

Gérer un ordinateur sous Linux

Attribution - Partage dans les Mêmes Conditions :
<http://creativecommons.org/licenses/by-sa/3.0/fr/>

Table des matières

I - Contexte	3
II - La mémoire vive sous Linux	4
III - Exercice : Appliquer la notion	6
IV - Utilisation du disque	7
V - Exercice : Appliquer la notion	10
VI - Les processus	11
VII - Exercice : Appliquer la notion	15
VIII - Les services	16
IX - Exercice : Appliquer la notion	19
X - Lire les logs	20
XI - Exercice : Appliquer la notion	23
XII - Quiz	24
Solutions des exercices	28
Crédits des ressources	35
Contenus annexes	36

I Contexte

Durée : 2h

Environnement de travail : Linux en ligne de commande

Pré-requis : Aucun

Pour bien utiliser son ordinateur, il convient d'en comprendre le fonctionnement. On pense aux composants et à comment ils interagissent entre eux, pour la partie physique, mais aussi être capable de trouver et interpréter des informations sur l'état d'exécution de la machine et des différents programmes, pour la partie logicielle. Ce cours va donner quelques outils pour connaître l'état de fonctionnement d'un ordinateur (utilisation de la mémoire, du disque, etc.) puis donner les méthodes utilisées pour piloter certains programmes essentiels au fonctionnement de l'ordinateur.

II La mémoire vive sous Linux

Objectifs

- Savoir obtenir l'utilisation de la RAM sur la machine
- Comprendre la subtilité entre mémoire libre et disponible

 Rappel

Architecture Von Neumann (cf. p.36)

Mémoire vive et mémoire secondaire (cf. p.39)

Utilisation mémoire vive

 Attention

La mémoire vive est essentielle au bon fonctionnement de l'ordinateur : si elle vient à manquer les différents programmes risquent de fortement ralentir (en attendant que de la mémoire se libère pour leur utilisation), voire même de cesser de fonctionner.

 Méthode

La commande `free` permet de mesurer l'utilisation globale de mémoire vive sur la machine. Ou utilise l'option `-h` pour utiliser un affichage simple à lire pour un humain.

```
1 $ free -h
2           total        used         free       shared  buff/cache   available
3 Mem:      7,7Gi        3,7Gi        2,3Gi        832Mi        1,7Gi        2,9Gi
4 Swap:      3,7Gi         0B          3,7Gi
```

L'information est donnée pour la mémoire RAM ainsi que pour le swap. On y trouve les 3 informations principales :

- la quantité de mémoire sur la machine, ici 7,7 Gio pour la RAM et 3,7 Gio pour le swap
- la quantité de mémoire utilisée actuellement, ici 3,7 Gio
- la quantité de mémoire libre, ici donc 2,3 Gio

Cependant si l'on fait la soustraction entre la mémoire totale et la mémoire utilisée, on n'obtient pas ce qui est indiqué comme mémoire disponible. Pour cela il faut se pencher sur les 3 colonnes suivantes :

- `shared` indique la part de mémoire qui est partagée entre plusieurs programmes, cela n'a pas beaucoup d'importance
- `buff/cache` indique la quantité de mémoire utilisée par l'OS pour ses buffers et du cache
- `available` indique une estimation de la mémoire qui est disponible si un programme en demande

Mémoire libre vs mémoire disponible

⊕ Complément

En réalité le système d'exploitation utilise une partie de la mémoire vive pour optimiser son fonctionnement et proposer une meilleure performance, en utilisant deux mécanismes.

Le premier est les `buffers`, qui permettent de stocker de l'information, pour une durée très courte, en attente de son traitement (par exemple un fichier qui vient d'arriver par le réseau et qui va être écrit sur le disque).

Le second est le `cache`, qui permet de garder en mémoire des informations que l'on pense réutiliser prochainement. Par exemple un fichier qui est souvent lu : pour accélérer la lecture, le système d'exploitation peut garder en `cache`, donc dans la RAM, une copie du fichier.

Les mécanismes de `buffers` ou de `cache` pourraient faire l'objet d'un cours entier, ce qu'il est important de retenir c'est que le système d'exploitation utilise une partie de la mémoire vive pour optimiser son fonctionnement. Et cette mémoire utilisée peut-être, en partie, libérée très rapidement car elle n'est pas essentielle au fonctionnement de l'ordinateur. C'est pour cela que l'on distingue la mémoire libre (celle qui est effectivement inutilisée) et la mémoire disponible (la mémoire libre + une partie des `buffers` et `cache` qui peuvent être libérés instantanément si besoin).

Multiples de l'octet

⚠ Attention

Lorsque l'on manipule des valeurs ayant pour unité des octets, on connaît généralement les multiples suivant :

- kilooctet (ko) pour 1000 octets
- mégaoctet (Mo) pour 1 million d'octets
- gigaoctet (Go) pour un milliard d'octets

C'est ce que l'on appelle les préfixes du Système International d'unités¹, ce sont des puissances de 10.

Historiquement en informatique on manipule souvent des puissances de 2, en particulier lorsque l'on mesure des quantités de donnée. Pour cela on utilise les préfixes binaires² :

- kibioctet (Kio) pour 1024 octets
- mébioctet (Mio) pour 2^{20} octets (1 048 576)
- gibioctet (Gio) pour 2^{30} octets (1 073 741 824)

Il est donc important de ne pas mélanger les deux unités, car même si l'ordre de grandeur reste similaire, la différence devient significative pour les multiples élevées (giga, téra, péta, etc.)

À retenir

La mémoire vive est utilisée aussi bien par les programmes que par le système d'exploitation, pour stocker des données de fonctionnement mais aussi pour optimiser les performances du système. On peut obtenir rapidement un aperçu de l'utilisation qui est faites grâce à la commande `free`.

1. https://fr.wikipedia.org/wiki/Pr%C3%A9fixes_du_Syst%C3%A8me_international_d%27unit%C3%A9s

2. https://fr.wikipedia.org/wiki/Pr%C3%A9fixe_binaire

III Exercice : Appliquer la notion

Question

[solution n°1 p. 28]

Sur votre machine, regardez la consommation mémoire lorsque peu d'applications sont ouvertes, par exemple pas de navigateur Web. Ensuite ouvrez votre navigateur Web avec une dizaine d'onglets et regardez la consommation de mémoire. Quelle quantité semble utiliser le navigateur ?

IV Utilisation du disque

Objectifs

- Savoir ce qu'est une partition
- Comprendre la notion de point de montage des disques sous Linux
- Savoir obtenir l'utilisation disque

Disques et partitions

Az Définition

Le système d'exploitation ne va pas directement utiliser le disque physique mais utiliser ce que l'on appelle une partition sur le disque. Une partition est une partie du disque sur laquelle l'OS va pouvoir gérer de l'espace de stockage comme il le souhaite, en y créant un système de fichier (c'est le protocole qui sera utilisé pour stocker les informations sur le disque).

Les partitions permettent d'isoler sur le disque des espaces de stockages destinés à des usages différents. Par exemple il est courant d'avoir une partition pour le stockage des données et une qui sert pour le swap. On peut aussi envisager d'avoir plusieurs systèmes d'exploitations installés sur le même ordinateur, chacun utilisera une partition qui lui est dédié sur le disque.

Afficher les disques

Méthode

Pour afficher les disques et leurs partitions sur la machine, on utilise la commande `fdisk`, véritable couteau suisse de la gestion des disques.

```
1 $ sudo fdisk -l
2 Disque /dev/sda : 232,89 GiB, 250059350016 octets, 488397168 secteurs
3 Modèle de disque : Crucial_CT250MX2
4 Unités : secteur de 1 × 512 = 512 octets
5 Taille de secteur (logique / physique) : 512 octets / 4096 octets
6 taille d'E/S (minimale / optimale) : 4096 octets / 4096 octets
7 Type d'étiquette de disque : dos
8 Identifiant de disque : 0xff6f6034
9
10 Périphérique Amorçage Début Fin Secteurs Taille Id Type
11 /dev/sda1 * 2048 1953791 1951744 953M 83 Linux
12 /dev/sda5 1955840 488396799 486440962 232G 83 Linux
```

Beaucoup d'informations s'affichent, comme le nom (`/dev/sda`), la taille (232,89 GiB) ou le modèle du disque (`Crucial_CT250MX2`). En bas, la liste des partitions qui se trouvent sur le disque :

- `/dev/sda1` qui fait 953 Mo
- `/dev/sda5` qui fait 232 Go

Si il y a plusieurs disques dans la machine, ils s'afficheront à la suite, sous le même format.

Montage de partition sous Linux

Pour le stockage des fichiers, Linux va associer un point de montage à chaque partition. Un montage indique à un dossier dans l'arborescence qu'il doit stocker tout les fichiers et sous-dossiers qui le compose sur une même partition du disque. Sur les ordinateurs simples, il n'y a souvent qu'une seule partition pour un seul point de montage : / qui utilise une unique partition pour stocker la totalité des fichiers. Cependant on peut imaginer qu'une partie de l'arborescence, par exemple /home, soit stockée sur une partition à part.

La commande df

[Méthode](#)

La commande `df` permet d'obtenir de nombreuses informations sur l'utilisation du disque sur la machine.

```
1 $ df -Th
2 Sys. de fichiers          Type      Taille Utilisé Dispo Uti% Monté sur
3 udev                     devtmpfs  3,9G   0    3,9G   0% /dev
4 tmpfs                    tmpfs     785M   1,7M 783M   1% /run
5 /dev/sda5                 ext4     230G  128G   88G   60% /
6 tmpfs                    tmpfs     3,9G   242M 3,6G   7% /dev/shm
7 tmpfs                    tmpfs     5,0M   4,0K 5,0M   1% /run/lock
8 tmpfs                    tmpfs     4,0M   0    4,0M   0% /sys/fs/cgroup
9 /dev/sda1                 ext2     938M  209M 730M  23% /boot
10 tmpfs                   tmpfs     785M   44K 784M   1% /run/user/1000
```

On trouve, des informations dans chaque colonne :

- le nom de la partition
- le type de système de fichier
- la taille totale
- l'espace utilisé
- l'espace disponible
- un pourcentage de l'espace utilisé
- le point de montage de la partition

Le premier constat est qu'il y a beaucoup plus de lignes qu'on ne pensait. En effet la commande affiche, en plus des partitions, les zones mémoires `tmpfs` se trouvant dans la mémoire vive, ainsi qu'un système de fichier nommé `udev`. Ce sont des espaces utilisés et gérés par le système d'exploitation qui ne nous intéresse pas vraiment ici, on peut donc les filtrer.

```
1 $ df -Th -x tmpfs -x devtmpfs
2 Sys. de fichiers          Type      Taille Utilisé Dispo Uti% Monté sur
3 /dev/sda5                 ext4     230G  128G   88G   60% /
4 /dev/sda1                 ext2     938M  209M 730M  23% /boot
```

On observe donc que la partition `/dev/sda5` est montée sur `/` et qu'elle est utilisée à 60%. L'autre partition `/dev/sda1` est utilisée pour stocker ce qui se trouve dans `/boot` (dossier contenant le nécessaire pour démarrer, en particulier le noyau de Linux).

