11/97

République Algérienne Démocratique et Populaire
Ministère de l'enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique




*Département d'électronique*

**PROJET DE FIN D'ETUDES**

Pour l'obtention du diplôme d'ingénieur d'état
en Electronique

THÈME

# INSTALLATION ET MISE EN ŒUVRE DU SYSTEME LINUX

***Proposé par :***
M^me^ M. BEDDEK
M^r^ R. SADOUN

***Etudié par :***
M^r^ R . BECETTI
M^lle^ S.BOUCHAMA

***Dirigé :***
M^me^ M. BEDDEK
M^r^ R. SADOUN

***Promotion :*** *Octobre 1997*
E.N.P. 10, AVENUE HASSEN BADI - EL-HARRACH - ALGER

République Algérienne Démocratique et Populaire
Ministère de l'enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique




*Département d'électronique*

**PROJET DE FIN D'ETUDES**

Pour l'obtention du diplôme d'ingénieur d'état
en Electronique

THÈME

# INSTALLATION ET MISE EN ŒUVRE DU SYSTEME LINUX

***Proposé par :***
M^me^ M. BEDDEK
M^r^ R. SADOUN

***Etudié par :***
M^r^ R . BECETTI
M^lle^ S.BOUCHAMA

***Dirigé :***
M^me^ M. BEDDEK
M^r^ R. SADOUN

***Promotion :*** *Octobre 1997*
E.N.P. 10, AVENUE HASSEN BADI - EL-HARRACH - ALGER

# Dédicaces



*Je dédie ce modeste travail*

- *A mes chers parents,*
- *A mon frère et mes soeurs,*
- *A mes amis,*
- *A tous mes proches.*

*Samira*

*Je dédie ce modeste travail*

- *A mes chers parents et grands parents,*
- *A mes frères et mes soeurs,*
- *A mes oncles et tantes,*
- *A tous mes cousins et cousines*
- *A tous mes proches et amis.*

*Redha*



# Remerciements

Nous tenons tout d'abord à remercier nos parents pour leur soutien moral et financier.

Nous adressons nos sincères remerciements à nos promoteurs Mme BEDDEK et M. SADOUN pour leur suivi tout au long de ce projet. Nous les remercions encore pour les critiques et suggestions dont ils nous ont faits part et pour les moyens qu'ils ont mis à notre disposition.

Nous exprimons notre profonde gratitude à Mr ABDELLOUEL, Mr FARAH, Mme HAMMAMI et Mr LARBES pour leurs orientations et pour le support qu'ils nous ont apporté sur le plan matériel et en documentation.

Nos remerciements vont également à tous les enseignants qui ont contribué à notre formation, aux travailleurs de la bibliothèque centrale, du CCU, du CERIST, du CDTA et de l'ENSI. Nous remercions en particulier Hacen, Fouzia et Faîza du DATA GENERAL.

Enfin, nous remercions nos frères, nos amis, et toute personne ayant de loin ou de près contribué à réaliser ce mémoire.

## ملخص

"لينوكس " نظام تشغيل سليل "يونكس" لا يخضع لحقوق الطبع .

الهدف من هذه المذكرة هو دراسة النظام " لينوكس " عن طريق دراسة أنظمة التشغيل بصفة عامة، و النظام "يونكس" بصفة خاصة ، تقديم خصوصياته و تطبيقاته من جهة، و تقديم "نواته" ، من جهة أخرى، وذلك بإعطاء أمثلة عن كيفية تشخيصه و تطويره

## Abstract

Linux a version of UNIX is exempted from the copyright . The aim of our work is to present via the study of the operating systems in géneral way and UNIX in particular study, its caracterictics,tools and applications in one side , and in the other side to présent « the kernel » giving exemples of personnalisation ( using the scripts shell) and developpements exemples (using the C language).

## Résumé

Linux est un clone d'UNIX, exempt de droits de reproduction .
Le but de notre travail est de présenter à travers l'étude des systèmes d'exploitation en général et d'UNIX en particulier, ses caractéristiques, ses outils et applications d'une part et d'une autre part, de présenter « le noyau » en donnant des exemples de personnalisation ( en utilisant les scripts shell ) et des exemples de développement ( en utilisant le langage C ).

# SOMMAIRE

# INTRODUCTION GENERALE

**Introduction :**

L'informatique met en oeuvre des ressources importantes et coûteuses, tant en ce qui concerne le matériel que les applications. Un souci d'économie conduit à rendre ces ressources communes à un groupe d'utilisateurs. Cette fonction est remplie par un ensemble de programmes et de dispositifs câblés qui constituent le système d'exploitation.

Les systèmes d'exploitation ont tous le même but: contrôler les activités de l'ordinateur. Ils diffèrent dans la manière d'accomplir leurs tâches et dans les diverses caractéristiques qu'ils proposent. UNIX, le système d'exploitation le plus populaire pour les machines multi - utilisateurs, est unique par sa conception modulaire qui permet aux utilisateurs d'ajuster le système à leurs besoins particuliers.

Le système UNIX comporte plusieurs traits distinctifs. C'est un système:

- multitâche
- multi - utilisateurs
- transportable
- doté d'un choix important de commandes puissantes et de logiciels d'application

Linux, un clone d'UNIX pour ordinateurs personnels, multitâches, multi - utilisateurs, supportant l'interface graphique X Window, le TCP/IP, et beaucoup d'autres applications. Linux a réussit a rendre une nouvelle jeunesse à l'un des plus anciens systèmes d'exploitation, toujours universellement utilisé, UNIX.

Notre travail consiste à étudier le système d'exploitation LINUX. Pour cela nous commençons par donner brièvement les bases théoriques des systèmes d'exploitation, nous insistons particulièrement sur le domaine de communication.

Afin de comprendre le fonctionnement de LINUX , nous citons le cas d'UNIX où nous présentons les points essentiels pour la gestion du système.

La plus grande partie du travail a été consacrée au système LINUX. Dans un aspect utilisation, sont présentées les applications les plus intéressantes sous LINUX tel que

l'éditeur de texte emacs. Nous nous intéressons au protocole réseau TCP/IP dans le but de faire communiquer deux P.C fonctionnant avec des systèmes d'exploitation différents .

Nous présentons un aspect de développement pour montrer la possibilité de personnaliser le système en écrivant ou modifiant des scripts (tels que le .xinitrc et le rc.local), pour expliquer le fonctionnement des appels système (tel que MOUNT) ou pour utiliser le compilateur GCC en développant un programme qui fait la conversion analogique numirique de la carte PA300.

Enfin, Pour les personnes intéressées par le système, nous présentons dans les annexes les étapes à suivre pour l'installation de LINUX et le matériel nécessaire.

Pour illustrer notre travail une annexe contenant des exercices d'administration du système dont parmi eux des problèmes que nous avons eu a résoudre au cours de nos applications.

# Chapitre 1
# Les systèmes d'exploitation

# LES SYSTEMES D'EXPLOITATION:

## 1.1 Définition [1]:

Pour définir le système d'exploitation, nous citons ses deux fonctions principales.

### 1.1.1 Le système d'exploitation en tant que machine virtuelle:

Le système d'exploitation est ce programme qui soustrait le matériel aux regards du programmeur et offre une vue simple et agréable de fichiers nommés qui peuvent être lus et écrits. Le système d'exploitation masque de la même manière beaucoup d'aspects fastidieux en ce qui concerne les interruptions, les horloges, la gestion de la mémoire et les autres tâches de bas niveau.

De ce point de vue, la fonction du système d'exploitation est de présenter à l'utilisateur l'équivalent d'une **machine virtuelle** plus facile à programmer que le matériel.

### 1.1.2 Le système d'exploitation en tant que gestionnaire de ressources:

Le système d'exploitation sert à gérer les différentes fonctions d'un système complexe. Lorsqu'un ordinateur est partagé entre plusieurs utilisateurs, la nécessité de gérer et de protéger la mémoire, les périphériques d'E/S et les autres ressources est encore plus évidente. Le rôle du système d'exploitation, de ce point de vue est de connaître à tout moment l'utilisateur d'une ressource, de gérer les accès à cette ressource, d'en accorder l'usage et d'éviter les conflits d'accès entre les différents programmes ou entre les utilisateurs.

## 1.2 Historique des systèmes d'exploitation[1]:

On distingue quatre générations d'ordinateurs auquelles correspondent quatre types de systèmes d'exploitation de complexité croissante suivant le développement de la technologie des circuits.

### 1.2.1 La première génération

Vers le milieu de l'année 1940, des machines à calculer ont été construite au moyen de tubes électronique. Ces machines étaient énormes, remplissaient les salles avec des centaines de tubes et étaient bien moins rapides que le plus modeste des ordinateurs domestiques contemporains.

A cette époque, un seul groupe de personne concevait, construisait, programmait, utilisait et effectuait la maintenance d'une machine. La programmation se faisait intégralement en langage machine; les systèmes d'exploitation étaient encore inconnus.

### 1.2.2 La deuxième génération

Les systèmes d'exploitation travaillaient selon la technique «.Closed shop » : Il n'était pas possible pour un utilisateur de communiquer avec l'ordinateur de manière interactive (avec les commandes). Les utilisateurs transmettaient leurs paquets de cartes perforées à un opérateur qui seul avait l'accès au calculateur. Les programmes étaient établis sous forme de cartes perforées et en retour l'utilisateur recevait un listing imprimé[2].

Les ordinateurs de la seconde génération étaient surtout utilisés pour les calculs scientifiques et d'ingénierie, par exemple pour résoudre les équations différentielles. Ils étaient pour l'essentiel programmés en FORTRAN ou en assembleur. Les systèmes d'exploitation les plus connus étaient FMS (Fortran Monitor System) et IBSYS, le système d'exploitation d'IBM pour le 7094.

### 1.2.3 La troisième génération

IBM introduit la série d'ordinateurs System/360. Les machines ne différaient qu'en terme de prix et de performance (mémoire maximale, vitesse de processeur, etc.). La série 360 fut la première ligne d'ordinateurs à utiliser des circuits intégrés qui lui ont permis d'offrir un rapport coût/performance bien supérieur à celui des ordinateurs de la deuxième génération qui étaient construits uniquement à partir de transistors.

L'idée de départ était que tout logiciel, y compris le système d'exploitation, devait fonctionner sur les petits et gros systèmes. Il en résulta un système d'exploitation énorme (tel que l'OS/360) très complexe et probablement deux ou trois fois plus important que FMS.

Au début des années 70, le projet MULTICS (Multiplexed information and computing Service) est développé au MIT(Massachusetts institut for technology ) pour une machine General Electric. Le système inclut la pagination[1] et la segmentation [2].

Un autre événement majeur de la troisième génération est le développement phénoménal des mini-ordinateurs qui a débuté avec le DEC PDP-1. Il fut rapidement suivi d'une série d'autre PDP dont le plus performant était le PDP-11.
C'est ainsi qu'on parle de la portabilité des systèmes d'exploitation (voir chap. 2).

### 1.2.4 La quatrième génération

L'époque de l'ordinateur personnel est venue avec le développement des circuits LSI ( Large Scale Intégration ). Deux systèmes d'exploitation ont dominé le marché: MS-DOS écrit par Microsoft.Inc pour l'IBM PC et les machines utilisant le processeur Intel 8088 (et ses successeurs), et UNIX surtout répandu parmi les ordinateurs Motorola 68000.

Depuis le milieu des années 1980, on a assisté au développement des réseaux d'ordinateurs individuels qui fonctionnent sous des **systèmes d'exploitation en réseau** ou sous des **systèmes d'exploitation distribués**. Sous un système d'exploitation sous réseau, les utilisateurs connaissent l'existence des différents ordinateurs, peuvent se connecter sur une machine distante et transférer des fichiers d'une machine à une autre.

Dans un système distribué, les utilisateurs ne doivent pas savoir où se trouvent leurs fichiers et où leurs programmes sont exécutés; cette tâche incombe au système d'exploitation qui doit s'en acquitter de manière transparente pour l'utilisateur.

Le tableau suivant présente les dates approximatives et le materiel correspondant a ces générations [3]:

---

[1] Le concept de mémoire virtuelle paginée peut être vu comme un cache, la mémoire physique constituant un cache pour la mémoire secondaire.
D'une façon générale, on appelle cache tout dispositif matériel ou logiciel qui stocke dans une zone, dont l'accès est rapide, des données en petite quantité choisies parmi des données en grande quantité qui sont, elles, stockées dans une zone dont l'accès est plus lent que celui du cache[3]
Dans le cas de la pagination, la mémoire primaire est la mémoire physique, et la mémoire secondaire est l'espace sur disque dur.
La mémoire physique ne contient, à un instant donné, que les pages du programme utilisées récemment[3].
[2] La segmentation permet un adressage relatif dans l'espace virtuel: l'espace logique est découpé en segments, à l'intérieur desquels les adresses sont relatives au début du segment[3].

| Années | Caractéristiques |
|---|---|
| **44-50** Premiers prototypes | Pas de système d'exploitation |
| **51-57** Première machine commerciale | Superviseur, traitement par lots simplifié |
| **58-64** Machines à transistors | superviseur, premier système a temps partagé |
| **64-78** Machines à circuits intégrés | systèmes multiprogrammés |
| **78- 80** Minis et Micros Ordinateurs | systèmes modernes interactifs,mémoires virtuelle. |
| **80** > réseaux, circuits VLSI, | interfaces graphiques, multifenêtres, . . . |

**Tab. -1.1-** Les générations de matériel

**Remarque** l'idée de traitement par lots (connu pendant la deuxième génération) été de connecter un ensemble de travaux puis de les transférer sur des bandes magnétiques en utilisant un ordinateur peu onéreux.

**La notion de superviseur:**

La notion de superviseur est apparue dés que les périphériques d'entrées-sorties deviennent plus performants, permettant le «parallélisme» CPU-E/S. Le superviseur était capable de lire le travail suivant une bande d'entrée, lui donner le CPU; reprendre le contrôle, mettre les résultats sur bande. L'organisation s'est faite a travers un regroupement des travaux en lots ( sur bandes magnétiques au début) avec des cartes de contrôle très simples et des entrées - sorties standardisées.

## 1.3 Principes des systèmes d'exploitation :

### 1.3.1 Multiprogrammation et temps partagé [9,10] :

L'unes des fonctions des systèmes d'exploitation est l'exécution simultanée sur une même machine d'un grand nombre de programmes différents. Pour pouvoir gérer efficacement plusieurs programmes, le système d'exploitation doit :

- Etre capable de reprendre le contrôle du CPU à tout moment pour répondre à un événement extérieur.
- Posséder un algorithme de choix (l'ordonnancement) permettant de décider à un instant donné lequel parmi les programmes présents doit recevoir le contrôle du CPU.

- Répartir la mémoire physique disponilble entre les différents programmes en concurrences pour fournir un certain nombre de directives permettants à l'opérateur de controler les programmes en cours (détruire des programmes, changer les prioritées,...).
- Permettre à plusieurs programmes de partager des données en mémoire pour optimiser l'utilisation de celle-ci.
- Protéger les données et les programmes en mémoire et sur disques.

### 1.3.2 . Les Processus (tâches) :

Le processus est un concept clé de tous les systèmes d'exploitation . Il correspond à l'exécution d'un programme. Il est représenté par le «descripteur de tâche » qui rassemble les informations essentielles concernant la tâche:

- **Pointeur-suivant :** c'est un pointeur qui permet le chaînage de descripteur de tâche.
- **Contexte:** Sauvegarde les informations qui sont necessaires a la reprise de l'exécution d'une tâche provisoirement suspendue .
- **Etat de la tâche :** Indique l'état courant de la tâche (active , bloquée, . . . ) .
- **Supervision :** Contient les informations concernants la priorité de la tâche et les droit d'accés.

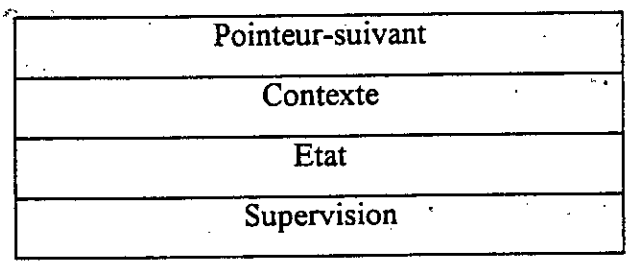



**Figure -1.3.a-** descripteur du processus.

Les processus peuvent être ammenés a agir sur les mêmes variables ou a accéder aux mêmes peripheriques . Un contrôle de ces ressources communes( section critique) est nécessaire afin d'éviter d'importantes erreurs. Ce partage de ressources communes est appelé **« exclusion mutuelle ».**

Afin de résoudre le problème de l' exclusion mutuelle, on utilise les sémaphores qui activent un processus en attente d'une section critique tant que cette dernière est en exécution.

En dehors de l'exclusion mutuelle les processus peuvent être ammenés à coopérer entre eux de façon à s'exécuter dans un ordre prévu. Par exemple un processus **A** ne peut exécuter une instruction **a** que si l'instruction **b** a été exécutée auparavant par le processus **B** .

### 1.3.3 Le système de fichier [4]:

Un fichier est une collection organisée d'informations. Un système de fichier est un ensemble de programmes organisés pour faciliter la constitution et l'emploi de telle collections. Les principales possibilités offertes par ces programmes sont:

- le contrôle de l'implantation des fichiers sur leur support physique;
- la définition de la structure logique des fichiers;
- l'adaptation de cette structure aux contraintes de bonne utilisation.

Un utilisateur connaît ses fichiers par leurs noms. Le catalogue sert à l'établissement du lien entre le nom du fichier et son adressage. Il comporte une entrée par fichier contenant les informations suivantes:

- Le nom du fichier
- Adresse d'implémentation
- Des attributs éventuels (date de création de fichier, type de fichier,...)
- Des informations pour la gestion de l'espace.

### 1.3.4Outils fournis par le système d'exploitation :

L'un des principaux avantages d'un système d'exploitation multiutilisateur est de permettre le partage de certains périphériques. Le système d'exploitation doit fournir aux opérateurs qui ont à les manipuler un certain nombre d'outils :

- Un système de fil d'attente pour imprimantes qui permettent de choisir l'ordre dans lequel sortent les listings.
- Des moyens de surveillance du système lui permettant de se rendre compte si un programme ne prend pas toutes les ressources.
- Des outils de diagnostic en cas de pannes.
- Des outils permettant à l'opérateur d'échanger des messages aves les utilisateurs dont la console peut etre éloignée.

## 1.4 Système de communication [4,5]

Il a pour but de régler l'échange de messages entre deux entités d'un système informatique ou de plusieurs systèmes informatiques organisés en réseau. Chaque partenaire est supposé capable d'interpréter les messages qu'il reçoit et d'engendrer des réponses. Un partenaire peut être un programme, un appareil ou un opérateur humain.

Pour que deux partenaires se comprennent, il faut qu'ils aient un langage commun; cela peut être difficile à obtenir s'ils utilisent des ordinateurs ou des systèmes d'exploitation différents; de plus un troisième partenaire s'immisce entre eux: le système de télécommunication, en général géré par une administration indépendante.

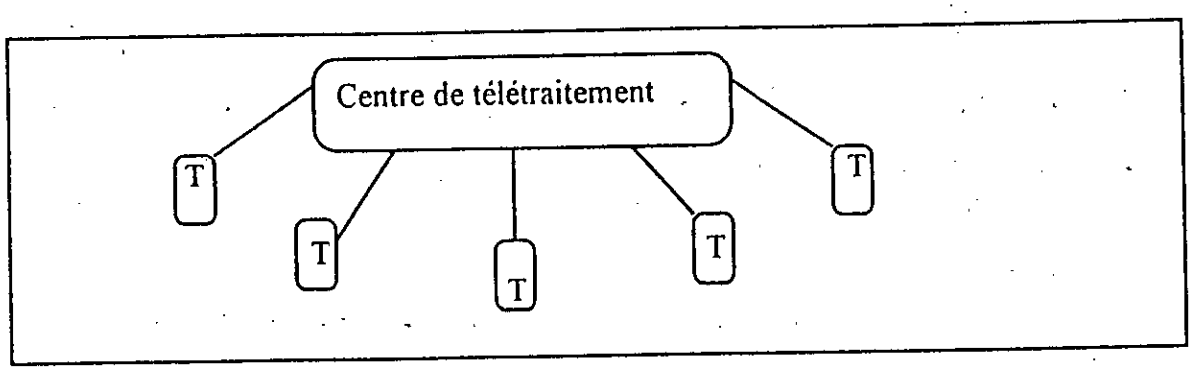



**Figure -1.4.a-** Un système de télé traitement

Dans ce cas c'est l'unité de traitement qui s'occupe de la gestion des procédures de transmission et la gestion des files d'attente. Pour alléger la tâche de l'unité de traitement, des pré-processeurs ou ordinateurs frontaux et des concentrateurs sont apparus un peut plus tard.

Les frontaux permettent d'améliorer les performances de l'unité de traitement et ils peuvent être spécialisés dans la gestion d'un réseau de transmission de données.

**Modèle de référence de l'ISO**

Le système de communication est encore en état d'évolution sous l'impact à la fois des progrès technologiques et des efforts de standardisation. L'ISO (International

Organization for Standardization ) a entrepris la définition d'un modèle OSI (Open System Interconnection) qui fournit un cadre de référence commode pour situer les systèmes existants et les développements futurs.

Le modèle OSI est un modèle en couche, chaque couche rassemblant des machines ou entités chargées d'un groupe de fonctions analogues. Les machines virtuelles d'une même couche peuvent appartenir à des sites différents, et par conséquent n'avoir aucun lien direct entre elles. La fonction primordiale du système est de fournir aux entités de plus haut niveau, les applications, un moyen de communiquer avec d'autres entités indépendamment de leur localisation.

Les conventions d'emploi d'un système distribué se répartissent en *interfaces* et *protocoles*. Une interface régit l'échange d'informations entre une entité de la couche n et l'entité de la couche n-1 à laquelle elle s'adresse lorsqu'elle a besoin de service.
(voir la figure 1.4.b)

Le protocole est un ensemble de conventions permettant à deux entités d'une même couche d'échanger des messages. Un protocole d'échange entre entités de la couche n s'implémente au moyen d'interfaces avec une ou plusieurs de la couche n-1 chargées du transport des informations. Ce processus peut être itéré jusqu'à un niveau où l'échange direct d'informations est possible; c'est en général le plus bas niveau, celui des lignes de télécommunication.

Les diverses couches du modèle de l'ISO sont décrites partant de la base.
(voir la figure 1.4.c)

**1) Lignes physiques:** il s'agit soit de lignes destinées à la transmission de la parole, soit de lignes spécialement adaptées à la transmission d'informations digitales.

**2) Liaison:** Les lignes physiques permettent l'échange de flots de bits. Pour pouvoir les interpréter, il est nécessaire d'assurer la synchronisation, afin de discerner les frontières de caractères et les frontières de message. La couche liaison est responsable de l'acheminement sans erreurs de blocs information sur des liaisons de données.

**3) Réseau:** la fonction essentielle de cette couche est de diriger les messages à travers des liaisons enchaînées depuis l'émetteur jusqu'au destinataire. C'est au niveau de cette couche

qu'est utilisée l'adresse du destinataire, par consultation, à chaque noeud, d'une table de routage.

**4) Transport:** La station de transport à pour but de fournir une interface unifiée entre le réseau et les utilisateurs, et en particulier d'adapter la longueur des messages que ceux ci engendrent à la longueur optimale de transport sur le réseau.

**5) Session:** une session est une liaison entre deux entités adressable du réseau. Une entité peut gérer plusieurs entités simultanées; chacune de ces liaisons lui est connue au moyen au moyen d'un nom local (nom de porte ou de boîte aux lettres). Les sessions sont établies au moyen du protocole de connexion qui permet de réserver les ressources nécessaires (essentiellement de la mémoire pour les tampons d'émission- réception) dans les noeuds traversés.

**6) Présentation:** La couche de présentation traduit chaque message de et vers le langage propre de l'entité connectée. Il s'agit donc essentiellement d'un outil de normalisation.

**7) Application:** La dernière couche est celle des applications. Elle correspond aux tâches du système d'exploitation local.

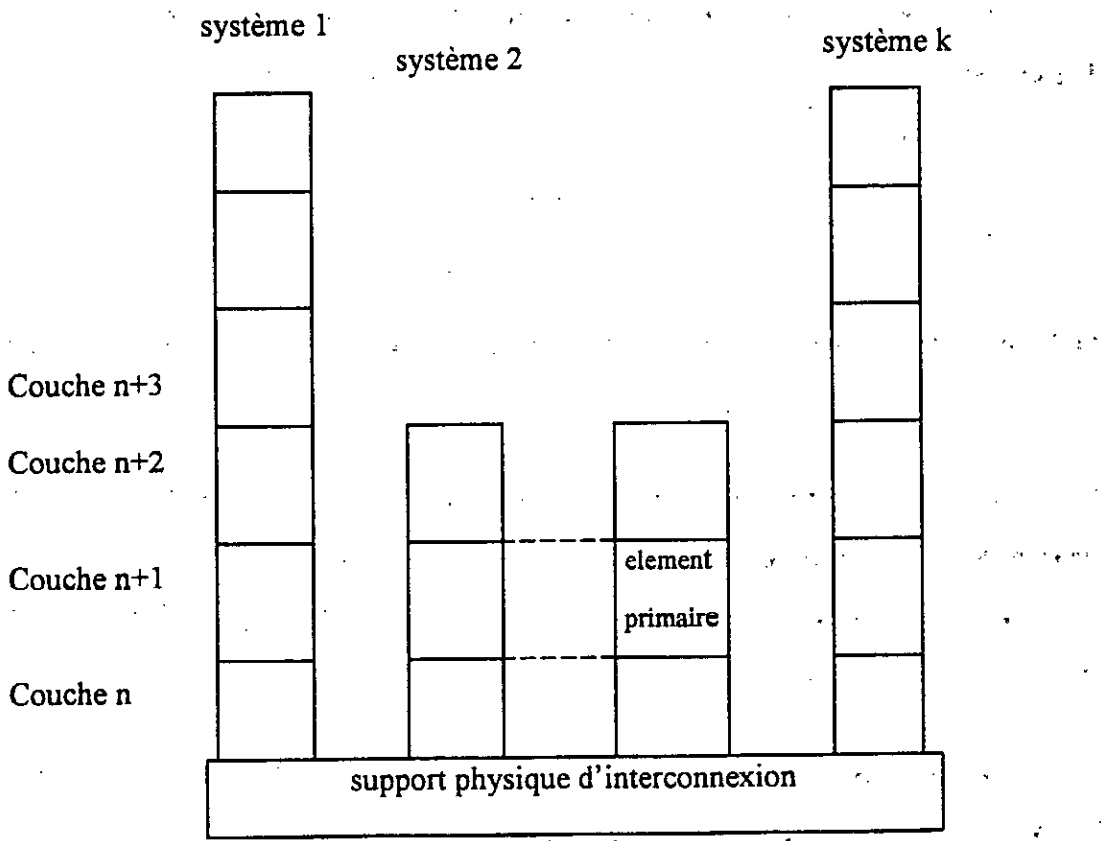



**Figure -1.4.b-** une architechture en couches

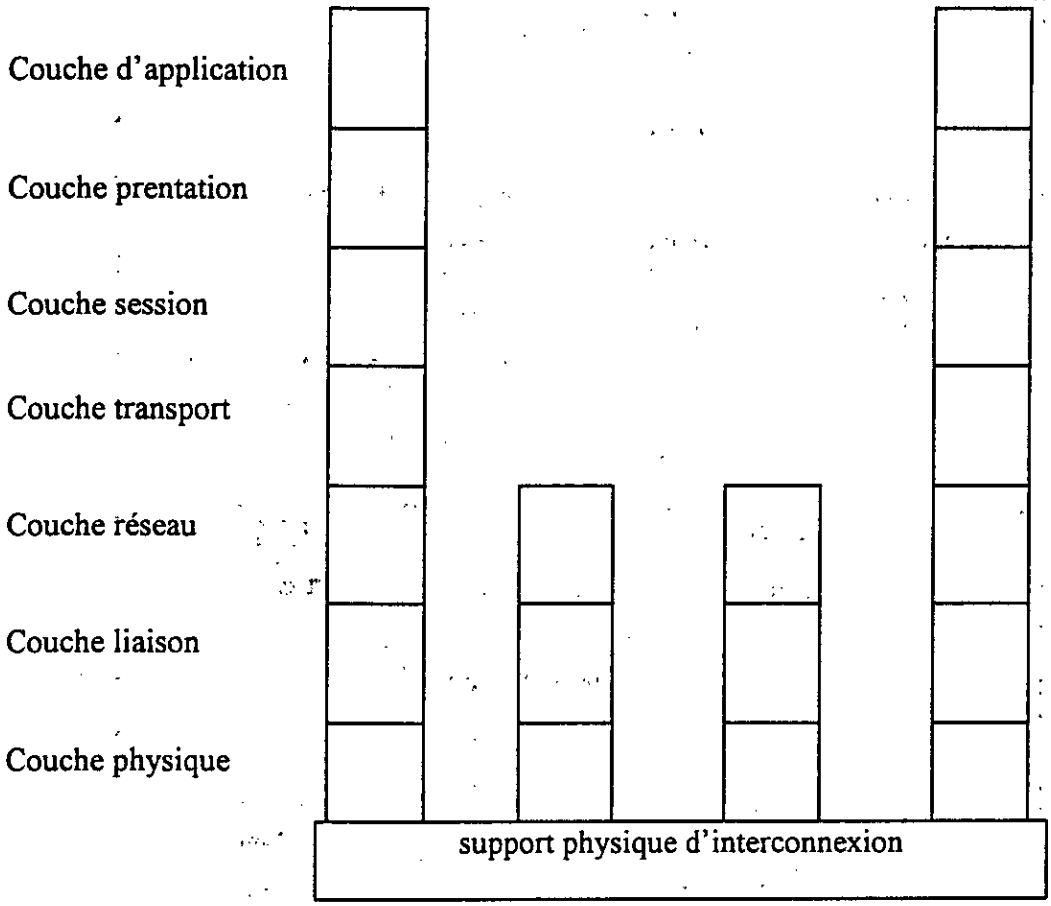



**Figure -1.4.c-** L'architechture OSI

**Topologie**

La topologie désigne la manière dont les stations de travail et les serveurs de fichiers sont reliés au réseau.

Il exite différentes formes géométriques de réseaux. On peut rencontrer des réseaux en forme de ligne (bus), d'étoile, de boucle (ou anneau), d'arbre ou graphe (réseau maillé).

Les réseaux en ligne ou en boucle correspondent le plus souvent aux besoins de connexion de micro-ordinateurs entre eux. Un réseau en étoile se rencontre autour d'un serveur ou d'un concentrateur de consoles. Si le concentrateur était à son tour connecté à un centre de traitement, on obtiendrait un réseau en arbre ( voire les figures -1.4.d- . . . -1.4.g- )

La forme d'un réseau joue sur la complexité de la mise en oeuvre (choix de chemin, identification,...) et sur la fiabilité en cas de panne. Une coupure de ligne ou panne de noeud provoque des perturbations. Seul le réseau maillé résiste bien, car il offre de nombreux chemins entre une paire de noeuds.

