Utilisation du Système Unix

Olivier Dalle

Programme de la Formation

1. Introduction à Unix
2. Initiation à l'utilisation d'Unix
3. Xwindow : l'environnement graphique
4. Zsh : utilisation avancée en mode texte
5. Filtres et expressions régulières
6. Elements d'administration

Introduction à Unix

Historique

- 69 : Naissance aux Bell Labs (Thompson & Ritchie)
  - Successeur (modeste) de Multics
- 72: Réécriture en C (Ritchie)
  - Langage C créé pour l'occasion
  - 90% du code noyau en C
- 75 : diffusion des sources de la V6 aux universités
  - Large diffusion
  - Naissance de la branche Berkeley BSD
    - Branche ATT, conduit à Unix System V R4 (1990)
    - Branche BSD, conduit à BSD 4.4 (1995)
- ~92 : noyau Linux 0.11
- Nombreuses standardisations successives
  - POSIX (1003.1 et 1003.2), XPG, Unix95, Unix 98

Les Raisons du Succès

- Système écrit à l'aide d'un langage de haut niveau
  - Un peu plus lent, mais portage facile
- Interface utilisateur simple
- Appel systèmes réutilisables pour l'écriture de commandes
  - Possibilité d'ajouter ou d'améliorer des commandes
  - Ex: Projet GNU
- Le système de gestion de fichiers (SGF) est hiérarchique
- Système multi-utilisateurs, multi-tâches, multi-...
- Entrées/sorties généralisées
  - Contrôle des périphériques par de "simples" fichiers
- Disponibilité de versions gratuites
  - Linux, variantes BSD, ...

Système d'Exploitation = Ensemble de Programmes

- Le noyau
  - Chargement lors du démarrage (*boot*)
  - Gros programme (Linux = 4 millions de lignes de code)
    - Gestion des évènements Matériels
      - disques, souris, clavier, ...
    - Gestion des ressources
      - Ex: Partage du CPU (time sharing)
    - Gestion Structures de Données
      - Systèmes de Fichiers, Identité des utilisateurs, ...
    - Fournit des « services » élémentaires : Appels Systèmes
      - «Ecrire fichier», «Obtenir la date», «Envoyer message», ...
- Programmes de services supplémentaires
  - Bannière de login, Partage de fichiers distants, ...
- Utilitaires de base ...

Propriétés Remarquables des Systèmes d'Exploitation

- Mono- ou Multi-tâches
  - Mono = exécution de tâches multiples en séquence
    - *Batch* = traitement par lots
  - Multi = exécution simultanée de plusieurs programmes
    - *Time Sharing* : partage du temps CPU entre les programmes
- Mono- ou Multi-utilisateurs
  - Mono = 1 seul utilisateur
  - Multi- = plusieurs utilisateurs en même temps
    - Chaque nouvel utilisateur lance une session indépendante
    - Interaction au travers d'un terminal (ou console)
      - Physiquement connecté : machine dite « multi-poste »
      - Connexion distante : par le réseau
      - Consoles virtuelles (linux) : plusieurs sessions sur un même terminal

Exemples de Systèmes

- MSDOS : mono-tâche, mono-utilisateur
- Windows95/98 : multi-tâches, mono-utilisateur
  - multi-tâches « coopératif » = peu interactif
- Windows NT/XP : multi-tâches, multi-utilisateurs
  - Une seule session à la fois dans la version de base
  - Sessions multiples possibles sur versions serveur
    - Depuis un autre SE Windows « client »
    - Grâce à certains prog spécialisés (Citrix, Rdesktop, ...)
- Unix (linux) : multi-tâches, multi-utilisateurs, multi-sessions, multi-postes ...