Connaître la taille des fichiers

[Méthode](#)

Si la commande `df` permet d'obtenir une idée de l'utilisation du disque, cela n'est cependant pas très précis. Par exemple si l'on voit qu'une partition est bientôt remplie, on veut ensuite identifier clairement ce qui prends de la place.

Pour cela on utilise la commande du.

```
1 $ du -sh documents
2 13G  documents
```

La commande prends en paramètre un fichier/dossier et retourne la taille de celui-ci grâce à l'option -s. Sans cette option, et lorsqu'il s'agit d'un dossier, la commande affiche séparément la taille de la totalité des fichiers et sous-dossiers, de manière récursive (ce qui n'est pas très pratique). On peut aussi donner une liste en paramètres de la fonction.

```
1 kyane@europa:~$ du -sh ./*
2 1,8G  ./code
3 13G   ./documents
4 28G   ./downloads
5 12G   ./media
```

À retenir

Le système d'exploitation utilise des partitions de disque en tant qu'espace de stockage. Chaque partition stocke une partie de l'arborescence des fichiers, souvent la totalité lorsque / est monté sur une unique partition. En utilisant les commandes df et du il est possible d'avoir des informations sur l'utilisation des disques, et la taille des fichiers.

V Exercice : Appliquer la notion

Après avoir utilisé la commande `df`, vous remarquez qu'une partition est bientôt remplie.

```
1 $ df -Th -x tmpfs -x devtmpfs
2 Sys. de fichiers      Type      Taille Utilisé Dispo Uti% Monté sur
3 /dev/sda1             ext2      938M   209M   730M   23% /boot
4 /dev/sda2             ext4     225G   126G   88G    60% /
5 /dev/sda3             ext4     2,5G   2,4G   0,1G   96% /var
```

Question

[solution n°2 p. 28]

Comment investiguer pour trouver ce qui prends le plus de place ? Quelle commande utiliser pour obtenir une liste classée des dossiers et fichiers dans `/var` ?

VI Les processus

Objectifs

- Savoir ce qu'est un processus
- Savoir lister les processus et prendre en compte les ressources qu'ils utilisent

Processus

Az Définition

Un processus est un programme en cours d'exécution. Il exécute des instructions sur le processeur et peut utiliser de la mémoire vive ou de l'espace disque. Lorsque l'on démarre un logiciel, celui-ci va créer un ou plusieurs processus pour effectuer ses tâches.

Chaque processus possède un identifiant unique appelé PID

Processus parent

Az Définition

Chaque processus possède un processus parent.

En effet chaque processus est créé par un autre processus, qui est donc son processus parent. Le seul processus à ne pas avoir de parent est le premier à être lancé au démarrage : il s'agit du processus `init`, qui a pour PID 1, et qui démarre le système d'exploitation.

Lister les processus : `ps -ax`

Syntaxe

```
1 ps -ax
```

La commande `ps` permet de lister les processus. On utilise l'option `ax` pour afficher tous les processus de la machine (pas uniquement ceux de la console ouverte ou ceux ayant une interface graphique).

Exemple

```
1 ps ax
2  PID TTY          STAT       TIME COMMAND
3    1 ?        Ss          0:09 /sbin/init
4    2 ?        S           0:00 [kthreadd]
5    3 ?        I<          0:00 [rcu_gp]
6    4 ?        I<          0:00 [rcu_par_gp]
7    6 ?        I<          0:00 [kworker/0:0H-kblockd]
8    9 ?        I<          0:00 [mm_percpu_wq]
9   10 ?        S           0:00 [ksoftirqd/0]
10   11 ?        I           0:12 [rcu_sched]
11   12 ?        S           0:00 [migration/0]
12   13 ?        S           0:00 [cpuhp/0]
13   14 ?        S           0:00 [cpuhp/1]
14   15 ?        S           0:00 [migration/1]
15   16 ?        S           0:00 [ksoftirqd/1]
16   18 ?        I<          0:00 [kworker/1:0H-events_highpri]
17   19 ?        S           0:00 [cpuhp/2]
```

```

18      20 ?          S      0:00 [migration/2]
19      21 ?          S      0:19 [ksoftirqd/2]
20 [...]
21     15371 ?        S      0:00 /bin/sh -c /usr/lib/firefox/firefox
22     15372 ?        Sl     16:56 /usr/lib/firefox/firefox
23     15433 ?        Sl     7:14 /usr/lib/firefox/firefox -contentproc -childID 1 -
isForBrowser -prefsLen 1 -prefMapSize 236261 -parentBuildID 20201112153044 -
appdir /usr/lib/firefox/browser 153
24     15504 ?        Sl     1:58 /usr/lib/firefox/firefox -contentproc -childID 3 -
isForBrowser -prefsLen 6414 -prefMapSize 236261 -parentBuildID 20201112153044 -
appdir /usr/lib/firefox/browser
25     15554 ?        Sl     3:53 /usr/lib/firefox/firefox -contentproc -childID 4 -
isForBrowser -prefsLen 7212 -prefMapSize 236261 -parentBuildID 20201112153044 -
appdir /usr/lib/firefox/browser
26     27333 ?         S      0:00 /bin/sh -c spotify
27     27334 ?        Sl     0:43 spotify
28     27337 ?         S      0:00 /usr/share/spotify/spotify --type=zygote --no-
zygote-sandbox --no-sandbox --log-file=/usr/share/spotify/debug.log --log-
severity=disable --product-version=Spotif
29     27338 ?         S      0:00 /usr/share/spotify/spotify --type=zygote --no-
sandbox --log-file=/usr/share/spotify/debug.log --log-severity=disable --
product-version=Spotify/1.1.42.622 --lang=
30     27352 ?        Sl     0:30 /usr/share/spotify/spotify --type=gpu-process --
field-trial-handle=10118108429751651102,14475875873737677976,131072 --enable-
features=CastMediaRouteProvider --di
31     27363 ?        Sl     0:02 /usr/share/spotify/spotify --type=utility --
utility-sub-type=network.mojom.NetworkService --field-trial-
handle=10118108429751651102,14475875873737677976,131072 -
32     27379 ?        Sl     1:18 /usr/share/spotify/spotify --type=renderer --no-
sandbox --log-file=/usr/share/spotify/debug.log --field-trial-
handle=10118108429751651102,14475875873737677976,13
33     28731 pts/0      R+     0:00 ps ax
34

```

La résultat est ici tronqué, car on constate qu'il y a beaucoup de processus qui s'exécutent en simultané sur la machine. Pour chaque, on peut voir leur PID (première colonne) ainsi que leur nom (dernière colonne). La plupart ont des noms qui ne sont pas très parlant, en fait il s'agit de la commande qui a servi à lancer le processus. Cependant avec un peu de déduction, on peut supposer ici qu'il y a Firefox et Spotify qui sont en train de fonctionner.

On remarque aussi le processus 1 /sbin/init, ainsi que le dernier de la liste ps ax : c'est la commande que l'on vient de lancer, qui est un processus comme les autres.

Utilisation des ressources

 Remarque

L'un des rôles du système d'exploitation est d'allouer les ressources à chaque processus, en particulier de l'espace en mémoire vive et du temps d'exécution sur le processeur.

Chaque processus utilisant des ressources sur le système, il est intéressant de pouvoir savoir quel processus utilise quoi. Par exemple si l'on manque de mémoire vive, identifier le processus le plus gourmand s'avère intéressant pour ramener le système à un état stable (soit en coupant ce processus si il n'est pas utile, ou en allégeant les tâches qu'il fait, ou en ajoutant de la mémoire, etc.). Il existe de nombreuses manière d'obtenir cette information, l'option "u" de la commande ps par exemple. Mais l'outil favori des administrateurs système est sans doute doute htop.

The screenshot shows the htop interface with a list of processes. The top of the list is highlighted in blue. The processes are all named 'contentproc' and are children of the 'Firefox' process. They are all running under the 'kyane' user. The CPU usage for all these processes is 0.5%. The memory usage is 9.5% for each process. The command column shows the path to the Firefox binary: '/usr/lib/firefox/firefox'. The parent PID is 1 for all these processes. The parent build ID is 2020112153044. The prefLen is 1 and the prefMapSize is 236261 for all these processes. The parent build ID is 2020112153044.