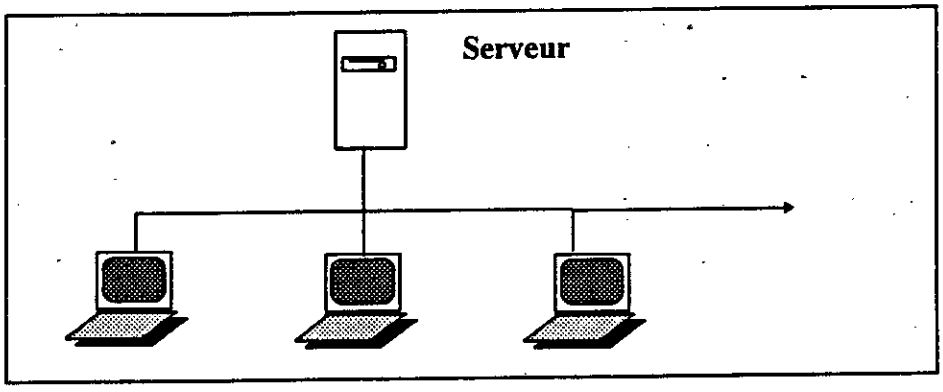



**Figure -1.4.d-** Réseau en bus

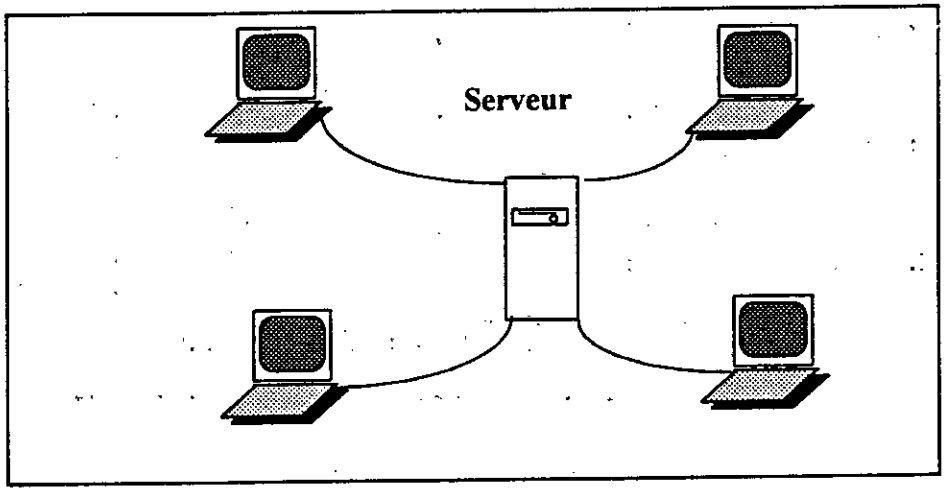



**Figure -1.4.e-** Réseau en étoile

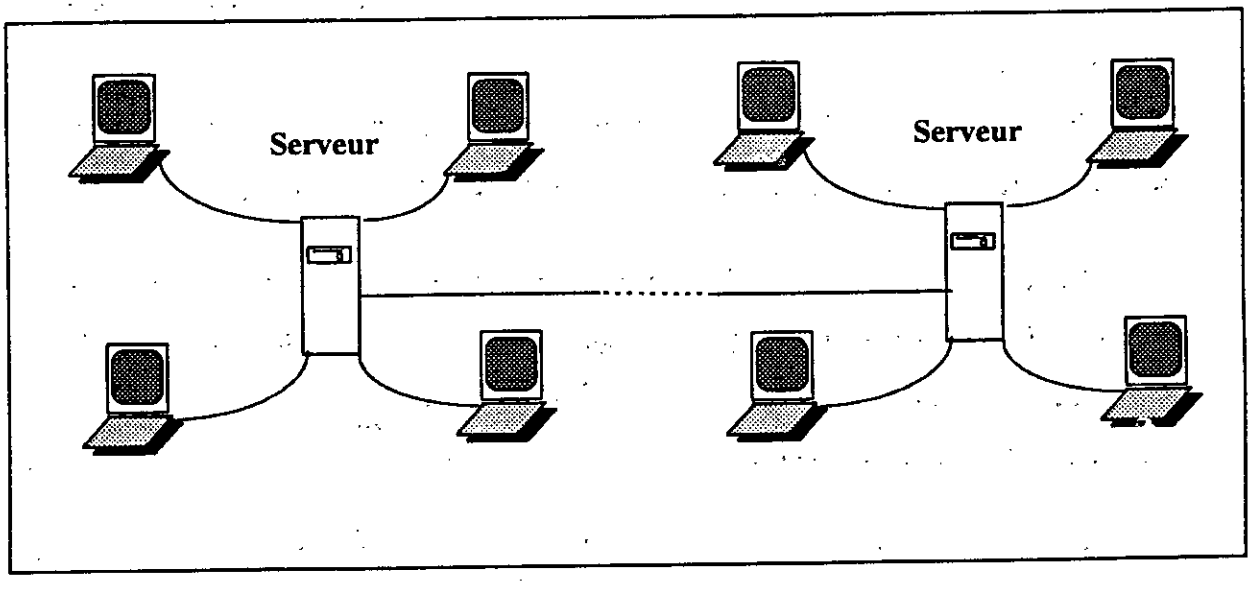



**Figure-1.4.f-** Réseau en arbre

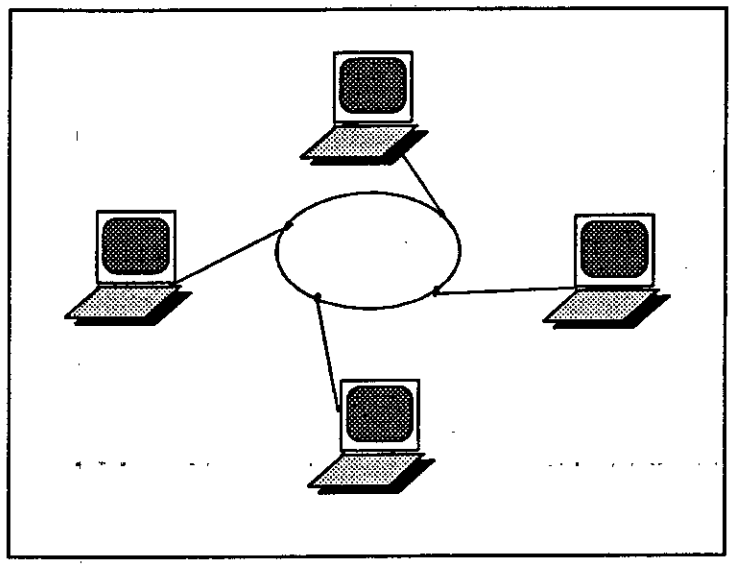


**Figure-1.4.g-** Réseau en anneau

## Conclusion

Etant donné la complexité de l'étude des systèmes nous avons limité les notions théoriques afin que l'exploitation de ce document soit plus aisée.

Pour comprendre le fondement des systèmes d'exploitation ,nous présentons un ensemble de concepts qu'il s'agit de connaître dans le domaine d'UNIX.

# Chapitre2
# UNIX

# 2 LE SYSTEME UNIX

## 2.1 Historique[1,2]:

L'histoire du système d'exploitation UNIX débute en 1969. A cette époque, Ken Thompson et Dennis Ritchie travaillaient dans la société Bell Laboratories, un centre de recherche commun a AT&T (American Téléphone and Télégraphe corporation) et Western Electric. Ils participaient à un vaste projet de programmation en vue de développer le système d'exploitation MULTICS, projet commun à AT&T, Général Electric et MIT.

Les Bell Labs décidèrent d'abandonner MULTICS. Thompson récupéra un PDP-7 de Digital muni d'un écran graphique, et il décida d'écrire une version simplifiée et mono-utilisateur de MULTICS. Brian Kernighan baptisa avec humour ce système UNICS (UNiplexed Information and Computer Service), nom qui fut changé par la suite en UNIX.

En 1973, les utilitaires et la plus grande partie du noyau furent réecrits en C, langage mis au point dans l'intervalle par Ritchie à partir d'un autre langage, moins complet qui s'appelait "B".

Les Laboratoires Bell ont distribué pratiquement gratuitement UNIX aux universités. Il fut rapidement adapté à l'INTERDATA 7/32, aux VAX, et à bien d'autres ordinateurs. UNIX est le système d'exploitation qui a été le plus porté, et son utilisation s'est généralisée rapidement.

**Evolution du système UNIX**

L'évolution du système UNIX est représentée dans la figure-2.1- Cette représentation ne considère pas les versions qui ne sont pas reconus comme commertialiser.

## 2.2 CARACTERISTIQUES DU SYSTEME UNIX :

### 2.2.1 Un système portable[7]:

Un système est portable lorsqu'il peut être transposé sans problème d'un type de plate -forme à un autre. A l'origine, UNIX ne pouvait être installé que sur les mini - ordinateurs de type DEC PDP-7. Aujourd'hui, toutes les versions d'UNIX fonctionnent aussi bien sur des ordinateurs portables que sur des stations de travail.

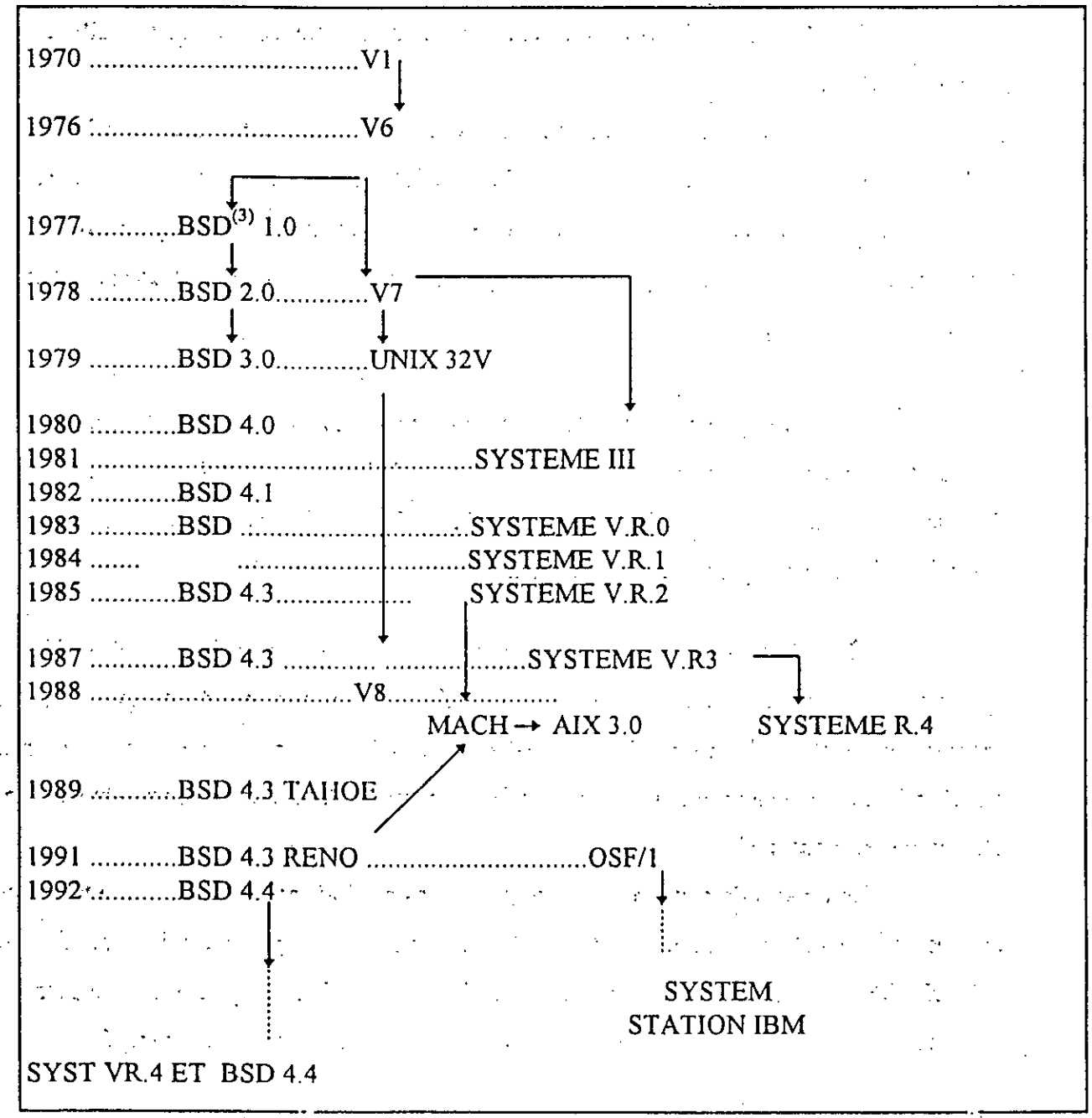



**Figure -2.1-** Evolution du système UNIX

(3) Berkley Software Distribution

### 2.2.2 Un système multitâche et multi-utilisateurs[2]:

Dès le départ, UNIX a été conçu comme un système d'exploitation multitâche et multi-utilisateurs.

Un système d'exploitation multi-utilisateurs est un système permettant à plusieurs utilisateurs de travailler sur la même machine UNIX. Un système d'exploitation est dit multitâche, s'il permet l'exécution de plusieurs programmes en « parallèle ». Ces caractéristiques permettent de gagner du temps au niveau des traitements et également de mieux amortir des logiciels onéreux.

Les développements d'UNIX sont sans conteste à mettre en rapport avec l'état des concepts de système d'exploitation au moment de la définition de ses fondements. A cette époque, le principe de base était de rechercher la meilleure utilisation possible du matériel, car c'était l'élément le plus coûteux du système.

Ce n'est qu'avec la diffusion en masse des PC, que ce principe a évolué. Ainsi le système d'exploitation MSDOS, prévu pour une utilisation mono-poste du PC est appelé S.E mono-utilisateur. De la même façon, MS-DOS ne sait charger en mémoire et exécuter simultanément plusieurs programmes que de façon extrêmement limitée. Il s'agit d'un système d'exploitation mono-tâche.

Des essais de combinaison de ces options à priori inconciliables ont été réalisés par des systèmes d'exploitation du type d'OS/2, de la société IBM ou l'extension graphique de MICROSOFT, WINDOWS. Dans ce cas, la machine ne travaille que pour un. Il va sans dire que les systèmes UNIX pour PC essaient également de prendre pieds sur ce marché.

L'illustration suivante présente quelques systèmes d'exploitation connus, sous l'aspect du multitâche et multi-utilisateurs.

| | mono-utilisateur | multi-utilisateurs |
|---|---|---|
| **mono-tâche** | MS DOS<br>CP/M | |
| **multitâches** | OS/2<br>WINDOWS | WINDOWS NT<br>UNIX VM/CMS<br>BS2000 VMS LINUX |

## 2.3 PHILOSOPHIE ET STRUCTURE DU SYSTEME UNIX :[2]

Pour réaliser une application précise, en adoptant le principe de la modularité, il devient évident de décomposer l'ensemble de taches en sous taches simples chacune est prise en charge par un processus.

La coopération entre ces divers processus aboutit a la réalisation de la tache finale.

On voit apparaître ainsi, dans la figure -2.2-, l'idée d'une chaîne de programmes. Chacun passant son résultat partiel au programme suivant.

Naturellement ,comme il aurait été très inefficace qu'un résultat d'un programme fut passé au programme suivant de la chaîne par l'intermédiaire d'un fichier disque, Thompson et Ritchie eurent l'idée de reprendre une méthode qui à été déjà utilisée sur le système d'exploitation d'origine des PDP; la redirection des E/S, en la généralisant .

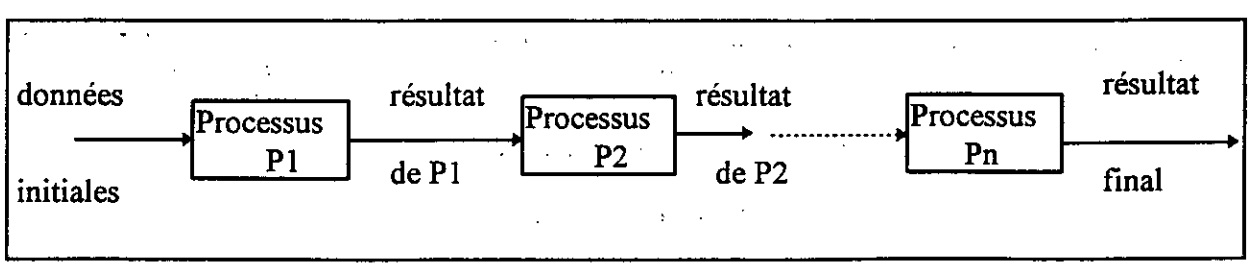



**Figure -2.2--** La logique de programmation sous UNIX.

De cette façon, par une simple commande au système, on peut indiquer que la sortie d'un programme est dirigée sur la sortie d'un autre. Les données sont alors passées du premier programme au second par le système directement de la mémoire à la mémoire, sans passer par un périphérique physique. Cette technique a été appelée "pipe" par les auteurs du système.

Dans une ligne de commande elle est simplement indiquée par le caractère "||", ainsi la commande A|B|C indique que le résultat de la commande ou du programme A doit servir d'entrée au programme B, dont la sortie sera elle-même l'entrée de C. Dans ce cas B est appelé un filtre.

Au début, UNIX a été conçu comme un outil de travail comportant un noyau minimum, la plupart des fonctions classique dévolues à un système d'exploitation étant réalisée par des programmes indépendants donc facilement modifiables. La possibilité de modifier facilement le système a été l'une des principales raisons de son succès dans les universités (outre le fait qu'il était disponible pour une somme dérisoire en comparaison avec les systèmes d'exploitation "officiels" des constructeurs).

## 2.4 Architecture du système UNIX [13]

La structure d'UNIX est constituée de couches concentriques:

**1 le noyau :** lui même est constituer de deux couches :

- Le noyau central ("core") contient les fonctions de base et la gestion de processus; c'est un automate d'affectation de type partagé.
- Le noyau complet contient les conducteurs d'interfaces E/S.

**2 La coquille** ("shell") est l'interpréteur de commandes par lequel l'utilisateur interagit avec UNIX.

**3 Applications** La dernière couche contient les outils et les applications : l'ensemble des programmes et utilitaires qui ont été écrits petit à petit pour améliorer le système.

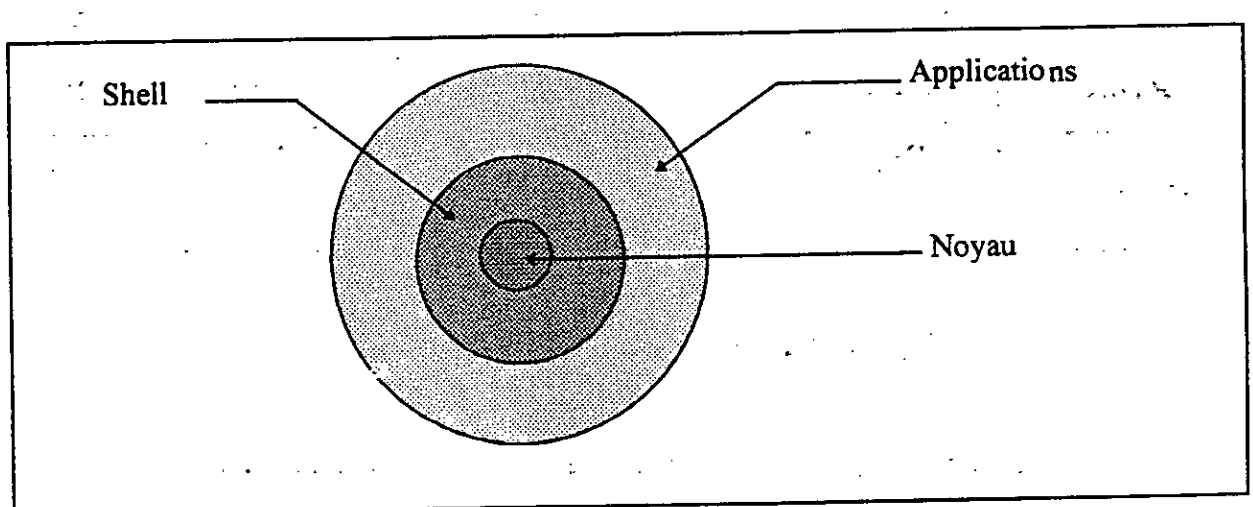



**Figure-2.4-** Architecture d'UNIX

### 2.4.1 Le noyau

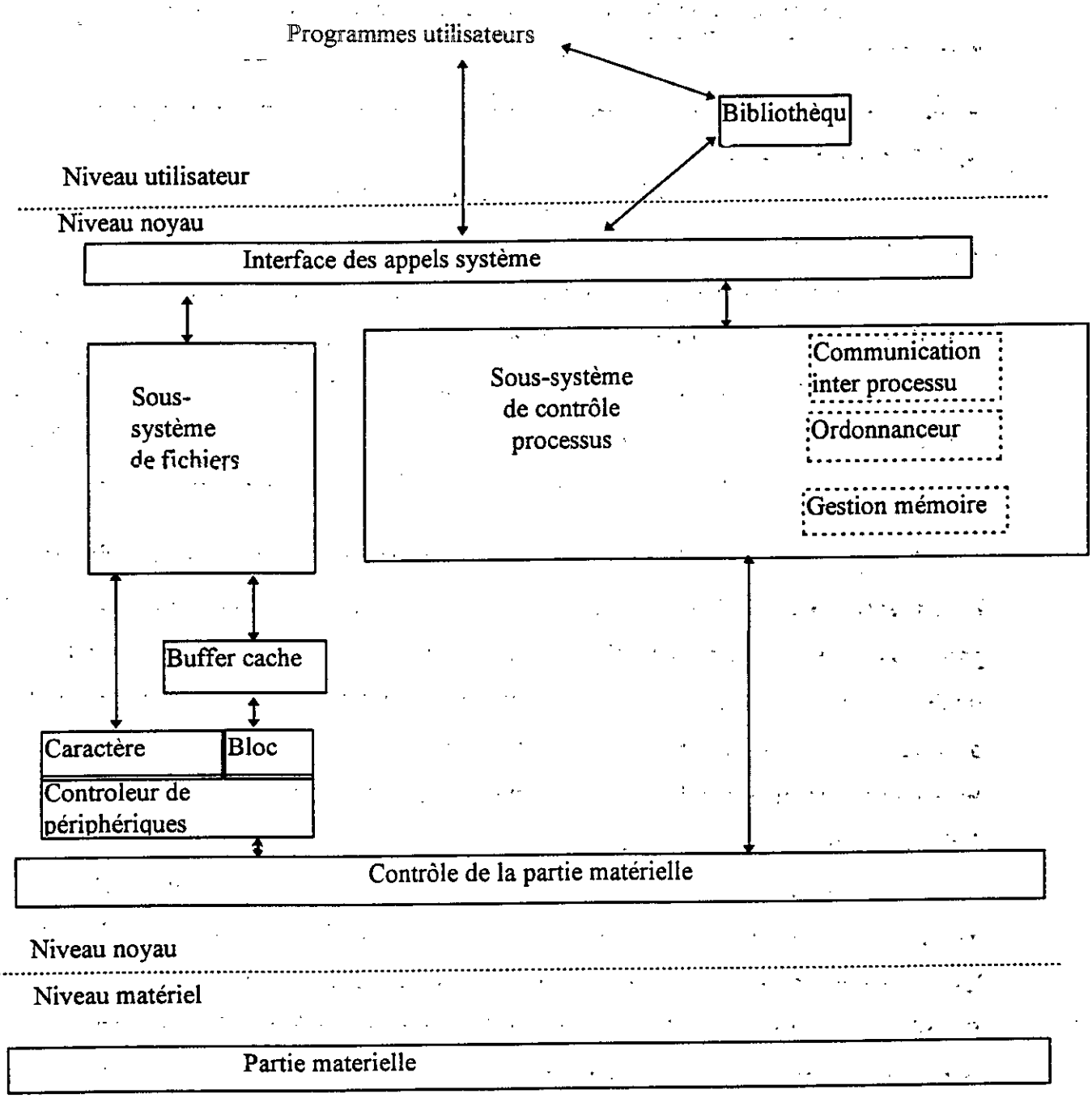
La couche la plus importante du système UNIX est le noyau, il est à considérer comme etant le système d'exploitation.

La figure-2.4.1- présente un bloc diagramme du noyau montant divers module et leurs rapports les uns avec les autres.

L'interface entre les appels système et la bibliothèque est la frontière entre les programmes utilisateurs et le noyau. Les appels système ressemblent aux appels de fonctions ordinaires des programmes C, et les bibliothèques établissent les liens entre ces fonctions primitives nécessaires pour rentrer dans le système. Les programmes en langage assembleur peuvent, cependant, invoquer les appels système directement sans passer par la bibliothèque des appels systèmes. Les programmes utilisent fréquemment d'autres bibliothèques standard des entrées-sorties qui permettent une utilisation plus sophistiquée des appels système.

La figure-2.4.1- répartit l'ensemble des appels système entre ceux qui interagissent avec le sous-système de fichiers et ceux qui interagissent avec le sous-système de contrôle des processus. Le sous-système de fichiers administre les fichiers, allouant de l'espace pour un fichier, administrant l'espace libre, contrôlant l'accès aux fichiers, et extrayant les données pour les utilisateurs. Les processus interagissant avec le sous-système de fichier via un ensemble spécifique d'appel système, tel que open (pour ouvrir un fichier en lecture ou en écriture), close, read, write, stat (consulter les attributs d'un fichier).

Le sous-système de fichiers accède aux données d'un fichier en utilisant un mécanisme de mémorisation dans des tampons qui régule le flux de données entre le noyau et les périphériques de stockage secondaires. Ce mécanisme interagit avec les contrôleurs d'entrées-sorties des périphériques blocs pour instaurer un transfert de données depuis et vers le noyau. Les contrôleurs de périphériques sont les modules du noyau qui contrôlent les opérations sur les périphériques. Les périphériques d'entées-sorties blocs sont des périphériques de stockage à accès aléatoire; autrement leurs contrôleurs les font apparaître comme des périphériques à stockage aléatoire au reste du système. Le sous système de fichier aussi interagit directement avec les contrôleurs d'entrés-sorties "raw", appelés parfois périphériques caractères, incluent tous les périphériques qui ne sont pas blocs.




**Frigure -2.4.1-** Bloc diagramme du noyau du système UNIX

Le sous-système de contrôle de processus est responsable de la synchronisation des processus, de la communication entre les processus, de la gestion mémoire, et de l'ordonnancement des processus. Le sous- système de fichiers et le sous-système de contrôle des processus interagissent lors du chargement d'un fichier en mémoire en vue de son

exécution: le sous système de fichier lit les fichiers exécutables en mémoire avant de les exécuter.

Parmi les appels système de contrôle de processus: fork (crée un nouveau processus), exec (recouvrir un processus en exécution par l'image d'un programme) exit (mettre fin a l'exécution d'un processus), wait (synchroniser l'exécution de processus avec l'exit d'un processus précédemment forké), brk (contrôler la taille de mémoire allouée à un processus)

### 2.4.1.1 La gestion de processus et ordonnancement [2]:

Un utilisateur sous UNIX exécute des programmes dans un environnement appelé "processus utilisateur ". Quand il a besoin d'une fonction du système, il appelle une subroutine. quelque part dans cette subroutine du système il y a un changement de mode: donc sous UNIX "processus système" et" processus utilisateur" sont différentes phases d'un même processus.

La figure 2.4.1.1 décrit les structures de données utilisées par UNIX pour gérer un processus. Elle contient une entrée par processus. Cette entrée est allouée à la création du processus et libérée quand le processus se termine. Cette table est résidante en mémoire et est toujours directement adressable par le noyau du système. Un processus est représenté par trois zones en mémoire.

Le segment de texte utilisateur ("user text segment"): c'est le code du programme qui s'exécute. Ce segment est accessible en lecture seule ("read only"). Lorsque plusieurs processus exécutent le même programme au même instant, le système les fait partager une seule copie du code en mémoire par l'intermédiaire d'une table supplémentaire gérée par le système: la table des textes.

Le segment de données utilisateur (" user data segment"): il contient les données modifiées par le programme. Le système n'utilise pas ce segment pour stocker ses données propres, en particulier, il n'y met pas de tampons d'E/S.

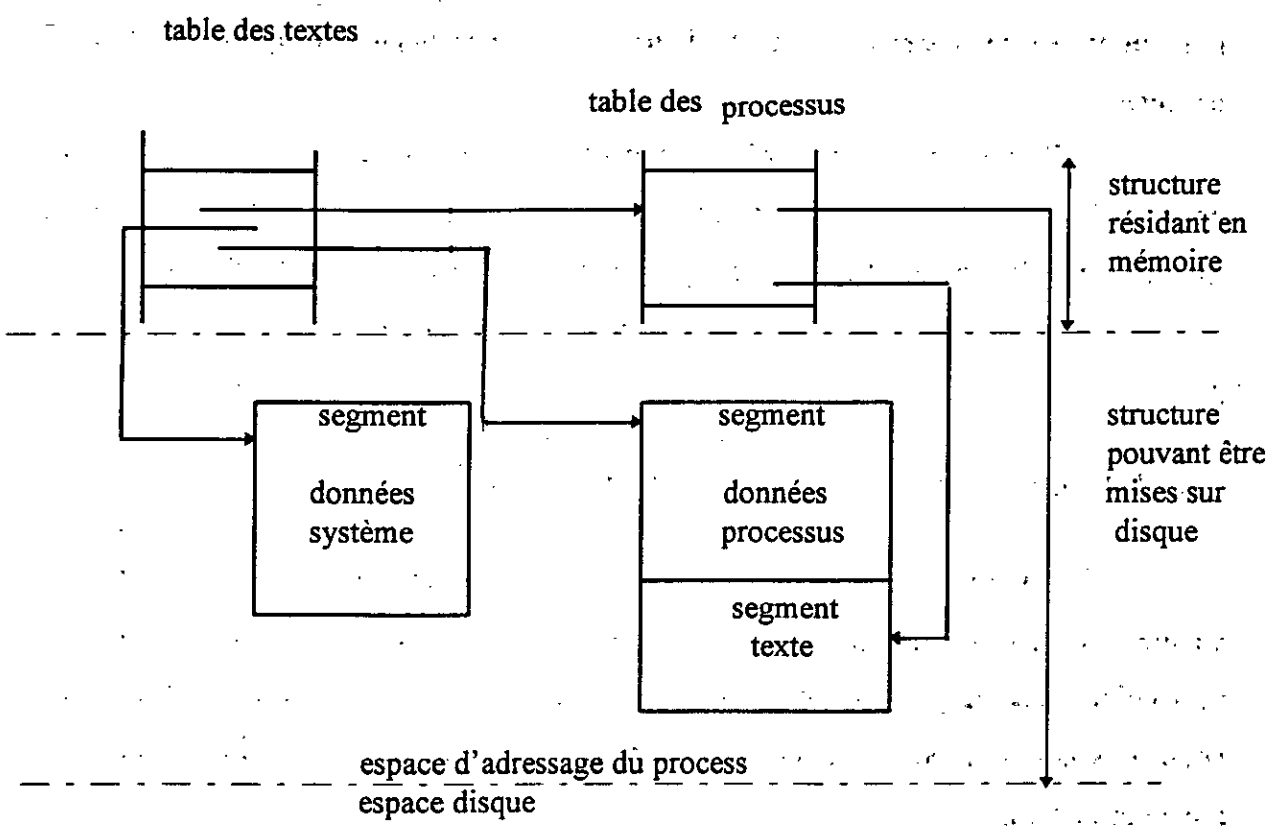



**Figure -2.4.1.1** - les structures de données de gestion des processus sous UNiX.

Le segment des données systèmes ("system data segment "): il contient toutes les données dont le système a besoin quand le processus est actif (ie chargé en mémoire), par exemple le contexte du processus (information sensible), les descripteurs des fichiers ouverts par le processus l'information de facturation (CPU dépensé), une pile pour les phases ("système") du processus.

