

LE PSEUDO-TERMINAL SUR LINUX

OUMAR MAGOMADOV

15 novembre 2023

TABLE DES MATIÈRES

1	Introduction	1
2	Le fonctionnement d'un terminale	2
2.1	Le pilote de terminal	2
2.1.1	Le mode canonique (coocked mode)	3
2.1.2	Le mode non-canonique (raw mode)	3
2.2	L'émulateur de terminal	4

1 INTRODUCTION

Un *terminal passif*¹, était au début, un périphérique physique connecté à une unité centrale appelé un *mainframe*². Ce terminal était seulement un dispositif d'entrées/-sorties et ne pouvait pas traiter les données seuls sans le mainframe. C'était le mainframe qui traitait les données et renvoyait le résultat au terminal passif pour affichage.



FIGURE 1 – Ordinateur de la NASA en 1962.

Aujourd'hui, avec l'avènement des ordinateurs personnels, ces derniers sont capables de traiter les données de manière autonome sans nécessiter un mainframe. Les terminaux ne sont plus nécessairement physiques aujourd'hui, mais peuvent être émulés de manière virtuelle par le système d'exploitation.

Pour pouvoir interagir avec le système d'exploitation en utilisant un terminale virtuelle, il faut passer par une **console virtuelle**. Le répertoire spécial qui regroupe tout les périphériques utiliser par la machine est `/dev`. Dans ce répertoire, il y'a au total 64 fichiers spéciaux nommé **tty** (pour TeleTYpewriter). Chaque fichier **tty** représente une console virtuelle qui peut être utilisé en tapant les touches `CTRL+ALT+F[1 à 6]`.

Une console virtuelle permet d'établir une communication entre l'utilisateur et le noyau. Quand l'utilisateur entre une commande, il est interprété par le shell. L'utilisation d'une console virtuelle est possible uniquement dans un environnement non graphique. Tandis que dans un environnement graphique, il existe différent **émulateur de terminal** qui permettent de utiliser un terminal graphique, il existe entre autre un émulateur de terminal comme **xterm**.

Le point commun entre ces 2 types de terminaux virtuelles c'est qu'ils utilisent un **pseudo-terminal**. Un pseudo-terminal est un périphérique virtuel qui permet d'établir une communication bidirectionnelle et asynchrone entre 2 ou plusieurs processus. Un processus (esclave) fournis une interface qui est comme un terminal classique et l'autre processus (maître) va contrôler ce terminal.

1. Un dispositif d'affichages et de saisie simples qui ne réalisent pas de traitement important et dépend d'une unité centrale.

2. Un ordinateur qui sert d'unité centrale pour des réseaux de terminaux.

2 LE FONCTIONNEMENT D'UN TERMINALE

Un terminal exécute un **shell** qui permet d'interagir avec le système d'exploitation. Ce dernier interprète les commandes entrées par l'utilisateur et reçoit les réponses du système d'exploitation. Mais avant que les commandes entrées par l'utilisateur n'arrive au shell, elles sont d'abord stocker dans un tampon d'entrée de taille fixe³ définis par le système d'exploitation. En réalité, il existe 2 tampons distincts, une qui stock les entrées et une autre qui stock les sorties. Celui qui gère les tampons dans un terminal est appelé un **pilote de terminal**.

2.1 Le pilote de terminal

Le pilote de terminal va utiliser 2 tampons, qui sont des files, pour gérer les entrées et sorties. Le pilote interprète les caractères stocké dans les tampons de 2 manières différentes. Soit les caractères sont interprétés par lignes, c'est le mode canonique, aussi appelé **cooked mode**. Soit les caractères sont interprétés byte par byte, c'est le mode non-canonique, aussi appelé **raw mode**. Par défaut, le terminal utilise le mode canonique. Pour chaque **read**, une ligne entière du tampon d'entrée est retourné. Le mode non-canonique est utilisé, par exemple, quand on utilise **Vi**.

