Institut
Mines-Télécom

Langage C et Systèmes d'exploitation

Introduction

Responsable : Etienne Borde (C218-3)

Auteur : Thomas Robert

Ce que nous avons vu jusque là

- **Le langage C**
  - Comment on l'utilise (types, variables, fonctions…)
  - Comment son exécution fonctionne (utilisation de la mémoire, …)
  - Comment on le compile (principes et réalisation avec gcc, make…)

- **Ce que nous allons voir à partir de maintenant**
  - Les principes de fonctionnement d'un système d'exploitation UNIX
  - Les services offerts par un système d'exploitation UNIX
    - Gestion des ressources de calcul
    - Gestion des fichiers
    - Gestion des entrées/sorties
    - Gestion de la mémoire

- **Le langage C va nous servir à illustrer cela, avec un peu de pratique…**

Support d'exécution une introduction

Fonctionnalités et exemples

- **Fonctions :**
  - Fournir les moyens de stocker de l'information
  - Fournir les moyens d'exécuter le traitement correspondant à un programme
  - Fournir des moyens d'interagir avec d'autres supports d'exécution ou l'environnement physique.

- **Exemples de plateformes différentes :**

Portables

Ordinateur de bord

Poste de travail

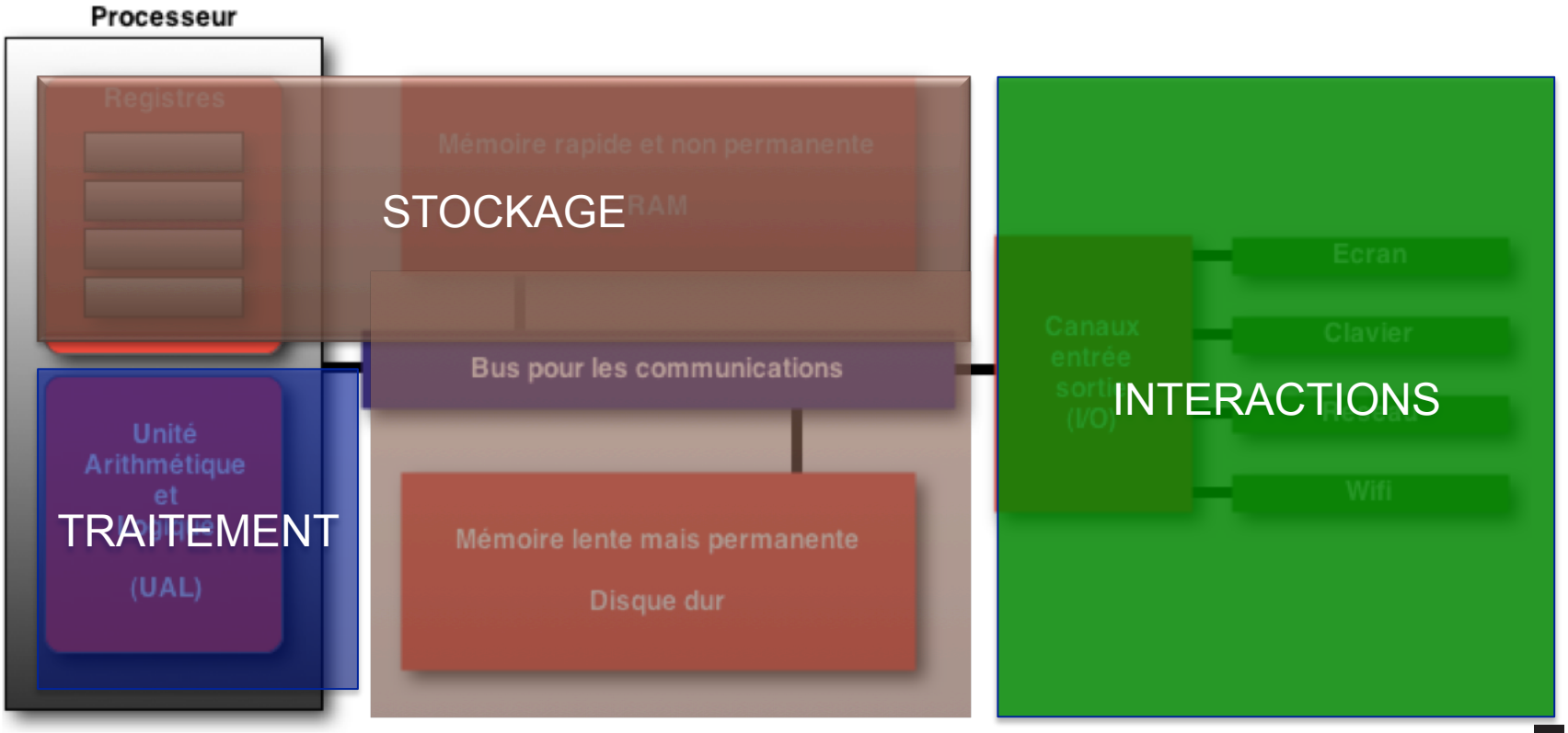
Un support d'exécution : les éléments clés

