0 zombie
%Cpu(s):  0,0 ut,  0,0 sy,  0,0 ni,100,0 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
KiB Mem : 6214852 total, 4727412 libr,  76276 util, 1411164 tamp/cache
KiB Éch: 2093052 total, 2093052 libr,      0 util. 5505340 dispo Mem

  PID UTIL.  PR  NI  VIRT  RES  SHR S  %CPU %MEM  TEMPS+ COM.
  3910 ij04115  20   0  21580  8508  7164 R   0,3  0,1  0:00.03 top
    1 root      20   0   6620  4828  3660 S   0,0  0,1  0:04.17 systemd
```

```
saphyr.ens.math-info.univ-paris5.fr - PuTTY
11 root      rt      0      0      0      0 S   0,0  0,0  0:00.00 kworker/2:1+
12 root      rt      0      0      0      0 S   0,0  0,0  0:00.00 kworker/2:1+
13 root      20      0      0      0      0 S   0,0  0,0  0:00.00 kworker/2:1+
15 root      0 -20     0      0      0      0 S   0,0  0,0  0:00.00 kworker/2:1+
16 root      rt      0      0      0      0 S   0,0  0,0  0:00.00 kworker/2:1+
17 root      rt      0      0      0      0 S   0,0  0,0  0:00.00 kworker/2:1+
18 root      20      0      0      0      0 S   0,0  0,0  0:00.00 kworker/2:1+
20 root      0 -20     0      0      0      0 S   0,0  0,0  0:00.00 kworker/2:1+
21 root      rt      0      0      0      0 S   0,0  0,0  0:00.00 kworker/2:1+

[ij04115@saphyr:~]:dim. sept. 06$ kill -SIGHUP 3910
[ij04115@saphyr:~]:dim. sept. 06$
```

```
20 root      0 -20     0      0      0 S   0,0  0,0  0:00.00 kworker/2:1+
21 root      rt      0      0      0      0 S   0,0  0,0  0:01.67 watchdog/3

[ij04115@saphyr:~]:dim. sept. 06$ top
top - 16:36:37 up 10 days, 14:45,  2 users,  load average: 0,07, 0,03, 0,01
Tâches: 118 total,  1 en cours, 117 en veille,  0 arrêté,  0 zombie
%Cpu(s):  0,0 ut,  0,0 sy,  0,0 ni,100,0 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
KiB Mem : 6214852 total, 4728660 libr,  75004 util, 1411188 tamp/cache
KiB Éch: 2093052 total, 2093052 libr,      0 util. 5506600 dispo Mem

  PID UTIL.  PR  NI  VIRT  RES  SHR S  %CPU %MEM  TEMPS+ COM.
    7 root      20   0      0      0      0 S   0,3  0,0  2:25.52 rcu_sched
  839 root      20   0 49072 15532  6788 S   0,3  0,2  2:29.25 fail2ban-server
 3912 ij04115  20   0  21584  8580  7236 R   0,3  0,1  0:00.03 top
    1 root      20   0   6620  4828  3660 S   0,0  0,1  0:04.17 systemd
```

```
saphyr.ens.math-info.univ-paris5.fr - PuTTY
12 root      login as: ij04115
13 root      ij04115@saphyr.ens.math-info.univ-paris5.fr's pass
15 root      Last login: Sat Sep  5 21:43:08 2020 from 88.121.6.23
16 root      [ij04115@saphyr:~]:dim. sept. 06$ kill -SIGHUP 3910
17 root      [ij04115@saphyr:~]:dim. sept. 06$ kill -SIGINT 3912
18 root      [ij04115@saphyr:~]:dim. sept. 06$
20 root

21 root      rt      0      0      0      0 S   0,0  0,0
22 root      rt      0      0      0      0 S   0,0  0,0
23 root      20      0      0      0      0 S   0,0  0,0

[ij04115@saphyr:~]:dim. sept. 06$
```

```

23 root      20   0   0   0   0 S   0,0  0,0
[ij04115@saphyr:~]:dim. sept. 06$ top
top - 16:44:38 up 10 days, 14:53,  2 users,  load average: 0,
Tâches: 117 total,  1 en cours, 116 en veille,  0 arrêté
%Cpu(s):  0,0 ut,   0,0 sy,   0,0 ni,100,0 id,   0,0 wa,   0,0 hi
KiB Mem : 6214852 total, 4728800 libr,  74844 util, 1411
KiB Éch: 2093052 total, 2093052 libr,    0 util. 5506

  PID  UTIL.   PR  NI   VIRT   RES   SHR  S   %CPU  %MEM
3914 ij04115  20   0  21552   8488   7164 R    0,3   0,1
  1 root      20   0   6620   4828   3660 S    0,0   0,1

```