PID	USER	NI	MI	VI	RES	SHR	CPU	MEM	TIME	COMMAND
15433	kyane	20	0	4067H	748M	381K	S	0.5	13:15.54	/usr/lib/firefox/firefox -contentproc -childID 1 -isForBrowser -prefLen 1 -prefMapSize 236261 -parentBuildID 2020112153044
15438	kyane	20	0	4067H	748M	381K	S	0.5	0:28.22	/usr/lib/firefox/firefox -contentproc -childID 1 -isForBrowser -prefLen 1 -prefMapSize 236261 -parentBuildID 2020112153044
15438	kyane	20	0	4067H	748M	381K	S	0.5	0:10.26	/usr/lib/firefox/firefox -contentproc -childID 1 -isForBrowser -prefLen 1 -prefMapSize 236261 -parentBuildID 2020112153044
15438	kyane	20	0	4067H	748M	381K	S	0.5	0:00.86	/usr/lib/firefox/firefox -contentproc -childID 1 -isForBrowser -prefLen 1 -prefMapSize 236261 -parentBuildID 2020112153044
15437	kyane	20	0	4067H	748M	381K	S	0.5	0:02.74	/usr/lib/firefox/firefox -contentproc -childID 1 -isForBrowser -prefLen 1 -prefMapSize 236261 -parentBuildID 2020112153044
15439	kyane	20	0	4067H	748M	381K	S	0.5	0:07.17	/usr/lib/firefox/firefox -contentproc -childID 1 -isForBrowser -prefLen 1 -prefMapSize 236261 -parentBuildID 2020112153044
15442	kyane	20	0	4067H	748M	381K	S	0.5	0:07.30	/usr/lib/firefox/firefox -contentproc -childID 1 -isForBrowser -prefLen 1 -prefMapSize 236261 -parentBuildID 2020112153044
15441	kyane	20	0	4067H	748M	381K	S	0.5	0:00.84	/usr/lib/firefox/firefox -contentproc -childID 1 -isForBrowser -prefLen 1 -prefMapSize 236261 -parentBuildID 2020112153044
15445	kyane	20	0	4067H	748M	381K	S	0.5	0:30.62	/usr/lib/firefox/firefox -contentproc -childID 1 -isForBrowser -prefLen 1 -prefMapSize 236261 -parentBuildID 2020112153044
15452	kyane	20	0	4067H	748M	381K	S	0.5	0:03.60	/usr/lib/firefox/firefox -contentproc -childID 1 -isForBrowser -prefLen 1 -prefMapSize 236261 -parentBuildID 2020112153044
15440	kyane	20	0	4067H	748M	381K	S	0.5	0:06.01	/usr/lib/firefox/firefox -contentproc -childID 1 -isForBrowser -prefLen 1 -prefMapSize 236261 -parentBuildID 2020112153044
15444	kyane	20	0	4067H	748M	381K	S	0.5	0:00.81	/usr/lib/firefox/firefox -contentproc -childID 1 -isForBrowser -prefLen 1 -prefMapSize 236261 -parentBuildID 2020112153044
15453	kyane	20	0	4067H	748M	381K	S	0.5	0:00.18	/usr/lib/firefox/firefox -contentproc -childID 1 -isForBrowser -prefLen 1 -prefMapSize 236261 -parentBuildID 2020112153044
15454	kyane	20	0	4067H	748M	381K	S	0.5	0:00.27	/usr/lib/firefox/firefox -contentproc -childID 1 -isForBrowser -prefLen 1 -prefMapSize 236261 -parentBuildID 2020112153044
15455	kyane	20	0	4067H	748M	381K	S	0.5	0:04.75	/usr/lib/firefox/firefox -contentproc -childID 1 -isForBrowser -prefLen 1 -prefMapSize 236261 -parentBuildID 2020112153044
15456	kyane	20	0	4067H	748M	381K	S	0.5	0:05.66	/usr/lib/firefox/firefox -contentproc -childID 1 -isForBrowser -prefLen 1 -prefMapSize 236261 -parentBuildID 2020112153044
15457	kyane	20	0	4067H	748M	381K	S	0.5	0:00.80	/usr/lib/firefox/firefox -contentproc -childID 1 -isForBrowser -prefLen 1 -prefMapSize 236261 -parentBuildID 2020112153044
15458	kyane	20	0	4067H	748M	381K	S	0.5	0:00.80	/usr/lib/firefox/firefox -contentproc -childID 1 -isForBrowser -prefLen 1 -prefMapSize 236261 -parentBuildID 2020112153044
15459	kyane	20	0	4067H	748M	381K	S	0.5	0:00.09	/usr/lib/firefox/firefox -contentproc -childID 1 -isForBrowser -prefLen 1 -prefMapSize 236261 -parentBuildID 2020112153044
15597	kyane	20	0	4067H	748M	381K	S	0.5	0:04.73	/usr/lib/firefox/firefox -contentproc -childID 1 -isForBrowser -prefLen 1 -prefMapSize 236261 -parentBuildID 2020112153044
15598	kyane	20	0	4067H	748M	381K	S	0.5	0:04.84	/usr/lib/firefox/firefox -contentproc -childID 1 -isForBrowser -prefLen 1 -prefMapSize 236261 -parentBuildID 2020112153044
15599	kyane	20	0	4067H	748M	381K	S	0.5	0:04.81	/usr/lib/firefox/firefox -contentproc -childID 1 -isForBrowser -prefLen 1 -prefMapSize 236261 -parentBuildID 2020112153044
15988	kyane	20	0	4067H	748M	381K	S	0.5	0:00.70	/usr/lib/firefox/firefox -contentproc -childID 1 -isForBrowser -prefLen 1 -prefMapSize 236261 -parentBuildID 2020112153044
16178	kyane	20	0	4067H	748M	381K	S	0.5	0:03.61	/usr/lib/firefox/firefox -contentproc -childID 1 -isForBrowser -prefLen 1 -prefMapSize 236261 -parentBuildID 2020112153044
22450	kyane	20	0	4067H	748M	381K	S	0.5	0:11.40	/usr/lib/firefox/firefox -contentproc -childID 1 -isForBrowser -prefLen 1 -prefMapSize 236261 -parentBuildID 2020112153044
22553	kyane	-11	0	4067H	748M	381K	S	0.5	0:02.19	/usr/lib/firefox/firefox -contentproc -childID 1 -isForBrowser -prefLen 1 -prefMapSize 236261 -parentBuildID 2020112153044
26663	kyane	20	0	4067H	748M	381K	S	0.5	0:00.18	/usr/lib/firefox/firefox -contentproc -childID 1 -isForBrowser -prefLen 1 -prefMapSize 236261 -parentBuildID 2020112153044
27800	kyane	39	15	4067H	748M	381K	S	0.5	0:00.00	/usr/lib/firefox/firefox -contentproc -childID 1 -isForBrowser -prefLen 1 -prefMapSize 236261 -parentBuildID 2020112153044
27801	kyane	39	15	4067H	748M	381K	S	0.5	0:00.00	/usr/lib/firefox/firefox -contentproc -childID 1 -isForBrowser -prefLen 1 -prefMapSize 236261 -parentBuildID 2020112153044
27802	kyane	39	15	4067H	748M	381K	S	0.5	0:00.00	/usr/lib/firefox/firefox -contentproc -childID 1 -isForBrowser -prefLen 1 -prefMapSize 236261 -parentBuildID 2020112153044
27803	kyane	39	15	4067H	748M	381K	S	0.5	0:00.00	/usr/lib/firefox/firefox -contentproc -childID 1 -isForBrowser -prefLen 1 -prefMapSize 236261 -parentBuildID 2020112153044
27805	kyane	20	0	4067H	748M	381K	S	0.5	0:00.00	/usr/lib/firefox/firefox -contentproc -childID 1 -isForBrowser -prefLen 1 -prefMapSize 236261 -parentBuildID 2020112153044
32289	kyane	20	0	4067H	748M	381K	S	0.5	0:00.00	/usr/lib/firefox/firefox -contentproc -childID 1 -isForBrowser -prefLen 1 -prefMapSize 236261 -parentBuildID 2020112153044
15372	kyane	20	0	3808H	650M	100K	S	4.0	8:3	22:21.24 /usr/lib/firefox/firefox
15404	kyane	20	0	3808H	650M	100K	S	1.3	8.3	13:36.03 /usr/lib/firefox/firefox
15375	kyane	20	0	3808H	650M	100K	S	1.3	8.3	2:30.12 /usr/lib/firefox/firefox
15403	kyane	20	0	3808H	650M	100K	S	0.7	8.3	1:07.39 /usr/lib/firefox/firefox
15408	kyane	20	0	3808H	650M	100K	S	0.0	8.3	1:14.53 /usr/lib/firefox/firefox
15376	kyane	20	0	3808H	650M	100K	S	0.0	8.3	0:08.07 /usr/lib/firefox/firefox
27808	kyane	39	15	4067H	748M	381K	S	0.7	8.3	0:01.59 /usr/lib/firefox/firefox
15378	kyane	20	0	3808H	650M	100K	S	0.0	8.3	0:22.36 /usr/lib/firefox/firefox
15382	kyane	20	0	3808H	650M	100K	S	0.0	8.3	0:01.35 /usr/lib/firefox/firefox
15476	kyane	20	0	3808H	650M	100K	S	0.0	8.3	0:20.31 /usr/lib/firefox/firefox

On voit par exemple ici de nombreux processus pour Firefox, qui consomment 9,5% de la RAM de la machine.

À retenir

Tout les programmes créent des processus sur la machine pour pouvoir exécuter des instructions. Ces processus consomment des ressources (mémoire vive, temps processeur) qui sont réparties par le système d'exploitation. Il est possible de lister les processus et leur utilisation des ressources avec la commande htop, pour identifier d'éventuels processus trop gourmands.

➕ Complément

htop explained: explanation of everything you can see in htop/top on Linux, peteris.rocks/blog/htop, 2019.

VII Exercice : Appliquer la notion

Nous allons utiliser htop pour observer un peu les processus sur la machine.

Question 1

[solution n°3 p. 29]

Ouvrez htop et trouvez le processus qui utilise le plus le processeur.

Question 2

[solution n°4 p. 29]

On souhaiterait connaître l'arborescence de parents de ce processus. Comment peut-on procéder ?

VIII Les services

Objectifs

- Savoir ce qu'est un service, ou daemon.
- Savoir piloter les services avec `systemctl`.

Mise en situation

Lorsque l'on liste les processus en cours d'exécution, on peut effectivement voir les programmes que l'on a démarré ou les commandes que l'on a lancé. Cependant, en cherchant un peu, on peut aussi voir des processus qui semblent inconnus, comme `rsyslogd`, `ntpd`, etc. Ce sont en fait des programmes qui s'exécutent en arrière plan et que l'on appelle des services.

Service

Az Définition

Un service (parfois appelé *daemon*) est un programme qui s'exécute en arrière-plan, plutôt contrôlé par le système d'exploitation que par l'utilisateur directement.