- **Un calculateur = regroupement de composants électroniques fournissant des capacités/ressources**
  - de stockage des données
  - de capture de stimuli extérieurs (physiques)
  - de communication de données avec d'autres calculateurs
  - de calcul et manipulation de données, contrôle d'exécution

- **Un système d'exploitation = logiciel pour organiser, simplifier et optimiser l'accès aux ressources**
  - Mise en place du partage et protection des ressources *(traitement / stockage)*
  - Simplification de l'usage et de l'accès aux ressources

Vision architecturale du calculateur

- **Intérêt : présenter une cartographie des ressources + de leurs interactions**



Organisation du stockage

- **Organisation des bits en octet = 8 bits (granularité usuelle), plus pratique à écrire en base 16 (symboles: 0123456789ABCDEF)**

- **Problème du volume :**
  - Mozilla : 178 Mo de données utilisées dans certains cas
  - Base de données client : ~quelques Téra octets

Comment retrouver les données intéressantes dans un tel volume ?

- **Index et référence :**
  - Index : une fonction associant un identifiant à une donnée (mieux quand il est unique)
  - Référence : une valeur d'identifiant pouvant être utilisé pour récupérer la donnée qui lui est associée via l'index.
  - résoudre la référence = Index(Référence) →obtenir la valeur

En pratique (1) mémoire

- **Mémoire (RAM) : composant permettant de stocker un ensemble d'octets de grande taille multiple d'une puissance de 2^K**

- **Caractéristiques :**
  - Index **:** 1 relation associant 1 numéro à chaque emplacement pouvant mémoriser 1 octet
    (le numéro est appelé **adresse**)
  - Granularité d'accès : groupe de 1 à 8 octets d'index consécutifs
    (ce groupe est appelé **mot mémoire)**
  - Référence à **un octet : une adresse**
  - Référence à **plusieurs octets contigus :**
    1 adresse @ + nombre d'octets à lire à partir de @

En pratique (2) Les autres stockages

- **Justification technologique de leur intérêt (et/ou):**

  Plus rapide, moins couteux utilisable directement par le processeur …
  beaucoup de raisons très variées

- **Registres : stockage mémorisant quelques octets**
  - taille = mot mémoire (8 octets sur une machine 64 bits)
  - Directement nommés => pas d'index à base de numéros
  - là où le processeur réalise ses calculs
- **Stockage de masse (disque dur) :**
  - Accès = bloc pour transferts vers ou depuis la mémoire (ensemble d'octet ~ 1000 octets)
  - Index structuré : (cas disque dur non SSD)
    - Plateau, cylindre … ~ adresse postale: rue, ville, pays

Capacités de traitement (1)

- **Composant clé : le processeur**
- **Traitements = instructions (opérations *élémentaires*)**
- **Jeu d'instructions = {des opérations élémentaires supportées par le processeur}**
  - Calculs numériques
  - Transferts de données entre supports de stockage
  - Evaluations de conditions booléennes

