



Université Abdelmalek Essaadi
Faculté des Sciences et Techniques de
Tanger

FST
Tanger

Systèmes et Réseaux Informatiques

Pr. Abdelhamid ZOUHAIR

Intitulé du module	Systèmes et Réseaux Informatiques
Etablissement dont relève le module	Faculté des Sciences et Techniques de Tanger
Filière	Licence en Ingénierie de Développement d'Applications Informatiques
Semestre d'appartenance du module	S5
Version 1	A. U: 2023/2024

Objectif du Module

Systèmes :

- Présenter les systèmes d'exploitation UNIX/Linux: Structure d'un système d'exploitation, les commandes UNIX/Linux, droits d'accès, les Processus, Système de gestion des fichiers UNIX/Linux...

Réseaux :

- Compréhension du rôles des différents matériels et logiciels dans un réseau local.
- Les techniques d'installation et de configuration des composants d'un réseau local : câblages, cartes réseaux, protocoles, serveurs d'applications...
- Les méthodologies d'installation de différentes solutions de réseaux locaux.

Introduction à l'administration d'un Réseau d'ordinateurs.

2

Plan : Système d'Exploitation

1. INTRODUCTION AUX SYSTEMES D'EXPLOITATION
2. LES PROCESSUS
3. LES ENTREES/SORTIES
4. LA GESTION DE LA MÉMOIRE
5. LES SYSTEMES DE FICHIERS



3

Système d'Exploitation

INTRODUCTION AUX SYSTEMES D'EXPLOITATION

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

2- Structure d'un système d'exploitation

- a- Systèmes monolithiques
- b- Systèmes à couches
- c- Machines virtuelles
- d- Modèle Client / Serveur

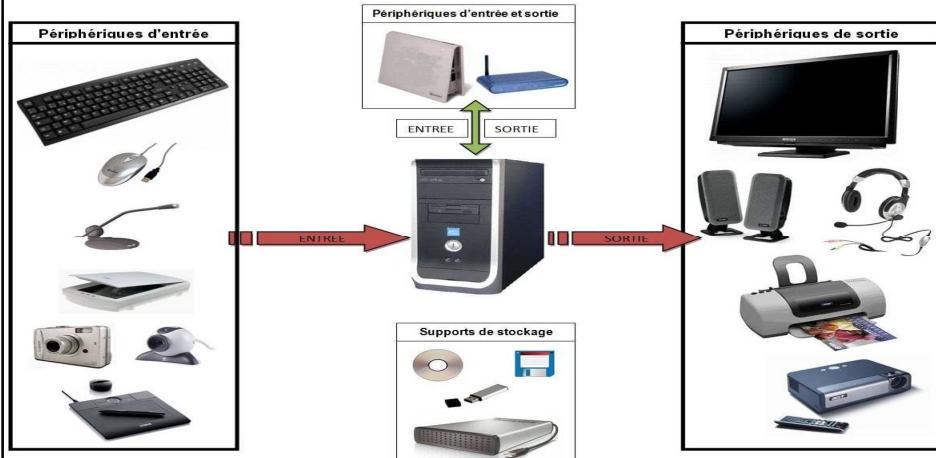
3- Structure du système d'exploitation Unix / Linux



4

Système d'Exploitation

Introduction



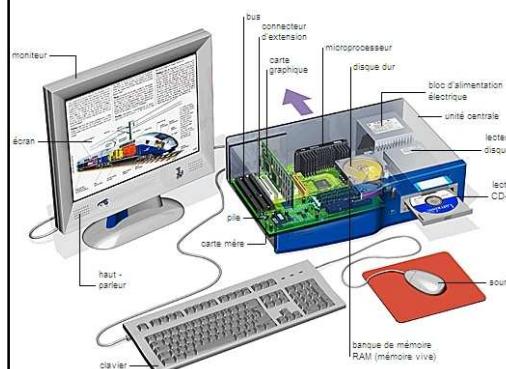
5

Système d'Exploitation

Un ordinateur est constitué :

1. Du matériel

- Dispositifs physiques
- Langage machine

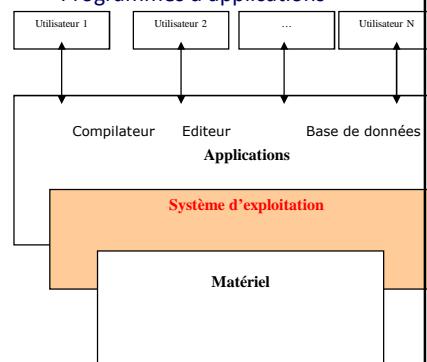


2. Et du Logiciels (Software)

■ Système d'Exploitation

■ Programmes

- Programmes système
- Programmes d'applications



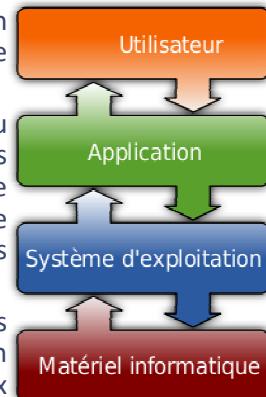
Qu'est ce qu'un système d'exploitation ?

6

Système d'Exploitation

■ Définition (Système d'Exploitation)

- Un système d'exploitation (*Operating System* ou OS) est un ensemble de programmes spécialisés qui permet l'utilisation des **ressources** matérielles d'un ordinateur. Il assure le démarrage (**Boot**) de l'ordinateur et l'exécution des logiciels applicatifs.
- Le système d'exploitation est composé d'un noyau (**kernel**) où sont regroupés tous les programmes basiques du système, d'une interface (**shell**) qui joue le rôle d'intermédiaire entre l'utilisateur et le système d'exploitation et d'un système de fichiers (**File System ou FS**) pour la gestion des données.
- D'autres programmes appelés pilotes génériques sont souvent inclus dans le système d'exploitation afin de permettre l'interaction avec de nouveaux périphériques (clavier, souris imprimante, disque dur, etc.).



7

Système d'Exploitation

Exemples de systèmes d'exploitations :

- Il existe un grand nombre de systèmes d'exploitation, mais les systèmes d'exploitation les plus répandus sont:
 - **Windows** (pour les PC et les serveurs) ;
 - **Mac OS** (pour les ordinateurs Mac d'Apple) ;
 - **GNU/Linux** (pour les PC et les serveurs) et
 - **Unix** (pour les serveurs).
- Pour les téléphones, on trouve **Android** (Google), **iOS** (iPhone et iPad: Apple), **Symbian** et **Windows Phone** de Microsoft.



Système d'Exploitation

■ Les principales fonctions d'un Système d'Exploitation

- la gestion du **processeur** qui réalise les opérations d'ordonnancement des processus.
- la gestion des **opérations** de mise à jour des processus ainsi que leur synchronisation et la communication entre eux.
- la gestion de la **mémoire** qui consiste essentiellement à réaliser les opérations de l'allocation et du suivi de l'occupation mémoire.
- la gestion des **entrées / sorties**.
- la gestion des **réseaux**.
- la gestion des **utilisateurs** : gérer les droits d'accès aux fichiers, comme au matériel.
- **Les périphériques** : écran, imprimante, disque dur, réseau. Il s'assure que les programmes puissent les utiliser de façon standard.

La partie du Système d'Exploitation qui assure les services de base comme la communication entre matériel et logiciel est appelé **noyau.**

9

Système d'Exploitation

Structure d'un Système d'Exploitation

- Le noyau (**kernel**) d'un système d'exploitation est **l'ensemble logiciel indivisible** minimal qui est systématiquement chargé au démarrage.



Système d'Exploitation

Structure d'un Système d'Exploitation

☞ Le noyau est généralement exécuté dans un espace mémoire séparé de l'espace des applications: **espace noyau**. Par opposition à **l'espace utilisateur**. Le passage entre ces deux espaces se fait via des appels systèmes. L'intérêt de cette séparation est que le système ne crash/se plante pas si une application plante.

▪ Les différents types d'architecture de systèmes d'exploitation (**kernel**).

1. Systèmes monolithiques
2. Systèmes à couches / Modulaire / Multicouches
3. machines virtuelles
4. modèle Client / Serveur

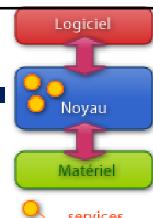


Système d'Exploitation

Structure d'un Système d'Exploitation

1. Systèmes monolithiques (non modulaires)

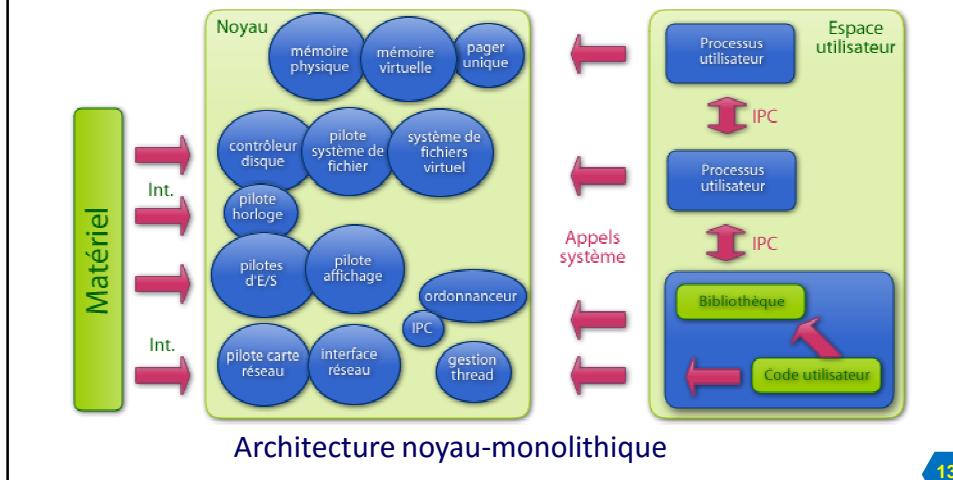
- ❑ Le SE **monolithique** (**programme unique et indivisible**) est un **ensemble de procédures**, chacune pouvant appeler toute autre à tout instant.
- ❑ Monolithique signifie formé d'un seul bloc. Il est conçu pour être autonome ; ses composants sont interconnectés et interdépendants plutôt qu'associés de manière flexible comme dans le cas des programmes modulaires.
- ❑ Pour effectuer un appel système, on dépose dans un registre les paramètres de l'appel et on exécute une instruction spéciale appelée **appel superviseur ou appel noyau**.
- ❑ Son exécution commute la machine du mode utilisateur au mode superviseur ou noyau et **transfère le contrôle au SE**.
- ❑ Le SE analyse les paramètres déposés dans le registre mentionné plus haut et en déduit la procédure à activer pour satisfaire la requête.
- ❑ A la fin de l'exécution de la procédure système, le SE rend le contrôle au programme appelant.



Système d'Exploitation

Structure d'un Système d'Exploitation

1.Systèmes monolithiques : Noyaux monolithiques non modulaires



Système d'Exploitation

Structure d'un Système d'Exploitation

1.Systèmes monolithiques : Noyaux monolithiques non modulaires

- Certains systèmes d'exploitation, comme d'anciennes versions de **Linux**, certains **BSD** ou certains vieux **Unix** ont un noyau monolithique. C'est-à-dire que l'ensemble des fonctions du système et des pilotes sont regroupés dans un seul bloc de code et un seul bloc binaire généré à la **compilation**.
- De par la simplicité de leur concept mais également de leur excellente vitesse d'exécution, les noyaux monolithiques ont été les premiers à être développés et mis en œuvre.
- Cependant, au fur et à mesure de leurs développements, le code de ces noyaux monolithiques a augmenté en taille et il s'est avéré difficile de les maintenir. Le support par les architectures monolithiques des **chargements à chaud** ou dynamiques implique une augmentation du nombre de pilotes matériels compilés dans le noyau, et par suite, une augmentation de la taille de l'empreinte mémoire des noyaux. Celle-ci devient rapidement inacceptable.
- Les multiples dépendances créées entre les différentes fonctions du noyau empêchaient la relecture et la compréhension du code. L'évolution du code s'est faite en parallèle à l'évolution du matériel, et des problèmes de portage ont alors été mis en évidence sur les noyaux monolithiques.

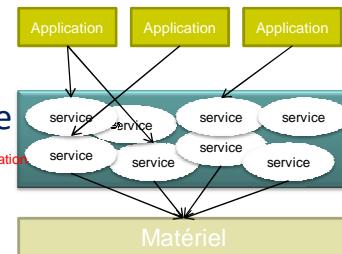
14

Système d'Exploitation

Structure d'un Système d'Exploitation

1. Systèmes monolithiques : Noyaux monolithiques non modulaires

- Un seul bloc contenant l'ensemble des services système (mode noyau).
 - Facilité de conception
 - Problème de surcharge de mémoire
 - Performance peut être au RDV ...
 - Code dur à maintenir
 - Dos, très vieux UNIX et Linux, etc.



15

Système d'Exploitation

Structure d'un Système d'Exploitation

2. Systèmes à couches / Modulaire : Noyaux monolithiques modulaires

- Pour résoudre les problèmes évoqués des noyaux monolithiques, les noyaux monolithiques sont devenus modulaires.
- Dans ce type de noyau, seules les parties fondamentales du système sont regroupées dans un bloc de code unique (monolithique).
- Les autres fonctions, comme les pilotes matériels, sont regroupées en différents modules qui peuvent être séparés tant du point de vue du code que du point de vue binaire.
- La très grande majorité des systèmes actuels utilisent cette technologie : **Linux**, la plupart des **BSD** ou **Solaris**.
- Par exemple avec le noyau **Linux**, certaines parties peuvent être non compilées ou compilées en tant que modules chargeables directement dans le noyau.
- La modularité du noyau permet le chargement à la demande de fonctionnalités et augmente les possibilités de configuration.
- Les distributions Linux, par exemple, tirent profit des modules chargeables lors de l'installation. L'ensemble des pilotes matériels sont compilés en tant que modules. Le noyau peut alors supporter l'immense variété de matériel trouvé dans les compatibles PC.
- Après l'installation, lors du démarrage du système, seuls les pilotes correspondants au matériel effectivement présent dans la machine sont chargés en mémoire vive. La mémoire est économisée.

16

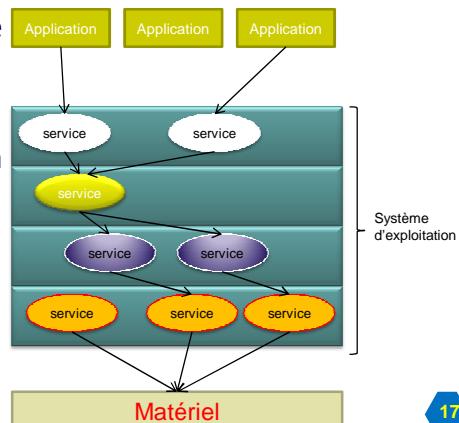
Système d'Exploitation

Structure d'un Système d'Exploitation

2. Systèmes à couches / Modulaire :Noyaux monolithiques modulaires

➤ **Système d'Exploitation** organisé en **hiérarchie de couches**. Chacune construite sur la base des services offerts par la couche inférieure.

- Facilité de conception et de développement
- Code plus organisé et maintenable.
- Chargement des fonctionnalités à la demande
- Linux, BSD, SOLARIS



17

Système d'Exploitation

Structure d'un Système d'Exploitation

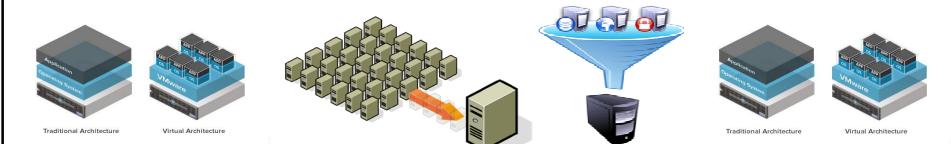
3. machines virtuelles

➤ Une **machine virtuelle** (virtual machine) est une illusion d'un appareil informatique créée par un logiciel d'émulation.

➤ La virtualisation : au lieu de multiplier les machines physiques avec un seul système d'exploitation, on utilise une machine physique pour virtualiser plusieurs systèmes d'exploitations. La virtualisation a notamment été créée pour répondre à la problématique de la sous-utilisation des ressources matérielles.

La virtualisation représente un des meilleurs pratiques informatique à savoir :

- **Architecture multi-processeurs** (plusieurs processeurs dans une machine).
- **Architecture multi-cœurs** (plusieurs cœurs dans un processeur).
- **Architecture hyper-threadé**.



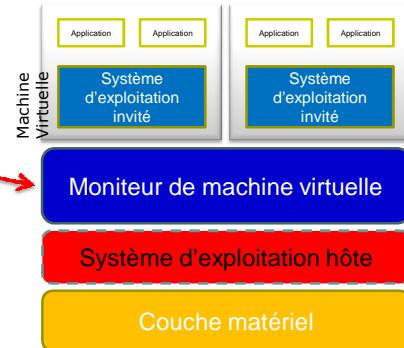
Système d'Exploitation

Structure d'un Système d'Exploitation

3. machines virtuelles

Possibilité de mettre plusieurs **Système d'Exploitation** sur une seule machine physique.

- Le moniteur de machine virtuelle (hyperviseur) intercepte les instructions privilégiées envoyées par le SE invité, les vérifie (politique de sécurité) et les exécute.
 - KVM (Kernel Virtual Machine)
 - Microsoft Hyper-V
 - VMware
 - Virtual PC
 - ...



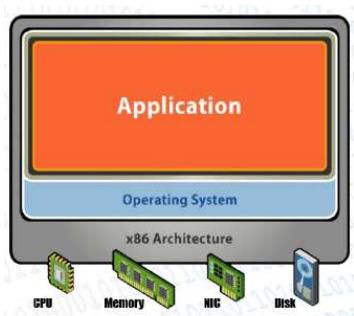
19

Système d'Exploitation

Structure d'un Système d'Exploitation

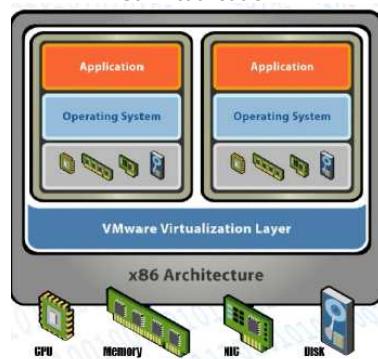
3. machines virtuelles

Normal



- Système d'exploitation / machine
- Exécution de plusieurs applications / serveur augmente le risque d'interruption de service global
- En général, 1 serveur = 1 application

Avec Virtualisation



- Ajout de la couche de virtualisation
- Chaque machine virtuelle possède ses propres applications et système d'exploitation
- Possibilités d'exécuter plusieurs systèmes d'exploitations sur la même machine physique

20

Système d'Exploitation

Les solutions techniques de Virtualisation

Kernel Virtual Machine (KVM)



- ☞ **KVM (Kernel Virtual Machine)** est une machine virtuelle libre pour Linux.
- ☞ Fonctionne sur les architectures x86 disposant des technologies Intel VT ou AMD SVM (AMD-V).
- ☞ Le module est intégré dans le noyau Linux depuis la version 2.6.20 et permet une virtualisation matérielle et donc une accélération de la virtualisation de système d'exploitation (hyperviseur type 1).
- ☞ C'est un système optimisé pour la virtualisation de serveur.
- ☞ KVM permet de faire fonctionner facilement des machines virtuelles aux OS variés, Windows, Linux, etc., à faibles coûts dans des environnements de production.

VMware



vmware

- ☞ **VMware** est une société informatique américaine fondée en 1998, filiale d'EMC Corporation depuis 2004, qui propose plusieurs produits propriétaires liés à la virtualisation d'architectures x86. C'est aussi par extension le nom d'une gamme de logiciels de virtualisation.
- ☞ Propose plusieurs produits propriétaires liés à la virtualisation d'architectures x86.

