Terminaux et consoles, explications

- Objet : Définitions et explications des termes « terminal » et « console ».
- Niveau requis :
  débutant
- Pour aller vers :
  avisé
- Commentaires : Utilisation de l'interpréteur de commande bash dans la section tuto.
- Débutant, à savoir : Utiliser GNU/Linux en ligne de commande, tout commence là !.
- Suivi :
  en-chantier
  - Création par agp91 21/04/2023
  - Testé par <...> le <...> Fix Me!
- Commentaires sur le forum : Lien vers le forum concernant ce tuto [1)]

Nota :

Contributeurs, les Fix Me! sont là pour vous aider, supprimez-les une fois le problème corrigé ou le champ rempli !

Introduction

Pour différencier les différents types d'**interfaces en mode texte**, nommées aussi **interfaces de ligne de commande** ou **CLI** (**C**ommande **L**ine **I**nterface) qui sont mises à notre disposition, nous utilisons communément deux termes :

- **Console** pour désigner, les interfaces que nous obtenons par la combinaison des touche `Ctrl+Alt+F1` à `F6` (sur les systèmes Debian GNU/Linux modernes).
- **Terminal**, pour désigner dans un environnent graphique, la fenêtre d' uneinterface en mode texte.

Cela est simple et précis.
Mais quand nous souhaitons aller plus loin, c'est réducteur et peut porter à confusion.

Par exemple, lorsque que nous lisons la page du manuel du terminal fenêtre **xterm**, nous pouvons y lire :

Extrait de la page du manuel d'xterm

```
XTERM(1)            X Window System                    XTERM(1)

NAME
       xterm - terminal emulator for X
```

Ici, même si **terminal** est qualifié d'**émulateur pour X**, nous pouvons encore comprendre.

Mais lorsque nous ouvrons la page du manuel de la commande **agetty**.

Qui est le programme, qui nous permet d'obtenir les six **consoles**, disponibles par les combinaisons des touches, `Crtl+Alt+F…`.
Nous pouvons y lire :

Extrait de la page du manuel d'agetty

```
AGETTY(8)       Administration Système            AGETTY(8)

...

DESCRIPTION
       agetty  ouvre  un  port  de  terminal...
```

Nous comprenons donc là, qu'un terminal est aussi autre chose, qu'une interface fenêtre dans un environnement graphique.
Ce qui étonne, c'est que, ne soit pas utilisé le terme console.

Alors, nous décidons d'élargir nos connaissances. Et après une longue immersion dans l'Internet.
Notre vocabulaire se trouve grandement amélioré :

- Console, console physique, console Linux, console système, console virtuelle, console texte, console graphique,
- Terminal, terminal physique, terminal Linux, émulateur de terminal, terminal virtuel, terminal texte, terminal intelligent (par-ce qu'il y a des terminaux sots ? 😀), terminal muet, pseudo-terminal, terminal graphique, terminal X, terminal fenêtre, multiplexeur de terminal, terminal série…
- TTY, PTY

Même si certaines de ces dénominations sont compréhensibles par leurs qualificatifs. Par exemple : Consoles/terminaux texte (qui n'utilisent que du texte), consoles/terminaux graphique (qui peuvent afficher des images).

… Il se peut, que nous ne comprenons plus rien, tant les définitions trouvées sont disparates et même contradictoires.
Nous commençons à regretter le temps, où il nous semblait, que nous avions tout compris avec seulement deux définitions.

> Pouvons-nous utiliser indifféremment les mots **terminal** et **console**, comme nous le souhaitons ?
>
> > Non, évidement que non !

Il est proposé dans cette page de wiki, de revenir aux définitions (parfois historiques), qui permettent de désigner chacun des éléments de nos système.
Et ainsi apporter éclaircissement dans la confusion.

Plus bas, il est parfois fait référence à la documentation des sources du noyau, écrite par Linus Torvalds.
Étant sous systèmes GNU/Linux, avec un noyau Linux, de et dirigé par Linus Torvalds,
C'est donc lui, ici, qui aura le dernier mot.

Allons-y, retour aux sources.

Définitions

Les terminaux (machines)

Terminal

Un terminal est un appareil qui dispose simplement d'une entrée et d'une sortie. Connecté directement à un ordinateur (central), ou par réseau, il permet de communiquer avec ce dernier. De lui envoyer des informations (l'entrée) et d'en communiquer le retour (la sortie).

Les **terminaux** sont des points finaux (terminus) d'un réseau. Après eux, il y a nous.

Actuellement, les **terminaux** disposent presque tous d'un clavier comme entrée (et/ou d'un pointeur, souvent une souris). Et pour sortie un écran.
Aux débuts de l'informatique, la sortie était une imprimante, avec un rouleau de papier.

La saisie depuis l'entrée d'un **terminal** (par exemple, un clavier), est envoyée à l'ordinateur, qui traite l'information et retourne le résultat sur la sortie du **terminal**.

Sur un **terminal**, aucun programme n'est exécuté, ils sont exécutés sur l'ordinateur. Le terminal est utilisé uniquement, comme point d'entrée/sortie pour communiquer avec le programme distant.

Terminal d'impression

Un **terminal d'impression**, parfois nommé **terminal papier**, est un terminal doté d'un clavier en entrée et d'une imprimante en sortie. Les premier étaient nommés téléimprimeurs ou téléscripteurs.

En plus du clavier (ou pas), certains disposaient (aussi) en entrée, d'un lecteur de carte perforée. Et d'une perforeuse en sortie. Permettant ainsi la sauvegarde et la restauration de l'ordinateur central.

Terminal vidéo

Un **terminal vidéo** est un terminal dont la sortie est un écran vidéo. Ils remplacèrent avantageusement les terminaux d'impressions.

Terminal mode caractère

Un **terminal mode caractère** ou **mode octet** ou **mode brut (raw mode)**, communique avec l'ordinateur caractère/octet par caractère/octet.

Un jeux de caractère prédéfinit (par exemple le code ASCII) est utilisé. Ces jeux de caractère disposent de quelques caractères de contrôle tel que la tabulation, l'effacement arrière d'un caractère, le saut de ligne, etc.

Par exemple, avec le jeux de caractère ASCII, les caractères sont codés de 0 à 127. Les 32 premiers (de 0 à 31) et le dernier (127), sont des caractères de controle[2].

Un caractère saisi au clavier est envoyé à l'ordinateur, le quel, soit :

- Le retourne au terminal :
  - S'il s'agit d'un caractère imprimable, il est affiché à l'écran.
  - S'il s'agit d'un caractère de contrôle (non imprimable), le terminal applique son interprétation à l'écran.
- Si le caractère est un caractère de contrôle qui met fin à l'écriture d'une commande. Il retourne au terminal, caractère par caractère, le résultat de la commande.

Pratiquement chaque terminal dispose de son(/ses) propre(s) jeu(x) de caractère, incluant plus ou moins des caractères de contrôle.

Avec l'évolution du matériel, le nombre de caractère de contrôle, a pu être augmenté, en utilisant des séquences de caractère, nommées séquences d'échappement.
Ces séquences débutent par un caractère d'échappement. Comme par exemple le caractère ANSI ESC (ESCape).
Les séquences d'échappement ANSI[3] en sont un aboutissement standardisées.

Terminal mode ligne

Avec encore plus de matériel embarqué, en particulier de la mémoire et parfois un processeur. Les **terminaux mode ligne** sont apparus.

En **mode ligne**, nommé aussi **mode canonical**, la communication avec l'ordinateur se réalise ligne de caractère par ligne de caractère.

Chaque caractère d'une ligne est conservé dans un tampon, tout en étant affiché à l'écran pour l'édition.
La ligne n'est envoyée à l'ordinateur, qu'après l'usage du caractère de fin de ligne ou fin de bloc.

> (stackexchange)(en) Mode ligne de l'IBM 1050 (Traduction) : L'IBMs 1050[4] , un terminal d'impression, introduit en mars 1963, a divergé (le premier?) des TTY [5] avec les E/S à caractère unique, non synchronisés, en introduisant le **mode ligne**. Le Control Unit 1051[6] gérait un tampon de ligne. Alors que chaque frappe était directement affichée, une ligne n'était envoyée à l'unité centrale que lorsque EOB (End Of Block) était pressé. Un saut de ligne, impliquait un EOB. Après l'envoi, le clavier était verrouillé, jusqu'à ce qu'il soit libéré par l'unité centrale.

Terminal mode bloc

Les **terminaux mode bloc** ont encore étendus les fonctionnalités d'édition. Libérant avantageusement l'ordinateur et de la bande passante du réseau. Ainsi plus de **terminaux** pouvaient être connecté à l'ordinateur.

Le tampon contient un bloc (une matrice) de caractère, souvent la page entière de l'écran. Permettant ainsi, par exemple d'envoyer à l'ordinateur un formulaire entier. L'ordinateur retourne alors le rafraîchissement de l'écran (la page) à afficher.

Le **mode bloc** peut être utilisé par les terminaux graphique, pour afficher des graphismes et des images.

Un **mode ligne** n'est pas un **mode bloc**.

- Le **mode bloc** utilise une matrice de caractère ou de pixel définit. Par exemple 80×24 caractères.
- Dans un **mode ligne**, la longueur de la ligne est déterminée par le nombre de caractère précédent le caractère de fin de ligne. Un nombre maximal de caractère est imposé.

Certains terminaux utilisent les deux modes (**mode bloc** et **mode ligne**) :

> (retrocomputing.stackexchange)(en) Questions (Traduction) : Le tout premier système de terminal (graphique) d'IBM, le 2250[7)] de 1964, utilisait également un mode ligne/bloc pour la saisie de texte. Alors que le terminal était directement connecté au canal d'E/S de l'ordinateur, un tampon de données local était utilisé pour l'édition. Le texte était affiché et pouvait être modifié en mode ligne. Lorsqu'un « message était composé », comme on l'appelait à l'époque, une interruption d'E/S sur l'ordinateur était émise, pour transmettre l'intégralité du bloc de données.

Terminal texte

Un **terminal texte**, qu'il soit en mode caractère, en mode ligne ou en mode bloc, n'affiche que du texte.

Pourtant il est possible de réaliser des graphismes rudimentaires en utilisant les caractères, pour former des fenêtres, des boites de dialogues, etc..

Certains **terminaux texte** disposent de jeu de caractères semi-graphique[8)]. Dont les matrices, ne sont pas des caractères mais des graphiques. Avec des séquences d'échappement, il est même possible de réaliser des animations[9)].

L'art ASCII[10)] permet de dessiner dans un **terminal texte**.

```
 _________________
< Debian Facile >
 ----------------_________________
   \   .--.    < mouoai facile >
    \ |o_o |    ---------------
      |:_/ |      \   ^__^
     //   \ \      \  (oo)\_______
    (|     | )        (__)\       )\/\
   /'\_   _/`\           ||----w |
   \___)=(___/           ||     ||
```

Le programme **cowsay**[11] à été utilisé pour ce dessin.