Architecture Fonctionnelle : Principe Directeur

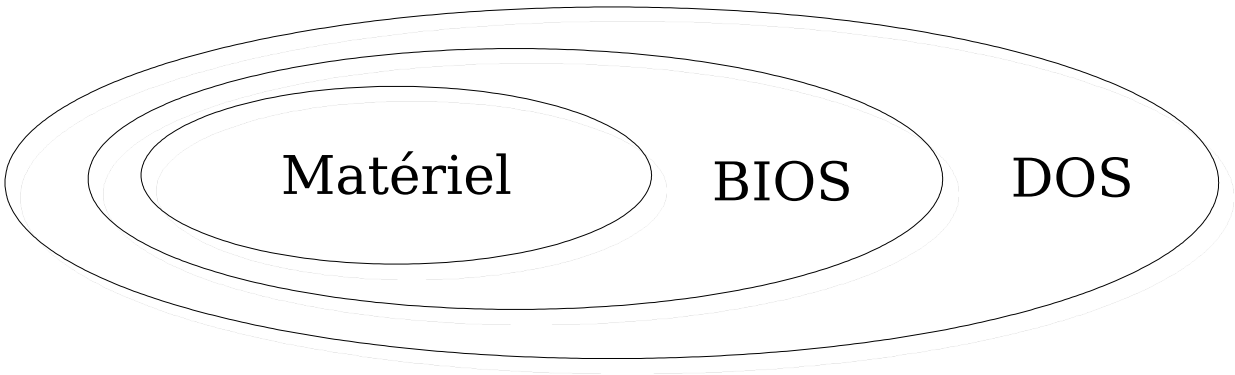
- Tout programme est formé (qq soit langage) :
  - D'instructions internes (au langage)
  - D'appel à des primitives externes
    - Généralement fournies par une bibliothèque
    - Au travers d'une API (Application Programming Interface)
  - Schéma récursif
    - Une bibliothèque peut s'appuyer sur les services d'autres bibliothèques ...

- Système d'exploitation
  - Fournit les primitives externes « ultimes »
    - Le plus bas niveau accessible par un programme

Illustration Concrète : le Cas Simple(iste) MSDOS

S'appuie sur une couche logicielle intermédiaire : le BIOS

- Fournie par le constructeur
- Offre des services de bas niveau élémentaires
  - Tests (POST), « Trouver boot sector », « lire secteur disque n°XX et le stocker en mémoire à l'adresse mémoire YY », ...