- ☞ **VMware Workstation**



- ☞ **VMware GSX Server**

- ☞ **VMware Server**

- ☞ **VMware ESX**

- ☞ **vCenter**



21

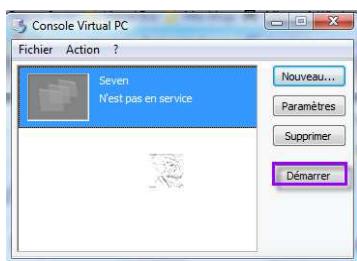
Système d'Exploitation

Les solutions techniques de Virtualisation

Microsoft Hyper-V



Virtual PC



22

Système d'Exploitation

Structure d'un Système d'Exploitation

4. L'architecture Client/serveur

- ❑ Cette tendance s'est accentuée dans les SE contemporains en tentant de réduire le SE à un noyau minimal.
- ❑ Une des formes les plus accentuées de cette évolution est **l'architecture client/serveur**.
- ❑ Le protocole ou **l'architecture client / serveur** désigne un mode de transmission d'information (souvent à travers un **réseau**) entre plusieurs **programmes** ou **processus** : l'un, qualifié de **client**, envoie des requêtes ; l'autre, qualifié de **serveur**, attend les requêtes des clients et y répond. Le serveur offre ici un **service** au client.
- ❑ La plupart des fonctionnalités d'un SE sont reportées dans des processus utilisateurs. Pour demander un service comme la lecture d'un bloc de fichier, le processus client envoie **une requête** à un processus serveur qui effectue le travail et envoie **une réponse**.

23

Système d'Exploitation

Structure d'un Système d'Exploitation

Systèmes à micro-noyaux

- Les limitations des noyaux monolithiques ont amené à une approche radicalement différente de la notion de noyau : les systèmes à micro-noyaux.
- Les systèmes à micro-noyaux cherchent à minimiser les fonctionnalités dépendantes du noyau en plaçant la plus grande partie des services du système d'exploitation à l'extérieur de ce noyau, c'est-à-dire dans **l'espace utilisateur**. Ces fonctionnalités sont alors fournies par de petits serveurs indépendants possédant souvent leur propre espace d'adressage.
- Un petit nombre de fonctions fondamentales est conservé dans un noyau minimalistique appelé « **micronoyer** ». L'ensemble des fonctionnalités habituellement proposées par les noyaux monolithiques est alors assuré par les services déplacés en **espace utilisateur** et par ce micro-noyau.
- Ce type de système permet de gagner en robustesse et en fiabilité, tout en facilitant la maintenance et l'évolutivité.

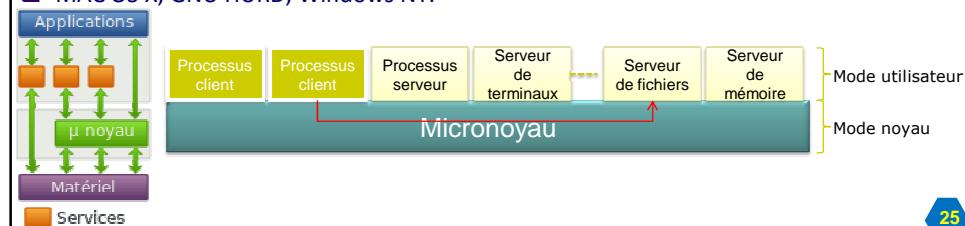
24

Système d'Exploitation

Structure d'un Système d'Exploitation

Systèmes à micro-noyaux

- ❑ Déplace plusieurs fonctions de SE vers des « **processus serveur** » s'exécutant en mode utilisateur → réduction au maximum de la taille du code privilégié (en mode noyau).
 - **But:** gérer les communications entre applications et serveurs pour:
 - Renforcer la politique de sécurité
 - Permettre l'exécution de fonctions système (accès aux registres d'E/S, etc.).
- ❑ Fiabilité augmentée : si un processus serveur « crash », le système continue à fonctionner et il est possible de relancer ce service sans redémarrer.
- ❑ Modèle facilement étendu à des systèmes distribués.
- ❑ MAC OS X, GNU HURD, Windows NT.



25

Système d'Exploitation

Structure d'un Système d'Exploitation

Systèmes à micro-noyaux

Avantages et inconvénients d'un système à micro-noyau

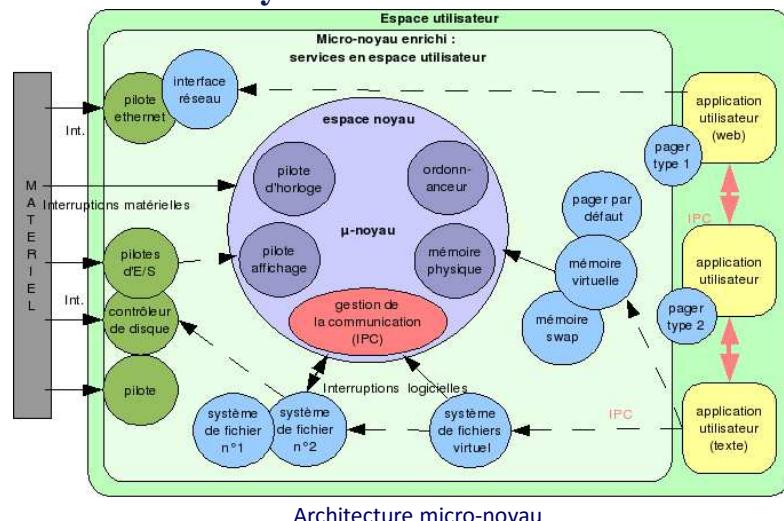
- **Protection de la mémoire et de la faible interdépendance entre les services.** Les erreurs provoquées par les applications en mode utilisateur sont traitées plus simplement que dans le mode noyau et ne mettent pas en péril la stabilité globale du système. L'intervention sur une fonctionnalité défectueuse consiste à arrêter l'ancien service puis à lancer le nouveau, sans devoir redémarrer toute la machine.
- De plus, en réduisant les possibilités pour les services de pouvoir intervenir directement sur le matériel, la sécurité du système est renforcée. Le système gagne également en possibilités de configuration. Ainsi, seuls les services utiles doivent être réellement lancés au démarrage. Les interdépendances entre les différents services sont faibles. L'ajout ou le retrait d'un service ne perturbe pas l'ensemble du système. La complexité de l'ensemble est réduite.
- La stabilité de l'ensemble est améliorée : une erreur d'un service en mode protégé a peu de conséquences sur la stabilité de l'ensemble de la machine.
- Ils sont beaucoup plus compacts que les noyaux monolithiques. **6 millions** de lignes de code pour le **noyau Linux 2.6.0** contre en général moins de **50 000 lignes** pour les **micro-noyaux**. La maintenance du code exécuté en mode noyau est donc simplifiée. Le nombre réduit de lignes de code peut augmenter la portabilité du système.

26

Système d'Exploitation

Structure d'un Système d'Exploitation

Systèmes à micro-noyaux



27

Système d'Exploitation

Structure d'un Système d'Exploitation

Systèmes à Noyaux hybrides

- **Noyaux hybrides** : désigne principalement des noyaux qui reprennent des concepts à la fois des noyaux monolithiques et des micro-noyaux, pour combiner les avantages des deux.
- Lorsque, au début des années 1990, les développeurs et concepteurs se sont aperçus des faiblesses des premiers micro-noyaux, certains réintégrèrent diverses fonctionnalités non fondamentales dans le noyau, pour gagner en performance. Les micro-noyaux « purs » semblaient condamnés à l'échec.
- Alors que la philosophie générale des systèmes à micro-noyaux est maintenue (seules les fonctions fondamentales sont dans l'espace noyau), certaines fonctions non critiques, mais très génératrices d'appels système, sont réintégrées dans l'espace noyau. Ce compromis permet d'améliorer considérablement les performances en conservant de nombreuses propriétés des systèmes à micro-noyaux. Un exemple de ce type de noyau hybride est le noyau de **Mac OS X**.

28

Système d'Exploitation

Structure d'un Système d'Exploitation

Systèmes à Noyaux temps réel

- Les noyaux temps réel sont fonctionnellement spécialisés. Ce sont des noyaux généralement assez légers qui ont pour fonction de base stricte de garantir les temps d'exécution des tâches.
- Très utilisés dans le monde de l'électronique embarquée, ils sont conçus pour tourner sur des plates-formes matérielles limitées en taille, puissance ou autonomie.
- Les noyaux temps réel peuvent adopter en théorie n'importe quelle architecture précédemment listée. Ils fournissent souvent deux interfaces séparées, l'une spécialisée dans le temps réel et l'autre générique. Les applications temps réel font alors appel à la partie temps réel du noyau.
- VxWorks est un noyau propriétaire temps réel très implanté dans l'industrie bien que les systèmes à base de noyau Linux se déplient énormément et aient un succès grandissant via RTAI et Xenomai (RTLinux étant breveté).

29

Noyau	Noyau monolithique	Noyau monolithique modulaire			Noyau hybride	Temps réel	Exemples de systèmes d'exploitation associés
		Micro-noyau	Micro-noyau enrichi	Micro-noyau hybride			
IRIX	Oui			Non	Oui	✓ Oui	IRIX
Jaluna			✓ Oui	✓ Oui	✓ Oui	✓ Oui	Jaluna/Chorus
L4		✓ Oui			✓ Oui	✓ Oui	GNU/HURD ; GNU/L4linux
Linux < 1.2	✓ Oui			✗ Non	✗ Non	✗ Non	GNU/Linux
Linux > 1.2	✓ Oui			✗ Non	✓ Oui (Patch)	✓ Oui	GNU/Linux
LinuxWorks			✓ Oui	✓ Oui	✓ Oui	✓ Oui	GNU/Linux/LinuxWorks
Unix SysV4 / SunOS 5	✓ Oui			✗ Non	✗ Non	✗ Non	Solaris 7 et suivant
VxWorks			✓ Oui	✓ Oui	✓ Oui	✓ Oui	Windows/VxWorks, BSD/VxWorks
Windows NT (Noyau de)			✓ Oui	✓ Oui	✗ Non	✗ Non	Windows NT
Xenomai			✓ Oui	✓ Oui	✓ Oui	✓ Oui	GNU/Xenomai
XNU			✓ Oui	✓ Oui	✓ Oui	✓ Oui	Mac OS X, Darwin

30

Système d'Exploitation

Structure d'un Système d'Exploitation

5. Autre classification des Systèmes d'Exploitation

- ☞ **Système d'Exploitation temps partagé**: garantir le partage équitable du temps processeur et des ressources dans le but de maximiser le temps de traitement et de réduire le temps de réponse moyen.
- ☞ **Système d'Exploitation embarqué**: SE prévus pour fonctionner sur des machines de petite taille, (PDA ou des appareils électroniques autonomes: sondes spatiales, robot, ordinateur de bord, etc.), possédant une **autonomie réduite** → gestion avancée de l'énergie + ressources limitées ...
- ☞ **Système d'Exploitation temps réel**: garantir les temps de réponse
 - Systèmes à contraintes souples/molles: systèmes acceptant des variations minimales de temps de réponse (systèmes multimédias)
 - Systèmes à contraintes dures: gestion stricte du temps pour conserver l'intégrité du système (déterminisme logique et temporel et fiabilité)

31

Système d'Exploitation

Structure d'un Système d'Exploitation

UNIX/Linux est Multi-Utilisateurs :

- ☞ **Multi-User** : Plusieurs utilisateurs sous Unix. Chacun dispose de l'ensemble des ressources du système. Comme tout système multi-utilisateur, Unix comporte des mécanismes d'identification et de protection permettant d'éviter toute interférence entre utilisateurs.

☞ Deux types des Utilisateurs :

1. **Users normaux** : compte avec
 - Login et password
 - Espace de travail protégé (rep. privé -home directory),
2. **Super-User root gère tout le système**

32

Système d'Exploitation

Structure d'un Système d'Exploitation

UNIX/Linux est Multi-tâches :

- **Multi-tâches** : Unix est multi-tâches car plusieurs programmes peuvent être en cours d'exécution en même temps sur une même machine.
- **Un processus** est une tâche en train de s'exécuter. On appelle processus, l'image de l'état du processeur et de la mémoire au cours de l'exécution du programme.
- En fait, à chaque instant, le **processeur** ne traite qu'au plus un seul des programmes lancés. La gestion des processus est effectuée par le système.³³

Système d'Exploitation

Structure d'un Système d'Exploitation

Manipulation d'un OS :Types d'IHM

- **Textuelle** : exemple : terminal Unix /Linux, MS-DOS.
- **Graphique** : exemple : Windows, KDE ou Gnome Unix /Linux.

Terminal et shell

- **Terminal** = fenêtre graphique exécutant un shell
- **Shell** = programme interactif permettant d'exécuter des commandes

À quoi sert un shell ?

- Permet de lancer rapidement des tâches
- gérer les processus
- le shell fournit toutes les fonctionnalités de base pour pouvoir lancer des commandes.

34

Système d'Exploitation

Historique de Unix/Linux

- Unix est une famille de systèmes d'exploitation, multi-tâche et multi-utilisateur, qui a été initialement développé par les Laboratoires Bell par (entre autre) **Ken Thompson** et **Dennis Ritchie** aux Etats-Unis à partir de 1969 afin d'être utilisé sur des ordinateurs gros système.

Dennis MacAlistair Ritchie (September 9, 1941 - October 12, 2011) American computer scientist.

He created the **C programming language** and, with long-time colleague **Ken Thompson**, the **Unix operating system** and **B programming language**.



Ken Thompson (assis) et Dennis Ritchie (debout) en 1970

35

Système d'Exploitation

- 1969 : premier système Unix écrit par Ken Thompson aux Bell Labs d'AT&T, en assembleur sur PDP-7, puis en langage "B". A la base appelé Unics (jeu de mot formé à partir de Multics, gros système développé fin 60 par le MIT et General Electric)
- 1973 : Dennis Ritchie, qui a inventé le langage "C", récrit Unix en C avec Thompson (vers.4). Rend possible et entraîne le partage d'Unix sur de nombreuses machines (sources d'Unix distribués à de nombreuses universités et sociétés commerciales). Une 3^e personne, Brian Kernighan, contribue également fortement aux premiers développements d'Unix.
- Les versions 1 à 3 du système sont développées en langage assembleur, puis en C à partir de la version 4.
- En 1975, la première licence du système est vendue au département d'informatique de l'Université de l'Illinois. À partir de là, l'influence d'Unix dans le monde académique se développe. L'université de Berkeley en Californie développe une nouvelle version dérivée appelée BSD (Berkeley Software Distribution).
- 1976 : publication de la version 6 du Manuel Unix ("manuel du programmeur").
- 1977 : divergence en 2 grandes familles Unix :
 1. AT&T : System III, puis **System V** (1990 : release 4, abrégé SVR4)
 2. Uni. de Californie à Berkeley : **BSD** (Berkeley Software Distribution, act. vers.4.4), originalités : propre système de fichier, C-shell, gestion virtuelle mémoire, job contrôle, liens symboliques, TCP/IP...

36

Système d'Exploitation

- ☞ **1980 : Bjarne Stroustrup (Bell Labs) définit le langage C++** (extension "objet" du C)
- ☞ Dans les années 80, AT&T autorise le clonage d'UNIX par d'autres constructeurs. De nouveaux systèmes dérivés d'Unix voient le jour comme AIX sur IBM.
- ☞ Les deux branches, System V et BSD, sont considérées comme les deux branches majeures d'Unix.
- ☞ **1987 : apport du MIT en matière de graphique et fenêtrage : Xwindow version 11** (act. release 6, abrégé X11R6).
- ☞ Dans les années 80 et au début des années 90, formation de deux Consortiums concurrents:
 - ☞ HP et IBM choisissent **System V**.
 - ☞ Sun et DEC étendent plutôt la branche **BSD**.
- ☞ AT&T et Sun finiront par collaborer pour fusionner **System V** et SunOS pour donner naissance à **Solaris** (sortie de Solaris, l'Unix de Sun dérivé du System V en 1992).
- ☞ La branche **BSD** donnera naissance à un système que vous connaissez : **Mac OS**.
- ☞ 1990 : coopération AT&T et Sun Microsystems : version V.4. Diffusion du langage Perl.

Remarque: Systèmes Unix like

Quand AT&T autorise le clonage de son système, on l'a vu de nombreuses versions apparaissent. L'incompatibilité entre ces nouveaux systèmes dérivés d'Unix, souvent propriétaires, conduit à développer des standards, dont POSIX et la spécification Unix. Dans les années 80 et 90, de nouveaux systèmes sont développés. Basés sur Unix, libres, ils sont destinés à être similaires à un système Unix sans être nécessairement certifiés. On les appelle des systèmes Unix-like.

37

Système d'Exploitation

Origines du Logiciel Libre (GNU's Not Unix):

- ☞ **Richard Stallman** (chercheur en informatique du MIT quitte son poste et se consacre à l'écriture d'un système d'exploitation Libre du nom de GNU ...).
- ☞ énonce clairement le concept de logiciel libre (« free ») pour développer des logiciels libres et ouverts, **mais il lui manque un noyau libre.**
- ☞ Démarre le projet **GNU (1984)**. **But : re-créer un système d'exploitation complet (Unix-like), composé uniquement de logiciels libres.**
- ☞ Crée la FSF (Free Software Fundation, 1985) pour gérer le projet GNU.
- ☞ **Remarque :** « Free » dans la culture signifie « libre », pas nécessairement « gratuit » ou « non commercial»



Système d'Exploitation

Linux, juste le noyau !

- Le projet GNU arrive en 1991 avec de très nombreux outils libres, mais il lui manque un élément central : **le noyau. Cet élément est essentiel car il gère la mémoire, le microprocesseur**, les périphériques comme le clavier, la souris, les disques durs. .
- En 1991, un étudiant finlandais, **Linus** Torvalds, décide de créer un clone libre d'UNIX. Ce clone d'UNIX va s'appeler **Linux** (Linus+UNIX).



Linus Torvalds

39

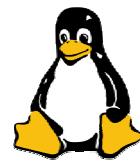
Système d'Exploitation

- «Au sens strict, Linux est le nom du noyau de système d'exploitation libre, multitâche, multiplate-forme et multi-utilisateur de type UNIX. Linus Torvalds, commence à développer un noyau et demande aux personnes intéressées d'y contribuer. La licence GPL a été publiée à la même époque et Linus Torvalds s'est laissé persuader de placer son noyau sous cette dernière.
- Le système d'exploitation actuellement connu est donc un assemblage des outils GNU fonctionnant sur un noyau Linux, on parle donc de **GNU/Linux** avec le slash, « / » pour « **GNU** sur **Linux** ».
- **GNU/Linux** est un système d'exploitation *unix-like* : il possède les même fonctionnalités qu'Unix (selon la norme POSIX), mais implémentées différemment.
- Il s'agit d'un **OS libre** : l'utilisation, l'étude et la modification sont autorisées techniquement et légalement.

40

Système d'Exploitation

- Il existe beaucoup (beaucoup !) de systèmes basés sur GNU+Linux.
On les appelle des distributions.
- Distribution = **noyau Linux + utilitaires GNU**
- On peut citer notamment les distributions mères dont sont issues toutes les autres :
 - ArchLinux
 - **Debian**
 - Gentoo
 - Red Hat
 - Et parmi les distributions grand public, on retrouve aussi **Ubuntu**, basée sur **Debian**, ou Fedora, basée sur Red Hat.
- L'adoption de Linux décolle ensuite dans les années 90 et+, et il s'impose sur les serveurs et les systèmes embarqués. En usage domestique, Linux est notamment utilisé sur les netbooks avec Chrome OS. Son succès le plus important chez les particuliers est sur le marché des smartphones avec **Android**.