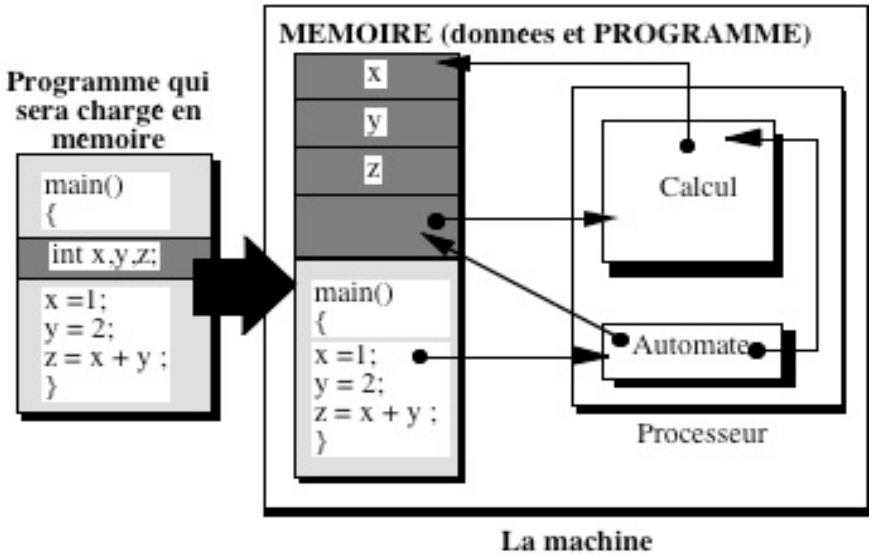
    ….

- **Problème: Comment indiquer au processeur les instructions à exécuter ?**

Capacités de traitement (2) : Un peu d'histoire… ça commence en 1801

- **Ancêtre de l'ordinateur : Métier à tisser « Jacquart »**
- **Objectif : tisser un motif à partir de fils de couleurs différentes**
- **Principe : automatisation des mouvements de crochets/navette**
- **Mise en œuvre :**
  - Carton perforé = représentation de la description des mouvements
  - Répétition du cycle :
    - Décoder le mouvement suivant (navette et crochets) 1 pour chaque
    - Opérer les mouvements décodés

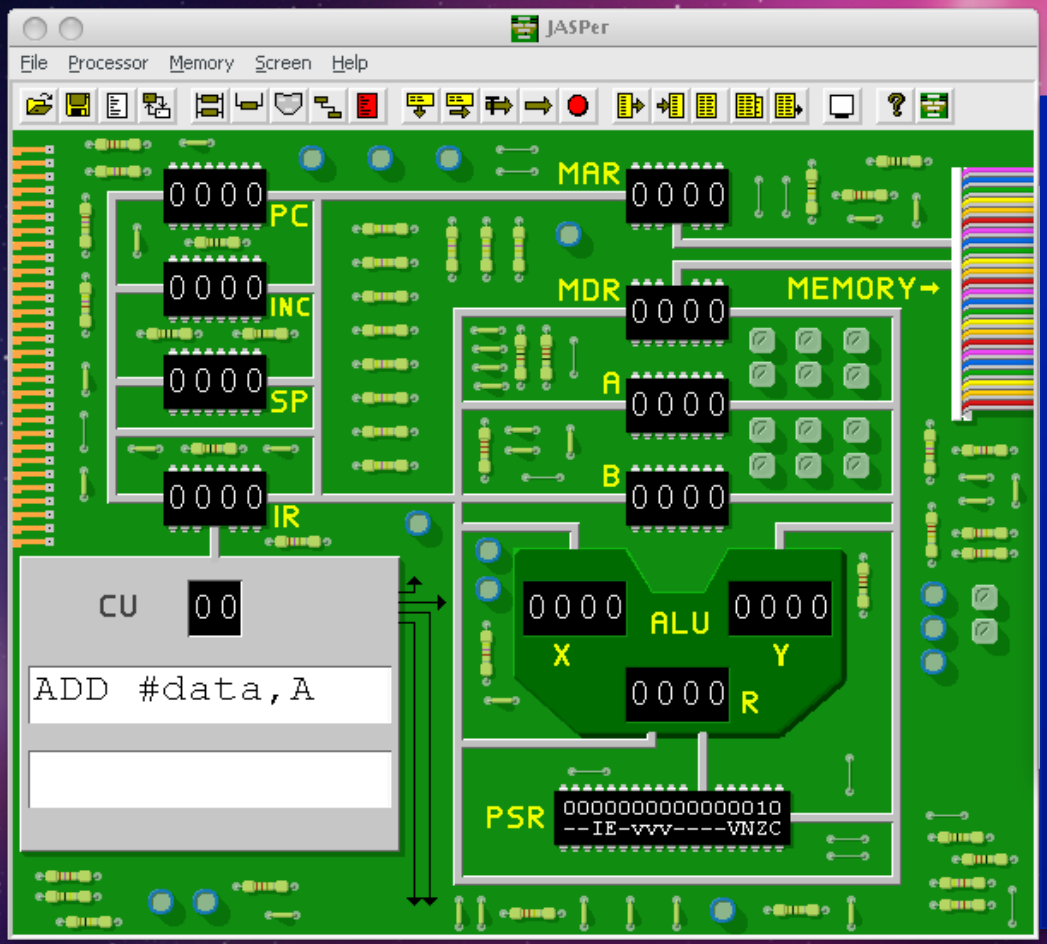
Capacités de traitement (3)
Rappel Architecture Von Neumann