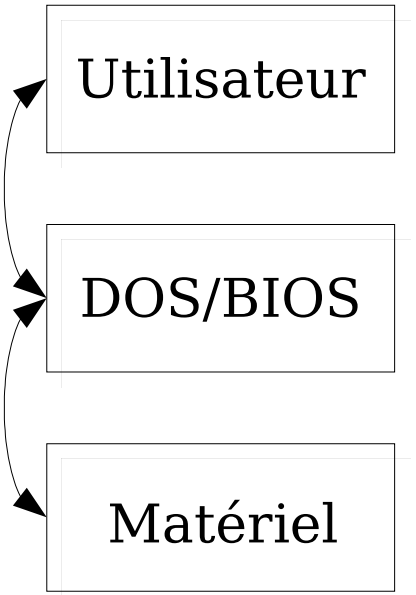
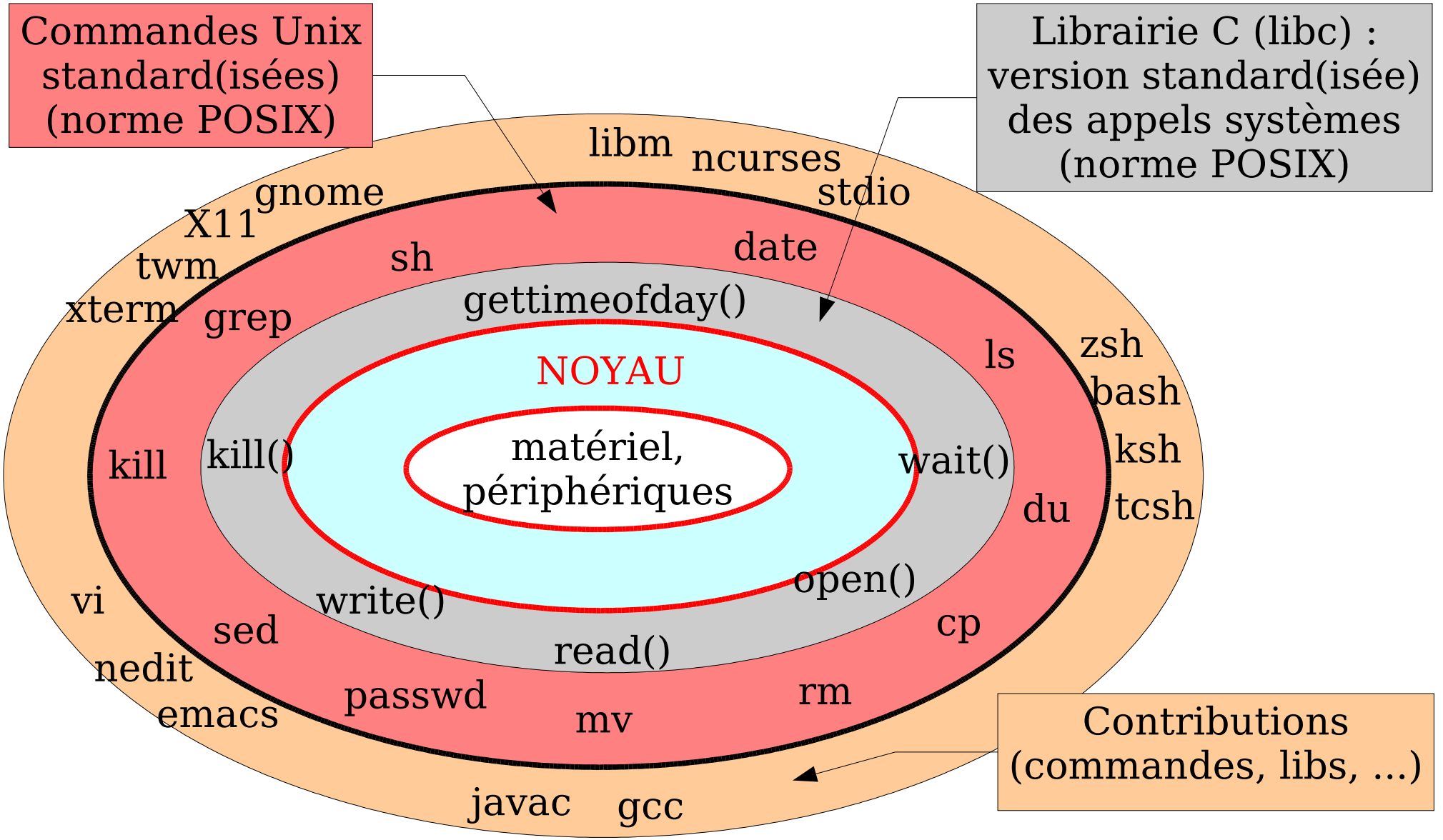
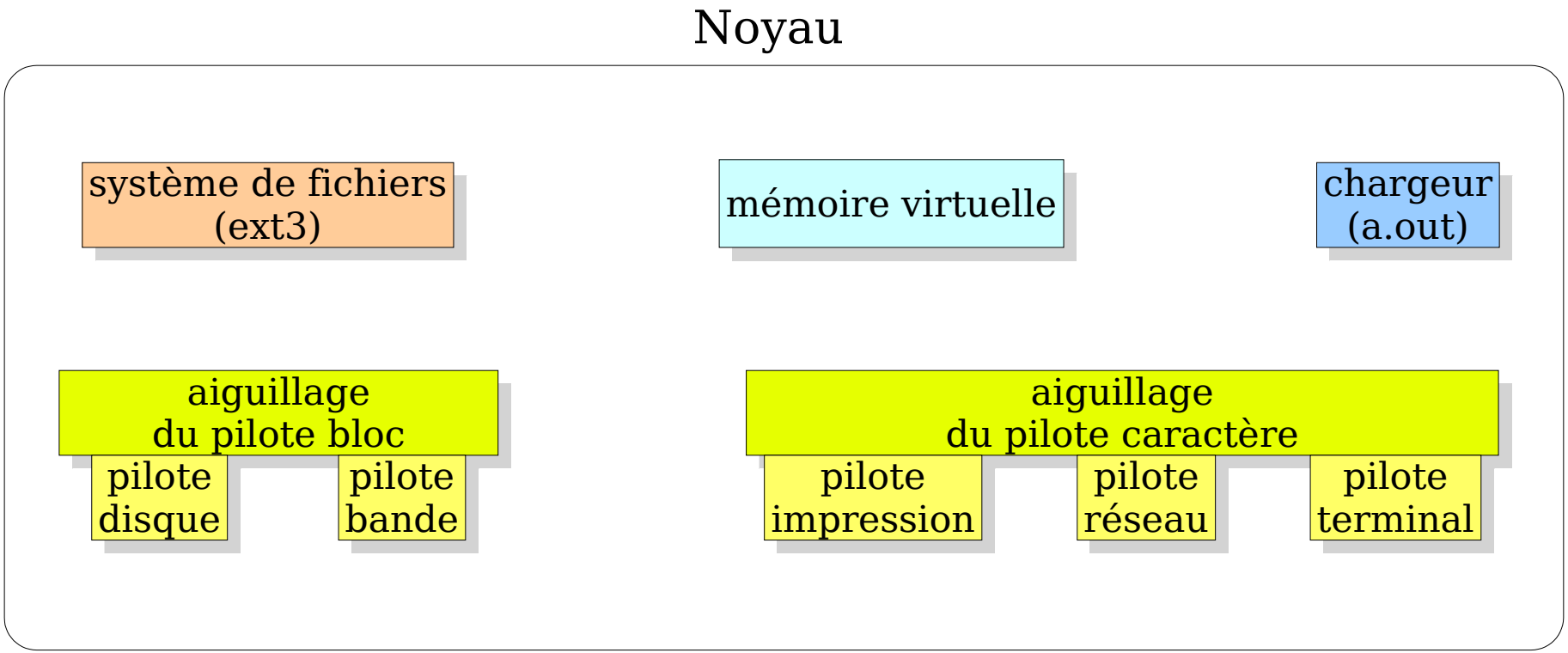