41

Système d'Exploitation

Unix propriétaire/libre

Les principaux Unix propriétaires

 IBM	AIX®
 HP	HP-UX®
 SCO	Tru64 UNIX® UnixWare®
 SGI	IRIX®
 SUN	SOLARIS®

Les principaux Unix Libres

OpenBSD	
FreeBSD	
NetBSD	
Mac-OS X	
les GNU/Linux	



THE *Open* GROUP
Making standards work™
<http://www.unix.org>

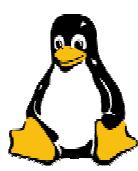
The Open Group

- Possède la marque déposée **UNIX®**
- Publie la norme « Single UNIX Specification »
(intègre les normes précédentes : X/Open Company's XPG4, IEEE's POSIX Standards et ISO C)

42

Système d'Exploitation

Systèmes Unix/Linux : Synthèse



BSD (1977)

MULTICS (1965)

UNICS (1969)

Systèmes Unix
v1 à v7
(1972 - 1979)

System V (1983)

FreeBSD
OpenBSD
NetBSD
NextStep
MacOS

Projet
GNU
(1983)
Linux
(1991)

UnixWare
HP-UX
AIX
IRIX
Solaris

43

Système d'Exploitation



Richard
Stallman



GNU
Utilitaires



Linus
Torvalds



Linux
Noyau

Distributions GNU / Linux (SE)



redhat



debian



openSUSE



Mandriva

44

Plan : Système d'exploitation

INTRODUCTION AUX SYSTEMES D'EXPLOITATION

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

2- Structure d'un système d'exploitation

a- Systèmes monolithiques

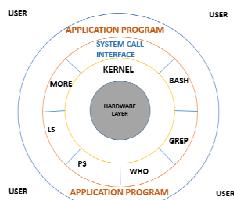
b- Systèmes à couches

c- machines virtuelles

d- modèle Client / Serveur



3- Structure du système d'exploitation Unix



45

Système d'Exploitation

3- Structure du système d'exploitation Unix

- Le système d'exploitation Unix / Linux a fait l'objet d'une normalisation appelée POSIX (Portable Operating System Interface) par l'IEEE (IEEE 1003). Le système Linux (ou GNU/Linux) constitue une implantation de ce standard Unix.
- POSIX spécifie, dans dix-sept documents différents, les interfaces utilisateurs et les interfaces logicielles, la ligne de commande standard et l'interface de script qu'est le Bourne shell. Les autres commandes, services et utilitaires, ... Les services d'entrées/sorties de base (fichiers, terminaux, réseau) doivent être présents ; le système doit supporter certains attributs spécifiques pour les fichiers. POSIX définit aussi une interface de programmation standard, et celle-ci est prise en charge par la plupart des systèmes d'exploitation récents.
- Linux peut être déployé aujourd'hui sur de multiples configurations matérielles allant de processeurs embarqués aux super-calculateurs, et aussi les ordinateurs personnels.
- Plusieurs diffuseurs de Linux se partagent le marché. On dispose ainsi de distributions RedHat, Debian, Ubuntu, etc. Ces versions diffèrent essentiellement par :
 - leur facilité d'installation et de **mises à jour du noyau** ;
 - les nombreux **paquetages** disponibles ;
 - **l'interface graphique** usager, GNOME ou KDE, et
 - la richesse des pilotes de périphériques développés.

46

Système d'Exploitation

3- Structure du système d'exploitation Unix

La conception générale du système Unix/Linux

➤ Le système Unix/Linux s'appuie sur deux **concepts fondamentaux**:

1. pour la gestion des traitements (les exécutions de programmes), le concept de **processus** :
 - Ce concept encapsule l'exécution d'un programme de façon à assurer l'allocation et le contrôle des ressources nécessaires à l'exécution du programme.
 - Il permet d'optimiser l'utilisation des ressources disponibles (processeur(s), mémoires, périphériques, fichiers, pipes, sockets...) dans la mesure où plusieurs processus peuvent exister en parallèle ;
2. pour la gestion des données, le concept de **fichiers** : ce concept permet d'assurer une rémanence et un contrôle des données des usagers du système.

47

Système d'Exploitation

3- Structure du système d'exploitation Unix

La conception générale du système Unix/Linux

➤ Unix/Linux gère un nombre dynamique d'objets processus et/ou

fichiers et maintient une structure arborescente entre ces objets. En effet, **tout processus est un descendant d'un processus unique initial racine** et **tout fichier est placé dans une arborescence de répertoires ayant une racine unique**.

➤ Processus et fichiers sont aussi réunis par la notion d'usager qui sert de base aux mécanismes de protection des ressources. Tout processus s'exécute pour un usager appartenant à un groupe. Tout fichier possède de la même manière un créateur identifié. L'identification d'un usager définit ainsi une capacité d'accès de cet usager aux ressources du système via les processus qu'il engendrera.

48

Système d'Exploitation

3- Structure du système d'exploitation Unix

L'architecture du système Unix est une architecture en couche, sa conception est modulaire et basé sur trois couches bien définie .

❑ **Le noyau** : Partie centrale d'UNIX: C'est le cœur du système GNU/Linux.

- Il gère les ressources matérielles du système: assure la gestion de la mémoire, du partage du processeur entre les différentes tâches.
- Résident en mémoire et se charge au démarrage (boot).
- Les autres composants logiciels doivent passer par lui pour accéder au matériel.

❑ **L'interface Utilisateur (Shell)** : C'est un utilitaire qui interprète les commandes de l'utilisateur et assure leur exécution.

- Couche logicielle bien séparée du noyau.
- Possibilité d'écrire ses propres fonctions au moyen de programme en langage Shell (Shellscripts) / utilisables par la suite comme n'importe quelle commande. Principaux shells : Bourne shell, C shell, Korn shell et Bourne-Again shell (bash).

❑ **Programmes utilitaires (Applications)** :

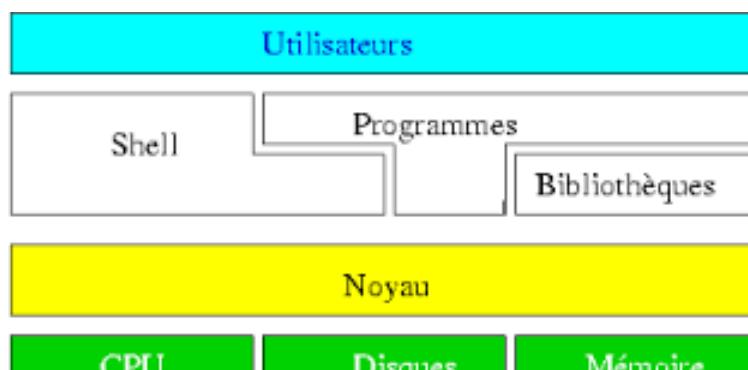
- Ensemble des outils disponibles (programmes, bibliothèque, les éditeurs de texte, les commandes, ...). Les applications sont des programmes utilisateur incluant mais non limités à : Navigateurs Internet; Traitement de texte; Tableurs ...

❑ **Matériel** : La couche inférieure, appelée couche d'abstraction matérielle HAL (*Hardware Abstraction Layer*), est chargée de masquer les particularités matérielles.

49

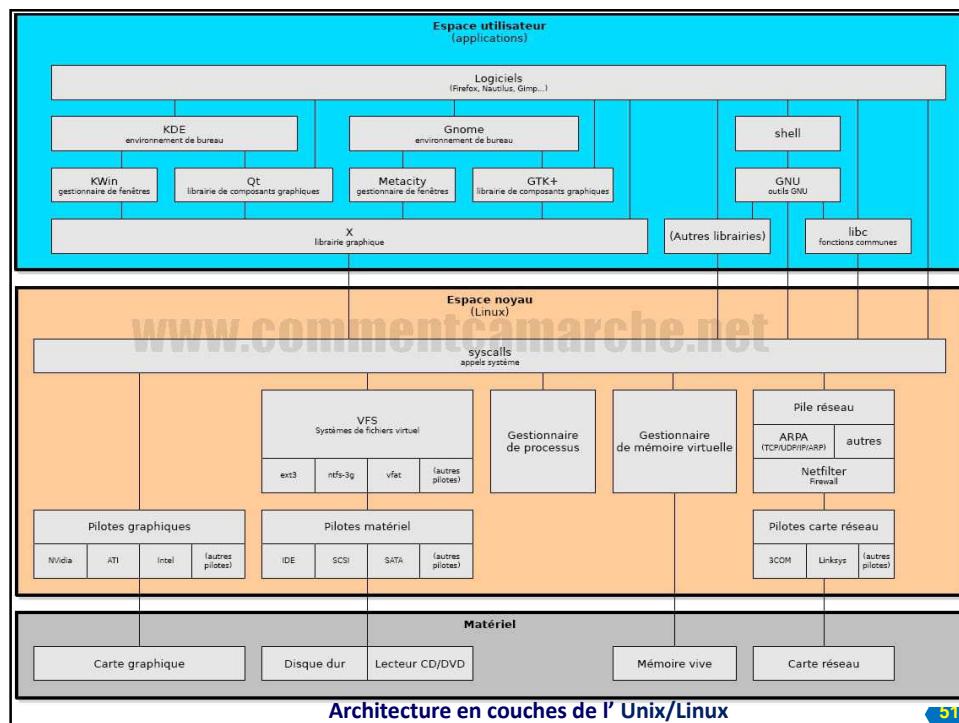
Système d'Exploitation

3- Structure du système d'exploitation Unix



Architecture en couches de l' Unix/Linux

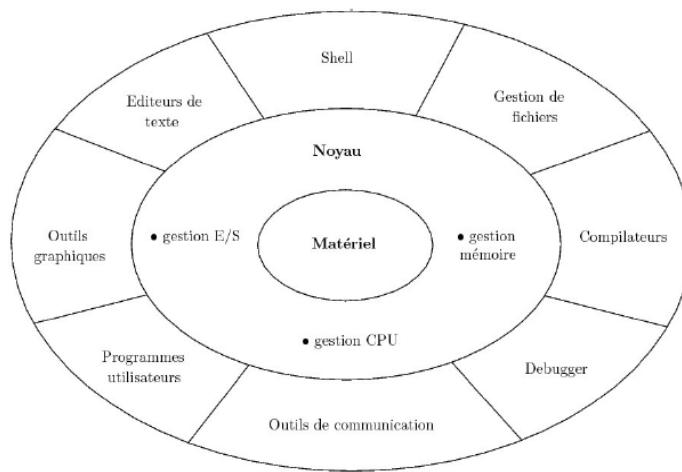
50



Système d'Exploitation

3- Structure du système d'exploitation Unix

L'architecture du système **Unix/Linux** est une architecture en couche, sa conception est modulaire et basé sur couches bien définie.



Système d'Exploitation

Architecture du système Unix/Linux : Noyau

- Le Noyau est le programme qui assure la gestion de la mémoire, le partage du processeur entre les différentes tâches à exécuter et les E/S de bas niveau.
- Il est lancé au démarrage du système (le boot) et s'exécute jusqu'à son arrêt.
- C'est un petit programme, qui est chargé en mémoire principale.
- Le Rôle principal du Noyau est d'assurer une bonne répartition des ressources de l'ordinateur (mémoire, processeur(s), espace disque, imprimante(s), accès réseaux), sans intervention des utilisateurs!**

Il s'exécute en mode superviseur (ou super-utilisateur ou root).

- Tous les autres programmes, qui s'exécutent sur l'ordinateur, fonctionnent en mode utilisateur.
- Linux est architecturé autour d'un noyau (en anglais kernel) chargé de prendre en charge le matériel.

Remarque : distribution Unix/Linux

l'assemblage d'un ensemble de logiciels autour d'un noyau Linux afin de fournir un système clé en main. Les distributions les plus connues sont : Debian, RedHat, Slackware, Ubuntu, Fedora... ;

53

Plan : Système d'Exploitation

1. INTRODUCTION AUX SYSTEMES D'EXPLOITATION

2. LES PROCESSUS

3. LES ENTREES/SORTIES

4. LA GESTION DE LA MÉMOIRE

5. LES SYSTEMES DE FICHIERS



54

Système d'Exploitation

LES PROCESSUS

- 1- Le concept de processus
- 2- La communication inter-processus
- 3- Le problème des philosophes
- 4- Ordonnancement des processus
- 5- Les processus de l'UNIX



55

Système d'Exploitation

Le concept de processus

- Un **processus** (en anglais, process), est un programme en cours d'exécution et il peut être défini comme :
 - Un ensemble d'**instructions** à exécuter, pouvant être dans la **mémoire morte**, mais le plus souvent chargé depuis la **mémoire de masse** vers la **mémoire vive** ;
 - un **espace d'adressage** en mémoire vive pour stocker la pile, les données de travail, etc. ;
 - des **ressources** permettant des entrées-sorties de données, comme des ports réseau.
- Un processus peut être démarré par un utilisateur par l'intermédiaire d'un périphérique ou bien par un autre processus : les « applications » utilisateurs sont des ensembles de processus.
- **Problème:** Un processeur ne peut exécuter qu'une seule instruction à la fois.
- **BUT:** Partager un (ou plusieurs) processeur entre différents processus.
- **Attention!!!** Ne pas confondre **processus** et **processeur**

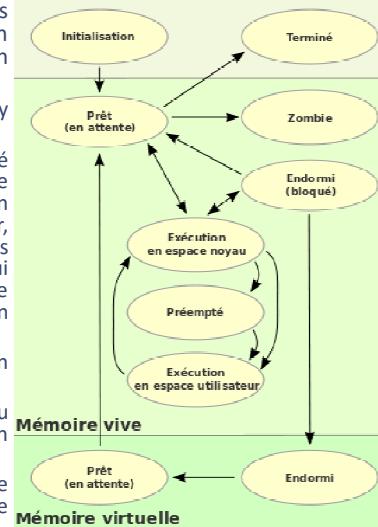
Processus et Programme

56

Système d'Exploitation

Le concept de processus

- Un ordinateur équipé d'un système d'exploitation multitâches est capable d'exécuter plusieurs processus de façon quasi simultanée.
- États d'un processus (existent dans la plupart des systèmes d'exploitation) : Les états successifs d'un processus sont généralement représentées par un diagramme d'état.
 - 1.**Initialisation** : C'est le premier état d'un processus. Il y attend que l'**ordonnanceur** le place dans l'état prêt.
 - 2.**Prêt** ou En attente : Dans cet état, le processus a été chargé en **mémoire centrale** et attend son exécution sur le **processeur**. Il peut y avoir beaucoup de processus en attente car, sur un ordinateur équipé d'un seul processeur, les processus doivent passer un par un. Les processus disponibles sont rangés dans une **file** ; les autres, ceux qui attendent quelque chose (données provenant du disque dur, une connexion internet, etc.) ne sont pas pris en compte. Cette file d'attente est gérée par l'**ordonnanceur**.
 - 3.**Élu** ou **Exécution** : Le processus est en cours d'exécution par le processeur.
 - 4.**Endormi** ou Bloqué : Le processus a été **interrompu** ou attend un événement (la fin d'une opération d'**entrée/sortie**, un **signal**, ...).
 - 5.**Terminé** : Le processus est terminé, c'est-à-dire soit le résultat est connu, soit le programme a été forcé de s'arrêter.



Système d'Exploitation

Le concept de processus

- **États particuliers** : Selon les systèmes d'exploitation, ces différents états peuvent aussi être possibles :
 1. **Zombie** : Si un processus terminé ne peut pas être déchargé de la mémoire, par exemple parce que son processus parent n'a pas récupéré son signal de terminaison, il passe dans un état appelé zombie.
 2. **Swappé** : Lorsqu'un processus est transféré de la mémoire centrale dans la **mémoire virtuelle**, il est dit « swappé ». Un processus swappé peut être dans un état endormi ou prêt.
 3. **Préempté** : L'ordonnanceur a décidé de suspendre l'activité d'un processus. Par exemple, un processus qui consomme trop de temps CPU finira par être préempté. Un ordonnanceur préemptif utilise aussi l'indice de priorité pour décider le processus qui sera préempté.
- **Exécution en espace utilisateur** : L'exécution a lieu dans un espace limité : seules certaines instructions sont disponibles.
- **Exécution en espace noyau** : Par opposition au mode utilisateur, l'exécution du processus n'est pas limitée. Par exemple, un processus dans cet état peut aller lire dans la mémoire d'un autre.

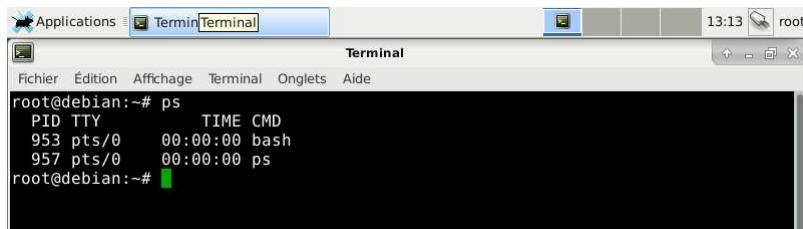
Système d'Exploitation

Le concept de processus

Caractéristiques d'un processus : parmi celles-ci, on peut citer : son identification (PID), les identifications de son processus père (PPID), de son utilisateur propriétaire (UID) et de son groupe propriétaire (GID), éventuellement son terminal d'attachement (TTY), etc.

- Linux/UNIX: Lister des informations sur les Processus

Commande ps



```
root@debian:~# ps
  PID TTY      TIME CMD
 953 pts/0    00:00:00 bash
 957 pts/0    00:00:00 ps
root@debian:~#
```

59

Système d'Exploitation

Le concept de processus

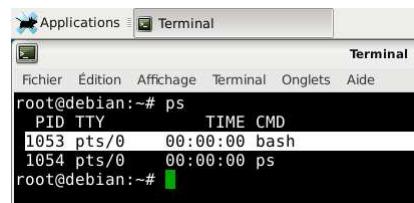
Un **processus linux** est un programme en action qui s'exécute en mémoire et dans le processeur de votre ordinateur.

Il y a deux catégories de processus:

1. Le processus utilisateur (**process user**) et,
2. le processus système généralement appelé processus **démon** (**daemon process**).

Les processus utilisateurs ce sont des processus qui sont créés par un utilisateur particulier.

Par exemple le premier processus lancé par un utilisateur en se connectant à un terminal est le processus **bash**.



```
root@debian:~# ps
  PID TTY      TIME CMD
1053 pts/0    00:00:00 bash
1054 pts/0    00:00:00 ps
root@debian:~#
```

60

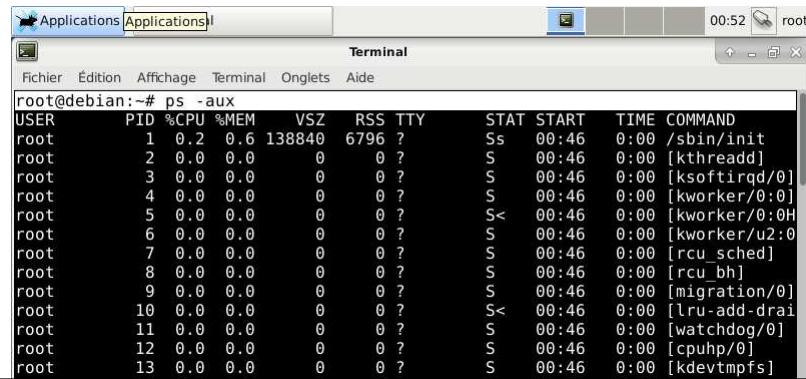
Système d'Exploitation

Le concept de processus : processus démon

Les processus systèmes (**daemon process**) ce sont des processus qui sont créés lors du lancement du système ou à une date fixée par un sysadmin.

On peut prendre le processus **init** comme exemple, c'est le premier processus lancé par le noyau linux lors du lancement.

- La commande **ps -aux** Afficher la liste de tous les processus en cours d'exécution (Processus utilisateur et démon)



A terminal window titled "Terminal" showing the output of the command "ps -aux". The output lists various processes with their PID, TTY, CPU usage, memory usage, and command names. The "COMMAND" column shows system daemons like /sbin/init, [kthreadd], [ksoftirqd/0], [kworker/0:0H], [rcu_sched], [rcu_bh], [migration/0], [lru-add-drain], [watchdog/0], [cpuhp/0], and [kdevtmpfs].