En particulier le segment système n'est pas adressable depuis le processus (ie n'est pas cartographie dans l'espace virtuel du processus). Il est ainsi partagé.

L'ordonnancement utilise des priorités variables. Plus un processus a utilisé l'unité centrale pendant le dernier intervalle, plus le système lui donne une faible priorité. Toutes les priorités sont réaffectées à la fin de chaque intervalle. Ainsi, toutes choses étant égales par ailleurs, les processus obtiennent l'unité centrale selon un tourniquet dont le cycle est la longueur de l'intervalle multiplié par le nombre de processus. Toutefois, les processus qui effectuent des E/S, utilisant moins l'unité centrale, se voient affecter une priorité plus élevée que les processus qui ne font pas de calculs. De plus, un processus de haute priorité, réveillé par

l'arrivée de l'événement sur lequel il s'était mis en état d'attente, va obtenir l'unité centrale par préemption si le processus qui s'exécute est de plus basse priorité que lui.

### 2.4.1.2 La gestion des entrées/sorties:

Le système d'E/S est séparé en deux sous -systèmes indépendants: les E/S structurées, ou par blocs (block I/O), et les E/S non structurées, ou par caractères (character I/O). Les périphériques sont caractérisés par deux numéros:

-un numéro principe (major device number),

-et un numéro secondaire(minor device number),

et une classe : par bloc ou par caractères. Pour chaque classe, il existe un tableau des points d'entrée dans les conducteurs d'interface. Le numéro principal sert d'index dans ce tableau. Le numéro secondaire est passé au conducteur en argument lors d'une E/S. Usuellement, il sert à désigner un numéro d'unité. Les périphériques du type « bloc » sont vus comme étant constitué de N blocs de 512 octets, accessible de façon aléatoire. En ce qui concerne les programmes utilisateurs, les périphériques sont vus logiquement comme des fichiers d'un type particulier, et les primitives d 'E/S sont identiques en tout point à celles permettant l'accès aux fichiers «standards».

### 2.4.1.3 La gestion de la mémoire[2]:

Cette couche du système d'exploitation est étroitement dépendante du matériel. Par exemple, on ne peut mettre en oeuvre une gestion de mémoire virtuelle par pagination que si le matériel contient des dispositifs permettant d'accélérer le processus de traduction d'adresse . Si au départ d'UNIX on faisait des échanges de programmes dans la mémoire ("swapping"), les versions récentes (UNIX.syst.V de ATT et UNIX.BERKLEY du 4.2) ont une gestion de mémoire virtuelle par demande de pages.

### 2.4.1.4 Le système de fichier[3]:

Le système de fichier d'UNIX est hiérarchisé classiquement selon une structure arborescente. Il contient trois types de fichiers: les fichiers ordinaires, les fichiers spéciaux et les fichiers catalogues .

**a) Les fichiers ordinaires:** ils sont vus comme une suite d'octets, numérotés logiquement de zéro à(N-1), ou N est le nombre d'octets dans le fichier. Le système de gestion de fichier d'UNIX ne propose aucune structure interne d'un fichier et entraîne la responsabilité des programmes utilisateurs.

**b) Les fichiers spéciaux**

permettent de traiter les peripheriques d'entrée/sortie comme des fichiers .Il ont un numero de periphirique majeur et un autre mineur.

Le numero de periphirique majeur fait référance au type de périphirique (tel que le lecteur de disquette)

Le numero de periphirique mineur permet d'adresser un periphirique particulier.

**c) Les fichiers catalogues:** ils sont les noeuds de l'arbre qui forme la structure hiérarchique de l'ensemble des fichiers d'un disque. Un fichier catalogue contient donc une liste de noms de fichiers, certains étaient des fichiers ordinaires, d'autres étant eux mêmes des fichiers catalogues.

**Exemple:**

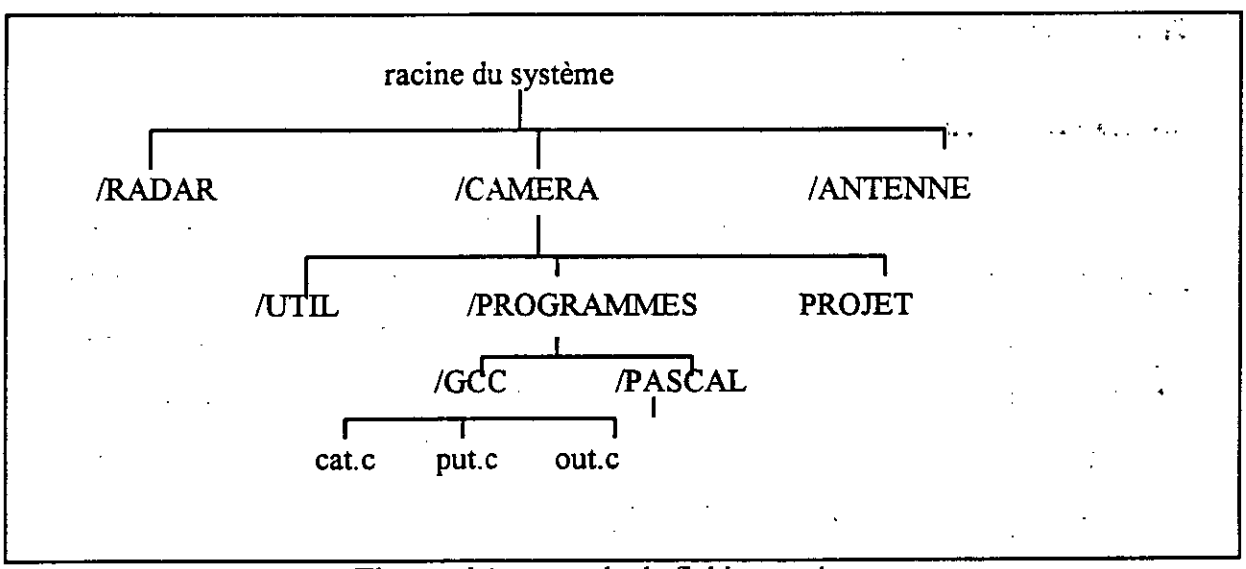



**Figure -b1-** exemple de fichier catalogue

***La structure des fichiers catalogues:***

Un catalogue contient la liste des noms des fichiers qui lui sont rattachés.

Chaque fichier est identique identifié par un numéro unique appelé nombre i ("i-number"). Un même fichier peut être rattachés à plusieurs catalogues, la description de l'endroit ou le fichier est stocké sur le disque n'est pas écrite dans le catalogue, mais dans une zone spéciale du disque qu'on va le décrire ci-dessous.

Un catalogue est simplement un fichier ordinaire dans lequel seul le système a le droit d'écrire. Il contient des entrées de 16 octets: 14 octets pour le nom de fichier et 2 octets pour le nombre i du fichier.

***La structure d'un disque:***

Un disque est vu par le système comme un tableau de blocs de 512 octets accessibles de façon aléatoire. Le système de gestion de fichier découpe le disque en quatre régions.

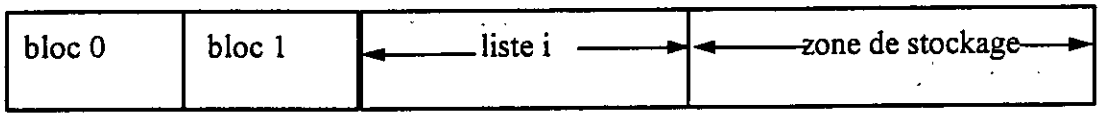



**figure -b2-** Régions sur un disque UNIX

Dans cette figure on trouve les zones suivantes:

-*Bloc 0*: qui est réservé pour la mise en route du système.

-*Bloc 1* : est appelé le "super-bloc", il contient entre autre le nom du disque ("label"), la taille du disque est la frontière des autres zones.

-*liste i*: contient la liste des définitions des fichiers . Chaque définition est une structure de 64 octets appelée noeuds i ("i node"). Le numéro d'un noeud i dans la liste i n'est autre que le nombre i sur le bloc suivant de la liste.

Un noeud i contient 13 adresses sur le disque sous la forme de 13 numéros de blocs. Les 10 premières adresses sont celles des 10 premiers blocs du fichier: si le fichier contient plus de blocs. La onzième adresse porte sur un bloc de 128 blocs suivants du fichier. Si le fichier est plus grand que 138 blocs (70656 octets), la douzième adresse porte sur un bloc dont chacune des 128 adresses porte sur un bloc de 128 adresses sur 128 blocs du fichier. Pour les fichiers encore plus gros que 8459264 octets (8 Mega-octets), la treizième adresse utilise un schéma à trois niveaux d'indirection , ce qui porte la taille maximum d'un fichier à plus de 1 gigaoctet soit 1 082 201 088 octets .

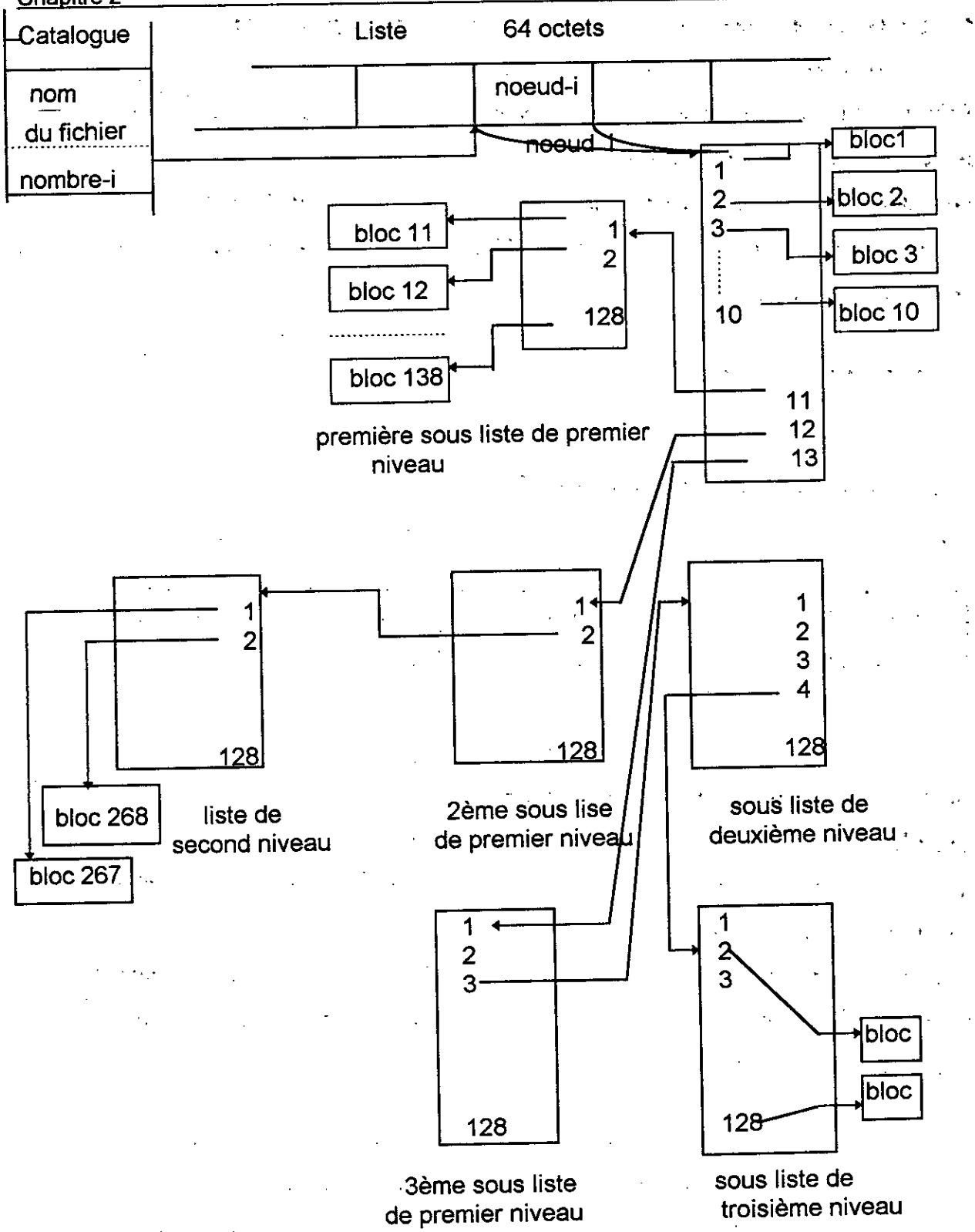



**Figure -b3-** Implantation des fichiers sur disque sous UNIX

### 2.4..2 L'interpréteur de commande (Shell):

Le Shell matérialise, aux yeux de l'utilisateur, le noyau du système d'exploitation et les autres programmes utilitaires qui y sont rattachés(figure2.4). Au fil des années toute une série de Shell a été développée. Le plus connu et le plus répandu est le Bourne- Shell, d'après le nom de son inventeur, Steve Bourne. Il existe depuis le milieu des années 70.[3]

L'une des caractéristiques principales du système Unix est que sont shell est programmable se qui le rend l'un des systèmes d'exploitation les plus souples.

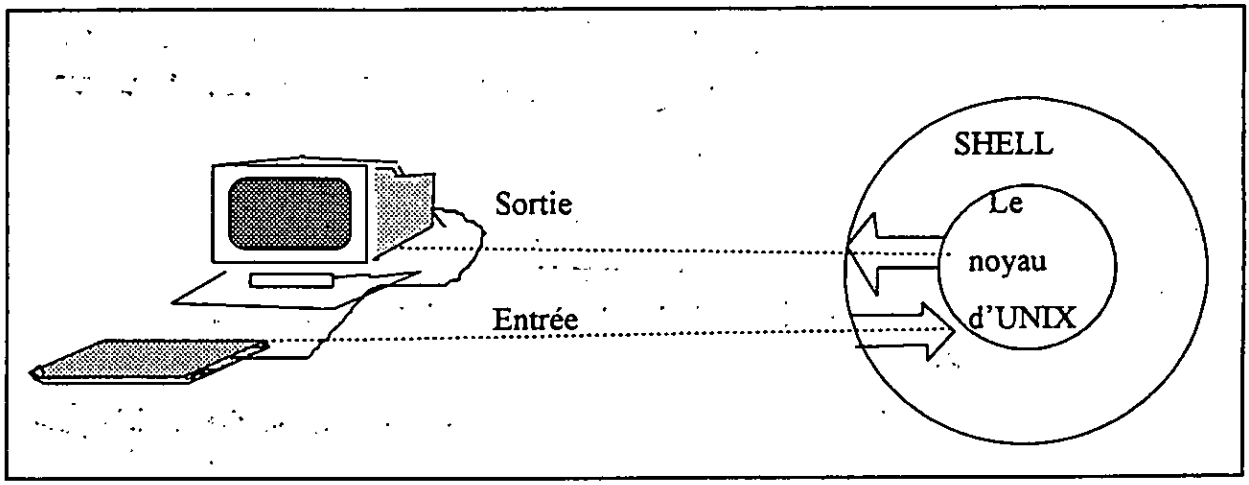



**Figure -2.4.a** - Le shell enveloppe le système d'exploitation UNIX

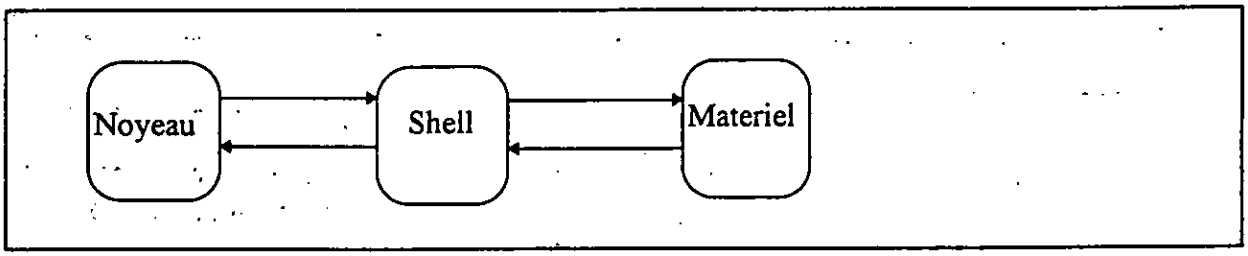



**Figure -2.4.b-** Le shell fait l'intermédiaires entre le noyeau et le matériel

Le fonctionnement de la coquille est le suivant (voir figure2.4.c):l'entrée standard par défaut est le clavier. A la fin d'une ligne de commande, l'utilisateur frappe touche "retour chariot ". La ligne de commande est alors analysée si aucune erreur de syntaxe n'est détectée, le programme correspondant à la commande demandée est lu sur le disque, chargé en mémoire et exécuté. Si une erreur est constatée dans la syntaxe, un message est écrit sur la console. Lorsque le programme est terminé, le contrôle est reprit par la coquille, qui se met en attente de la commande suivante.[2]

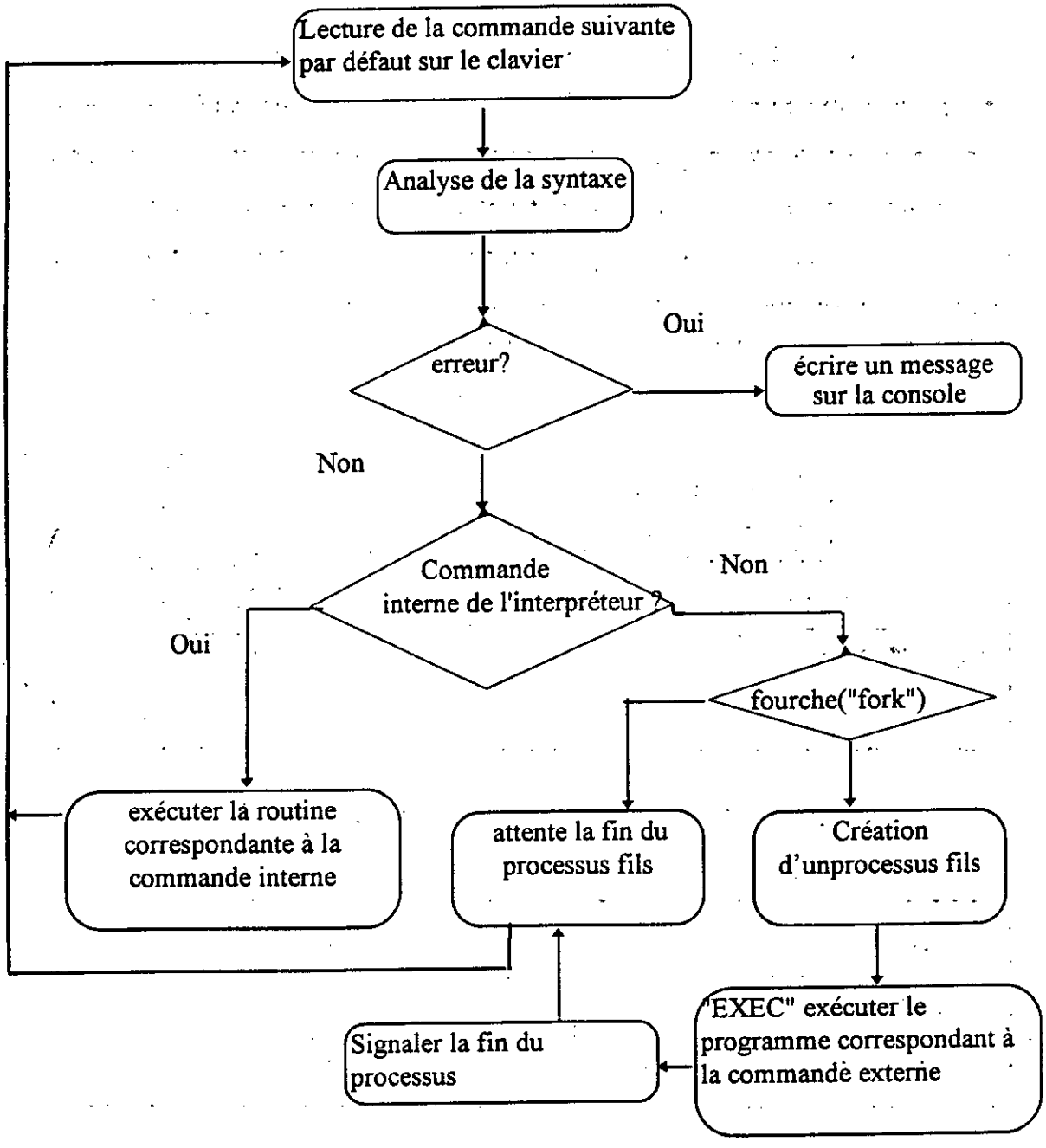



**Figure2.4.c-** Traitement des commandes sus UNIX

### 2.4.3 Applications:

Parmi les principales applications disponibles sous UNIX, les compilateurs de langages, tels que le C ou le Fortran, des optimiseurs de code(optionnels), des éditeurs de lien, . . .

UNIX support aussi des éditeurs de texts tel que **vi** et d'autres application telle que l'interface graphique XWindow.

## 2.5 Les systèmes UNIX sur PC:

Depuis l'introduction des PC, beaucoup de constructeurs se sont penchés sur le problème de savoir si UNIX est portable ces machines et comment réaliser l'opération. Le premier à avoir essayé fut Microsoft lui même, qui diffusa un produit appelé XENIX. En 1979, Microsoft prit une licence UNIX et développa le système d'exploitation XENIX. Jusqu'à cette date les PC étaient tous basés sur le processeur Intel 80286. Mais face au développement des processeurs et des innombrables modifications sur le système, les PC représentés déjà une solution économique; et le fait de rendre une machine accessible à plusieurs utilisateurs n'a fait que renforcer dans le bon sens le rapport performance/prix.

L'apparition du processeur 80386 à 32 bits, la baisse sensible des prix du matériel entraîna une véritable percée d'UNIX sur le marché des PC.

## 2.6 Conclusion

UNIX s'est imposé progressivement sur toute la gamme des systèmes informatiques, du PC portables au super - calculateur.

Parmi les différentes versions pouvant être trouvées sur le marché, nous nous intéresserons à Linux, la version "Slackware".

# Chapitre3
# LINUX présentation

## LE SYSTEME LINUX

### 3.1 Historique[8]:

Linux est une version d'UNIX gratuite et librement diffusable développée à l'origine par Linus Torvalds a l'université de Helsinki, en Finland. Il fut inspiré de Minix, un petit système UNIX développé par Andy Tanenbaum.
Linux a été développé avec l'aide de nombreux programmeurs et spécialistes UNIX, grâce au réseau mondial Internet, autorisant quiconque ayant suffisamment de connaissances à participer activement à l'évolution du système.

### 3.2 Caractéristiques du système[8]:

Linux offre toutes les caractéristiques communément trouvées dans les systèmes UNIX, plus quelques unes qui lui sont spécifiques.
Linux est un système d'exploitation multitâche et multiutilisateur , développé dans un esprit de portabilité.

Les principales caractéristiques spécifiques à Linux sont:

- Le contrôle des processus (POSIX [5]) .
- Les pseudo-terminaux et le support de claviers internationaux ou personnalisés.
- Les consoles virtuelles qui permettent de commuter clavier et écran entre plusieurs sessions de travail en mode texte.
- Linux supporte différents systèmes de fichiers tels que ex2fs[6] , les systèmes de fichiers Minix et Xenix, le système de fichier MS-DOS, ce qui permet dans ce dernier cas d'utiliser directement les disques ou disquettes DOS.
- Linux est doté d'une implémentation complète du protocole réseau TCP/IP[7]

[5] En 1986 naquit la première proposition de définition d'un système d'exploitation qui permettrait d'assurer la portabilité des applications d'une machine sur l'autre. Cet essai de standardisation est connu sous le terme de POSIX (Portable Operating system Interface UNIX) [3].
[6] Le second système de fichier étendu.
[7] Transmission control protocol/ Internet protocol.

- Les bibliothèques partagées: le code des fonctions d'usage courant est stocké une seule fois en mémoire.

Les différents programmes utilisant ces fonctions contiennent juste une référence à ce code ce qui permet de réduire la taille des exécutables et l'occupation mémoire.

- Le système de sécurité : UNIX procure plusieurs moyens d'assurer la sécurité du système, parmi eux :

* Le mot de passe qui protège l'accès au système.
* Le contrôle d'accès qui protège l'accès aux fichiers individuels.
* Un code pour rendre un fichier indéchiffrable.

- Structure du driver sous Linux

UNIX considère chacun des périphériques comme un fichier séparé. Lorsque le système nécessite un nouveau périphérique, l'administrateur crée le lien nécessaire entre ce périphérique et le noyau. Grâce à ce lien que l'on appelle aussi pilote, le noyau et le périphérique communiquent parfaitement chaque fois que ce dernier est appelé à servir.

Les systèmes d'exploitation autre qu'UNIX ne supporte que certains périphériques. UNIX permet l'installation de périphérique quels que soient leur nombre et leur type puisque chacun d'entre eux est considéré séparément grâce au pilote qui le relie au noyau.

Le driver est représenté par la structure suivante:

* Des procédures partagées par tous les périphériques.
* Des paramètres présents dans tous les drivers mais qui peuvent différer d'un périphérique à un autre.
* Des paramètres spécifiques qui diffèrent d'un périphérique à un autre.

### 3.3 Linux - MSDOS

En général, on utilise à la fois Linux et MS-DOS sur le même ordinateur. Beaucoup d'utilisateur de Linux ont besoin de MS-DOS pour exploiter ses applications commerciales qui ne sont pas disponible sous Linux.

MS-DOS n'utilise qu'une partie des possibilités des processeurs INTEL. A l'opposé, Linux exploite toutes leurs possibilités.
Linux permet d'utiliser UNIX sur PC sans payer le prix généralement élevé des autres implémentations UNIX sur PC.

Aucun autre système d'exploitation pour Intel n'a atteint le niveau de popularité de MS-DOS, en grande partie parce que le prix des autres systèmes était inabordable pour un particulier. Linux étant gratuit offre l'occasion de choisir.

Il existe des outils permettant de communiquer avec MS-DOS. Par exemple, il est très facile d'accéder à des fichiers MS-DOS depuis Linux. Il existe aussi un émulateur MS-DOS qui permet d'exécuter beaucoup d'applications populaires.

### 3.4 Les différents shells [5]

L'interpréteur de commande appelé **shell** est un programme qui lit et exécute les commandes tapées au clavier par l'utilisateur.

Les shells disponible sous Linux sont principalement:

*bash* Le Bourne Again shell. Le plus couramment utilisé sous UNIX, compatible avec le shell Bourne, créé et distribué par le projet GNU[8] . Offre l'édition de la ligne de commandes.

*csh* C shell. Développé à Berkley. Compatible avec le shell Bourne, mais possède une interface de programmation très différente. N'offre pas l'édition de la ligne de commandes, mais possède une puissante alternative grâce à l'historique des commandes.

*ksh* Le K shell. C'est le premier shell à introduire des technologies modernes (dont certaines empruntées au C shell) dans le shell Bourne, avec lequel il est compatible. Il offre l'édition de la ligne de commandes.

[8] Les programmes GNU sont créés par la "Free Software Foundation". Ilssont liés au développement de Linux depuis l'origine.

*sh* Bourne shell. Le shell original. N'offre pas l'édition de la ligne de commandes.

*tcsh* C shell amélioré. Offre l'édition de la ligne de commandes et d'autres caractéristiques évoluées.

*zsh* Z shell. Le plus récent. Compatible avec le shell Bourne, offre l'édition de la ligne de commandes.

### 3.5 Administration du système

L'administration de tout système UNIX doit être faite avec soin et demande un certain degré de responsabilité. Les travaux d'administration se font sous le compte **root**. Les permissions et les autres sécurités d'accès ne s'appliquent pas à cet utilisateur spécial. C'est-à-dire que root peut accéder et modifier sans aucune restriction n'importe quel fichier du système, quel qu'en soit le propriétaire.

L'administration du système Linux met la sécurité en avant. Elle permet aux utilisateurs de choisir comment ils désirent rendre accessible leurs propres fichiers, elle évite également que les utilisateurs ne puissent endommager le système.

Très souvent, l'invite du compte superviseur diffère de celle des utilisateurs normaux. Traditionnellement, elle contient un dièse (#) pour root, alors que les autres comptes ont un $ ou un % .

**Gestion du système**

**a) Gestion de l'espace mémoire [3]**

L'administrateur doit être informé en permanence de l'occupation de la place mémoire. Il doit être capable d'avertir les utilisateurs si cette place mémoire devient insuffisante pour la suite des opérations. La commande *df* indique l'occupation en mémoire pour les systèmes de fichiers.

La commande *df* peut être piloté par les options suivantes:

| Option | Signification |
|---|---|
| -t | Affiche en plus , le nombre de blocs de données dans ce système de fichier |
| -v | En employant l'option vebose, l'occupation de la mémoire est indiquée sous forme de pourcentage |

**Tab -3.1-** option de la commande ***df***

**b) Gestion des comptes utilisateurs**[5,6,7]:

**Ajouter un utilisateur**

L'orsqu'on ajoute un utilisateur, celui ci détient alors une entrée dans le fichier mot de passe */etc/passwd*. L'entrée se présente alors sous cette forme:

*logon-name:encripted-passwd:user-ID:group-ID:user-nformation:logondirectory:logon-shell*

Cette syntaxe implique que les champs sont séparés par le symbole ":".La liste des champs se trouve dans le tableau suivant:

| Champs | Description |
|---|---|
| logon-name | Nom utilisé pour se logger |
| encrypted-passwd | Mot de passe indispensable pour authentifier l'utilisateur<br>Principale protection contre les violations de sécurité |
| user-ID | Nombre unique que le système d'exploitation utilise pour identifierl'utilisateur |
| group-ID | Nombre ou nom unique qui sert à identifier le groupe principal de cet utilisateur. Ce dernier peut changer de groupe si l'utilisateur le lui permet |
| user-information | Description de l'utilisateur contenant par exemple son nom |
| Logon-directory | Repertoire d'accueil de l'utilisateur |
| logon-shell | Shell empreinte par l'utilisateur lorsqu'il se logge |

**Tab -3.2-** caractéristiques d'un compte d'un utilisateur

Il existe un programme interactif qui demande les informations nécessaires et paramètre le système automatiquement. La commande ***adduser*** renvoie une série de prompts (invites) et demande au super-utilisateur les renseignements appropriés.

