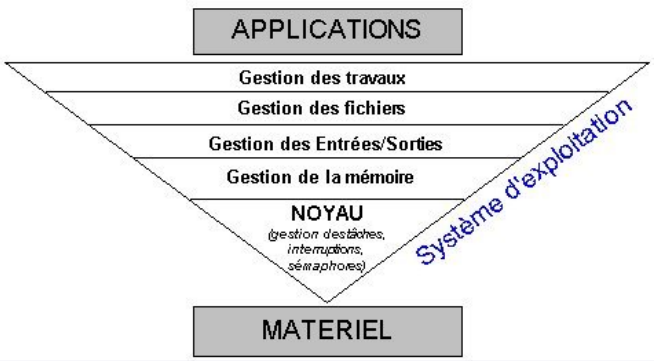
IC1: Système d'exploitation et Programmation Système

IUT Béthune
DUT2 Réseaux & Télécommunications

Daniel Porumbel

Langage Humain → Langage du Matériel

Le système d'exploitation doit gérer l'exécution de la chaîne :
action utilisateur→programmes→action physique



Composition

gestion des processus (fils d'exécutions), communication inter-processus, la gestion de la mémoire, sécurité des données et communications réseau, systèmes de fichiers, pilotes, utilisation processeur, interface utilisateur, programmes utilitaires, etc.

Taches Importantes Gérés par le SE

- la communication avec le clavier, l'écran, le disque dur, les périphériques
  - Linux : tout périphérique est vu comme un fichier par les programmes
- le déroulement des programmes et l'accès en parallèle à la mémoire
- la sécurisation des données stockées
  - comment vérifier le mot de passe de l'utilisateur `root` sans le stocker en clair ?

Communication avec le clavier et l'écran

Principes LINUX

- le clavier = le fichier `/dev/tty`
- le terminal = le fichier indiqué par la commande `tty`
- l'écran est géré par un serveur *X*.
  - Les programmes sont des clients qui envoient des requêtes au serveur *X*
  - La commande `ssh -X IPDIST` permet de lancer sur l'écran local des programmes qui tournent sur `IPDIST`

Processus et fils d'exécutions

Les processus parallèles ne sont toujours exécuté sur plusieurs processeurs.

- ordonnanceur permet d'affecter à chaque processus un temps de processeur
  - contrôlable avec cpulimit
- Il est difficile de gérer l'accès aux ressources
  - Si deux processus veulent écrire en même temps le même fichier ?

Commandes utiles : top,ps -x

Les détails d'un serveur

- La communication `serveur web`⟷`navigateur web` est basée sur des messages `http`
- Un important mot clé `http` : `GET`
  - Exemple : `GET / HTTP/1.0`
  - La page principale (/) est demandée via le protocole `http`, version 1.0
- Pour communiquer en `http`, il faut établir une connexion `tcp` : `socket Serveur`⟷`socket Client`
- Le port serveur est très souvent 80, ou 443 ou 8080
- Un message `http` est coupé en plusieurs segments `tcp`
- Chaque appel Ruby `clientTcp.puts(...)` représente un segment `tcp`

Exemple communication `http` via `tcp`

1. Requête GET envoyée
   - Ouverture sockets TCP et transfert GET
   - La requête GET est reçue
2. Le TCP transfère la réponse HTTP à la socket du client
3. La réponse HTTP est affichée
4. Fermeture des sockets

[image @ Pascal Nicolas]