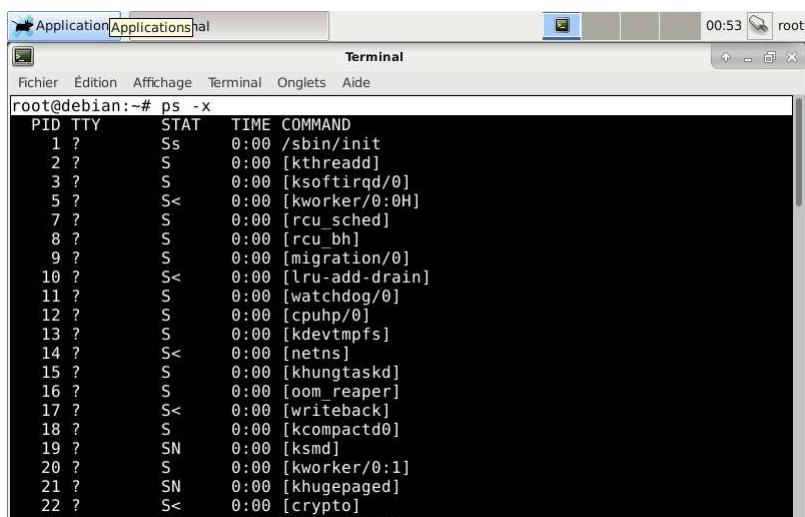
USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.2	0.6	138840	6796	?	Ss	00:46	0:00	/sbin/init
root	2	0.0	0.0	0	0	?	S	00:46	0:00	[kthreadd]
root	3	0.0	0.0	0	0	?	S	00:46	0:00	[ksoftirqd/0]
root	4	0.0	0.0	0	0	?	S	00:46	0:00	[kworker/0:0]
root	5	0.0	0.0	0	0	?	S<	00:46	0:00	[kworker/0:0H]
root	6	0.0	0.0	0	0	?	S	00:46	0:00	[kworker/u2:0]
root	7	0.0	0.0	0	0	?	S	00:46	0:00	[rcu_sched]
root	8	0.0	0.0	0	0	?	S	00:46	0:00	[rcu_bh]
root	9	0.0	0.0	0	0	?	S	00:46	0:00	[migration/0]
root	10	0.0	0.0	0	0	?	S<	00:46	0:00	[lru-add-drain]
root	11	0.0	0.0	0	0	?	S	00:46	0:00	[watchdog/0]
root	12	0.0	0.0	0	0	?	S	00:46	0:00	[cpuhp/0]
root	13	0.0	0.0	0	0	?	S	00:46	0:00	[kdevtmpfs]

61

Système d'Exploitation

Le concept de processus : processus démon

La commande **ps -x** Afficher la liste des processus systèmes (démon)



A terminal window titled "Terminal" showing the output of the command "ps -x". The output lists various processes with their PID, TTY, CPU usage, memory usage, and command names. The "COMMAND" column shows system daemons like /sbin/init, [kthreadd], [ksoftirqd/0], [kworker/0:0H], [rcu_sched], [rcu_bh], [migration/0], [lru-add-drain], [watchdog/0], [cpuhp/0], [kdevtmpfs], [netns], [khungtaskd], [oom_reaper], [writeback], [kcompactd0], [ksmd], [kworker/0:1], [khugepaged], and [crypto].

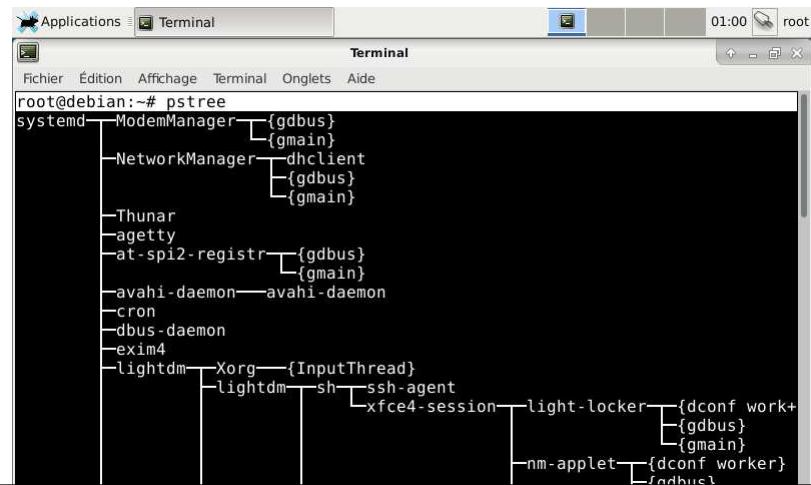
PID	TTY	STAT	TIME	COMMAND
1	?	Ss	0:00	/sbin/init
2	?	S	0:00	[kthreadd]
3	?	S	0:00	[ksoftirqd/0]
5	?	S<	0:00	[kworker/0:0H]
7	?	S	0:00	[rcu_sched]
8	?	S	0:00	[rcu_bh]
9	?	S	0:00	[migration/0]
10	?	S<	0:00	[lru-add-drain]
11	?	S	0:00	[watchdog/0]
12	?	S	0:00	[cpuhp/0]
13	?	S	0:00	[kdevtmpfs]
14	?	S<	0:00	[netns]
15	?	S	0:00	[khungtaskd]
16	?	S	0:00	[oom_reaper]
17	?	S<	0:00	[writeback]
18	?	S	0:00	[kcompactd0]
19	?	SN	0:00	[ksmd]
20	?	S	0:00	[kworker/0:1]
21	?	SN	0:00	[khugepaged]
22	?	S<	0:00	[crypto]

62

Système d'Exploitation

Le concept de processus : processus démon

La commande **pstree** Afficher les processus sous forme d'arborescence tout en affichant leur lien de parenté.



```
root@debian:~# pstree
systemd—ModemManager—{gdbus}
                         |—{gmain}
                         |
                         NetworkManager—dhclient—{gdbus}
                                         |—{gmain}
                                         |
                                         Thunar
                                         agetty
                                         at-spi2-registr—{gdbus}
                                         |—{gmain}
                                         avahi-daemon—avahi-daemon
                                         cron
                                         dbus-daemon
                                         exim4
                                         lightdm—Xorg—{InputThread}
                                         |—lightdm—sh—ssh-agent—xfce4-session—light-locker—{dconf work+}
                                         |                                         |—{gdbus}
                                         |                                         |—{gmain}
                                         nm-applet—{dconf worker}
                                         |—{gdbus}
```

63

Système d'Exploitation

Le concept de processus : processus démon dans Windows

Dans Windows, les démons sont appelés **services**.



Gestionnaire des tâches	Fichier	Options	Affichage	Processus	Performance	Historique des applications	Démarrage	Utilisateurs	Détails	Services
Nom	PID	Description	Statut							Groupe
HomeGroupListener		Écouteur du Groupement résidentiel	Arrêté							LocalSystemNetworkRestricted
hidserv		Service du périphérique d'interface utilisateur	Arrêté							LocalSystemNetworkRestricted
fhsvc		Service d'historique des fichiers	Arrêté							LocalSystemNetworkRestricted
embeddedmode		embeddedmode	Arrêté							LocalSystemNetworkRestricted
DsSvc	908	Service de partage des données	En cours d'exécution							LocalSystemNetworkRestricted
dot3svc		Configuration automatique de réseau câblé	Arrêté							LocalSystemNetworkRestricted
DevQueryBroker		Service Broker de découverte en arrière-plan DevQuery	Arrêté							LocalSystemNetworkRestricted
DeviceAssociationService	908	Service d'association de périphérique	En cours d'exécution							LocalSystemNetworkRestricted
CscService		Fichiers hors connexion	Arrêté							LocalSystemNetworkRestricted
AudioEndpointBuilder	908	Générateur de points de terminaison du service Audio ...	En cours d'exécution							LocalSystemNetworkRestricted
XboxNetApiSvc		Service de mise en réseau Xbox Live	Arrêté							netsvcs
XblGameSave		Jeu sauvegardé sur Xbox Live	Arrêté							netsvcs
XblAuthManager		Gestionnaire d'authentification Xbox Live	Arrêté							netsvcs
wuauserv		Windows Update	Arrêté							netsvcs
wlidsvc		Assistant Connexion avec un compte Microsoft	Arrêté							netsvcs
Winnmgmt	612	Infrastructure de gestion Windows	En cours d'exécution							netsvcs
werclplsupport		Prise en charge de l'application Rapports et solutions a...	Arrêté							netsvcs
UsoSvc		Mettre à jour le service Orchestrator	Arrêté							netsvcs
UserManager	612	Gestionnaire des utilisateurs	En cours d'exécution							netsvcs
Themes	612	Thèmes	En cours d'exécution							netsvcs
ShellHWDetection	612	Détection matériel noyau	En cours d'exécution							netsvcs
SharedAccess		Partage de connexion Internet (ICS)	Arrêté							netsvcs

Système d'Exploitation

LES PROCESSUS

- 1- Le concept de processus
- 2- **La communication inter-processus**
- 3- Le problème des philosophes
- 4- Ordonnancement des processus
- 5- Les processus de l'UNIX



65

Système d'Exploitation

La communication inter-processus

Inter-Process Communication (IPC)

- ❑ Les systèmes d'exploitation actuels offrent la possibilité de créer **plusieurs processus** (threads) concurrents qui coopèrent pour réaliser des **traitements complexes** d'une **même** application.
- ❑ Les processus ne sont pas des entités indépendantes. Ils doivent partager les ressources de l'ordinateur. Dans certains cas, ils doivent communiquer entre eux pour se synchroniser ou pour communiquer de l'information.
 - Les processus s'exécutent sur un même ordinateur (monoprocesseur) ou multiprocesseur ou sur différents ordinateurs, et ils peuvent **échanger** des informations.
- ❑ La communication interprocessus (**IPC**) consiste à transférer (**échanger**) des données entre les processus. Par exemple, un navigateur Internet peut demander une page à un serveur, qui envoie alors les données HTML.

66

Système d'Exploitation

La communication inter-processus

Inter-Process Communication (IPC)

La communication interprocessus peut se faire de différentes manières : **mémoire partagée**, **mémoire mappée**, **tubes**, **files** et **socket**.

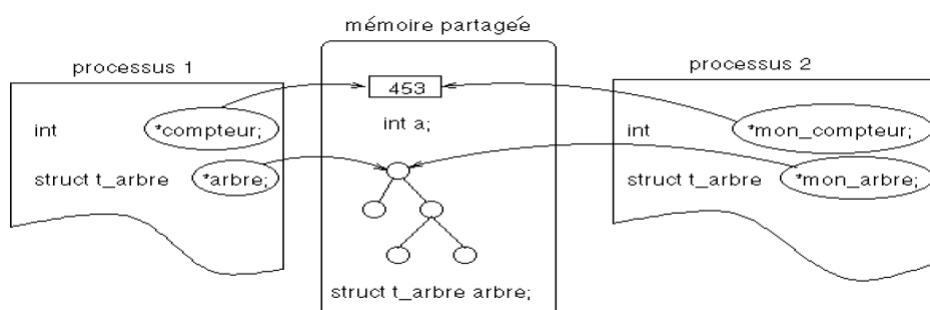
- La **mémoire partagée** permet aux processus de communiquer simplement en lisant ou écrivant dans un emplacement mémoire prédéfini.
- La **mémoire mappée** est similaire à la mémoire partagée, excepté qu'elle est associée à un fichier.
- Les **tubes** permettent une communication séquentielle d'un processus à l'autre.
- Les **files** FIFO sont similaires aux tubes excepté que des processus sans lien peuvent communiquer car le tube reçoit un nom dans le système de fichiers.
- Les **signaux**.
- Les **sockets** permettent la communication entre des processus sans lien, pouvant se trouver sur des machines distinctes.

67

Système d'Exploitation

La communication inter-processus: Mémoire Partagée

- La **mémoire partagée** permet à deux processus ou plus d'accéder à la même zone mémoire comme s'ils avaient leurs pointeurs dirigés vers le même espace mémoire. Lorsqu'un processus modifie la mémoire, tous les autres processus voient la modification.



68

Système d'Exploitation

La communication inter-processus: Mémoire Partagée

- La **mémoire partagée** est la forme de communication interprocessus la plus rapide car tous les processus partagent la même mémoire. Elle évite également les copies de données inutiles.
- Pour utiliser un segment de **mémoire partagée**, un processus doit allouer le segment. Puis, chaque processus désirant accéder au segment doit l'attacher. Après avoir fini d'utiliser le segment, chaque processus le détache. À un moment ou à un autre, un processus doit libérer le segment.
- Sous Unix, un processus alloue un segment de mémoire partagée en utilisant `shmget` (« Shared Memory GET », obtention de mémoire partagée):
 - Son premier paramètre est une clé entière qui indique le segment à créer.
 - Le second paramètre indique le nombre d'octets du segment.
 - Le troisième paramètre est un ensemble d'indicateurs binaires décrivant les options demandées : s'agit-il d'un nouveau segment ou un segment existant qu'il faut attacher, permissions de lecture/écriture,
 - ... etc

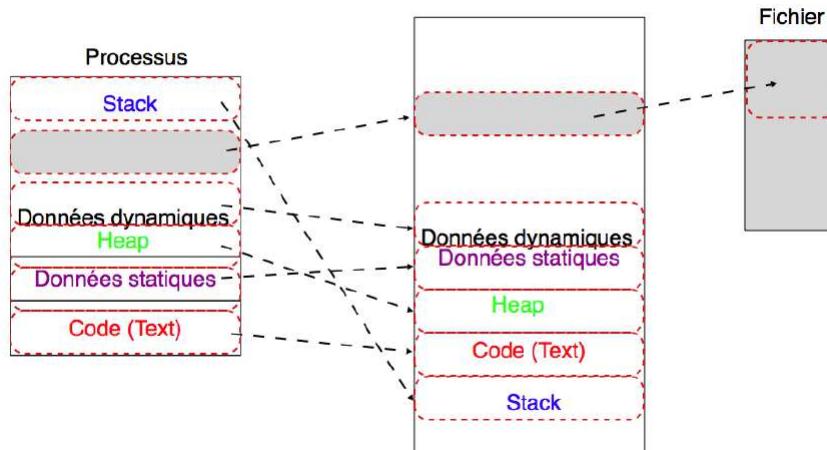
69

Système d'Exploitation

La communication inter-processus: Mémoire Mappée

- La **mémoire mappée** permet à différents processus de communiquer via un **fichier partagé**.

Mémoire RAM

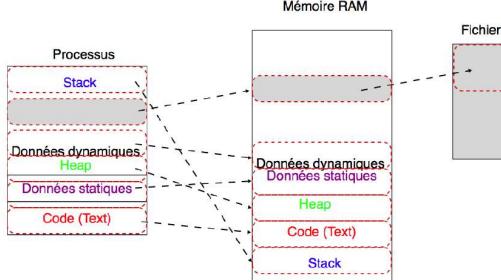


70

Système d'Exploitation

La communication inter-processus: Mémoire Mappée

- ❑ Lorsqu'un processus Unix veut **lire** ou **écrire** des **données** dans un **fichier**, il utilise en général les appels systèmes open, read, write et close directement. Ce n'est pas la seule façon pour accéder à des données sur un dispositif de stockage.
- ❑ Grâce à la mémoire virtuelle, il est possible de placer le contenu d'un fichier ou d'une partie de fichier dans une zone de la mémoire du processus. Cette opération peut être effectuée en utilisant l'appel système **mmap**. Cet appel système permet de rendre des pages d'un fichier accessibles à travers la table des pages du processus comme illustré dans la figure ci-dessous.

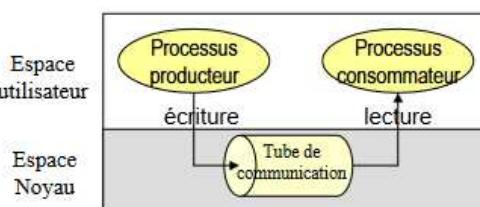


71

Système d'Exploitation

La communication inter-processus: Tubes

- ❑ **Tube** de communication: canal (en anglais pipe) dans lequel un processus peut écrire des données (producteur) et un autre processus peut les lire (consommateur).
- ❑ C'est un moyen de communication **unidirectionnel** inter-processus. C'est le moyen de communication le plus simple entre deux processus.
- ❑ Un des moyen de communication les plus utilisé sous Unix est le tube.
- ❑ Pour avoir une communication bidirectionnelle entre deux processus, il faut créer deux tubes et les employer dans des sens opposés.
- ❑ Un tube est un tampon qui sert de moyen de communication entre un processus producteur et un processus consommateur
 - Le tampon a une capacité finie
 - Il est **unidirectionnel** (sous Linux)
- ❑ Les données sont transmises dans l'ordre FIFO (First In First Out)
 - Le consommateur lit les données dans l'ordre dans lequel elles ont été produites



72

Système d'Exploitation

La communication inter-processus: Tubes

Exemple 1 :

- La commande shell suivante crée deux processus qui s'exécutent en parallèle et qui sont reliés par un tube de communication pipe. Elle détermine le nombre d'utilisateurs connectés au système en appelant who, puis en comptant les lignes avec wc : **who | wc -l**
- Le premier processus réalise la commande who. Le second processus exécute la commande wc -l. Les résultats récupérés sur la sortie standard du premier processus sont dirigés vers l'entrée standard du deuxième processus via le tube de communication qui les relie.
- Le processus réalisant la commande who ajoute dans le tube une ligne d'information par utilisateur du système. Le processus réalisant la commande wc -l récupère ces lignes d'information pour en calculer le nombre total. Le résultat est affiché à l'écran. Les deux processus s'exécutent en parallèle, les sorties du premier processus sont stockées sur le tube de communication.

```
root@debian:~# who | wc -l
1
root@debian:~#
```

73

Système d'Exploitation

La communication inter-processus: Tubes

Exemple 2 : |

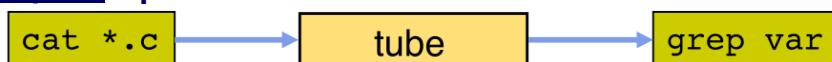
La commande : **cat pipe | grep wc**

Permet de créer et communiquer deux processus :

1. le premier cat pipe pour ouvrir le fichier pipe et ;
2. le 2^{eme} pour chercher la chaîne wc.

```
root@debian:~# cd Bureau
root@debian:~/Bureau# cat pipe |grep wc
ls | wc -l
ls | wc -l
ls | wc -l
ls | wc -l
root@debian:~/Bureau#
```

Exemple 3 : |



`cat *.c | grep var`

a) crée un tube et deux processus : p1 qui exécute cat *.c, p2 qui exécute grep var

b) connecte la sortie de p1 à l'entrée du tube et l'entrée de p2 à la sortie du tube

Système d'Exploitation

La communication inter-processus: Tubes

Les Caractéristiques

- ❑ Un tube possède deux extrémités :
 1. une est utilisée pour la lecture ;
 2. l'autre pour l'écriture.
- ❑ La gestion des tubes se fait en mode FIFO : Les données sont lues dans l'ordre dans lequel elles ont été écrites dans le tube.
- ❑ La lecture dans un tube est destructrice : Une fois une donnée est lue elle sera supprimée.
 - Plusieurs processus peuvent lire dans le même tube mais ils ne liront pas les mêmes données, car la lecture est destructive.
- ❑ Un tube a une capacité finie.
 - Il y a une synchronisation type producteur/consommateur.
 - si le tube est vide le lecteur (consommateur) attend qu'on écrit dans le tube avant de lire.
 - si le tube est plein, un écrivain (producteur) attend qu'il y a de la place pour pouvoir écrire.

75

Système d'Exploitation

La communication inter-processus: Tubes

Il existe 2 types de tubes:

1. Les **tubes** ordinaires (**anonymes**): ne permettent que la communication entre processus issus de la même application (entre père et fils, ou entre frères).
2. Les **tubes nommés**: permettent la communication entre processus qui sont issus des applications différentes. (communication entre processus créés dans des applications différentes).

76

Système d'Exploitation

La communication inter-processus: Tubes

Tubes ordinaires (anonymes)

- ❑ Les tubes sans nom sont des liaisons unidirectionnelles de communication. La taille maximale d'un tube sans nom varie d'une version à une autre d'Unix, mais elle est approximativement égale à 4 Ko.
- ❑ Les tubes sans nom peuvent être créés par le shell ou par l'appel système pipe().
- ❑ Les tubes sans nom du shell sont créés par l'opérateur binaire « | » qui dirige la sortie standard d'un processus vers l'entrée standard d'un autre processus. Les tubes de communication du shell sont supportés par toutes les versions d'Unix.

77

Système d'Exploitation

La communication inter-processus: Tubes

1. Création d'un tube sans nom

- ❑ Un tube de communication sans nom est créé par l'appel système pipe(), auquel on passe un tableau de deux entiers :

```
int pipe(int descripteur[2]);
```
- Au retour de l'appel système pipe(), un tube aura été créé, et les deux positions du tableau passé en paramètre contiennent deux descripteurs de fichiers.
- Un descripteur comme une valeur entière que le système d'exploitation utilise pour accéder à un fichier. Dans le cas du tube on a besoin de deux descripteurs : le descripteur pour les lectures du tube et le descripteur pour les écritures dans le tube.
- ❑ L'accès au tube se fait via les descripteurs. Le descripteur de l'accès en lecture se retrouvera à la position 0 du tableau passé en paramètre, alors que le descripteur de l'accès en écriture se retrouvera à la position 1. Seul le processus créateur du tube et ses descendants (ses fils) peuvent accéder au tube. Si le système ne peut pas créer de tube pour manque d'espace, l'appel système pipe() retourne la valeur -1, sinon il retourne la valeur 0.

