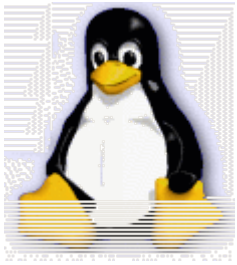
COMMUNICATION	GESTION DU PORT SERIE SOUS LINUX	COURS



1. Gestion de fichiers

1.1 Préambule

Linux enregistre ses données dans un système de fichiers, sur le disque.

Qu'est-ce qu'un système de fichiers ?

Par système de fichiers il faut entendre l'art et la manière dont les données sont gérées par le système d'exploitation. Linux offre un système de fichiers hiérarchique grâce auquel il est facile de répartir entre les différents utilisateurs l'espace disponible sur le disque.

1.2 Types de fichiers

La base du système de fichiers est le fichier individuel (file). On distingue plusieurs types de fichiers :

- Les fichiers ordinaires → **ordinary files**,
- Les répertoires → **directory files**,
- Les fichiers spéciaux → **special files ou devices**.

1.2.1 Les fichiers ordinaires

Ces fichiers contiennent soit du texte en clair (courrier, fichier source de programme, tableaux, …) ou un programme exécutable. Dans ce dernier cas on parle également de fichier binaires (binary files).

1.2.2 Les répertoires

Les répertoires permettent d'organiser l'espace du disque. Les fichiers normaux sont placés dans des répertoires et regroupés. Ces répertoires peuvent eux mêmes contenir des sous-répertoires et des fichiers.

1.2.3 Les fichiers spéciaux

Ces fichiers représentent les interfaces avec les périphériques gérés par le système d'exploitation. Chaque accès en lecture ou en écriture sur un périphérique est directement dirigé vers ce périphérique. Ce type de fichier est utilisé par le port série.

Qu'est-ce qu'un fichier de périphérique ?

Même les périphériques sont très souvent représentés (gérés) par des fichiers sous UNIX.
Exemple : monter un lecteur CD-ROM comme un fichier :

```
$mount –t iso9660 /dev/hdc/mnt/cd_rom
$ cd /mnt/cd_rom
```

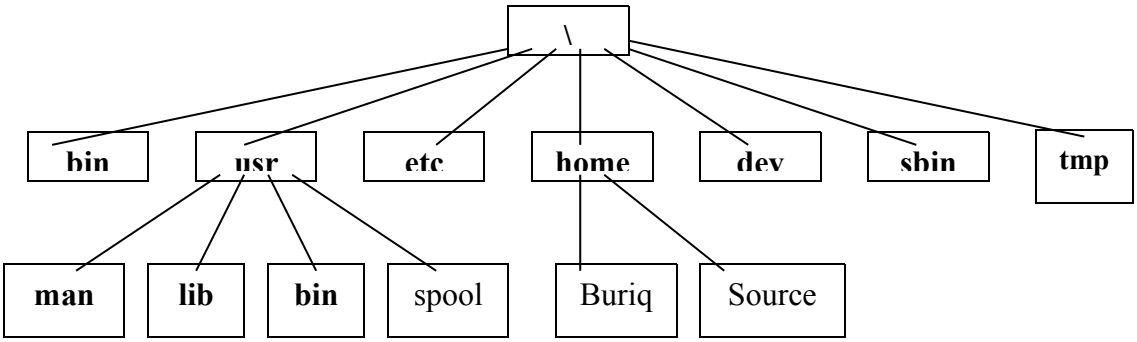
qui prend le fichier de périphérique du CD-ROM (chargé sous le nom hdc au cours du démarrage) et monte son contenu actuel comme structure de fichier sous /mnt/cd_rom.

Les principaux fichiers de périphériques, ont peut citer 3 importants :

- /dev/console
- /dev/tty
- /dev/null

1.3 Arborescence standard de Linux

Dans tous les systèmes Linux, certains programmes ou fonctions doivent toujours être disponibles. En général, ces programmes sont toujours situés au même endroit de l'arborescence. Néanmoins il y a quelques petites variantes.
Cette arborescence reprend celle d'Unix :



Dans tous les systèmes Linux, certains programmes ou fonctions doivent toujours êtres disponibles. En général, ces programmes sont toujours situés au même endroit de l'arborescence. Voici une description des répertoires principaux :