Terminal graphique

Un **terminal graphique** est un terminal mode bloc, qui permet d'afficher du texte, des graphiques et des images.

Terminal passif ou intelligent

Terminal passif ou **terminal muet** est la traduction sobre de **dumb terminal**. D'autres traductions peuvent apparaître : stupide, sot, idiot, etc..

Terminal intelligent est la traduction de **intelligent terminal** ou **smart terminal**

La signification de ces termes ont évolué dans le temps. Au fur et à mesure que les terminaux embarquaient plus de matériel. Offrant plus de fonctionnalité.

Au départ, les **terminaux passifs** étaient les **terminaux mode caractère**, qui n'offrent aucune fonction d'édition et qui sollicitent l'ordinateur, dés lors qu'une touche est appuyée.

A contrario, les **terminaux intelligent** déchargeaient l'ordinateur des taches de saisie et/ou d'affichage (séquences d'échappement, mode ligne, mode bloc, affichage de graphisme).

De nos jours, est aussi nommé **terminal intelligent**, un client léger. Ou un client lourd, qui participe aux traitements des données (par exemple lorsque qu'il retourne une somme qu'il a lui même calculé).

D'un point de vu marketing, une distinction est faite entre les **terminaux smart**, qui regroupent ceux qui soulagent l'ordinateur de l'édition et de l'affichage. Et les **terminal intelligent**, qui regroupent les clients lourds.

Voir aussi :

Le **terminal intelligent** par l'historique IBM : (stackexchange)(en) Questions : When did smart terminals arrive ?

- **Terminal passif** (**dumb terminal**) : (fr.wikipedia) Terminal passif - (wikipedia)(en) Computer terminal : Dumb terminal - (oxfordreference)(en) Dumb terminal - (webopedia)(en) Dumb terminal
- **Terminal intelligent** (**intelligent terminal**) : |(wikipedia)(en) Computer terminal : Intelligent terminal - (oxfordreference)(en) Intelligent terminal (1) - (oxfordreference)(en) Intelligent terminal (2) - (webopedia)(en) Intelligent terminal
- **Terminal smart** (**smart terminal**) : (wikipedia)(en) Smart terminal - (webopedia)(en) Smart terminal

Si tous s'accordent à peut prés pour définir un **terminal passif**.

Il est difficile de déterminer qui est intelligent ou qui ne l'est pas.
Cela dépendent de la référence, sur la quelle nous nous appuyons.
Ainsi, un **terminal intelligent** peut-être :

- Un terminal mode caractère qui disposent suffisamment de matériel pour interpréter les séquences d'échappement.
- Un terminal mode ligne,
- Un terminal mode bloc,
- Un terminal graphique, qui affiche un rendu que ne dispose pas l'ordinateur hôte.
- Un terminal client léger, qui permet de se connecter, aux services d'accès à distance d'un ordinateur hôte.
- Un terminal client lourd, qui dispose d'une capacité de calcul supplémentaire (en plus du fonctionnement du client). Pour retourner à l'ordinateur hôte, le résultat d'un traitement. Déchargeant ainsi, l'ordinateur hôte de certains calculs.

Les consoles (machines)

Une **console**, nommée aussi **console système** ou **pupitre de commande**, est une machine, un terminal, qui a un accès particulier (privilégier) à l'ordinateur central.

Elle permet la communication avec cet ordinateur, depuis son démarrage.
Les messages du système, y sont affichés.

C'est à travers elle, que peut-être administrer l'ordinateur.
Par exemples : Installer le système d'exploitation ; Configurer le noyau ; Passer des options durant le démarrage ; Installer et configurer les services ; Etc.

Ainsi, une **console (système)**, se distingue d'un simple **terminal**, qui lui, ne fait que se connecter à l'un des services démarré sur l'ordinateur.

Les émulateurs de terminaux

Un **émulateur de terminal** est un programme qui émule une machine terminal.

Nos ordinateurs (PC) ne sont pas des terminaux, mais avec les ressources qu'ils disposent, ils peuvent à travers un programme, émuler le comportement d'un terminal.

Un ordinateur hôte, ne fait pas la différence entre une machine terminal et un **terminal émulé**.

TTY

TTY est l'acronyme **T**ele**TY**pe[12], un téléimprimeur commercialisé en 1953 par la sociètée Teletype Corporation[13], filiale de Western Electric Compagny[14], rachetée par AT&T[15] en 1930.

Le nom de marque “Télétype” à été breveté par Les Établissements Édouard-Belin[16] en 1925.

Plus simplement, d'un point de vue système, **TTY** signifie **terminal** (puisqu'un **Teletype** est un **terminal**).

Explications

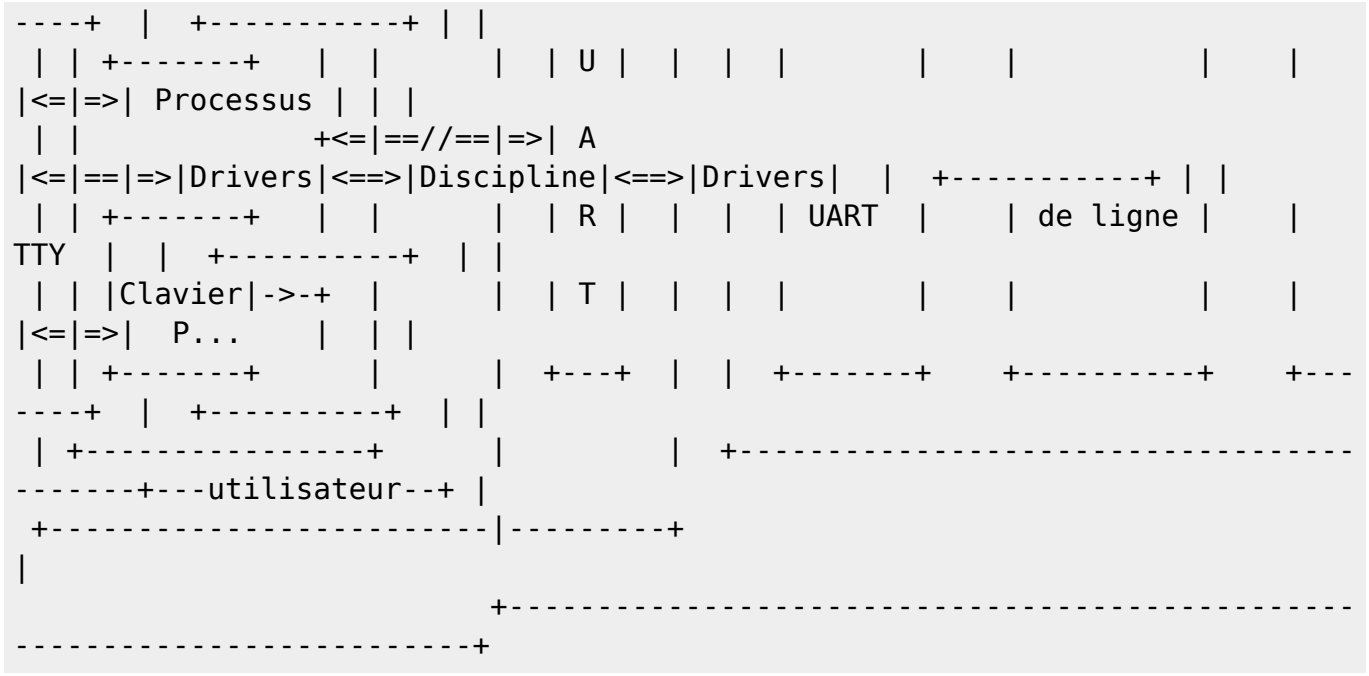
Des définitions données ci-dessus, nous pouvons retenir :

- Un **terminal** est un appareil disposant d'une entrée (un clavier) et d'une sortie (une imprimante, un écran, etc).
- Un **terminal vidéo** est un terminal disposant d'un écran vidéo.
- Une **console système** ou une **console** est un terminal connecté à un ordinateur, qui affiche les messages du système de cet ordinateur. Et qui permet de le configurer depuis le démarrage.
- Un **terminal texte** est un terminal qui n'affiche que du texte.
- Un **terminal graphique** est un terminal qui affiche du texte et des images.
- Un **émulateur de terminal** ou un **terminal émulé** est un programme qui émule un terminal.
- **TTY** signifie **terminal**.

Au début de l'informatique, pour communiquer avec les ordinateurs, les téléimprimeurs ont été utilisés pour fournir une entrée et une sortie. Les téléimprimeur ont évolué en terminal vidéo.

Schématiquement, nous avions ceci :

```
 +------------Matériel---------------+
 |                        +---------|-------------------------Ordinateur--
-------------------------+
 | +----Terminal----+      |         |  +-------------------------Logiciel-
-----------------------+ |
 | | +-------+      |      |         |  +------------------Espace-Noyau-----
-------+-----Espace-----+ |
 | | |Sortie |<--+  |      |  +---+  |  |  +-------+    +----------+    +---
```



Un téléscripteur (un terminal), qui ici, pourrait être une console (système), est connecté par un câble (une paire de fils) à l'ordinateur.

- Lorsque le terminal envoie le signal d'un caractère pressé au clavier. Il est reçu par un composant électronique de l'ordinateur, nommé **UART** (Universal Asynchronous Receiver Transmitter)[17] et ses drivers. Ce qui transforme le signal de la transmission série[18] en une transmission parrallèle[19] et en octet.
- L'octet est transmis à la **discipline de ligne**, qui dispose d'un tampon d'édition et de quelques fonctions rudimentaires d'édition.
  - Par exemple :
    - Un caractère tapé, est transmis à la **discipline de ligne**. Il est ajouté au tampon et renvoyé vers la sortie du **terminal**, pour affichage/impression.
    - … Mais nous souhaitons le supprimer. Alors depuis le **terminal**, nous envoyons le caractère de contrôle d'effacement arrière.
    - Arrivé dans la **discipline de ligne**, le tampon et la sortie du terminal sont mis à jour (effacement du dernier caractère).
    - → Ainsi, ni le caractère, ni l'ordre de l'effacer, ne sont envoyés aux **drivers TTY**.
- La **discipline de ligne** envoie, aux **drivers TTY**, le contenu de son tampon, que lorsque le caractère de fin de ligne est reçu.
- En ce sens, la **discipline de ligne** travail en mode ligne (ce comportement peut-être évidement modifié pour un autre mode).
- Puis, les **drivers TTY** se chargent de transmettre, la séquence reçue de la **discipline de ligne**, au processus approprié.

Depuis,
Avec l'évolution, les ordinateurs sont dans nos maisons. Pour que nous puissions communiquer avec eux, ils émulent, avec les ressources qu'ils disposent (mémoire et microprocesseurs), un terminal. L'entrée et la sortie de ce terminal sont, le clavier et l'écran (seul matériel restant des téléscripteur) et la souris. Ils sont directement connecter à l'ordinateur.

Poursuivons…

La console Linux

Dans un système GNU/Linux, la **console Linux** émule un terminal au sein du noyau.

