Systèmes centralisés

Philippe Quéinnec

http://queinnec.perso.enseeiht.fr/Ens/so.html

20 avril 2020

Déroulement de l'enseignement

- Introduction, rappel des concepts (Philippe Quéinnec)
- L'API système en C (Zouheir Hamrouni)
- L'API système en ligne de commande (Zouheir Hamrouni)
- Conception interne d'un système (Philippe Quéinnec)
  - Contrôle des processus
  - Mémoire virtuelle
  - Système de fichiers

Plan

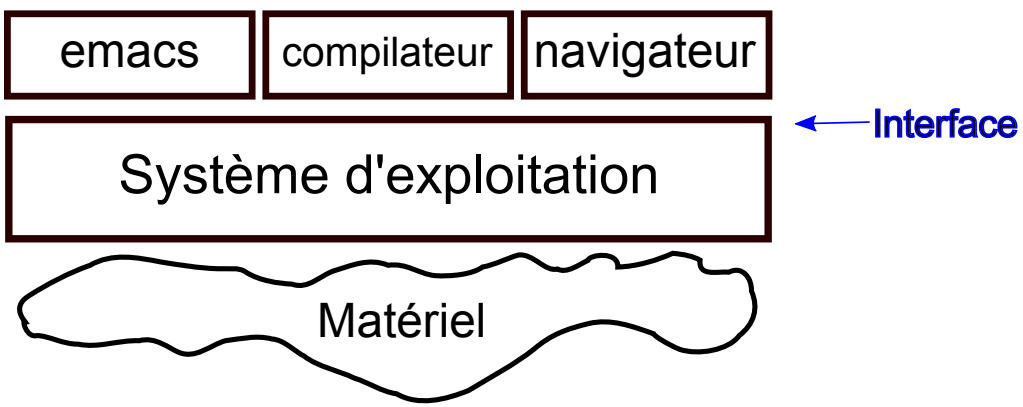
Interfaces

- Un service est caractérisé par son interface
  - L'interface est l'ensemble des fonctions accessibles aux utilisateurs du service
  - Chaque fonction est définie par
    - son format (la description de son mode d'utilisation) - sa syntaxe
    - sa spécification (la description de son effet) - sa sémantique
    - Ces descriptions doivent être précises, complètes (y compris les cas d'erreur) et non ambiguës
- Principe de base : séparation entre interface et réalisation
  - Les descriptions de l'interface d'un service doivent être totalement indépendantes du mode de réalisation du service
  - Facilite la portabilité
  - Permet de remplacer une réalisation du service par une autre, à condition qu'elle réalise la même interface

Définition

Qu'est-ce qu'un système d'exploitation ?

Le système d'exploitation est l'intermédiaire entre l'ordinateur matériel et les applications qui l'utilisent.



Rôles

Adaptation d'interface

Le système fournit une interface plus commode à utiliser que celle du matériel :

- plus haut niveau d'abstraction
- dissimulation des limitations physiques et du partage des ressources entre programmes / utilisateurs

Gestion de ressources

Gère les ressources matérielles : mémoire, processeurs, disques...
Gestion = allocation, partage, protection

Un mal nécessaire

Le système d'exploitation est indispensable :

- pour gérer efficacement l'exécution des programmes
- pour assurer le confinement des erreurs en cours d'exécution
- pour gérer efficacement les ressources : processeurs, mémoires...
- pour offrir une interface d'utilisation « agréable » aux usagers
- pour garantir une protection des données entre usagers

Un programme difficile à développer

Des programmes complexes qui ont eu un impact

- Sur le génie logiciel :
  - Modularité
  - Couches de logiciel
  - Machine virtuelle
  - Parallélisme
- Sur les architectures matérielles
  - Notion de mode d'exécution
  - Mécanismes de protection mémoire
  - Notion d'interruption, de déroutement

Interfaces d'un système d'exploitation

Un système d'exploitation présente en général plusieurs interfaces :

- interface de programmation (Application Programming Interface)
  - utilisable depuis des programmes s'exécutant sur le système
  - ensemble d'appels systèmes
- interface utilisateur (interface de commande)
  - utilisable par un usager humain
  - ensemble de commandes
    - textuelles (exemple : `rm foo.*`)
    - graphiques (exemple : déplacer l'icône d'un fichier vers la corbeille)
- l'interface utilisateur est programmée au moyen de l'interface de programmation