Répertoire	Signification
/	Répertoire racine. Il contient directement ou indirectement, tous les autres répertoires
/bin	Contient des commandes employés par tous les utilisateurs : ls, rm, cp, …
/usr	De tous temps ce répertoire a été surchargé de différentes choses ayant de près ou de loin un rapport avec l'utilisateur. On y trouve des utilitaires, des bibliothèques, des BAL, …
/usr/bin	Les commandes les moins utilisées se trouvent dans ce dossier. On y trouve les commandes pour la communication entre les utilisateurs.
/usr/man	Contient le manuel en ligne.
/usr/spool	Contient les fichiers intermédiaires et de configuration pour la gestion de l'impression et des programmes à lancer régulièrement.
/usr/src	Contient le code source des programmes Linux.
/etc	Contient les commandes et les fichiers nécessaires à l'administrateur du système On y trouve les fichiers passwd, group, inittab et le dossier rc.d contenant les fichiers de démarrage du système.
/sbin	Linux stocke dans ce répertoire les commandes pour la gestion du système, notamment **adduser** permettant d'ajouter de nouveaux utilisateurs.
/tmp	Certaines commandes génèrent des données temporaires, ces fichiers sont ensuite supprimées lorsque la commande est terminée.
/dev	Le répertoire **dev** contient un fichier de périphérique pour chaque périphérique et composant matériel du système. Exemple : • console → console, écran principal) • tty → terminaux, interface série) • lp → connexion imprimante • fichier de périphérique pour les disque durs : hd (IDE, EIDE), sd (SCSI) • Lecteur de disquette : fd
/home	Les répertoires personnels sont généralement placés dans ce répertoire.

Exemple : **Interface série : /dev/ttyS0 → port COM 1**

2. Notion de terminal

2.1 Définition

Le terme de terminaux désigne des entités qui vont de la console de l'ordinateur sur lequel tout utilisateur travaille, à la fenêtre xterm, en passant par la liaison série où une personne peut se connecter à un autre ordinateur via un modem.

2.2 Accès

L'accès aux périphériques est effectué via des fichiers spéciaux. Un fichier spécial, exemple : /dev/ttyS1 apparaît dans l'arborescence de la même façon qu'un fichier ordinaire, mais aucun espace disque ne lui est alloué. A chaque fichier spécial correspond un pilote de périphérique dont le code est intégré au noyau.
Le noyau gère le terminal de manière à offrir une interface homogène aux processus.
La programmation d'un port série utilisera donc les appels systèmes de gestion de fichier :
open (), read (), write (), close ().
Une fois qu'un processus a ouvert un fichier spécial, ses requêtes de lecture et d'écriture ne sont pas transmises au système de fichier mais au pilote du périphérique correspondant.
Pour Unix, le terminal est défini comme un canal de communication associé à une discipline de ligne → structure termios, qui détermine le comportement de ce canal. Le noyau incorpore aussi deux buffers pour conserver les touches appuyées afin qu'elles soient lues par le programme et pour enregistrer les caractères envoyés jusqu'à ce que le terminal soit prêt à les afficher.

2.3 Les différents modes → canonique et non canonique

Un terminal peut être utilisé dans deux modes nommés traditionnellement :

- canonique : c'est le mode par défaut, les saisies sont transmises au processus ligne par ligne. Ce qui signifie que tant que l'utilisateur n'a pas appuyé sur « Entrée », le pilote de terminal n'envoie rien au processus qui effectue la lecture. De plus avec certaines fonctions de saisie, comme fgets, l'utilisateur pourra revenir en arrière et corriger sa ligne avant de valider.
- Non canonique ou brut (raw) : le terminal transmet immédiatement les caractères sur lesquels on appuie. Ce mode permet de construire des applications permettant de gérer les touches fléchées, les sauts de page, …

2.4 Consoles virtuelles **/dev/tty1 → /dev/tty63**

Les consoles virtuelles sont utilisées lorsqu'un utilisateur se connecte physiquement sur la machine. Habituellement le fichier de configuration **/etc/inittab** permet à 6 consoles virtuelles d'être accessibles par les touches ALT + F1 à ALT + F6.

2.5 Speudo terminal : **/dev/ptyp0 → /dev/ptysf, /dev/ttyp0 → /dev/ttysf**

Un speudo-terminal est un périphérique géré par le noyau. Celui-ci est composé d'un speudo-terminal maître associé à un /dev/pty.. et d'un pseudo-terminal esclave associé à un /dev/tty…

Tout ce qu'on écrit sur le speudo-terminal maître est lisible sur le pseudo-terminal esclave est immédiatement accessible du côté maître.
Les pseudo-terminaux s'utilisent exactement comme les fichiers normaux, c'est-à-dire avec les appels systèmes open (), read (), write (), et close (). Toutefois, speudo-terminal maître ne peut être ouvert que par un seul processus.