Elle permet d'utiliser le clavier et l'écran comme un terminal (ou plutôt comme une **console**).

C'est l'interface que nous disposons lors du démarrage (boot) de l'ordinateur ou lorsque nous démarrons en mode mono-utilisateur (ou mode dépannage).

Linux Torvald, dans les sources du noyau à écrit :

> (git.kernel)(en) Torvalds/Linux/drivers/tty/Kconfig (traduction) : La console système est le périphérique qui reçoit tous les messages du noyau et des avertissements et qui permet les connexions en mode mono-utilisateur.

Ainsi la **console Linux** est **la console** de l'ordinateur. Elle est, ce qu'est une console système pour l'ordinateur central.

La console Linux **n'existe que dans l'espace noyau et le matériel.**

L'espace noyau fait référence aux anneaux théoriques de sécurité d'un système informatique[20)].

Il y a 4 niveaux de sécurité (numérotés de 0 à 3) :

- Le **niveau 0** : Le noyau lui-même.
- Le **niveau 1** : Les drivers qui ordonnent le matériel.
- Le **niveau 2** : Les drivers qui implémentent l'accès et l'autorité aux ressources.
- Le **niveau 3** : L'espace utilisateur. Où nous pouvons utiliser les ressources pour exécuter nos programmes.

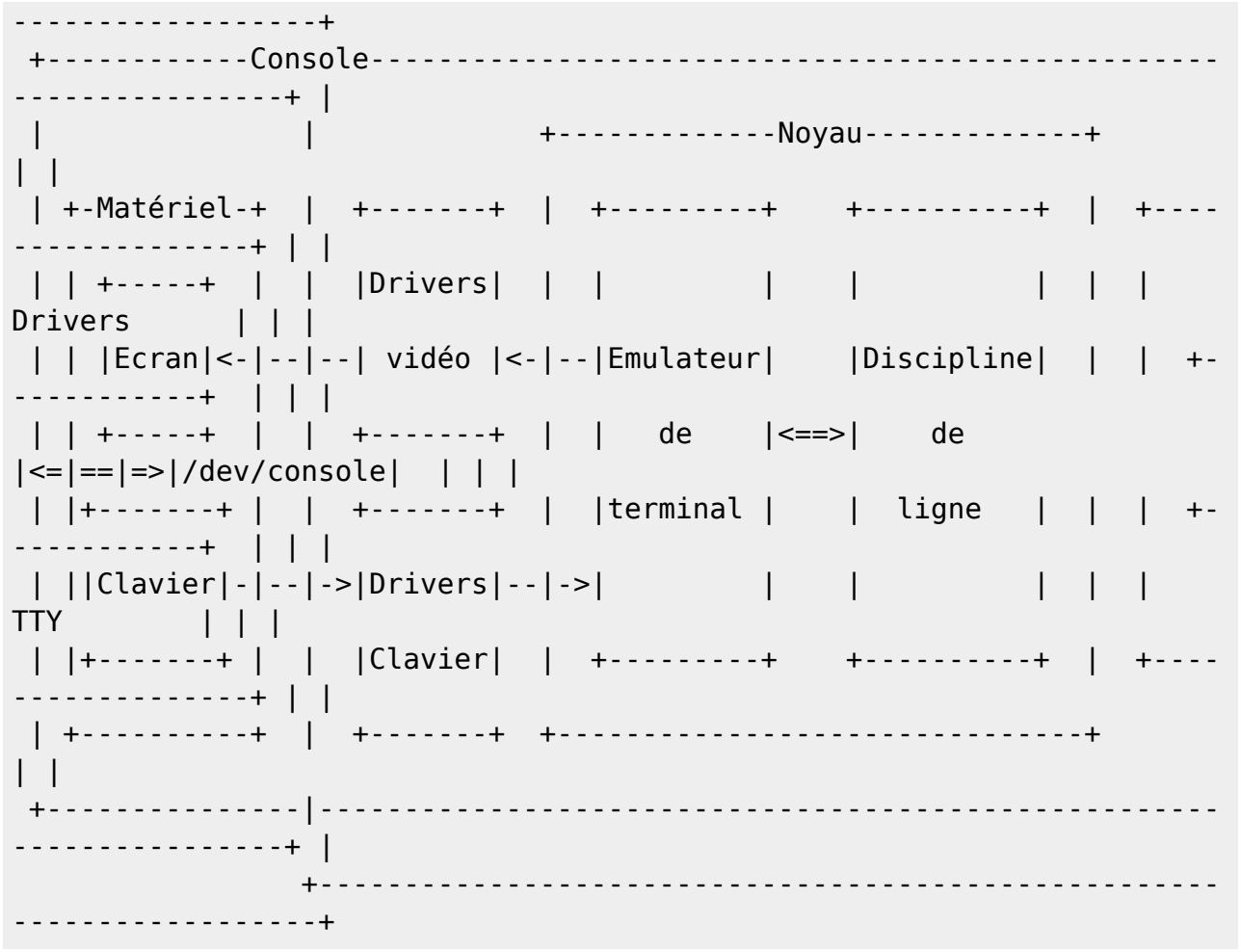
Depuis l'espace utilisateur, nous accédons au ressources du système (détenue par le noyau), que par le niveau 2.
... Ainsi le noyau est protégé de tout accès direct.

Pour simplifier ce schéma, **les trois premiers anneaux** (0, 1 et 2),
Sont regroupés en un seul, pour être **nommer** `l'espace noyau`.
Ainsi nous voyons le système avec :

- L'**espace noyau** (noyau + drivers)
- Et l'**espace utilisateur**.

La console Linux

```
                  +-----------------------Logiciel--------------------
------------------+
                  +---------------------Espace-noyau-------------------
```



Le matériel UART et ses drivers ont disparu, laissant la place aux **drivers vidéo et clavier**.

Le terminal est émulé au sein du noyau.
Le noyau nous le présente donc comme **terminal physique**.
Son émulateur fut **nommé Linux**.
Ce qui donna **le type** `linux`, pour le différencier des autres types de terminaux.

Avec les systèmes GNU/Linux modernes, sur un PC et un écran moderne, la **console** est un terminal graphique.

La documentions du noyau indique :

> (kernel)(en) Documentation : fbcon.txt (traduction) : La console framebuffer (fbcon), comme son nom l'indique, est une console de texte exécutée au-dessus du périphérique framebuffer. Il a les fonctionnalités de n'importe quel pilote de console de texte standard, tel que la console VGA, avec les fonctionnalités supplémentaires qui peuvent être attribuées à la nature graphique du framebuffer.

Le noyau des systèmes Debian GNU/Linux récents, dispose des drivers framebuffer, par défaut.

Les terminaux virtuels

Les **terminaux virtuels** sont le “multiplexage” de la console Linux. Ils Rendent accessible la console dans l'espace utilisateur.

Linux Torvald, dans les sources du noyau à écrit :

> (git.kernel)(en) Torvalds/Linux/drivers/tty/Kconfig (traduction) : Si vous dites Y ici, vous obtiendrez un support pour les terminaux avec des périphériques d’affichage et de clavier. Celles-ci sont dites « virtuelles » car vous pouvez exécuter plusieurs terminaux virtuels (également appelés consoles virtuelles) sur un terminal physique… Vous avez besoin d’au moins un terminal virtuel pour utiliser votre clavier et votre moniteur.

Virtualisant la console, ils sont aussi nommés **consoles virtuelles**.

Par défaut, un système GNU/Linux déployé sur un PC, est configuré pour retourner les messages du système sur les **terminaux virtuels**.
Dans ce cas ils agissent comme une console système.

Mais cela peut etre configuré autrement.

Linux Torvald, dans les sources du noyau à écrit :

> (git.kernel)(en) Torvalds/Linux/drivers/tty/Kconfig (traduction) : Si vous répondez Y ici, un terminal virtuel (le périphérique utilisé pour interagir avec un terminal physique) peut être utilisé comme console système. C’est le mode d’opération le plus courant, vous devez donc dire Y ici, sauf si vous souhaitez que les messages du noyau soient envoyés uniquement sur un port série (auquel cas vous devez dire Y à « Console sur le port série », ci-dessous).

Nous pouvons donc, dire N (non).
Dans ce cas, les **terminaux virtuels**, n'ont plus rien de la définition de console système. Et sont relégués à être de simple **terminaux (virtuels)**.
Il est aussi possible de passer une option au noyau durant le démarrage, en précisant vers quel périphérique le système doit envoyer ses messages.

Un système Debian GNU/Linux moderne, permet d'utiliser 63 **terminaux virtuels**.
Dont 6 sont accessibles avec la combinaison des touches `Ctrl+Alt+F1` à `F6`.
Et sont nommés respectivement **tty1** à **tty6**.

Deux terminaux virtuels (tty1 et tty2), utilisés pour l'exécution d'un shell chacun.

```
               +-----------------------------Logiciel-----------------
---------------+
               +----------------------Espace-noyau--------------------
---+--Espace---+
```

```
                |             +---------Noyau---------+  +--Drivers-TTY-
-+ | utilisat. |
 +-Matériel-+  |  +-------+  |  +---------+   +---+  |  |  +---------+
| |  +-----+  |
 | +-----+  |  |  |Drivers|  |  |         |   |
|<=|==|=>|/dev/tty1|<=|=|=>|shell|  |
 | |Ecran|<-|--|--| vidéo |<-|--|Emulateur|   | D |  |  |  +---------+
| |  +-----+  |
 | +-----+  |  |  +-------+  |  |   de    |<=>| D |  |  |
| |           |
 |+-------+ |  |  +-------+  |  |terminal |   | L |  |  |  +---------+
| |  +-----+  |
 ||Clavier|-|--|->|Drivers|--|->|         |   |
|<=|==|=>|/dev/tty2|<=|=|=>|shell|  |
 |+-------+ |  |  |Clavier|  |  +---------+   +---+  |  |  +---------+
| |  +-----+  |
 +----------+  |  +-------+  +----------------------+  +--------------
-+ |           |
              +-------------------------------------------------------
---+-----------+
Alice a ouvert deux terminaux virtuels, sur son PC, démarrant chacun un
shell.
DDL : Discipline de ligne
```

Les systèmes GNU/Linux sont multi-utilisateurs. Cet avantage permet d'utiliser plusieurs **terminaux virtuels** avec, si nous le désirons, un utilisateur différent.

Même si plusieurs **terminaux virtuels** peuvent être ouvert, un seul uniquement, peut être au premier plan.

Sur un système non graphique, après son démarrage, la sortie du premier **terminal virtuel** est affiché à l'écran.
Le programme **agetty**, qui à ouvert le fichier périphérique **/dev/tty1**, sanctionne son accès, en demandant un nom d'utilisateur.
Pour aller plus loin, nous devons saisir notre nom d'utilisateur, suivit de la touche `entrée`.
Une fois fait, le programme **agetty** est remplacé par le programme **login**, qui nous demande notre mot de passe.
S'il est exacte, **login** crée un processus fils, pour exécuter le **shell**, renseigné dans le fichier /etc/passwd (souvent bash).

