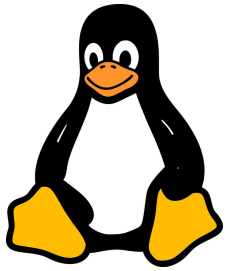
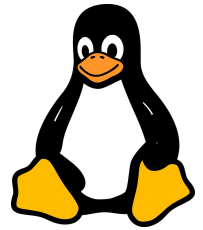
TERMINAL, SHELL, TTY, PTS, PROCESSUS ET JOBS





TeleTYpewriter

Sur Unix, **tty** provient du mot anglais **T**ele**TY**pewriter que l'on peut traduire par **téléimprimeur**, **télétype** ou **téléscripteurs**.

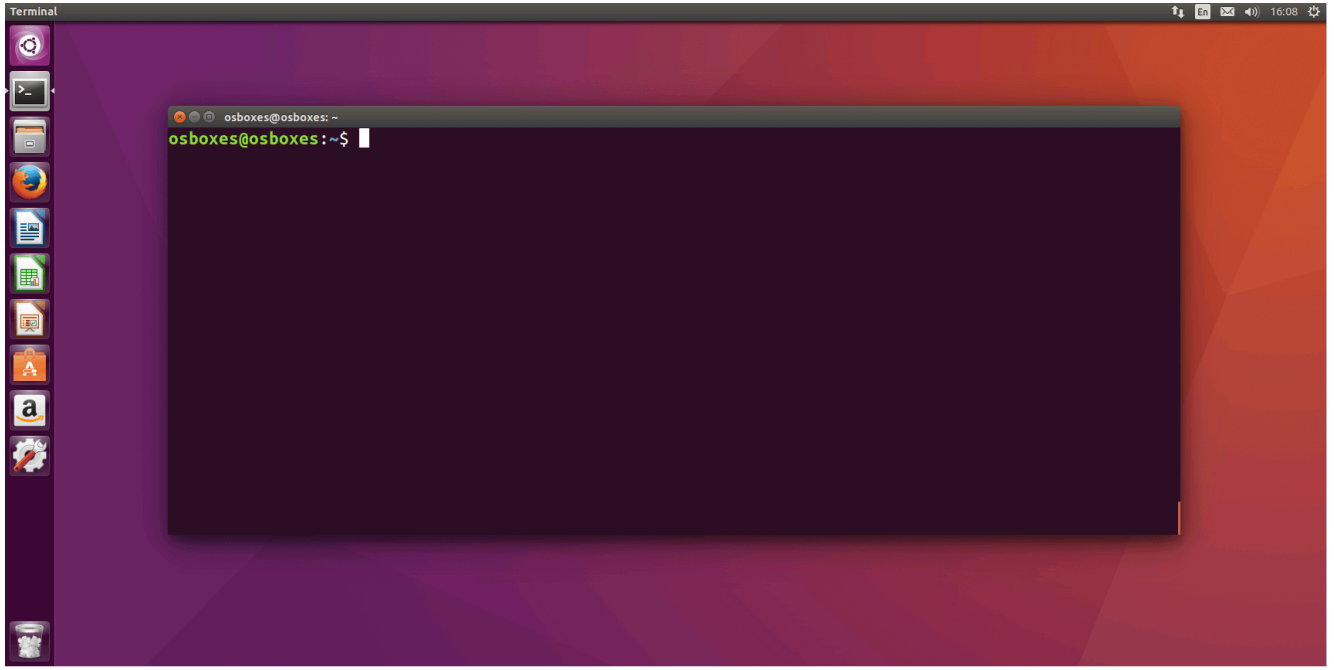
A la fin du XIXe siècle et pendant tout une partie du XXe siècle, les communications à longue distance ont été effectués par **téléscripteur**.
c'est un appareil d'entrée/sortie qui comprend un clavier alphanumérique, un émetteur de signaux électriques commandé par les barres du clavier, un traducteur des signaux reçus et un dispositif d'impression sur page ou sur bande.
Cela permet envoyer et recevoir des messages dactylographiés à travers divers canaux de communication tels qu'une liaison filaire ou onde radio.
Les messages télégraphiées ont utilisé les **téléscripteurs**.