```

saphyr.ens.math-info.univ-paris5.fr - PuTTY
1 ro login as: ij04115
2 ro ij04115@saphyr.ens.math-info.univ-paris5.fr's pass
3 ro Last login: Sat Sep  5 21:43:08 2020 from 88.121.6.23
5 ro [ij04115@saphyr:~]:dim. sept. 06$ kill -SIGHUP 3910
7 ro [ij04115@saphyr:~]:dim. sept. 06$ kill -SIGINT 3912
9 ro [ij04115@saphyr:~]:dim. sept. 06$ kill -SIGQUIT 3914
10 ro [ij04115@saphyr:~]:dim. sept. 06$
11 root      20   0   0   0   0 S   0,0  0,0
12 root      20   0   0   0   0 S   0,0  0,0
[ij04115@saphyr:~]:dim. sept. 06$

```

La commande `nohup` permet de lancer une commande de telle sorte que le processus l'exécutant ignore les signaux SIGINT, SIGQUIT et SIGHUP.

```
[ev00000@saphyr ~]$ nohup ./a.out &
```

The "&" symbol at the end of the command instructs bash to run `nohup mycommand` in the background.

```
[1] 1197
```

```
[ev00000@saphyr ~]$ nohup: ajout Ã la sortie de 'nohup.out'
```

Lancer un processus puis lui émettre chacun des signaux précédents.

a) Que se passe-t-il ? Les processus réagissent-ils de la même façon que précédemment ?

NAME

`nohup` - run a command immune to hangups, with output to a non-tty

SYNOPSIS

```
nohup COMMAND [ARG]...
nohup OPTION
```

DESCRIPTION

Run `COMMAND`, ignoring hangup signals.

If standard input is a terminal, redirect it from an unreadable file. If standard output is a terminal, append output to 'nohup.out' if possible, '\$HOME/nohup.out' otherwise. If standard error is a terminal, redirect it to standard output. To save output to `FILE`, use 'nohup `COMMAND` > `FILE`'.


```
[ij04115@saphyr:~/dir_test2]:jeu. sept. 10$ nohup ps
nohup: les entrées sont ignorées et la sortie est ajoutée à 'nohup.out'
[ij04115@saphyr:~/dir_test2]:jeu. sept. 10$ ls
nohup.out
[ij04115@saphyr:~/dir_test2]:jeu. sept. 10$ cat nohup.out
  PID TTY          TIME CMD
27641 pts/1    00:00:00 bash2
27666 pts/1    00:00:00 ps
[ij04115@saphyr:~/dir_test2]:jeu. sept. 10$
```

```
[ij04115@saphyr:~/test_nohup]:jeu. sept. 10$ cat test.c
#include <stdio.h>
int main (){
    int a = 4;
    int *p = &a;

    printf("%p",p);

    return 0;
}

[ij04115@saphyr:~/test_nohup]:jeu. sept. 10$ gcc test.c
[ij04115@saphyr:~/test_nohup]:jeu. sept. 10$ ls
a.out  test.c
```

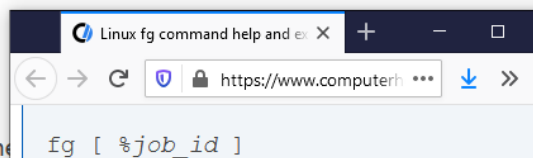
```
[ij04115@saphyr:~/test_nohup]:jeu. sept. 10$ nohup ./a.out &
[1] 28242
[ij04115@saphyr:~/test_nohup]:jeu. sept. 10$ nohup: les entrées sont ignorées et
la sortie est ajoutée à 'nohup.out'
```

The "&" symbol at the end of the command instructs bash to run **nohup mycommand** in the background. It can be brought back to the foreground with the **fg** bash [builtin](#) command.

When using **&**, you'll see the bash job ID in brackets, and the process ID (PID) listed after. For example:

```
[1] 25132
```

You can use the `fg [%job_id]`



5 Modification de priorité par défaut d'une commande nice

La commande `nice` lance l'exécution d'une commande en ajoutant la valeur d'un argument numérique au paramètre d'ordonnancement de son processus. La valeur de l'argument numérique doit être comprise entre 1 et 19 (valeur par défaut, 10). La commande, ainsi lancée, verra sa priorité abaissée.

```
[ev000000@saphyr ~]$ nice ./a.out &
```

```
[1] 1327
```

```
[ev000000@saphyr ~]$ ps -l
```

Usage:

`ps [options]`

Formats de sortie:

`-l`

`format long`

F	S	UID	PID	PPID	C	PRI	NI	ADDR	SZ	WCHAN	TTY	TIME	CMD
0	S	60933	1301	1300	0	75	0	-	1134	wait	pts/1	00:00:00	bash
0	R	60933	1327	1301	99	93	10	-	361	-	pts/1	00:00:03	a.out
0	R	60933	1328	1301	0	77	0	-	1034	-	pts/1	00:00:00	ps

S -- Process Status

The status of the task which can be one of:

D = uninterruptible sleep

R = running

S = sleeping

T = stopped by job control signal

t = stopped by debugger during trace

Z = zombie

uid user ID number

pid PID a number representing the process ID (alias tgid).

ppid PPID parent process ID.

a) Lancer une commande avec une priorité moindre que celle par défaut.