Voir aussi : Console.

Les terminaux fenêtre

Avec la venue des interfaces graphiques, il fut nécessaire (utile) de disposer de programme proposant dans une fenêtre, un terminal.
Ces programmes sont nommés **terminaux fenêtre** ou **fenêtres de terminal**.
Ils émulent dans une fenêtre, un terminal texte ou un terminal graphique.

Tous les **émulateurs de terminal** n'émule pas forcément un terminal graphique, cela dépend du programme lui même (l'émulateur) et de son implantation dans une distribution.

Sur les systèmes Debian GNU/Linux, l'interface graphique est disponible sur le 7em terminal virtuel (`Ctrl` + `Alt` + F7).

Ainsi, lors du démarrage du système, le 7em terminal virtuel est affiché.
Son usage est sanctionné par un login graphique nommé gestionnaire de session graphique (login manager) ou gestionnaire d'affichage (display manager).
Pour aller plus loin, nous devons y saisir notre login de connexion (couple nom d'utilisateur + mot de passe).

Dès lors, lorsque nous ouvrons **un terminal fenêtre**, aucun login nous est demandé.
Nous sommes directement connectés avec notre nom d'utilisateur.
Le programme qui est renseigné dans le fichier /etc/passwd (souvent le **shell bash**) y est exécuté.

Nous pouvons ouvrir autant de **terminaux fenêtre** que nous avons besoin.

Dans cette page **serveur X** désigne le serveur d'affichage d'un système GNU/Linux graphique.

Mais aussi l'ensemble de l'architecture et des composants nécessaires au fonctionnement du système graphique.

Les **clients X** sont les programmes qui s"exécutent dans une fenêtre. Ils se connectent au **serveur X** qui se charge de les afficher.
Par exemple, le terminal fenêtre **xterm** est un client X qui émule un terminal.
(Il est par défaut présent, dans un environnent X.)

Pour émuler le comportement d'un terminal, les **terminaux fenêtre** n'utilise pas la console.
C'est un programme (un processus) en espace utilisateur qui émule le terminal. L'émulateur accède au matériel (le clavier et l'écran) par le **serveur X**.
L'émulation est rendue disponible pour un autre processus (par exemple un shell) en utilisant les pseudo-terminaux.

Ainsi, les **terminaux fenêtre** sont aussi nommés **émulateurs de terminal pour X** (qu'il ne faut pas confondre avec les terminaux X).

Voir aussi : Terminal.

Les pseudos terminaux

Les **pseudos terminaux** permettent d'utiliser un émulateur de terminal, que nous exécutons dans l'espace utilisateur ou sur une machine distante.

Pour être différenciés des terminaux virtuels (TTY), les **pseudo-terminaux** sont nommés **PTY**

Ils sont composés de deux parties :

D'une **partie esclave**. A la quelle est connecté, un processus qui souhaite communiquer avec un terminal (émulé). Par exemple un shell.

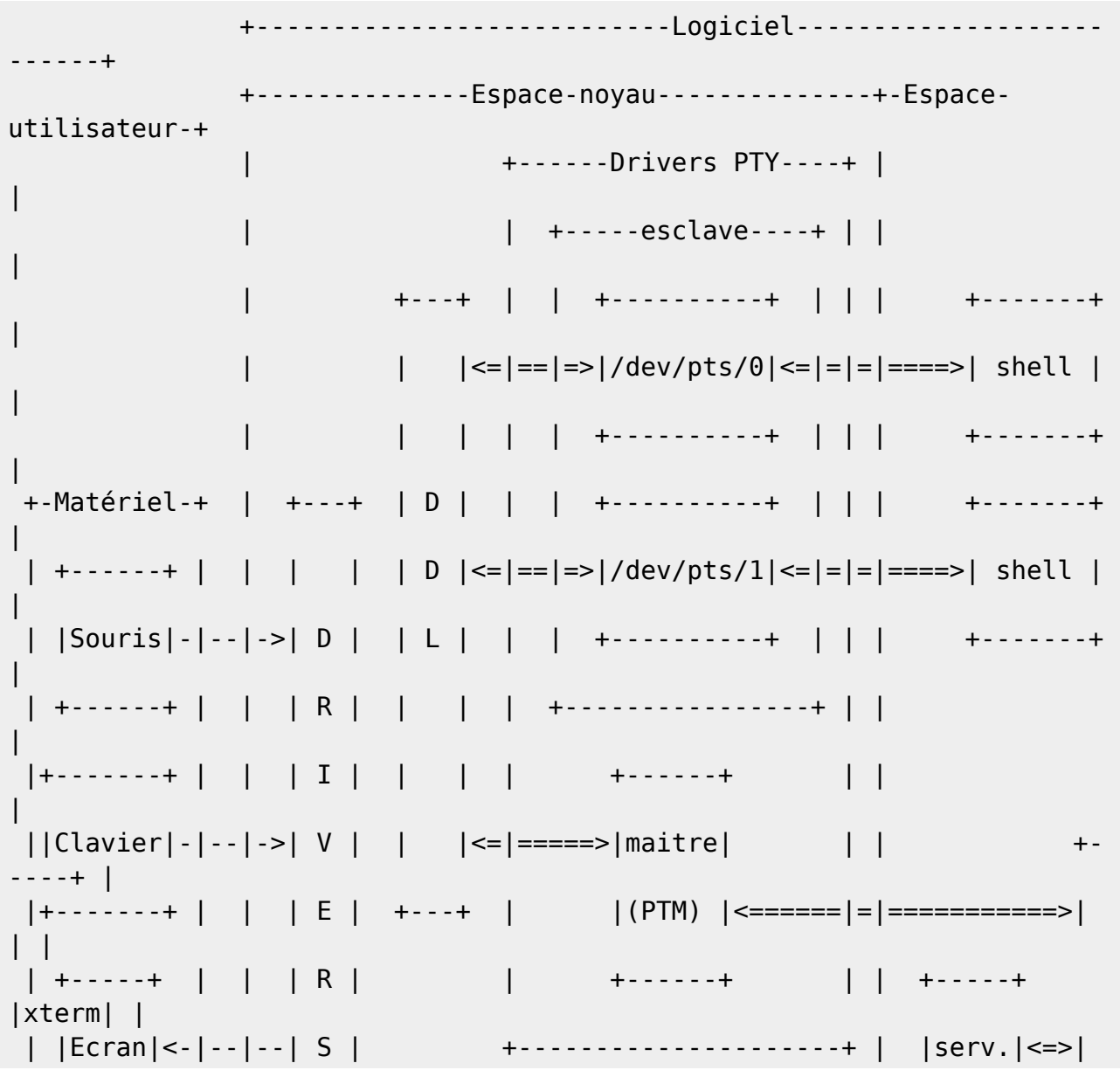
Et d'une **partie maître**. A la quelle est connecté :
Un processus qui émule un terminal.
Ou un processus de connexion au serveur d'accès à distance (sshd). Dans ce cas le terminal émulé provient du client.

Pseudo-terminaux et terminaux fenêtre

Lorsque qu'un utilisateur ouvre un terminal fenêtre (xterm), pour exécuter un shell, un pseudo-terminal est crée.

L'émulateur de terminal (xtern) communique avec la **partie maître**.
Et le shell communique avec la **partie esclave**, via le fichier périphérique **/dev/pts/N** (où N est un nombre).

Deux terminaux fenêtre émulés par le processus xterm



```
| |
 | +-----+  |  |  |   |<================================|=>|  X  |   +-
----+ |
 +----------+  |  +---+                                  |  +-----+
|
               +-----------------------------------------+-------------
------+
Alice à ouvert deux terminaux fenêtre sur son PC.
DDL : Discipline de ligne
```

Pour le même utilisateur, un seul processus d'**xterm**, permet d”émuler plusieurs terminaux.

Tous ce qui arrive sur l'entrée du **terminal émulé** (xterm), depuis le **serveur X**, est envoyé à l'entrée de la **partie esclave**, via la **discipline de ligne**.

Tous ce qui est envoyé sur la sortie de la **partie esclave**, est envoyé en passant par la **ligne de discipline**, au **serveur X** pour être affiché à l'écran. Dans se sens, la ligne de discipline est configuré en mode brut (mode raw). Aucun caractère n'est mis en tampon. La ligne de discipline transmet simplement les octet provenant de la partie esclave à la partie maître, octet par octet.

Ainsi la **partie esclave**, peut être “vue” par le processus au quel elle est connectée, comme un **terminal physique** (VT100, VT220, VT340[21)], etc).

Linus Torvalds a écrit dans les sources du noyau

> (git.kernel)(en) Torvalds : linux.git : Drivers : TTY : kconfig (traduction) : Un pseudo terminal (PTY) est un dispositif logiciel composé de deux moitiés : un maître et un esclave. Le dispositif esclave se comporte de la même manière qu’un terminal physique ; Le dispositif maître est utilisé par un processus pour lire et écrire des données à l’esclave, émulant ainsi un terminal. Les programmes typiques pour le côté maître sont les serveurs telnet et xterms.

Extrait de la page du manuel de pty (7)

```
https://fr.manpages.org/pty/7

Un  pseudo-terminal  est  une  paire de périphériques en mode caractère
virtuels qui fournissent un canal de communication bidirectionnelle. Un
bout  du canal est appelé le maître ;  l’autre bout se nomme l'esclave.
Le bout  esclave  du  pseudo-terminal  fournit  une  interface  qui  se
comporte  exactement  comme  un  terminal  classique.  Un processus qui
s’attend à être connecté à un terminal peut ouvrir le bout esclave d’un
pseudo-terminal, puis être piloté par un programme qui a ouvert le bout
maître. Tout ce qui est écrit sur le maître est fourni au processus via
l’esclave, comme si cela était écrit sur un terminal... Réciproquement,
tout ce qui est écrit  sur  l’esclave peut être lu par le processus qui
est connecté au périphérique  maître.
```

Pseudo-terminaux et shell distant

Lorsqu'un utilisateur distant accède à une machine pour y exécuter un shell.

Le serveur d'accès à distance (sshd), auquel se connecte le client, crée un processus fils.
Ce processus, créer lui même un processus fils. Ce dernier est attribué à l'utilisateur connecté.
Et un **pseudo-terminal** est créer.