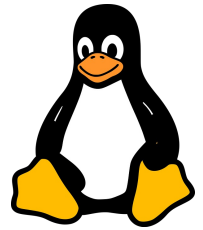
Du TeleTYpewriter au Terminal informatique

Dans les années 70 sont apparus, des téléimprimeurs ressemblant à des machines à écrire qui avait pour sortie une imprimante.

Puis à la fin des 1970, les téléimprimeurs ont été largement remplacés par des **terminaux informatiques** entièrement électroniques qui ont généralement un moniteur d'ordinateur au lieu d'une imprimante.

TERMINAL



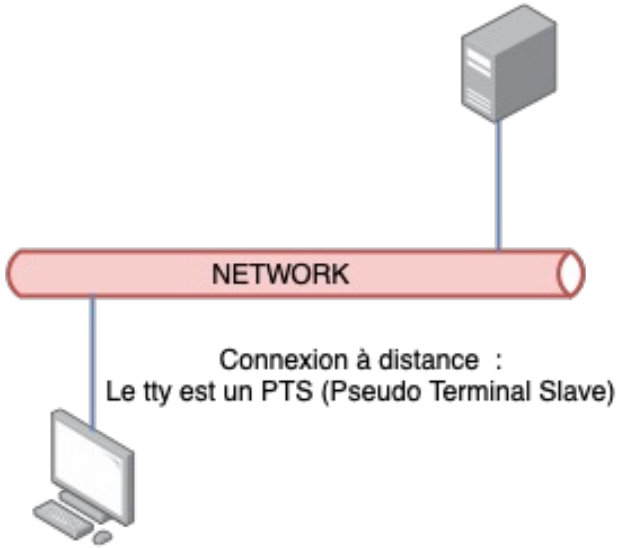


Ainsi, dans de nombreux langages de programmation, le texte de sortie de l'affichage dans une fenêtre de terminal est appelé "impression".

À l'origine, ce n'était pas une métaphore mais un fait.
De ce fait, le terme tty correspond au téléimprimeur qui correspond aux terminaux utilisés au moment de la création **d'Unix** qui est un périphérique d'entrée/sortie.

Linux étant un descendant libre d'Unix, il hérite des terminologies.

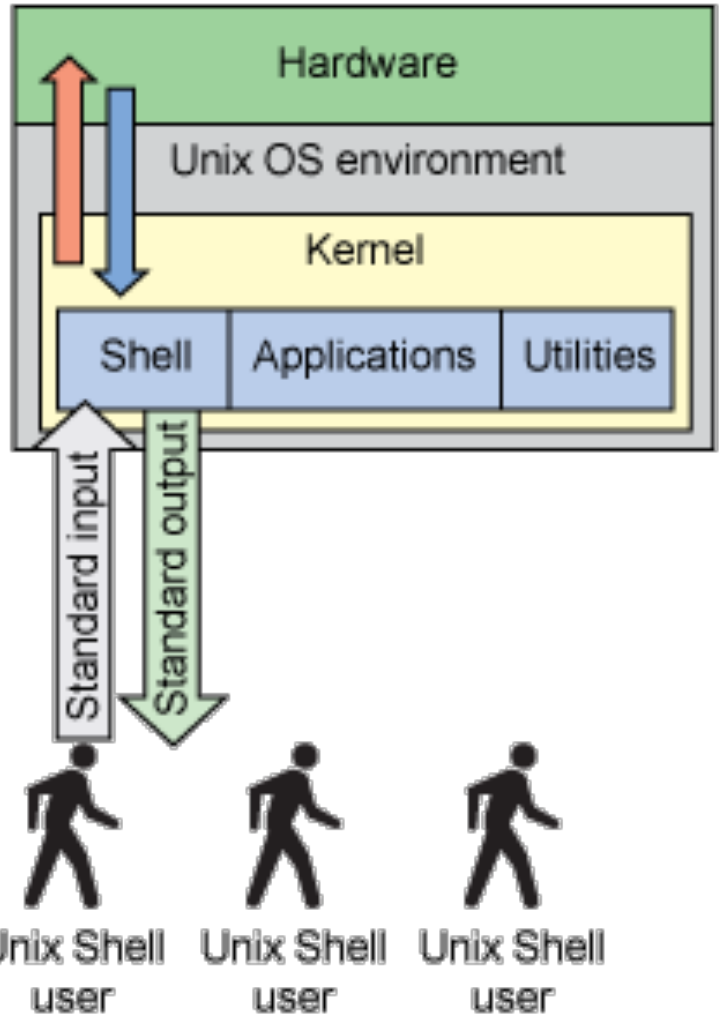
TTY OU PTS



• **Les ports TTY** sont des connexions directes à l'ordinateur tel qu'un clavier / souris ou une connexion série sur le périphérique.