*Créer les mots de passe des utilisateurs:*

La commande *passwd* sert à créer les mots de passe des utilisateurs. L'administrateur du système est censé attribuer un mot de passe à chaque utilisateur qui se rajoute au système. Les utilisateurs peuvent changer de mot de passe attribué au moment de la connexion. Exemple:

*$passwd enp <entrée>*

*Enter new passwd: nouveau mot de passe <entrée>*

*Re-type new passwd: nouveau mot de passe <entrée>*

Le mot de passe est alors codifié et rangé dans le fichier /etc/passwd. Il est important de prendre son temps et de choisir un mot de passe qui soit conforme aux règles que voici:

- Les mots de passe doivent être composés d'au moins six caractères; huit de préférence.
- Les mots de passe doivent comporter des majuscules et des minuscules, des symboles de ponctuation et des chiffres.

- **Retirer un utilisateur**

C'est le fichier /etc./passwd qui gère les comptes utilisateurs. La méthode la plus simple est de supprimer l'entrée correspondante à l'utilisateur. Pour lui ôter uniquement la possibilité de se logé, on place une étoile(*) dans le second champs.

**Ajouter un groupe**

On peut ajouter un nouveau groupe en éditant directement le fichier /etc/passwd ou en utilisant la commande *groupadd. Exemple:*

*groupadd étudiant <entrée>*

Le groupe est rajouté et un numéro unique d'identification lui est attribué. Le numéro d'identification est par défaut le premier chiffre disponible qui soit supérieur à 99 et ceux de tout autre groupe.

Si on veut attribuer un numéro en choisissant un chiffre inférieur à99, on utilise la commande groupadd avec l'option -g, par exemple:

*groupadd -g25 étudiant <entrée>*

**Effacer un groupe**

Pour effacer un groupe, il faut éditer le fichier /etc/group et supprimer l'entrée, ou bien lancer la commande *groupdel.* Exemple:

*groupdel étudiant <entrée>*

*Gestion des répertoires d'accueil*

Pour accueillir beaucoup d'utilisateurs sur le système, essayer de regrouper les répertoires d'accueil de la manière la plus cohérente possible. En général, il vaut mieux placer tous les répertoires d'accueil dan un seul répertoire général.

Le répertoire /home est par défaut le répertoire général pour les répertoires des utilisateurs. Les utilisateurs de *étudiant* auront ainsi des comptes dans /home/étudiant.

**c) Les droit d'accès** [3,9]:

Chaque fichier de l'arborescence d'UNIX est doté d'une identification spécifique. Les informations d'identification les plus importantes sont affichées par la commande *ls* avec l'option *-l.*

**Exemple :**

```
$ls -l
drwxr-xrw-  2  enp  100  16  jan 20, 9:55  fichier1
```

type de fichier / Droits d'accès — Nombre de référence / Possesseurs — Groupe / Taille — Date de dernière modification — Nom

Chaque utilisateur est doté d'un numéro d'utilisateur et d'un numéro de groupe. UNIX distingue les divers utilisateurs uniquement sur la base de leur numéro et non pas d'après leur nom. Il en va de même pour les fichiers, leur gestion se fait sur la base d'un numéro qui permet d'accéder aux données d'identification, en l'occurrence:

- Le nomde l'utilisateur propriétaire du fichier
- Le nomdu groupe auquel ce fichier est affecté

A chaque accès d'un utilisateur sur un fichier, il y a trois cas de figure possibles:

1- Le numéro de l'utilisateur est identique au numéro d'utilisateur référencé dans l'en-tête du fichier. si c'est le cas, l'utilisateur est le propriétaire du fichier.

2- Si ces deux numéros ne correspondent pas, le système vérifie si le numéro de groupe de l'utilisateur est identique au numéro de groupe du fichier. Dans ce cas, l'utilisateur appartient au même groupe que celui auquel le fichier est affecté.

3- Dans tous les autres cas, il n'y a correspondance ni entre les numéros d'utilisateurs ni entre les numéros de groupes, c'est les utilisateurs qui ne sont ni propriétaires ni partie permanente du groupe d'affectation de ce fichier.

**Signification des droits d'accès**

Un fichier peut être doté d'une des possibilités suivantes:

1- Accessible en lecture, en écriture et exécutable pour le propriétaire.

2- Accessible en lecture, en écriture et exécutable pour le groupe auquel le fichier est affecté.

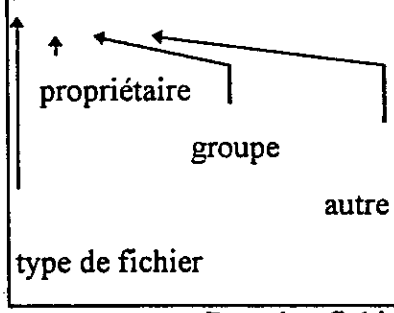
3- Accessible en lecture, en écriture et exécutable pour les autres utilisateurs

| Code d'autorisation | Signification |
|---|---|
| r | Lecture (readable) |
| w | écriture (writable) |
| x | exécutable en tant que programme (executable) |

**Tab -3.3-** les droits d'accés

**Exemple:**

```
$ ls -l
-rwxrwxrwx  1  enp  100  16  jan 25, 10:00  fichier3
```




Pour les fichiers normaux, le premier champ est toujours (-). Le (d) indique qu'il s'agit d'un répertoire.

- **Modification des droits d'accès:** La commande *chmod* permet de fixer les droits d'accès à un fichier ou répertoire, en utilisant la modification par les symboles ou bien la modification par les chiffres en base de 8. Dans le premier cas, le propriétaire utilise la syntaxe suivante:

  *chmod {a,u,g,o} {+,-,=} {r,w,x}* <entrée>

**Exemple:**

$ *ls -l fichier4*

-rw-r-- --- 1 enp étudiant Mai 13, 23:19 fichier4

$*chmod g+wx fichier4*

$*ls -l fichier4*

-rw-rwx --- 1 enp étudiant Mai 13, 23:19 fichier4

$*chmod uo+x fichier4*

$*ls -l ficher4*

-rwxrwx --x 1 enp étudiant Mai 13, 23:19

$*chmod a-x fichier4*

$*ls -l fichier4*

-rw-rw- --- 1 enp étudiant Mai 13, 23:19 fichier4

| Valeur | Description |
|---|---|
| *a qui ?* | |
| a | a tous (all): le propriétaire, le groupe et les autres utilisateurs |
| g | le groupe du propriétaire (group) |
| o | aux autres utilisateurs (other) |
| u | au propriétaire uniquement (user) |
| *Opération* | |
| + | addition du droit d'accès |
| - | retrait du droit d'accès |
| = | spécification du droit d'accès |

**Tab -3.4-** les droit d'accès des groupes et des utilisateurs

La deuxième forme de la commande *chmod* utilise la syntaxe suivante:

*chmod Nombre en base 8 <fichier>*

| *Propriétaire* | *Groupe* | *Autres utilisateurs* |
|---|---|---|
| r w x | r w x | r w x |
| 400 200 100 | 40 20 10 | 4 2 1 |

**Tab -3.5-** autres option de ***chmod***

Le nouveau droit d'accès pour un fichier est la résultante de la somme de ces chiffres. Ainsi pour affecter les droits d'accès en lecture / écriture pour le propriétaire et lecture seul pour le groupe et les autres utilisateurs, on utilise la commande:

*chmod 644 <fichier>*

L'autorisation en lecture /écriture du propriétaire donne la somme de 600, l'accès en lecture du groupe correspond au chiffre 40 et celui des autres utilisateurs correspond à 4.

## 3.6 Le système de fichiers

Pour Linux, un système de fichier est un périphérique formaté de manière à pouvoir stocker des fichiers.

Linux dispose de différents type de système de fichiers tels que le second système de fichier étendu, plus connu sous le nom de **ext2fs**; le système de fichier **MS-DOS**, qui permet d'accéder aux disquettes et disques durs MS-DOS depuis Linux; et plusieurs autres, y compris celui des CD-ROM, ISO-9660.

Le tableau présente la liste des système de fichier couramment supportés par Linux.

| Système de fichiers | Identification | Commentaires |
|---|---|---|
| Etendu, version2 | ext2 | Le plus courant sous linux, très performant |
| Etendu | ext | Obsolete, remplacé par ext2 |
| Minix | minix | Le tout premier celui de minix, peu utilisé |
| XIA | xia | Un peu mieux que minix, rarement utilisé |
| UMSDOS | umsdos | Permet d'avoir linux sur une partition dos |
| MS-DOS | msdos | Celui de ms dos pour accéder à ces fichiers |
| HPFS | hpfs | Partition HPFS, en lecture seulement |
| /proc | proc | Syst. de fichiers virtuel, pour info noyau |
| ISO 9660 | iso9660 | La plupart des CD-Rom |
| XENIX | xenix | Celui du système d'exploitation XENIX |
| System V | sysv | Accès aux fichiers de differents système V |
| Coherent | coherent | Celui du système d'exploitation coherent |

**Tab-3.6-** Les different types de système de fichiers supportés par LINUX

### 3.6.1 Montage des système de fichiers

Sous Linux, il n'est pas possible d'accéder à un système de fichier qu'après l'avoir monté dans un répertoire existant. Cette opération s'effectue à l'aide de la commande **mount**. Sa sysntaxe est la suivante:

*mount -t <type> <périphérique> <point de montage>*

Dans laquelle <type> indique le type de système de fichier, <périphérique> désigne le périphérique physique sur lequel il se trouve, et <point de montage> est le répertoire dans lequel il doit être monté.

Par exemple si le système de fichier se trouve sur une disquette, on emploie la commande suivante: *mount -t msdos /dev/fd0 /mnt*

Cet exemple fera apparaître dans /mnt le contenu d'une disquette MS-DOS.

Le *démontage* du système de fichier le rend à nouveau inaccessible. On pourra alors réutiliser le répertoire /mnt comme point de montage. Cette opération s'effectue par la commande *umount*. Par exemple : *umount /dev/fd0*

### 3.6.2 Organisation interne du système de fichiers :

Un système de fichiers de Linux est une entité complète qui comporte des noeuds d'information, des répertoires et des blocs de données. Il peut être stocké à n'importe quel

périphérique bloc, comme une disquette ou un disque dur. Dans tous les cas, l'organisation du système de fichiers est la même, elle est représentée dans la figure La figure -3.1-

Chaque système de fichiers commence par un **bloc de démarrage** (boot block). Lorsqu'on allume l'ordinateur, le matériel charge ce bloc en mémoire et y effectue un saut. Tous les disques ne peuvent pas servir de périphérique de démarrage, mais pour uniformiser cette structure chaque périphérique possède un bloc de démarrage. Le bloc de démarrage n'est plus utilisé après le démarrage du système.

Le **super-Bloc** contient les informations relatives à l'organisation du système de fichiers. Il est présenté à la figure -3.2-.

La fonction principale du super- bloc est d'informer le système de fichiers de la taille des divers éléments de la figure -3.1-.

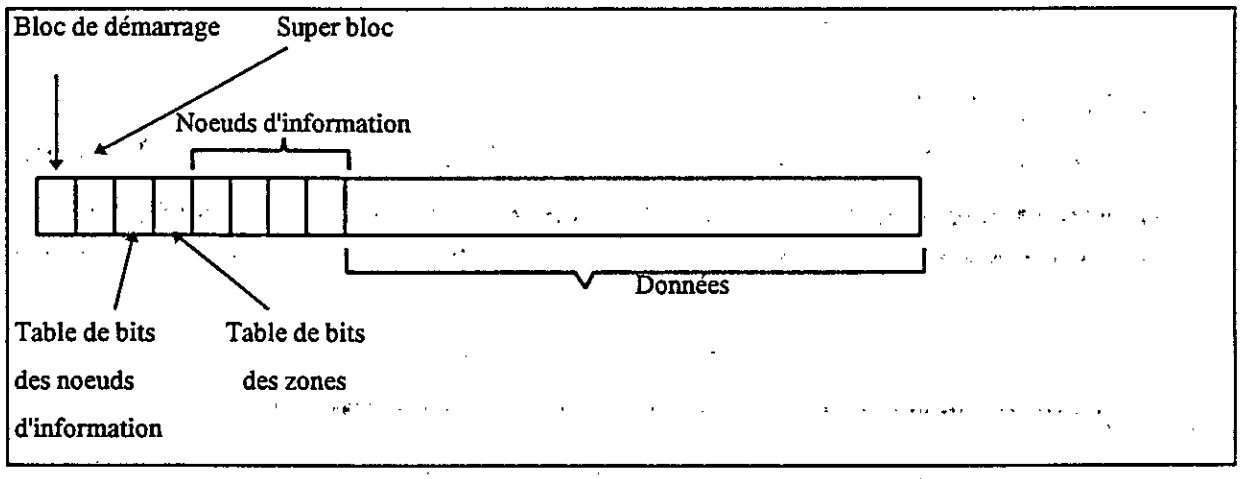



**Figure -3.1-** Organisation du système de fichier

Au démarrage, le super-bloc du périphérique racine est chargé dans une table en mémoire. De même, lorsque d'autres périphériques sont montés, leurs super-blocs sont copiés en mémoire. La table des super-blocs contient quelques informations qui ne sont pas présentés sur les disques, telles que le périphérique d'où provient le super-bloc, un champ qui indique si le périphérique a été monté en lecture uniquement, et un indicateur qui est positionné lorsque la copie du super-bloc en mémoire est modifiée.

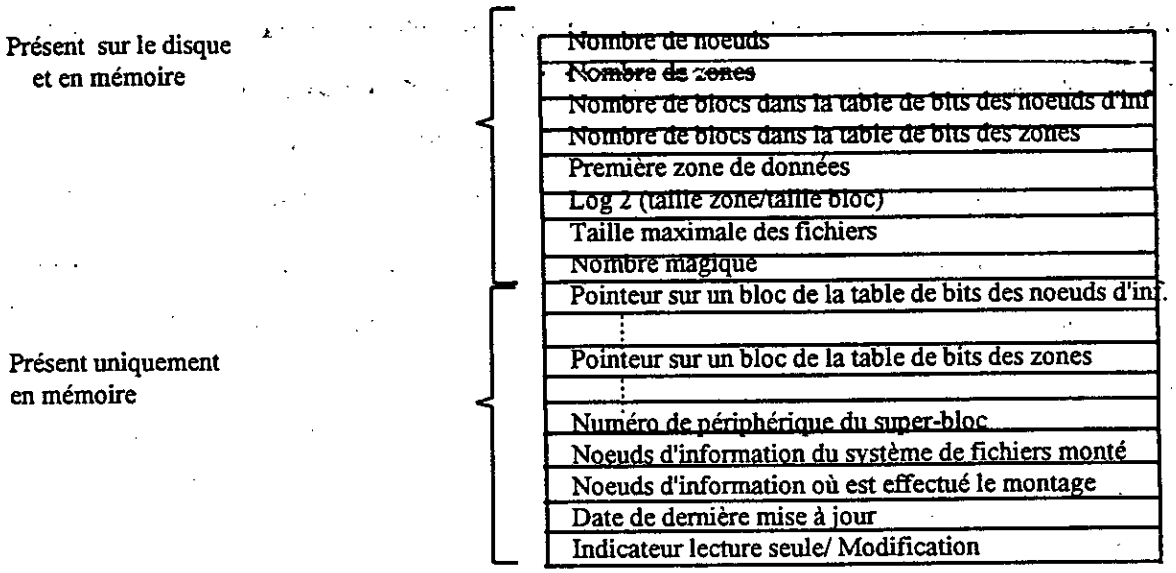
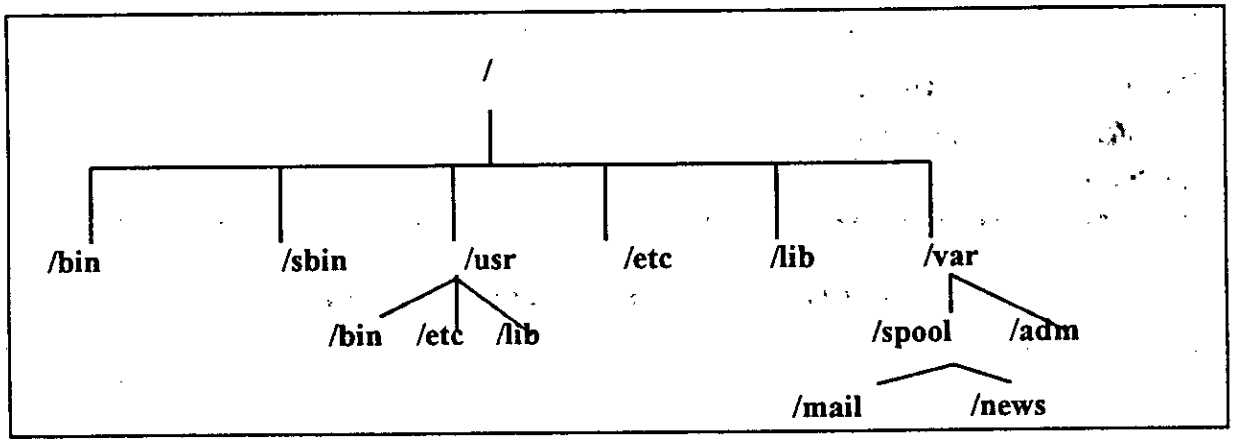



**Figure -3.2-** Le super-bloc

### 3.6.3 Les répertoires Linux [7]

Sous Linux , l'espace fichier visible par les utilisateurs est fondé sur une arborescence dont la racine se situé au sommet. Le schéma suivant donne un exemple d'arborescence:




**/bin** signifie "binaire", ou exécutable. C'est l'endroit ou se trouvent beaucoup de programmes systèmes essentiels.

**/sbin** Contient les commandes essentielles pour l'administration, la configuration et le démarrage du système, utilisées par le superviseur. Ce sont celles qui doivent être absolument disponibles, même si le système de fichier /usr n'est pas encore monté.

**/usr** Contient un certain nombre de sous répertoires qui contiennent les programmes de configuration les plus utiles(tel que le fichier de configuration de Xfree86).

**/dev** Ce répertoire contient des pilotes de périphériques: ce sont des fichiers spéciaux qui permettent aux applications d'accéder au matériel.

**/etc** Contient des fichiers de configuration système , programmes et utilitaires.
**/home** Contient les répertoires personnels des utilisateurs.

**/lib** Contient les bibliothèques partagées. Au lieu d'avoir dans chaque programme une copie individuelle de ses routines, le code est stocké une seule fois au même endroit pour tout le monde.

**/tmp** C'est le répertoire où sont stockés tous les temporaires générés par certains programmes.

**/var** On y place tous les fichiers et répertoires ayant tendance à changer de taille très souvent. Par exemple, dans /var/spool le système place tous les fichiers en attente de traitement, ainsi on y trouve les courriers électroniques non encore lus dans /var/spool/mail; les documents en attente d'impression dans /var/spool/lp ...

### 3.7 La gestion multiprocessus [6]

Le tableau suivant énumère les commandes qui rendent possible le contrôle des capacités multi-utilisateurs et multitâches de Linux.

| Commande | Action |
|---|---|
| at | Exécute des commandes à un temps donné |
| batch | Exécute des commandes lorsque la charge du système le permet |
| cron | Exécute des commandes planifiées |
| kill | Arrête les processus |
| nice | Ajuste la priorité d'un processus avant qu'il ne soit lancé |
| nohup | Permet à un processus de poursuivre son exécution aprés une déconnecté |
| Ps | Affiche les informations sur les processus |
| renice | Affiche les utilisateurs connectés au système |
| who | Affiche les utilisateurs connecté au système |

**Tab-3.7-** Les commandes de multiprocessus

Linux travaille sur une des tâches de la file (liste de tâches) pendant un moment, met celle-ci de coté et en commence une autre, et ainsi de suite. Il retourne ensuite à la prémière activité et reprend son traitement. Linux poursuit ces cycles jusqu'à ce qu'il ait terminé une tâche et la retire de la file. Selon cet arrangement, parfois appelé temps partagé, les ressources du système sont réparties entre les tâches.

- **Lancer un processus en tâche de fond**

Pour exécuter un processus comme un travail en arrière-plan, la commande de lancement est suivie d'un &.

*Exemple* :

```
$ lp <fichier> &
2140
$
```

Le numéro 2140 est le PID. C'est un numéro associé au processus, engendré par la commande lp.

Le numéro lp est le processus fils du shell courant, abituellement un processus père attend que l'un ou plusieurs de ses processus fils se terminent avant de poursuivre. Si on veut

que le père continue sans que le fils aie terminé, la commande qui amorce le processus fils doit être liée au symbole &.

**- Exécuter les commandes à des moments donnés avoc at**

Pour planifier une ou plusieurs commandes à un moment donné, on utilise la commande at en précisant au minimum l'heure de l'exécution.

**Exemple:** at 1:23 <entrée>
lp fichier
echo "fichier imptrimé"

Le tableau suivant résume les façons d'utiliser la commane at.

| Format | Action |
|---|---|
| at hh:mm | Programme le travail à l'heure (hh) et la minute voulue (mm) en réference aux 24 heures de l'horloge |
| at hh:mm mois jour année | Programme de travail à l'heure(hh), la minute (mm), le mois, le jour et l'année voulue |
| at-l | Enumère les travaux programmés |
| at-r ID-Tytravail | Annule le travail en utilisant correspondant à ID-Travail |

**Tab -3.8 -** Les options de la commande ***at***

## 3.8 Les outils

### 3.8.1 Les commandes principales[7,8]

Les principales commandes utilisées sous Linux sont les suivantes :

**cd** Change le répertoire de travail courant.

Syntaxe *cd répertoire*

**chmod** Change le mode des fichiers

Le mode d'un fichier contrôle les droits d'accès qui lui sont associés.

Syntaxe *chmod [option] liste de fichier*

**cp** Copie des fichiers : un fichier vers un autre ou une liste de fichiers dans un répertoire.

Syntaxe *cp [option] source destination*

**df** Affiche la quantité d'espace disponible sur le disque.

Syntaxe *df [option] système de fichier*

**find** Parcourt les répertoires spécifiés, générant une liste de fichiers concordant avec le critère précis. La recherche peut s'effectuer sur le nom, la taille, la date de création, la date de modification et ben d'autres critères.

Syntaxe *find [liste de répertoire] critère*

**grep** Recherche des motifs dans les fichiers et informe lorsque ces motifs sont trouvés.

Syntaxe *grep [option] expression-à-rechercher liste-de- fichier*

**less** Permet de visualiser un fichier page par page tout comme la commande **more.**

Syntaxe *less [option] nom de fichier*

**ls** Enumère les fichiers trouvés sur le système.

Syntaxe *ls [option] liste de fichiers*

**man** Affiche la page de manuel correspondant à la commande donnée en paramètre.

Syntaxe *man <commande>*

**mcopy** Copie les fichiers à partir et vers un système de fichier MS DOS.

Syntaxe *mcopy [option] source destination*

**mdel** Efface un fichier MS DOS d'un système de gestion e fichier DOS.

*Syntaxe mdel -v fichier MS DOS*

**mdir** Affiche le contenu d'un répertoire MS DOS.

Syntaxe *mdir -w répertoire*

**mkdir** Crée de nouveau répertoires dans le système de fichiers.

Syntaxe *mkdir <répertoire1>...<répertoireN>*

**mv** Déplacement de fichiers.

mv permet aussi de renommer un fichier (en effaçant l'original)

Syntaxe *mv <fichier1>...<fichierN> <Destination>*

**rm** Effacement de fichiers. L'option **-i** demande une confirmation a chaque effacement.

Syntaxe *rm -i <fichier1>...<fichierN>*

### 3.8.2 L'éditeur vi et l'éditeur emacs[7,8]

On édite un texte, soit en créant un nouveau, soit en modifiant un texte existant. Dans l'un ou l'autre cas, le texte est gardé en mémoire dans une zone de sauvegarde appelée un tampon. L'utilisation d'un tampon empêche de charger directement le contenu d'un fichier tant qu'on ne décide pas de sauvegarder ce tampon.

#### a) L'éditeur vi

L'éditeur vi est le premier éditeur de texte réellement plein écran qui fut disponible sur les systèmes UNIX. Il est basé sur les mêmes principes que beaucoup d'autres applications UNIX: chaque programme doit offrir une fonction bien spécifique, et pouvoir communiquer avec le reste du système.

Pour démarrer vi, il suffit de taper son nom au prompt de shell, ou bien appeler la commande vi avec le nom de fichier en argument. Quand vi devient actif, l'écran du terminal s'efface et un caractère tilde(~) apparaît sur le coté gauche de chaque ligne sauf pour la première.

L'éditeur vi fonctionne en deux modes : *le mode commande* et *le mode insertion*.

En mode commande, vi interprète ce que vous frappez au clavier comme des instructions, dans le cas d'erreur de commande, vi émet un signal sonore. Pour saisir du texte on utilise le mode insertion en complétant après ou avant le curseur.

***Les étapes à suivre pour créer un fichier :***

1- Lancer vi : taper ***vi <entrée>***, ou bien vi nom de fichier ***<entrée>***

2- Passer en mode insertion : appuyer sur ***<a>*** pour insérer du texte après le curseur, ou sur ***<i>*** pour insérer du texte devant le curseur

3- Saisir le texte

4- Passer en mode commande : appuyer sur *<echap>*

5- Enregistrer le tampon dans le fichier : taper *w nom de fichier*

6- Quitter vi : taper ***q <entrée>***

**b) L'éditeur emacs**

Emacs est un gros programme offrant bien plus de possibilité que n'importe quelle autre application UNIX. Il contient son propre interpréteur de langage **LISP**. Il permet la compilation et le déboguage de programmes, la lecture et l'envoi de courrier électronique, il intègre ses propres didacticiels et documents.

*Les étapes à suivre pour éditer un fichier emacs*

1- Lancer emacs : taper *emacs <entrée>*

2- Saisir le texte (taper le texte dans le tampon)

3- Enregistrer le tampon dans le fichier : appuyer sur *<ctrl-x> <ctrl-c>* et répondre *y* à la demande d'enregistrement du fichier

4- Quitter emacs *: appuyer sur <ctrl-x> <ctrl-c>*

Lancer des commandes depuis Emacs

Emacs offre des interfaces pour de nombreux programmes qui fonctionnent alors dans un tampon l'éditeur. Par exemple, il existe des modes Emacs pour lire ou expédier du courrier électronique ou utiliser le shell.

Pour poster du courrier électronique depuis Emacs, La commande <Ctrl-c> m ouvrira un tampon qui permet de saisir un message, puis de l'expédier en utilisant la commande<Ctrl-c> <Ctrl s>.

```
To :  -
Subject :
--text follows this line- -


- - - - - Emacs : * mail *                    (Mail) - - - -All- - - - - -
```

RMAIL est l'interface offerte par Emacs pour lire le courrier électronique. La commande destinée à lancer RMAIL est <Alt-x> rmail.

Lorsque vous lancez RMAIL, Emacs commence par convertir les messages contenus dans votre « boîte aux lettres » dans un format spécial qu'il utilise pour la gestion du courrier électronique.

**Configuration d'Emacs**

Le fichier contenant la configuration personnelle de chaque utilisateur s'appelle *.emacs* et doit se trouver dans son répertoire personnel.

Le fichier .emacs contient du code écrit en LISP Emacs, qui définit des fonctions conditionnant l'environnement de chacun. Nous donnons l'exemple suivant concernant l'affectation des touches.

**Exemple :**

On édite l'instruction suivante dans le fichier.emacs :

```
; Lecture du courrier.
( globel-set-key) « \C-c r «  ' rmail)
```

Les commentaires sont introduit par des points virgules **(;)** , la commande **globel-set-key** est une affectation globale qui fonctionne depuis n'importe où dans Emacs, elle remplace la séquence <Alt-x> rmail avec les touches <Ctrl-c> r.

### 3.9 Programmation sous Linux

Linux offre une interface de programmation évoluée, utilisant les outils GNU et le débogueur gdb.

On trouve également un certain nombre de langages de programmation tel que Perl et le LTPS. Quelque soit les besoins en matière de développement, Linux est une plate - forme de choix pour les programmes de qui le souhaitent n'ont aucune peine à descendre au niveau système .

#### 3.9.1 La programmation avec gcc [ 2,6]

Le langage C est le plus universellement utilisé pour le développement sous UNIX, lui même écrit en C. Le compilateur gcc de GNU est l'un des compilateurs les plus performants qui puisse exister actuellement, il supporte tout les standards de programmation C modernes en usage - comme le standard Américain National Standardisation Institute (ANSI) - aussi bien que certaines extensions qui lui sont spécifique. C'est aussi un compilateur C++ pour la programmation orientée objet. Des bibliothèques de classes sont également fournies, comme la bibliothèque iostream familière aux programmeurs C. gcc supporte aussi objective C, qui est un langage orienté objet.

**Caractéristiques de gcc**

- Le compilateur gcc supporte la syntaxe C « standard » spécifiée en grande partie par le standard ANSI.
- La plupart des compilateurs C proposent l'option -o pour mettre en route l'optimisation du code généré ; gcc propose plusieurs niveaux d'optimisation.
- Le compilateur insère des marques dans les fichiers objets permettant à un débogueur de localiser les fonctions, variables et lignes de programme correspondantes dans l'exécutable compilé de cette manière. A l'aide d'un débogueur comme gdb on peut tester toutes portion du programme et voir simultanément le code source original.
- gcc génère du code assembleur ce qui permet d'apprendre des astuces en mode protégé sous Linux.
- gcc comprend son propre assembleur.

- **Appel du compilateur C**

Les programmes en C sont enregistrés dans des fichiers texte tout à fait normaux. Lors de l'affectation d'un nom a ces fichiers, il faut tenir compte de leur extension spécifique *.c* ; si le fichier n'a pas cette extension, il ne sera pas pris en compte par le compilateur.

Le compilateur C est appelé par la ligne de commande. Toutes les indications de configuration pour ce compilateur sont appelées par la ligne de commande.

Les étapes de la génération des programmes

Lors de la première phase, le préprocesseur traite le fichier. Il convertit au besoin les constantes symboliques ou les macros et rajoute éventuellement d'autres fichiers.

Le compilateur traduit dans une seconde phase les instructions du C en instructions machine pour le processeur du PC. Il n'est pas nécessaire que dans le cadre du programme, l'ensemble des fonctions soit directement programmé. Il y a aussi moyen de faire référence à des fonctions définies ultérieurement.

Ce n'est que dans la troisième phase que ces références externes seront mise en place par le Linker. Ce Linker met en rapport les programmes traduits et les bibliothèques nécessaires et cré un programme exécutable à condition qu'aucune erreur n'ait apparue.