Exemples d'interfaces sur Unix

- Interface de programmation en C

```
...
while (nbbytes != 0) {
      int lu = read (from, buf, MSGSIZE);
      if (lu <= 0) return ERROR;
      nbbytes -= lu; char *p = buf;
      while (lu != 0) {
          int ecrit = write (to, p, lu);
          if (ecrit == -1) return ERROR;
          p += ecrit; lu -= ecrit;
      }
 }
```

- Interface textuelle de commande

  `cp fic1 fic2`

- Interface graphique de commande

  Sélectionner l'icône de `fic1`, appuyer sur la touche ctrl + déplacer l'icône vers le dossier destination

API/ABI

ABI Application Binary Interface : définition d'une interface au niveau du *code exécutable* : taille des types, convention d'appels des fonctions, format des fichiers de code...

API Application Programming Interface : définition d'une interface au niveau du *code source* : noms des fonctions, types des paramètres...

- API système : API offerte par le système d'exploitation (= les « appels systèmes »)

```
fd = open("toto.txt", O_RDONLY);
read(fd, buf, 100);
close(fd);
```

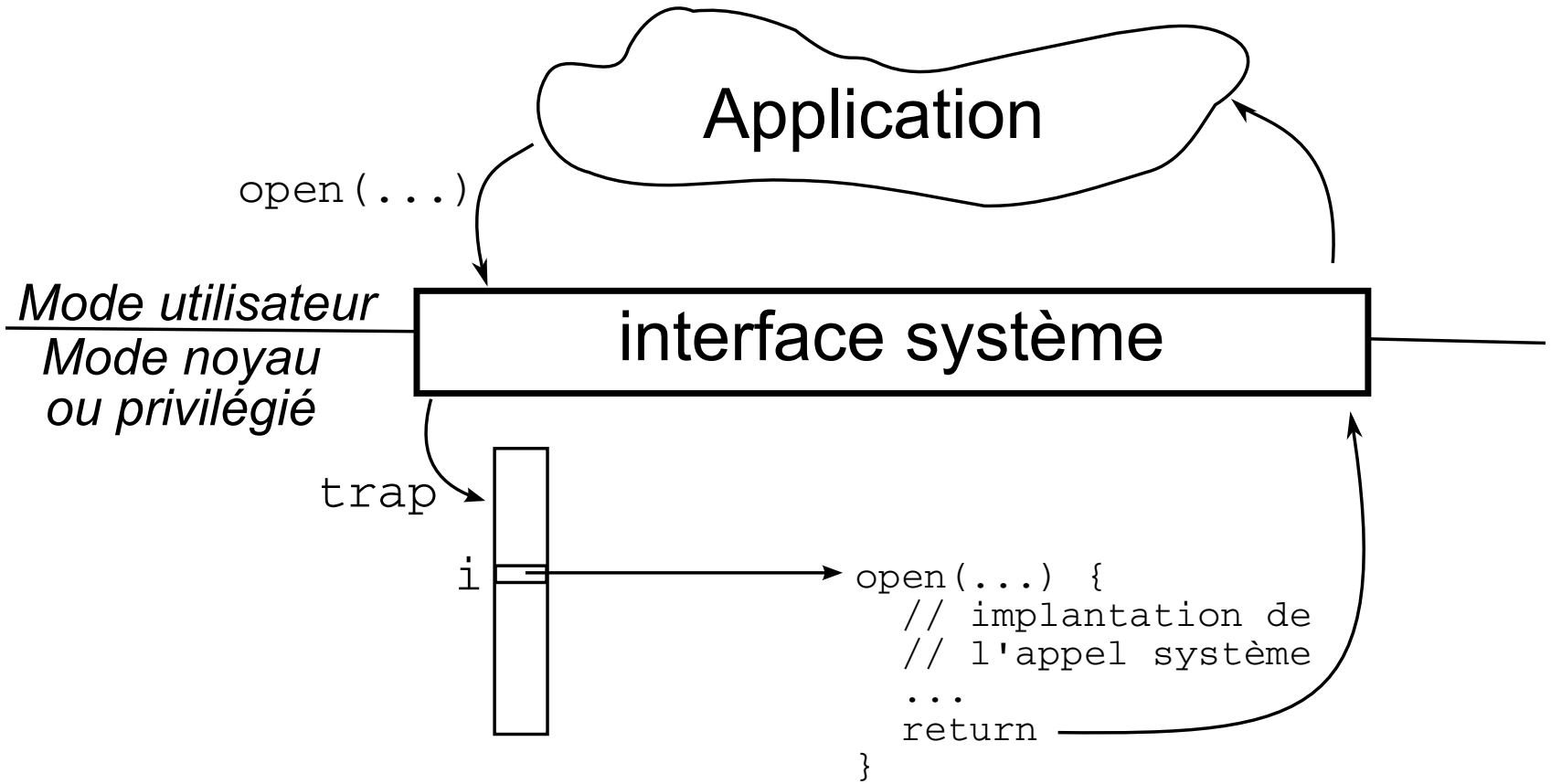
- API langage : API du langage de programmation, implantée via l'API système

```
open(fichier, Name => "toto.txt");
get_line(fichier, chaine);
close(fichier);
```