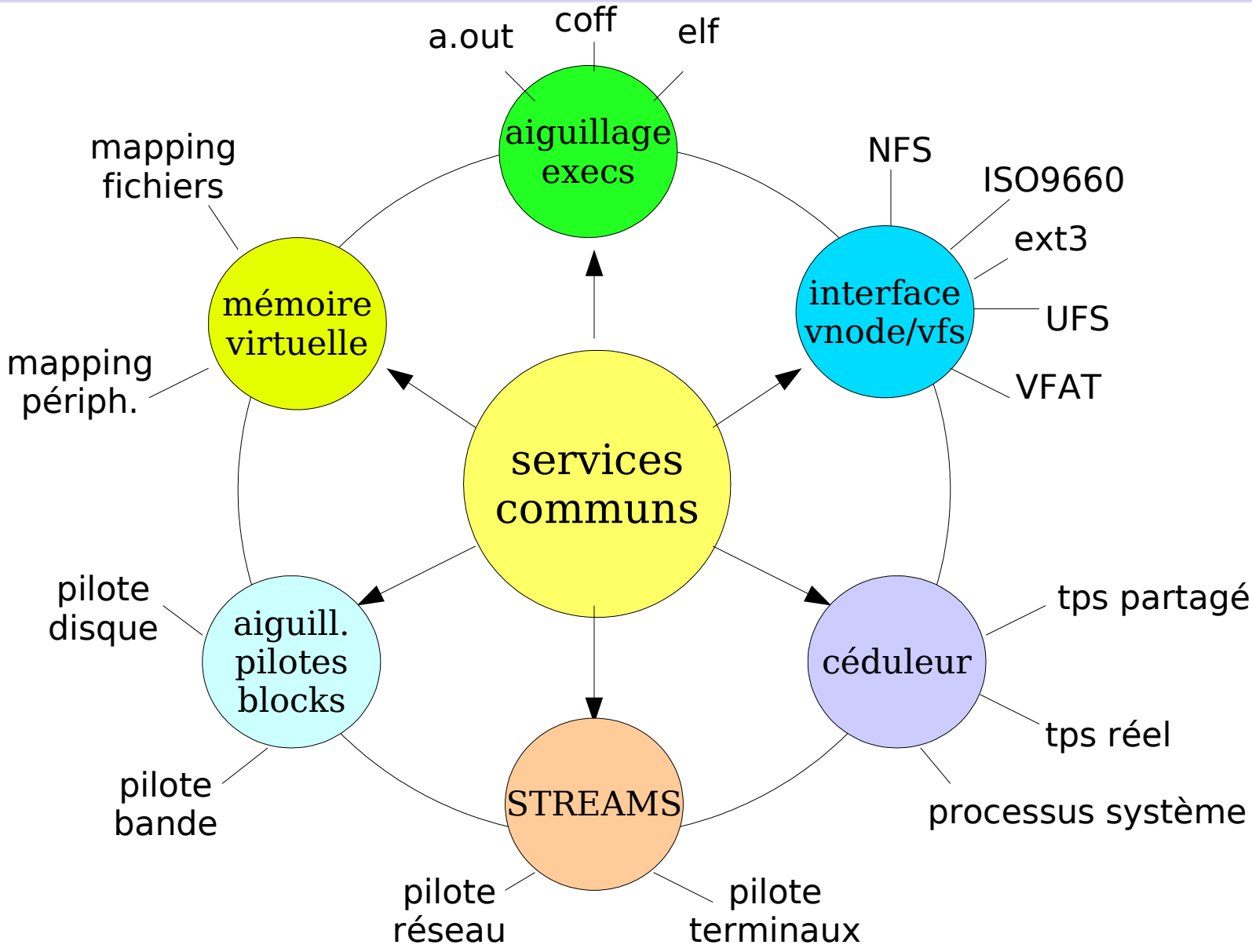
Schéma Global d'Organisation d'Unix



Noyau Unix Traditionnel (en voie de disparition)



Noyau Unix Moderne



Obtenir un service du Système : l'Appel Système

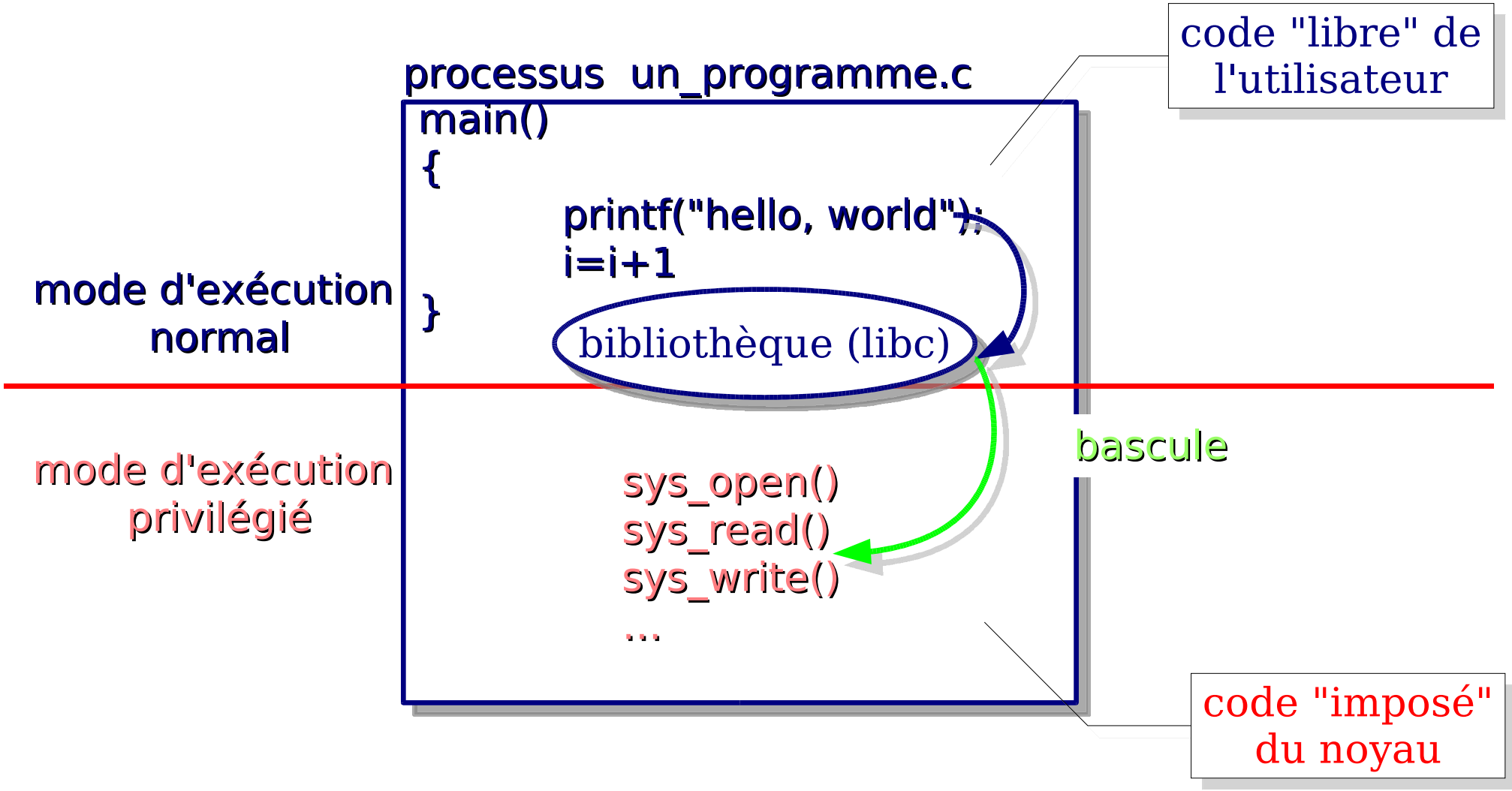
- Principe fondamental : créer des « points de passage » obligatoires
  - Le SE est le seul intermédiaire possible pour obtenir certains services
    - Accès fichiers, lancement programmes, manipulation de partitions, allocation mémoire, transmission de messages,...
  - Appel Système = (seul) moyen de communiquer avec le SE (pour un programme)
- l'AS ressemble à un appel de fonction ...
  - ... Mais son exécution est strictement protégée
    - Protection robuste assurée au niveau matériel (du CPU)
    - Exécution déclenchée par interruption logicielle