Option du compilateur

En général, le compilateur n'est pas appelé directement. On passe le plus souvent par un programme qui gère les diverses phases de cette opération sur la base d'options. La commande de pilotage s'appelle gcc ou bien gcc.

| Option | Signification |
|---|---|
| P | Demande au compilateur de n'appeler que le pré-processeur. Pour finir il existera un fichier ayant le même nom, avec l'extension -i. |
| -c | Avec cette option, le compilateur est appelé aprés le processeur sans que le Linker n'intervienne. Le résultat est un fichier avec le même nom et une extension.o. Ce fichier contient les codes machine du programme |
| -g | Permet la collaboration entre le programme et des logiciels de débogage |
| -o | Le programme exécutable est placé dans un fichier appelé a.out. Avec l'option -o, on peut indiquer le nom directement |

**Tab-3.9-** Les options de *gcc*

**Exemple** :

```
$ ls
fichier1.c
fichier2.c
$gcc -c fichier1.c
$ls
fichier1.c          fichier1.o
fichier2.c
$gc -o fichier1.c
$ls
a.out*              fichier2.c
fichier1.c          fichier1.o
$mv  a.out   fichier1
$ls
fichier1*           fichier2.c
fichier1.c          fichier1.o
$gcc -o fichier2  fichier2.c

$ls
fichier1*           fichier1.c          fichier1.o
fichier2*           fichier2.c
```

### 3.9.2 La programmation sous shell (les scripts)

Le shell programmable est une caractéristique d'UNIX qui fait de lui le système d'exploitation le plus souple actuellement disponible.

**L'alias de commande**

L'alias de commande permet de donner un nom à celle ci. Exemple : La commande *man* visualise la documentation de Linux et les pages d'aides. Pour faire apparaître le mot *help* comme alias, on utilise la syntaxe suivante :

*alias help=man*

Pour faire apparaître le mot *accès* en alternative à la place de la commande *ls -l*, on utilise la syntaxe :

*alias accès= "ls -l "*

Les guillemets (" ") sont nécessaires parce que le shell attend que l'alias se termine par un espace ou par un <entrée>.

**Les scripts shells**

Un script shell est une collection de plusieurs commandes shell dans un fichier. Les avantages de cette approche sont :

- On n'a pas à retaper une suite de commande.
- On détermine les étapes pour atteindre l'objectif.
- On simplifie les opérations pour tous les utilisateurs.

**Exemple :**

- Utiliser l'éditeur de texte vi ou emacs pour mettre des commandes tel que who, cal et date dans un fichier intitulé infos
- Rendre ce fichier exécutable avec la commande : *chmod +x infos*

Pour obtenir les résultats des trois commandes, taper la commande infos.

**Ecrire des programmes**

*Utiliser echo*

La commande *echo* affiche des messages d'information explicitant ce qui se passe dans un script shell. Exemple :

echo « Quel heure est il ? »

date

A l'exécution de ce fichier on trouve :

Quel heure est il ?

Mon Dec 20 :06 1997

**Utiliser des commentaires**

Le commentaire est une note ignorée par le shell et qui explique sa fonction. Chaque caractère à partir du signe # jusqu'à la fin de la ligne, fait partie d'un commentaire. Exemple :

#Ce fichier s'appelle infos
echo « Regardons qui est sur le système «
who # Affiche les personnes connectées

**Utiliser la commande read**

La commande *read* arrête le script et attend une entrée en provenance du clavier. Quand <entrée> est pressée, le script continue. Si <Ctrl-d> est entré pendant que la commande read attend une entrée, le script est interrompu.

**Exemple :**

#Copier le fichier spécifier dans le répertoire /usr
echo « Entrez le fichier à copier »
read nom-de-fichier
cp $nom-de-fichier /usr

**Programme avec des structures de contrôle**

Il y a deux structures de contrôle dans la programmation shell : les décisionnelles et les itératives. Dans les structures décisionnelles, telles que *if...then...else...*et *case*, le shell décide quelles commandes exécuter suivant la valeur de l'expression. Dans les structures itératives telles que les boucle *for* et *while*, on peut exécuter une suite de commandes sur une série de fichiers.

- Utiliser case

**Syntaxe :** *case mot in*

motif) déclaration
motif) déclaration

.....

```
esac
```

**Exemple :**

```
echo « Choisissez I pour imprimer un fichier »
echo « Choisissez E pour effacer un fichier »
read choix
case $choix in
P|p) echo « nom de fichier à imprimer »
read nom-de-fichier
lp $nom-de-fichier
D|d) echo « nom de fichier à effacer «
read nom-de-fichier
rm $nom-de-fichier
*) echo fait
esac
```

Dans cet exemple la barre verticale est utilisée pour donner un choix pour la recherche. P|p, signifie que le P majuscule ou minuscule est reconnu comme concordant. Le motif * est utilisé pour représenter tous les motifs explicitement établis.

- Utiliser la structure if

La structure *if...then...else...fi* est la structure décisionnelle qui vous permet de sélectionner un ou deux cours d'actions basées sur le résultat de la commande. La portion else de la structure est optionnelle. Exemple :

```
# Ce script teste le type de fichier avec la commande test
# On utilise la commande else pour tester les fichiers réguliers
echo « Le fichier à tester »
read fichier
        if      test -f $fichier
        then    echo « fichier »
        else    echo « répertoire »
```

```
fi
```

- Utiliser les structures d'itérations

Les structures de contrôle itératives vous permettent d'écrire des scripts shells qui contiennent des boucles à l'aide de for et while. Exemple :

Pour copier tous les fichiers dont le nom finit par les caractères.txt dans le répertoire texdir en utilisant la boucle for :

```
for i in*.txt
do
cp $i textdir/$i
done
```

- Transmettre des variables au shell

Une variable initialisée à l'intérieur d'un shell a cette valeur uniquement au sein de ce même shell. La valeur disparaît ou est réinitialisée quand on quitte ce dernier. Exemple :

```
# On fixe la valeur de la variable today
today=Samedi
# Affiche de la valeur de la variable
echo $today
today=dimanche  #On fixe today a une autre valeur
echo « Aujourd'hui nous somme $today »
```

### 3.10 Le système X Window

Le système X Window est un interface graphique standard des systèmes UNIX. C'est un environnement très puissant supportant de nombreuses applications. Avec X Window, l'utilisateur peut avoir plusieurs terminaux graphique simultanément à l'écran, chacun contient un programme différent.

Le X est issue du projet Athena développé au MIT (Massachusetts institute of technology) et Digital Equipement Corporation. La version courante est la version X11R6 depuis 1994.

Une implémentation complète a été réalisée pour Linux.: XFree86 est un standard de X11 particulièrement optimisé, basé le processus i386. XFree est fourni dans la distribution du MIT, il comprend beaucoup d'extensions et d'optimisations pour les cartes vidéo les plus couramment rencontrées sur les machines de type PC.

X se base sur une architecture client/serveur dans laquelle le serveur X est un programme qui gère tous les accès à l'écran graphique, la souris et le clavier. Un client X est une application qui communique avec ce serveur en lui passant des requêtes. Parmi ces clients, xterm (qui émule un terminal dans une fenêtre) et xman (un lecteur de pages de manuel).

Les clients fonctionnant sous X sont affichés à l'intérieur d'une ou plusieurs fenêtres sur l'écran. La manière dont ces fenêtres sont manipulées est contrôlée par un client appelé le gestionnaire de fenêtres, fonctionnant en même temps que tous les autres programmes.

**Démarrage de XFree86**

Pour lancer XWindow , la commande à utiliser est *startx* . Cette commande lance le serveur X et exécute les instructions contenues dans le fichier.*xinitc*. Ce fichier est un shell script contenant les clients X que vous désirez lancer au départ. Pour sortir de X, pressez la combinaison de touches <ctrl> <alt> <backspace>.

**Configurer l'environnement :**

Le fichier responsable de la configuration du Xwindow est le: /usr/11R6/lib/ConfigXF86/ ConfigXF86 , dans ce fichier ,on fait configurer la souris,la carte graphique,le moniteur .

**Personnaliser l'environnement**

Pour ne pas être obligé de subir les valeurs par défaut, nous citons un exemple de script qui nous permet de personnaliser l'environnement de travail. Une fois le X lancé par la commande startx, celle ci appelle le programme destiné à démarrer le programme destiné à démarrer le serveur X : xinit. Ce dernier exécute le script .xinitrc. S'il n'existe pas c'est le fichier de configuration global /usr/X11R6/lib/X11/xinit/xnitrc qui sera pris en compte.

### 3.11 Le réseau [5,8]

Linux comporte une implémentation complète des protocoles réseau **TCP/IP** (Transport Control protocol/Internet Protocol). TCP/IP est devenu le protocole le plus connu et le plus utilisé dans le monde entier pour la mise en réseau des ordinateurs.

Avec Linux et une simple carte Ethernet, on peut se connecter a un réseau local ou (avec les liaisons adéquates) à Internet, le réseau TCP/IP mondial.

Pour installer un réseau de machine UNIX ou Linux, il suffit d'un contrôleur Ethernet dans chaque ordinateur, et les câbles appropriés. De même si l'école ou l'entreprise est connectée au réseau Internet, il devient très simple de se connecter à ce réseau.

Linux supporte également les protocoles **SLIP** (Serial Line Internet Protocol) et **PPP** (Point to Point Protocol) qui permettent de se connecter à Internet (ou tout autre réseau TCP/IP) par modem.

*Le protocole réseau TCP/IP*

Pour bien utiliser la puissance TCP/IP, il faut bien comprendre ses principes de base. Le Transport Control Protocol/Internet Protocol est une suite de protocoles définissant comment les machines doivent communiquer entre elles via un réseau, aussi bien qu'en interne avec les autres couches de cette suite de protocoles. TCP/IP fut créé pour le réseau de l'*Advanced Research Projects Agency, ARPA-Net*, destiné à la recherche militaire scientifique. Par conséquent on entend parler quelquefois de « DRAP Internet Protocols » pour désigner le TCP/IP. Depuis, beaucoup d'autres réseaux utilisant cette technologie ont vu

le jour, aussi bien que des milliers de petits réseaux locaux et régionaux tout autour du monde. La majorité d'entre eux sont interconnectés pour former un conglomérat connu sous le nom de Internet.

Sur le réseau TCP/IP, chaque machine se voit attribuer une adresse IP, qui est un nombre de 32 bits unique identifiant le système. Elle est en général représentée sous forme de quatre nombres décimaux séparés par des points. Exemple :

| partie réseau | partie hôte |
|---|---|
| 128.17 . | 75.20 |

L'adresse IP est divisée en deux parties : l'adresse du réseau et l'adresse du hôte. Un hôte est en général une machine séparée sur le réseau. La taille de ces deux champs dépend du type du réseau. Par exemple, pour un réseau de classe B (pour lequel le premier octet de l'adresse est compris entre 128 et 191 ) ; les deux premiers octets identifient le réseau et les deux derniers la machine. Donc, dans l'adresse citée ci-dessus, le réseau est 128.17 et le système est 75.20. Cette machine est donc la 75.20 du réseau 128.17.

De plus la partie machine de l'adresse IP peut être subdivisée pour permettre une adresse de sous-réseau. Les sous-réseaux permettent de diviser les grands réseaux en plusieurs plus petits, chacun pouvant être géré indépendamment. Exemple

| partie réseau | partie sous réseau | partie hôte |
|---|---|---|
| 128.17 | .75 | .20 |

Pour en voir une illustration, dans ce cas, l'adresse IP 128.17.75.20 identifiera le système 20 du sous-réseau 75 du réseau 128.17. Les protocoles désirant communiquer par TCP/IP, spécifient généralement l'adresse de destination ainsi qu'un numéro de port. L'adresse de destination est utilisée pour router les données d'un système à l'autre. Le numéro de port est un nombre de 16 bits qui désigne un service particulier ou une application

spécifique située sur la machine distante, à qui ces données sont destinées. On peut se représenter ces ports comme des numéros de bureaux dans un grand immeuble administratif, le bâtiment possède une adresse, mais chaque service est dans un bureau différent.

Le programme « telnet » permet à tout utilisateur de se connecter à une autre machine. Sur le système distant, il y a le démon [9] « telnetd » qui est à l'écoute sur un port spécifique, des nouvelles connections.

L'utilisateur qui exécute telnet, spécifie l'adresse de la machine sur laquelle il veut se connecter, et le programme telnet tente d'effectuer une connexion sur le port spécifique de l'ordinateur distant. Si cette opération réussit, telnet et telnetd peuvent alors communiquer entre eux pour offrir une session distante à l'utilisateur en question.

**La famille TCP/IP**

La famille TCP/IP comporte un certain nombre de protocoles :

TCP (Transmission Control Protocol) a la charge de fournir une communication fiable, sans erreur, basée sur une connexion entre deux processus pouvant s'exécuter sur différentes machines du réseau.

UDP (User Datagram Protocol) est similaire à TCP, à la différence qu'aucune connexion n'est effectuée et que le service n'est pas fiable. Les programmes utilisant UDP doivent implémenter leur propre méthode d'acquittement de synchronisation de synchronisation si nécessaire. TCP et UDP transmettent les données par blocs, connus sous le nom de paquets. Chaque paquet contient une portion de l'information à transmettre, ainsi qu'un en-tête indiquant l'adresse et le port de destination.

IP(Internet Protocol) se situe sous les deux protocoles. Son rôle est de transmettre et diriger les paquets TCP ou UDP à travers le réseau. Pour cela, IP encapsule chaque paquet dans un nouveau dans un nouveau paquet appelé datagramme IP, qui contient un en-tête avec

[9] Démon, en anglaisi daemon , qui signifie Disk And Extension MONitor, c'est à dire qui n'est pas invoquer manuellement mais attend en tâche de fond que quelque coditions se produisent.

les informations nécessaires au routage : le datagramme IP transporte aussi les adresses source et destination des machines.

Le réseau est un ensemble de machines qui sont interconnectées par un lien physique quelconque (Ethernet, lignes séries, radio, satellite, ...). Chaque réseau possède ses propres méthodes internes de routage des paquets, localement.

Les réseaux sont connectés les uns aux autres par des passerelles. Une passerelle est un ordinateur qui est en connexion directe avec deux réseaux ou plus : il sert alors d'intermédiaire pour échanger des informations entre ces réseaux, et d'ériger les paquets de l'un à l'autre.

Configuration de TCP/IP sur Ethernet.

On présume qu'on dispose d'une machine qui fera partie d'un réseau local existant et que par conséquent, les passerelles servant de nom sont déjà configurées et disponibles et que le système dispose d'un réseau TCP/IP.

La méthode la plus simple est de lancer la commande *net config* ou bien reconfigurer de nouveau le noyau. Les fichiers qui gèrent la partie Network sont */etc/rc.d/rc.inet.1* et */etc/rc.d/inet.2*. Par défaut, ces fichiers ne sont pas lancés à l'amorçage du système, il faudra les placer dans le fichier /rc.d/rc.local. Ce fichier est équivalent a autoexec.bat dans le MS-DOS.

**NB** : Slackware Linux propose l'installation d'un noyau compilé avec la gestion TCP/IP activé, si le choix a été accepté, on n'aura pas a recompiler le noyau.

**La configuration réseau**

Avant de pouvoir configurer le module TCP/IP, on doit disposer de certaines informations sur le réseau :

*L'adresse* IP : il s'agit d'une adresse unique attribuée à la machine. Elle se présente sous *forme* de quatre nombres décimaux séparés par des points.

**Exemple :** 128.253.153.54

*Net masck* : C'est un nombre similaire à l'adresse IP. Il est utilisé par le système pour le routage des messages.
Le netmasck est choisi par l'administrateur du réseau lors de sa mise en place.

La plus part des réseaux de classe C utilisent la valeur 255.255.255.0, alors qu'un réseau de classe B utilise plutôt 255.255.0.0. Si on ne fixe rien, Linux fixera par défaut un net masck correspondant à une absence de sous-réseau.

*Adresse réseau:* Il s'agit d'une adresse IP combinée bit à bit par un *et* logique avec le net masck. Par exemple si l'adresse IP est 128.253.154.32 et le net masck 255.255.255.0, alors l'adresse réseau sera 128.253.154.0.

L'adresse broadcast: Elle est utilisée pour diffuser des messages à l'ensemble des machines du sous-réseau. Pour l'obtenir on effectue un *ou* logique entre l'adresse IP et le complément à 1 du net masck. Ainsi avec un netmasck à 255.255.255.0, il faudrait faire un ou logique avec la valeur 0.0.0.255. Dans cet exemple l'adresse broadcast vaudrait 128.253.154.255.
*Adresse de* passerelle: C'est l'adresse IP de la machine qui permet de communiquer avec le " monde extérieur " . Très souvent la passerelle possède une adresse similaire à celle du système mais avec « 1 » pour dernier nombre. Ainsi il y a de fortes chances que l'adresse de la passerelle dans l'exemple cité soit 128.253.154.1.

*L'adresse du serveur de noms*: La plus part des sites importants utilisent un serveur de noms dont le rôle est de gérer une table de correspondance entre adresse IP et nom de machine.

**SLIP ET PPP**

On pourra avoir besoin de certaines de ces informations pour configurer la machine. Les prestataires d'accès SLIP et PPP utilise deux méthodes pour la gestion des adresses IP, soit par l'adresse immuable IP (static IP) de la machine, soit elle est attribuée à chaque fois qu'on soit connecté au serveur (dynamic IP).

**Configurer SLIP** SLIP(Serial Line Internet Protocol) permet d'utiliser TCP/IP avec une liaison série, ou par l'intermédiaire d'un modem. On doit avoir l'accès a un serveur supportant le protocol SLIP.

Les deux principaux programmes relatifs à SLIP sont *dip* et *slattch*. Ils permettent tous deux d'initialiser une connexion SLIP sur un port série. Il ne suffit pas, en effet, de connecter le serveur SLIP avec un programme de communication comme Kermit et de lancer « ifconfig » et « root » pour que tout fonctionne, un appel système spécial « ioctl » doit être fait pour utiliser le port série. Le programme « dip » peut être utilisé pour initialiser le port série et le modem. « dip » permet aussi de configuer la machine en serveur SLIP. Le programme « slattch » initialise la liaison série en mode SLIP.

Contrairement au protocole Ethernet, il n'existe que deux machines sur un réseau SLIP : la machine dont on dispose et le serveur. Pour cette raison on parle de liaison PPP : une extension de ce concept le PPP qui a été réalisé pour Linux.

Quant on est connecté au serveur SLIP, celle ci doit attribuer une adresse IP. En pratique, le serveur envoie l'adresse IP attribuée, ainsi que l'adresse de la passerelle au début de la connexion. « dip » intercepte ces valeurs et les utilise pour configurer la machine. La configuration d'une connexion SLIP est très semblable à la configuration d'un réseau TCP/IP.

**Transfert de données par UUCP**

Le protocole UUCP(UNIX TO UNIX COPY) est un procédé permettant de transférer des informations entre deux systèmes. Avec UUCP, les machines se contactent par modem et peuvent transférer automatiquement des fichiers des courriers électroniques, et des articles de forums Usenet...

Si on n'a aucun accès TCP/IP ou SLIP, UUCP permet de dialoguer avec le monde extérieur. La plus part des logiciels standard gérant le courrier électronique ou les News Usenet peuvent être configurer pour utiliser UUCP comme système de communication.

## Le courrier électronique

Comme tous les systèmes UNIX, Linux dispose d'un grand nombre de logiciels permettant l'envoi et la récupération de courrier électronique qui peut être local (entre deux

utilisateurs d'une même machine) ou liée à un réseau, les courriers sont alors transmis entre différentes machines par TCP/IP.

Un système de gestion de courrier électronique contient habituellement deux couches : Le *mailer* est le programme exécuté par les utilisateurs pour écrire le courrier ou lire ceux qui leur sont destinés.

La couche de transport de l'acheminement du courrier sortant et stockage du courrier entrant en respectant divers protocoles. Un utilisateur n'a pas à savoir, il se contente d'interagir avec le mailer. Le logiciel le plus populaire sous Linux est Smail. Simple a configurer, il peut gérer des courriers en local ou en réseau.

**Utilisation du FTP**

FTP (File Transfert Protocol) est un protocole utilisé pour transférer des fichiers par le réseau Internet. Avec le programme *ftp*, on peut envoyer ou recevoir des fichiers. Dans la plus part des cas, on se contente de la fonction de réception.

**Application:**

Lancer ftp (cas d'un réseau local)

**Exemple :**

« Pol1 » représente la machine d'adresse 175.62.0.9

« station » représente une station graphique de 64Mo de RAM et d'adresse 175.62.0.2

« AV1 » est le système DGU (Data Général Unix) d'adresse 175.62.0.13

Dans ce cas le réseau est 175.62 de classe B.

Pour demander une connexion a la station :

```
Pol1 #ftp 175.62.0.2 <entrée>
220 Pascal FTP server
(version wu-2.4 (1) SUN 15  14 :35 :10 MET  DST 1997) ready
Name (station : Pol1) :<log name>
```

Nous devons à présent nous identifier auprès du seveur de la station. Et comme on travaille sous le nom de Pol1 c'est ce nom qui est proposé par défaut.

Après, il demande le loging-name pour ce logguer dans un compte. Si on ne veut pas accéder à aucun compte on met ononymous. SI on se loggue on nous demande le mot de passe

```
Password : * * * * *
Login ok acces restrictions apply
ftp>
```

Pour avoir les commandes on met « help » ou « ? ». Pour explorer la station généralement les commandes du shell sont valable ex (ls,dir, ed,cd...,).

Pour quitter le server ftp il suffit d'écrire Bye ou quit à l'invite.

```
*ftp>Bye
211 Goodbye
*ftp>quit
211 Goodbye
Pol1#
```

**Conclusion :**

Etant donné l'importance de l'administration du système et les risques de travailler sous le compte root ,il est conseillé d'utiliser ce compte le moins possible .

Les commandes citées dans ce chapitre sont celles qui sont fréquemment utilisées. Linux dispose d'un manuel qui contient des documentations sur ces diverses commandes et options .

De nombreux éditeurs sont disponible sous Linux . Beaucoup d'utlisateurs préfèrent « emacs » qui dispose d'un langage de commandes dérivé de LISP, avec lequel beaucoup de logiciels ont étés écrits. Mais l'utilisation de « vi » reste indispensable dans des situations telle que la perte du mot de passe du compte root.

Linux supporte des protocoles de communication offrant plusieurs possiblités pour l'environnement du système d'exploitation.

# Chapitre 4
# LINUX Développement

## 4.1 Introduction

Dans ce chapitre nous donnons des exemples de personnalisation ou de développement du système Linux. Nous montrerons la possibilité d'accéder aux scripts système et d'y ajouter des commandes.

Un autre exemple nous permet de voir le fontionnement des appels système MOUNT et UMONT qui nous permettrons de comprendre le fonction des systèmes d'exploitations enfin nous utiliserons le compilateur gcc pour compiler un programme qui fait la conversion analogique numérique de la carte PA300.

## 4.2 Programmation en script

a) Script de configuration du XWindow :

(1) # ! / bin / sh

(2) Lancer xterm

(3) xterm - geometry 40 x 25 + 10 + 100 -fg black -bg white &
xterm -geometry + 200 + 5 -fg black -bg white &
xterm -geometry + 200 + 200 -fg black -bg white &

(4) # autres clients X

(5) oclock-geometry 70 x 70 + 5 + 5 &

(6) xload-geometry 85 x 60 + 85 + 5 &

(7) wsetroot-solid darkslateblue &

(8) # Le gestionnaire de fenêtre

(9) exec fvwm

xterm est appelé avec plusieurs options. Le paramètre-geometry permet de spécifier la taille et la position, son format est le suivant :
<taille x>x<taille y>+<position x>+<position y>

La fenêtre est positionnée aux coordonnées (position x, position y) (où (0,0) est le coin supérieur gauche).

La fenêtre est de 40 caractères de large et 25 caractères de haut. Les arguments -fg et -bg (foreground et background) permettent d'indiquer les couleurs de forme de fond. Si la taille n'est pas précise xterm utilise celle par défaut (80x24).

Pour afficher la liste complète des fontes disponibles sur le système, utilisez la commande ; xlsfonts.

Le client oclock est une simple pendule anologique.

xload affiche sous forme graphique la charge de la machine (nombre de processus en fonctionnement) en fonction du temps.

La ligne (7) remplace la couleur du fond de l'écran par un bleu foncé.

Le gestionnaire de fenêtre fvwm est responsable de la manipulation des fenêtres à la souris, il est lancé par la commande : exec fvwm

Le processus fvwm va remplacer le programme xinit. Ainsi le serveur X se terminera avec l'arrêt de fvwm en utilisant la combinaison de touches <ctrl> <alt> <backspace>.

La figure 4.2 montre l'écran obtenue aprés le lancement de ce programme

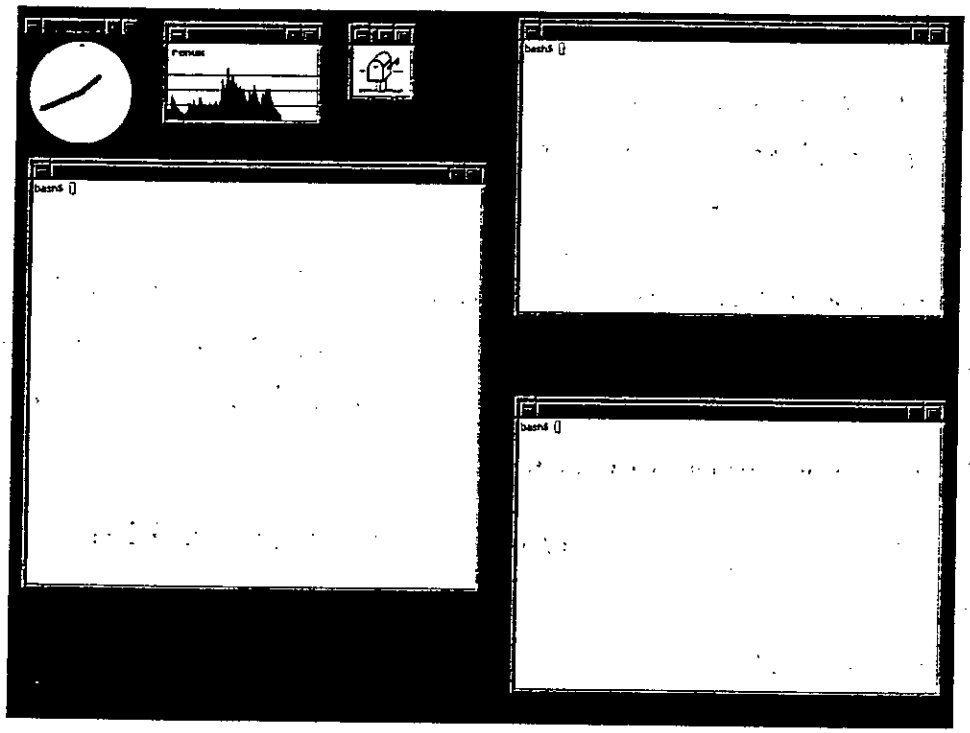


**Figure 4.2** l'écran aprés l'éxecution du programme

**b) Les principaux programmes pour imprimer**

- Le système d'impression d'UNIX comprend cinq programmes au minimum. Ils appartiennent au super-utilisateur et a son grouppe (daemon), ils possèdent les permissions indiquées dans le tableau ...

| Droit d'accès du fichier | Chemin d'accès |
|---|---|
| - rwsr - sr - x | /usr/bin/Ipr |
| - rwsr - sr - x | /usr/bin/Ipq |
| - rwsr - sr - x | /usr/bin/Ipc |
| - rwsr - sr - x | /usr/bin/Iprm |
| - rwsr - s ..... | /usr/bin/Ipd |

Par défaut, les commandes Ipr, Iprm, Ipc et Ipq agissent sur une imprimante appelée Ip. Il existe des pages d'aides pour toutes ces commandes qui nous permettent d'avoir plus d'informations sur les différentes options qu'on peut utiliser.

- Le système est typiquement configuré avec plusieurs répertoires « spool », un pour chaque imprimante.

Ils permettent de stocker les données avant d'être imprimées par / etc/Ipd.
Par exemple, si l'imprimante est appelée ps-nff, elle a pour répertoire de spoule /usr/spool/Ipd/ps-nff.

Chaque répertoire de spoule contient quatre fichiers : **sep, errs, lock** et **status**.
Ces fichiers possèdent les droits d'accès suivants: - rw-rw-r- le fichier **sep** contient un compteur pour l'affectation de numéros de requêtes pour **Ipr**, status contient le message devant être transmis par la commande Ipc stat.

**Lock** est utilisé par Ipd pour s'assurer de n'imprimer qu'un seul fichier à la fois sur une imprimante et errs est un compte rendu des incidents survenus sur l'imprimante.
Le fichier **errs** n'est pas nécessaire et peut s'appeler comme vous le désirez. Le nom est spécifié dans **/etc/printcap**, mais le fichier doit exister de façon que Ipf puisse lui transmettre ses rapports.

• **/etc/printcap** est un fichier texte qui appartient au super-utilisateur.

Une entrée de printcap décrit une imprimante. Elle fournit un nom pour un périphérique physique et décrit de façon de lui transmettre les données.

On peut avoir plusieurs entrées définissant différentes façons de transmettre les données à une même imprimante.

Par exemple, une imprimante physique peut supporter les formats Post Script et HP Laser jet, en fonction de la séquence de caractère envoyée avant chaque travail d'impression.

On définit alors deux imprimantes, chacun traite un format : les programmes générant les données HP les enverront à l'imprimante HP, pendant que les programmes générant des données Post Script les imprimeront sur l'imprimante Post Script.

**Exemple d'entrée de printcap.**

```
# Le format de l'imprimante
HP Laser jet plus

# Ip est le périphérique d'impression (la première imprimante parallèle)
: Ip=/dev/Ip1:\

#sd signifie répertoire « spool » ou les données sont collectées
: sd=/usr/spool/Ip1:\

#If signifie le fichier où les erreurs doivent être rapportées.
lf=/usr/spool/lp1/hp - log:
```

**c) scripts de démarrage du système :**

Dans le répertoire */etc/rc.d,* nous trouvons un ensemble de fichiers concernant la configuration du réseau (tel que *rc.inet1*) et le démarrage du système.

- **Le script rc.local**

(1) # ! /bin/sh

(2)# mettre n'importe quelle commande de démarrage

(3)# Exécuter keyboard map

(4)sh /etc/rc.d/rc.keymap

(5)# Exécuter gpm

(6)echo " Exécution de gpm . . . "

(7)gpm -t &

la ligne (1) de ce script indique le shell utilisé "le BASH shell", les lignes (2),(3) et (5) sont des commentaires,la ligne (4) fait appel à un autre script *rc.keymap* qui lui même fait exécuter le fichier de configuration de clavier.

La ligne 6 affiche sur l'écran 'exécution de gpm'. La commande gpm fait la configuration de la souris (Microsoft par défaut ).