• **Les connexions PTS** sont des connexions SSH ou des connexions Telnet. Toutes ces connexions peuvent se connecter à un shell qui vous permettra de saisir des commandes à l'ordinateur.

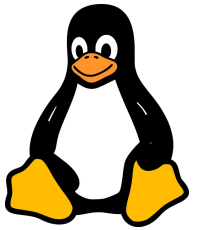
SHELL



Le shell est un programme qui agit comme **interface entre le système et l'utilisateur**. On parle également d'interface système. Il comprend un interpréteur en ligne de commande qui accepte l'entrée de l'utilisateur via le clavier, l'évalue, met en route les programmes en conséquence et renvoie le résultat à l'utilisateur sous forme d'une sortie texte. Chaque shell a par ailleurs son propre langage de programmation permettant d'écrire des scripts shell, par exemple, pour appeler des programmes ou simplifier des tâches administratives.

Chaque shell fonctionne dans un **terminal**.

PROCESSUS

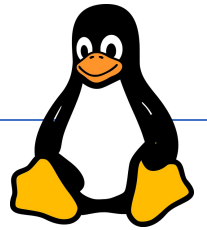


Un processus est une instance d'un programme en cours d'exécution. Chaque fois que vous lancez une application ou une commande dans un système d'exploitation, un processus est créé. Les processus ont plusieurs caractéristiques importantes :

- Identifiant Unique (PID) : Chaque processus est identifié par un numéro d'identification unique appelé PID (Process ID).

- Environnement d'Exécution : Un processus a son propre espace de mémoire, ses variables d'environnement, sa priorité d'exécution, son état (en cours d'exécution, en sommeil, arrêté, etc.), et d'autres attributs.

- Indépendance : Un processus peut s'exécuter indépendamment des autres processus, bien qu'il puisse communiquer ou être affecté par d'autres processus.



Exemple : la commande ls est un processus créé au moment de son exécution.

1. Lancement de la Commande : Quand vous tapez ls et appuyez sur Entrée dans le terminal, le shell interprète cette commande et lance un processus pour l'exécuter.

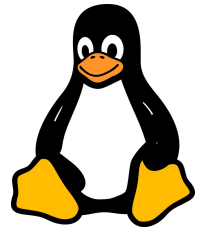
2. Création du Processus : Le système d'exploitation alloue un Process ID (PID) unique au nouveau processus et crée un espace d'exécution séparé pour lui. Ce processus a son propre espace de mémoire, ses propres privilèges de sécurité, et son propre environnement d'exécution.

3. Exécution et Terminaison : Le processus exécute la commande ls, qui liste les fichiers et répertoires dans le répertoire courant, puis se termine. La durée de vie d'un processus de commande comme ls est généralement très courte car il effectue sa tâche rapidement et se termine.

4. Retour au Shell : Une fois que la commande a terminé son exécution, le contrôle revient au shell, qui vous présente à nouveau l'invite de commande pour entrer de nouvelles commandes.

Les processus créés pour des commandes comme ls sont généralement des processus "en premier plan", ce qui signifie qu'ils prennent le contrôle de votre terminal jusqu'à ce qu'ils se terminent. Vous pouvez également lancer des processus en "arrière-plan" dans les shells Unix et Linux, ce qui vous permet de continuer à utiliser le terminal pendant que ces processus s'exécutent.

Commande PS

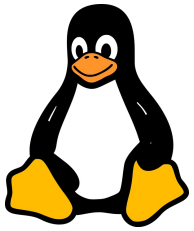


```
root@srvovh:~# ps
    PID TTY          TIME CMD
1149637 pts/0    00:00:00 bash
1149643 pts/0    00:00:00 ps
root@srvovh:~#
```

PS permet d'afficher les processus lancés dans cette session de terminal.