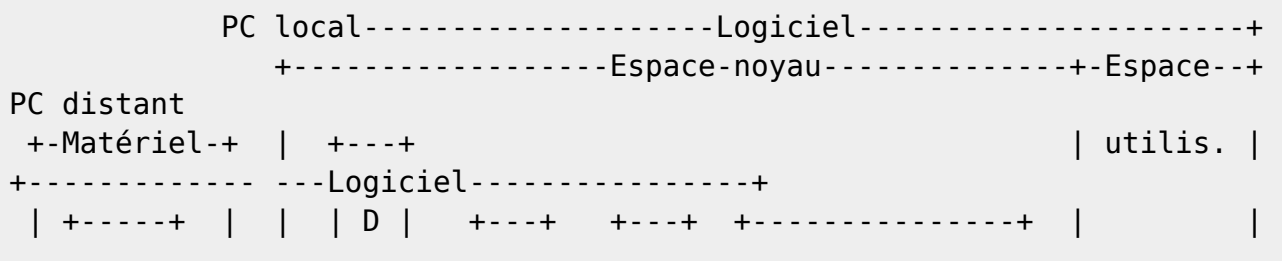
Le processus de connexion communiquera avec la partie maître.
Le processus shell, communiquera avec la **partie esclave**, par le fichier périphérique **/dev/pts/N** (où N est un nombre).

Deux accès distants au shell sur un ordinateur hôte

Le terminal utilisé provient du client.
Il peut être de deux types :

- Un terminal virtuel

Accès à distance via un terminal virtuel



- Ou un terminal fenêtre

Accès à distance via un terminal fenêtre

```
+-----------------+------------------------+
 | |Ecran|<-|--|--| S |                        |  |serv.|<=>|     | |
 | +-----+  |  |  |   |<=======================|=>|  X  |   +-----+ |
PC distant
 +----------+  |  +---+                        |  +-----+           |
               +--------------------------------+--------------------+
Bob utilise ssh depuis un terminal fenêtre, pour se connecter et ouvrir
un shell distant, sur le PC d'Alice.
DDL : Discipline de ligne
PTM : Drivers PTY maître
```

Les consoles graphiques

Une **consoles graphique** * est une **console système graphique**.
Elle permet de communiquer avec l'ordinateur depuis sont démarrage.
Sa sortie est un graphique.

Les consoles graphiques qui permettent d'utiliser les machines virtuelles (qui disposent d'un système graphique) en sont l'exemple.

Une telle console affiche les messages du système et permet la communication entrante, depuis le démarrage de la machine virtuelle.
Une fois démarrée, elle permet d'utiliser le système graphique.

Les machines virtuelles sont vues comme des machines physiques distantes.
Pour qu'une console graphique (qui est locale) puisse communiquer avec une machine (virtuelle) distante, le réseau doit être utilisé.
Des projets comme **VNC**[22)] ou **SPICE**[23)] ont développés des protocoles réseaux, dont la console communique graphiquement avec la machine.

- **VNC** véhicule un flux graphique.
- **SPICE** véhicule un flux vidéo.

Qui dit vidéo, dit graphisme et audio…

Pouvons-nous voir là, une *console système graphique audio* ?

Un autre exemple de **console graphique** est celle que nous utilisons pour communiquer avec un BIOS graphique.

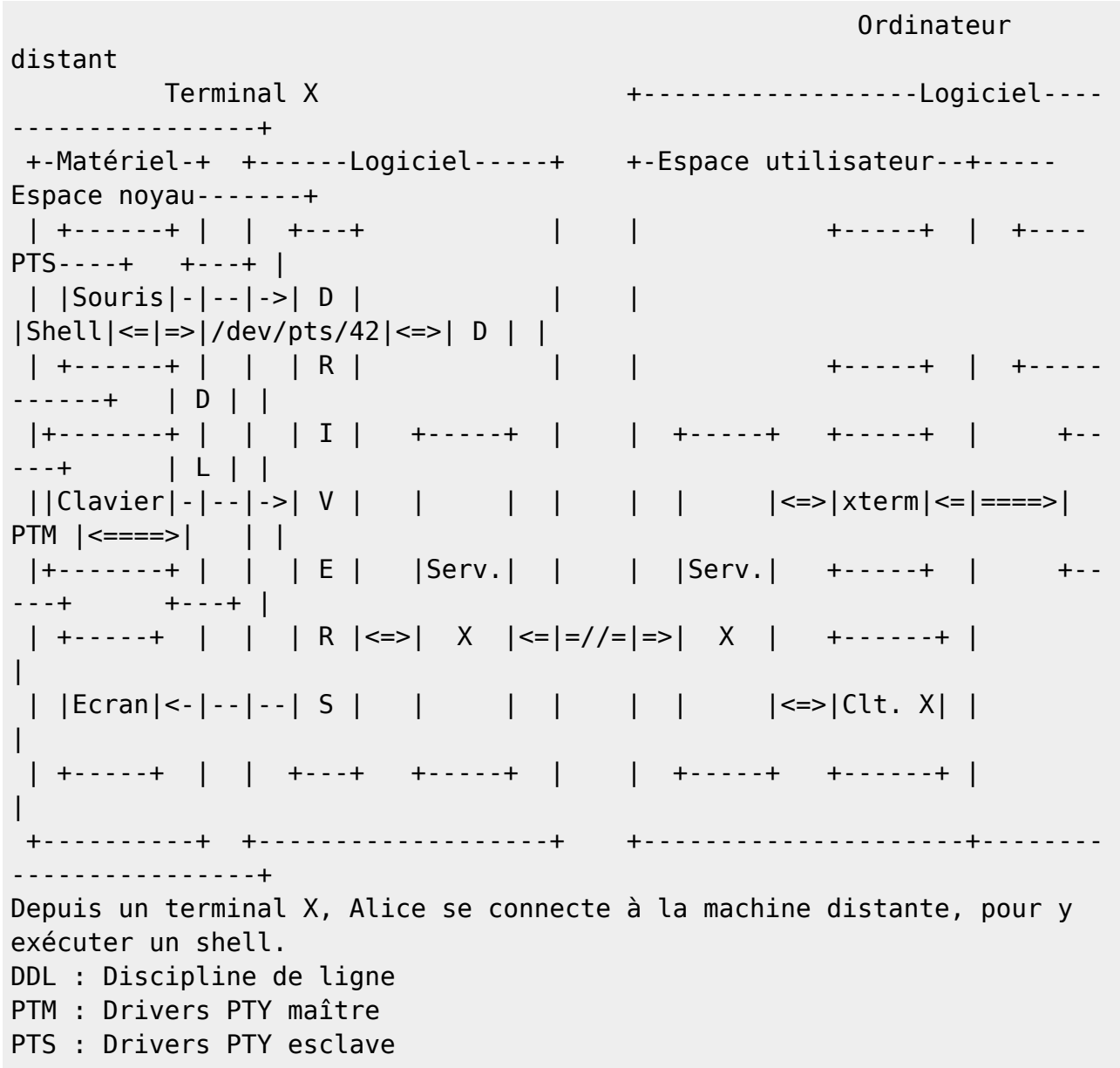
Les terminaux X

Un terminal X est un client léger. Un ordinateur avec peut de ressources (processeur et mémoire), parfois même sans disque.

Il dispose :

- D'un système exploitation (qui peut être chargé depuis le réseau).
- Du serveur X, qui lui permet d'utiliser :
  - Le clavier et la souris en entrée
  - L'écran en sortie
  - Le réseau pour se connecter à d'autre serveur X. Pour y exécuter des clients X , qu'il affichera sur son écran.

Accès à un ordinateur via un terminal X



Multiplexeurs de terminal

Les **multiplexeurs de terminal** permettent d'augmenter le nombre d'interfaces d'un terminal virtuel ou fenêtre.

Ils virtualisent ou émulent des terminaux dans un terminal.

Chaque terminal obtenu exécute un processus shell.
Chaque shell peut être :

- Exécutés sous l'autorité d' un utilisateur différent,
- Placés en arrière plan,
- Remis au premier plan,
- Être supprimer,
- Etc.

Certains **multiplexeur de terminal** n'affiche qu'un seul terminal multiplexé (celui du premier plan). D'autres permettent d'en afficher plusieurs à l'écran ou dans une fenêtre.

T0 - Tutos, préparatifs

Voila le moment où nous devons expliquer par la pratique.

Itinéraire

Cette suite de tutos propose un voyage initiatique dans le monde des terminaux GNU/Linux.

- T0 - Tutos préparatifs
  - Itinéraire **nous sommes ici**
  - Recommandations (conseil pour un meilleur confort)
  - Conventions (qui est quoi et qui somme nous)
  - **Sauvegarde du fichier** `~/.bashrc`
  - Usage de root
- T1 - Le type $TERM
  - Présentation de la variable d'environnement TERM.
  - Son usage dans le fichier `.bashrc` pour coloriser l'invite de l'interpréteur bash.
- T2 - Configuration avec stty
  - Manipulation de la configuration d'un terminal.
- T3. La commande tty
  - Retourne le chemin du fichier périphérique du terminal utilisé.
  - **Modification du fichier** `~/.bashrc`
- (A venir) T4 - Les terminaux virtuels
  - Premières approches
  - Visualisation des processus et des fichiers périphériques, **Modification du fichier** `~/.bashrc`
  - Colorisation de l'invite de bash, **Modification du fichier** `~/.bashrc`
  - Graphisme avec le framebuffer.
- (A venir) T5 - Les terminaux fenêtres
  - Visualisation des processus et des fichiers périphériques
  - Redimensionnement du terminal
  - Graphisme avec l’émulateur xterm.
- (A venir) T6 - Les terminaux connectés
  - Visualisation des processus et des fichiers périphériques associés ;

- (A venir) T7 - Les périphériques tty et tty0
  - Deux fichiers de périphérique spéciaux.
- (A venir) T8 - Le périphérique console
  - Le terminal du noyau.
- TX - Retour à l'état initial
  - Fin des tutos. Le système est remis à son état initial.
  - **Restauration du fichier** `~/.bashrc`

Recommandations

Pour un confort pratique, nous conseillons de réaliser cette suite de tutos dans une machine virtuelle.

Car nous allons utiliser des terminaux virtuels en ligne de commande.
Il nous sera alors pas possible de lire les instruction affichées sur cette page.

Disposer d'une machine virtuelle, dont la console est affichée dans une fenêtre, pour exécuter les tutos.
Permet d'avoir accoté, la fenêtre du navigateur qui affiche les instruction de cette page.

Pour créer facilement (sans ligne de commande) une machine virtuelle, voir le tuto Virt-manager : Un gestionnaire graphique de machine virtuelle.

Vous devriez déjà avoir une sauvegarde de votre fichier `.bashrc`.
Mais, nous ne jouerons pas là dessus.

... Alors, nous le sauvegarderons tout même plus bas

Conventions

Dans cette suite de tutos, nous utilisons les termes suivants :

Désignations	Noms	Périphériques (fichiers)
Console linux	Console	/dev/console
Terminal virtuel	Tty**N**	/dev/tty**N**
Terminal fenêtre	Pts**N**	/dev/pts/**N**
Terminal connecté	Pts**N**	/dev/pts/**N**
Terminal	Désigne indifféremment un terminal physique ou émulé, la console Linux, un terminal virtuel, fenêtre ou connecté.	
Avec **N**, le numéro du terminal (un décimal entier).		

Nous utilisons ici d'un PC disposant d'un système Debian GNU/linux 11 (Bullseye).
Le bureau graphique **xfce4** y a été installé.