Appels système

- Le système d'exploitation implante une machine abstraite système
- Tout programme exécutable peut faire appel aux instructions du noyau durant son exécution : appels système ou primitives du noyau
- Ces primitives implantent une machine système permettant d'utiliser des concepts de haut niveau : processus, exceptions, mémoire virtuelle, fichiers, pipes, sockets, etc
- Ces primitives définissent une machine abstraite indépendante de la configuration matérielle sous-jacente $\Rightarrow$ Portabilité des programmes
- Ces primitives sont du code partagé par tout programme qui s'exécute

Appels système



Appels système

- Effectuer des opérations contrôlées, qu'une application ne peut pas faire en mode utilisateur
- Droits différents entre mode applicatif et mode privilégié
- Similaire à un appel de procédure, mais vers du code noyau
- Interface noyau prédéfinie
- Support processeur pour basculer en mode privilégié (`trap`)
- Généralisation : anneaux de protection
- Distinguer les appels systèmes (`open`, `write`...) et les fonctions de plus haut niveau construites avec (en C : `fopen`, `printf`...)

Plan

Concepts apportés par l'OS

Gestion d'activités

- déroulement de l'exécution
- processeur → processus

Gestion d'information

- conservation de l'information
- partage de l'information
- mémoire principale + disque → mémoire virtuelle
- disque → fichier

Gestion des interactions

- interface avec l'utilisateur
- impression, connexion réseau, périphériques d'E/S
- écran, clavier, souris → système de fenêtrage
- imprimante, réseau,... → flot de données

Processus

Définition

Un processus est l'entité dynamique représentant l'exécution d'un programme.

Programme = description statique
≠ Processus = activité dynamique

Intérêt

- abstraction de la notion d'exécution séquentielle, indépendance de la disponibilité effective d'un processeur
- représentation des activités parallèles
- unité d'allocation des ressources
- unité de protection et d'isolation
- unité comptable