- **Caractéristiques principales**
  - Représentation binaire des instructions en mémoire
  - Stockage sur les mêmes supports des données ET de la **représentation** des instructions

- **Automatisation de l'exécution**
  - Compteur ordinal (CO): registre contenant l'adresse de la prochaine instruction à exécuter
  - Répétition systématique de la logique d'exécution
    - Récupérer l'instruction associée à la référence contenue dans CO
    - Déterminer la nature de l'opération décrite par l'instruction
    - Récupérer les données d'entrée de l'opération

Représentation visuelle du fonctionnement d'un calculateur (processeur + mémoire) [1]



A trier

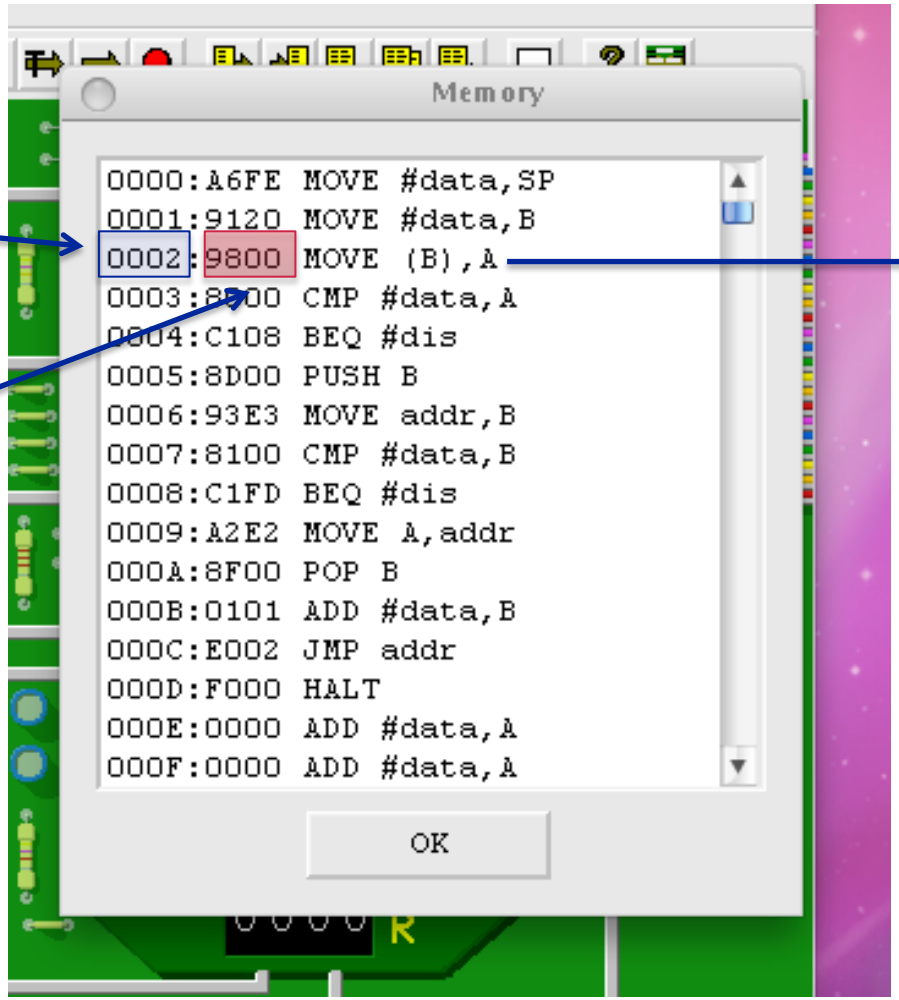
.DS_Store

Structure du processeur

- **Registres : PC,SP,IR,A,B**
- **Unité de contrôle (CU)**
  - Interprète IR
  - Orchestre les transferts inter registres
- **Mémoire**
  - Mots de 2 octets
  - Référence dans MAR
  - Valeur dans MDR
- **Codage instructions**:
  IR = 00 00 => A reçoit A +0
  - 2 premiers = opération
  - 2 derniers = données