Après reboot (redémarrage), le gestionnaire d'affichage **lightdm** nous propose de saisir notre nom

d'utilisateur et notre mot de passe.
Ici, nous utilisons l'utilisateur **DF**, pour nous connecter.

Une fois connecté, nous sommes sous le bureau d'**Xfce**.

Nous lançons une **fenêtre de terminal** (ici, **xfce4-treminal**).
... L'interpréteur de commande **bash** y est exécuté.

Les fichiers de configuration de bash de l'utilisateur **DF**, n'ont pas été modifiés.
Les fichiers **~/.profile** et **~/.bashrc**, sont les fichiers donnés par défaut par un système Debian 11 GNU/Linux.

Sauvegarde

Plusieurs fois, nous modifierons notre fichier `.bashrc` :

- Au tuto T3 - La commande tty, pour ajouter le nom du terminal utilisé à l'invite (le prompt) de l'interpréteur de commande (bash).
- Au tuto T4 - Les terminaux virtuels, pour forcer la colorisation de l'invite pour les terminaux virtuels et nous y ajouterons deux alias qui nous faciliterons l'écriture de commande.

Ces modifications pourraient être réalisées en exécutant leurs commandes directement dans le terminal, sans modifier le fichier `.bashrc`.
Mais elles seraient appliquées qu'à l'interpréteur de commande de ce terminal (dans l'interface où elles ont été exécutées) et ne le seraient pas pour les autres.
C'est pourquoi nous plaçons nos commandes dans le fichier `.bashrc`, qui lui, est interprété à chaque démarrage interactif de bash.

Ainsi ces modifications seront disponibles pour tous shells bash que nous exécuterons, lorsque nous ouvriront un nouveau terminal.

Avant de modifier le fichier `~/.bashrc`, nous devons en réaliser une sauvegarde.
Ainsi nous pourrons restaurer son état actuel, à la fin de notre voyage.

Nous le copions simplement dans le répertoire **`~/DF-tuto`**, que nous créons préalablement, dans le répertoire personnel de notre utilisateur.

```
cd ~            # Déplacement dans le dossier personnel
mkdir -v DF-tuto    # Création du répertoire DF-tuto

# Copie du fichier .bashrc dans DF-tuto
cp -v .bashrc DF-tuto
```

```
mkdir: création du répertoire 'DF-tuto'
'.bashrc' -> 'DF-tuto/.bashrc'
```

Voila qui est bien fait

Root

Nous utiliserons l'utilisateur **root**[24] via la commande **su**[25] pour exécuter des commandes sous son autorité.

Le mot de passe root doit donc être connu.

T1 - Le type $TERM

La variable d'environnement **TERM** contient le type du terminal utilisé.

Tous les terminaux n'ont pas les même fonctionnalités que nous nommons **capacités**, ni les mêmes méthodes pour y parvenir. Par exemples : Certains disposent de touche de contrôle que d'autre non pas ; Les séquences d'échappement peuvent différer ; Certains sont capables d'afficher des couleurs, d'autres sont mono-chrome ; Etc.

Le système dispose d'une base de données, constituée de fichiers binaires, pour mémoriser toutes les différences. Chaque type de terminal dispose de son fichier décrivant ses capacités.

La première base de données (1978) fut nommés **termcap**[26] pour “terminal capcities” (capacités des terminaux).
Termcap est noté obsolète sur de nombreux système (dont Debian GNU/Linux).
Elle laisse la place à **terminfo**[27].

Extrait de la page de manuel terminfo (5)

```
https://manpages.org/terminfo/5 (traduction)

Terminfo  est une base de données décrivant les terminaux,  utilisée
par des
programmes  orientés  écran  tels que nvi(1),  lynx(1),  mutt(1) et
d'autres
applications curses, utilisant des appels de haut niveau à des
bibliothèques
telles que ncurses (3NCURSES ).  Il est également  utilisé via des
appels de
bas niveau par des applications non curses qui  peuvent être orientées
écran
(telles que clear(1)) ou non écran (telles que tabs(1)).
```

Ainsi, un programme peut s'il le souhaite, consulter la variable **TERM**, pour connaître le type de terminal qu'il utilise. Et consulter la base de données **terminfo**, via la librairie **ncurses**[28] (ou pas), afin de communiquer correctement avec le terminal. Par exemple, il est inutile d'envoyer la séquence d'échappement affichant le texte en rouge, sur des terminaux mono-chrome.

Terminfo reconnaît de nombreux types de terminal.
La commande **toe -a**[29] en retourne la liste.

Nous n'affichons pas ici la liste des terminaux connus par **terminfo**, car elle est trop longue.

Nous demandons donc à la commande **wc -l**[30] d'en compter le nombre (nombre de ligne retourné par la commande **toe -a**).

```
echo "Nombre de type de terminaux reconnus par terminfo : $(toe -a | wc -l)"
```

```
Nombre de type de terminaux reconnus par terminfo : 1771
```

1771 terminaux. (Pas tout à fait, car certains sont des liens pointant vers d'autres. Ainsi un type peut disposer de plusieurs noms.)
Beaucoup sont exotiques et nous les rencontrons jamais.

Les fichiers des capacités sont compilés avec la commande **tic**[31]. Depuis des fichiers sources au format **term**[32]
Ils sont situés dans différents répertoires. Le fichier **/etc/terminfo/README** nous l'indique.

/etc/terminfo/README

```
(Traduction)
Ce repétoire est pour les descriptions de terminfo du système local.
Par défaut,
ncurses recherchera en premier dans ${HOME}/.terminfo, puis dans
/etc/terminfo
(ce répertoire), puis dans /lib/terminfo, et enfin dans
/usr/share/terminfo.
```

Les fichiers de description de terminfo (des capacités des terminaux), placés dans les répertoires **/lib/terminfo** et **/usr/share/terminfo**, sont classés par ordre alphabétique dans des sous-dossiers.

Par exemple dans le répertoire **/lib/terminfo** :

```
ls /lib/terminfo
```

```
a  c  d  E  h  l  m  p  r  s  t  v  w  x
```

```
echo /lib/terminfo/?/*
```

```
/lib/terminfo/a/ansi /lib/terminfo/c/cons25 /lib/terminfo/c/cons25-debian
/lib/
terminfo/c/cygwin /lib/terminfo/d/dumb /lib/terminfo/E/Eterm
/lib/terminfo/E/Et
erm-color /lib/terminfo/h/hurd /lib/terminfo/l/linux...
```

Les fichiers de **terminfo** sont des fichiers binaires :

```
file /lib/terminfo/l/linux
```

```
/lib/terminfo/l/linux: Compiled terminfo entry "linux"
```

Pour montrer la capacité des terminaux, nous pouvons utiliser la commande **man** (qui affiche les pages des manuels).
Elle utilise la commande **less** pour pager les pages d'un manuel.
La commande **less** utilise **terminfo** pour obtenir les capacités du terminal utilisé.
La variable **TERM** renseigne alors quel fichier de **terminfo** doit être lu.

```
man bash
```

La page de manuel est affichée, nous pouvons y naviguer avec les touches Page↑, Page↓, Fin et Début.
Bien… Quittons avec la touche Q.
Puis relançons la commande en précisant un autre type de terminal.

```
TERM=vt102 man bash
```

Les touches Page↑, Page↓, Fin et Début ne fonctionne plus !
C'est normal… Le terminal VT102, ne prend pas en charge ces touches.
Quittons avec Q.

Modifier la variable **TERM** ne modifie pas le type de terminal utilisé.
Elle indique seulement aux applications qui le souhaitent quel type de terminal utiliser.

Un autre exemple, avec la commande **top** (qui affiche la liste les processus exécutés) :

```
top
```

Puis appuyons plusieurs fois sur la touche Z.
Nous pouvons observer une différence à chaque fois que nous appuyons sur Z.
Bien… Quittons avec la touche Q.
Puis relançons **top** en précisant un autre type de terminal.

```
TERM=vt102 top
```

Appuyons plusieurs fois sur Z. Puis Q pour quitter.
Il n'y a plus de changement de couleur !
C'est normal… Le terminal VT102 est un terminal mono-chrome .

Affichons le type de terminal de notre terminal fenêtre (ici **xfce4-terminal**).

```
echo $TERM
```

```
xterm-256color
```

Xfce4-terminal est de type **xterm-256color**. Nous indiquant (entre autres) qu'il prend en charge 256 couleurs.

Actuellement **xterm-256color** est le type utilisé par la plus part des terminaux fenêtre. Mais pas tous...

Les différents types de terminal courants		
Terminaux	**Bureaux GUI**	**Types**
Terminaux virtuels		
Tty1 à tty6	Aucun	linux
Terminaux fenêtres		
Xterm / uxterm	Indéfinit	xterm
Xfce4-terminal	Xfce	xterm-256color
Lxterminal	LXDE	xterm-256color
Mate-terminal	Mate	xterm-256color
Terminator	Gnome	xterm-256color
Gnome-terminal	Gnome/Cinnamon	xterm-256color
Terminator	Gnome	xterm-256color
Konsole	KDE	xterm-256color
Eterm	Enlightenment	Eterm
Terminology	Enlightenment	xterm-256color
Aterm (urxvt)	Indéfinit	rxvt-unicode-256color
Mutilpexeur de terminal		
Screen	Aucun/indéfinit	screen.linux, screen.xterm, screen.xterm-256color, screen.Eterm, etc
Tmux	Aucun/indéfinit	screen

Tous les terminaux cités ci-dessus permettent la colorisation du texte. Ceux qui l'indique permettent l'usage de 256 couleurs, les autres utilisent 8 couleurs.

La commande **infocmp <type>** permet de retourner via **terminfo**, les capacités d'un terminal.

Ci-dessous, nous utilisons une boucle **for**, qui va pour chaque type de terminal listé,

- Mémoriser le type dans la variable **t**,
- Extraire le nombre de couleur qu'il supporte, de la sortie de la commande **infocmp**, dans la variable **c**.