```
nice -PRIORITY COMMAND
```

```
[ij04115@saphyr:~]:jeu. sept. 10$ nice top &
[1] 28674
[ij04115@saphyr:~]:jeu. sept. 10$ ps -l
F S  UID  PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  11978 28667 28666  0  80   0 -  5254 wait  pts/1    00:00:00 bash2
0 T  11978 28674 28667  0  90  10 -  1863 signal pts/1    00:00:00 top
0 R  11978 28675 28667  0  80   0 -  1906 -      pts/1    00:00:00 ps

[1]+  Arrêté               nice top
```

```
[ij04115@saphyr:~]:jeu. sept. 10$ nice -7 top &
[2] 28676
[ij04115@saphyr:~]:jeu. sept. 10$ ps -l
F S  UID  PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  11978 28667 28666  0  80   0 -  5254 wait  pts/1    00:00:00 bash2
0 T  11978 28674 28667  0  90  10 -  1863 signal pts/1    00:00:00 top
0 T  11978 28676 28667  0  87   7 -  1863 signal pts/1    00:00:00 top
0 R  11978 28677 28667  0  80   0 -  1906 -      pts/1    00:00:00 ps

[2]+  Arrêté               nice -7 top
```

c) Lancer une commande avec une priorité supérieure à celle par défaut.

Quel est le résultat de votre action ? Pourquoi ?

```
[ij04115@saphyr:~]:jeu. sept. 10$ nice -15 top &
[3] 28678
[ij04115@saphyr:~]:jeu. sept. 10$ ps -l
F S  UID  PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  11978 28667 28666  0  80   0 -  5254 wait  pts/1    00:00:00 bash2
0 T  11978 28674 28667  0  90  10 -  1863 signal pts/1    00:00:00 top
0 T  11978 28676 28667  0  87   7 -  1863 signal pts/1    00:00:00 top
0 T  11978 28678 28667  0  95  15 -  1863 signal pts/1    00:00:00 top
0 R  11978 28679 28667  0  80   0 -  1906 -      pts/1    00:00:00 ps

[3]+  Arrêté               nice -15 top
```

b) Un argument négatif permet de lancer une commande avec une plus grande priorité.

```
[ij04115@saphyr:~]:jeu. sept. 10$ nice -7 top &
[1] 28460
[ij04115@saphyr:~]:jeu. sept. 10$ ps -l
F S  UID  PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  11978 28403 28402  0  80   0 -  5254 wait  pts/1    00:00:00 bash2
0 T  11978 28460 28403  0  87   7 -  1863 signal pts/1    00:00:00 top
0 R  11978 28461 28403  0  80   0 -  1906 -    pts/1    00:00:00 ps

[1]+  Arrêté                  nice -7 top
[ij04115@saphyr:~]:jeu. sept. 10$
```

niceness values range from -20 to 19, with the former being most favorable, while latter being least. In case you want to associate a negative nice value to the process, then you'll have to use double hyphen.

Please note that you need to have root privileges to associate a negative nice value to a process. And precisely for this reason, your ps command to confirm the new niceness should contain 'root' instead of the other user name.

```
$ ps -lu root | grep test-new
```

```
[ij04115@saphyr:~]:jeu. sept. 10$ nice --7 top &
[2] 28481
[ij04115@saphyr:~]:jeu. sept. 10$ nice: impossible de définir le niveau de priorité: Permission non accordée
```


6

Affichage du temps d'exécution d'un processus

time

La commande `time` permet de lancer une commande quelconque avec affichage, à la fin du processus l'exécutant, d'informations relatives à son temps d'exécution.

```
[ev00000@saphyr ~]$ time ps > fps
```

```
real    0m0.017s
user    0m0.004s
sys     0m0.007s
```

save a record of the time command information in a file called 'fps'

Use the cat command to display output on screen

```
$ cat fps
```

Purpose Run command/programs or script and summarize system resource usage

The **time** command show the following information on screen

(on the standard error output, by default) about resources used by command :

1. **real time** - correspond au temps réel de la tâche.
2. **user time** - correspond au temps utilisateur, c'est à dire le temps CPU utilisé par le programme utilisateur
3. **sys time** - définit le temps système, cela correspond au temps utilisé par le système pour gérer l'exécution de la tâche.

Le temps CPU de la tâche = temps user + temps sys

A **central processing unit (CPU)**, also

called a central processor, main processor or just processor, is the electronic circuitry within a computer that executes instructions that make up a computer program.

a) Tester la commande `time` avec certaines commandes.