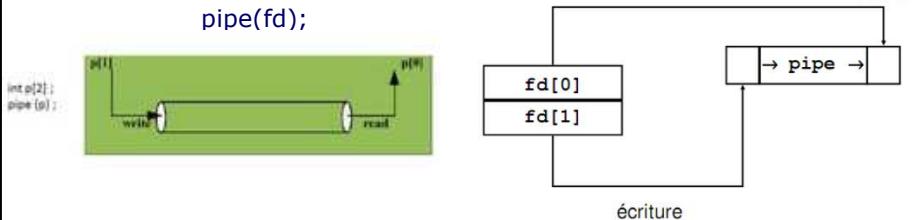
78

Système d'Exploitation

La communication inter-processus: Tubes

1. Création d'un tube sans nom

- **Exemple:** Les déclarations suivantes créent le tube montré sur la figure suivante :
`int p[2];
pipe(fd);`



Le processus père crée un tube de communication sans nom en utilisant l'appel système `pipe()`

- Le processus père crée un ou plusieurs fils en utilisant l'appel système `fork()`
- Le processus écrivain ferme l'accès en lecture du tube
- De même, le processus lecteur ferme l'accès en écriture du tube
- Les processus communiquent en utilisant les appels système `write()` et `read()`
- Chaque processus ferme son accès au tube lorsqu'il veut mettre fin à la communication via le tube.

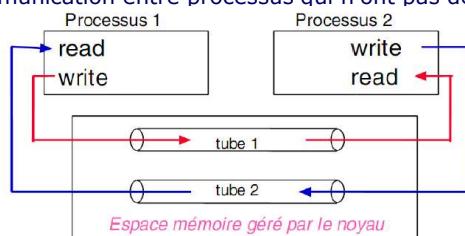
79

Système d'Exploitation

La communication inter-processus: Tubes

2. Les Files Fifo (Tubes nommés)

- Un tube nommé est un tube qui a un nom dans le système de fichiers. Tout processus peut l'ouvrir à condition d'en connaître le nom et d'en avoir les droits d'accès. Un processus ayant ouvert un tube en écriture (resp. lecture) est suspendu tant qu'il n'existe pas de processus lecteur(resp. écrivain). Les processus n'ont pas forcément de lien de parenté (comme c'est le cas pour les tubes anonymes).
- L'appel système `mkfifo(nom_type, mode)` crée un tube nommé, l'argument mode indique les permissions à accorder au tube .
- Contrairement aux tubes anonymes, d'une part les tubes nommés possèdent un nom dans le système de fichier et d'autre part ils peuvent être utilisés comme moyen de communication entre processus qui n'ont pas de lien de parenté.



80

Système d'Exploitation

La communication inter-processus: Tubes

2. Les Files Fifo (Tubes nommés)

Exemple: Client-serveur avec des tubes nommés

- ❑ On suppose qu'un serveur a accès à un fichier notes.txt contenant les notes d'étudiants. Chaque client envoie ses requêtes dans le tube nommé accesspoint : il écrit dans le tube par write(accesspoint,"PID-210+Toto",12); signifiant qu'il souhaite connaître la note de l'étudiant Toto("210" est son numéro de PID).
- ❑ Le serveur lit dans le tube read(accesspoint, tamponreq,100); (où tamponreq est un tableau de caractères).
- ❑ Il cherche la note correspondant à Toto dans notes.txt. Puis il crée le tube nommé rep210 et écrit par write(rep210,"14",2); la note de Toto(14 en l'occurrence).
- ❑ Enfin, le client récupère la note par read(rep210,tableau,20); où tableau est un tableau de caractères.
- ❑ Dans cet exemple on a utilisé le protocole de communication suivant:
 - Les clients envoient leur requête sur le tube accesspoint puis reçoivent les réponses sur le tube repPID (où PID est le numéro de pid du client).
 - Quand un client se termine, son tube nommé est détruit. L'envoi d'un message client(une requête) consiste en une écriture dans le tube nommé du serveur. La réception d'un message client consiste en la lecture dans le tube nommé du client.

81

Système d'Exploitation

La communication inter-processus: Les signaux

- ❑ Les signaux sont une autre forme de communication entre processus. Ils sont utilisés pour rendre compte à un processus d'un événement ou d'une erreur. Ils peuvent être générés à la suite d'un événement software (CTRL-C, violation de segment) ou hardware (erreur de bus, périphérique non prêt).
- ❑ Il existe différents signaux pré-définis par le système lui-même qui provoquent la terminaison du processus si celui-ci ne prend fait rien à la réception d'un tel signal. Les comportements par défaut des signaux sont les suivants :
 - Le signal est ignoré après avoir été reçu.
 - Le processus est terminé après la réception.
 - Un fichier core sera écrit puis le processus se terminera.
 - Le processus se paralyse à la réception du signal.

82

Système d'Exploitation

La communication inter-processus: Les signaux

- ❑ La création d'un fichier core donnera toutes les informations nécessaires pour permettre d'étudier le moment précis ou le processus à reçu ce signal. Il y a 32 signaux définis, certains peuvent être interceptés et pris en charge par le processus d'autres ne peuvent être intercepté ni ignorés. L'ensemble des signaux définis par le système linux se trouvent dans /usr/include/bits/signum.h.
- ❑ La fonction qui permet d'envoyer un signal à un processus est kill() est l'équivalent de la commande kill sous Unix :

```
int kill(pid_t pid, int sig)
```

 - Si pid est positif, sig est envoyé au processus pid.
 - Si pid est nul, sig est envoyé à tous les processus appartenant au même groupe que le processus appelant.
 - Si pid vaut -1 le signal est envoyé à tous les processus sauf le premier (init).
 - La fonction renvoie 0 en cas de réussite, -1 en cas d'échec.

83

Système d'Exploitation

La communication inter-processus: Les signaux

Il existe différents signaux pré-définis par le système lui-même qui provoquent la terminaison du processus

Quelques exemples de signaux

- SIGHUP (1) : fermeture terminal ⇒ à tous les processus attachés
- SIGINT (2) : Ctrl-D dans un terminal ⇒ au processus au premier plan
- SIGQUIT (3) : demander au processus de terminer (Ctrl-D dans un terminal)
- SIGILL (4) : instruction illégale (envoyé par le noyau)
- SIGFPE (8) : division par 0 (envoyé par le noyau)
- SIGKILL (9) : pour terminer un processus
- SIGSEGV (11) : accès mémoire invalide (envoyé par le noyau)
- SIGTERM (15) : argument par défaut de la commande kill
- SIGCHLD (17) : envoyé par le noyau lors de la mort d'un fils
- SIGCONT (18) : redémarre un processus suspendu (avec bg ou fg)
- SIGTSTP (20) : suspend un processus (généré par Ctrl-z)
- SIGUSR1 (10) : libre, sémantique définie pour chaque processus
- SIGUSR2 (12) : libre, sémantique définie pour chaque processus

84

Système d'Exploitation

La communication inter-processus: Les signaux

États d'un signal

Un signal est **envoyé** à un processus destinataire et **reçu** par ce processus. Tant qu'il n'a pas été pris en compte par le destinataire, le signal est **pendant**. Lorsqu'il est pris en compte (exécution du traitant), le signal est dit **traité**.

Qu'est-ce qui empêche que le signal soit immédiatement traité dès qu'il est reçu ?

- Le signal peut être **bloqué**, ou **masqué** (c'est à dire retardé) par le destinataire. Il est délivré dès qu'il est débloqué.
- En particulier, un signal est **bloqué** pendant l'**exécution du traitant** d'un signal du même type ; il reste bloqué tant que ce traitant n'est pas terminé.

Point Important : il ne peut exister qu'**un seul signal pendant** d'un type donné (il n'y a qu'un bit par signal pour indiquer les signaux de ce type qui sont pendants). **S'il arrive un autre signal du même type, il est perdu.**

85

Système d'Exploitation

La communication inter-processus: Les signaux

Envoi d'un signal

Un processus peut envoyer un signal à un autre processus. Pour cela, il utilise la primitive `kill` (appelée ainsi pour des raisons historiques ; un signal ne tue pas forcément son destinataire).

Utilisation : `kill(pid_t pid, int sig)`

Effet : Soit `p` le numéro du processus émetteur du signal

Le signal de numéro `sig` est envoyé au(x) processus désigné(s) par `pid` :

- Si `pid > 0` le signal est envoyé au processus de numéro `pid`
- Si `pid = 0` le signal est envoyé à tous les processus du même groupe que `p`
- Si `pid = -1` le signal est envoyé à tous les processus
- Si `pid < 0` le signal est envoyé à tous les processus du groupe `-pid`

Restrictions : un processus (sauf s'il a les droits de `root`) n'est autorisé à envoyer un signal qu'aux processus ayant le même `uid` (identité d'utilisateur) que lui.

Le processus de numéro 1 ne peut pas recevoir certains signaux (notamment `SIGKILL`). Étant donné son rôle particulier, il est protégé pour assurer la sécurité et la stabilité du système.

86

Système d'Exploitation

La communication inter-processus: Les Sockets

- ❑ Un socket est un dispositif de communication bidirectionnel pouvant être utilisé pour communiquer avec un autre processus sur la même machine ou avec un processus s'exécutant sur d'autres machines. Les programmes Internet comme Telnet, rlogin, FTP, talk et le World Wide Web utilisent des sockets.
- ❑ Pour créer un socket, il faut indiquer trois paramètres: le style de communication, l'espace de nommage et le protocole.
- ❑ Un style de communication contrôle la façon dont le socket traite les données transmises et définit le nombre d'interlocuteurs. Lorsque des données sont envoyées via le socket, elles sont découpées en morceaux appelés paquets. Le style de communication détermine comment sont gérés ces paquets et comment ils sont envoyés de l'émetteur vers le destinataire.
 - Le style connexion garantit la remise de tous les paquets dans leur ordre d'émission. Si des paquets sont perdus ou mélangés à cause de problèmes dans le réseau, le destinataire demande automatiquement leur retransmission à l'émetteur.
 - Le style datagramme ne garantit pas la remise ou l'ordre d'arrivée des paquets. Des paquets peuvent être perdus ou mélangés à cause de problèmes dans le réseau.

87

Système d'Exploitation

La communication inter-processus: Les Sockets

- ❑ L'espace de nommage d'un socket spécifie comment les adresses de socket sont écrites. Une adresse de socket identifie l'extrémité d'une connexion par socket. Par exemple, les adresses de socket dans « l'espace de nommage local » sont des noms de fichiers ordinaires. Dans « l'espace de nommage Internet », une adresse de socket est composée de l'adresse Internet (également appelée adresse IP) d'un hôte connecté au réseau et d'un numéro de port. Le numéro de port permet de faire la distinction entre plusieurs sockets sur le même hôte.
- ❑ Un protocole spécifie comment les données sont transmises. Parmi ces protocoles, on peut citer TCP/IP; les deux protocoles principaux utilisés pour Internet, le protocole réseau AppleTalk; et le protocole de communication locale d'UNIX.

88

Système d'Exploitation

LES PROCESSUS

- 1- Le concept de processus
- 2- La communication inter-processus
- 3- **Le problème des philosophes**
- 4- Ordonnancement des processus
- 5- Les processus de l'UNIX



89

Système d'Exploitation

Problème des philosophes

- Il s'agit d'un problème théorique très intéressant proposé et résolu aussi par Dijkstra. Cinq philosophes sont assis autour d'une table. Sur la table, il y a alternativement cinq plats de spaghetti et cinq fourchettes.
- Un philosophe passe son temps à manger et à penser. Pour manger son plat de spaghetti, un philosophe a besoin de deux fourchettes qui sont de part et d'autre de son plat. chaque philosophe de se livrer à ses activités (penser et manger) sans jamais être bloqué.



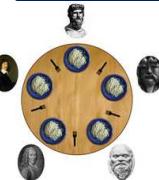
- Il est important de signaler que si tous les philosophes prennent En même temps chacun une fourchette, aucun d'entre eux ne pourra prendre l'autre fourchette (situation d'interblocage). Pour éviter cette situation, un philosophe ne prend jamais une seule fourchette. Les fourchettes sont les objets partagés. L'accès et l'utilisation d'une fourchette doit se faire en exclusion mutuelle. On utilisera le sémaphore mutex pour réaliser l'exclusion mutuelle.

90

Système d'Exploitation

Problème des philosophes

- La situation est la suivante (resumé):
 - Cinq philosophes (initialement, mais il peut y en avoir beaucoup plus) sont assis autour d'une table;
 - Chacun des philosophes a devant lui un plat de spaghetti;
 - À gauche de chaque plat de spaghetti se trouve une fourchette.
- Un philosophe n'a que trois états possibles, associés aux actions suivantes:
 - Penser pendant un temps indéterminé;
 - Être affamé pendant un temps déterminé et fini (sinon il y a famine);
 - Manger pendant un temps déterminé et fini.
- Des contraintes extérieures s'imposent à cette situation:
 - Pour manger, un philosophe a besoin de deux fourchettes: celle qui se trouve à gauche de sa propre assiette, et celle qui se trouve à droite (c'est-à-dire les deux fourchettes qui entourent sa propre assiette);
 - Quand un philosophe a faim, il va se mettre dans l'état «affamé» et attendre que les fourchettes soient libres;
 - Si un philosophe n'arrive pas à s'emparer d'une fourchette, il reste affamé pendant un temps déterminé, en attendant de renouveler sa tentative.



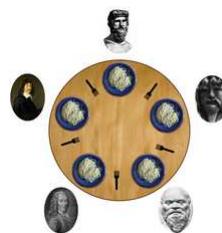
Le problème consiste à trouver un ordonnancement des philosophes tel qu'ils puissent tous manger, chacun à leur tour. Cet ordre est imposé par la solution que l'on considère comme celle de Dijkstra avec sémaphores ou Courtois avec des compteurs.

91

Système d'Exploitation

Problème des philosophes : Solutions

- L'une des principales solutions à ce problème est celle du **sémaphore**, proposée également par **Dijkstra**.
- Une autre solution consiste à attribuer à chaque philosophe un temps de réflexion aléatoire en cas d'échec (cette solution est en réalité incorrecte).
- Il existe des compromis qui permettent de limiter le nombre de philosophes gênés par une telle situation, notamment une toute simple se basant sur la technique hiérarchique de Havender qui limite le nombre de philosophes touchés à un d'un côté et deux de l'autre.



92

Système d'Exploit



Problème des philosophes : Solutions

La solution de Chandy/Misra

- En 1984, K. M. Chandy et J. Misra proposèrent une nouvelle solution permettant à un nombre arbitraire n d'agents identifiés par un nom quelconque d'utiliser un nombre m de ressources. Le protocole élégant et générique est le suivant:
 1. Pour chaque paire de philosophes pouvant accéder à la même fourchette, on commence par la donner à celui des deux qui a le plus petit nom (selon une certaine relation d'ordre). Toute fourchette est soit propre soit sale. Au début, toutes les fourchettes sont sales.
 2. Lorsqu'un philosophe veut manger, il doit obtenir les fourchettes de ses deux voisins. Pour chaque fourchette qui lui manque, il émet poliment une requête.
 3. Lorsqu'un philosophe qui a une fourchette en main entend une requête pour celle-ci,
 - soit la fourchette est propre et il la garde.
 - soit la fourchette est sale, alors il la nettoie et il la donne.
- Après qu'un philosophe a fini de manger, ses deux fourchettes sont devenues sales. Si un autre philosophe avait émis une requête pour obtenir une de ses fourchettes, il la nettoie et la donne.

93

Système d'Exploitation

LES PROCESSUS

- 1- Le concept de processus
- 2- La communication inter-processus
- 3- Le problème des philosophes
- 4- **Ordonnancement des processus**
- 5- Les processus de l'UNIX



94

Système d'Exploitation

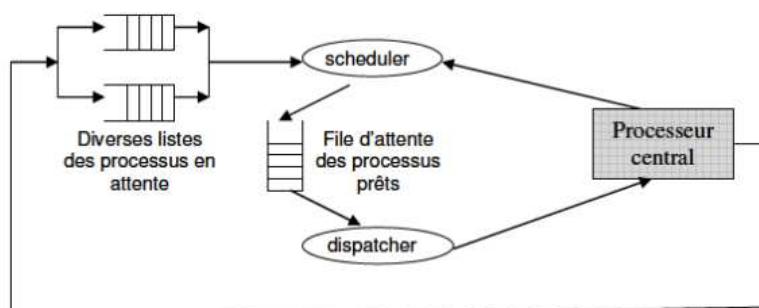
Ordonnancement des processus

- Dans les systèmes d'exploitation, l'**ordonnanceur** est le composant du **noyau** du système d'exploitation choisissant l'ordre d'exécution des **processus** sur les **processeurs** d'un ordinateur. En anglais, l'ordonnanceur est appelé **scheduler**.
- **Critères** d'ordonnancement:
 - L'ordre d'arrivée
 - Durée d'exécution
 - La priorité

95

Système d'Exploitation

Ordonnanceur



Scheduler : Ordonnanceur en anglais

96

Système d'Exploitation

Ordonnancement des processus : Stratégies d'ordonnancement

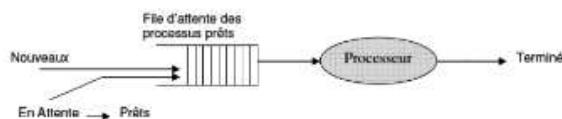
- Les critères permettant de comparer les stratégies d'ordonnancement:
 - Le taux d'utilisation de l'unité centrale: le rapport entre la durée pendant laquelle l'unité centrale est active et la durée totale.
 - Le temps de traitement moyen: la moyenne des intervalles de temps séparant la soumission et l'accomplissement d'un processus.
 - Le temps d'attente moyen: la moyenne des intervalles de temps séparant le lancement d'un processus et sa mise en exécution.
- Un bon algorithme d'ordonnancement doit:
 - Maximiser le taux d'utilisation de l'UC et le débit;
 - Minimiser le temps moyen de traitement;
 - Minimiser le temps moyen d'attente;
 - Minimiser le temps de réponse

97

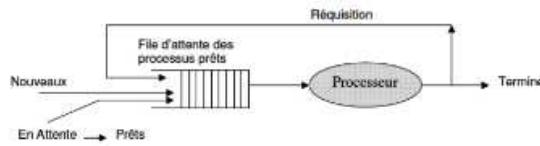
Système d'Exploitation

Algorithmes d'ordonnancement

- Deux types d'algorithmes d'ordonnancement:
 - Les algorithmes **non-préemptifs (sans réquisition)** empêchent l'appropriation du processeur par un processus avant la fin du processus courant



- Les algorithmes **préemptifs (avec réquisition)** : possibilité d'appropriation du processeur par un processus avant la fin du processus courant



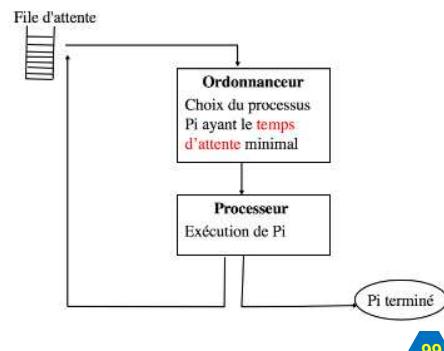
98

Système d'Exploitation

Algorithmes d'ordonnancement

Algorithme non-préemptif: FIFO

- ❑ FIFO traite les processus dans l'ordre de leur soumission (**date d'arrivée**) sans aucune considération de leur temps d'exécution.
- ❑ L'organisation de la file d'attente des processus prêts est donc tout simplement du "**First In First Out**".
- ❑ L'algorithme FIFO consiste à choisir à un instant donné, le processus qui est depuis le **plus longtemps** dans la file d'attente, ce qui revient à choisir celui disposant du temps d'arrivée minimal et l'exécuter pendant un temps d'exécution bien défini.
- ❑ Ce procédé est répété jusqu'à épuisement des processus dans la file d'attente.