👁 Exemple

Les services peuvent remplir de nombreuses fonctions différentes, quelques exemple :

- NetworkManager est un service sur certaines distribution Linux qui permet de gérer la configuration réseau
- NTPD est un service qui sert à gérer l'heure de la machine (grâce au protocole NTP³)
- `unattended-upgrades` permet de gérer les mises à jour automatiquement, en arrière plan, sur les systèmes utilisant APT

Systemd

Sur Ubuntu, comme beaucoup autres distributions Linux, un logiciel est au coeur du système d'exploitation et permet, entre autre, de gérer les différents services : Systemd. Systemd est le logiciel qui initialise le système d'exploitation au démarrage (dont le fameux processus `init` fait partie) et qui démarre et pilote les différents services de la machine.

systemctl

🔧 Méthode

Systemd est un logiciel très complexe, nous ne nous intéresseront ici que à la partie qui permet de gérer les services de la machine. Pour cela on peut utiliser la commande `systemctl`

```
1 systemctl --type=service
```

³ https://fr.wikipedia.org/wiki/Network_Time_Protocol

Cette commande permet de lister tout les services sur la machine, associé à leur description. On peut voir qu'il y en a plusieurs dizaines par défaut. Il est possible d'afficher le statut d'un service :

```

1 kyane@europa:~$ systemctl status rsyslog
2 ● rsyslog.service - System Logging Service
3    Loaded: loaded (/lib/systemd/system/rsyslog.service; enabled; vendor
   preset: enabled)
4    Active: active (running) since Mon 2020-12-14 09:54:21 CET; 4h 54min ago
5 TriggeredBy: ● syslog.socket
6    Docs: man:rsyslogd(8)
7          man:rsyslog.conf(5)
8          https://www.rsyslog.com/doc/
9    Main PID: 777 (rsyslogd)
10   Tasks: 4 (limit: 9345)
11   Memory: 2.5M
12   CGroup: /system.slice/rsyslog.service
13           └─777 /usr/sbin/rsyslogd -n -iNONE
14
15 déc. 14 09:54:20 europa systemd[1]: Starting System Logging Service...
16 déc. 14 09:54:21 europa rsyslogd[777]: imuxsock: Acquired UNIX socket
   '/run/systemd/journal/syslog' (fd 3) from systemd. [v8.2010.0]
17 déc. 14 09:54:21 europa systemd[1]: Started System Logging Service.
18 déc. 14 09:54:21 europa rsyslogd[777]: [origin software="rsyslogd"
   swVersion="8.2010.0" x-pid="777" x-info="https://www.rsyslog.com"] start

```

Ici on voit le statut du service rsyslog, un service qui gère les messages d'erreurs ou d'information pour les autres logiciels sur la machine. L'information importante (qui s'affiche en couleur dans le terminal) est que le service est active (running), donc qu'il est en fonctionnement. On peut aussi voir, sur la seconde ligne de résultat, que le service est enabled, ce qui signifie qu'il est lancé automatiquement au démarrage de l'ordinateur.

Cycle de vie d'un service

Un service peut-être dans plusieurs états, les principaux étant :

- active lorsqu'il est démarré (ou activating pendant la phase de démarrage)
- inactive lorsqu'il est arrêté (ou deactivating pendant la phase d'extinction)
- failed lorsqu'il s'est arrêté suite à une erreur

Méthode

L'état d'un service peut-être modifié avec les commandes suivantes :

- `systemctl start SERVICE` pour démarrer un service stoppé
- `systemctl stop SERVICE` pour stopper un service en fonctionnement
- `systemctl restart SERVICE` pour redémarrer un service, cela revient à faire un stop puis un start
- `systemctl reload SERVICE` pour demander à un service de recharger sa configuration sans s'arrêter

 Remarque

Dans le cas d'une utilisation comme ordinateur personnel, il est peu courant de manipuler les services, ou même Systemd. En effet les services sont gérés par Systemd, qui les démarre et les arrête en fonction du besoin. Cependant dans le cas d'un serveur, il arrive que l'on ajoute des services à la machine (par exemple un serveur web, ou une base de donnée) et qu'il soit nécessaire de les piloter avec `systemctl`. Il est donc important de garder en tête ces notions et commandes de base.

À retenir

Un programme qui s'exécute en arrière plan est un service. Les services sont gérés par Systemd sous Ubuntu, mais il est possible de les piloter soi-même à l'aide de la commande `systemctl`.

IX Exercice : Appliquer la notion

Question 1

[solution n°5 p. 30]

Un service nommé `cron` est installé par défaut sur la plupart des distributions Linux. D'après sa description, à quoi sert-il ?

Question 2

[solution n°6 p. 30]

Quel est son état ? Est-il systématiquement démarré ?

X Lire les logs

Objectifs

- Comprendre ce que sont les logs
- Savoir trouver les fichiers de logs
- Savoir consulter les logs d'un service

Les logs

Az Définition

Les logs sont des messages produits par les programmes pour donner des informations sur leur état de fonctionnement et/ou les erreurs qui surviennent lors de l'exécution.

Les différents programmes ou services du système d'exploitation peuvent écrire des logs qui seront consultables de différentes manières. La plupart se trouvent dans un dossier nommé `/var/log`, mais certains sont gérés par Systemd.

👁 Exemple

Dans le dossier `/var/log` d'une machine, on constate qu'il y a de nombreux fichiers et dossiers pour contenir les logs de différents types.

```
1 $ ls /var/log/
2 alternatives.log boot.log cups dpkg.log fsck
  lastlog mail.warn runit tallylog vbox-setup.log
3 apache2 bootstrap.log daemon.log exim4 hp
  lpr.log messages samba tor wtmp
4 apt borg-backup.log debug faillog installer
  mail.err ntpstats slim.log ufw.log Xorg.0.log
5 aptitude btmp dibbler firebird journal
  mail.info private snort unattended-upgrades
6 auth.log cron.log dmesg fontconfig.log kern.log
  mail.log prometheus syslog user.log
```

Il n'est pas important de savoir à quoi sert chaque fichier de log. Ils portent généralement un nom explicite (celui du service ou du logiciel).

Syslog

Az Définition

Pour faciliter le traitement des messages de tout les programmes existant, un protocole existe : Syslog⁴. C'est un protocole qui définit aussi bien la manière d'envoyer des messages de logs entre différents programmes que le format standardisé que doivent avoir des messages.

⁴ <https://fr.wikipedia.org/wiki/Syslog>

Syslog et rsyslog

 Méthode

Sous Linux, un fichier `/var/log/syslog` va contenir tout les logs échangés sur la machine en utilisant le protocole Syslog. Généralement, le service ne charge de la gestion de ces messages est le service `rsyslog`.

```

1 $ tail /var/log/syslog
2 Dec 14 16:12:18 europa prometheus-node-exporter[1564]: level=error ts=2020-12-
  14T15:12:18.884Z caller=collector.go:161 msg="collector failed" name=rapl
  duration_seconds=0.005177803 err="open /sys/class/powercap/intel-
  rapl:0/energy_uj: permission denied"
3 Dec 14 16:13:08 europa systemd[1]: Starting Collect apt metrics for prometheus-
  node-exporter...
4 Dec 14 16:13:08 europa systemd[1]: Starting Collect SMART metrics for
  prometheus-node-exporter...
5 Dec 14 16:13:08 europa systemd[1]: prometheus-node-exporter-smartmon.service:
  Succeeded.
6 Dec 14 16:13:08 europa systemd[1]: Finished Collect SMART metrics for
  prometheus-node-exporter.
7 Dec 14 16:13:09 europa systemd[1]: prometheus-node-exporter-apt.service:
  Succeeded.
8 Dec 14 16:13:09 europa systemd[1]: Finished Collect apt metrics for prometheus-
  node-exporter.
9 Dec 14 16:13:18 europa prometheus-node-exporter[1564]: level=error ts=2020-12-
  14T15:13:18.880Z caller=collector.go:161 msg="collector failed" name=rapl
  duration_seconds=0.000294815 err="open /sys/class/powercap/intel-
  rapl:0/energy_uj: permission denied"
10 Dec 14 16:14:18 europa prometheus-node-exporter[1564]: level=error ts=2020-12-
  14T15:14:18.891Z caller=collector.go:161 msg="collector failed" name=rapl
  duration_seconds=0.002966102 err="open /sys/class/powercap/intel-
  rapl:0/energy_uj: permission denied"
11 Dec 14 16:15:18 europa prometheus-node-exporter[1564]: level=error ts=2020-12-
  14T15:15:18.890Z caller=collector.go:161 msg="collector failed" name=rapl
  duration_seconds=0.000811931 err="open /sys/class/powercap/intel-
  rapl:0/energy_uj: permission denied"

```

Ce fichier est un bon point d'entrée pour investiguer un problème sur la machine. Chaque ligne représente un log avec le format suivant :

- la date où le log a été émis ;
- le nom de la machine (ici europa), le protocole Syslog permet de centraliser les logs de plusieurs machines au même endroit si besoin ;
- le nom du processus qui a émis le log, et son PID entre crochet. Ici on voit des logs émis par `systemd` (PID 1) et `prometheus-node-exporter` (PID 1564) ;
- le message effectivement émis par le programme.

Journald

`Systemd` s'accompagne d'un service, `journald`, qui gère les logs de tout les services de la machine. En fait, lorsque l'on utilise la commande `systemctl status SERVICE` on peut observer une partie des logs les plus récents du service, gérés par `journald`.

 Méthode

La commande `journalctl` qui va avec `journald` permet de consulter des logs de manière plus pratique que de simplement regarder dans des fichiers de logs. Par exemple elle permet de n'obtenir les messages de logs que d'un service en particulier.

```
1 journalctl -u cron
```

On peut aussi lister tout les logs, mais uniquement pour depuis l'allumage de la machine (et non ceux des précédentes sessions d'utilisation).

```
1 journalctl -b
```

Il est même possible d'être plus précis, et de demander les logs d'une durée spécifique

```
1 journalctl --since "1 hour ago"
```

```
1 journalctl --since "2020-12-01 20:15:00" --until "2020-12-10 14:21:42"
```

Enfin ajouter l'option `-f` permet d'afficher en direct les nouveaux logs produits. Très utile lors d'une phase de debug.