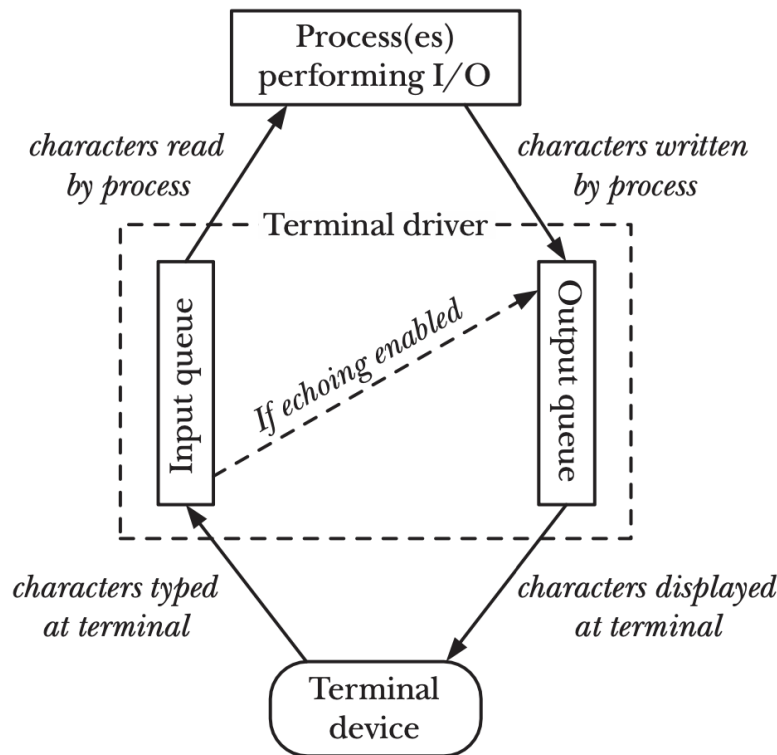


FIGURE 2 – Le fonctionnement d'un pilote de terminal

3. Le tampon d'entrée peut-être dynamique selon le système d'exploitation, dans le cas contraire elle possède une taille fixe.

2.1.1 Le mode canonique (*cooked mode*)

Ce mode permet au pilote de terminal de traiter les caractères par lignes. La fin d'une ligne est délimitée par le caractère **newline**. Quand l'utilisateur presse la touche **ENTER**, un code ASCII est généré, qui est généralement le code 10. De ce fait, le pilote de terminal sait qu'il faut s'arrêter après ce caractère spéciale et prendre en compte toute la ligne jusqu'au délimiteur. Une opération de lecture **read()** sera alors effectuer et la ligne sera retourner au processus. Les caractères spéciaux sont aussi prise en charge sur ce mode.

2.1.2 Le mode non-canonique (*raw mode*)

Ce mode permet au pilote de terminal de traiter une caractère à la fois. Un délimiteur n'est pas utiliser dans ce type de mode, la lecture **read()** est effectuer directement. L'outil d'édition de texte comme **Vi** utilise ce type de mode, les caractères spéciaux ne sont pas prise en charge pour évité toute confusion avec les commandes sur Vi.

De plus, selon le mode choisi, le pilote de terminal peut aussi traiter les caractères spéciaux. Les caractères spéciaux peuvent être de 2 types : **l'édition de l'input** ou **les signaux**.

- Le caractère spécial pour l'édition d'input est par exemple la touche **DEL**, aussi appelé un caractère *ERASE*.
- Le caractère spécial **INTR** correspond généralement à une combinaison de touches **CONTROL+C** qui va envoyer un signal d'interruption.

Exemple pour le mode canonique

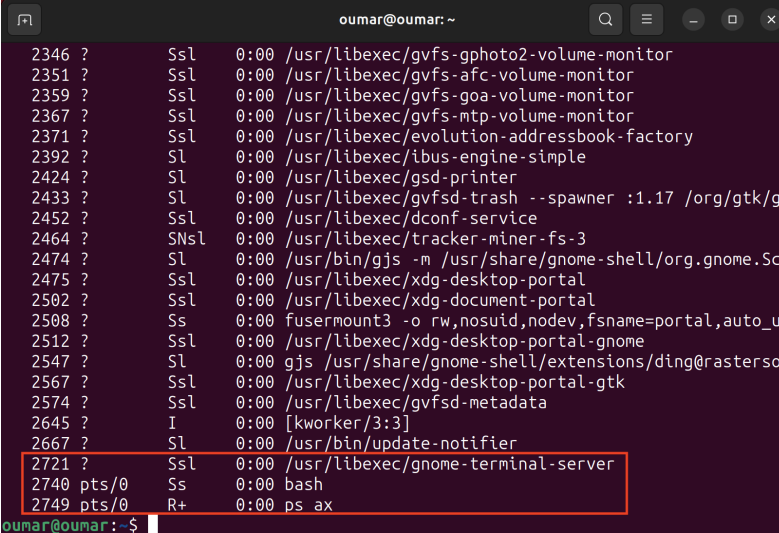
Lorsque un utilisateur tape sur le clavier du texte. Chaque caractère est défini dans un code ASCII. Si on tape **abc**, celui-ci correspond en réalité à **97 98 99**. C'est donc ces 3 codes qui sont stocker dans le tampon d'entrée. Maintenant, pour que le pilote de terminale sache exactement quand s'arrêter et quel ligne retourner, il faut utiliser un caractère spéciale. Comme dit précédemment, le caractère spéciale *newline* correspond au code ASCII **10**. Dans le tampon d'entrée, une fois la touche **ENTER** pressé, le code ASCII **10** est ajouter à la fin dans le tampon d'entrée. Là, le pilote de terminal distingue le délimiteur et effectue une lecture **read()** et renvoie la ligne lue au processus.

