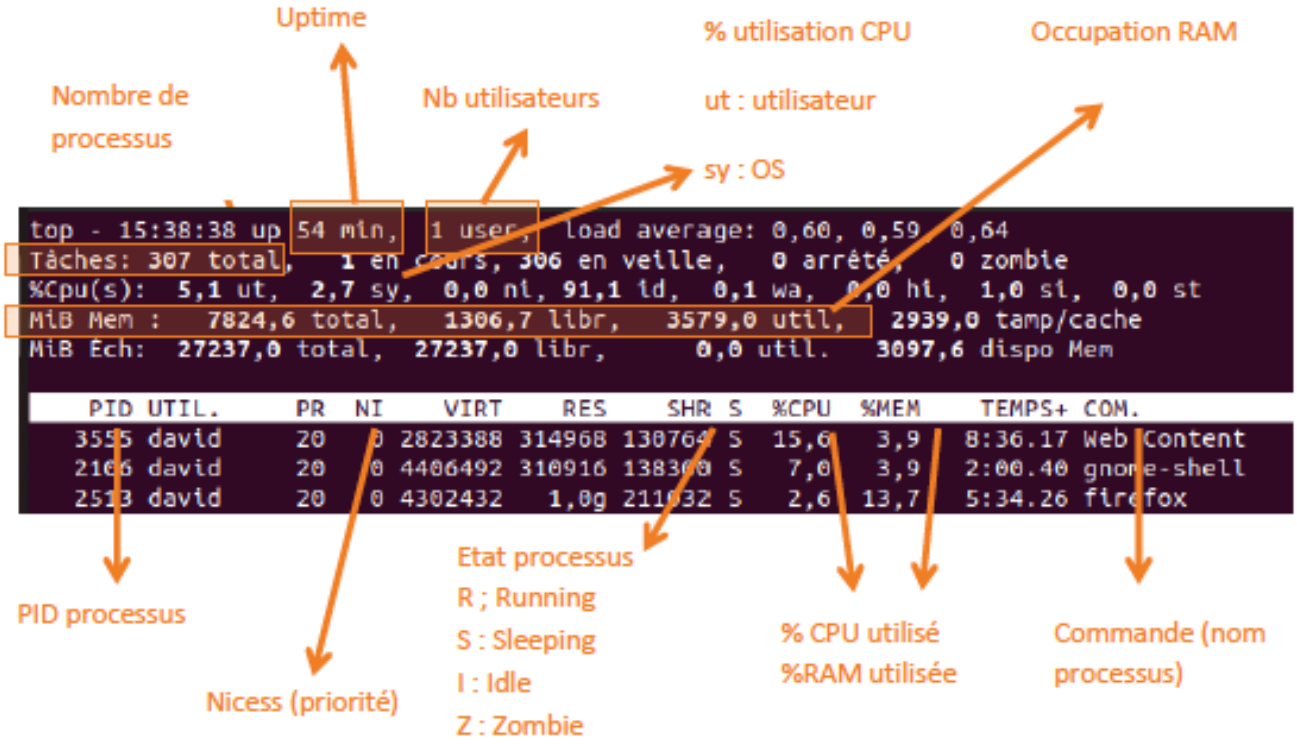
TP PROCESSUS

Sous Linux (par WSL ou directement), tester la commande `top`. Si vous avez utilisé WSL, tester également la commande `ps` dans le shell Windows, et comparez les deux résultats en nombre de processus. Ces commandes permet de lister tous les processus actifs : `ps` de manière statique et `top` de manière dynamique (équivalent du gestionnaire des tâches sous Windows)
On lit le résultat de `top` comme ceci :



On le quitte avec `q` (ou `ctrl-c` pour la méthode forte)
`V` permet d'avoir une vue arborescente des processus
`k` permet de tuer un processus

- Tuez `top` en notant son PID, relancez-le et comparer le nouveau PID avec l'ancien
- Tuez `init`

Ouvrez une deuxième console/une deuxième instance de Linux par WSL, placez les deux fenêtres côte à côte pour pouvoir en vois le déroulement simultané.