Puis utiliser la commande **printf** pour afficher de façon formaté le contenu des variables **t** et **c**.

```
echo "Types           Couleur(s)"
```

```
echo "-------------------------"
for t in vt102 vt220 linux xterm xterm-color xterm-256color Eterm screen
do                  # Pour chaque type listé
    c=$(infocmp $t | grep 'colors#')    # Récupère la ligne contenant le
nombre de couleur, de la commande infocmp
    c=${c#*colors\#}            # Supprime tout ce qui se trouve avant le
nombre de couleur
    c=${c%%,*}                # Supprime tout ce qui se trouve après le
nombre de couleur
    printf "%-16s %3u\n" $t $c        # Affiche le type et le nombre de
couleur
done
unset t c               # Supprime les variables utilisées
```

```
Types           Couleur(s)
-------------------------
vt102              0
vt220              0
linux              8
xterm              8
xterm-color        8
xterm-256color   256
Eterm              8
screen             8
```

La commande **infocmp <type1> <type2>** permet de comparer les capacités deux types de terminal.
Sans option, seuls les différences sont affichées.

```
infocmp xterm xterm-256color
```

```
comparing xterm to xterm-256color.
    comparing booleans.
    ccc: F:T.
    comparing numbers.
    colors: 8, 256.
    pairs: 64, 65536.
    comparing strings.
    initc: NULL,
'\E]4;%p1%d;rgb\:%p2%{255}%*%{1000}%/%2.2X/%p3%{255}%*%{1000}%/%2.2X/%p4%{25
5}%*%{1000}%/%2.2X\E\\'.
    oc: NULL, '\E]104\007'.
    rs1: '\Ec', '\Ec\E]104\007'.
    setab: '\E[4%p1%dm', '\E[%?%p1%{8}%<%t4%p1%d%e%p1%{16}%<%t10%p1%{8}%-
%d%e48;5;%p1%d%;m'.
    setaf: '\E[3%p1%dm', '\E[%?%p1%{8}%<%t3%p1%d%e%p1%{16}%<%t9%p1%{8}%-
%d%e38;5;%p1%d%;m'.
    setb:
'\E[4%?%p1%{1}%=%t4%e%p1%{3}%=%t6%e%p1%{4}%=%t1%e%p1%{6}%=%t3%e%p1%d%;m',
NULL.
```

```
    setf:
'\E[3%?%p1%{1}%=%t4%e%p1%{3}%=%t6%e%p1%{4}%=%t1%e%p1%{6}%=%t3%e%p1%d%;m',
NULL.
```

Sur un système Debian GNU/Linux, la variable **TERM** est utilisée pour savoir si le prompt (l'invite) de bash peut-être colorisé.
Cela se passe dans le fichier **.bashrc** situé dans le répertoire de l'utilisateur.

Extrait du fichier .bashrc sur un système Debian GNU/Linux 11

```
...
case "$TERM" in
    xterm-color|*-256color) color_prompt=yes;;
esac
...
if [ "$color_prompt" = yes ]; then
PS1='${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h\[\033[00m\]
:\[\033[01;34m\]\w\[\033[00m\]\$ '
else
    PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w\$ '
fi
...
```

La commande **case**, permet de définir si la variable **TERM**, contient **xterm-color** ou si son contenu se termine par **-256color**.
Si c'est le cas, la variable **color_prompt** est valoriser avec **yes**.
...
Puis est testé, si la variable **color_prompt** contient **yes**.
Si oui, alors le prompt (définit dans la variable **PS1**) dispose des séquences d'échappement de colorisation.
Sinon, il est définit sans.

T2 - Configuration avec stty

La commande **stty** (set tty) permet d'afficher ou de modifier la configuration d'un terminal.

L'option **-a** (ou **--all**) de la commande **stty** permet de retourner la configuration du terminal utilisé :

```
stty -a
```

```
speed 38400 baud; rows 24; columns 80; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>;
eol2 = <undef>; swtch = <undef>; start = ^Q; stop = ^S; susp = ^Z; rprnt =
^R;
werase = ^W; lnext = ^V; discard = ^O; min = 1; time = 0;
-parenb -parodd -cmspar cs8 -hupcl -cstopb cread -clocal -crtscts
```

```
-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl ixon -
ixoff
-iuclc -ixany -imaxbel iutf8
opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0
ff0
isig icanon iexten echo echoe echok -echonl -noflsh -xcase -tostop -echoprt
echoctl echoke -flusho -extproc
```

La première ligne
`speed 38400 baud; rows 24; columns 80; line = 0;`
Indique :

- La vitesse de transition de l'entrée et de la sortie du terminal.
- Le nombre de ligne et de colonne de la sortie du terminal.
- Et le numéro de la discipline de ligne.

Puis nous trouvons la configuration des caractères de contrôle :
`intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>; eol2 = <undef>; swtch = <undef>; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R; werase = ^W; lnext = ^V; discard = ^O; min = 1; time = 0;`.
Ainsi configuré :

- Ctrl+C permet d'arrêter l'exécution du processus du premier plan. Ou de mettre fin à l'écriture d'une ligne de commande, sans l'interprétée.
- Ctrl+\, émet le signal quit.
- Ctrl+D, envoie le signal de fin de fichier. Il indique la fin d'un flux. S'il est utilisé à l'invite (au prompt) et qu'aucune saisie ne soit effectuée, le signal est envoyé à l'entrée du terminal. Ce qui a pour effet d'indiquer la fin du flux de l'entrée standard et donc de fermer le terminal. Puisque le terminal, sans son entrée n'a plus aucun intérêt.
- Ctrl+Z, suspend l'exécution du processus du premier plan, pour le placé en arrière plan.
- Ctrl+S, suspend les processus (en premier plan et en arrière plan) qui écrivent sur la sortie standard.
- Ctrl+Q, reprend l'exécution des processus suspendus par Ctrl+S.
- Ctrl+?, efface le caractère précédent le curseur (équivalent à ← Retour Arrière)
- Ctrl+U, "coupe" du début de la ligne, jusqu'au curseur. (La ligne entière, si le curseur est en fin de ligne.)
- Ctrl+W, "coupe" du début d'un mot, jusqu'au curseur. (Le mot entier, si le curseur est à la fin du mot.)
- Ctrl+R, effectue une recherche dans l'historique.

La commande **stty** ne permet pas de configurer tout les caractères de contrôle disponibles dans un terminal.

En voici d'autres très utiles :

- Ctrl+A : Déplace le curseur en début de ligne (équivalent à Début).
- Ctrl+E : Déplace le curseur en fin de ligne (équivalent à Fin).
- Ctrl+K : Coupe tout ce qui est après le curseur.
- Ctrl+Y : Colle (yank) ce qui a été coupé.

Vient en suite les paramètres configurables. Tous les paramètres disponibles sont donnés. Ceux préfixés par le signe négatif (**-**) sont désactivés.

Les paramètres de contrôle :
`-parenb -parodd -cmspar cs8 -hupcl -cstopb cread -clocal -crtscts`.

Les paramètres d'entrée :
`-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl ixon -ixoff -iuclc -ixany -imaxbel iutf8`.

Les paramètres de sortie :
`opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0 ff0`.

Et les paramètres locaux :
`isig icanon iexten echo echoe echok -echonl -noflsh -xcase -tostop -echoprt echoctl echoke -flusho -extproc`.

Pour une description des paramètres, voir la page du manuel de la commande **stty**[33)] avec la commande `man stty`.

Sans option la commande **stty** retourne la vitesse de transmission, le numéro de la discipline de ligne. Et la différence entre une configuration saine (sane en anglais) et la configuration actuelle.

```
stty
```

```
speed 38400 baud; line = 0;
-brkint -imaxbel iutf8
```

Nous remarquons qu'à l'ouverture d'un terminal, sa configuration à déjà été modifiée.

- **-brkint**, indique que le paramètre **brkint** (le « break » provoque un signal d'interruption) est désactivé.
- **-imaxbel**, indique que le paramètre **imaxbel** (émettre un bip sonore et ne pas vider un tampon d'entrée plein lors de l'arrivée d'un caractère) est désactivé.
- Le paramètre **iutf8** (supposer que les caractères d'entrées sont codés en UTF-8) est activé.

L'option **-g** (ou **--save**) de la commande **stty** retourne la configuration actuel du terminal dans un format compréhensible par **stty**.

```
stty -g
```

```
4500:5:bf:8a3b:3:1c:7f:15:4:0:1:0:11:13:1a:0:12:f:17:16:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0
```

C'est très pratique pour sauvegarder la configuration du terminal,

- Dans un fichier avec `stty -g > fichier`

    - Restauré avec `stty $(cat fichier)`
- Ou dans une variable avec `var=$(stty -g)`
    - Restauré avec `stty $var`

Sauvegardons notre configuration dans la variable **stty**

```
stty=$(stty -g)
echo stty=$stty
```

```
stty=4500:5:bf:8a3b:3:1c:7f:15:4:0:1:0:11:13:1a:0:12:f:17:16:0:0:0:0:0:0:0:0
:0:0:0:0:0:0:0:0
```

Voila qui est bien fait
Jouons un peu avec les paramètres.

Le paramètre de sortie **onlcr** indique au terminal de convertir un saut de ligne en un saut ligne + un retour chariot.
... Désactivons le :

```
stty -onlcr
```

Demandons l'affichage de la différence avec une configuration saine :

```
stty
```

```
speed 38400 baud; line = 0;
                         -brkint -imaxbel iutf8
                                              -onlcr
```

Le saut de ligne ne dispose plus de retour chariot.
Réactiver le paramètre permet de revenir à l'état normal.

```
stty onlcr
```

Continuons à jouer et grimpons d'un niveau en désactivant le paramètre local **echo**.

```
stty -echo
```

Puis affichons la différence avec la configuration saine :

```
stty
```

```
speed 38400 baud; line = 0;
-brkint -imaxbel iutf8
-echo
```

La saisie ce fait en aveugle.
Le paramètre **echo** fait l'affichage des caractères en entrée. Alors le désactiver...
Réactiver le paramètre permet de revenir à l'état normal.

```
stty echo
```

Les paramètres spéciaux **rows** et **cols** permettent d'indiquer au noyau le nombre de ligne et de colonne que dispose le terminal.

Commençons par récupérer les valeurs actuelles, retournées par le paramètre **size** :

```
stty size
```

```
24 80
```

```
s=$(stty size)      # Récupère le retour de stty size dans la variable s

r=${s% *}       # Mémorise le nombre de ligne dans la variable r
c=${s#* }       # Mémorise le nombre de colonne dans la variable c

echo "Nombre de ligne : $r"
echo "Nombre de colonne : $c"

unset s             # Destruction de la variable s
```

```
Nombre de ligne : 24
Nombre de colonne : 80
```

Et modifions en le nombre :

```
stty rows 10 cols 40
man stty
```

```
STTY(1Commandes de l'utilisateuSTTY(1)

NOM
       stty  - Modifier et afficher la
       configuration de  la  ligne  de
       terminal

SYNOPSIS
       stty    [-F    PÉRIPHÉRIQUE   |
line 1 (press h for help or q to quit)
```

Appuyons sur la touche Q pour quitter. Et restaurons les valeurs initiales :

```
stty rows $r cols $c
```

```
unset r c   # Destruction des variables r et c
```

La commande **stty** dispose de paramètre combiné (qui regroupent plusieurs paramètres en un seul).

Par exemples :
Le paramètre **raw** permet de passer le terminal en mode brut.
La discipline de ligne passe alors en mode caractère et aucune modification n'est appliquée ni à l'entrée, ni à la sortie.

```
stty raw
stty
```

```
speed 38400 baud; line = 0;
                          min = 1; time = 0;
                                       -brkint -icrnl -imaxbel
                                                           -opost
                                                                 -
isig
-icanon
```

Remarquons que le retour chariot n'est plus ajouté au saut de ligne. Cela n'est pas obtenu comme plus haut, par la négation du paramètre **onlcr** (**-onclcr**). Mais par la négation du paramètre **opost** (effectuer un post-traitement de la sortie).