Les pseudo-terminaux sont principalement utilisés par les serveurs réseau et les systèmes de fenêtrage comme X-Windows.

3. Les Entrées Sorties d'un terminal

Lorsqu'un programme est appelé à partir de l'invite de commande, le Shell s'assure que les flux d'entrée-sortie standards sont reliés au programme afin de pouvoir exécuter cette application.
Quel sont les Entrées Sorties standards ?

- Clavier, → sdtin
- Ecran, → stdout
- Error, → sterr

Qu'est-ce que la redirection ?
Il arrive fréquemment qu'un programme Unix, même interactif, de rediriger les entrés-sorties vers un fichier ou un autre programme.
Exemple : ps –ef > file_process

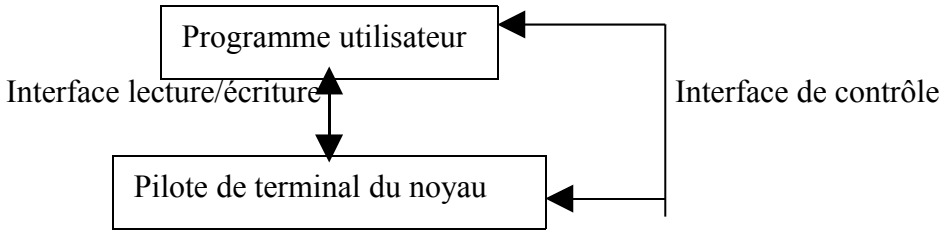
4. Pilote de terminal et GTI

4.1 Préambule

Un programme a parfois besoin d'un contrôle sur le terminal encore plus précis que ce que les opérations simples sur les fichiers permettent. Unix propose un ensemble d'interfaces autorisant le contrôle du pilote du terminal, ce qui offre un contrôle accru du traitement des E/S.

4.2 Qu'est-ce que **GTI** ? → **General Terminal Interface**

Nous pouvons utiliser une série d'appels de fonctions nommée GTI pour contrôler le terminal. GTI est différente des fonctions utilisées pour la lecture ou l'écriture. Cela permet de conserver inchangée l'interface de données (lecture/écriture) tout en offrant un contrôle détaillé du comportement du terminal. Ce qui ne veut pas dire que l'interface d'E/S du terminal soit simple : elle doit gérer une multitude de périphériques différents.

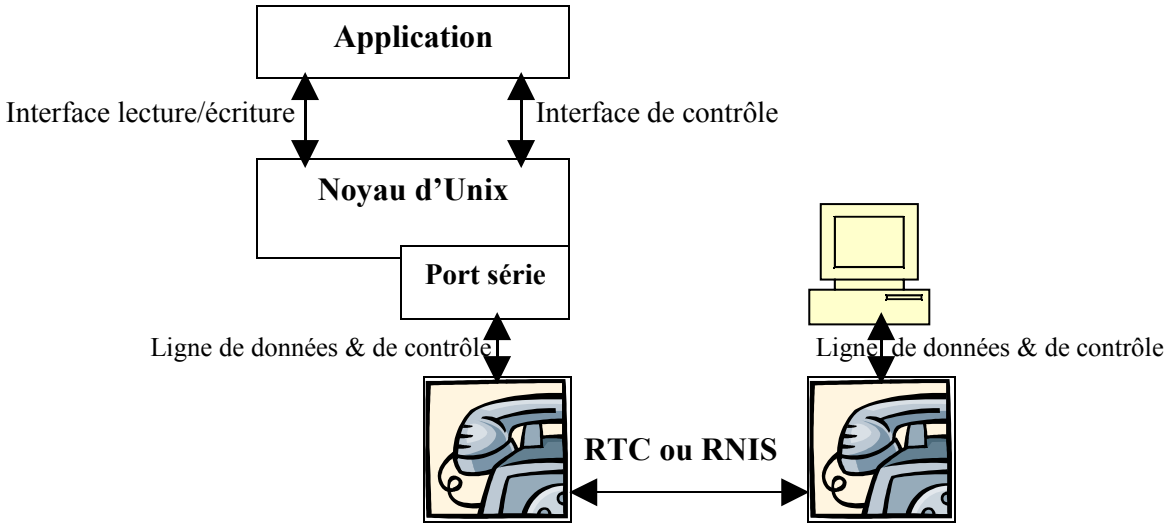


Que contrôler ?
Selon la terminologie d'Unix, l'interface de contrôle définit une ligne de conduite (line discipline). Celle-ci procure à un programme une flexibilité remarquable du comportement du terminal.
Les principales caractéristiques qu'il est possible de contrôler sont :