99

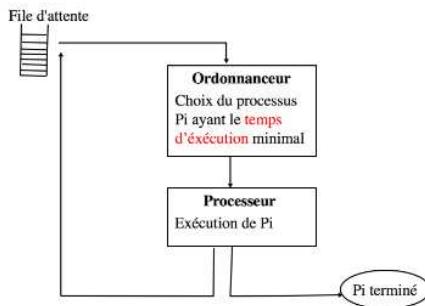
Système d'Exploitation

Algorithmes d'ordonnancement

Algorithme non-préemptif: SJF

SJF: Shortest Job First.

- ❑ SJF choisit de façon prioritaire les processus ayant le plus **court temps d'exécution** sans réellement tenir compte de leur date d'arrivée.
- ❑ Ce procédé est répété jusqu'à épuisement des processus dans la file d'attente.

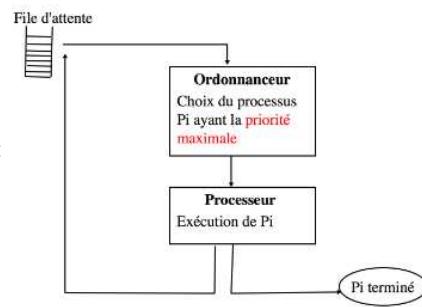


100

Système d'Exploitation

Algorithme non-préemptif: Algorithme basé sur la priorité

- ❑ Les algorithmes fondés sur les priorités attribuées par le système d'exploitation aux processus choisissent les processus les **plus prioritaires** sans prise en considération d'une manière générale des données durée d'exécution et date d'arrivée des processus.
- ❑ Le principe de cet algorithme consiste à associer à chaque processus une priorité. Le processeur est alloué au processus ayant la **plus grande priorité** pendant un quantum de temps Q.
- ❑ Ce procédé est répété jusqu'à épuisement des processus se trouvant dans la file d'attente.
- ❑ C'est une généralisation des algorithmes basés sur les priorités avec réquisition.
- ❑ Les priorités peuvent être dynamiques. Elles sont, dans ce cas recalculé après chaque quantum de temps.



101

Système d'Exploitation

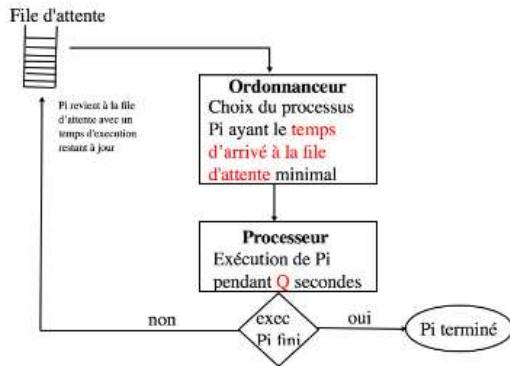
Algorithme préemptif: algorithme du tourniquet

- ❑ Le **Round Robin** (RR) décrit une stratégie dite du tourquinet où on procède à un recyclage des processus sur le processeur tant que ceux-ci ne se sont pas terminés.
- ❑ Lorsqu'un processus est élu, on lui attribue une tranche de temps fixe, appelé **quantum**, pendant laquelle il s'exécute. Au bout de ce temps, on ne poursuit plus l'exécution du processus, on lui retire donc le processeur et on le réinsère dans la file des processus prêts.
- ❑ Là, il devra attendre sa prochaine élection.
- ❑ Ainsi, le processus se voit attribuer successivement plusieurs tranches de temps avant d'atteindre sa terminaison.
- ❑ Le **RR** choisit le premier processus sur la file d'attente et l'affecte au processeur pendant un Quantum Q. Le passage d'un processus à un autre se fait selon l'ordre d'arrivée du processus dans la file d'attente.
- ❑ Le tourniquet constitue donc une généralisation de l'algorithme FIFO.

102

Système d'Exploitation

Algorithme préemptif: algorithme du tourniquet



103

Système d'Exploitation

Algorithme préemptif: SRT (Shortest Remaining Time)

SRT choisit le processus dont le temps d'exécution restant est **le plus court**.

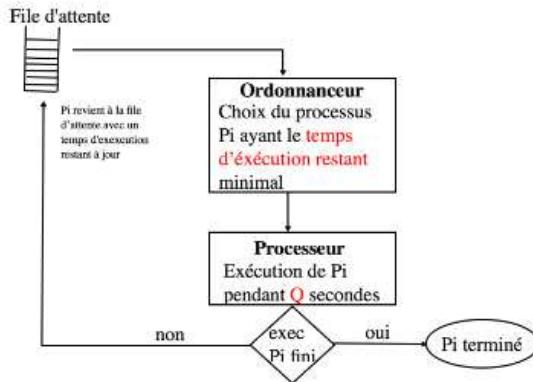
- On procède comme dans le cas de la stratégie du tourquin et à l'attribution d'un quantum fixe au delà duquel on cherche à élire le processus le plus court en terme de temps d'exécution restant pour atteindre sa terminaison.
- Le **SRT** est une généralisation sans réquisition de l'algorithme SJF. L'algorithme choisit la tâche pour laquelle le temps d'exécution est minimal et l'affecte au processeur pendant un quantum temps Q. Le passage d'un processus à un autre se fait en fonction du temps d'exécution restant associé au processus se trouvant dans la file d'attente.

104

Système d'Exploitation

Algorithme préemptif:

Algorithme préemptif: SRT (S_{hortest R_{emaining T_{ime}}}



105

Système d'Exploitation

LES PROCESSUS

- 1- Le concept de processus
- 2- La communication inter-processus
- 3- Le problème des philosophes
- 4- Ordonnancement des processus
- 5- Les processus de l'UNIX



106

Système d'Exploitation

Les processus de l'UNIX

- ❑ Chaque processus à un bloc de contrôle qui le décrit, un IDentifiant unique, peut avoir des enfants ou un père, un état, ses registres, sa mémoire et sa pile, une priorité..
- ❑ un **bloc de contrôle de processus** ou PCB (de l'anglais process control block) est une structure de données du **noyau** d'un système d'exploitation représentant l'état d'un processus donné

PCB	
Pointeur	État du processus
	Numéro du processus
	Compteur d'instructions
	Registres
	Limite de la mémoire
	Liste des fichiers ouverts
...	

- ❑ Lors d'un changement de contexte, le processus en cours est arrêté et un autre processus peut utiliser le CPU. Le noyau doit arrêter l'exécution du processus en cours, copier les valeurs des registres hardware dans le PCB, et mettre à jour [les registres avec les valeurs du nouveau processus.](#) 107

Système d'Exploitation

Les processus de l'UNIX

- ❑ Diverses implémentations existent selon les systèmes d'exploitation, mais un PCB contient en général :
 - **L'ID du processus (PID)**, l'ID du processus parent (**PPID**) et l'ID de l'utilisateur du processus (**UID**) ;
 - Les valeurs des **registres** correspondant au processus (l'état courant du processus, selon qu'il est élu, prêt ou bloqué) ;
 - Le **compteur ordinal** du processus ;
 - Le pointeur de pile : indique la position du prochain emplacement disponible dans la **pile mémoire** ;
 - **L'espace d'adressage** du processus ;
 - La liste des **descripteurs de fichiers** ;
 - La liste de gestion des **signaux** ;
 - D'autres informations telles que le temps **CPU** accumulé par le processus, etc.

PCB	
Pointeur	État du processus
	Numéro du processus
	Compteur d'instructions
	Registres
	Limite de la mémoire
	Liste des fichiers ouverts
...	

108

Système d'Exploitation

Les processus de l'UNIX

- ❑ Un processus est toujours créé par un autre processus appelé processus parent.
- ❑ Ainsi tout processus a un processus parent sauf le tout premier. Ce tout premier processus est appelé **init** et son identifiant est égal à 1 (PID = 1). Deux types de processus existent :
 - ❑ Les processus **utilisateurs**, tous issus du shell de connexion ;
 - ❑ Les processus « **démons** » :
 - « **démon** » est une traduction abusive de daemon signifiant deferred auxiliary executive monitor (divers moniteurs exécutifs auxiliaires).
- ❑ Certains processus sont permanents, c'est à dire qu'ils sont lancés au démarrage du système et arrêtés uniquement à l'arrêt du système. On appelle ces processus des daemons, le terme démon est une francisation, daemon sont des abréviations.
- ❑ Ces processus daemon assurent un service et sont souvent lancés au démarrage de la machine.
- ❑ Les principaux services assurés par des processus daemon sont l'impression, les tâches périodiques, les communications, la comptabilité, le suivi de tâche...

109

Système d'Exploitation

Les processus de l'UNIX

- ❑ Démarrage Linux/UNIX: un processus spécial appelé **init** est présent dans l'image d'amorçage.
- ❑ Lorsqu'il s'exécute, il lit un fichier indiquant combien de terminaux sont présents;
- ❑ il génère un nouveau processus par terminal.
- ❑ Ce processus attendent une ouverture de session (login);
- ❑ Si l'une d'elles réussit, le processus de login exécute un **SHELL pour accepter des commandes**.
- ❑ Ces commandes peuvent lancer d'autres processus, et ainsi de suite.
- ❑ **Tous les processus de l'ensemble du SE appartiennent à un arborescence unique, dont init est la racine.**

110

Système d'Exploitation

Les processus de l'UNIX : Création d'un processus

- ❑ Événements provoquant la création d'un processus : initialisation système, exécution d'un appel système par un processus en cours, requête utilisateur sollicitant la création d'un processus.
 - ❑ Linux/UNIX: L'appel système fork crée un clone du processus appelant.
 - ❑ Après le **fork** les deux processus, père et fils, ont la même image mémoire et les mêmes fichiers ouverts.
 - ❑ Généralement, le processus enfant exécute l'appel système execve pour modifier son image mémoire et exécuter un nouveau programme.
- ❑ **Exemple:**
- Utilisateur adresse la commande sort au shell.
 - Le shell crée un processus fils qui exécute le programme sort
- ❑ Une fois qu'un processus est créé, le père et le fils disposent de leur propre espace d'adressage: s'il arrive que l'un des processus modifie un mot dans son espace d'adressage, le changement n'est pas visible par l'autre.
 - ❑ Linux/UNIX: l'espace d'adressage initial du fils est une copie de celui du père : pas de mémoire partagée en écriture.

111

Système d'Exploitation

Les processus de l'UNIX : Arborescence des processus

- ❑ La commande **pstree** permet de visualiser l'arborescence des processus.

```
root@debian:~# pstree
systemd
└─ModemManager
   └─{gdbus}
      └─{gmain}
NetworkManager
└─dhclient
   └─{gdbus}
      └─{gmain}
agetty
anacron
at-spi2-registr
└─{gdbus}
   └─{gmain}
avahi-daemon
└─avahi-daemon
cron
dbus-daemon
exim4
lightdm
└─Xorg
   └─{InputThread}
   └─lightdm
      └─sh
         └─ssh-agent
            └─xfce4-session
               └─Thunar
                  └─light-locker
                     └─{dconf worker}
                        └─{gdbus}
                           └─{gmain}
                     └─nm-applet
                        └─{dconf worker}
                           └─{gdbus}
                              └─{gmain}
                     └─polkit-gnome-au
                        └─{gdbus}
```

112

Système d'Exploitation

Les processus de l'UNIX : Arborescence des processus

- La commande **top** permet de visualiser dynamiquement les caractéristiques des processus (l'affichage est actualisé périodiquement).
- En plus des informations sur les processus, **top** donne des indicateurs sur l'état du système : occupation de la mémoire, de l'unité centrale...
- **top** montre l'évolution de ces indicateurs en « temps réel ».

```
top - 11:46:32 up 2 min, 1 user, load average: 0,22, 0,09, 0,02
Tasks: 111 total, 1 running, 110 sleeping, 0 stopped, 0 zombie
%Cpu(s): 1,0 us, 0,7 sy, 0,0 ni, 98,3 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 s
KiB Mem : 1029368 total, 634180 free, 177968 used, 208220 buff/cache
KiB Swap: 1046524 total, 1046524 free, 0 used, 702012 avail Mem

      PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
  442 root      20   0 341044 53400 29884 S  0,7  5,2  0:01.21 Xorg
 1089 root      20   0 44892 3672 3064 R  0,7  0,4  0:00.15 top
  982 root      20   0 179624 19756 16628 S  0,3  1,9  0:00.20 xfwm4
 1005 root      20   0 503776 36712 27812 S  0,3  3,6  0:00.49 xfce4-ter-
  1 root      20   0 138916 6764 5280 S  0,0  0,7  0:01.06 systemd
  2 root      20   0      0      0      0 S  0,0  0,0  0:00.00 kthreadd
  3 root      20   0      0      0      0 S  0,0  0,0  0:00.02 ksoftirqd+
  4 root      20   0      0      0      0 S  0,0  0,0  0:00.00 kworker/0+
  5 root      0 -20      0      0      0 S  0,0  0,0  0:00.00 kworker/0+
  6 root      20   0      0      0      0 S  0,0  0,0  0:00.00 kworker/u+
  7 root      20   0      0      0      0 S  0,0  0,0  0:00.07 rcu_sched
  8 root      20   0      0      0      0 S  0,0  0,0  0:00.00 rcu_bh
  9 root      rt  0      0      0      0 S  0,0  0,0  0:00.00 migration+
 10 root      0 -20      0      0      0 S  0,0  0,0  0:00.00 lru-add-d+
```

113

Système d'Exploitation

Les processus de l'UNIX: Visualiser les processus

- On peut visualiser les processus qui tournent sur une machine avec la commande:
- **ps** : liste les processus actifs lancés dans la console courante.
- **ps (options)**, les options les plus intéressantes sont **-e** (affichage de tous les processus) et **-f** (affichage détaillé).
- *La commande ps -ef donne un truc du genre:*

```
root@debian:~# ps -ef
UID      PID  PPID  C STIME TTY      TIME CMD
root      1      0  0 11:44 ?
root      2      0  0 11:44 ?
root      3      2  0 11:44 ?
root      5      2  0 11:44 ?
root      7      2  0 11:44 ?
root      8      2  0 11:44 ?
root      9      2  0 11:44 ?
```

La signification des différentes colonnes est la suivante:

- **UID** → nom de l'utilisateur qui a lancé le processus ;
- **PID** → correspond au numéro du processus ;
- **PPID** → correspond au numéro du processus parent ;
- **C** → correspond au facteur de priorité : plus la valeur est grande, plus le processus est prioritaire ;
- **STIME** → correspond à l'heure de lancement du processus (STIME= Start Time) ;
- **TTY** → correspond au nom du terminal ;
- **TIME** → correspond à la durée de traitement du processus ;
- **CMD** → correspond au nom du processus.

114

Système d'Exploitation

Les processus de l'UNIX: Visualiser les processus

- On peut aussi visualiser les processus d'un seul utilisateur, en tapant:
\$ **ps -u root**

```
root@debian:/# ps -u root
 PID TTY      TIME CMD
  1 ?        00:00:01 systemd
  2 ?        00:00:00 kthreadd
  3 ?        00:00:00 ksoftirqd/0
  5 ?        00:00:00 kworker/0:0H
  7 ?        00:00:00 rcu_sched
  8 ?        00:00:00 rcu_bh
  9 ?        00:00:00 migration/0
 10 ?       00:00:00 lru_add_drain
 11 ?       00:00:00 watchdog/0
 12 ?       00:00:00 cpuhp/0
 13 ?       00:00:00 kdevtmpfs
 14 ?       00:00:00 netns
 15 ?       00:00:00 khungtaskd
 16 ?       00:00:00 oom_reaper
 17 ?       00:00:00 writeback
 18 ?       00:00:00 kcompactd0
 19 ?       00:00:00 ksmd
 21 ?       00:00:00 khugepaged
 22 ?       00:00:00 crypto
 23 ?       00:00:00 kintegrityd
```

- **Remarque:** Pour visualiser les processus d'un seul utilisateur, en tapant on utilise la commande: \$ **ps -u nom_user**

115

Système d'Exploitation

Les processus de l'UNIX: Visualiser les processus

Lister des informations sur les Processus

ps au : affiche la liste des processus utilisateurs

- **a** :Processus de tous les utilisateurs
- **U** :Afficher le format orienté utilisateur (affiche des informations supplémentaires sur les processus en cours d'exécution)

```
root@debian:# ps au
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START  TIME COMMAND
root      422  0.0  0.1 14536  1808  tty1      Ss+ 11:44  0:00 /sbin/agetty
root      442  0.3  5.5 343788 56400  tty7      Ssl+ 11:44  0:06 /usr/lib/xorg
root     1289  0.0  0.3 20100  3848 pts/0      Ss+ 12:01  0:00 bash
root     1376  0.0  0.3 19920  3480 pts/1      Ss 12:16  0:00 bash
root     1377  0.0  0.3 38308  3168 pts/1      R+ 12:16  0:00 ps au
root@debian:~#
```

STAT: status du processus

Certains états des processus

R Running or runnable (on run queue)

S Interruptible sleep (waiting for an event to complete)

Z Defunct ("zombie") process, terminated but not reaped by its parent.

116

Système d'Exploitation

Les processus de l'UNIX: Visualiser les processus

ps -aux : affiche la liste de tous les processus, avec leur numéro PID, le terminal tty où ils ont été lancés (sinon ?). Voici la liste des colonnes du tableau obtenu:

- **USER** ➔ à quel utilisateur appartient le processus.
- **PID** ➔ est le numéro qui identifie le processus
- **%CPU** ➔ en % les ressources du microprocesseur utilisées par le processus.
- **%MEM** ➔ en % les ressources en mémoire vive utilisées par le processus.
- **RSS** ➔ mémoire réellement utilisée en ko par le processus.
- **START** ➔ l'heure à laquelle le processus a été lancé.

- D'un UNIX à l'autre la sortie peut changer. Sous LINUX par exemple **ps -al** permet une sortie assez riche, en faisant un **man ps**, vous aurez l'éventail de tous les paramètres possibles.

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.1	0.6	138916	6784	?	Ss	11:44	0:01	/sbin/init
root	2	0.0	0.0	0	0	?	S	11:44	0:00	[kthreadd]
root	3	0.0	0.0	0	0	?	S	11:44	0:00	[ksoftirqd/0]
root	5	0.0	0.0	0	0	?	S<	11:44	0:00	[kworker/0:0]
root	7	0.0	0.0	0	0	?	S	11:44	0:00	[rcu_sched]
root	8	0.0	0.0	0	0	?	S	11:44	0:00	[rcu_bh]
root	9	0.0	0.0	0	0	?	S	11:44	0:00	[migration/0]
root	10	0.0	0.0	0	0	?	S<	11:44	0:00	[lru-add-drai]
root	11	0.0	0.0	0	0	?	S	11:44	0:00	[watchdog/0]
root	12	0.0	0.0	0	0	?	S	11:44	0:00	[cpuhp/0]
root	13	0.0	0.0	0	0	?	S	11:44	0:00	[kdevtmpfs]
root	14	0.0	0.0	0	0	?	S<	11:44	0:00	[netns]

117

Système d'Exploitation

Les processus de l'UNIX

Remarques sur Processus ou Tâches en arrière-plan

- Lorsqu'on exécute une commande UNIX, le système attend la fin de l'exécution de celle-ci pour permettre de relancer une nouvelle commande.
- Lorsqu'une commande ne nécessite pas de dialogue avec l'utilisateur, il est possible de l'exécuter en arrière plan. Pour cela, il suffit d'ajouter le caractère & à la fin de la ligne de commande. Le système lance la commande et redonne immédiatement la main à l'utilisateur. Les commandes lancées en **arrière-plan** permettent de lancer plusieurs programmes à partir d'un même terminal.
- Pour interrompre un processus en **arrière-plan**, il faut :
 - soit de sortir de la session complète; tous les processus lancés à partir de cette session sont tous interrompus;
 - soit de rechercher le numéro de processus et lui envoyer un signal lui demandant de s'interrompre grâce à la commande **kill**.
- La liste des processus en **arrière-plan** s'obtient par la commande: **jobs**

118

Système d'Exploitation

Les processus de l'UNIX

Processus en avant-plan et en arrière-plan

- ❑ Par défaut, une commande s'exécute en avant-plan (foreground (fg)).
- ❑ Par exemple, l'utilisateur saisit la commande **date**.
- ❑ Le shell crée un processus enfant et attend qu'il se termine.
- ❑ Le processus enfant exécute la commande **date**.
- ❑ Les processus parent et enfant s'exécutent **séquentiellement** (l'un après l'autre).
 - Une seule commande est donc exécutée à la fois.
- Une commande peut aussi s'exécuter en arrière-plan (en anglais *background*: **bg**).
 - Par exemple, l'utilisateur saisit **date &**.
 - Le shell crée un processus enfant et n'attend pas qu'il se termine.
 - Le processus enfant exécute la commande **date**.
 - Les deux processus, parent et enfant, s'exécutent alors "**simultanément**".