Forme d'un programme

Programme source

```
int main() { printf("coucou"); }
```

Fichier binaire exécutable

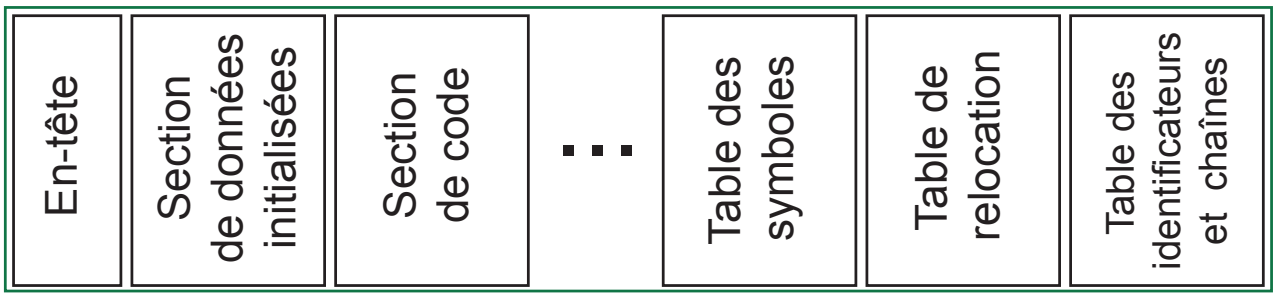
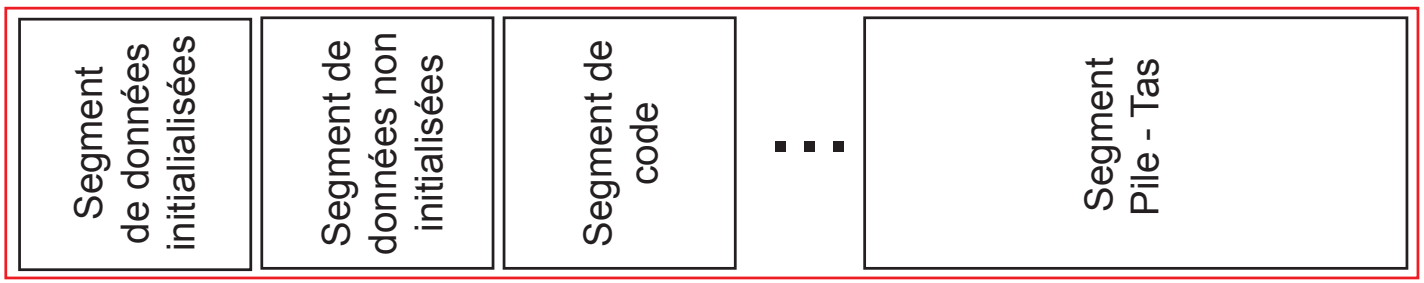
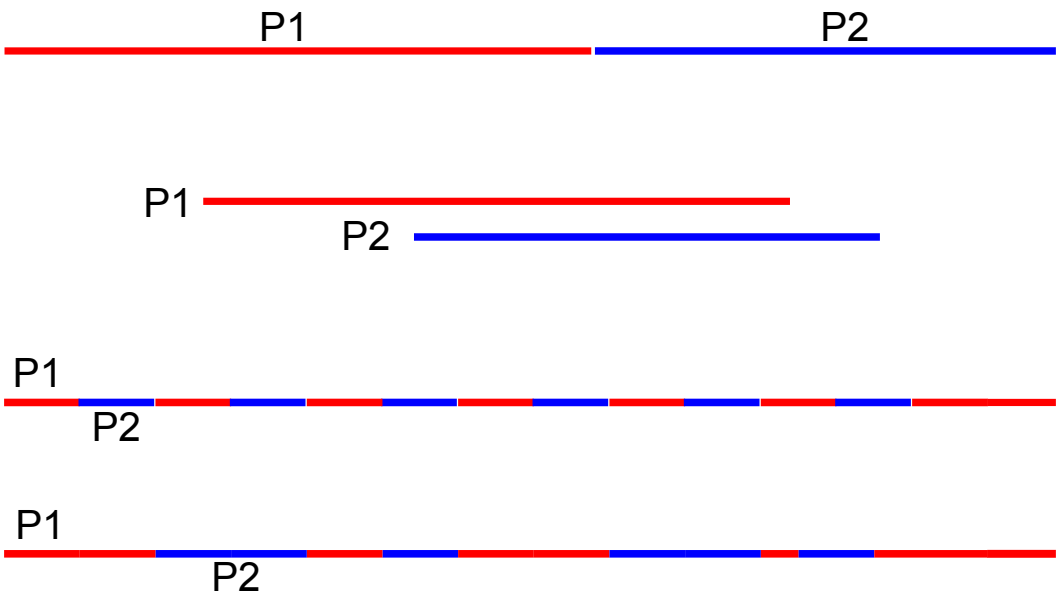


Image en mémoire



Parallélisme et pseudo parallélisme



exécution séquentielle

exécution parallèle (2 processeurs)

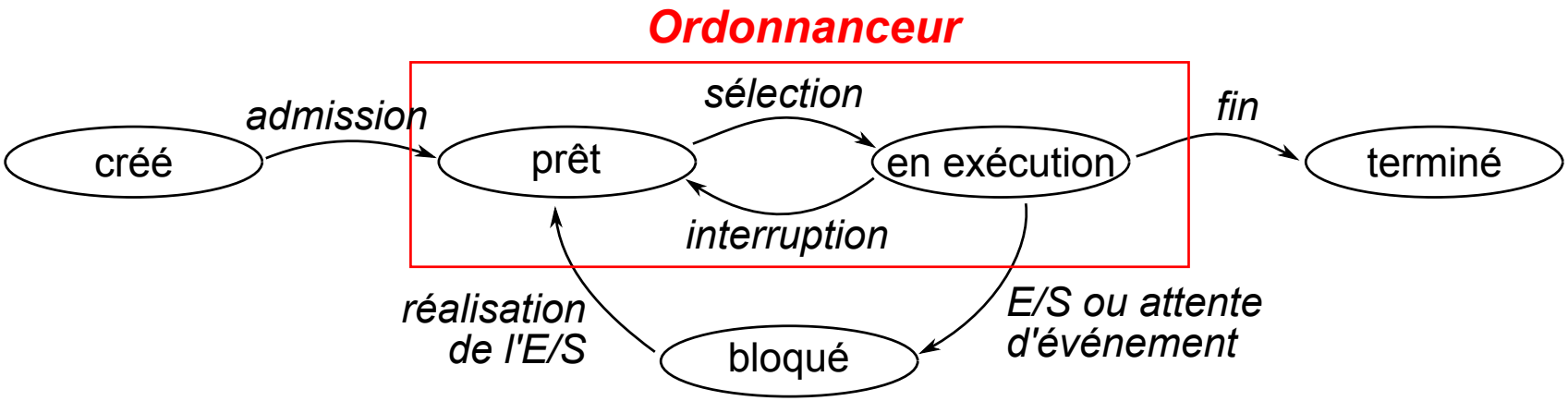
exécution pseudo-parallèle (1 processeur)