Caractéristiques	Description
Edition de ligne	RetArr autorisé
Tampon	Caractères lus immédiatement ou seulement après un délai paramétrable
Echo	Contrôle de l'écho, exemple : lecture d'un mot de passe
CR/CF	Comportement associé à \n en E/S
Vitesse de ligne	Peu utile sur une console PC, mais capitale dans le cas d'un modem ou d'un terminal connecté à un port série.

4.3 Modèle matériel

L'architecture type (utilisé par certains sites Unix) consiste à avoir une machine Unix connectée via un port série à un modem, puis par l'intermédiaire d'une ligne téléphonique et d'un autre modem à un terminal distant. Configuration utilisée par certains provider !



5. Les ports série avec caractéristiques

5.1 Adresse et IRQ

Un port d'E/S permet de recevoir des données depuis, et d'en envoyer à partir d'un ordinateur. Il existe plusieurs types de ports d'E/S, comme les ports série, les ports parallèles, les contrôleurs de disques durs, les cartes Ethernet, etc. Les modems et les terminaux dont nous allons traiter n'utilisent que des ports série. Chacun de ces ports possède une adresse d'E/S, et un vecteur d'interruption (IRQ). Les quatre ports série suivants correspondent à COM1 - COM4 :

```
ttyS0 (COM1) adresse 0x3f8 IRQ 4
ttyS1 (COM2) adresse 0x2f8 IRQ 3
ttyS2 (COM3) adresse 0x3e8 IRQ 4
ttyS3 (COM4) adresse 0x2e8 IRQ 3
```

5.2 Fichier de configuration

Linux stocke les configurations des ports dans un fichier de configuration, suivant le cas :

- serial
- serieporter

6. Programmation du port série → La structure Termios

6.1 Introduction

Afin de gérer la liaison série par un programme écrit en langage C il faut utiliser la structure Termios. Termios est l'interface standard définie par POSIX. Elle est similaire à l'interface de System V, termio. L'interfaçage est contrôlée en définissant des options dans une structure de type termios, ainsi qu'en ayant recours à un petit nombre de fonctions.
Il faut inclure le fichier : **termios.h**

Nota : les programmes utilisant les fonctions définies dans termios.h doivent être liées à la bibliothèque idoine, en général, la bibliothèque curses ou lncurses.
Pour Linux, ce sera respectivement ncurses et –lncurses.

Compilation avec l'option –lcurses à la fin de la ligne de commande.

$gcc –o comm comm.c –lcurses ou $ gcc –I/usr/include/ncurses –o comm comm.c -lncurses

6.2 La structure termios (POSIX) → #include termios.h

```
struct termios {
        tcflag_t        c_iflag ;       // Modes d'entrée
        tcflag_t        c_oflag ;       // Modes de sortie
        tcflag_t c_cflag ;              // Modes de contrôle
        tcflag_t c_lflag ;              // Modes locaux
        cc_t            c_cc[NNNCCS] ;  // Caractères de contrôle
        } ;
```

Le type tcflag_t est considéré comme un tableau de bits. On peut donc tester le positionnement d'un indicateur par un & (conjonction binaire) avec sa macro-définition.
Par exemple :
`(c_iflag & IXOFF )` est vrai si le drapeau est positionné.
Il est possible d'initialiser une structure termios pour le terminal en appelant la fonction tcgetattr, dont le prototype est :

```
int tcgetattr (int fd, struct termios *termios_p) ;
```

Cet appel écrit les valeurs courantes des variables de l'interface du terminal dans la structure référencée par termios_p. Si ces valeurs seraient par la suite modifiées, il est alors possible de leur redonner leurs valeurs initiales à la fonction tcsetattr () :

```
int tcsetattr (int fd, int actions, const struct termios *termios_p) ;
```

Le champ actions de tcsetattr contrôle la mise en œuvre des modifications. Il existe 3 valeurs :

Valeurs	Description
TCSANOW	Modifie immédiatement les valeurs
TCSADRAIN	Modifie les valeurs après la fin de la sortie en cours
TCSAFLUSH	Modifie les valeurs après le fin de la sortie en cours, mais élimine toute entrée actuellement disponible n'ayant pas encore été renvoyée par un appel read

6.3 Les différents modes

Les options pouvant être manipulées afin d'affecter le terminal se regroupent en 5 modes.