- Dans la deuxième fenêtre, cherchez tous les fichiers python avec la commande `find / -name "*.py"`
- Dans la première fenêtre, voyez l'apparition du processus, son pourcentage d'utilisation du CPU et celui d'occupation mémoire, etc.
- Et quand vous en avez assez, tuez-le.
- Testez dans la deuxième console `sleep 5`, `sleep 10000`. Observez (toujours avec `top`) le comportement dans la 1ère console. Que se passe-t-il si, dans la console du `sleep`, vous faites `ctrl-z` ? Dans quel état se trouve le processus ? Testez `fg` (foreground, premier plan), `ctrl-z` puis `bg` (background). Notez les différences, à la fois dans `top` et pour ce que vous pouvez faire dans la 2ème console.
- Tuez éventuellement le processus `sleep`, lancez-en un autre avec `sleep 10000 &`. Que constatez-vous ?

Dans la nouvelle console, commencez à éditer un nouveau texte, avec `vim` ou `nano`
Dans la première, lancez la commande `ps -ef` (l'option `e` donne aussi les daemons, `f` affiche toutes options de base).
La documentation Linux donne la signification des différents champs :
- `UID` : identifiant utilisateur effectif ;

- `PID` : identifiant de processus ;
- `PPID` : `PID` du processus parent, celui qui a engendré le processus ;
- `C` : partie entière du pourcentage d'utilisation du processeur par rapport au temps de vie des processus ;
- `STIME` : l'heure de lancement du processus ;
- `TTY` : terminal de contrôle
- `TIME` : temps d'exécution
- `CMD` : nom de la commande du processus

Testez également `ps -elf` (`l` comme long, pour encore plus d'informations), et `pstree`, qu'il vous faudra peut-être installer (`sudo apt install pstree`, éventuellement après `sudo apt-get`).
Tuez le processus d'écriture dans la deuxième fenêtre. Essayez de le faire de deux manières : avec son PID et avec le PID du parent (PPID).
Peut-on tuer le processus `bash` de la 2ème fenêtre dans tous les cas ? Pourquoi ?
Remarquez les « signaux envoyés » pour tuer le processus (on aurait pu les détailler en cours).

BLOCAGE DE PROCESSUS SOUS LINUX

La commande `flock` permet de gérer les verrous sous shell Linux. On va à nouveau travailler sur deux shells Linux côte à côte.
Créer un fichier `verrou0`.
Lancer la commande `flock -F verrou0 yes` , qui verrouille le fichier `verrou0`, et envoie un suite ininterrompue de « yes » sur la sortie standard.
Lancer la même commande dans la deuxième fenêtre. Que remarquez-vous ? Comment faire pour que la deuxième commande s'exécute (vous n'avez pas le droit à ctrl-c pour la 1ère commande, et pas le droit de tuer la deuxième commande) ?

COMPLÉMENT SUR LES PROGRAMMES EXÉCUTABLES

Les programmes exécutables ont un format de fichier particulier, qui est pris en charge par le noyau du système.
Ces formats sont différents suivant les systèmes :

- ELF (Executable and Linkable Format) sous Linux
- PE (Portable Executable) sous Windows
- Mach-O (Mach-object) sous Mac OS X

L'utilisation d'un format permet de segmenter le programme en plusieurs sections selon leur usage, comme : le code en langage machine (*qui sera accessible en lecture et en exécution*) du programme, les données telles que les constantes (a*ccessibles en lecture*), les variables globales et les données non initialisées (*accessibles en lecture et écriture*). Mais il permet aussi de fournir des informations importantes au système telle que le type d'exécutable, l'architecture à laquelle il est destiné, ou encore le point d'entrée du programme (c'est à dire l'adresse de début du code à exécuter).

Ces informations sont accessibles avec la commande `readelf`. Testez `readelf -a /bin/sleep` et remarquez/trouvez :

- Le système dans lequel le programme est exécutable ;
- Les machines sur lesquelles le programme est exécutable ;
- Le point d'entrée du programme (c'est-à-dire là où le programme commence en mémoire).

A quoi sert `Magic` (si ce n'est que tous les elfes pratiquent la magie) ?