À retenir

Tout les programmes produisent des messages pour donner des informations sur leur état de fonctionnement. Ces messages sont écrits dans le dossier `/var/log` de la machine, mais peuvent être consultés de manière plus ergonomique, dans le cas des services, à l'aide de la commande `journalctl`.

XI Exercice : Appliquer la notion

Sur votre machine, choisissez un service existant pour lequel consulter les logs. Cela peut-être apache2 pour un serveur web, ou NetworkManager sur un ordinateur personnel.

Question

[solution n°7 p. 30]

Comment afficher les logs de ce service uniquement, sur les 2 derniers jours, en affichant en continu les nouveaux logs produits ?

XII Quiz

Exercice 1 : Quiz - Culture

[solution n°8 p. 30]

Exercice

À quel endroit le système d'exploitation peut stocker une information en cache, pour y accéder plus rapidement ?

A Dans la mémoire vive (RAM)

B Sur le disque

C Dans un *buffer*

D Sur un service systemd

Exercice

Une partition est :

A Un bout de fichier stocké sur le disque

B Le buffer utilisé par le système d'exploitation avant de jouer les sons sur le haut-parleur

C Une partie de disque physique utilisée par le système d'exploitation

Exercice

Combien d'octets représente 1 Mo

A 1 000 000

B 1 048 576

C 1000

D 1024

Exercice 5 : Quiz - Méthode

[solution n°9 p. 31]

Exercice

Comment obtenir un aperçu de la mémoire vive utilisée sur la machine

A Avec la commande free.

B Avec la commande memory.

C Avec la commande usage.

D Avec la commande htop.

Exercice

Comment redémarrer le service nommé rsyslog ?

A rsyslog restart

B service rsyslog restart

C systemctl rsyslog restart

D service restart rsyslog

E systemctl restart rsyslog

Exercice

Comment obtenir une liste des processus en exécution sur la machine ?

A Avec la commande htop

B En faisant ls /proc

C Avec la commande ps

D Avec la commande proc

Exercice 9 : Quiz - Code

[solution n°10 p. 32]

Exercice

Dans quel état se trouve le service suivant ?

```

1 $ systemctl status minissdpd.service
2 ● minissdpd.service - keep memory of all UPnP devices that announced themselves
3    Loaded: loaded (/lib/systemd/system/minissdpd.service; enabled; vendor preset:
4    enabled)
5    Active: failed (Result: exit-code) since Mon 2020-12-14 09:54:38 CET; 7h ago
6    Docs: man:minissdpd(1)
7    Process: 1385 ExecCondition=/usr/lib/minissdpd/minissdpd-systemd-wrapper -t
8           ${MiniSSDPd_INTERFACE_ADDRESS} (code=exited, status=0/SUCCESS)

```

```

7 Process: 1403 ExecStart=/usr/lib/minissdpd/minissdpd-systemd-wrapper
  ${MiniSSDPd_INTERFACE_ADDRESS} $MiniSSDPd_OTHER_OPTIONS (code=exited,
  status=1/FAILURE)
8
9 déc. 14 09:54:38 europa minissdpd-systemd-wrapper[1403]: Error parsing address/mask
  (or interface name) : eth0
10 déc. 14 09:54:38 europa minissdpd-systemd-wrapper[1403]: can't parse "eth0" as a
  valid address or interface name
11 déc. 14 09:54:38 europa minissdpd-systemd-wrapper[1403]: Usage: /usr/sbin/minissdpd
  [-d] [-6] [-s socket] [-p pidfile] [-t TTL] [-f device] -i <interface> [-i
  <interface2>] ...
12 déc. 14 09:54:38 europa minissdpd-systemd-wrapper[1403]: <interface> is either an
  IPv4 address with mask such as
13 déc. 14 09:54:38 europa minissdpd-systemd-wrapper[1403]:
  192.168.1.42/255.255.255.0, or an interface name such as eth0.
14 déc. 14 09:54:38 europa minissdpd-systemd-wrapper[1403]: By default, socket will
  be open as /var/run/minissdpd.sock
15 déc. 14 09:54:38 europa minissdpd-systemd-wrapper[1403]: and pid written to file
  /var/run/minissdpd.pid
16 déc. 14 09:54:38 europa systemd[1]: minissdpd.service: Control process exited,
  code=exited, status=1/FAILURE
17 déc. 14 09:54:38 europa systemd[1]: minissdpd.service: Failed with result 'exit-
  code'.
18 déc. 14 09:54:38 europa systemd[1]: Failed to start keep memory of all UPnP devices
  that announced themselves.
19

```

 A active B inactive C failed

Exercice

Que pouvons nous déduire de la ligne, au format Syslog, suivante ?

```

1 Dec 14 15:54:38 europa dbus-daemon[762]: [system] Successfully activated service
  'org.freedesktop.nm_dispatcher'

```

 A La machine se nomme dbus - daemon B Le log date du 14 décembre C Le service ayant produit le log est system D Le PID du service est 762

Exercice

Voici le résultat d'une commande. Combien est-ce qu'il y a de disques et de partitions ?

```

1 $ sudo fdisk -l
2 Disk /dev/vda: 9.3 GiB, 10000000000 bytes, 19531250 sectors
3 Units: sectors of 1 * 512 = 512 bytes
4 Sector size (logical/physical): 512 bytes / 512 bytes
5 I/O size (minimum/optimal): 512 bytes / 512 bytes
6 Disklabel type: gpt
7 Disk identifier: 041DD09F-F16E-1546-9C36-8FDB06DB31EA
8

```

```
9 Device      Start      End  Sectors  Size Type
10 /dev/vda1  262144 19529202 19267059 9.2G Linux filesystem
11 /dev/vda14  2048     8191     6144     3M BIOS boot
12 /dev/vda15  8192    262143  253952   124M EFI System
13
14 Partition table entries are not in disk order.
```

A Trois disques de une partition

B Un disque de trois partitions

C Un disque de une partition

D Un disque de 15 partitions

Solutions des exercices

Solution n°1

[exercice p. 6]

On utilise la commande `free -h` pour avoir une mesure de l'utilisation mémoire, par exemple ici sur un ordinateur avec quelques programmes ouverts.

```
1          total        used         free       shared  buff/cache   available
2 Mem:    7,7Gi         2,1Gi        4,3Gi        397Mi     1,3Gi       4,9Gi
3 Swap:   3,7Gi         0,0Ki        3,7Gi
4
```

On ouvre ensuite un navigateur web Firefox avec 20 onglets.

```
1          total        used         free       shared  buff/cache   available
2 Mem:    7,7Gi         3,4Gi        2,5Gi        559Mi     1,7Gi       3,5Gi
3 Swap:   3,7Gi         0,0Ki        3,7Gi
4
```

On constate une utilisation de 1,3 Gio supplémentaires. De plus le navigateur semble provoquer l'utilisation d'un peu de cache ou de buffers car la valeur est passée de 1,3 à 1,7 Gio (c'est normal pour ce type d'application qui utilise le réseau). Au total, la mémoire réellement disponible est passée de 4,9 à 3,5 Gio, soit 1,4 Gio entièrement monopolisés par le navigateur.

Solution n°2

[exercice p. 10]

On peut utiliser la commande `du` sur le dossier `/var` avec la commande `sort` pour faire un tri.

```
1 $ sudo du -sh /var/* | sort -h
2 0 /var/lock
3 0 /var/run
4 4,0K /var/local
5 4,0K /var/opt
6 8,0K /var/www
7 1,1M /var/mail
8 2,2M /var/spool
9 15M /var/backups
10 26M /var/tmp
11 205M /var/cache
12 804M /var/lib
13 1,5G /var/log
```

On voit ici que le dossier `/var/log` est assez important par rapport aux autres. On peut continuer dans son arborescence.

```
1 $ sudo du -sh /var/log/* | sort -h
2 [...]
3 1,1M /var/log/wtmp.1
4 1,4M /var/log/daemon.log
5 1,6M /var/log/syslog.1
6 1,9M /var/log/kern.log.1
7 2,1M /var/log/messages.1
8 5,7M /var/log/daemon.log.1
9 12M /var/log/boot.log
10 15M /var/log/installer
11 1,4G /var/log/journal
```

Le dossier `journal` occupe à lui seul presque tout l'espace utilisé du dossier. À partir de là, il faudra investiguer pour savoir ce que contient ce dossier et si il est possible de faire de la place.

Solution n°3

[exercice p. 15]