Représentation instructions en mémoire

Référence = adresse



Interprétation

Valeur = Code instruction

ATTENTION
notation base 16
0123456789ABDEF

Instruction et comportements modifiant CO

- **Instruction de contrôle : instruction ayant un impact sur la valeur de CO**
  - Branchement de déroutement et restauration (paire d'instructions call-ret, int-iret…)
  - Branchement conditionnel (selon une condition)
  - Branchement inconditionnel (systématique)

- **Observation : possibilité de concevoir une séquence d'instructions**
  - Transférant des données entre registre et mémoire
  - Réalisant des calculs / comparaison
  - Déterminant l'adresse de la prochaine instruction à exécuter

Le cas particulier du Déroutement avec restauration

On veut calculer

$$\sum_{j=121}^{242} j$$

Rappel Somme 1…N = N*(N-1)/2
=> calcul possible en O(1)

idée N dans Ra et résultat dans Rb
Rb doit contenir ((Ra-1) * Ra)/2

Calcul de N*(N-1)/2 :

@	Instruction
911	Rb ← Ra-1
912	Rb ← Ra*Rb
913	Rb ← Rb/2
914	….
915	….

Le cas particulier du Déroutement avec restauration

On veut calculer

$$\sum_{j=121}^{242} j$$

Rappel Somme 1…N = N*(N-1)/2
=> calcul possible en O(1)

idée N dans Ra et résultat dans Rb
Rb doit contenir ((Ra-1) * Ra)/2

Calcul de N*(N-1)/2 :

@	Instruction
1	Ra ← 120
2	Call 911
3	Rc ← Rb
4	Ra← 242
5	Call 911
6	Rb ← Rb -Rc

@	Instruction
911	Rb ← Ra-1
912	Rb ← Ra*Rb
913	Rb ← Rb/2
914	Ret
915	….

Bilan sur les capacités d'exécution

- **Représentation en mémoire de séquence d'opérations dont l'exécution est automatisée**
- **Capacité à définir la prochaine instruction à exécuter**
- **Capacité à réutiliser une même séquence d'instructions dans différents « contextes »**
- **Capacité à pouvoir arrêter une séquence d'exécution pour faire autre chose (expliqué en cours de système):**
  - Soit parce que le processeur ne sait pas continuer
  - Soit parce qu'il faut exécuter un autre traitement (qui peut n'avoir aucun rapport avec celui en cours).

Rappel :
Un support d'exécution : les éléments clés

- **Un calculateur = regroupement de composants électroniques fournissant des capacités/ressources**
  - de stockage des données
  - de capture de stimuli extérieurs (physiques)
  - de communication de données avec d'autres calculateurs
  - de calcul et manipulation de données, contrôle d'exécution

Problèmes : partager efficacement ces ressources entre plusieurs applications, utilisateurs, machines, etc.

Chargement / exécution d'un programme: vue simplifiée

- **Charger chaque « section en mémoire »**
- **Définir les registres d'adresses de début de section**
- **Calculer l'adresse du point d'entrée**
- **Placer les valeurs des paramètres à leur place attendue en mémoire**
- **Modifier CO pour exécuter le point d'entrée**

…. Exécution du programme ….

Exécution de plusieurs applications

- **Pendant l'exécution d'un programme, celui-ci utilise le registre CO pour stocker l'adresse de la prochaine instruction à exécuter… Et met à jour le CO en fonction du code du programme (i.e. branchement, boucle, etc.)**

- L'exécution d'un programme met à jour le CO pour exécuter l'ensemble des instruction du programme jusqu'à sa terminaison (donc potentiellement jusqu'à l'infinie)

- Comment donc exécuter un autre programme « en même temps »?
  - Interrompre l'exécution du programme « en cours »
  - Et puis… ?