Exemples:

\$ **sleep 20** → Lancement d'une commande en foreground (fg)

\$ **sleep 15 &** → Lancement d'une commande en background (bg)

119

Système d'Exploitation

Les processus de l'UNIX

Suspension et reprise d'un processus

- Sous Unix, il est possible de suspendre (ou stopper) le processus en avant-plan en tapant **CTRL-Z**.
- Le processus suspendu (ou stoppé) pourra reprendre ultérieurement.
- Il existe deux façons de reprendre un processus suspendu (ou stoppé):
 - En **avant-plan** par la commande **fg** (**foreground**),
 - En **arrière-plan** par la commande **bg** (**background**).

Par exemple :

```
root@debian:~# firefox &
[1] 1493
root@debian:~# [GFX1-]: Unrecognized feature VIDEO_OVERLAY
```

- Un " **job**" est défini comme étant un processus en arrière-plan ou suspendu.
- La commande **jobs** permet de lister ces processus (**jobs**).
- **jobs -l** : liste des jobs avec leur numéro de job ainsi que leur numéro de processus système.

```
root@debian:~# jobs
[1]+  En cours d'exécution      firefox &
root@debian:~#
```

120

Système d'Exploitation

Les processus de l'UNIX

Terminaison d'un processus

Un processus peut se terminer suite à l'un des 4 événements :

- ❑ Sortie normale, lorsque le processus a terminé sa tâche (sous Unix par l'appel système exit)
- ❑ Sortie suite à une erreur (e.g. division par 0, inexistence d'un fichier passé en paramètre)
- ❑ Tué par un autre processus (sous Unix par l'appel système kill)
- ❑ Toutes les ressources du processus sont libérées par le SE.

121

Système d'Exploitation

Les processus de l'UNIX

Suppression (ou terminaison) d'un processus

- ❑ Généralement un processus se termine à la fin de l'exécution de la dernière instruction ; il est alors détruit par le système d'exploitation.
- ❑ Un utilisateur peut terminer (ou détruire) un processus lancé en avant-plan en tapant **CTRL-C**.
- ❑ Un utilisateur peut aussi terminer (ou détruire ou tuer) un processus avec la commande **kill** en envoyant un "signal" à un processus.
- ❑ Par défaut, la commande **kill** envoie le signal **15** de terminaison (**SIGTERM**).
 - **kill PID_processus**
- ❑ La commande **kill** peut aussi forcer la terminaison d'un processus en envoyant le signal **9** de destruction (**SIGKILL**).
- ❑ **kill -9 PID_processus** ou **kill -15 PID_processus**
- ❑ La commande GNU/Linux **kill all** envoie un signal à tous les processus de même nom.
- ❑ Notez que le **droit de détruire un processus est réservé à son propriétaire**.
- Pour détruire un processus, il faut repérer son numéro de processus (c-à-d: son PID): **ps -ax**, puis le supprimer par la commande **kill** :
 - **kill numéro_processus**
- vérifier les processus en attente : **jobs**
- tuer un job en attente : **kill % numéro_job**

122

Système d'Exploitation

Les processus de l'UNIX

Changer la priorité d'un processus

- ❑ Les processus tournent avec un certain degré de priorité, un processus plus prioritaire aura tendance à s'accaparer plus souvent les ressources du système pour arriver le plus vite possible au terme de son exécution. C'est le rôle du système d'exploitation de gérer ces priorités.
- ❑ Vous disposez de la commande **nice** pour modifier la priorité d'un processus.
- ❑ La syntaxe est la suivante : **nice -nombre commande**
- ❑ Plus la valeur du nombre est grande, plus la priorité est faible.
Par exemple une valeur de 0 donne la priorité la plus haute.
- ❑ La fourchette (ou la marge) de valeurs dépend de la distribution de l'UNIX qu'on utilise.
Par exemple : **nice -5 ps -ef**
- ❑ Généralement on utilise **nice** sur des commandes qui prennent beaucoup de temps lors de ses exécutions, alors que l'effet de **nice** est imperceptible (ou inutile), sur des commandes courantes.

Commande nohup (No Halt Up !)

- ❑ Quand on se déconnecte, le système envoie le signal HUP aux commandes restantes en arrière plan, pour les interrompre. Pour maintenir la commande même après déconnexion, on tape: **nohup commande &**
- ❑ les sorties seront dirigées dans le fichier **nohup.out**

123

Plan : Système d'Exploitation

1. INTRODUCTION AUX SYSTEMES D'EXPLOITATION
2. LES PROCESSUS

3. LES ENTREES/SORTIES

4. LA GESTION DE LA MÉMOIRE
5. LES SYSTEMES DE FICHIERS



124

Système d'Exploitation

LES ENTREES/SORTIES

Entrées/sorties et redirections

- ❑ Un programme (Processus) a pour rôle de traiter des données. Il peut donc prendre des données, les manipuler et renvoyer le résultat de ces manipulations. On peut voir le programme comme une boîte noire à qui on passe des ingrédients et duquel peuvent ressortir deux scénarios: des messages d'erreur ou des résultats. On parle d'**entrée standard**, de **sortie standard** et de **sortie d'erreur**.
- ❑ Par exemple, la commande cat, si on ne lui passe pas d'arguments se contente de recopier sur la sortie standard ce qu'on lui donne en entrée.
- ❑ **Remarque:** lorsque nous parlons des entrées sorties, nous parlons aussi des périphériques de l'ordinateur. On considérera que les périphériques sont des fichiers à part entière car, sous UNIX, des fichiers spéciaux permettent l'accès aux périphériques se trouvent dans le répertoire /dev. Dans la plupart des cas ce que l'on y copie va vers le périphérique

125

Système d'Exploitation

LES ENTREES/SORTIES

Entrées/sorties et redirections

- ❑ Lancez la commande cat, tapez un texte sur plusieurs lignes. Terminez en tapant Ctrl+d.

```
root@debian:~# cat
Bonjour chers Etudiants Filiere Licence LIDAI
Bonjour chers Etudiants Filiere Licence LIDAI

Merci
Merci
c'est juste un test
c'est juste un test
root@debian:~#
```

- ❑ Le comportement par défaut est donc que l'entrée standard est ce que vous tapez au clavier après avoir lancé la commande, la sortie standard, ce qui apparaît à l'écran.

126

Système d'Exploitation

LES ENTREES/SORTIES

Entrée/sorties standard d'un processus

- ❑ Chaque processus possède 3 flux standards qu'il utilise pour communiquer en général avec l'utilisateur :
 1. l'entrée standard nommée **stdin** (identifiant 0) : il s'agit par défaut du clavier ;
 2. la sortie standard nommée **stdout** (identifiant 1) : il s'agit par défaut de l'écran ;
 3. la sortie d'erreur standard nommée **stderr** (identifiant 2) : il s'agit par défaut de l'écran.
- ❑ La plupart des commandes que nous avons vues jusqu'à maintenant ne demandaient pas d'entrée, certaines (ls par exemple) donnaient cependant une sortie. Il est possible de rediriger ces entrées et sorties.
- ❑ Ces flux peuvent être redirigés afin que le processus interagisse avec un autre au lieu d'interagir avec l'utilisateur.

127

Système d'Exploitation

LES ENTREES/SORTIES

Qu'est-ce qu'une redirection ?

- ❑ Les redirections sont en fait un mécanisme de communication inter-processus fourni par Linux pour permettre aux programmes (processus) et aux fichiers de communiquer entre eux, en transférant la sortie d'un programme/fichier vers l'entrée d'un autre programme/fichier.
- ❑ Dans Linux, les redirections sont matérialisées par des **opérateurs < >**. L'opérateur de redirection est essentiellement un tampon ou un bloc de données qui ont deux descripteurs de fichiers ; l'un est utilisé pour la lecture, et l'autre pour l'écriture.
- ❑ Les symboles **<>** sont les opérateurs de redirection ;
 1. le symbole supérieur à **>** est utilisé pour envoyer la sortie d'un programme (processus) vers un fichier en entrée; et
 2. Le symbole inférieur à **<** est utilisé pour envoyer la sortie d'un fichier à un programme (processus) en entrée.

128

Système d'Exploitation

LES ENTREES/SORTIES

Qu'est-ce qu'une redirection ?

- ❑ **La sortie standard utilisant le symbole >** : la sortie de l'opérande de gauche est envoyée à l'opérande de droite, donc l'opérande de droite prend l'entrée à partir de l'extrémité de lecture du symbole >.
- ❑ **Sortie standard utilisant le symbole <** : La sortie de l'opérande de droite est envoyée à l'opérande de gauche, donc l'opérande de gauche prend l'entrée de l'extrémité d'écriture du symbole <.
- ❑ Ce qui nous donne :
 - **commande < fichier d'entrée** : Lit un fichier entrée
 - **commande > fichier de sortie** : Crée/Ecrit un fichier en sortie
 - **commande >> fichier de sortie** : Modifier un fichier en sortie
- ❑ **Remarque:** La redirection des entrées-sorties est l'une des plus importantes fonctionnalités du shell Linux qui a donné à la ligne de commande Linux sa flexibilité et la possibilité d'exister encore après toutes ces années.

129

Système d'Exploitation

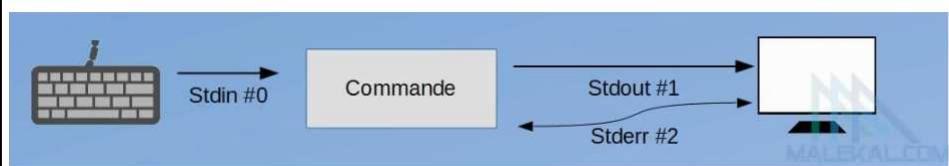
LES ENTREES/SORTIES

Qu'est-ce qu'une redirection ?

Stdin, stdout et stderr

Lorsque vous exécutez une commande Linux, il y a trois flux de données qui jouent un rôle dans cette commande :

- **L'entrée standard (stdin)** est la source des données d'entrée. Par défaut, stdin est tout texte saisi au clavier. Son ID de flux est 0.
- **La sortie standard (stdout)** est le résultat de la commande. Par défaut, elle est affichée à l'écran. Son ID de flux est 1.
- **L'erreur standard (stderr)** est le message d'erreur (le cas échéant) produit par les commandes. Par défaut, stderr est également affiché à l'écran. Son ID de flux est 2.



Système d'Exploitation

LES ENTREES/SORTIES

Stdin, stdout et stderr

The diagram illustrates the standard I/O streams in a Linux system. Data flows from a keyboard icon labeled "Stdin #0" to a central box labeled "Commande". From the "Commande" box, data is sent to a monitor icon labeled "Stdout #1" and to a terminal window icon labeled "Stderr #2". A curved arrow also points from the terminal window back to the "Commande" box.

Entrée/Sortie	Fichier	Numéro	Description
STDIN	/dev/stdin	0	Entrée Standard (Standard Input) En général le clavier
STDOUT	/dev/stdout	1	Sortie Standard (Standard output) En général l'écran
STDERR	/dev/stdout	2	Standard error

Les entrées et sorties Stdin, stdout et stderr de Linux

Système d'Exploitation

LES ENTREES/SORTIES

Qu'est-ce qu'une redirection ?

- ❑ Les flux de sorties comportent des numéros. Lorsque vous utilisez le symbole de redirection de sortie >, il signifie en fait 1>. En d'autres termes, vous dites que le flux de données avec l'ID 1 est sorti ici.
- ❑ Lorsque vous devez rediriger le stderr, vous utilisez son ID comme 2> ou 2>>. Cela signifie que la redirection de sortie concerne le flux de données stderr (ID 2).
- ❑ A retenir :
 - Il existe trois flux de données. Un flux d'entrée, stdin (0) et deux flux de données de sortie, stdout (1) et stderr (2)
 - Le clavier est le périphérique stdin par défaut et l'écran est le périphérique de sortie par défaut
 - La redirection de la sortie est utilisée avec > ou >> (pour le mode append)
 - La redirection des entrées est utilisée avec <. Le stderr peut être redirigé à l'aide de 2> ou 2>>
 - Le stderr et le stdout peuvent être combinés en utilisant 2>&1

132

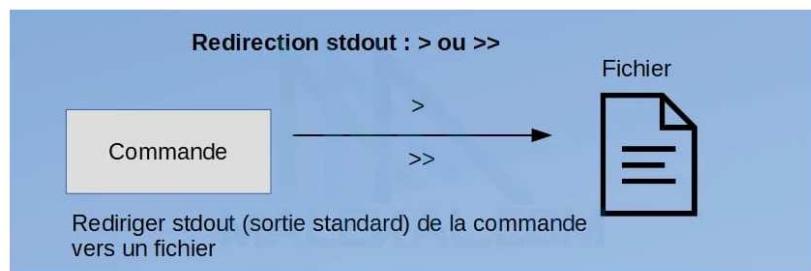
Système d'Exploitation

LES ENTREES/SORTIES

La redirection de sortie dans Linux

Rediriger la sortie standard vers un fichier

- La redirection de la sortie stdout d'une commande vers un fichier se fait de la manière suivante : **commande > fichier**



133

Système d'Exploitation

LES ENTREES/SORTIES

La redirection de sortie dans Linux

Rediriger la sortie standard vers un fichier

- Par exemple pour rediriger le résultat de la commande ps vers un fichier sortie.txt :

```
ps > sortie.txt  
Cat sortie.txt
```

```
root@debian:~# ps > sortie.txt  
root@debian:~# cat sortie.txt  
  PID TTY      TIME CMD  
1531 pts/0    00:00:00 bash  
1532 pts/0    00:00:00 ps  
root@debian:~#
```

134

Système d'Exploitation

LES ENTREES/SORTIES

La redirection de sortie dans Linux

Rediriger la sortie standard vers un fichier

- Par exemple pour rediriger le résultat de la commande ls vers un fichier sortie_Fichier.txt :

Ls > sortie_Fichier.txt

Cat sortie_Fichier.txt

```
root@debian:~# ls > sortie_Fichier.txt
root@debian:~# cat sortie_Fichier.txt
Bureau
Documents
Images
Modèles
Musique
Public
sim2023.txt
SIM2023.txt
sortie_Fichier
sortie_Fichier.txt
sortie.txt
Téléchargements
Vidéos
```

135

Système d'Exploitation

LES ENTREES/SORTIES

La redirection de sortie dans Linux

Rediriger la sortie standard vers un fichier

- Le fichier de sortie vers lequel le std out est redirigé est créé avant l'exécution de la commande prévue. Pourquoi ? Parce qu'il faut que la destination de sortie vers laquelle la sortie sera envoyée soit prête. Si le fichier est déjà existant alors cela va écraser son contenu.

- Si vous désirez **modifier le contenu du fichier**, il faut **doubler l'opérateur >** : **ls >> sortie.txt**

```
root@debian:~# cd Bureau
root@debian:~/Bureau# ls >> sortie.txt
root@debian:~/Bureau# cat sortie.txt
PID TTY      TIME CMD
1086 pts/0    00:00:00 bash
1119 pts/0    00:00:00 ps
Kernel
répertoire1
sortie.txt
tree_1.8.0-1_amd64.deb
utilisateurs autorisés
root@debian:~/Bureau#
```

La sortie de la commande va s'ajouter à la fin du fichier.

136

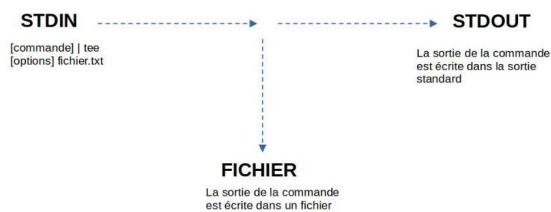
Système d'Exploitation

LES ENTREES/SORTIES

La redirection de sortie dans Linux

Rediriger la sortie standard vers stdin et un fichier

- La commande **tee** lit l'entrée standard (stdin) et l'écrit à la fois sur la sortie standard (stdout) et dans un ou plusieurs fichiers. tee fait généralement partie d'un pipeline, et un nombre quelconque de commandes peuvent la précéder ou la suivre.



137

Système d'Exploitation

LES ENTREES/SORTIES

La redirection de sortie dans Linux

Rediriger la sortie standard vers stdin et un fichier

- La commande **tee** :

La syntaxe est la suivante :

[commande] | tee [options] <nom du fichier>

Remarque:

- L'écrasement du contenu du fichier est le comportement par défaut de la commande tee.
- Utilisez l'argument -a (ou -append) pour ajouter la sortie de la commande à la fin du fichier.

[commande] | tee -a <nom du fichier>

138

Système d'Exploitation

LES ENTREES/SORTIES

La redirection de sortie dans Linux

Rediriger la sortie standard vers stdin et un fichier

- La commande **tee :ls | tee -a lidai.txt**

```
root@debian:~# ls | tee -a lidai.txt
Bureau
Documents
Images
lidai.txt
Modèles
Musique
Public
sim2023.txt
SIM2023.txt
sortie_Fichier
sortie_Fichier.txt
sortie.txt
Téléchargements
Vidéos
root@debian:~# cat lidai.txt
Bureau
Documents
Images
lidai.txt
```

139

Système d'Exploitation

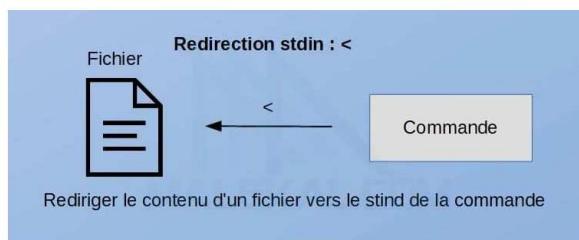
LES ENTREES/SORTIES

La redirection de sortie dans Linux

Rediriger l'entrée standard vers un fichier

- Vous pouvez utiliser la redirection stdin pour transmettre le contenu d'un fichier texte à une commande comme celle-ci :

commande < fichier



140

Système d'Exploitation

LES ENTREES/SORTIES

La redirection de sortie dans Linux

Rediriger l'entrée standard vers un fichier

- ❑ Vous pouvez utiliser la redirection stdin pour transmettre le contenu d'un fichier texte à une commande comme celle-ci :
commande < fichier
- ❑ Prenez la commande **tr** par exemple. Cette commande peut faire beaucoup de tâches, mais dans l'exemple ci-dessous, elle convertit le texte d'entrée des minuscules en majuscules :
tr a-z A-Z < fichier.txt
- ❑ TR est une commande utilisé pour traduire, convertir et/ou supprimer des caractères de l'entrée standard, écriture sur la sortie standard. tr fonctionne avec un texte en entrée.
- ❑ Remarque: la commande **tr a-z A-Z < fichier.txt** est équivalente à la commande **cat fichier.txt | tr a-z A-Z**
 - il est conseillé d'utiliser stdin plutôt que pipe, notamment pour éviter l'utilisation inutile de la commande cat.