Dans l'interface de htop, on voit que la touche F6 permet d'ouvrir le menu pour trier les processus.

```

1 | [|||||] 13.5% Tasks: 153, 1170 tbr, 1 running
2 | [|||||] 15.7% Load average: 0.51 0.61 0.71
3 | [|||||] 12.3% Uptime: 01:51:34
4 | [|||||] 12.5%
Mem| [|||||] 4.59G/7.66G
Swp| [|||||] 138M/3.73G

PID USER    PRI NI  VIRT  RES  SHR  S  CPU  MEM%  TML= Command
772 root     20  0  229M 10500 6148 S  0  0  0.1  0:00.37 /usr/libexec/polkitd --no-debug
777 root     20  0  218M  4376 1688 S  0  0  0.1  0:00.14 /usr/sbin/rsyslogd -n -iNONE
778 rtkit    21  1  159M  1160  892 S  0  0  0.0  0:00.14 /usr/libexec/rtkit-daemon
779 root     20  0  1188M 4932 3992 S  0  0  0.1  0:00.03 /usr/sbin/smartd -n
780 root     20  0  14000 264 432 S  0  0  0.1  0:00.51 /lib/systemd/systemd-logind
782 root     20  0  192M 11880 9324 S  0  0  0.1  0:01.15 /usr/libexec/udisks2/udisksd
784 root     20  0  14748 7760 6796 S  0  0  0.1  0:00.10 /sbin/wpa_supplicant -u -s -O /run/wpa_supplicant
785 daemon   20  0  532 216 844 S  0  0  0.0  0:00.00 /usr/sbin/atd -f
793 root     20  0  928M 140 2380 S  0  0  0.0  0:01.04 /usr/sbin/sssd
803 avahi    20  0  968 584 8 S  0  0  0.0  0:00.00 avahi-daemon: chroot helper
831 root     20  0  928M 140 2380 S  0  0  0.0  0:00.01 /usr/sbin/sssd
832 root     20  0  928M 140 2380 S  0  0  0.0  0:00.00 /usr/sbin/sssd
833 root     20  0  928M 140 2380 S  0  0  0.0  0:00.77 /usr/sbin/sssd
834 root     20  0  928M 140 2380 S  0  0  0.0  0:00.00 /usr/sbin/sssd
835 root     20  0  928M 140 2380 S  0  0  0.0  0:00.00 /usr/sbin/sssd
836 root     20  0  928M 140 2380 S  0  0  0.0  0:00.02 /usr/sbin/sssd
837 root     20  0  928M 140 2380 S  0  0  0.0  0:00.03 /usr/sbin/sssd
838 root     20  0  928M 140 2380 S  0  0  0.0  0:00.03 /usr/sbin/sssd
839 root     20  0  928M 140 2380 S  0  0  0.0  0:00.03 /usr/sbin/sssd
840 root     20  0  928M 140 2380 S  0  0  0.0  0:00.03 /usr/sbin/sssd
866 root     20  0  229M 10500 6148 S  0  0  0.1  0:00.00 /usr/libexec/polkitd --no-debug
880 root     20  0  392M 11880 9324 S  0  0  0.1  0:00.00 /usr/libexec/udisks2/udisksd
899 root     20  0  218M  4376 1688 S  0  0  0.1  0:00.05 /usr/sbin/rsyslogd -n -iNONE
901 root     20  0  218M  4376 1688 S  0  0  0.1  0:00.07 /usr/sbin/rsyslogd -n -iNONE
1187 rtkit    20  0  159M  1160  892 S  0  0  0.0  0:00.05 /usr/libexec/rtkit-daemon
1188 rtkit    RT  1  159M  1160  892 S  0  0  0.0  0:00.03 /usr/libexec/rtkit-daemon
1189 root     20  0  392M 11880 9324 S  0  0  0.1  0:00.00 /usr/libexec/udisks2/udisksd
1199 root     20  0  229M 10500 6148 S  0  0  0.1  0:00.05 /usr/libexec/polkitd --no-debug
1209 root     20  0  235M  8016 7344 S  0  0  0.1  0:00.08 /usr/sbin/ModemManager
1218 root     20  0  258M 10944 1756 S  0  0  0.2  0:00.00 /usr/sbin/ModemManager --no-daemon
1220 root     20  0  258M 10944 1756 S  0  0  0.2  0:00.26 /usr/sbin/ModemManager --no-daemon
1223 root     20  0  2868M 820 2280 S  0  0  0.1  0:00.03 /usr/sbin/cupsd -l
1230 root     20  0  119M 16648 736 S  0  0  0.2  0:00.12 /usr/bin/python3 /usr/share/unattended-upgrades/unattended-upgrade-shutdown --wait-for-signal
1235 root     20  0  109M  3016  556 S  0  0  0.5  0:04.30 /usr/bin/containerd

Help F2Setup F3Search F4Filter F5Free F6SortBy F7Ice F8Ice F9Kill F10Quit
  
```

Un menu s'ouvre sur le côté et l'on peut choisir notre critère de tri, ici le pourcentage d'utilisation CPU.

```

Sort by  PID
PID      1
USER     353
PRIORITY 386
NICE     736
M_SIZE   756
M_RESIDENT
M_SHARE  762
STATE    763
PERCENT CPU
PERCENT MEM
TIME     778
Command  779
        780
  
```

On peut constater, dans ce cas précis, que le processus le plus gourmand en CPU est Firefox.

```

1 | [|||||] 6.5% Tasks: 153, 1173 tbr, 1 running
2 | [|||||] 8.4% Load average: 0.84 0.66 0.65
3 | [|||||] 7.7% Uptime: 02:08:01
4 | [|||||] 10.9%
Mem| [|||||] 4.77G/7.66G
Swp| [|||||] 136M/3.73G

PID USER    PRI NI  VIRT  RES  SHR  S  CPU  MEM%  TML= Command
12273 kyane    20  0  2591M 292M 158M S 13  0  3.6  4:00.05 /usr/lib/firefox/firefox -contentproc -childID 29 -isForBrowser -prefLen 10430 -prefMapSize 236261 -parentBuildID 202011215
3843 kyane    20  0  3805M 4720 1020 S  7  0  6.1 19:15.11 /usr/lib/firefox/firefox
13651 kyane    20  0  9404  304  152 S  3  2  0.1 0:29.12 htop
3876 kyane    20  0  3805M 474M 188M S  2  6  6.1 5:11.24 /usr/lib/firefox/firefox
12275 kyane    20  0  3501M 3820 1500 S  1  0  3.6 0:22.79 /usr/lib/firefox/firefox -contentproc -childID 30 -isForBrowser -prefLen 10430 -prefMapSize 236261 -parentBuildID 202011215
  
```

Solution n°4

[exercice p. 15]

htop permet d'afficher les processus sous la forme d'un arbre en appuyant sur la touche F5.

Dans cet affichage on peut repérer le parent de chaque processus. Pour plus de lisibilité il est possible de marquer un processus d'une couleur jaune (pour mieux le repérer) en appuyant sur la touche Espace, et de réduire l'arborescence des processus qui ne nous intéressent pas à l'aide de la touche F6.

```

PID PPID CPU MEM VSZ RSS CODE WCHAN STATE S RTD
1 root 20 0 160M 6704 504 S 0 0 0.1 0:04.29 /sbin/init
13731 kyane 20 0 2428 580 504 S 0 0 0.0 0:00.00 /bin/sh -c thunar
12577 kyane 20 0 2428 520 452 S 0 0 0.0 0:00.00 /bin/sh -c ls-sensible-terminal
9356 kyane 20 0 2428 512 440 S 0 0 0.0 0:00.00 /bin/sh -c sceniclient4.2
5990 root 20 0 241M 8048 6764 S 0 0 0.1 0:00.00 /usr/libexec/upowerd
2298 kyane 20 0 2428 572 504 S 0 0 0.0 0:00.00 /bin/sh -c /usr/share/code/code --no-sandbox --unity-launch
5044 root 20 0 8704 900 712 S 0 0 0.0 0:00.04 /usr/sbin/uidm --socket-activation
3842 kyane 20 0 2428 576 504 S 0 0 0.0 0:00.00 /bin/sh -c /usr/lib/firefox/firefox
3843 kyane 20 0 3806M 440M 181M S 7.1 5.6 19:52.22 /usr/lib/firefox/firefox
13976 kyane 20 0 3806M 440M 181M S 0 0 5.6 0:00.01 /usr/lib/firefox/firefox
13898 kyane 20 0 3806M 440M 181M S 0 0 5.6 0:00.00 /usr/lib/firefox/firefox
13888 kyane 20 0 3806M 440M 181M S 0 0 5.6 0:00.01 /usr/lib/firefox/firefox
12315 kyane 20 0 3806M 440M 181M S 0 0 5.6 0:00.17 /usr/lib/firefox/firefox
12385 kyane 20 0 2780M 218M 145M S 0 0 2.8 1:27.00 /usr/lib/firefox/firefox -contentproc -childID 30 -isForBrowser -prefLen 10430 -prefMapSize 236261 -parentBuildID 2
12283 kyane 20 0 3806M 440M 181M S 0 0 5.6 0:00.03 /usr/lib/firefox/firefox
12273 kyane 20 0 2951M 272M 141M S 10.3 3.5 4:39.84 /usr/lib/firefox/firefox -contentproc -childID 29 -isForBrowser -prefLen 10430 -prefMapSize 236261 -parentBuildID 2
12389 kyane 20 0 2951M 272M 141M S 0 0 3.5 0:00.00 /usr/lib/firefox/firefox -contentproc -childID 29 -isForBrowser -prefLen 10430 -prefMapSize 236261 -parentBuildID 2
12388 kyane 20 0 2951M 272M 141M S 0 0 3.5 0:01.28 /usr/lib/firefox/firefox -contentproc -childID 29 -isForBrowser -prefLen 10430 -prefMapSize 236261 -parentBuildID 2
12318 kyane 20 0 2951M 272M 141M S 0 0 3.5 0:00.00 /usr/lib/firefox/firefox -contentproc -childID 29 -isForBrowser -prefLen 10430 -prefMapSize 236261 -parentBuildID 2
12384 kyane 20 0 2951M 272M 141M S 0 0 3.5 0:00.03 /usr/lib/firefox/firefox -contentproc -childID 29 -isForBrowser -prefLen 10430 -prefMapSize 236261 -parentBuildID 2
12393 kyane 20 0 2951M 272M 141M S 0 0 3.5 0:00.01 /usr/lib/firefox/firefox -contentproc -childID 29 -isForBrowser -prefLen 10430 -prefMapSize 236261 -parentBuildID 2
12392 kyane 20 0 2951M 272M 141M S 0 0 3.5 0:00.07 /usr/lib/firefox/firefox -contentproc -childID 29 -isForBrowser -prefLen 10430 -prefMapSize 236261 -parentBuildID 2
12293 kyane 20 0 2951M 272M 141M S 0 0 3.5 0:00.00 /usr/lib/firefox/firefox -contentproc -childID 29 -isForBrowser -prefLen 10430 -prefMapSize 236261 -parentBuildID 2

```

Sur cette vue, on voit que notre processus a pour parent un processus `/usr/lib/firefox/firefox`, qui a lui même pour parent `/bin/sh -c /usr/lib/firefox/firefox`, qui a pour parent le premier processus `/sbin/init`.

Solution n°5

[exercice p. 19]

On peut obtenir la description du service grâce à `systemctl`.

- 1 `$ systemctl status cron`
- 2 ● `cron.service - Regular background program processing daemon`

Il s'agit d'un service, très connu sous Linux, qui permet d'exécuter des tâches à intervalles réguliers sur la machine.

Solution n°6

[exercice p. 19]

Toujours dans le retour de la commande `systemctl status`, on peut voir que le service est active et enabled ce qui signifie qu'il est en cours de fonctionnement et est lancé au démarrage de l'ordinateur.