6.3.1 Modes d'entrées

Les modes d'Entrée contrôlent la manière dont les caractères reçus (clavier ou port série) sont traités par le pilote de périphérique avant d'être traités par le pilote de périphérique.
Voici les macros utilisables pour c_iflag :

Macro	**Description**
BRKINT	Provoque l'émission du signal SIGINT lors d'une réception du caractère *break.* Les tampons d'entrée et de sortie sont vidés.
IGNBRK	Ignore toute condition *break* présente sur la ligne.
ICRNL	Convertit les retours chariots en saut de ligne.
IGNCR	Ignore tout retour de chariot reçu.
INLCR	Convertit les sauts de ligne en retour chariot.
IGNPAR	Ignore les caractères présentant des erreurs de parité.
INPCK	Réalise un test de parité lors de la réception de caractères.
PARMRK	Marque les erreurs de parité.
ISTRIP	Ramène à sept bits tout caractère reçu.
IXOFF	Active le contrôle de flux logiciel XON/XOFF sur les entrées. Les caractères START et STOP sont envoyés automatiquement par le système pour éviter des pertes de données.
IXON	Active le contrôle de flux logiciel XON/XOFF sur les sorties. Si le caractère STOP est reçu, le flux de données est suspendu jusqu'à ce que le caractère START soit reçu.

Nota : SI ni BRKINT ni IGNBRK ne sont activés, une condition break présente sur la ligne sera lue comme un caractère NULL (0x00).

6.3.2 Modes de sortie

Ces modes contrôlent la façon dont les caractères de sorties du buffer sont traités avant d'être transmis au port série ou à l'écran. Contrairement au champ précédent, il n'y a ici qu'un seul attribut définit par POSIX → **OPOST**, tous les autres étant des extensions.
OPOST sert à autoriser l'action des autres attributs du champ **c_oflag.**
Voici les macros utilisables pour c_oflag :

Macro	Description
ONLCR	Convertit tout saut de ligne en retour chariot/saut de ligne (CR/NL).
OCRNL	Convertit tout retour chariot en saut de ligne.
ONOCR	Pas de sortie retour chariot en colonne 0.
ONLRET	Un saut de ligne accomplit également un retour chariot.
OFILL	Envoyer des caractères de remplissage pour constituer un temps de latence.
OFDEL	Utiliser DEL comme caractère de remplissage à la place de NULL.
NLDLY	Sélection du saut de ligne pour les retards.
CRDLY	Sélection du retour chariot pour les retards.
TABDLY	Sélection du caractère TAB pour les retards.
BSDLY	Sélection du caractère BACKSPACE pour les retards
VTDLY	Sélection du caractère de tabulation verticale pour les retards
FFDLY	Sélection du caractère saut de page pour les retards

Nota : Si OPOST n'est pas activé, tous les autres drapeaux sont ignorés.

6.3.3 Modes de contrôles

Ces modes contrôlent les caractéristiques de la ligne série (baud rate, data bits, parity, stop bits et contrôle hardware du flux) entre l'ordinateur et le terminal. Dans le cas où le terminal n'est pas réellement connecté par une liaison série (consoles virtuelles, pseudo-terminaux), ce champ est ignoré.
Voici les macros utilisables pour *c_cflag :*

Macro	Description
CLOCAL	Ignore toute ligne de statut du modem .
CREAD	Autorise la réception de caractères. Si ce bit n'est pas levé, aucun caractère n'est reçu. Si des caractères sont néanmoins envoyés, ils sont perdus. Cela permet, par exemple, de bloquer le fonctionnement d'un clavier de terminal.
CBAUD	Elimine le choix précédent du baud rate (non définie en Posix1.)
Bxxxxxx	B110, B300, B1200, B2400, B4800, B9600, B19200, B38400, B57600, 115200, B230400, B460800
CSIZE	Elimine le choix précédent du nombre de bits de donnée .
CSx	Utilise 5 bits (CS5), 6 bits(CS6), 7 bits(CS7) ou 8 bits(CS8).
PARENB	Active la génération et la détection de la parité (parité paire).
PARODD	Utilise la parité impaire plutôt que paire.
CSTOPB	Utilise 2 bits d'arrêts par caractère, au lieu d'un.
HUPCL	Raccroche le modem à la fermeture (signal DTR invalidé)
B0	Le signal DTR est invalidé (à coupure de ligne)
CRTSCTS	Active le contrôle de flux hardware (signaux RTS et CTS). Non définie n Posix1. Certains O.S. bloquent le signal RTS à l'état *space*.