Information utile

Il est possible de modifier les caractéristiques d'un périphérique terminal grâce à l'interface **termios** sur Linux. Plusieurs fonctions sont à disposition comme : **tcgetattr** qui permet de stocker les caractéristiques d'un terminal dans une structure à partir de sont descripteur de fichier, et la fonction **tcsetattr** qui permet d'appliquer la nouvelle structure sur le terminal. Les modifications qu'on peut apporter avec ça est entre autres : modifier le mode du terminal, etc...

2.2 L'émulateur de terminal

L'émulateur de terminal permet de créer une interface textuelle en qui imite le fonctionnement d'un terminal physique. Tout comme les **console**s **virtuelles**, l'émulateur de terminal utilise les pseudos-terminaux pour créer un canal de communication entre 2 ou plusieurs processus. Lorsqu'un émulateur de terminal est lancé, il initie la création d'une paire de pseudo-terminaux (pty) comprenant un côté maître et un côté esclave. Le côté maître est associé à l'émulateur de terminal lui-même, et le côté esclave est associé à un processus qui tourne le **bash**.



```

oumar@oumar: ~
2346 ?      Ssl  0:00 /usr/libexec/gvfs-gphoto2-volume-monitor
2351 ?      Ssl  0:00 /usr/libexec/gvfs-afc-volume-monitor
2359 ?      Ssl  0:00 /usr/libexec/gvfs-goa-volume-monitor
2367 ?      Ssl  0:00 /usr/libexec/gvfs-mtp-volume-monitor
2371 ?      Ssl  0:00 /usr/libexec/evolution-addressbook-factory
2392 ?      Sl   0:00 /usr/libexec/ibus-engine-simple
2424 ?      Sl   0:00 /usr/libexec/gsd-printer
2433 ?      Sl   0:00 /usr/libexec/gvfsd-trash --spawner :1.17 /org/gtk/g
2452 ?      Ssl  0:00 /usr/libexec/dconf-service
2464 ?      Sssl 0:00 /usr/libexec/tracker-miner-fs-3
2474 ?      Sl   0:00 /usr/bin/gjs -m /usr/share/gnome-shell/org.gnome.Sc
2475 ?      Ssl  0:00 /usr/libexec/xdg-desktop-portal
2502 ?      Ssl  0:00 /usr/libexec/xdg-document-portal
2508 ?      Ss   0:00 fusermount3 -o rw,nosuid,nodev,fsname=portal,auto_u
2512 ?      Ssl  0:00 /usr/libexec/xdg-desktop-portal-gnome
2547 ?      Sl   0:00 gjs /usr/share/gnome-shell/extensions/ding@rasterso
2567 ?      Ssl  0:00 /usr/libexec/xdg-desktop-portal-gtk
2574 ?      Ssl  0:00 /usr/libexec/gvfsd-metadata
2645 ?      I    0:00 [kworker/3:3]
2667 ?      Sl   0:00 /usr/bin/update-notifier
2721 ?      Ssl  0:00 /usr/libexec/gnome-terminal-server
2740 pts/0  Ss   0:00 bash
2749 pts/0  R+   0:00 ps ax
oumar@oumar: ~$

```

FIGURE 3 – L'affichage de la commande **ps ax**

Les données entrées passent d'abord par l'émulateur de terminal qui joue le rôle de maître. Une fois les données reçues, elles sont envoyées au côté esclave, qui est le processus qui lance **bash**. Une fois que **bash** a interprété la commande, le retour est renvoyé au côté maître depuis le côté esclave. Il y a donc bien une communication bidirectionnelle entre le maître et l'esclave.

REFERENCES

- [1] Michael Kerrisk. *The Linux Programming Interface*. No Starch Press, 2010.
- [2] Système d'exploitation gnu. <https://www.gnu.org/>.
- [3] Serge Lakovlev. Terminal on linux. <http://www.iakovlev.org/>.