Chapitre 1

Introduction

Ceci est un polycopié de cours de licence informatique sur les systèmes d'exploitations en général et plus spécialement sur la famille Unix.
Ce poly est le support utilisé pour les licences et pour les Apprentis ingénieurs de la filière Informatique et Réseau.

1.1 Unix

1.1.1 Pourquoi unix ?

Pourquoi ce choix d'unix comme sujet d'étude pour le cours ?

- LE PRIX
- la disponibilité des sources
- L'intélligence des solutions mise en oeuvre
- de grande ressource bibliographique
- il faut mieux apprendre les conceptes fondamentaux dans un système simple et ouvert puis passer a des systèmes propriétaires et fermés que l'inverse.
- parceque je ne vais pas changer mon cours tout de suite

1.1.2 le succès d'Unix et de linux

Le succès d'UNIX sans doute parce que :
- Ecrit dans un langage de haut niveau : C (C++, Objective C) ;
- une interface simple et puissante : les shells, qui fournissent des services de haut niveau ;
- Des primitives puissantes qui permettent de simplifier l'écriture des programmes ;
- Un système de fichier hiérarchique qui permet une maintenance simple et une implémentation efficace ;
- Un format générique pour les fichiers, le flot d'octets qui simplifie l'écriture des programmes ;
- Il fournit une interface simple aux périphériques ;
- Il est multi-utilisateurs et multi-tâches ;
- Il cache complètement l'architecture de la machine à l'utilisateur.

1.1.3 Des points forts

- Système né dans le monde de la recherche
  intégration de concepts avancés
- Diffusion ouverte
  accès aux sources

- Langage (de haut niveau)
  compilation séparée, conditionnelle, paramétrage, précompilation
- Enrichissement constant
- Ouverture (paramétrabilité du poste de travail)
- Souplesse des entrées/sorties
  uniformité
- Facilités de communication inter-systèmes
- Communautés d'utilisateurs (/etc/groups)
- Langages de commandes (flexibles et puissants)
- Aspect multi-utilisateurs
  connections de tout type de terminal, bibliothèques, etc
- Parallélisme
  multi-tâches : "scheduling" par tâche
  communication entre tâches
  multiprocesseurs
- Interface système/applications
  appels système, bibliothèque
- le système de gestion de fichiers
  hiérarchie
- Interfaces graphiques normées : X11.
- Profusion d'interfaces graphiques sous linux Gnome et KDE en particulier

1.1.4 Des points faibles

- Fragilité du S.G.F.
  pertes de fichiers possible en cas de crash
  réglé avec les SGF journalisés
- Gestion et rattrapage des interruptions inadapté au temps réel
  Des évolutions avec RLlinux et OS9.
- Mécanisme de création de processus lourd
  de nombreuses améliorations en particulier les threads.
- Une édition de liens statique
  Amélioration avec les librairies partagées.
  des Modules noyau chargeables/déchargeables dynamiquement
- Rattrapage d'erreur du compilateur C standard peu aisé !
  Ces bugs sont corrigées
- Coût en ressources
- reste globalement efficasse
- Gestion

1.2 Structure générale des systèmes d'exploitation

Un système d'exploitation est un programme qui sert d'interface entre un utilisateur et un ordinateur.

Un système d'exploitation est un ensemble de procédures manuelles et automatiques qui permet

à un groupe d'utilisateurs de partager efficacement un ordinateur. *Brinch Hansen.*

Il est plus facile de définir un système d'exploitation par ce qu'il fait que par ce qu'il est. *J.L. Peterson.*

Un système d'exploitation est un ensemble de procédures cohérentes qui a pour but de gérer la pénurie de ressources. *J-l. Stehlé P. Hochard.*

Quelques systèmes :

le batch Le traitement par lot (disparus).

interactifs Pour les utilisateurs (ce cher UNIX).

temps réels Pour manipuler des situations physiques par des périphériques (OS9 un petit frère futé d'UNIX). L'idée est de gérer le temps vrai.En particulier de gérer des évènement aléatoires qui neccessite l'exécution d'une action en proirité absolue.

distribués UNIX ?, les micros noyaux ? l'avenir ?

moniteurs transactionnels Ce sont des applications qui manipulent des objets à tâches multiples comme les comptes dans une banque, des réservations, etc. L'idée est de décomposer l'activité en actions, chacune indépendantes des autres,pour ce faire elle sont écrites pour avoir un comportement dit "atomique" ainsi il n'y a pas de programme mais des évènement et des actions associées. Il n'y pas dans de changement de context pour traiter une action, c'est le système adapté pour traité de gros volumes de petites opérations.

SE orientés objets Micro Noyaux.