Nota : SI HUPCL est activé, lorsque le pilote du terminal détecte que le dernier descripteur de fichier référençant le terminal a été fermé, il utilise les lignes de contrôle du modem pour raccrocher la ligne.

6.3.4 Modes locaux

Ces modes contrôlent diverses caractéristiques d'un terminal.
Voici les macros utilisables pour *c_lflag*

Macro	Description
ECHO	Active l'écho local des caractères saisis.
ECHOE	Accomplit une combinaison *RetArr, Espace, RetArr* à la réception de ERASE.
ECHOK	Efface la ligne à la réception du caractère KILL.
ECHONL	Affiche les caractères saut de ligne.
ICANON	Active le traitement canonique des entrées.
IEXTEN	Active l'implémentation de fonctions spécifiques.
ISIG	Active les signaux SIGINT, SIGQUIT et SIGTSP lorsque les caractères INTR, QUIT et SUSP sont reçus..
NOFLSH	Désactive le nettoyage de la file d'attente.
TOSTOP	Envoie aux processus d'arrière-plan le signal SIGTTOU en cas de tentative d'écriture.

6.3.5 Mode avec caractères de contrôle spéciaux

Il s'agit d'un ensemble de caractères, tel que CTRL-C, ayant certaines conséquences lorsqu'il est saisi par l'utilisateur. Le tableau *c_cc* de la structure *termios* lie les caractères à chacune des fonctions supportées.

Mode canonique → Pour le mode canonique les indices du tableau sont :

Indice	Description
VEOF	Caractère EOF
VEOL	Caractère EOL
VERASE	caractère ERASE
VINTR	caractère INTR
VKILL	caractère KILL
VQUIT	caractère QUIT
VSUSP	caractère SUSP
VSTART	caractère START
VSTOP	caractère STOP

Mode non canonique → Pour le mode non canonique les indices du tableau sont :

Indice	Description
VINTR	caractère INTR
VMIN	valeur MIN
VQUIT	caractère QUIT
VSUSP	caractère SUSP
VTIME	valeur TIME
VSTART	caractère START
VSTOP	caractère STOP

NB : Il existe un certain recoupement dans l'utilisation de ces indices de tableau dans les 2 modes : canonique et non canonique. Pour cette raison, il est capital de ne jamais mélanger les valeurs provenant de ces deux modes.

Description des caractères

- INTR : Impose au pilote de terminal d'envoyer un signal SIGINT aux processus reliés au terminal.
- QUIT : Impose au pilote de terminal d'envoyer un signal SIGQUIT aux processus reliés au terminal.
- ERASE : Impose au pilote de terminal de supprimer le dernier caractère de la ligne.

- KILL : Impose au pilote de terminal de supprimer la totalité de la ligne.
- EOF : Impose au pilote de terminal de transmettre tous les caractères de la ligne à l'application lisant les entrées.
- VSUSP : Impose au pilote de terminal d'envoyer un signal SIGSUSP aux processus connectés au terminal.
- STOP : Agit pour empêcher d'autres sorties vers le terminal. Il sert à supporter le contrôle de flux XON:XOFF et est habituellement affecté au caractère ASCII XOFF (CTRL-S).
- START : Redémarre les sorties après un caractère STOP. C'est souvent le caractère ASCII XON (CTRL-Q).