une autre exécution pseudo-parallèle

(Pseudo-)parallélisme

- Faire s'exécuter plusieurs processus à la fois (ou comme si)
- Ne pas bloquer le système quand un processus se bloque (entrée/sortie)
- Se protéger contre un processus qui boucle indéfiniment sans libérer le processeur
  → Préemption
- Communiquer avec l'environnement
  → Flot de données
- Se protéger contre un processus qui lit ou écrit n'importe où en mémoire (données d'un autre processus ou données du système)
  → Protection mémoire
- Donner l'illusion que chaque processus a autant de mémoire qu'il le souhaite
  → Mémoire virtuelle

Préemption



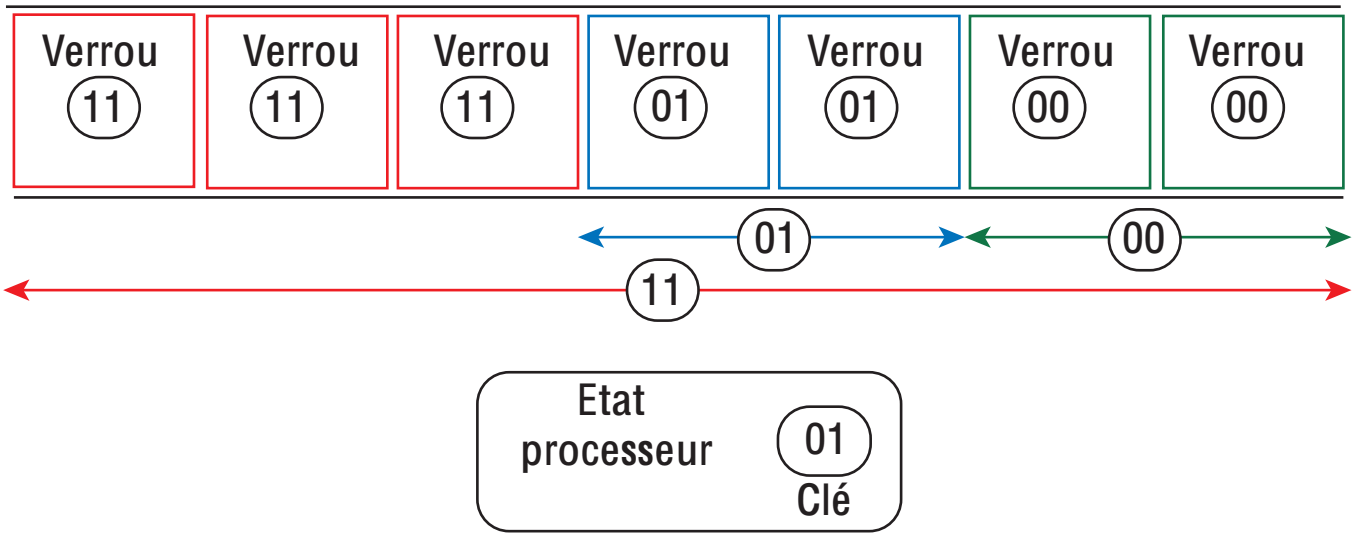
- partage du temps processeur
- interruption forcée du processus en cours pour libérer le processeur
- nécessité de sauvegarder le contexte d'exécution du processus
- politique d'allocation / sélection

Communications avec l'environnement

Un processus communique avec son environnement :

- par interruptions ou déroutements : signaux
- directement par flots de données entre processus (tube ou pipe)
- indirectement par flots de données liés à des données rémanentes : fichiers

Protection mémoire : un exemple élémentaire



- Besoin d'une clé passe-partout
- Nécessité d'interdire la programmation des verrous et du registre clé : Modes d'exécution

Solution actuelle : les Unités de Gestion Mémoire (MMU : Memory Management Unit)