Dans le cas ou la machine est utilisée par un seul utilisateur, ce dernier peut modifier le fichier rc.local en ajoutant des commandes telle que *mc* pour aboutir directement après le démarrage du système au menu "Midnight Commander" qui facilite l'accès au différents fichiers. Dans ce cas on obtient le script suivant :

(1) # ! /bin/sh

(2)sh /etc/rc.d/rc.keymap

(3)echo " Exécution de gpm . . . "

(4)gpm -t &

(5) mc

Pour aboutir directement au Xwindow la commande suivante sera ajouter a la place de mc :

xdm -noborder.

**4.3 Les systèmes de fichiers montés**

Les deux appels système MOUNT et UMOUNT permettent de réunir différents systèmes de fichiers situés sur différents périphériques mineurs de manière à ce qu'ils forment un seul arbre.

**Exemple** :montage d'une disquette du type MSDOS

Le montage de cette disquette se fait par la commande suivante :

mount -t msdos /dev/fd0 /mnt

où /dev/fd0 est le premier lecteur de disquette et /mnt est un répertoire de montage , une fois cette disquette montée elle sera considérée comme un simple répertoire .

Les cas fréquemment rencontrer où le montage échoue sont :

***Premier cas :***

L'utilisateur n'est pas connecté sous le compte root, le système lui renvoie un message d'erreur tel que « Permission denided ».

***Deuxième cas :***

L'utilisateur est connecter sous le compte root, il fait le montage d'une deuxième disquette sans démonter la première. Le système lui renvoi le message « /mnt busy ».

***Troisième cas :***

Erreur de syntaxe et dans ce cas l'erreur est du type drapeau et le shell intervient .

**Le fichier mount.c :** le fichier mount.c contient trois procédures : do -mount, do-umount et name -to - dev.

Avant d'effectuer le montage, la procédure do - mount vérifie qu'on n'est pas en présence d'erreurs telles que:

1- La commande mount n'est pas effectuer par le super-utilisateur

2- Un périphérique est déjà monté

3- Le système de fichiers à monter n'existe pas ou est un fichier spécial.

4- Le fichier spécial à monter a un mauvais numéro magique (magic number)

5- Le système de fichiers à monter est invalide ( par exemple, il ne possède pas de noeud d'information).

6- Il n'y a plus de place pour le super-bloc du système de fichiers à monter.

7- Il n'y a plus de place pour le noeud d'information de la racine du système de fichiers à monter.

En cas d'erreurs le super- bloc et les inodes sont libérés. Si il n'y a pas d'erreur le montage du système de fichier est possible, il est effectué en initialisant deux pointeurs dans le super-bloc(rip et root-ip), L'un d'eux pointe sur le noeud d'information où le système de

fichiers est monté et l'autre sur le noeud d'information de la racine du système de fichiers monté.

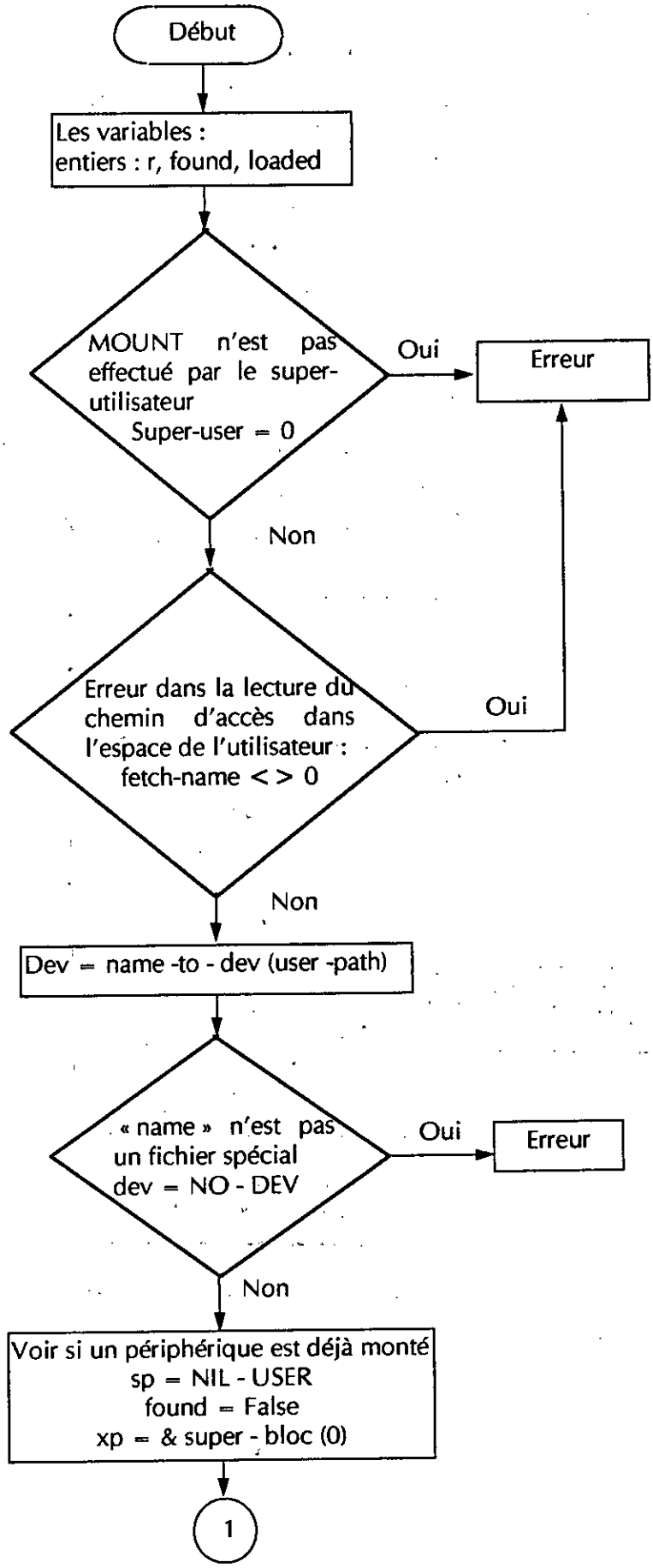
Le démontage d'un système de fichiers est plus simple que le montage: il faut s'assurer qu'aucun processus ne possède de fichiers ouverts ou un répertoire de travail dans un système de fichiers à démonter.

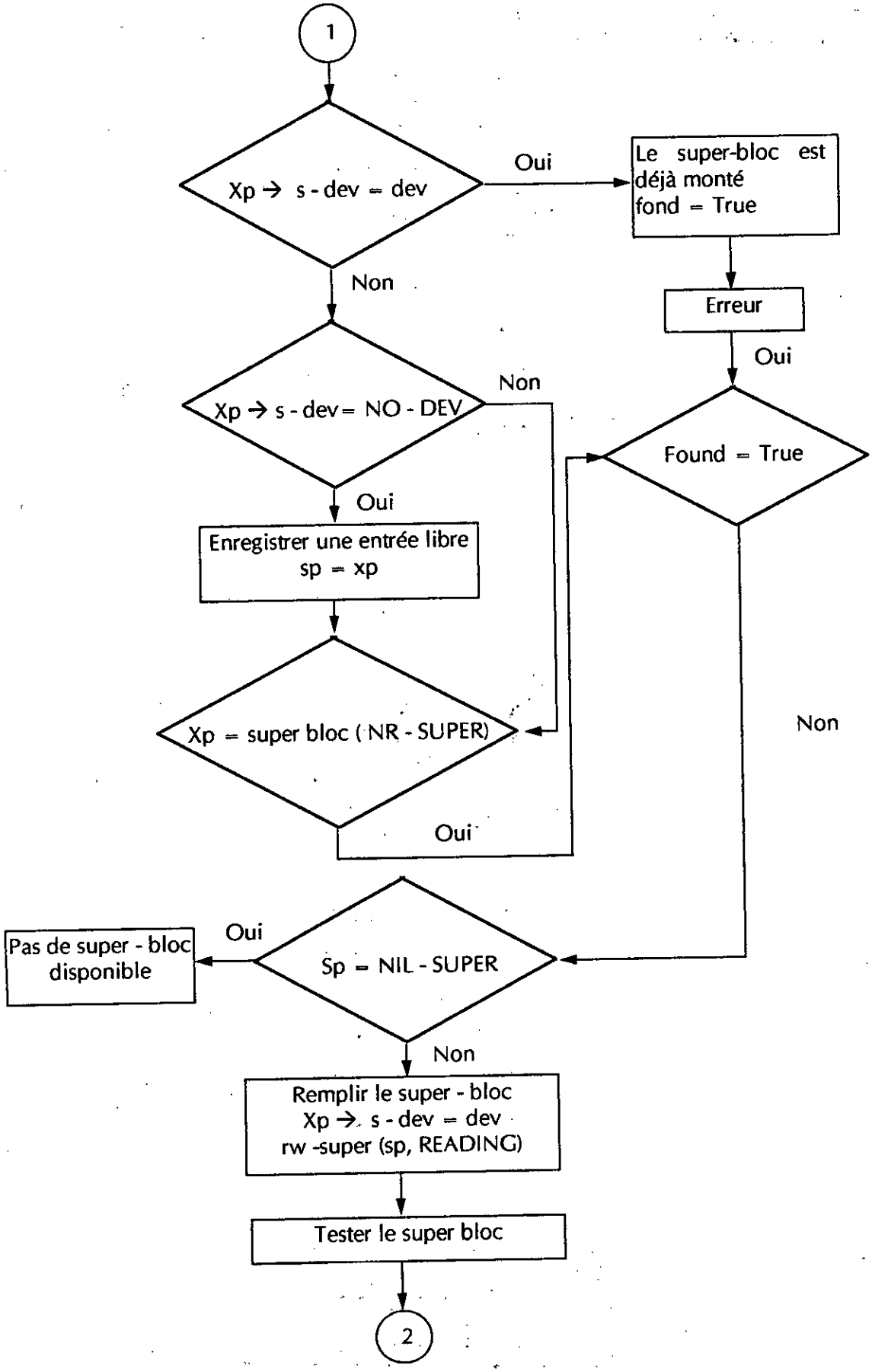
La vérification se fait en parcourant toute la table des noeuds d'information pour voir si aucun noeud n'appartient au système de fichiers à retirer à l'exception du noeud d'information de la racine. Dans le cas contraire l'appel UMOUNT échoue.

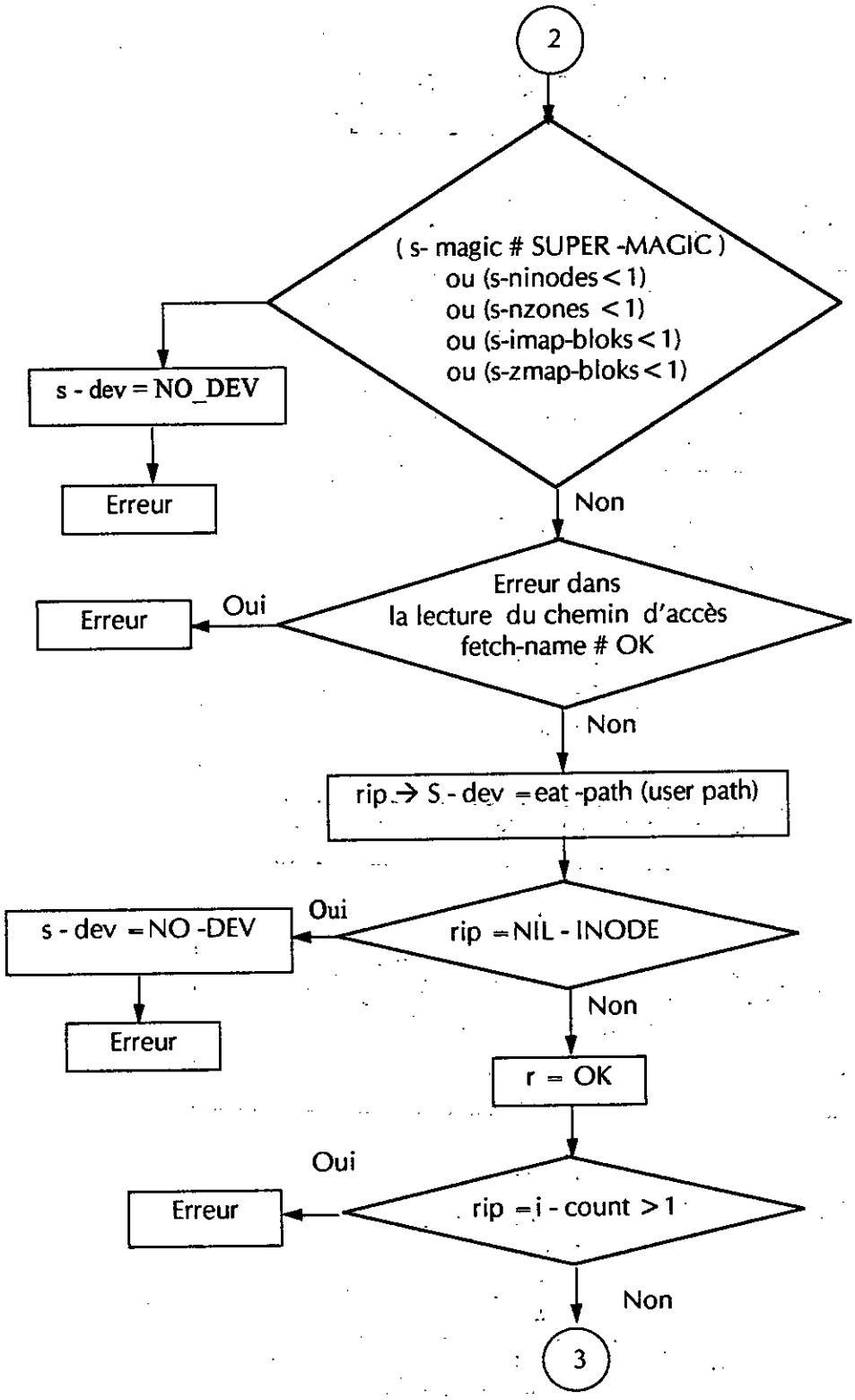
La procédure name-to-device prend un fichier spécial, détermine son noeud d'information et extrait ses numéros de périphérique (majeur et mineur) .

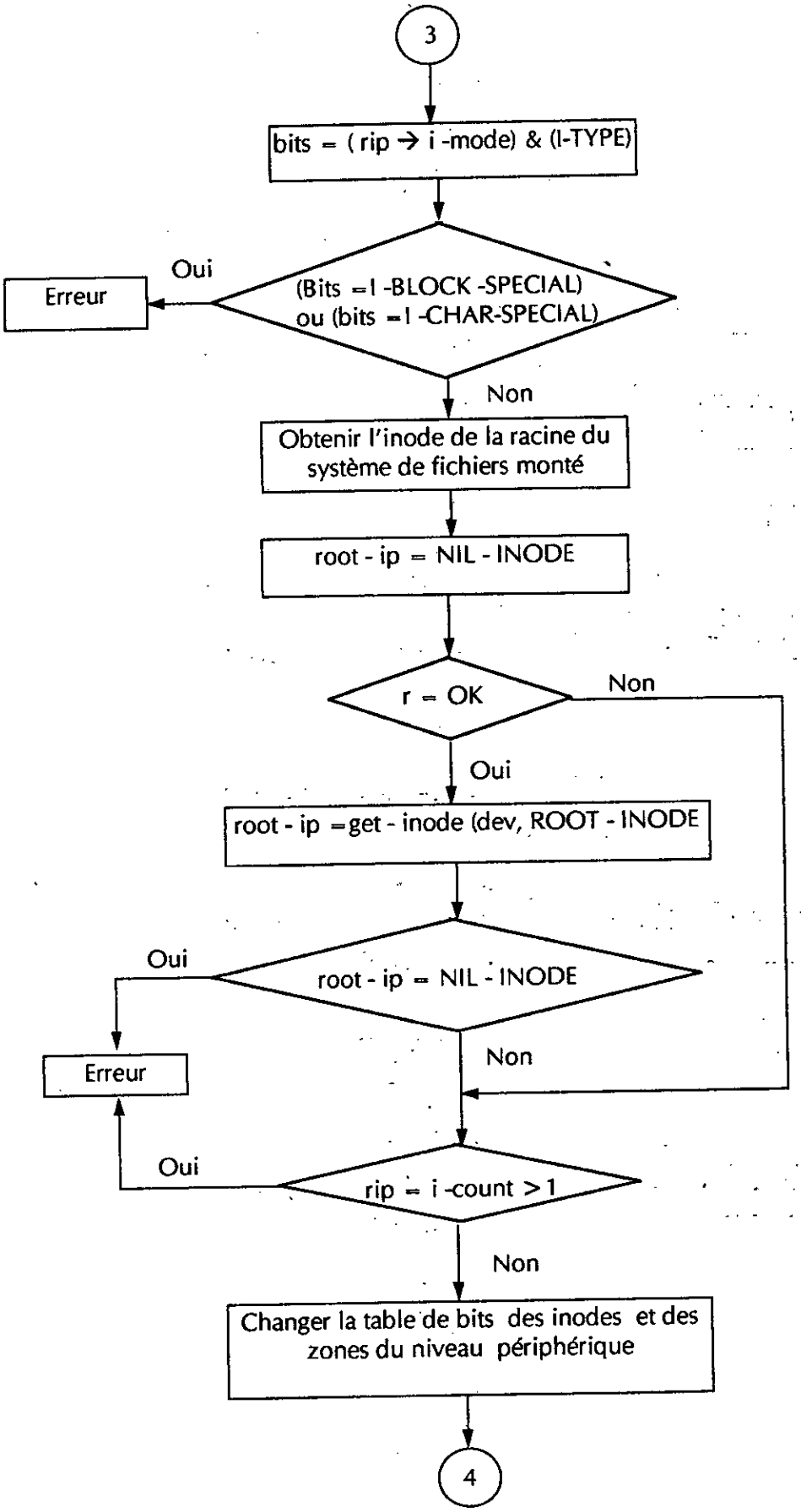
Afin de donner plus de précision au fonctionnement de ces appels système nous présentons les organigrammes suivant.

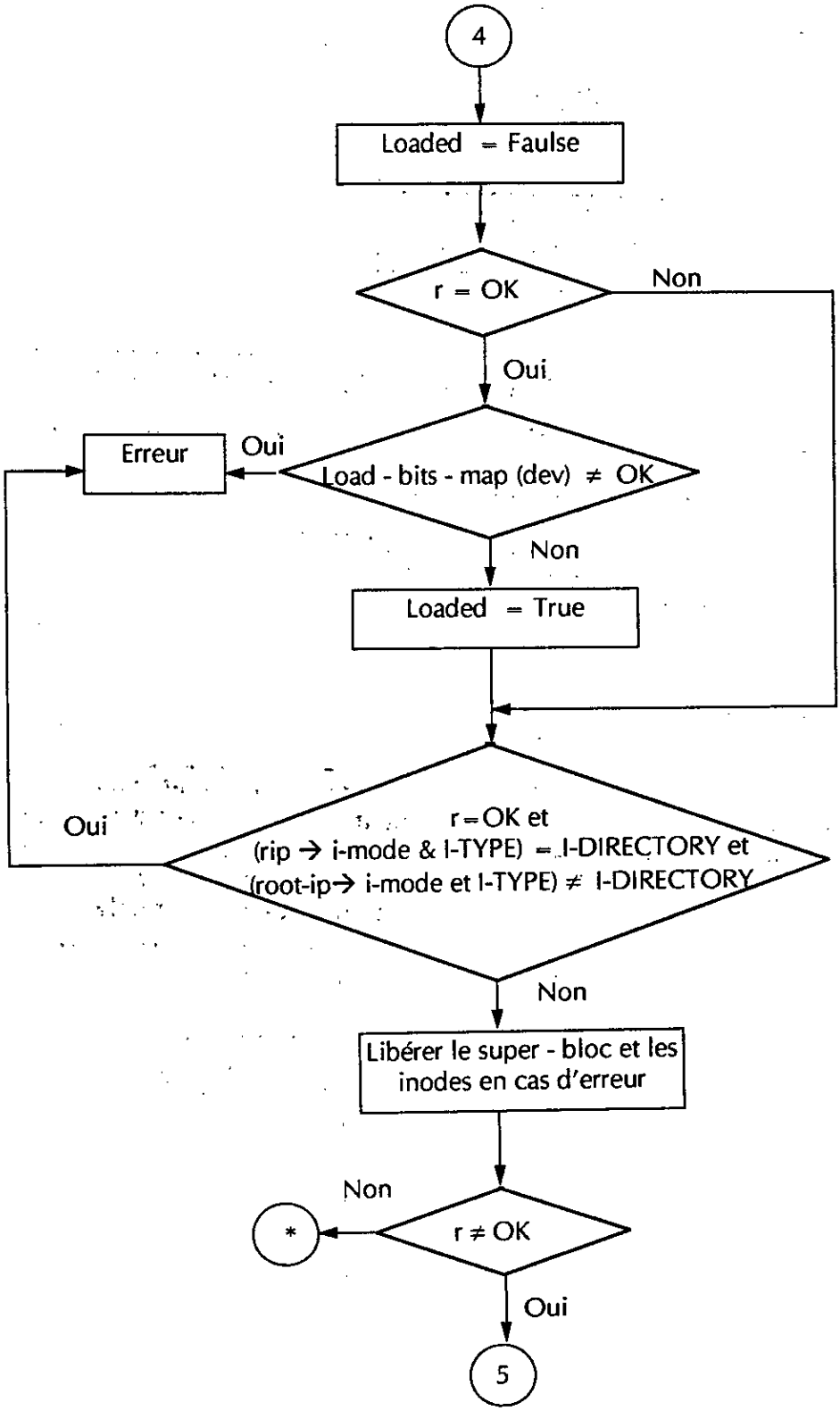
- **L'appel système MOUNT**

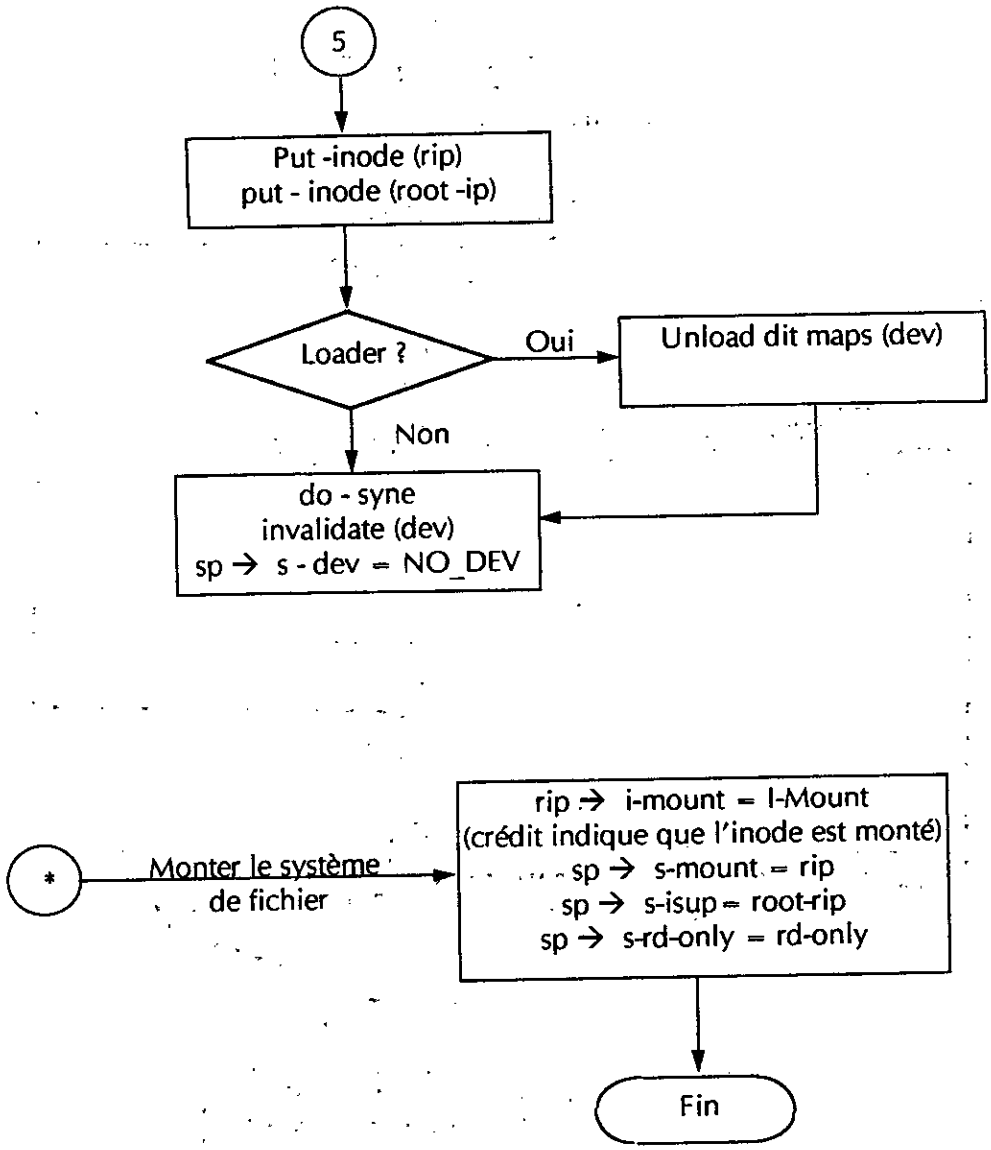



1
Xp → s - dev = dev
Oui
Le super-bloc est déjà monté
fond = True
Non
Erreur
Oui
Xp → s - dev = NO - DEV
Non
Found = True
Oui
Enregistrer une entrée libre
sp = xp
Xp = super bloc ( NR - SUPER)
Non
Oui
Pas de super - bloc disponible
Oui
Sp = NIL - SUPER
Non
Remplir le super - bloc
Xp → s - dev = dev
rw -super (sp, READING)
Tester le super bloc
2

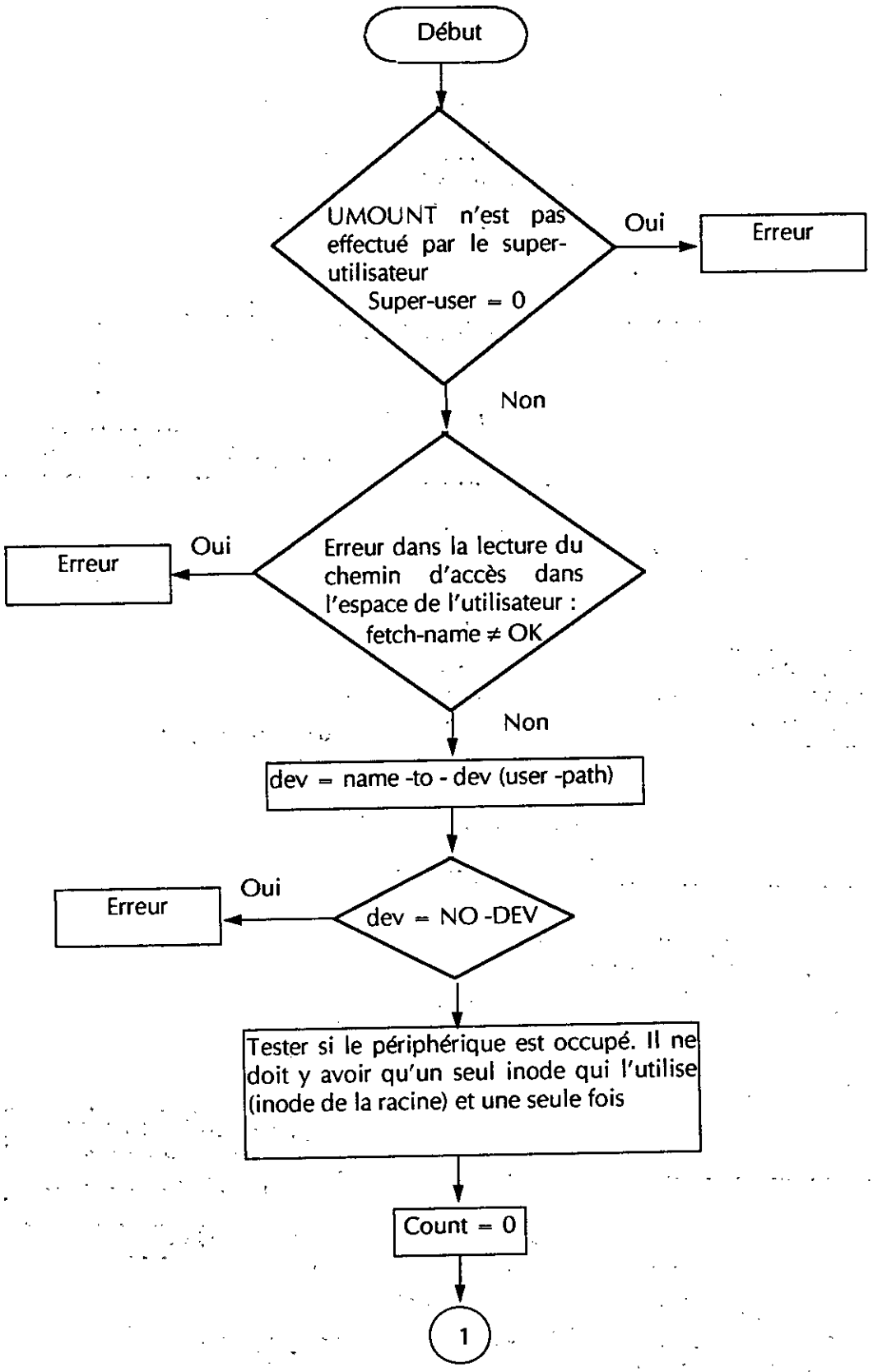

2
( s- magic # SUPER -MAGIC )
ou (s-ninodes < 1)
ou (s-nzones < 1)
ou (s-imap-bloks < 1)
ou (s-zmap-bloks < 1)
s - dev = NO_DEV
Erreur
Non
Erreur dans
la lecture du chemin d'accès
fetch-name # OK
Oui
Erreur
Non
rip → S - dev = eat -path (user path)
rip = NIL - INODE
Oui
s - dev = NO -DEV
Erreur
Non
r = OK
rip = i - count > 1
Oui
Erreur
Non
3