Valeurs de MIN et de TIME

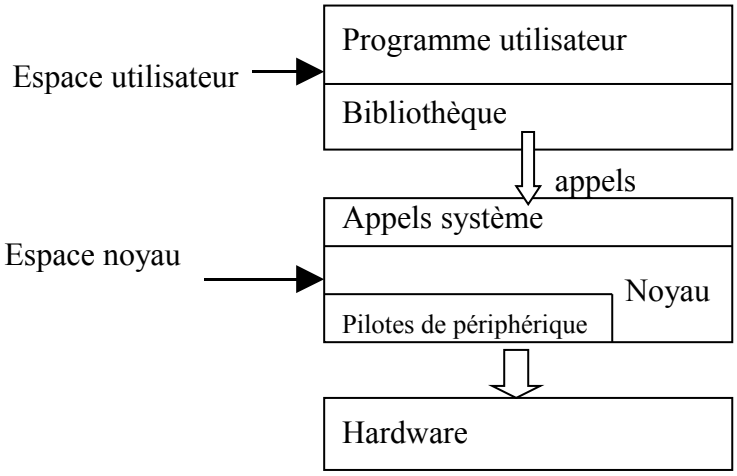
MIN	TIME	Réaction à un read ()
0	0	L'appel read() revient immédiatement. Si des caractères sont disponibles ils sont renvoyés tout de suite.
> 0	0	L'appel read() ne revient que lorsque MIN caractères au moins sont disponibles. Sinon, il peut bloquer indéfiniment.
0	> 0	L'appel read() revient dès qu'un caractère est disponible à la lecture, ou quand TIME dixième de seconde se sont écoulés. Si aucun caractère n'est lu parce que le délai est écoulé, il renvoie zéro. Sinon il renvoie le nombre de caractères lus.
> 0	> 0	L'appel read() ne reviendra que si MIN caractères sont reçus ou si le délai de TIME dixièmes de seconde entre deux caractères est écoulé. Le délai est réinitialisé à l'arrivée de chaque nouveau caractère. Attention ce délai n'est pris en compte qu'après l'arrivée du premier caractère.

Naturellement, dans toutes ces situations l'appel read() peut se terminer avant l'instant indiqué s'il a lu le nombre de caractères demandés en argument ou si un signal l'interrompt.

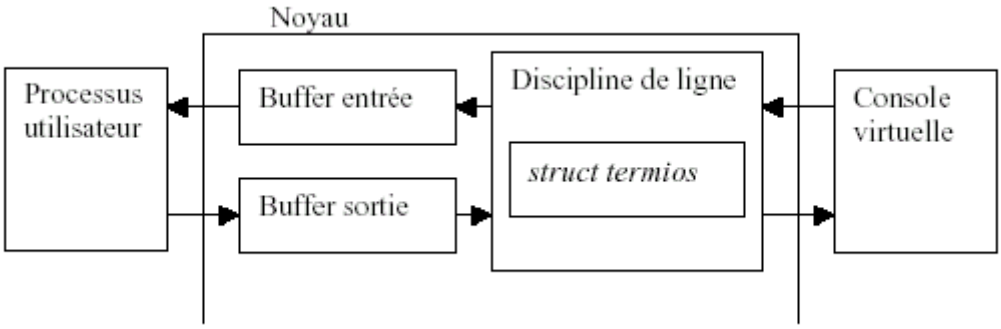
7. Fonctionnement

7.1 Mise en œuvre

Pour résumer le début, voici un schéma du système UNIX présentant l'emplacement des fonctions des différents fichiers relatifs à l'utilisateur, aux pilotes de périphériques, au noyau et au matériel :



7.2 Positionnement de termios



8. Utilisation du terminal à partir du Shell

8.1 Paramètres de termios → stty -a

La commande stty –a permet d'afficher les paramètres en cours de termios correspondant à votre terminal. *Exemple : la commande stty –a* →

speed 9600 baud; rows 0; columns 0; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <indéfini>;
eol2 = <indéfini>; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R; werase = ^W;
lnext = ^V; flush = ^O; min = 1; time = 0;
-parenb -parodd cs8 hupcl -cstopb cread clocal -crtscts
-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl ixon -ixoff
-iuclc -ixany -imaxbel
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

Entre autres choses, on peut voir que le caractère d'interruption est bien CTRL-C et que les options ICANON et ECHO sont activés car n'étant pas précédés du signe moins.

8.2 Lecture clavier

```
/* Name : testtty.c     Author : Edmond Rohrbacher  Date : 21/2/2007 */
/* Lecture de 1 caractère en provenance du terminal                 */
/* Status : opérationnel                                             */

#include <unistd.h>
#include <stdio.h>

int main ()
{
 int iCar;

  printf ("Mode canonique du terminal \n");
  printf ("Entrez un caractère : ");
  fflush (stdout);
  iCar = getchar();
  printf (" Caractère = %c \n", iCar);
  return 0;
}
```

9. Gestion du port RS 232 avec termios

9.1 Configuration de la vitesse

9.1.1 Configuration vitesse

Les fonctions POSIX cfsetispeed () et cfsetospeed () permettent de configurer la vitesse d'entrée ou de sortie d'un port série.

#include <termios.h>

#include <unistd.h>

int cfsetispeed (struct termios *config, speed_t vitesse);
int cfsetospeed (struct termios *config, speed_t vitesse);

Retourne 0 ou –1 en cas d'erreur

Le type speed_t représente la vitesse et peut prendre l'une des valeurs suivantes : B0, B50, B75, B110, B134, B150, B200, B300, B600, B1200, B1800, B2400, B4800, B9600, B19200, B38400, B57600, B115200.
La vitesse B0 sert à forcer le raccrochage du modem.