Mécanismes de Protection de l'Exécutif Système

Combinaison de deux mécanismes :

- Mode d'exécution privilégié
  - CPU bascule (du mode utilisateur) en mode noyau
    - Instructions supplémentaires (ex: masquer interruptions)
    - Registres supplémentaires (ex: gestion mémoire)
  - Bascule strictement contrôlée
    - Les programmes sont libres de basculer quand ils le veulent
    - La bascule entraîne l'exécution d'un (sous-)programme **préexistant**
      - Impossible de modifier les instructions de ce (sous-)programme depuis le mode utilisateur
- Espace d'adressage système
  - Zone mémoire réservée
    - Inacessible en mode utilisateur
    - Contient le code et les données de l'exécutif système

Illustration : Passage en Mode Noyau



La Notion de Processus

- Processus = instance d'un programme en cours d'exécution

- Dans un système multi-tâche
  - Les processus sont en compétition pour l'obtention des **ressources**
    - En particulier le processeur
    - Mais aussi : disques, réseau, mémoire, imprimante, ...
  - Partage du CPU (*time-sharing*)
    - Donne l'illusion d'exécution parallèle
      - pseudo-parallélisme sur mono-processeur
      - pseudo et vrai parallélisme sur multi-processeurs

Unix : un système basé des idées simples

- A l'origine seulement deux abstractions
  - Les processus : entités actives du système
  - Les fichiers : entités « passives »
    - Pour stocker données et programmes
    - Pour communiquer avec les pilotes et périphériques
      - Une souris est un fichier
      - Un disque dur est un fichier
      - Les partitions d'un disque sont des fichiers
      - Le lecteur de cdrom est un fichier ...
    - Pour faire communiquer des processus entre eux (tubes)
- Avec le temps : apparition de nouvelles abstractions
  - sockets (BSD) : connexions réseaux
  - STREAMS (SysV) : généralisation des flux
  - IPC (SysV) : sémaphores, mémoire partagée, file de msg