1.2.1 Les couches fonctionnelles

Couches fonctionnelles :
- Programmes utilisateurs
- Programmes d'application éditeurs/tableurs/BD/CAO
- Programmes système assembleurs/compilateurs/éditeurs de liens/chargeurs
- système d'exploitation
- langage machine
- microprogramme
- machines physiques

1.2.2 L'architecture du système

L'architecture globale d'UNIX est une architecture par couches (coquilles) successsives comme le montre la figure 1.2. Les utilisateurs ordinaire communiquent avec la couche la plus évoluée celle des applications (en générale aujourd'hui associé avec une interface graphique). Le programmeur lui va en fonction de ses besoins utiliser des couches de plus en plus profondes, plus précises mais plus difficiles a utiliser.
Chaque couche est construite pour pouvoir être utilisée sans connaitre les couches inférieures (ni leur fonctionnement, ni leur interface).
Cette hiérarchie d'encapsulation permet d'écrire des applications plus portables. En effet si elles sont écrites dans les couches hautes, le travaille de portage est fait par le portage des couches inférieures. Pour des applications où le temps de calcul prime devant la portabilité, les couches basses seront utilisées.

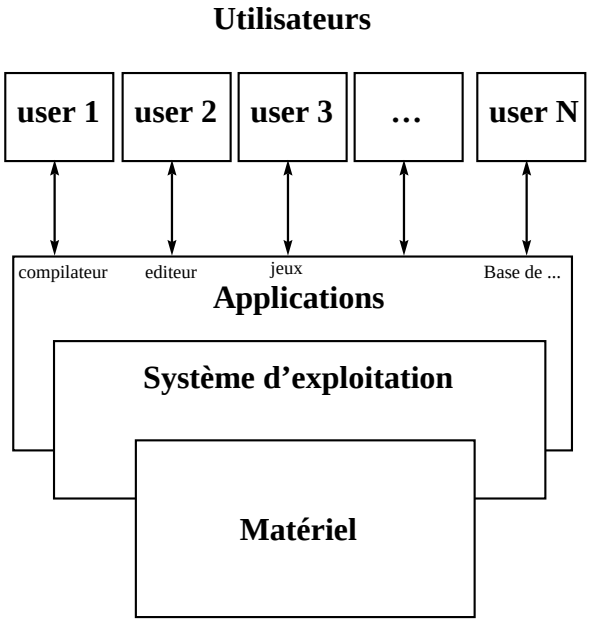

FIG. 1.1 – Vue générale du système

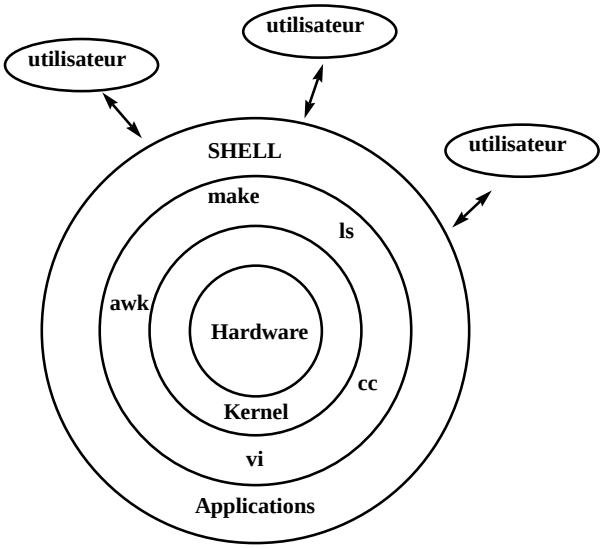

FIG. 1.2 – Point de vue utilisateur

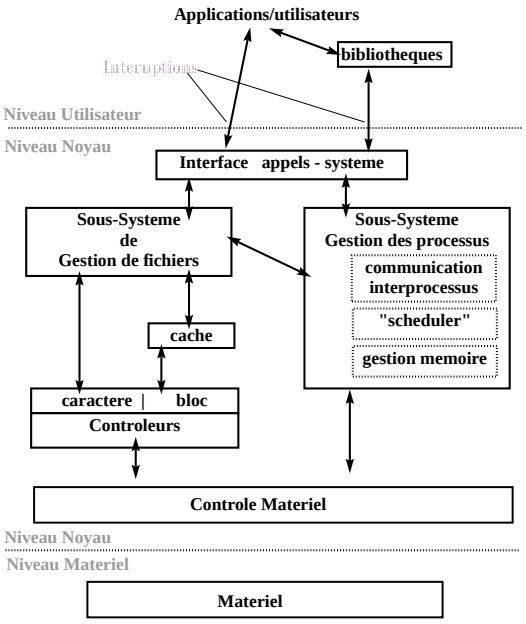

FIG. 1.3 – Architecture du noyau

1.2.3 L'architecture du noyau

L'autre approche architecturale est l'architecture interne du Noyau (kernel). C'est une architecture logicielle elle permet aux développeur de structurer le travail de développement. Le but ici est de simplifier la compréhension et la fabrication du système. Nous cherchons donc ici à décomposer le noyau en parties disjointes (qui sont concevables et programmables de façons disjointes). La Figure 1.3 donne une idée de ce que peut être l'architecture interne d'un noyau UNIX. Noter bien la position extérieure des bibliothèques .

1.3 historique

Il existe un site très agréable sur l'histoire des ordinateurs :

`http://www.computerhistory.org/`

.