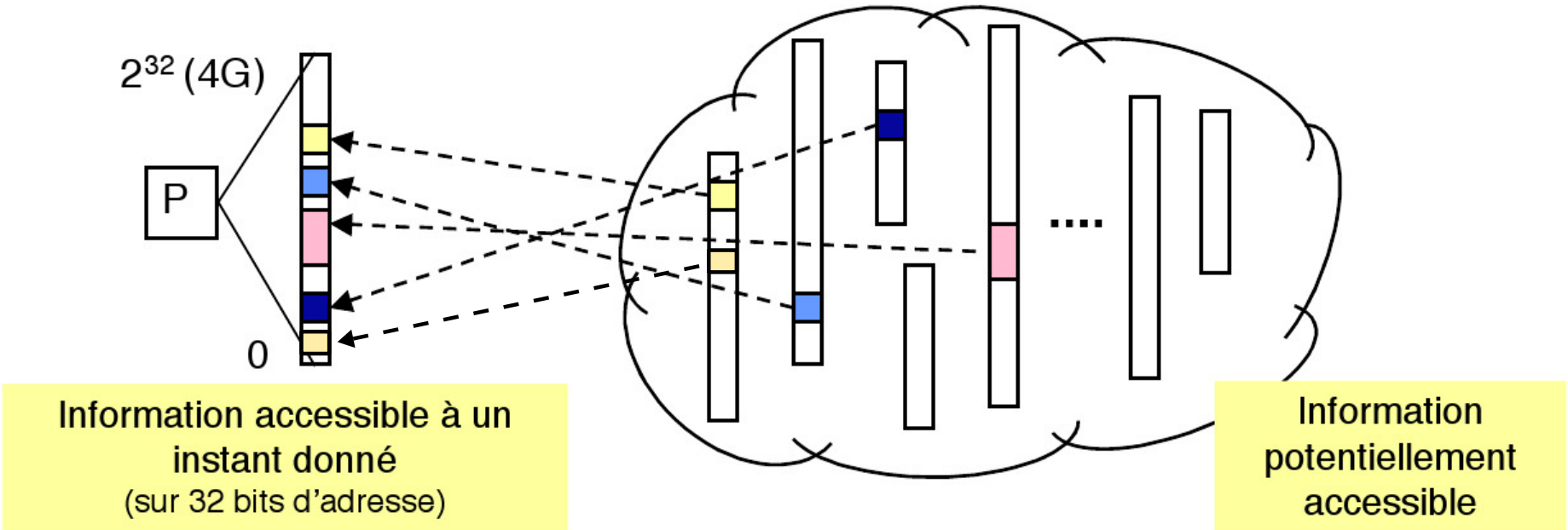
Mémoire virtuelle

Mémoire virtuelle d'un processus

Ensemble des emplacements accessible à ce processus, via une adresse dite virtuelle.

- Fenêtre sur l'ensemble des informations accessibles (le processeur ne peut adresser qu'un espace limité)
- Gestion économique des ressources : l'espace disque est beaucoup moins cher que la mémoire RAM
- Indépendance et protection des processus : chaque processus a sa propre mémoire virtuelle

Mémoire virtuelle



Source : S. Krakowiak ≪ Introduction aux systèmes et réseaux informatiques ≫.