```
[ij04115@saphyr:~]:mer. sept. 09$ time ps
  PID TTY          TIME CMD
15054 pts/3    00:00:00 bash2
15154 pts/3    00:00:00 ps

real    0m0.008s
user    0m0.000s
sys     0m0.004s
```

```
[ij04115@saphyr:~/test_dir]:mer. sept. 09$ time ps > my_file_name

real    0m0.013s
user    0m0.000s
sys     0m0.008s
[ij04115@saphyr:~/test_dir]:mer. sept. 09$ ls
my_file_name
[ij04115@saphyr:~/test_dir]:mer. sept. 09$ cat my_file_name
  PID TTY          TIME CMD
15054 pts/3    00:00:00 bash2
15168 pts/3    00:00:00 ps
[ij04115@saphyr:~/test_dir]:mer. sept. 09$
```

7

Mécanisme *Job control*

Caractères de contrôle : intr , quit et susp

Special Characters

Certain characters have special functions on input or output or both.

INTR

Special character on input, which is recognized if the ISIG flag is set. Generates a SIGINT signal which is sent to all processes in the foreground process group for which the terminal is the controlling terminal. If ISIG is set, the INTR character shall be discarded when processed.

QUIT

Special character on input, which is recognized if the ISIG flag is set. Generates a SIGQUIT signal which is sent to all processes in the foreground process group for which the terminal is the controlling terminal. If ISIG is set, the QUIT character shall be discarded when processed.

SUSP

If the ISIG flag is set, receipt of the SUSP character shall cause a SIGTSTP signal to be sent to all processes in the foreground process group for which the terminal is the controlling terminal, and the SUSP character shall be discarded when processed.

Ce mécanisme permet une gestion des processus créés à partir d'un `shell` supportant cette facilité. Il permet aux utilisateurs de ne voir que les processus qu'ils ont lancés sous le `shell` interactif dans lequel ils travaillent et qui constitue le cadre d'une **session de travail**. Chaque commande analysée par le `shell` correspond localement à un `job` ou tâche : d'un point de vue interne, il s'agit d'un groupe de processus.

Les processus lancés depuis le `shell` interactif peuvent se trouver dans l'un des états suivants :

En premier plan	En arrière plan	Suspendu
Dans une session, il existe au plus une tâche dont les processus peuvent lire et écrire sur le terminal de lancement et seront avisés lors de la frappe de caractères de contrôle sur le clavier de ce terminal	Les processus appartenant à une telle tâche ne peuvent pas lire au terminal et ne sont pas concernés par la frappe des caractères de contrôle	Les processus sont en sommeil et attendent que l'utilisateur demande explicitement leur passage à l'un des deux modes précédents
[ev00000@saphyr ~]\$ commande	[ev00000@saphyr ~]\$ commande&	

La commande jobs

La commande `jobs` fournit la liste des tâches créées par le shell. La tâche repérée par le caractère `+` est la tâche courante.

```
[ev00000@saphyr ~]$ jobs
```

```
[1]-  Running                  time ./a.out &
[2]+  Stopped                  vim scriptdate
```

La commande fg	La commande bg	La commande stop
La commande <code>fg</code> permet l'exécution en premier plan de la tâche courante.	La commande <code>bg</code> permet l'exécution en arrière plan de la tâche courante.	La commande <code>stop</code> permet l'interruption de l'exécution de la tâche courante. L'interruption de l'exécution d'un processus en premier plan peut être obtenue par la frappe de <code>ctrl z</code> .
<pre>[ev00000@saphyr ~]\$ nohup ./a.out & [1] 1426 [ev00000@saphyr ~]\$ jobs [1]+ Running time ./a.out & [ev00000@saphyr ~]\$ fg %1 time ./a.out ← E/S standards</pre>	<pre>[ev00000@saphyr ~]\$ jobs [2]- Stopped (tty output) vi script [3]+ Stopped netscape [ev00000@saphyr ~]\$ bg [3]+ netscape & [ev00000@saphyr ~]\$ jobs [2]+ Stopped (tty output) vi script [3]- Running netscape &</pre>	<pre>[ev00000@saphyr ~]\$ netscape ← frappe de ctrl z [3]+ Stopped netscape</pre>

Exemple utilisation de la commande bg :

Créer une tâche longue à effectuer (par exemple, archiver un dossier) :

```
tar -vcf test.tar TarDirectory/
```

CTRL+Z pour mettre la tâche en pause.

```
[1]+  Stopped    tar -vcf test.tar TarDirectory/
```

Passer la commande en arrière plan (pour qu'elle continue à s'exécuter)

```
bg %1
```