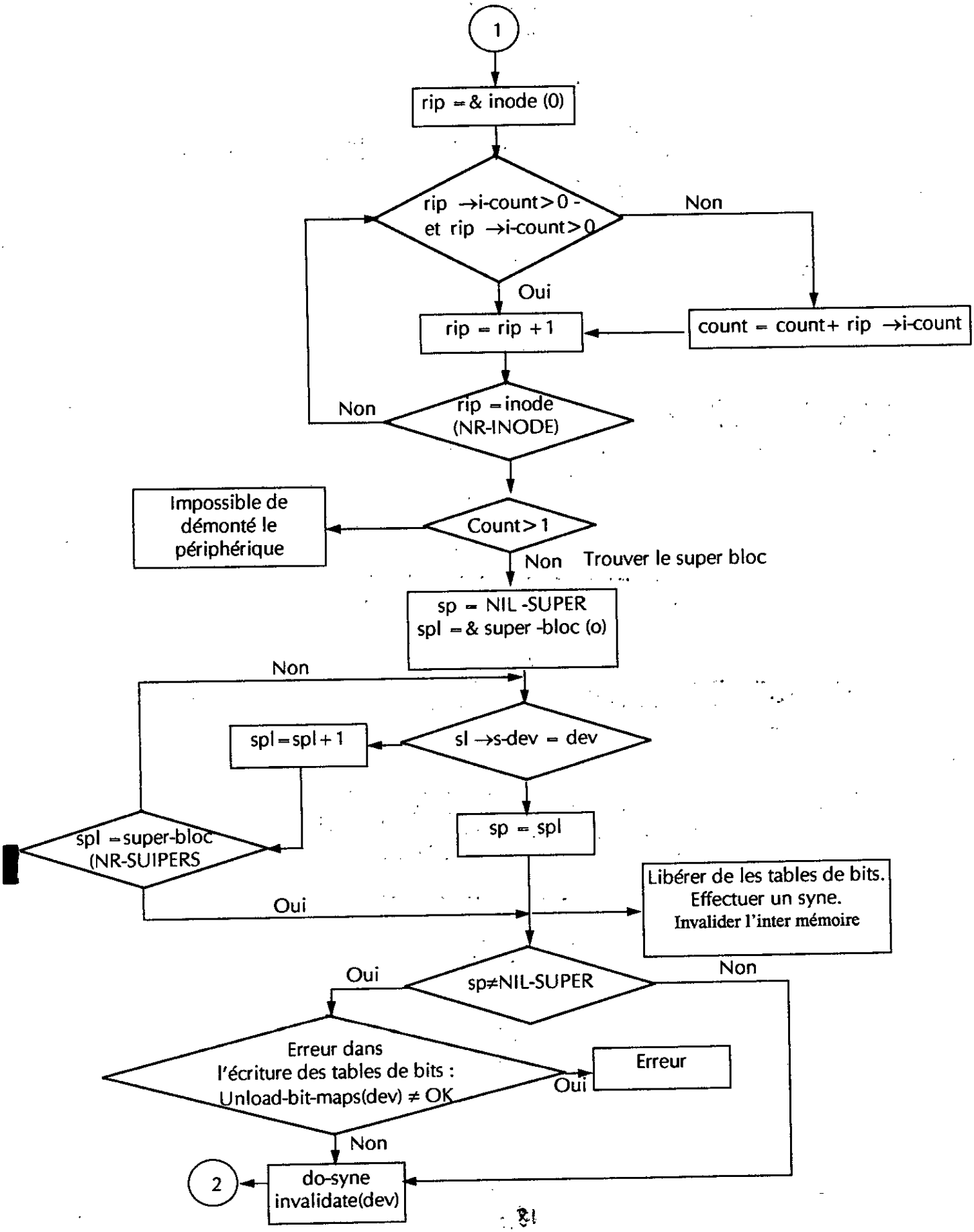

3
bits = ( rip → i -mode) & (I-TYPE)
(Bits =I -BLOCK -SPECIAL)
ou (bits =I -CHAR-SPECIAL)
Oui
Erreur
Non
Obtenir l'inode de la racine du système de fichiers monté
root - ip = NIL - INODE
r = OK
Non
Oui
root - ip =get - inode (dev, ROOT - INODE
root - ip = NIL - INODE
Oui
Erreur
Non
rip = i -count >1
Oui
Non
Changer la table de bits des inodes et des zones du niveau périphérique
4

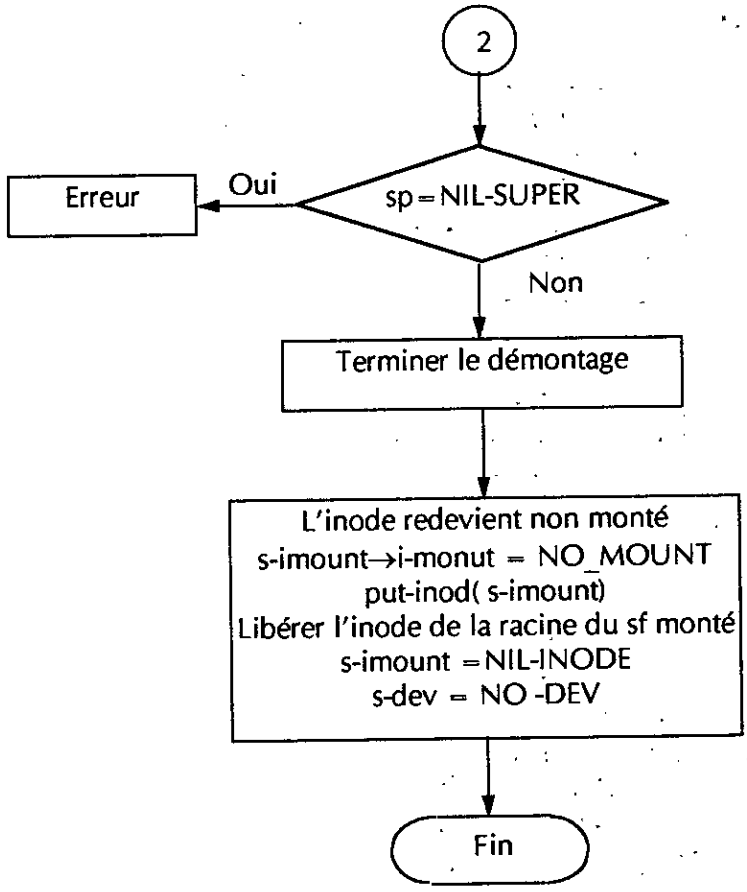

4
Loaded = Faulse
r = OK
Non
Oui
Erreur
Oui
Load - bits - map (dev) ≠ OK
Non
Loaded = True
Oui
r=OK et
(rip → i-mode & I-TYPE) = I-DIRECTORY et
(root-ip→ i-mode et I-TYPE) ≠ I-DIRECTORY
Non
Libérer le super - bloc et les inodes en cas d'erreur
Non
*
r ≠ OK
Oui
5


5
Put -inode (rip)
put - inode (root -ip)
Loader ?
Oui
Unload dit maps (dev)
Non
do - syne
invalidate (dev)
sp → s - dev = NO_DEV
*
Monter le système de fichier
rip → i-mount = I-Mount
(crédit indique que l'inode est monté)
sp → s-mount = rip
sp → s-isup = root-rip
sp → s-rd-only = rd-only
Fin

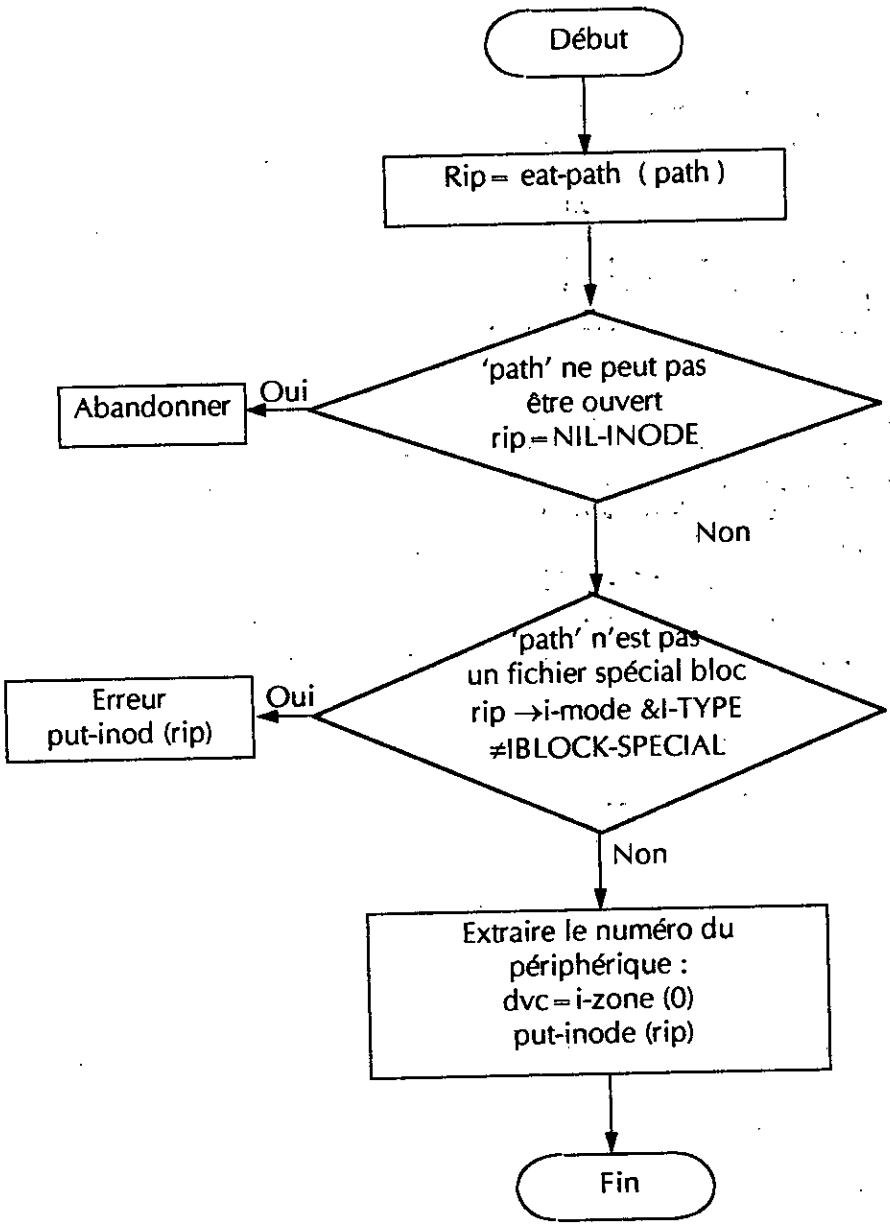
- **L'appel système UMOUNT**




1
rip =& inode (0)
rip →i-count>0 -
et rip →i-count>0
Non
Oui
rip = rip +1
count = count+ rip →i-count
Non
rip =inode
(NR-INODE)
Impossible de
démonté le
périphérique
Count>1
Non
Trouver le super bloc
sp = NIL -SUPER
spl =& super -bloc (o)
Non
spl=spl+1
sl →s-dev = dev
spl =super-bloc
(NR-SUIPERS
sp = spl
Libérer de les tables de bits.
Effectuer un syne.
Invalider l'inter mémoire
Oui
Oui
sp≠NIL-SUPER
Non
Erreur dans
l'écriture des tables de bits :
Unload-bit-maps(dev) ≠ OK
Erreur
Oui
Non
2
do-syne
invalidate(dev)


2
sp = NIL-SUPER
Oui
Erreur
Non
Terminer le démontage
L'inode redevient non monté
s-imount→i-monut = NO_MOUNT
put-inod( s-imount)
Libérer l'inode de la racine du sf monté
s-imount = NIL-INODE
s-dev = NO -DEV
Fin

- **L'appel système name-to-dev**



Liste des noms de procédures, variables globales, constantes et macro-instructions (du code source) utilisées dans le fichier *mount.c*

**eat-path** : la routine principale du mécanisme de conversion chemin/ilode

**do-sync** : recopie toutes les tables sur disque

**fetch-name** : lit un chemin d'accès dans l'espace de l'utilisateur

fetch-name (path, len, flag)

char *path pointeur sur le chemin d'accès
int len longueur du chemin, octet à 0 compris
int flag si flag=M1,M2 ou M3 le chemin peut être dans un message

**get-inode** : recherche un inode dans la table

**i-count** : nombre de fois l'inode est utilisé (0 si entrée libre)

**i-dev** : périphérique qui contient l'inode

**i-mode** : type de fichier

**invalidate**: supprime tous les blocs de l'antémémoire(*) d'un périphérique

**load-bit-map**: lit la table du périphérique racine ou nouveau monté

**name to dev**: convertit le fichier spécial bloc 'path' en un numéro de périphérique

**put-inode** : indique qu'un inode n'a plus à rester en mémoire

**rd-only** : variable de messade en entrée

**rw-super** : lit ou écrit un super-bloc

**s-dev** : à qui appartient le super utilisateur

**s-imount** :inode monté

**s-isup** : inode du répertoire racine des systèmes de fichiers

**s-imap-blocks**: nombre de blocs utilisés par tables bits inode

**s-magic**: nombre magic pour reconnaitre le super-bloc

**s-ninodes**: nombre d'inodes utilisable sur le périphérique

**s-nzones** : taille totale du périphérique

**s-rd-only** : mis à 1 si le système de fichiers est monté en lecture uniquement

**super-blocks** : numéro de bloc du super-bloc

**super-user** : super-user = 1, si l'appelant est le super utilisateur, 0 sinon

**s-zmap-blocks**:nombre de blocs utilisés par table bits zone

(*) L'antémémoire est une suite de blocs qui appartiennent logiquement au disque, mais qui sont situés en mémoire principale pour améliorer les performance [1]

**user-path** : chemin d'accès de l'appelant

**unload-bit-map**: écrit les tables de bits sur le disque après un MOUNT

| | |
|---|---|
| **FALSE** | 0 pour convertir les entiers en booléens |
| **I-BLOCK-SPECIAL** | fichier spécial bloc |
| **I-CHAR-SPECIAL** | fichier spécial caractère |
| **I-TYPE** | ce champ donne le type de l'inode |
| **NIL-INODE** | indique l'absence d'entrée inode |
| **NIL-SUPER** | utilisé si le super bloc se trouve en mémoire |
| **NO-DEV** | indique l'absence d'un numéro de périphérique |
| **NR-INODES** | nombre d'entrée dans la table d'inode |
| **NR-SUPERS** | nombre maximale de blocs dans la table super-bloc |
| **OK** | indique qu'il n'y a pas d'erreur |
| **READING** | lecture de données de disque |
| **ROOT-INODE** | numéro d'inode du répertoire racine |
| **SUPER-MAGIC** | nombre magique contenu dans le super-bloc |
| **TRUE** | 1 pour convertir les entiers en booléens |

### 4.4 Pilotage de la carte PA 300

#### 4.4.1 Introduction

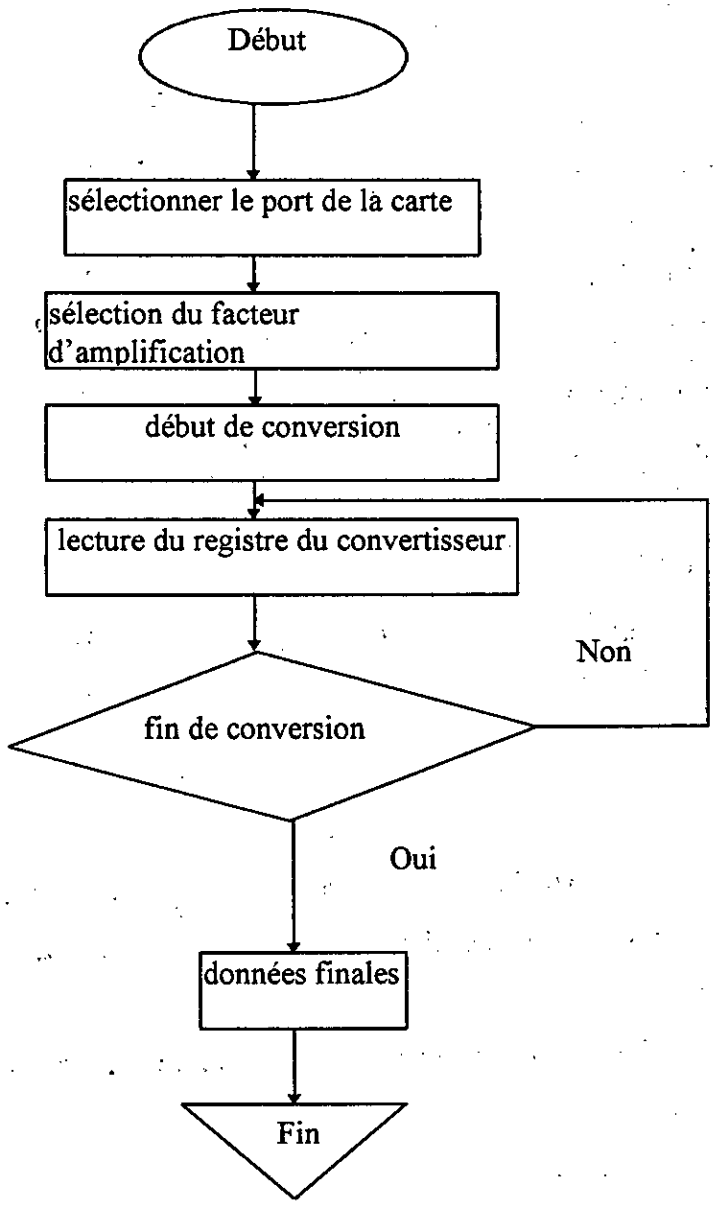
La carte PA 300 est un système de saisie analogique. Elle peut être utilisée dans les systèmes de commande de processus ou de machines, en laboratoire et pour des systèmes de test, pour la mesure en technique médicale. Grâce aux dispositifs de protection, filtres sur les entrées, ainsi que les entrée en courant 20mA, la carte PA 300 peut servir en milieu extrêmement exposé aux parasites. Un amplificateur de signal programmable a été prévu pour la saisie de signaux très faibles.

#### 4.4.2 Conversion analogique / numérique

La fonction principale de la carte PA 300 est la conversion analogique numérique. Elle a été équipée avec un module de saisie SDM 854 qui offre de nombreuse possibilités.

Jusqu'à 16 signaux analogiques peuvent y être branchés par l'intermédiaire d'un connecteur MIN-D à 37 poles.

Dans la plupart des applications, les différentes voies sont sélectionnées et converties au fur et à mesure. Le déroulement de la conversion est le suivant: sélection de la voie et du facteur d'amplification, démarrer la conversion, attendre la fin de conversion, lire les données. L 'organigramme suivant exécute ces fonction:




La conversion ANA / NUM est lancée 50 ns après l'adressage de la carte ( et le choix du facteur d'amplification ) si une seule carte PA 300 est utilisée dans l'ordinateur, deux adresses d'entrées-sorties sont occupées . Les adresses pour chaque carte doivent être sélectionnées de manière à ne pas y avoir de recouvrement .

Pour la visualisation des signaux acquis , on utilise une des précieuses application sous LUNIX le « GNUPLOT » similaire au « MATLAB » sous MSDOS ou OS/2. Le fichier script «Aquit.gnu » est un programme GNU PLOT qui permet la visualisation du signal aquit déjà trouver dans le fichier«Aquit.gnu » .

Programme AQUIT .GNU

```
# Choisir les valeurs max des axes
Set simples 50
Set title « résultats de l'acquisition »
plot 'aquit.dat' with lignes 1 3
```

Bien sûr, pour avoir accès au GNUPLOT il faut débuter une session X window, avoir un terminal graphique ( SHELL), taper GNUPLOT et puis taper la commande suivante « Set term X11 » pour choisir un terminal après lancer la commande suivante : load ' Aquit . gnu ' ou aquit gnu est le programme de visualisation cité ci-dessus .

Set simples 50 : valeurs comprises entre 50 et 50

Set title : le titre du schéma

Plot 'aquit.dat' : visualiser le fichier d'acquisition avec des lignes et avec la carte rouge type3.

## 4.5 Conclusion

La modification ou l'ajout de commande dans les scripts de démarage du système ou bien des scripts de démarrage des applications permettent à l'utilisateur de travailler dans un environnement satisfaisent .

Le développement sous Linux consiste le plus souvent à écrire des programmes en langage C et à utiliser des fonction

disponibles dans les bibliothèques.

Le fonctionnement de l'appel système MOUNT représenté dans l'organigramme ci-dessus est un exemple de développement de Linux.

Afin d'ajouter une nouvelle commande, il est intéressant d'utiliser les procédures qui existent déjà dans le système tels que "fetch-name" qui fait la lecture du chemin d'accès dans l'espace

de l'utilisateur, et "eat path" qui la routine principale du mécanisme de conversion chemin/inode.

Il est essentiel de tester avec précision la réussite de chaque appel système. Des constantes symboliques associées à chaque type d'erreur dont définies dans <sys/error.h>.

Le pilotage de la carte PA300 n'a pas fait l'objet d'un vrai pilote (Driver) car dans notre cas on n'a pas pris en compte tout le mécanisme des interruptions et des erreurs du a la violation de priorité .

# Conclusion

# Conclusion

L'installation et la mise en oeuvre du système LINUX sont les deux principaux objectifs de notre travail, pour présenter Linux on a commencé par donner un aperçu général des systèmes d'exploitation , on s'est intéressé en particulier dans notre étude au système UNIX, du fait que LINUX est une amélioration d'UNIX sur PC .

Linux, clone gratuit d'UNIX, comporte un grand nombre de commandes et de logiciels d'applications (parmi eux des applications volumineuses), nous avons omis de citer certains parmi eux faute de documentation .

L'installation de LINUX a été faite en plusieurs versions et plusieurs fois en général toutes ces installations ont été remise en cause (mauvais choix du noyau). La dernière installation a pris en compte toutes les possibilités qu'offre le matériel, un noyau relativement complet a été installé et là l'installation a été meilleure si on prend en compte les différentes applications fonctionnelles .

Nous avons présenté le plus important des outils d'application pour permettre aux utilisateurs du système de pouvoir le gérer correctement .

Grâce aux applications réseau sous LINUX, il est possible d'installer un réseau local de machines, et même d'accéder au réseau mondial INTERNET.

Le **bash;** shell par défaut de LINUX, offre de nombreux mécanismes pour personnaliser l'environnement . Il ne s'agit pas seulement d'un interpréteur de commandes pouvant être saisies, il fournit aussi un langage de programmation assez puissant. Dans notre cas,on s'est limité à donner quelques exemples à titre indicatif.

Les possibilités de développement qu'offre Linux sont nombreuses, nous avons exploité le code source de Linux afin de mieux comprendre les systèmes d'exploitation et d'envisager les possibilités de développement .

# BIBLIOGRAPHIE


[1] A.Tanenbeaum, « Les systèmes d'exploitation et mise en oeuvre », inter édition, Paris, 1989

[2] M.Wielsh, « Le grand livre d'UNIX », Micro application, Paris 1995

[3] M.Griffiths & M.Vayssade, « Architecture des systèmes d'exploitation », Edition Hermès, Paris, 1990

[4] Paul Feautrier, « Organisation des systèmes d'exploitation », Techniques d'ingénieur, H3110

[5] Guy Pjolle « Télématique réseau et application » Edition Eyrolles, Paris, 1986

[6] Matt Welsh & Lar Kaufman, « Le système Linux », Edition O'Reilly International Thomson, Paris, 1995

[7]Jack Takette, Jr. David Gunter, Lancer brown, « LINUX », Edition S&SM, Paris, 1995

[8]Daniel A.Tauber & Matt Welsh, « Linux mode d'emploi »,SYBEX Paris, 1996

[9]Christion Pelissier, « Utilisation et administration du système UNIX », Edition HERMES, Paris 1992

[10]R.Thomas &J.Yates, « Auser guide to the Unix system », Edition Bell, London, 1994

[11]Stene Bourne, « Le système UNIX », Edition Bell, Paris, 1989

[12]Douglas Comer, « Operating system designe : the xinu approach », Prentice-Hall International Editions, London, 1984

[13] BACH.J. Maurice, « Conception du système UNIX », Edition Masson, Paris, 1991

[14]Jean.R.Chamière, « Développer sous UNIX; outils pour la production de logiciel », E.D.P.S.I., Paris 1989

[15]J.YATES « Administration du système UNIX », Edition Hermes, Paris, 1994

[16]Application data « Descripteur technique, Mode d'emploi de la carte PA300

[17] M.Gabassi, B.Dupuyb « L'informatique répartie sous UNIX », Edition EYROLLES, Paris 1987

[18] M.J. BACH, « Conception du système UNIX », Edition MASSON &AT&T, Paris .

# ANNEXES

# ANNEXE A

## Installation de Linux

### Préparation -disquette d'amorçage

Pour installer Linux, il est nécessaire de répartitionner le disque dur afin de faire de la place au nouveau système d'exploitation.

- Créer une disquette d'installation MS DOS qui contient les fichiers CONFIG.SYS, AUTOEXE.BAT et tous les fichiers utile au démarrage de l'ordinateur ainsi que les fichiers FDISK.EXE et FORMAT .EXE qui permetteront de répartitonner le disque dur et le reformater.

- Un disque DOS est nécessaire pour la restauration du DOS.

-Pour sauvegarder tous les fichiers du disque C: sur le lecteur A:, on utilise la commande backup c: a:. Cette opération permet de réccupérer les données perdues pendant le partitionnement du disque dur.

- Créer les disques d'initialisation et de base avec rawrite

Le disque d'initialisation est utilisé pour démarrer le système Linux à l'installation. Il contient les pilotes de matériel dépouillé et une version élémentaire du système d'exploitation. Les divers fichiers sont reportés dans le tableau a.1.

| Nom | Description |
|---|---|
| bare.gz | Ne contient que les pilotes pour les disques durs IDE. N'utilisez ce disque que si vous n'avez pas l'intention d'installer un lecteur CD ROM. |
| xt.gz | Contient les supports de disques durs IDE et XT. |
| cd.gz | Contient les pilotes de disques durs IDE et les lecteurs de CD ROM non SCSI. |
| cdscsi.gz | Contient le support pour disque dur IDE et les pilotes de CD ROM non SCSI. |
| scsi.gz | Contient le support pour disques durs IDE et SCSI. |
| net.gz | Contient le support pour disques durs IDE et les réseaux TCP/IP. |
| scsinet.gz | Contient le support pour disques durs IDE et SCSI, les pilotes de CD ROM SCSI et le support pour réseaux TCP/IP. |
| sbpcd.gz | Content le support IDE et CD ROM de Creative Labs Sound Blaster. |

**Tab A.1** Les fichiers d'initialisation

Si ces fichiers sont compactés, utilisez le programme gzip pour les décompacter.

Le disque de base est nécessaire pour la création d'un système de fichiers élémentaire pour la première initialisation de Linux. Pour créer le disque de base on a besoin de l'un des fichiers mentionnés dans le tableau a.2

| Nom | Description |
|---|---|
| color144.tgz | Le disque d'installation couleur avec menu pour les lecteurs de 1,44 Mo. |
| umsds144.tgz | Une version du disque color144 à installer avec le système de fichier UMSDOS, qui permet d'installer Linux sur le répertoire d'un système de fichier MS DOS. |
| tty144.tgz | Le disque d'installation sur terminal pour les lecteurs 1,44 Mo. A utiliser si color144 ne fonctionne pas. |
| colrlite.tgz | Le disque d'installation sur terminal pour les lecteurs 1,2 Mo. |
| umsdos12.tgz | Une version du fichier colrlite pour installer le système de fichier UMSDOS. |
| tty12.tgz | Le disque d'installation sur terminal pour lecteurs 1.2 Mo. A utiliser si colrlite.tgz ne fonctionne pas. |

**Tab a.2** Les fichiers racine

On transfert le fichier d'initialisation et le fichier de base sur une disquette ou dans un répertoire qu'on appellera -par exemple - Linux. On a aussi besoin du programme rawrite qui se trouve dans le répertoire dos-util.

Exemple:

On choisit une disquette d'initialisation 3,25 de 1.44 Mo. A l'exécution de rawrite on obtient:

C:\ Linux> Rawrite

*RaWrite1.2-Write disk file to raw floppy disquette*

*Enter source file name: Color144*

*Enter destination drive: A:*

*Please insert a formated disquette into drive A: and press- Enter A:*

*Number of sectors per track for this disk is 18*

*Writing image to drive A/, press ^C to abort*

*Track:01 Head:1 sector :10*

Rawrite affiche le déroulement de l'opération tandis que le programme écrit l'image sur la disquette.

Après avoir écrit le disque d'initialisation on utilise une autre disquette pour écrire le disque de base. A l'exécution de rawrite, on utilise le nom de fichier image (tel que *bare* ou *scsi*) comme nom de fichier source au lieu du nom de fichier image d'initialisation (*color144*).

-Utiliser le progiciel Lininst:

Dans le répertoire \dos-util, il existe un programme Microsoft Windows, Lininst, pour automatiser la procédure de création des disques d'initialisation et de base.

- Partitionner le disque dur

Les partitions sont créées, détruites et gérées par un programme qui porte généralement le nom de FDISK.

On utilise le FDISK du DOS pour partitionner le disque DOS. Ensuite, on utilisera la version Linux fdisk pour créer les partitions Linux.

Après le répartitionnement du disque dur, on prépare la nouvelle partition du DOS (partition primaire):

* Initialiser l'ordinateur.
* Formater le disque approprié.
* Transférer les fichiers systèmes.

- Préparer le disque pour Linux

1- Initialiser Linux depuis les disques d'initialisation et de base.

2- Utiliser fdisk pour créer les partitions nécessaires:

A l'invite #, entrez *fdisk <entrée>*.

A l'invite fdisk, entrez m pour obtenir la liste des commandes.

La commande p imprime le tableau des partitions.

Pour aiouter des partitions la commande n affiche:

*Command Action*

*e extended*

*p primary*

Appuyer sur <*p*> <*entrée*>. fdisk demande alors le numéro de la partition qu'on veut créer. Par exemple, entrez 3 pour ajouter une troisième qui sera désignée par /dev/hda3.

Ensuite fdisk demande quel sera l'emplacement du premier cylindre en affichant par exemple:

First cylinder (42-1024)

Il est conseillé de choisir le premier emplacement disponible, dans ce cas entrer 42 <entrée>.

Maintenant fdisk demande de spécifier l'espace alloué à la partition:

Last cylinder or + size or M or + size K (42-1023):

Cet espace peut être exprimé en cylindre, en nombre d'octets, kilo-octets ou Méga-octets.

Par exemple, pour une machine de 8 Mo de RAM, on a besoin d'une partition de 16Mo en répondant de la manière suivante: +16M<entrée>.

On utilise la commande p pour vérifier le nouveau tableau des partitions.

- Créer une partition de permutation:

Pour créer un espace de permutation utilisez la commande mkswap en lui indiquant la partition à utiliser et la taille de la RAM virtuelle.

Par exemple, pour créer un espace de permutation sur la partition/dev/hda3 entrez la commande suivante à l'invité #:

#mkswap - c/dev/hda3 16447

Le nombre 16447 représente 16 Mo

L'option -c indique à mkswap de vérifier s'il n'y a pas de secteurs défectueux sur la partition.