Pour restaurer notre terminal dans son état d'origine, nous pouvons utiliser le paramètre **sane** (traduction : saine) avec les paramètres donnés plus haut avec la commande **stty** sans option ni paramètre.

```
stty sane -brkint -imaxbel iutf8
```

Nous aurions pu aussi utiliser la sauvegarde que nous avons réaliser dans la variable **stty**

```
stty $stty
```

Le mode brut (raw) empêche aussi l'utilisation de certaines touches de contrôle.
Comme par exemple le caractère d'interruption (Ctrl+C) qui termine le processus du premier plan.

Pour mettre cela en évidence, nous allons utiliser une boucle qui va afficher les secondes pendant 10 secondes.
Pendant son déroulement nous l'interrompons en appuyant sur les touches Ctrl+C.

```
(for t in {1..10}; do echo $t; sleep 1; done)
```

```
1
2
3
^C
```

Nous exécutons notre boucle **for** entre parenthèse **(**...**)**, pour l'exécuter dans processus fils.
Ainsi son processus se termine en même temps que la boucle.
Et la variable **t** n'existe pas, (puisque qu'elle à été initialisée dans le processus fils).

... Cela fonctionne, testons en mode brut.

```
stty raw
(for t in {1..10}; do echo $t; sleep 1; done)
```

```
1
 2
  3
   ^C4
      5
       6
        ^C7
           8
            9
              10
```

Cela ne fonctionne plus.
C'est le paramètre local **-isig** (la négation de **isig**), apporté par le paramètre combiné **raw** qui fait cela.

Restaurons notre terminal avec sa configuration d'origine.

```
stty $stty

unset stty  # Destruction de la variable stty
```

T3 - La commande tty

Dans un système GNU/Linux, un appareil (un périphérique) est caractérisé (accessible), par un fichier spécial, aussi nommé fichier de périphérique (ou plus communément, périphérique).
Ces fichiers sont placés dans le répertoire `/dev` et ses sous-répertoires (dev pour device en anglais, appareil en français).

Les périphériques des terminaux virtuels,
Sont nommés **tty<n>** (avec <n>, un entier décimal, de 1 à 63).
Ils sont situés dans le répertoire `/dev` (`/dev/tty`<n>).
Par exemples : `/dev/tty1`, `/dev/tty2`, etc.

Le fichier périphérique `/dev/tty0` est particulier, il caractérise toujours le terminal virtuel qui est affiché.
(Actuellement le 7em, puisque nous affichons l'interface graphique.)
Son usage est présenté dans la section T7 - Les périphériques tty et tty0.

Les périphériques des terminaux fenêtre ou connecté (les pseudo-terminaux)
Sont nommés **pts<n>** (avec <n> un entier décimal).
Leurs fichiers sont numérotés dans le répertoire `/dev/pts` (`/dev/pts/<n>`).
Par exemples : `/dev/pts/0`, `/dev/pts/1`, etc.

La commande tty,

Retourne le chemin du fichier périphérique du terminal utilisé par la commande.

La commande **tty**, ne doit pas être confondue avec le périphérique `/dev/tty`.

Le fichier périphérique `/dev/tty` est particulier, il caractérise toujours le terminal utilisé, qu'il soit virtuel, fenêtre ou connecté.
Son usage est présenté dans la section T7 - Les périphériques tty et tty0.

Le chemin de la commande **tty** est `/usr/bin/tty`.

```
which tty
```

```
/usr/bin/tty
```

```
tty
```

```
/dev/pts/0
```

La commande nous indique que le terminal que nous utilisons, est le périphérique `/dev/pts/0`.
Le chemin retourné montre que les terminaux fenêtres sont des pseudo-terminaux.

Nous venons de redémarrer ; Avons ouvert qu'un seul terminal fenêtre ; Et aucun PC distant n'est connecté au notre.
Alors le système n'a créé qu'un seul pseudo-terminal (celui que notre terminal utilise). Il l'a numéroté (et nommé) 0, le premier nombre disponible.
Mais, nous y reviendrons...

Par la suite, nous allons ouvrir plusieurs terminaux.
Pour mieux nous situer, nous allons insérer dans l'invite de notre interpréteur de commande (le prompt), le nom du périphérique utilisé.

L'invite (le prompt) affichée, est le retour de l'interprétation du contenu de la variable **PS1**, définit dans le fichier `~/.bashrc`.

Nous pourrions modifier cette variable directement dans notre terminal, mais la modification ne serait pas disponible pour les autres terminaux que nous ouvrirons.
Pour qu'elle le soit, elle doit être écrite dans le fichier `~/.bashrc`.

Rappel : A la section T0 - Tutos, préparatifs, nous avons sauvegardé le fichier `~/.bashrc` afin de pouvoir le restaurer plus tard.

Nous n'allons pas insérer le chemin complet du périphérique dans l'invite, mais son nom simplifié :

1. La commande **tty** permet de récupérer le chemin du périphérique.
2. De ce chemin (`/dev/tty<n>` ou `/dev/pts/<n>`), est formé le nom du périphérique (tty<n> ou pts<n>).
3. Puis, le nom obtenu est passé en majuscule, en étant ajouté au début du contenu de la variable PS1.

L'opérateur de redirection » permet d'ajouter la sortie standard à la fin d'un fichier.

Placer entre guillemets, simple ou double, une chaîne de caractère peut s'étendre sur plusieurs lignes.

Nous utilisons ici, les guillemets simples, pour que la chaîne de caractère ne soit pas interprété par bash.

```
echo '
# Insère le nom du terminal dans PS1
n=$(tty)    # Récupère le chemin du périphérique dans la variable n
n=${n#/dev/}        # Supprime /dev/ du chemin
n=${n///}       # Supprime le / restant (si pseudo-terminal)
PS1=${n^^}:$PS1 # Ajoute le nom en majuscule à PS1
unset n     # Destruction de n
' >> ~/.bashrc
```

Pour que cette modification soit prise en compte, il est nécessaire de réinterpréter le fichier `~/.bashrc`, avec la commande **.** (point) ou **source**.

La commande **.** (synonyme de **source**) est une commande interne de l'interpréteur bash.

Elle ne doit pas être confondue avec l'usage du point dans les chemins de systèmes de fichiers :

- Le point seul **.** ou **./** en début du chemin, indiquent le répertoire courant.
- Un nom de fichier (ou de dossier) débutant par un point (**.<nom>**) , indique que cet élément est caché (par exemple `.bashrc`).

```
. ~/.bashrc
```

```
PTS0:DF@pc24:~
```

Yes ! Maintenant, nous ne pouvons plus nous perdre.
Nous allons maintenant nous intéresser aux terminaux virtuels.

TX - Retour à l'état initial

... Nous arrivons à la fin de cette suite de tutos.
Nous remettons ici le système, tel qu'il était lorsque nous avons commencé notre voyage.

Le fichier `~/.bashrc` **doit être restauré**.

```
cd
cp -v ~/DF-tuto/.bashrc .
```

```
'DF-tuto/.bashrc' -> './.bashrc'
```

Et le répertoire `~/.DF-tuto` **supprimé**.

```
rm -vrf ~/DF-tuto
```

```
'DF-tuto/.bashrc' supprimé
répertoire 'DF-tuto' supprimé
```

Voila notre voyage est terminé.
La compagnie Debian-Facile vous remercie,
Et espère que vous avez réalisé un bon voyage. 😎

1)
N'hésitez pas à y faire part de vos remarques, succès, améliorations ou échecs !
2)
(fr.wikipedia) ASCII : Caractères de contrôle.
3)
(fr.wikipedia) Séquence d'échappement ANSI, (manpagesfr.free) man 4 console_codes
4)
(wikipedia)(en) IBMs 1050
5)
TTY est l'abréviation du nom **T**élé**ty**pe, un téléimprimeur
6)
L'IMB 1050 était connecté à son unité de contrôle 1051 : (terminal-wiki)(en) IBM 1050.
7)
(wikipedia)(en) IBM 2250
8)
(fr.wikipedia) Caractères semi-graphiques
9)
(vt100)(en) DEC animation
10)

(fr.wikipedia) Art ASCII
11)

Programme d'art ASCII **cowsay** (la vache dit)
12)

Un Teletype (en.wikipedia)
13)

(wikipedia)(en) Teletype Corporation
14)

(fr.wikipedia) Western Electric Company
15)

(fr.wikipedia) AT&T
16)

(fr.wikipedia) Les Établissements Édouard-Belin
17)

(fr.wikipedia) UART
18)

(fr.wikipedia) Transmission série
19)

(fr.wikipedia) Transmission parallèle
20)

Anneau de protection (fr)(wikipedia)
21)

L'émulation du terminal VT340 permet à **xterm** d'être un terminal graphique.
22)

VNC est une suite logiciel historique, de type client/serveur, qui permet l'usage d'une console graphique connectée à système distant. Il a alors été développé un protocole réseau, qui établit, la communication entre la console et la machine distante, par un flux graphique. (wikipedia)(fr) Virtual_Network_Computing
23)

SPICE est projet qui fourni une console graphique permettant d'accéder à une machine virtuelle. La communication entre la console et la machine virtuelle est réalisée par un flux vidéo. (wikipedia)(fr) SPICE (protocol) - (spice-space)(en)
24)

L'utilisateur **root** est l'administrateur de tout le système. Il a tout les droits sur tout. Il est aussi l'utilisateur qui exécute les programmes nécessaires au fondement du système.
25)

La commande **su** permet de changer d'utilisateur, si nous en connaissons le mot de passe.
26)

Termcap est une base de données (obsolète) des capacités des terminaux. (manpages.org)(fr) termcap (5)
27)

Terminfo est la base de données des capacités des terminaux, actuellement utilisée par les système GNU/Linux. (manpages.org)(en) terminfo (5)
28)

Ncurses (« New curses » : « Nouvelles malédictions ») est une bibliothèque de programmation qui offre des routines indépendantes des terminaux. Elle permet de réaliser des interfaces avec des fenêtres, de créer des menus déroulants, d'utiliser des couleurs, etc, sur des terminaux texte. En affranchissant les différentes syntaxes des séquences d'échappement. Évidement **ncurses** ne fait pas de miracle et ne peut coloriser du texte sur des terminaux nono-chrome. Ainsi **ncurses** permet d'obtenir le meilleur rendu d'affichage pour chaque terminal. (wiki.debian.org)(fr) Ncurses, (linuxfocus.org)(fr) Introduction à Ncurses.
29)

(nampage.org)(en) toe (1) : Table of terminfo entries
30)

(manpages.org)(fr) wc (1) : Afficher le nombre de lignes, de mots et d'octets d'un fichier

31)

(manpages.org)(en) tic (1) : The terminfo entry-description compiler

32)

(manpages.org)(en) term (5) : Format of compiled term file.

33)

(manpages.org)(fr) stty (1) : Modifier et afficher la configuration de la ligne de terminal