PID : Identifiant de Processus
TTY : terminal utilisé
TIME : durée d'utilisation du cpu pour ce processus
CMD : nom de la commande

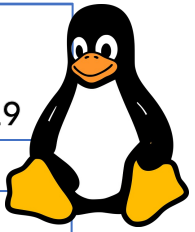
Commande PS aux



```
root@ns332411:~# ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.1 120520  5840 ?        Ss    2021  19:25 /lib/systemd/systemd --system --deserialize 19
root           2  0.0  0.0      0     0 ?        S     2021   0:04 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<    2021   0:00 [rcu_gp]
root           4  0.0  0.0      0     0 ?        I<    2021   0:00 [rcu_par_gp]
root           6  0.0  0.0      0     0 ?        I<    2021   0:00 [kworker/0:0H]
root           8  0.0  0.0      0     0 ?        I<    2021   0:00 [mm_percpu_wq]
root           9  0.0  0.0      0     0 ?        S     2021   0:18 [ksoftirqd/0]
root          10  0.0  0.0      0     0 ?        I     2021  41:08 [rcu_sched]
root          11  0.0  0.0      0     0 ?        I     2021   0:00 [rcu_bh]
root          12  0.0  0.0      0     0 ?        S     2021   0:00 [migration/0]
root          13  0.0  0.0      0     0 ?        S     2021   0:00 [cpuhp/0]
root          14  0.0  0.0      0     0 ?        S     2021   0:00 [cpuhp/1]
root          15  0.0  0.0      0     0 ?        S     2021   0:01 [migration/1]
root          16  0.0  0.0      0     0 ?        S     2021   0:43 [ksoftirqd/1]
root          18  0.0  0.0      0     0 ?        I<    2021   0:00 [kworker/1:0H-kb]
root          19  0.0  0.0      0     0 ?        S     2021   0:00 [cpuhp/2]
root          20  0.0  0.0      0     0 ?        S     2021   0:01 [migration/2]
root          21  0.0  0.0      0     0 ?        S     2021   1:00 [ksoftirqd/2]
root          23  0.0  0.0      0     0 ?        I<    2021   0:00 [kworker/2:0H-kb]
root          24  0.0  0.0      0     0 ?        S     2021   0:00 [cpuhp/3]
root          25  0.0  0.0      0     0 ?        S     2021   0:01 [migration/3]
root          26  0.0  0.0      0     0 ?        S     2021   0:10 [ksoftirqd/3]
```

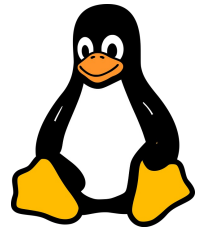
En combinant ces options (aux), la commande ps aux fournit une vue complète et détaillée de tous les processus en cours d'exécution sur le système, indépendamment de leur utilisateur ou terminal associé. Cela inclut les processus en arrière-plan, les processus système, et les processus lancés par d'autres utilisateurs.

```
root@ns332411:~# ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.1 120520  5840 ?        Ss    2021  19:25 /lib/systemd/systemd --system --deserialize 19
```



- **USER : root** - Le processus est exécuté par l'utilisateur root, qui est l'utilisateur administrateur du système.
- **PID : 1** - Le numéro de processus (Process ID) est 1, ce qui identifie ce processus comme le premier processus démarré par le noyau Linux lors du démarrage du système.
- **%CPU : 0.0** - Pourcentage du CPU utilisé par le processus au moment où la commande ps a été exécutée. Une valeur de 0.0 signifie qu'il n'utilisait pas activement le CPU à ce moment-là.
- **%MEM** : **0.1** - Pourcentage de la mémoire physique totale utilisée par le processus.
- **VSZ : 120520** - Taille de la mémoire virtuelle que le processus a réservée, exprimée en kilo-octets.
- **RSS** : **5840** - Taille de l'ensemble résident, qui est la partie de la mémoire que le processus a en mémoire RAM, également exprimée en kilo-octets.
- **TTY : ?** - Le processus n'est pas associé à un terminal spécifique. Le ? indique qu'il s'agit d'un processus de service ou d'un daemon qui n'est pas lié à un TTY.
- **STAT : Ss** - L'état du processus. S indique que le processus est en sommeil (sleeping), attendant un événement pour se réveiller. s signifie qu'il est un processus de session leader.
- **START : 19:25** - L'heure à laquelle le processus a été démarré.
- **TIME : 0:04** - Le temps CPU total que le processus a consommé depuis son démarrage, en minutes:secondes.
- **COMMAND : /lib/systemd/systemd --system --deserialize 19** - La commande complète utilisée pour démarrer le processus. Ici, il s'agit de systemd, le système d'initialisation et gestionnaire de services pour Linux, qui a démarré avec des options spécifiques.