La suite de ce cours

- **Partage de ressources de calcul**
  - Processus
  - Ordonnancement
  - Synchronisation
- **Gestion des entrées/sorties**
  - Accès au contenu d'un fichier en lecture et/ou écriture
- **Gestion du système de fichiers**
- **Partage de la mémoire**
  - Partitionnement
  - Pagination

- **Tout cela aide fortement à comprendre comment fonctionne n'importe quel ordinateur de « bureautique ».**

Quelques généralités sur les systèmes d'exploitation

Les services d'un système d'exploitation

- **optimiser l'utilisation des ressources (en fonction des critères : équité, satisfaire les tâches les plus prioritaires,…)**
  - Partage de ressources de calcul
    - Processus
    - Ordonnancement
    - Synchronisation
  - Gestion des entrées/sorties
    - Accès au contenu d'un fichier en lecture et/ou écriture
  - Gestion du système de fichiers
  - Partage de la mémoire
    - Partitionnement
    - Pagination

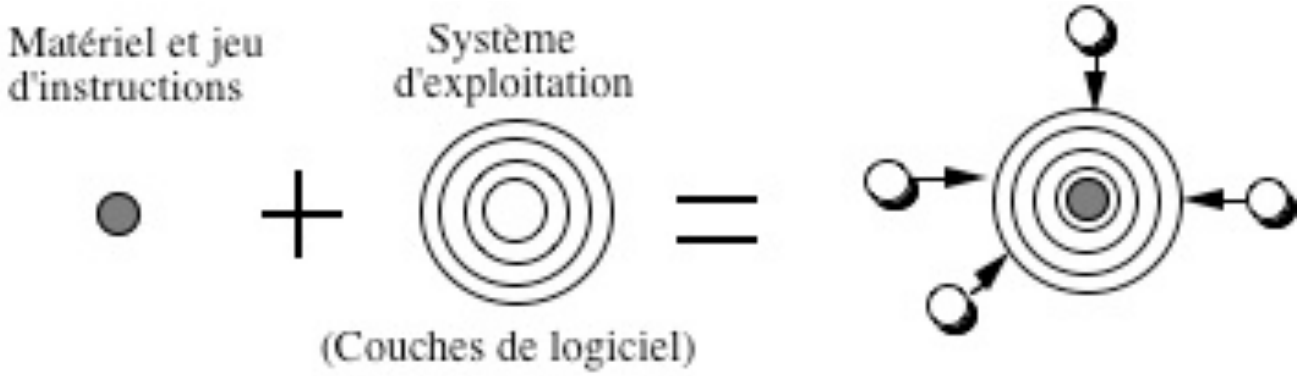
- **garantir l'indépendance de l'exécution d'un programme vis-à-vis du matériel,**

- **proposer une interface utilisateur souple et puissante**

- **Dans la suite de SELC, nous allons illustrer cela sur des systèmes d'exploitation de la famille UNIX**

Principes généraux

- Organisation en couches:
  - Objectif = assurer l'indépendance vis-à-vis des aspects matériels en définissant des services de hauts niveaux qui s'appuient sur des services de plus bas niveau (les aspects bas niveaux étant dépendant du type de support d'exécution).
  - Transparent pour l'utilisateur qui ne voit que des services de haut niveau



A propos d'UNIX

- **Développé par *Thompson* et *Ritchie* en 1969-70 aux laboratoires BELL (ATT), Dérivé de *CTSS* et de *MULTICS* (MIT).**

- **Principales caractéristiques :**
  - deux couches : kernel et user,
  - Très grande portabilité : noyau écrit en C à 90 %,
  - Nombreux outils disponibles,
  - toutes les ressources sont vues comme des fichiers (banalisation des entrées/sorties)
  - langage et interpréteur de commande *(shell)* puissant qui permet d'enchaîner facilement des commandes *(pipe)*.

Liens entre resources et systèmes d'exploitatation

- **Le système d'exploitation gère et utilise les ressources de la machine,**
- **Ressource = matériel fondamental :**
  - la mémoire,
  - le ou les processeur(s),
  - les périphériques,
  - les flux d'informations (les échanges entre la mémoire, les processeurs et les unités d'échanges),
  - la date et l'heure (estampillage),
  - les fichiers.
- **Le code source du système d'exploitation**
  - maintient une représentation de l'utilisation de ces ressources
    - Certaines des structures de données associées sont accessibles aux développeurs
  - Implémente des fonctions (majoritairement en C)
    - Certaines de ces fonctions sont accessibles aux utilisateurs