Fichiers

Fichier

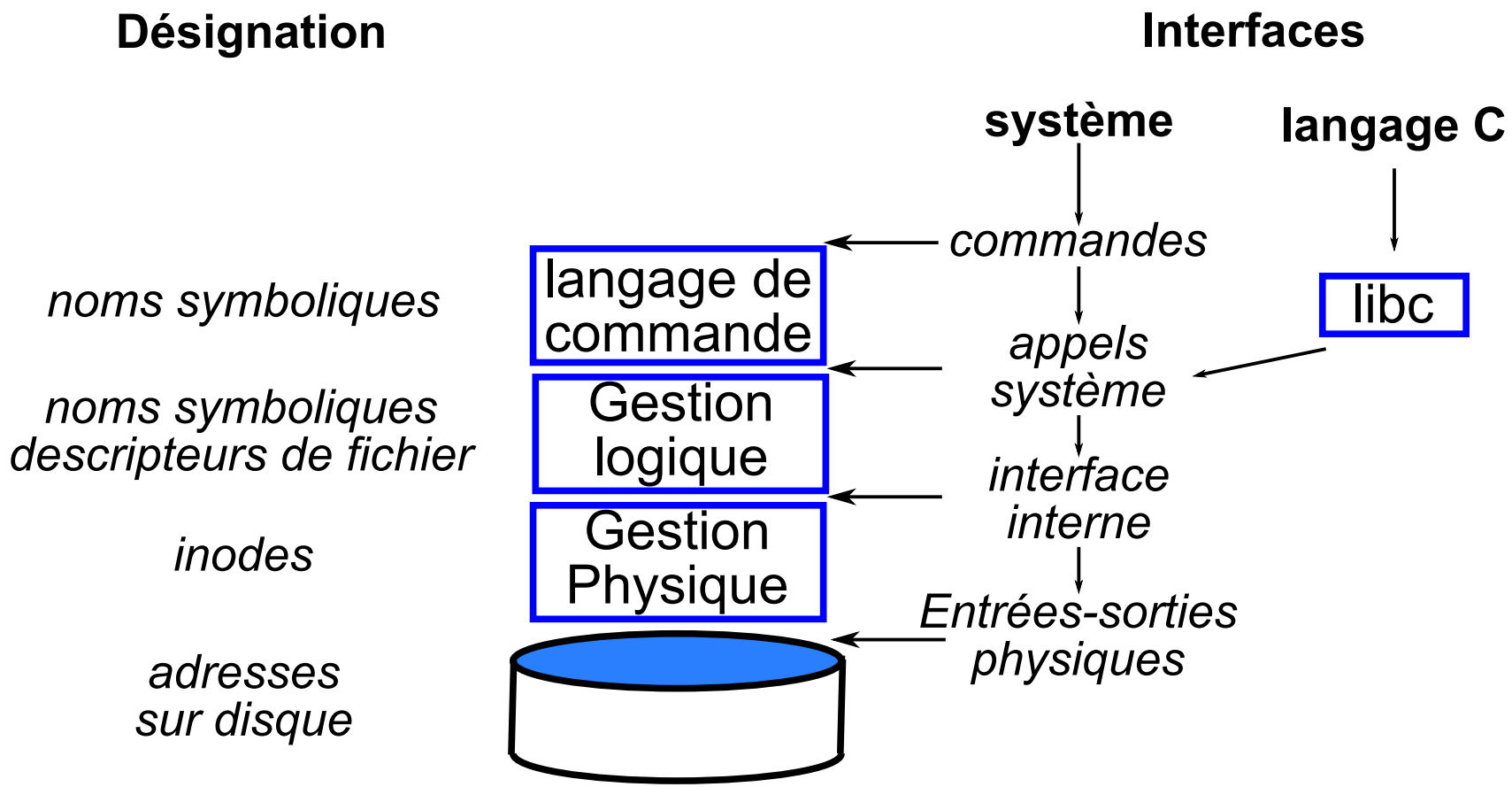
Ensemble d'informations regroupées en vue de leur conservation et de leur utilisation ultérieure

Système de gestion de fichiers (SGF/FS)

Partie du système d'exploitation qui conserve les fichiers et permet d'y accéder

- Conservation permanente des fichiers = indépendamment de l'exécution des programmes et de l'intégrité de la mémoire principale
- Organisation logique et désignation
- Partage et protection des fichiers

Organisation du SGF



Désignation symbolique

Organisation hiérarchique

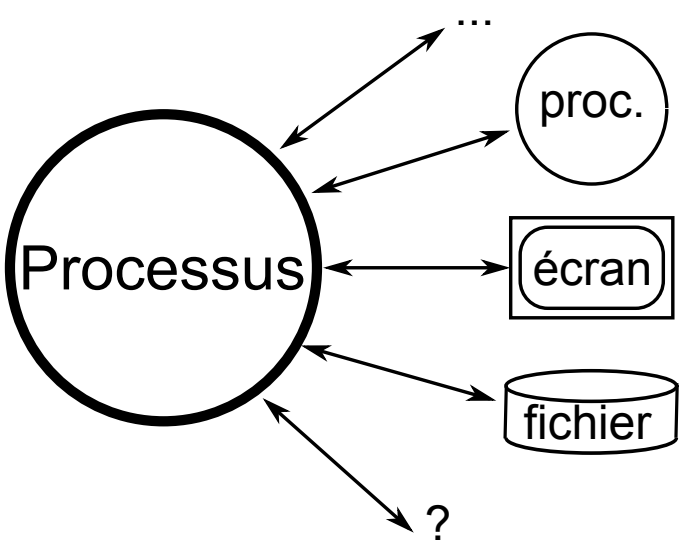
- Les noms forment une arborescence
- Nœuds intermédiaires = répertoires/directories (ce sont aussi des fichiers)
- Nœuds terminaux = fichiers simples (`toto.c`)
- Nom absolu = chemin d'accès depuis la racine (`/home/queinnec/toto.c`)
- Nom relatif = chemin relatif au répertoire courant (`../queinnec/toto.c`)