9.1.2 Lecture configuration vitesse

Pour lire la vitesse définie dans une structure termios, on peut utiliser :

speed_t cfgetispeed (struct termios *config) ;
speed_t cfgetospeed (struct termios *config) ;

9.2 Gestion du buffer

La fonction tcflush () ; permet de vider au choix le buffer d'entrée ou celui de sortie :

int tcflush (int fd, int sel_buffer) ;

Le paramètre sel_buffer peut prendre les valeurs suivantes :

Paramètre	Description
TCIFLUSH	Purge le buffer d'entrée
TCOFLUSH	Purge le buffer de sortie
TCIOFLUSH	Purge les deux buffers

9.3 Contrôle de flux logiciel

La fonction tcflow () permet d'assurer un contrôle software de flux. Cette fonction peut bloquer le buffer de sortie et donc bloquer les tentatives d'écriture qui ont lieu dans un processus écrivain. Lorsque la condition de déblocage se présentera, tcflow () pourra libérer le buffer de sortie et donc relancer l'exécution normale du processus écrivain.

int tcflow (int fd, int action) ;

Paramètre	Description
TCIOFF	Envoie au terminal un caractère STOP qui provoque l'arrêt des transmissions.
TCION	Envoie le caractère START au terminal. Il peut reprendre l'envoi des données.
TCOOFF	On interdit temporairement l'émission des données. Les tentatives d'écriture seront bloquantes.
TCON	On débloque l'émission des données. Les écritures pourront reprendre.

9.4 Gestion de la ligne série

La fonction tcsendbreak () permet d'envoyer un caractère Break sur la ligne série du PC.
Un caractère Break n'est pas vraiment un caractère mais plutôt une succession de bits à 0 pendant une durée supérieure au temps d'émission d'un caractère.

int tcsendbreak (int fd, int duration) ;

Le paramètre duration peut prendre les valeurs suivantes :

- 0 la transmission dure entre 0.25 et 0.5 seconde
- N > 0 la transmission dure entre N*0.25 et N*0.5 seconde.

9.5 Exemple de programme : émission sur port série

```
/* Name : Emis_tty1.c   Author : Edmond Rohrbacher    Date : 21/2/2007  */
/* Emission d'une chaîne de caractères sur le port série               */
/* Status : opérationnel                                               */

#include <fcntl.h>
#include <string.h>
#include <unistd.h>
#include <stdio.h>
#include <termios.h>
#define COM2 "/dev/ttyS1"

int main ()
{
 struct termios newconfig;
 char *message = "Salut Paulette\n";
 char cCar_emis;
 int iFdcom2;
 int i, iTaille_msg;

  /* ouverture non bloquante de COM2 */
  if ((iFdcom2 = open (COM2, O_RDWR | O_NONBLOCK | O_NOCTTY)) < 0)
      perror ("open");
  /* Récupère la config actuelle */
  if (tcgetattr (iFdcom2, &newconfig) < 0)
      perror ("tcgetattr");
  /* Configure la ligne RS 232 COM2 */
  cfsetospeed (&newconfig, B19200); // 19200 Bits/seconde

  newconfig.c_cflag &= ~CSIZE;
  newconfig.c_cflag |= CS7;         // 7 bits de Data
  newconfig.c_cflag |= PARENB;
  newconfig.c_cflag |= PARODD;      // Parité impaire
  newconfig.c_cflag &= CSTOPB;      // 1 stop bit
  /* Modifie la configuration actuelle  */
  if (tcsetattr (iFdcom2, TCSANOW, &newconfig) < 0)
      perror ("tcsetattr");
  iTaille_msg = strlen (message);
  printf ("Message émis : %s \n", message);
  for (i=0; i< iTaille_msg; i++)
     {
     cCar_emis = message[i];
     printf ("%c ", cCar_emis);
     if (write (iFdcom2, &cCar_emis, 1) <0)
          perror ("write");
     }
  close (iFdcom2);
 return 0;
}
```

10. Ressources bibliographiques

10.1Livres et revues

- Programmation Linux au Editions Eyrolles
-

10.2URLographie

en Français → Un article de Pierre Ficheux :
http://www.com1.fr/~pficheux/articles/lmf/serial
en Anglais → Serial programming guide for POSIX operating systems :
http://www.easyw.com/~mike/serial/