- 1 `$ systemctl status cron`
- 2 ● `cron.service - Regular background program processing daemon`
- 3 Loaded: loaded (/lib/systemd/system/cron.service; enabled; vendor preset: enabled)
- 4 Active: active (running) since Mon 2020-12-14 09:54:20 CET; 5h 19min ago
- 5

Solution n°7

[exercice p. 23]

On utilise plutôt la commande `journalctl` lorsqu'il s'agit d'un service précis et sur lequel on souhaite appliquer des filtres ?

- 1 `journalctl --since "2 days ago" -u NetworkManager -f`

L'option `--since` permet de filtrer sur les 2 derniers jours, l'option `-u` permet de spécifier un service en particulier et l'option `-f` l'affichage continu des logs.

Solution n°8

[exercice p. 24]

Exercice

À quel endroit le système d'exploitation peut stocker une information en cache, pour y accéder plus rapidement ?

A Dans la mémoire vive (RAM)

B Sur le disque

C Dans un *buffer*

D Sur un service `systemd`

🔍 Pour stocker une information (par exemple un fichier qui est souvent lu) en cache, le système d'exploitation utilise la mémoire vive, qui offre de meilleures performances que le disque.

Exercice

Une partition est :

A Un bout de fichier stocké sur le disque

B Le *buffer* utilisé par le système d'exploitation avant de jouer les sons sur le haut-parleur

(C) Une partie de disque physique utilisée par le système d'exploitation

Exercice

Combien d'octets représente 1 Mo

(A) 1 000 000

B 1 048 576

C 1000

D 1024

🔍 Nous avons :

- 1 Mo = 1 000 000 octets
- 1 Mio = 1 048 576 octets
- 1 ko = 1000 octets
- 1 kio = 1024 octets

Solution n°9

[exercice p. 24]

Exercice

Comment obtenir un aperçu de la mémoire vive utilisée sur la machine

(A) Avec la commande `free`.

B Avec la commande `memory`.

C Avec la commande `usage`.

D Avec la commande `htop`.

Exercice

Comment redémarrer le service nommé `rsyslog` ?

A `rsyslog restart`

B `service rsyslog restart`

C `systemctl rsyslog restart`

D `service restart rsyslog`

E `systemctl restart rsyslog`

Exercice

Comment obtenir une liste des processus en exécution sur la machine ?

A Avec la commande `htop`

B En faisant `ls /proc`

C Avec la commande `ps`

D Avec la commande `proc`

Solution n°10

[exercice p. 25]

Exercice

Dans quel état se trouve le service suivant ?

```

1 $ systemctl status minissdpd.service
2 ● minissdpd.service - keep memory of all UPnP devices that announced themselves
3    Loaded: loaded (/lib/systemd/system/minissdpd.service; enabled; vendor preset:
4    enabled)
5    Active: failed (Result: exit-code) since Mon 2020-12-14 09:54:38 CET; 7h ago
6    Docs: man:minissdpd(1)
7    Process: 1385 ExecCondition=/usr/lib/minissdpd/minissdpd-systemd-wrapper -t
8    ${MiniSSDPd_INTERFACE_ADDRESS} (code=exited, status=0/SUCCESS)
9    Process: 1403 ExecStart=/usr/lib/minissdpd/minissdpd-systemd-wrapper
10   ${MiniSSDPd_INTERFACE_ADDRESS} ${MiniSSDPd_OTHER_OPTIONS} (code=exited,
11   status=1/FAILURE)

```

```

9 déc. 14 09:54:38 europa minissdpd-systemd-wrapper[1403]: Error parsing address/mask
(or interface name) : eth0
10 déc. 14 09:54:38 europa minissdpd-systemd-wrapper[1403]: can't parse "eth0" as a
valid address or interface name
11 déc. 14 09:54:38 europa minissdpd-systemd-wrapper[1403]: Usage: /usr/sbin/minissdpd
[-d] [-6] [-s socket] [-p pidfile] [-t TTL] [-f device] -i <interface> [-i
<interface2>] ...
12 déc. 14 09:54:38 europa minissdpd-systemd-wrapper[1403]: <interface> is either an
IPv4 address with mask such as
13 déc. 14 09:54:38 europa minissdpd-systemd-wrapper[1403]:
192.168.1.42/255.255.255.0, or an interface name such as eth0.
14 déc. 14 09:54:38 europa minissdpd-systemd-wrapper[1403]: By default, socket will
be open as /var/run/minissdpd.sock
15 déc. 14 09:54:38 europa minissdpd-systemd-wrapper[1403]: and pid written to file
/var/run/minissdpd.pid
16 déc. 14 09:54:38 europa systemd[1]: minissdpd.service: Control process exited,
code=exited, status=1/FAILURE
17 déc. 14 09:54:38 europa systemd[1]: minissdpd.service: Failed with result 'exit-
code'.
18 déc. 14 09:54:38 europa systemd[1]: Failed to start keep memory of all UPnP devices
that announced themselves.
19

```

A active

B inactive

C failed

Exercice

Que pouvons nous déduire de la ligne, au format Syslog, suivante ?

```

1 Dec 14 15:54:38 europa dbus-daemon[762]: [system] Successfully activated service
'org.freedesktop.nm_dispatcher'

```

A La machine se nomme dbus - daemon

B Le log date du 14 décembre

C Le service ayant produit le log est system

D Le PID du service est 762

Exercice

Voici le résultat d'une commande. Combien est-ce qu'il y a de disques et de partitions ?

```

1 $ sudo fdisk -l
2 Disk /dev/vda: 9.3 GiB, 10000000000 bytes, 19531250 sectors
3 Units: sectors of 1 * 512 = 512 bytes
4 Sector size (logical/physical): 512 bytes / 512 bytes
5 I/O size (minimum/optimal): 512 bytes / 512 bytes
6 Disklabel type: gpt
7 Disk identifier: 041DD09F-F16E-1546-9C36-8FDB06DB31EA
8
9 Device      Start      End  Sectors  Size Type
10 /dev/vda1  262144  19529202 19267059  9.2G Linux filesystem
11 /dev/vda14  2048      8191     6144     3M BIOS boot
12 /dev/vda15  8192     262143  253952  124M EFI System

```

13

14 Partition table entries are not **in** disk order.

A Trois disques de une partition

B Un disque de trois partitions

C Un disque de une partition

D Un disque de 15 partitions

Crédits des ressources

Schéma d'une architecture Von Neumann p. 37

*Attribution - Partage dans les Mêmes Conditions - Schéma original Kapooht, traduction
Quentin Duchemin*

Ordinateur portable ouvert p. 38

Attribution - Quentin Duchemin

Contenus annexes

1. Architecture Von Neumann

Objectif

- Connaître le modèle conceptuel à l'origine du fonctionnement des ordinateurs modernes.

Mise en situation

Tous les ordinateurs partagent un modèle de conception similaire, hérité de l'**architecture de Von Neumann**. Cette architecture repose en premier lieu sur une unité de calcul et de contrôle qui est capable de manipuler l'information. On peut voir cela comme le bureau où on réalise tous les opérations. L'unité de calcul ne traite qu'une petite quantité d'information à la fois, et seulement des nombres binaires, mais il fait cela très rapidement.

Ensuite, l'ordinateur est doté d'une mémoire vive qui permet de stocker une plus grosse quantité d'informations. C'est là que l'ordinateur dépose les résultats des calculs une fois effectués et où il prend les nouvelles données à traiter. On peut voir cela comme une armoire avec des casiers où l'on range les données.

Enfin, il y a les périphériques qui permettent d'interagir avec le monde.

Architecture de Von Neumann

Cette architecture est un modèle conceptuel décrivant le fonctionnement d'un ordinateur. Elle est utilisée par la quasi-totalité des ordinateurs.

Ce modèle se compose de quatre parties :

- L'unité arithmétique et logique (UAL ou ALU en anglais),
- L'unité de contrôle,
- La mémoire,
- Les entrées/sorties.

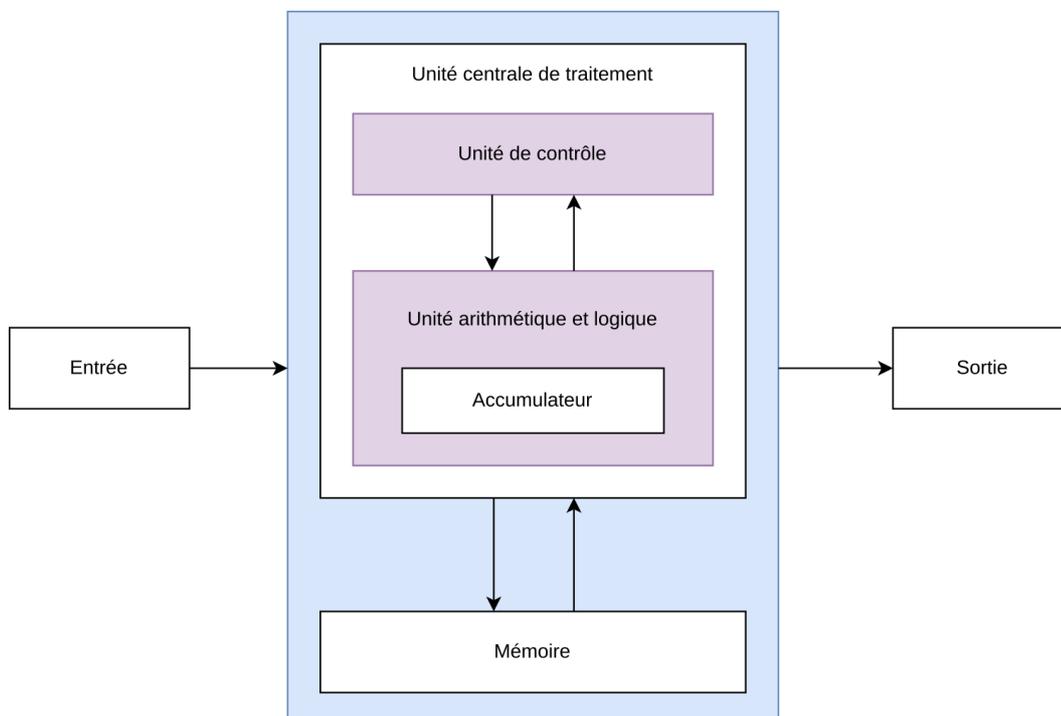


Schéma d'une architecture Von Neumann

L'unité arithmétique et logique

Ce composant est chargé de réaliser toutes les opérations de base : les opérations arithmétiques sur les nombres (addition, multiplication, etc.) et des opérations binaires (OR, AND, etc.).