Activer le système de permutation avec la commande swapon:

# swapon/dev/hda3

## Installer le système Linux:

Pour lancer l'installation entrez setup <entrée>

L'écran offre la sélection de menus suivante:

| | |
|---|---|
| Help | Lire le fichier slackware Setup Help |
| KEYMAP | Connecter le clavier si vous n'utilisez pas un clavier americain |
| QUICK | Choisir le mode d'installation rapide ou verber |
| MAKE TAGS | Les experts peuvent personnaliser les fichiers d'adresses pour présélectionner les logiciels |
| ADDSWAP | Installer la ou les partitions de permutation |
| TARGET | Installer les partitions cibles |
| SOURCE | Sélectionner le support source |
| DISKSETS | Décider quels jeux de disques installer |
| INSTALL | Installer les jeux de disques |
| CONFIGURE | Reconfigurer le système Linux |
| EXIT | Sortir de slackwore |

**1. La sélection ADDSWAP**

Il est plus facile de laisser le programme setup se charger de la création de la partition de permutation en sélectionnant ADDSWAP.

**2. La section TARGET**

Setup affiche les informations sue les partitions, ensuite vous demande de choisir un système de fichier. L'installation du système de fichier ext2 est recommandée. Setup vous demande alors de formater la partition choisie pour l'installation. Pour la densité d'inode, sélectionnez 2048 et cliquez OK.

**3. La selection SOURCE**

Cette sélection permet de choisir le support à partir duquel vous allez effectuer l'installation et le répertoire source (distribution/slackware).
Les logiciels sont sélectionnés à l'aide de la barre d'espacement.

Après la sélection des logiciels, vous entrez dans la sélection INSTALL, si vous n'avez fait pas les bonnes sélections, le programme vous renvoie au menu principal.

Le programme setup demande maintenant d'entrer le type d'invité que vous voulez voir apparaître pendant que le setup parcour le programme d'installation de chaque logiciel. Sélectionnez le mode d'invité Normal.
Sélectionnez ensuite INSTALL.

**4. Configurer le système**

Le programme setup a terminé le chargement de tous les logiciels que vous avez sélectionné. Vous devez maintenant configurer le système:

1- Créer un disque d'initialisation
2- Configurer le modem
3- Configurer la sourie
4- Installer LILO
5- Configurer le réseau
6- Configurer le sendmail (utilisation de courrier électronique)
7- Configurer le fuseau horaire

Lorsque vous aurez terminé l'installation et la configuration du système, le programme setup reviendra au menu principal.

**5.Installation du système X Window**

Tout d'abord, on doit s'assurer d'avoir le matériel approprié pour exécuter X Window, avoir suffisamment de mémoire et d'espace sur le disque (environ 21Mo d'espace sur le disque pour installer le système Xfree86 et les applications X Window). Il faut au moint 16 Mo de mémoire virtuelle pour exécuter Xwindow. La mémoire virtuelle est une combinaison de la RAM physique du système et de la quantité d'espace de permutation qu'on a allouée à Linux. Dans tous les cas on doit avoir avoir au moins 4Mo de RAM physique pour exécuter Xfree86 sous Linux ce qui nécessite une copie sur disque de 12Mo. Il est nécessaire d'avoir une carte vidéo contenant un jeu de puces de pilotes vidéo supporté par Xfree86.
Pour l'installation , on doit être super-utilisateur, soit en se connectant sur le compte root, sit en utilisant la commande *su* depuis le compte utilisateur.

La commande setup permet permet de selectionner a partir d'un menu principal la source d'installation, puis on demande l'installation de l'ensemble X et eventuellement XAD qui contient un certain nombre d'applications utilisant X Window.

**6 Amorçage du système [7]**

Il existe plusieurs façons d'amorcer Linux, soit depuis une disquette, soit depuis le disque dur.

**a) Utiliser une disquette d'amorçage**

Beaucoup de gens démarrent Linux à l'aide d'une disquette d'amorce. Cette disquette contient seulement un noyau Linux dans lequel a été écrit le nom de la partition du disque dur qui devra servir de système de fichiers principal. Le système pourra donc automatiquement retrouver ses différentes composantes sur le disque dur.

Une telle disquette est généralement créée par le programme d'installation de la distribution.

**b) Utiliser LILO**

LILO (Linux Loader) est un programme modifiant le secteur principal d'amorçage de votre disque dur pour vous donner le choix du système d'exploitation à lancer au moment du démarrage.

Après les messages classiques du BIOS, LILO signale sa présence avec un prompt ("LILO"). L'appui sur <tabulation> donne la liste des systèmes utilisables.
Il est aussi possible de configurer LILO pour lancer un des systèmes par défaut, si aucun nom n'est tapé au bout d'un certain temps.

**7 Arrêt du système**

Pour optimiser ses accès disques, le système UNIX n'écrit pas immédiatement sur le disque les modifications aux fichiers.
Il est donc très dangereux d'arrêter le système sans précaution ( en utilisant par exemple le bouton Marche/Arrêt ou le bouton reset): vous risquez de corrompre le système de fichier.

Pour arrêter correctement le système, il est nécessaire d'envoyer un signal aux différents processus en cours d'exécution (pour leur demander de terminer leurs opérations), d'attendre leur fin effective, puis d'intimer l'ordre au noyau de reporter sur disque toutes les modifications faites aux fichiers.

Tout cela est réalisé par la commande *shutdown*, dont la syntaxe est la suivante:

*shutdown <horaire> <message>*

ou *<horaire>* correspond à l'heure d'arrêt souhaitée( ou bien le mot *now* pour un arrêt immédiat) et *<message>* est un message envoyé aux utilisateurs.

# ANNEXE B

## Commandes de vi

### Touches de déplacement du curseur:

| Commande | Action |
|---|---|
| w | Déplace le curseur au début du mot suivant |
| b | Déplace le curseur au début du mot courant |
| e | Déplace le curseur à la fin du mot courant |

### Commandes pour ajouter du texte

| Raccourci clavier | Action |
|---|---|
| <Shift-a> | Ajoute du texte a la fin de la ligne courant |
| <Shift-i> | Insère du texte au début de la ligne courante |
| <o> | Ouvre une ligne en dessous de la ligne courante pour y ajouter du texte |
| <Shift-o> | Ouvre une ligne au-dessus de la ligne courante pour y ajouter du texte |

### Commandes de suppression de texte

| Raccourci clavier | Action |
|---|---|
| <x> | Efface les caractères se trouvant à la position du curseur |
| <d> <w> | Efface à partir de la position du curseur dans le mot courant jusqu'au début du prochain mot |
| <d> <$> | Efface à partir du curseur jusqu'à la fin de la ligne |
| <Shift-d> | Identique à <d> <$> |
| <d> <d> | Efface la ligne courante |

**Les commandes de modification et de remplacement**

| Raccourci clavier | Action |
|---|---|
| <r> | Remplace un seul caractère |
| <Shift-r> | Remplace une suite de caractères |
| <c> <w> | Modifie le mot courant à partir de la position du curseur jusqu'à la fin du mot |
| <c> <b> | Modifie le mot courant à partir du début du mot jusqu'au caractère précédent la position du curseur |
| <c> <$> | Modifie une ligne à partir de la position du curseur jusqu'à la fin de cette ligne |
| <C> <c> | Modifie la ligne entière |

**Les commandes de recherche**

| Commande | Action |
|---|---|
| /chaine | Effectue la recherche en avançant dans le tampon de la chîne |
| ?chaine | Effectue la recherche en reculant dans le tampon de la chaîne |
| <n> | Reprend la recherche dans la direction courante |
| <Shift-n> | Reprend la recherche dans la direction opposée |

## Commandes d'emacs

| Raccourci clavier | Description |
|---|---|
| **Enregistrer sur le disque** | |
| <ctrl-x> <ctrl-s> | Enregistrer le tampon courant sur le disque le disque |
| <ctrl-x> <ctrl-w> | Ecrit le tampon courant sur le disque et demande le nom demande le nom du nouveau fichier |
| <ctrl-x> <n> | Change le nom du fichier du tampon courant |
| <Esc> <z> | Transmet tous les tampons modifiés au disque et quitte emacs |
| **Lire à partir du disque** | |
| <ctrl-x> <ctrl-f> | Trouve le fichier, lit dans le nouveau tampon créé à partir du nom de fichier |
| <ctrl-x> <ctrl-r> | Lit le fichier dans le tampon courant en effaçant le contenu antérieur |
| <ctrl-x> <ctrl-i> | Insère le fichier dans le tampon courant |
| **Déplacer le curseur** | |
| <ctrl-f> | Avance d'un caractère |
| <ctrl-b> | Recule d'un caractère |
| <ctrl-a> | Se positionne au début de la ligne courante |
| <ctrl-e> | Se positionne à la fin de la ligne courante |
| <ctrl-n> | Avance d'une ligne |
| <ctrl-p> | Recule d'une ligne |
| <echap> <f> | Avance d'un mot |
| <echap> <b> | Recule d'un mot |
| <echap> <a> | Déplacement vers une ligne |
| <echap> <shift . > | Retour au début du tampon |
| <echap> <shift , > | Déplace à la fin du tampon |

| | |
|---|---|
| **Effacer et insérer** | |
| <ctrl-d> | Efface le caractère suivant |
| <ctrl-c> | Insère un espace |
| <echap> <d> | Efface le mot suivant |
| <ctrl-k> | Efface jusqu'à la fin de la ligne courante |
| <entée> | Insère une nouvelle ligne |
| <ctrl-o> | Ouvre une nouvelle ligne |
| <ctrl-w> | Efface la région délimitée par le marquage et le curseur |
| <echap> <w> | Copie la région dans le tampon de récupération |
| **Recherche et remplace** | |
| <ctrl-s> | Effectue une recherche en avance à partir du curseur |
| <ctrl-r> | Effectue une recherche en reculant à partir du curseur |
| <ctrl-x> <s> | |
| <ctrl-x> <r> | Répète la recherche en avançant |
| <ctrl-g> | Répète la recherche en reculant |
| <!> | Annule l'opération |
| <?> | Remplace le reste |
| <.> | Donne une liste des opérations |
| <y> | Remplace et quitte à l'endroit où la commande a été lancée |
| <n> | Remplace et continue l'opération de remplacement |
| | Ne remplace pas mais continue l'opération de remplacement |
| **Marquage du texte** | |
| <ctrl> <barre d'espace> | Place la marque à la position courante du curseur |
| <ctrl-x > <ctrl-x> | Echange la position du marquage avec celle du curseur |
| <ctrl-w> | Efface la région marquée |

| | |
|---|---|
| <echap- w> | Copie la région marquée dans le tampon de récupération |
| <ctrl-y> | Insère le tampon à la position courante du curseur |
| **Gestion du tampon** | |
| <ctrl-x> <b> | Bascule vers un autre tampon |
| <ctrl-x> <x> | Bascule sur le tampon suivant de la liste |
| <echap> <ctrl n> | Modifie le nom du tampon courant |
| <ctrl-x> <k> | Efface un tampon qui n'est |

# ANNEXE C

## Le matériel nécessaire

**Carte mère et microprocesseur :**

Linux fonctionne sur les machines équipées des microprocesseurs Intel 80386 ou 80486. Il fonctionne également sur le processeur Pentium d'Intel et sur les "clones" (tel que les processeur Cyrix et AMD)

Mémoire nécessaire :

Linux ne nécessite que très peu de mémoire pour fonctionner, en comparaison à d'autres systèmes modernes.

Linux fonctionne correctement avec seulement 4Mo de RAM même avec des programmes comme Xwindows, Emarcs ou autres. La zone de swap utilisée comme mémoire virtuelle permet au système d'exécuter des applications plus importante en stockant les portions inactives du code sur le disque dur.

**Espace disque :**

L'espace disque nécessaire pour installer Linux dépend des programmes à intaller. On peut utiliser le système minimal (sans les grosses applications telle que Xwindows) dans moins de 20 Mo; un système complet dans moins de 80Mo; et un très gros système avec de l'espace pour de nombreux utilisateurs et de futur extentions dans 100 à 150 Mo.

Ecran et carte vidéo :

Linux peut utiliser toutes les cartes vidéo standard rencontrées sur le PC : Hercules, CGA, EGA, VGA, SVGA.

Les environnement graphiques comme Xwindows ont par contre des besoins particuliers.

Les processus standard SVGA suivants sont supportés :

- Tseng ET3000, ET 4000AX, ET 4000/W32

- Western Digital/Paradise PVGA1
- Western Digital WD90C00, WD90C10, WD90C11, WD90C24, WD90C30, WD90C31, WD90C33
- Genoa GVGA
- Trident TVGA8800CS, TVGA8900B, TVGA8900C, TVGA8900LC, TVGA-9000, TVGA9000i, TVGA9100B, TVGA9200CX, TVGA9320, TVGA9400CX, TVGA9420
- ATI 18800, 18800-1, 28800-2, 28800-4, 28800-5 , 28800-6, 68800-3, 68800-6, 68800AX, 68800LX, 88800
- NCR 77C22, 77C22E, 77C22E+
- Cirrus Logic CLGD5420, CLGD5422, CLGD5424, CLGD5426, CLGD5428, CLGD5429, CLGD5430, CLGD5434, CLGD6205, CLGD6215, CLGD6225, CLGD6235, CLGD6220
- Compaq AVGA
- OAK OTI067, OTI077
- Advance Logic AL2101
- MX MX68000, MX680010
- Video7/ Headland Technologies HT216-32

Les cartes vidéo utilisant ces processeurs sont supportées sur tous types de bus.

**Cartes Ethernet**

Beaucoup de carte Ethernet parmi les plus polpulaires fonctionnent sous linux. Parmi elles :

- 3com 3c503, 3c503/16, 3c509
- Novell NE1000, NE2000
- Western Digital WD8003, WD8013
- Hewlett Packard HP27245, HP27247, HP272550

# ANNEXE D

## Exercices d'applications

### 1 Exercices sur l'administartion

**Exercice 1 :**

**1/** Créer un compte utilisateur qui appartient au groupe du super-utilisateur.

**2/** Si la commande utilisée ne fonctionne pas , quelle est la deuxième méthode possible ?

**Solution:**

1/ Pour créer un compte, on se connecte sous compte « root » et on utilise la commande ***adduser*** . Pour appartenir au groupe de root on doit avoir le même numeros de groupe GID (0) .

2/ Si la commande ***adduser*** ne fonctionne pas ( perte du fichier /sbin/adduser ) on édite directement la ligne correspondante à l'utilisateur sous la forme suivante :

nom-utilisateur : : UID :GID: nom-complet: répertoire recine: shell

Le deuxième champ est consacré au mot de passe crypté qui sera fixé par l'utilisateur à partir de son compte.

**Exercice 2 :**

**1/** Que doit faire un utilisateur pour accéder à son compte s'il oublie son mot de passe ?

**2/** Que doit faire le super-utilisateur pour se connecter sous root s' il oublie le mot de passe ?

**3/** Comment créer une disquette de maintenance ?

**Solution :**

1/ Si un utilisateur oublie son mot de passe ; le super-utilisateur peut éditer le fichier /etc/passwd et effacer le mot de passe crypté sur la ligne de l'utilisateur . Ce dernier pourra se connecter et fixer un nouveau mot de passe .

2/ Si le super-utilisateur oublie le mot de passe du compte root il doit suivre les étapes suivante:

- Utiliser la disquette de maintenance pour monter la position contenant habituellement le fichier /etc/passwd dans un répertoire (par exemple le répertoire /mnt) .
- éditer le fichier /mnt/etc/passwd
- effacer le mot de passe sur la ligne de root.

On obtiendra une ligne ressemblante à :

root: :0:0:root:/:/bin/sh

En redémarrant le système depuis le disque dur, le compte root peut être utiliser san mot de passe . une fois connecté la commande passwd permet de le fixé.

3/ La disquette de maintenance (boot/root) permet de lancer le système indépendamment de l'état du disque dur. Pour l'à créer il faut initialiser un système de fichiers sur une disquette pré formatée et y placer un noyau ainsi que tous les outils de commandes essentiels.

*Remarque:* Les deux disquettes boot et root de l'installation peuvent être utiliser comme disquette de maintenance

**Exercice 3 :**

**1/** Comment attribuer le droit de lecture , écriture et exécution d'un fichier à son proprietaire ;

**2/** Comment doit faire le super-utilisateur pour attribuer le droit d'utiliser l' imprimante a son groupe seulement ?

**3/** La commande ***at*** permet l'exécution d'une commande à un moment précis, comment peut faire le super-utilisateur pour interdire son utilisation à un utilisateur ?

**Solution :**

**1/** La commande est la suivante :

$chmod 700 nom-de-fichier

**2/** La commande précédante permet aussi d'attribuer le droit d'utiliser l'imprimante au groupe du super-utilisateur en remplacent 700 par 770 et la nom du fichier par / bin / lpr.

**3/** La procédure est la suivante :

L'utilisateur qui souhaitte employer la commande at ne doit pas être référencé dans le fichier / usr /lib /cron /at.deny.

Si le super-utilisateur édite dans ce fichier un nom d'utilisateur, ce dernier ne peut pas employer at .

**Exercice 4 :**

1/ De quelle manière peut on substituer la commande ls par dir ?

2/ Substituer le répertoire /usr par la lettre u en utilisant l'alias.

3/ Donner l'affichage de la commande dir u.

**Solution:**

1/ Première méthode: en utilisant un script

Editer le fichier suivant

```
! /bin/sh
# Ce script substitue la commande ls par dir
ls
```

Sauvegarder le fichier sous le nom 'dir'.

Utiliser la commande chmod pour pouvoir exécuter le fichier

```
$ chmod +x dir
```

La deuxième méthode: en utilisant 'alias'

```
$ alias dir=ls
```

2/ `$ alias u=/usr`

3/ `$ dir u`

`ls: b not found` # seul le premier mot est substitué

La substitution d'un alias n'a lieu que si le nom de l'alias, donc la commande est le premier mot. La seul exeption a la règle est le cas où le texte de l'alias se termine par un espace.

**Exemple:**

```
$ alias dir='ls '
$ alias u=/usr
$ dir u                 # correspond à ls /usr
```

**Exercice 5 :**

1/ Créer dans votre répertoire courant l'arborescence de l'illustration (figure-a- ) .

2/ Créer deux fichiers vides appelés "doc1" et "doc2" dans votre répertoire courant.

3/ Copier le fichier "doc1" dans le sous-répertoire A en lui donnant pour nom "doc3".

4/ Renommer le fichier "doc3" du répertoire A en "doc4".

5/ Créer une référence supplémentaire (link) sur le fichier doc4. Ce nouveau lien doit être placé dans le sous-répertoire C et avoir le nom doc5.

6/ Supprimer l'arborescence créée en demandant la confirmation. Réponder par "o" pour toutes les questions. Faite une exeption pour le fichier "doc5" qui ne doit pas être supprimé. Que remarquez vous?

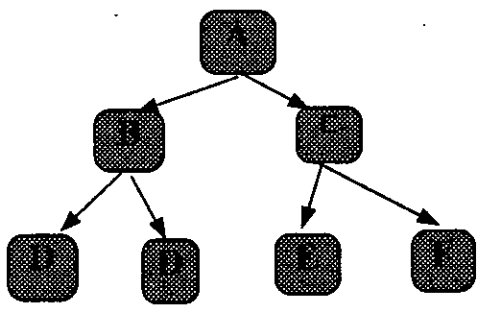


**Figure -a-**

**Solution :**

1/ on utilise la commande mkdir pour créer les répertoires et les sous répertoires .

$ mkdir A A/B A/C A/B/ D A/B /E A/C/F A/C/G.

2/ Créer les fichiers doc1 doc2 en utilisant la commande touch

$touch doc1 doc2

3/ Utiliser la commande cp pour copier ldoc1 dans le répertoire A

$cp doc1 A/doc3

4/ Renommer le fichier doc3 enutilisant la commande mv

mv doc3 doc4

5/ Pour chaque fichier dans un système UNIX , il existe un entete de fichier ( inod ) où sont eneregistrer des information de gestion relatives à ce fichier (taille, propriétaires , . . .). Par la

commande lu (luik) il est possible que plusieurs noms de fichiers fassent référance au même en-tête.

$ lu doc4 A/C / doc5

6/ Nous employons la suppression récurcive de la commande rm

$rm -ri A

Si vous reponder n à la demande de configuration de suppression du fichier doc5, la commande rm ne peut supprimer le répertoire C et par voie de conséquence pas non plus le répertoire A (car ils ne sont pas vides).

**Exercice 6:**

Un répertoire contient un grand nombre de fichiers, comment compter ces fichiers?

**Solution:**

La première possibilitée est d'utiliser la commande ls pour affichier et ensuite de les compter. Si les fichiers snt nombreux au point de ne pas tous tenir a l'écran , la solution consiste à ecrire tous les noms de fichiers dans un fichier et d'en faire compter le nombre de ligne par la commande wc(utilisation des possibilitées de redirection d'entrée / sorties).

le scr ipt suivant réalise l'opération demandée:

```
# !/bin/sh
ls > nom_de_fichier
cat nom_de_fichier
wc -l < nom_de_fichier
rm nom_de_fichier
```

**Exercice 7:**

Votre répertoire courant contient les fichiers suivants : Dix, dix, quatorze, Dixsept, Dishuit

1/ Transformer la sortie de la commaned ls a la commande grep

2/ Rechercher les lignes ayant les caracteristiques suiv antes .

a) le texte « ci » est placé a un endroit quelquenque de la ligne

b) Le texte « ze » est en fin de ligne

c) Le text dix ou Dix est seul dans la ligne

**Solution :**

a) $ ls | grep -n ei

Treize

b) $ ls | grep -n ze$

Treize quatorze

c) $ ls | grep -n ^[Dd]ix $

Dix dix dixsept dixhuit

**Exercice 8:**

Ecrire un script permettant de verifier si un utilisateur precis est connecté au système. Ce script se verra doté du nom de l'utilisateur. Il renverra par le canal de sortie standard un message indiquant si cet utilisateur travail ou non

**Solution:**

le script est le suivant

```
# !/bin/sh
  if [$#=0]
     then echo « erreur » & 2
      exit 1
  fi
  if who | grep « ^$1 »/dev/null 2>&1
  then
     echo « l'utilisateur \« $1\» est connecté
  else
     echo « L'utilisateur \« $1\» ne travail pas pour l'instant »
  fi
  exit 0
```

**NB** La commande « ...>fichier 2>&1 » redirige le canal de sortie satndard vers le fichier fichier et réunit en un même canal , le canal de sortie standard et le canal d'erreur standard .

**Exercice 9:**

1/ Donner la syntaxe de la commande find ?

2/ Quelle est la commande qui affiche le résultat d'une recherche page par page ?

**Solution:**

1/ Find parcourt les répertoires et leurs sous répertoires à la recherche de fichiers, ca syntaxe est la suivante:

find /repertoire(s) critère de sélection option

2/ Les noms des fichiers trouvés sont transmis par le canal de sorite standard . Dans beaucoup de cas on cherche à rediriger les messages d'erreurs vers le fichier de gestionnaire /dev/null (la crbeille).

Pour afficher le résultats de la recherche page par page , la commande ressemblera à :

find .....2>/dev/null | more

Pour transmettre également les messages d'erreurs a la commande more, il faudra rassembler le canal de sortie standard en un seul et même canal par la commande suivante:

find .....2>&1| more

**Exercice 10:**

Recherchez tous les fichiers vides à partir du répertoire actif. Faites en sorte que le système vous demande si ces fichiers doivent être supprimés.

**Solution:**

La supprission de fichiers peut être pilotée par la commande find et son optin -ok ou par l'optin -i de la commande rm

find . -isize o -exec rm -i {} \ ;

ou

find . -isize o -ok rm -i {} '; '

**NB** En dehord de l'option -print qui montre le chemain d'accés au fichier on peut egalement utiliser la commande -exec , la commande placée derirrière -exec doit être conclue par le paramètre ';' qui doit être caché par backslch ou des apostrophes.

**Exercice 11:**

Rechercher à prtir du répertoire courant , tous les fichiers dont les noms correspond aux critères suivants :

a) longeur de trois caractères

b) Commançant par une majuscule

c) Terminant par .c ou .f

d) Contenant un chiffre

**Solution:**

a) find .- name ''???'' -print

b) find .- name ''[A-Z]'' -print

c) find .- name ''*.[cf]'' -print

d) find .- name ''*[0-9]'' -print

**Exercice 12:**

comment monter un deuxième disque dur ?

**Solution:**

La procédure est la suivante :

- Configurer le disque dans le setup (Bios du système) .
- Démarrer une session LINUX normalement et se connecter en root .
- Configurer le avec fdisk en tapant la commande suivante :

fdisk /dev/hdb1 (le disque dur est du type IDE).

fdisk /dev/sda1 (le disque dur est du type SCSI).

(voir installation pour utilisation de fdisk sous LINUX )

- créer un répertoire (par exemple /dosd) ajouter la commande suivante dans le fichier /etc/fstab: mount type-système-fichier /dev/type-disque /dosd rw

redémarrer la machine .

**NB** Le premier disque utilisé et du type IDE.

**Exercices sur le développement**

**Exercice 13 :**

La commande startx permet de lancer le X Window . Le gestionnaire de fennêtre par défaut est le fvwm, comment configurer le système pour avoir le gestionnaire de fennêtre par défaut "*olwm*" .

**Solution :**

Pour changer le gestionnaire de fenêtres il faut éditer le fichier /usr/X11R6/lib/xinit/xinitrc et remplacer l'instruction ***exec fvwm*** par l'instruction ***exec olwm*** en se connectant sous root.

**NB** il existe plusieurs gestionnaires,on cite: olvwm , olwm , twm , mwm , tvwm et fvwm .

**Exercice 14:**

Généraliser le programme de pilotage de la carte pour n'importe quel port et n'importe quelle valeur du facteur d'amplificateur et de la voie choisie .

**Solution :**

Il suffit de replacer les valeurs 390 pour une variable globale du type entier et les valeurs 3 et 0 par des variables globales du type entier .

Les modifications seront :

int val - port, fact, ampli, voie choisie

in ( Val - port )

out ( Val - port , fact, ampli + voie choisie )

avec la fonction Set-tine on peut avoir une temporisation de 10 NS et de 10 NS en les exploit pour un échantillonnage convenable .

**Exercice 15 :**

Donner le programme Script qui visualise les données suivantes : ( en utilisent le GNUPLOT ) :

Sin (x), Cox (x) tg (cos (sinx))

arc sin(x), apccos (x), log (cosx)

**Solution :**

```
Set title «  La solution de l'exercice 15 »
Set no key
Set no border
Set noxtics
Set noytics
Plot [-50 :50] Sin(x), Cos(x), tg (Cos (sin(x))
# par défaut le premier graphe serait rouge (1) le second serait bleu (2) et le troisième sera vert (3).
Pause - 1 « taper , entrée pour continuer »
# pause - 1 « une pause avec écriture d'un message »
Clear
Set simples 50
Plot a sin(x), acos (x), log (cos (x))
Pause - 1  « taper , entrée pour retourner au menu principale »
```

**Exercice 16:**

Ecrire un programme qui permet de créer un processus en utilisant les appels systèmes Fork(), Wait(), Exec()

**Solution:**

Le programme père .c (l'executable est prog_pere )

```
# include <stdio.h>
  main()
```

```
{
 int P1,pid;
printf(‘’ début du processus père ‘’);
P1=fork()
    /*differenciation des procesus executable*/
     if(P1!=0) wait (1)
/* programme père attend ici*/
/* si non le programme fils sera executé*/
      else
       exec(‘’prog_fils’’,0);
Printf(‘’ le programme père continue mais il va s’arreter\n\n’’);
exit(0);
}
```

Le programme fils (l'éxecutable est prog_fils)

```
/* programme fils*/
main()
{
int pid;
/* identificateur du processus*/
printf(‘’ début d’éxecution du programme fils ’’);
pid = getpid();
/*recherche de l’identificateur*/
printf(‘’numero du processus  = %d\n’’,pid);
pritf(‘’ le processus filks se termine \n\n’’) ;
exit(0);
}
```

**Exercice 17**

Vous avez une nouvelle version du Xwindow , mais qui n'est pas une distribution slackware , Comment installer cette version ( la version de Linux que vous avez est la version slackware)

**Solution:**

La version slackware utilise les Makefiles pour l'installation se que n'est pas le cas pour notre version , la plus simples des méthodes est la suivante :

* copier tous les fichier des répertoires même pour les sous répertoires dans seul et unique répertoire .

* lancer la commande pkgtool

* un menu apparu il faut donc pricéser que le pakage à installer est le package x.

**Exercice18:**

Créer un processus en arrière plan en utilisant les différantes possibilités ?

**Solution:**

Supposant qu'on a un programme éxecutable alpha . Ce dernier va être lancer en arrière plan, pour cela il exit différante methodes on cite les plus utilisers

1- Tapper a linvite :alpha&

2- Tapper a linvite :alpha et puis changer de compte en utilisant <Ctrl><Alt><F1.....F6> et ce connecter sur le même compte

3- en utilisant at [heur][date][-argment] alpha.

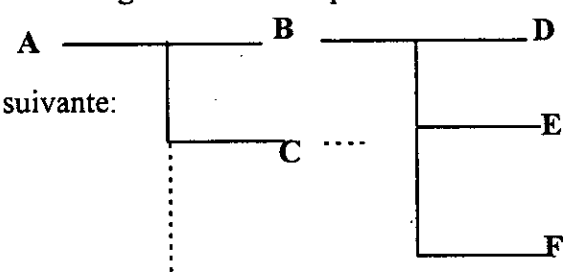
**NB** voire aussi les instructions de manipulation des processus .

**Exercice 19:**

Il est possible d'afficher tous les rép er toires se trouvant sous un répertoire donné , grâce a ls -R , maleresement cela ne donne pas une trés bonne idée de la stricture de l'arboréssance .

Ecrire un programme qui permet de mieu visualiser l'organisation d'un petit nombre de sous répertoire .

le mieu sera de formater la sortie de la manière suivante:



**Solution:**

Le programme qui réalise cette opération est le suivant :

```
#! /bin/sh
reclls()
# récursive  ls
{
unetab=' '\t ''
for lefichier in '' $@' '; do
  echo $ lefichier
      if [-d '' $lefichier '];then
      cefichier = $lefichier
      recdir $( command ls $lefichier)
 fi
done
inset dir unetab tab
}
# fonction récurcive dir
 recdir()
{ tab=$tab $unetab
   for fichier  in''$@''; do
 echo -e $tab $fichier
cefichier = $ cefichier /$fichier
if [-d ''$cefichier]; then
recdir$(command ls $cefichier)
fi
cefichier = ${cefichier %/*}
done
tab =${tab%/t}
 }
```

**Exercice 20**

Ecrire un script de configuration du XWindows qui lance une fenêtre xterm de taille (80 x 24) et de position (5,50).
Remplacer la valeur du fond de l'écran par un 'vert'.

**Solution** : On utilise un éditeur de texte pour saisir le programme de configuration suivant :

```
# ! / bin / sh
# Lancer xterm
xterm - geométry + 5 + 50 - fg black - bg inhibole.
# la taille (8 x 24) est la taille de xterm.
# par défaut, elle n'est pas mentionné xsetroot - solid green &.
# le gestionnaire de fenêtre exec fvwm.
```

Pour exécuter ce fichier, on utilise d'abord la commande chmod :

$ chmod +x nom du fichier.

L'exécution du fichier se fait à partir du XWindow.