Flot de données

Tout accès à l'information « fichier » est faite exclusivement par des flots de données :

- séquence d'octets non structurés
- lire = consomme dans le flot / écrire = ajoute dans le flot
- s'oppose à un accès mémoire (RAM = accès arbitraire) ou un accès structuré (base de données)
- exemple de flots = entrée clavier / fichier sur disque / lien réseau
- des bibliothèques utilisateur peuvent fournir des interfaces plus adaptées (e.g. lignes de caractères)

Flots de données



- Un processus peut ouvrir/fermer dynamiquement des connexions à des sources ou puits de données, via la notion de canal de communication = descripteur de fichier.
- Un processus hérite à sa création d'un environnement de communication standard

Événements

Interruptions

- Interruptions externes : fin d'E/S, horloge...
- Interruptions internes : division par zéro, branchement illégal...

→ déroutement vers une routine

Traitement

- interne au noyau : p.e. défaut de page mémoire
- utilisateur synchrone : p.e. E/S bloquante
- utilisateur asynchrone : p.e. signal

Plan

Quelques nombres

Opération	Latence (en ns)	Changement de repère
Exécuter 1 instruction CPU	0,1 - 10 (+ mémoire)	
Accès cache L1	0,5	1 s
Accès cache L2	6	12 s
Accès mémoire	100	3 m 20 s
Compresser 1 Ko	2 000 = 2 μs	1 h 6 m
Envoyer 1 Ko sur réseau 1 Gb/s	10 000 = 10 μs	5 h 33 m
Lire 1 Mo depuis la RAM	6 000 = 6 μs	3 h 20 m
Lire 1 Mo depuis disque SSD	100 000 = 0,1 ms	2 j 7 h 33 m
Position tête disque dur	3 000 000 = 3 ms	69 j 10 h
Lire 1 Mo depuis disque dur	1 000 000 = 1 ms	23 j 3 h
Paquet USA $\rightarrow$ Europe $\rightarrow$ USA	150 000 000 = 150 ms	9 an 184 j

Démarrage / boot

Si le système d'exploitation est un programme qui permet le démarrage et l'exécution des programmes, qui démarre le système ?

1. À la mise sous tension, un programme en ROM s'exécute (le BIOS sur PC) ;
2. Ce programme consulte les périphériques permanents (disque dur en général), charge en mémoire le premier bloc (master boot record) du premier périphérique présent et exécute le code contenu dans ce bloc ($\sim$ 400 octets) ;
3. Ce code contient l'emplacement + le code pour charger un second chargeur (de taille arbitraire) : après chargement, il transfère le contrôle à ce chargeur ;
4. Ce second chargeur peut présenter un menu de sélection entre plusieurs OS, charge en mémoire l'image de l'OS choisi (ou par défaut), et lui transfère le contrôle ;
5. L'OS récupère la mémoire utilisée par les chargeurs initiaux.

Les processus

La machine à processus

1. couche contexte d'exécution
2. couche pseudo-parallèle (coroutine = contexte indépendant du matériel)
3. couche parallèle (processus)
4. ordonnanceur

Mémoire virtuelle

1. principe
2. lien avec l'ordonnanceur

Système de fichiers

Les couches

1. couche physique : accès blocs disque
2. couche gestion des blocs (allocation/libération)
3. couche bufferisation des blocs en mémoire
4. couche inodes
5. couche accès fichier : descripteur de fichier
6. couche désignation/nommage

Plan

Conclusion

Un système d'exploitation...

est un (ensemble de) logiciel(s) qui fournit :

- des unités d'exécution et d'allocation de ressources (processus)
- la gestion des données temporaires (mémoire virtuelle)
- la gestion des données rémanentes (fichiers)

en masquant le plus possible les détails matériels.

Un programme un peu particulier

- Réactif : requête $\rightsquigarrow$ réponse
- Indispensable : sans lui, difficile de faire exécuter un programme à une machine
- Service commun utilisable par tous les autres programmes
- Cohabitation de programmes en mémoire centrale
- Robuste vis-à-vis des programmes exécutés
- Parallèle : entrelace l'exécution de plusieurs programmes
- De taille respectable
- Discret (performance, adaptations nécessaires)
- Très fiable