Masques logiques

⊕ Complément

Les masques logiques diffèrent de l'arithmétique sur les nombres. Ces masques travaillent sur la représentation binaire des données, et effectuent des opérations bit à bit. Une explication plus complète est disponible ici : https://fr.wikibooks.org/wiki/Les_op%C3%A9rations_bit_%C3%A0_bit/Les_masques.

L'unité de contrôle

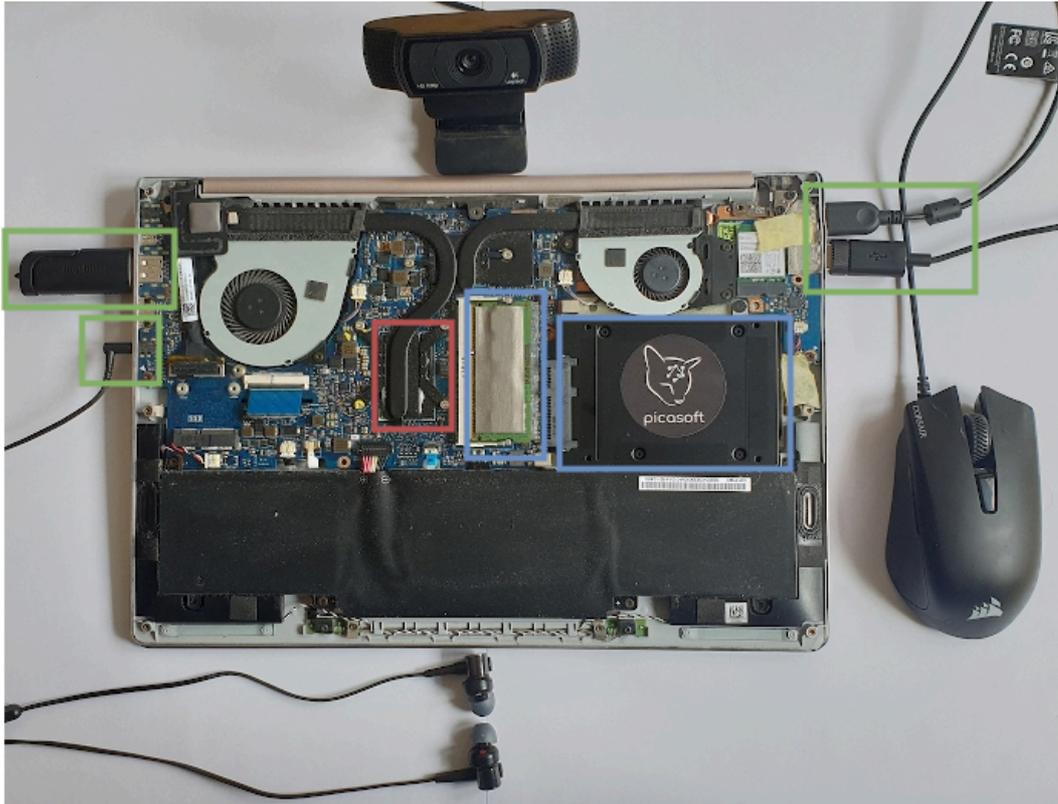
Ce composant est chargé d'ordonner les instructions et d'envoyer tous les calculs à effectuer à l'unité arithmétique et logique.

La mémoire

Cette mémoire stocke à la fois les instructions et les données. D'un côté, elle sera utilisée par l'unité de contrôle pour stocker les séquences d'instructions. De l'autre, elle sera utilisée par l'ALU pour stocker les données d'entrée d'un calcul et le résultat.

Entrées/Sorties

Il s'agit de toutes les interfaces permettant d'interagir avec l'ordinateur. Classiquement un clavier peut être vu comme une entrée et un écran comme une sortie.

 Exemple


Ordinateur portable ouvert

Sur cette image d'un ordinateur portable moderne, on observe bien l'architecture Von Neumann :

- En rouge, l'unité de contrôle et l'ALU, dont l'ensemble forme un processeur,
- En bleu, la mémoire (ici, RAM et SSD),
- En vert, les périphériques (ici, USB et JACK).

L'écran et le clavier intégrés à l'ordinateur portable sont aussi des périphériques.

Processeurs et Architecture de Von Neumann

Les processeurs modernes (aussi appelé CPU, pour *Central Processing Unit*) regroupent l'unité de contrôle et l'ALU. Beaucoup de processeurs ont de multiples **cœurs**, c'est à dire plusieurs ALU.

En quoi est-il différent d'autres modèles ?

 Complément

Une autre architecture très connue (mais moins répandue) existe : l'architecture Harvard. Elle se différencie de l'architecture de Von Neumann notamment par une séparation de la mémoire des instructions exécutée par la machine et de la mémoire de données.

À retenir

- Les ordinateurs modernes utilisent l'architecture dite de Von Neumann.
- L'architecture Von Neumann se compose en quatre parties : ALU, unité de contrôle, mémoire et entrées/sorties.
- Avec les avancées technologiques récentes (processeurs multi-cœurs, etc.), ce modèle perdure tout en évoluant.

2. Mémoire vive et mémoire secondaire

Objectifs

- Connaître la différence entre mémoire vive et mémoire secondaire ;
- Découvrir la notion de *swap*.

Mise en situation

La mémoire secondaire est la mémoire de stockage de l'ordinateur. Elle conserve les données même si l'ordinateur est éteint. C'est par exemple le cas d'un disque dur.

La mémoire vive est la mémoire de travail de l'ordinateur. Pour exécuter un programme l'ordinateur doit copier les données depuis une mémoire de stockage vers la mémoire vive.

Le développeur doit comprendre ce principe d'architecture. En effet, les copies entre les mémoires de stockage et la mémoire vive sont des actions très lentes, chaque fois qu'on en fait, on ralentit le programme. Pour une application web cela peut vite devenir un goulot d'étranglement. À l'inverse la mémoire vive est souvent assez limitée en volume, il n'est donc pas possible de remonter trop d'information à la fois dans la mémoire de travail.

Mémoire vive

Az Définition

La mémoire vive (ou RAM, pour *Random Access Memory*) est caractérisée par sa volatilité : les données s'effacent lors de l'extinction du système.

Elle contient les données des processus qui s'exécutent sur l'ordinateur et est constamment sollicitée par le processeur pour traiter ces données.

Mémoire secondaire

Az Définition

La mémoire secondaire est caractérisée par sa non-volatilité : les données sont conservées même lorsque le système est éteint.

Elle est moins sollicitée pas le CPU et contient les fichiers et données des programmes qui ne sont pas utilisés.

Types de mémoire secondaire

Il existe plusieurs types de mémoire secondaire classées selon la possibilité de les modifier, comme par exemple :

- ROM (pour *Read-Only Memory*) : ce type de mémoire n'est pas effaçable et est utilisé, par exemple, pour stocker les instructions de démarrage d'un ordinateur. Les données stockées sont enregistrées par le constructeur et ne pourront pas être altérées.

- EEPROM (pour *Electrically Erasable Programmable Read Only Memory*) : ce type est le plus répandu - c'est celui des clés USB, des cartes SD, etc. La suppression et la modification des données est possible.

Stockage de la mémoire secondaire

 Exemple

Les **disques durs** sont des exemples de mémoire secondaire.

Deux grandes familles se distinguent : les disques durs utilisant des variantes d'EEPROM - les **SDD** - et les disques durs à disques - appelés **HDD**.

On utilise improprement le nom de disque dur, même si seul ce deuxième type de disque historique correspond à cette appellation.

Dépassement de pile

 Attention

La mémoire vive étant limitée en taille, il peut arriver que des programmes essaient de stocker plus de données que ne le permet la mémoire vive. Lorsque cela arrive, on parle de **dépassement de pile** (ou *stack overflow*).

En tant que développeur, il faut faire attention à l'utilisation de la mémoire vive pour éviter que les programme ne dysfonctionnent. Avant le dépassement de pile, il peut également arriver que la vitesse d'exécution soit fortement ralentie.

La mémoire swap

Pour éviter le problème de dépassement de pile, certains systèmes d'exploitation placent des parties de la mémoire vive inutilisée sur la mémoire secondaire pour économiser la première.

Ce processus est appelé **échange** et on appelle **swap** la mémoire associée à ce procédé.

Si le processeur a besoin de données qui se trouvent dans le swap, un échange a de nouveau lieu. Cet échange est plus lent qu'un accès direct en RAM, mais en contrepartie de la mémoire vive aura pu être mieux utilisée.

Mémoire virtuelle

Az Définition

La mémoire swap est étroitement lié au concept de mémoire virtuelle⁵. La mémoire virtuelle est le regroupement de la mémoire vive et du swap.

Comparaison des temps d'accès mémoire

 Remarque

La mémoire vive a un temps d'accès en mémoire bien plus rapide que la mémoire secondaire.

À titre informatif, voici quelques exemples dans le tableau ci dessous.

⁵ https://fr.wikipedia.org/wiki/M%C3%A9moire_virtuelle

Type d'accès mémoire	Temps en nano-secondes
Mémoire vive	100
Mémoire secondaire (HDD)	10 000 000

ramfs pour accélérer ses programmes

⊕ Complément

ramfs est un système de fichiers temporaire, sur Linux, placé sur la RAM.

Cette pratique est intéressante lorsque le développeur souhaite accéder à un petit nombre de fichiers avec rapidité d'accès très haute. Il faut absolument limiter la taille des fichiers pour éviter le dépassement de pile. Une telle pratique peut accélérer significativement certaines opérations.

À retenir

- La mémoire vive est dédiée à des données peu nombreuses qui nécessitent un accès rapide.
- La mémoire secondaire est dédiée à des données potentiellement volumineuses qui ne nécessitent pas un accès immédiat.
- La mémoire vive est limitée en taille et certains mécanismes optimisent son utilisation et évitent des dépassements de pile.