JOB



Dans un shell Unix/Linux, un job est une commande ou un groupe de commandes exécuté dans une seule session du shell. Vous pouvez mettre des jobs en arrière-plan pour continuer à utiliser le shell et ramener ces jobs au premier plan si nécessaire.

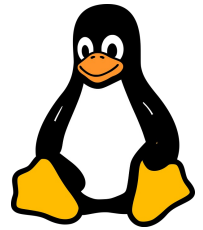
Commandes Clés:

- & : Exécuter une commande en arrière-plan.
- jobs : Afficher tous les jobs actuels.
- fg : Ramener un job en arrière-plan au premier plan.
- bg : Reprendre un job suspendu en arrière-plan.
- Ctrl+Z : Suspendre un job

Exécuter un Job en Arrière-Plan :

Pour exécuter un job en arrière-plan, ajoutez & à la fin de la commande :

```
command &
```



Contrôler les Jobs :

- Pour lister les jobs en cours, utilisez la commande jobs :

```
jobs
```

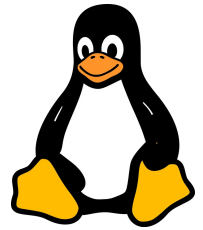
- Pour amener un job en premier plan, utilisez fg suivi de l'identifiant du job :

```
fg %job_number
```

- Pour suspendre un job en cours, utilisez **Ctrl+Z**.
- Pour reprendre un job suspendu en arrière-plan, utilisez **bg** suivi de l'identifiant du job :

```
bg %job_number
```

NICE ET RENICE



Priorité des Processus

Les commandes nice et renice permettent de définir ou de modifier la "niceness" d'un processus, qui est un nombre entre -20 (haute priorité) et 19 (basse priorité). Plus la "niceness" est basse, plus le processus a de priorité pour l'accès au CPU.

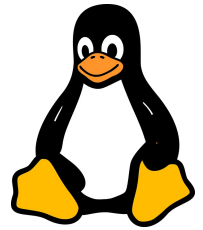
Exécuter un Processus avec une Priorité Définie
Pour lancer un nouveau processus avec une priorité définie :

```
nice –n niceness_value command
```

Modifier la Priorité d'un Processus Existant
Pour modifier la priorité d'un processus existant :

```
renice new_niceness_value –p pid
```

GESTION DES SIGNAUX DE PROCESSUS



Signaux Unix/Linux

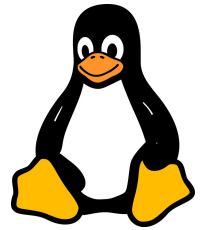
Les signaux sont des messages envoyés aux processus pour leur demander d'effectuer certaines actions ou de réagir à des événements.

- SIGSTOP : Arrête (suspend) un processus.
- SIGCONT : Reprend un processus suspendu.
- SIGTERM : Demande à un processus de se terminer proprement.
- SIGKILL : Force la terminaison d'un processus (ne peut pas être intercepté).

Envoi de Signaux aux Processus
Pour envoyer un signal à un processus : kill -signal pid

```
kill –SIGSTOP pid
kill –SIGCONT pid
kill –SIGTERM pid
kill –SIGKILL pid
```

GESTION DES SIGNAUX DE PROCESSUS



La commande kill -l dans les systèmes Unix et Linux sert à lister tous les signaux que kill peut envoyer aux processus. C'est une commande pratique lorsque vous avez besoin de voir tous les signaux disponibles et leurs noms, que vous pouvez ensuite utiliser avec kill pour envoyer ces signaux à des processus.

```
 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  ...
```