141

Système d'Exploitation

LES ENTREES/SORTIES

La redirection de sortie dans Linux

Rediriger l'entrée standard vers un fichier

- ❑ La commande qui convertit le texte d'entrée (fichier dans le dossier Bureau) des minuscules en majuscules :

tr a-z A-Z < fichier.txt

```
root@debian:~/Bureau# cat fichier
bonjour
ca va
chers etudiants lidairoot@debian:~/Bureau# tr a-z A-Z < fichier
BONJOUR
CA VA
CHERS ETUDIANTS LIDAIRoot@debian:~/Bureau#
```

142

Système d'Exploitation

LES ENTREES/SORTIES

La redirection de sortie dans Linux

Rediriger l'entrée standard vers un fichier

- Ci après un autre exemple où on envoie la sortie de fichier.txt à la commande head.

head < fichier

- Remarque: **la commande cat** est utilisée pour imprimer le contenu d'un fichier, sur le terminal. La commande cat imprime l'intégralité du fichier sur le terminal.
- Mais parfois on peut avoir besoin d'afficher que le début.
 - Pour ce faire, vous pouvez utiliser la commande **head** qui permet d'afficher les N premières lignes d'un fichier.
 - Head est l'inverse de la commande **tail**.
 - Par défaut, elle affiche les 10 premières lignes de chaque fichier à la sortie standard.
 - Lorsque vous avez besoin d'imprimer un nombre spécifique de lignes, vous pouvez utiliser **l'option -n** suivie du nombre de lignes.

head -n 2 fichier.txt

143

Système d'Exploitation

LES ENTREES/SORTIES

La redirection de sortie dans Linux

Rediriger l'entrée standard vers un fichier

- la commande head. **head < fichier**

```
root@debian:~/Bureau# head < fichier
bonjour
ca va
chers etudiants lidai

bonjour
ca va
chers etudiants lidai

bonjour
ca va
root@debian:~/Bureau# cat fichier
bonjour
ca va
chers etudiants lidai

bonjour
ca va
chers etudiants lidai

bonjour
ca va
```

144

Système d'Exploitation

LES ENTREES/SORTIES

La redirection de sortie dans Linux

Rediriger l'entrée standard vers un fichier

- La boucle while du shell Linux utilise aussi la redirection d'entrée. Ci-dessous, la boucle while lit ligne par ligne le contenu du fichier "fichier" envoyé en entrée.

```
while read -r ligne;
do
echo "$ligne" ;
done < fichier
```

```
root@debian:~/Bureau# while read -r ligne;
> do
> echo "$ligne";
> done < fichier
bonjour
ca va
chers etudiants lidai

bonjour
ca va
chers etudiants lidai
```

145

Système d'Exploitation

LES ENTREES/SORTIES

La redirection de sortie dans Linux

Rediriger la sortie standard d'erreur vers un fichier

- Dans Linux, les commandes retournent des messages d'erreurs lorsqu'elle ne parvienne pas à effectuer l'action demandée. Par exemple, si vous lister un fichier qui n'existe pas avec ls, ce dernier retourne un message d'erreur :

ls SSD

ls: impossible d'accéder à 'SSD': Aucun fichier ou dossier de ce type

```
root@debian:~/Bureau# ls SSD
ls: impossible d'accéder à 'SSD': Aucun fichier ou dossier de ce type
root@debian:~/Bureau#
```

Cette erreur s'affiche dans une sortie spécifique nommée stderr.

146

Système d'Exploitation

LES ENTREES/SORTIES

La redirection de sortie dans Linux

Rediriger la sortie standard d'erreur vers un fichier

- ❑ Il est tout à fait possible de rediriger la sortie standard d'erreur Linux vers un fichier.
 - ❑ Pour cela, on utilise l'opérateur 2>.
 - ❑ Par exemple ci-dessous, on redirige le message d'erreur vers erreur.txt
- ```
ls SSD 2> erreur.txt
Cat erreur.txt
ls: impossible d'accéder à 'SSD': Aucun fichier ou dossier de ce type
```

```
root@debian:~/Bureau# ls SSD 2> erreur.txt
root@debian:~/Bureau# cat erreur.txt
ls: impossible d'accéder à 'SSD': Aucun fichier ou dossier de ce type
root@debian:~/Bureau#
```

147

# Système d'Exploitation

## LES ENTREES/SORTIES

### La redirection de sortie dans Linux

#### Rediriger la sortie standard d'erreur vers un fichier

- ❑ On peut combiner une redirection de la sortie standard et de la sortie d'erreur dans une même commande.

```
ls -l MMD SSD > fichier.txt 2> erreur.txt
cat fichier.txt
cat erreur.txt
```

```
root@debian:~/Bureau# ls -l MID SSD > fichier.txt 2> erreur.txt
root@debian:~/Bureau# cat fichier
bonjour
ça va
chers étudiants lidai

bonjour
ça va
chers étudiants lidai

bonjour
ça va
chers étudiants lidai

root@debian:~/Bureau# cat erreur.txt
ls: impossible d'accéder à 'MID': Aucun fichier ou dossier de ce type
ls: impossible d'accéder à 'SSD': Aucun fichier ou dossier de ce type
root@debian:~/Bureau#
```

148

# Système d'Exploitation

## LES ENTREES/SORTIES

### La redirection de sortie dans Linux

#### Rediriger la sortie standard d'erreur vers un fichier

- ❑ On peut combiner une redirection de la sortie standard et de la sortie d'erreur dans une même commande.  
`ls -l MMD SSD > fichier.txt 2> erreur.txt  
cat fichier.txt  
cat erreur.txt`
- ❑ Dans l'exemple ci-dessus, la commande ls essaie d'afficher deux fichiers. Pour un fichier, elle obtient un succès et pour l'autre, elle obtient une erreur. En fait on a rediriger :
  - le stdout vers fichier.txt (avec >)
  - et le stderr vers erreur.txt (avec 2>).

149

# Système d'Exploitation

## LES ENTREES/SORTIES

### La redirection de sortie dans Linux

#### La redirection 2>&1 de Linux

- ❑ Pour rediriger stderr vers la même adresse que stdout il faut utiliser l'opérateur 2>&1.
- ❑ Reprenons l'exemple précédent et utilisons cette fois-ci 2>&1 pour rediriger stdout et stderr vers le même fichier.
- ❑ On peut combiner une redirection de la sortie standard et de la sortie d'erreur dans une même commande.

```
ls -l MMD SSD > fichier.txt 2> &1
cat fichier.txt
```

```
root@debian:~/Bureau# ls -l MID SSD > fichier.txt 2>&1
root@debian:~/Bureau# cat fichier.txt
ls: impossible d'accéder à 'MID': Aucun fichier ou dossier de ce type
ls: impossible d'accéder à 'SSD': Aucun fichier ou dossier de ce type
root@debian:~/Bureau#
```

150

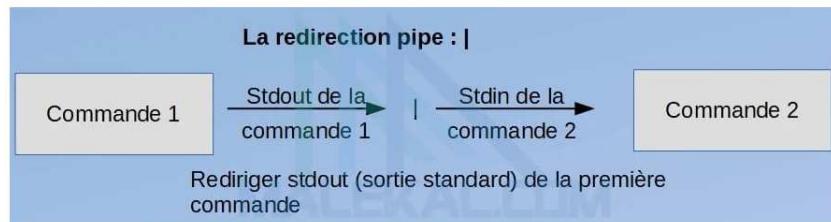
# Système d'Exploitation

## LES ENTREES/SORTIES

### La redirections entre les commandes avec pipe

- Un pipe sous Linux (**tube**) est simplement une barre verticale sur votre clavier. Il est utilisé pour considérer la commande qui se trouve à sa gauche comme une entrée pour la commande qui se trouve à sa droite.
- Avec **la redirection par pipe**, vous envoyez la sortie standard d'une commande vers l'entrée standard d'une autre commande.

**commande1 | commande2 | commande3**



151

# Système d'Exploitation

## LES ENTREES/SORTIES

### La redirections entre les commandes avec pipe

#### □ Exemple

on liste les fichiers texte avec la commande ls puis on envoie le résultat à la commande wc qui va compter le nombre de ligne données par ls. Ensuite on redirige le résultat de la commande wc dans le fichier compter.txt. Cela permet de compter le nombre de fichier texte dans un dossier.

**ls -lh \*.txt**

**ls \*.txt | wc -l > compter.txt**

**cat compter.txt**

```
root@debian:~# ls -lh *.txt
-rw-r--r-- 1 root root 2 nov. 12 13:15 compter.txt
-rw-r--r-- 1 root root 249 nov. 12 11:52 fichier.txt
-rw-r--r-- 1 root root 153 nov. 12 11:33 lidai.txt
-rw-r--r-- 1 root root 0 févr. 22 2023 sim2023.txt
-rw-r--r-- 1 root root 0 févr. 6 2023 SIM2023.txt
-rw-r--r-- 1 root root 143 nov. 12 11:11 sortie_Fichier.txt
-rw-r--r-- 1 root root 370 nov. 12 11:32 sortie.txt
root@debian:~# ls *.txt | wc -l > compter.txt
root@debian:~# cat compter.txt
```

7

root@debian:~#

152

## Plan : Système d'Exploitation

1. INTRODUCTION AUX SYSTEMES D'EXPLOITATION
2. LES PROCESSUS
3. LES ENTREES/SORTIES
- 4. LA GESTION DE LA MÉMOIRE**
5. LES SYSTEMES DE FICHIERS



153

## Système d'Exploitation

### LA GESTION DE LA MÉMOIRE



#### Généralités & Définitions

En informatique, la **mémoire** est un dispositif électronique numérique qui sert à stocker des données. La mémoire est un composant essentiel, présent dans tous les ordinateurs, les GPS et de nombreux appareils électroniques.

Il existe différents types de mémoire :

- ❑ la **mémoire vive** : abrégée avec l'acronyme anglais **RAM** (*random-access memory*), est la **mémoire informatique** dans laquelle peuvent être enregistrées les informations traitées par un appareil informatique. mémoire où chaque information stockée peut à tout moment être consultée, ou modifiée. La *mémoire centrale* des ordinateurs est la plupart du temps une mémoire vive volatile bien que le SSD remplace de plus en plus souvent ce rôle ; L'acronyme RAM date de 1965. **RAM** est composée de circuit intégrés, donc très rapide.  
Il y a deux types principaux de mémoire vive :
  - La **mémoire vive dynamique (DRAM)** qui, même sous alimentation électrique normale, doit être réactualisée périodiquement pour éviter la perte d'information ;
  - La **mémoire vive statique (SRAM)** qui n'a pas besoin d'un tel processus sous alimentation électrique normale.
- ❑ la **mémoire de masse** : est une mémoire de grande capacité, non **volatile** et qui peut être lue et écrite, entre autres, par un ordinateur.  
➢ composée de supports magnétiques (disque dur, bandes magnétiques...), beaucoup plus lente.

154

# Système d'Exploitation

## LA GESTION DE LA MÉMOIRE

### Généralités & Définitions



- ❑ **Mémoire morte** mémoire où les informations sont écrites une fois mais ne peuvent pas être modifiées. Les mémoires mortes sont utilisées par exemple pour stocker définitivement des logiciels.
- ❑ **Mémoire volatile** mémoire où les informations sont perdues lors de la mise hors tension de l'appareil. Par opposition, **une mémoire rémanente** ou non volatile est une mémoire où les informations sont conservées même après la mise hors tension de l'appareil. Les mémoires rémanentes sont utilisées pour les téléphones portables, les autoradios, les GPS, ou les appareils photo numériques.
- ❑ **Mémoire flash** mémoire rémanente dont le contenu peut être intégralement effacé en une seule opération.
- ❑ **Mémoire virtuelle** mécanisme qui permet de donner plus de mémoire au processeur pour travailler, en simulant la présence d'un type de mémoire tout en utilisant un autre type (par exemple un disque dur). Il est utilisé par exemple pour simuler la présence de **mémoire vive** en utilisant de la **mémoire de masse**.

155

# Système d'Exploitation

## LA GESTION DE LA MÉMOIRE

### Généralités & Définitions

#### Organisation de la mémoire physique

- Espace d'adresses contigües, ou presque
- Le BIOS fournit une carte de la mémoire

#### Carte de la mémoire physique

```
root@debian:~# dmesg
[0.000000] Linux version 4.9.0-6-amd64 (debian-kernel@lists.debian.org) (g
cc version 6.3.0 20170516 (Debian 6.3.0-18+deb9u1)) #1 SMP Debian 4.9.82-1+de
b9u3 (2018-03-02)
[0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-4.9.0-6-amd64 root=UUID=
385f9323-c0d4-4ebd-9d28-ce338df0e835 ro quiet
[0.000000] x86/fpu: Supporting XSAVE feature 0x001: 'x87 floating point re
gisters'
[0.000000] x86/fpu: Supporting XSAVE feature 0x002: 'SSE registers'
[0.000000] x86/fpu: Supporting XSAVE feature 0x004: 'AVX registers'
[0.000000] x86/fpu: xstate_offset[2]: 576, xstate_sizes[2]: 256
[0.000000] x86/fpu: Enabled xstate features 0x7, context size is 832 bytes
, using 'standard' format.
[0.000000] x86/fpu: Using 'eager' FPU context switches.
[0.000000] e820: BIOS-provided physical RAM map:
[0.000000] BIOS-e820: [mem 0x0000000000000000-0x000000000009fbff] usable
[0.000000] BIOS-e820: [mem 0x000000000009fc00-0x000000000009ffff] reserved
[0.000000] BIOS-e820: [mem 0x0000000000f0000-0x0000000000ffff] reserved
[0.000000] BIOS-e820: [mem 0x000000000100000-0x000000003fffffff] usable
[0.000000] BIOS-e820: [mem 0x000000003ffff000-0x000000003fffffff] ACPI dat
a
```

156

# Système d'Exploitation

## LA GESTION DE LA MÉMOIRE

### Gestion mémoire basique

On peut subdiviser les systèmes de gestion de la mémoire en deux catégories :

- ❑ les systèmes qui peuvent déplacer les processus en mémoire secondaire pour trouver de l'espace de swap.
- ❑ les systèmes qui n'utilisent pas la mémoire secondaire.

157

# Système d'Exploitation

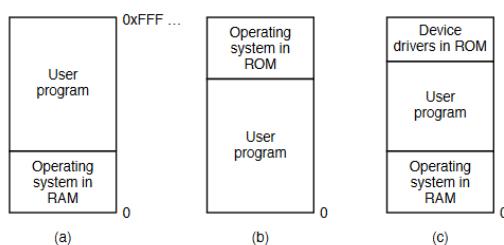
## LA GESTION DE LA MÉMOIRE

### Gestion de la mémoire sans va-et-vient ni pagination

#### Monoprogrammation sans va-et-vient ni pagination

L'approche la plus simple pour gérer la mémoire consiste à n'accepter qu'un seul processus à la fois (monoprogrammation) auquel on permet d'utiliser toute la mémoire disponible en dehors de celle qu'utilise le système.

Cette approche était utilisée, par exemple, dans les premiers micro-ordinateurs IBM PC utilisant MS-DOS comme système d'exploitation.



158

# Système d'Exploitation

## LA GESTION DE LA MÉMOIRE

### Gestion de la mémoire sans va-et-vient ni pagination

#### Multiprogrammation avec des partitions fixes

- ❑ Dans un système multiprogrammé, il faut gérer la répartition de l'espace entre les différents processus. Cette partition peut être faite une fois pour toute au démarrage du système par l'opérateur de la machine, qui subdivise la mémoire en partitions fixes.
- ❑ Chaque nouveau processus est placé dans la file d'attente de la plus petite partition qui peut le contenir (figure a). Cette façon de faire peut conduire à faire attendre un processus dans une file, alors qu'une autre partition pouvant le contenir est libre. L'alternative à cette approche consiste à n'utiliser qu'une seule file d'attente : dès qu'une partition se libère, le système y place le premier processus de la file qui peut y tenir (figure b).

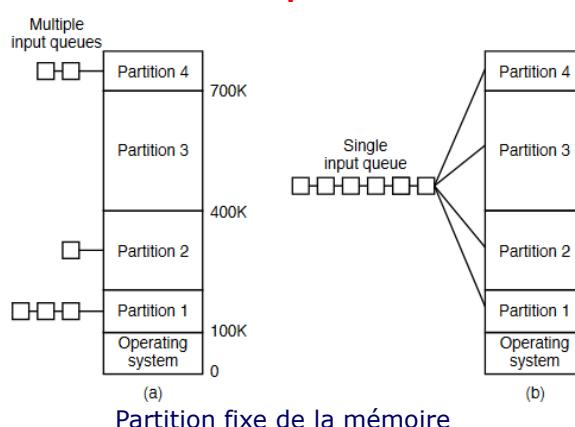
159

# Système d'Exploitation

## LA GESTION DE LA MÉMOIRE

### Gestion de la mémoire sans va-et-vient ni pagination

#### Multiprogrammation avec des partitions fixes



160

# Système d'Exploitation

## LA GESTION DE LA MÉMOIRE

### Swaping

- ❑ Dans un système qui peut déplacer les processus sur le disque quand il manque d'espace, le mouvement des processus entre la mémoire et le disque (va-et-vient) risque de devenir très fréquent, si on n'exploite pas au mieux l'espace libre en mémoire principale. Sachant que les accès au disque sont très lents, les performances du système risquent alors de se détériorer rapidement. Il faut alors exploiter au mieux l'espace libre en mémoire principale, en utilisant un partitionnement dynamique de la mémoire.
- ❑ Le fait de ne plus fixer le partitionnement de la mémoire rend plus complexe la gestion de l'espace libre. Celui-ci doit cependant permettre de trouver et de choisir rapidement de la mémoire libre pour l'attribuer à un processus. Il est donc nécessaire de mettre en œuvre des structures de données efficaces pour la gestion de l'espace libre.

161

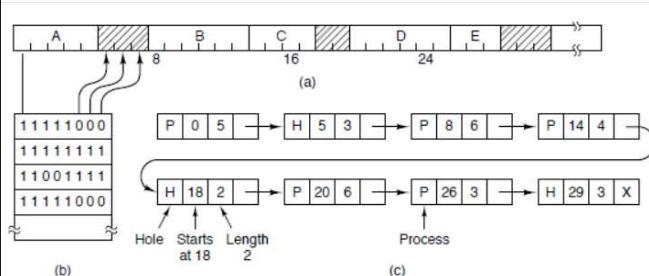
# Système d'Exploitation

## LA GESTION DE LA MÉMOIRE

### Swaping : Gestion de la mémoire par tables de bits

La mémoire est divisée en unités (quelques mots mémoire à plusieurs kilooctets), à chaque unité on fait correspondre dans une table de bits :

- ❑ la valeur 1 si l'unité mémoire est occupée ;
- ❑ la valeur 0 si l'unité mémoire est libre.



(a) Une partie de la mémoire occupée par cinq processus (A,B,C,D et E) et trois zones libres. Les régions hachurées sont libres. (b) La table de bits (0 = libre, 1 = occupée). (c) Liste chaînée des zones libres.

Pour allouer  $k$  unités contiguës, le gestionnaire de mémoire doit parcourir la table de bits à la recherche de  $k$  zéros consécutifs. Cette recherche s'avère donc lente pour une utilisation par le gestionnaire de la mémoire et est rarement utilisée à cet effet.

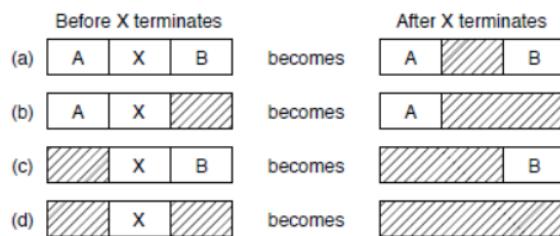
162

# Système d'Exploitation

## LA GESTION DE LA MÉMOIRE

### Swapping : Gestion de la mémoire par listes chaînées

- Une autre solution consiste à chaîner les segments libres et occupés. La figure ci-dessus montre l'exemple d'un tel chaînage, les segments occupés par un processus sont marqués (P) les libres sont marqués (H). La liste est triée sur les adresses, ce qui facilite la mise à jour.
- Lorsqu'on libère la mémoire occupée par un segment, il faut fusionner le segment libre avec le ou les segments adjacents libres s'ils existent :



Ex : Quatre combinaisons de voisins possibles d'un processus X qui termine et libère le segment qu'il occupe

163

# Système d'Exploitation

## LA GESTION DE LA MÉMOIRE

### La mémoire virtuelle

- Le principe de la mémoire virtuelle consiste à considérer un espace d'adressage virtuel supérieur à la taille de la mémoire physique, sachant que dans cette espace d'adressage, et grâce au mécanisme de va-et-vient sur le disque, seule une partie de la mémoire virtuelle est physiquement présente en mémoire principale à un instant donné.
- Ceci permet de gérer un espace virtuel beaucoup plus grand que l'espace physique sans avoir à gérer les changements d'adresses physiques des processus après un va-et-vient : car même après de multiples va-et-vient un processus garde la même adresse virtuelle. C'est notamment très utile dans les systèmes multiprogrammés dans lesquels les processus qui ne font rien (la plupart) occupent un espace virtuel qui n'encombre pas nécessairement l'espace physique.

164

# Système d'Exploitation

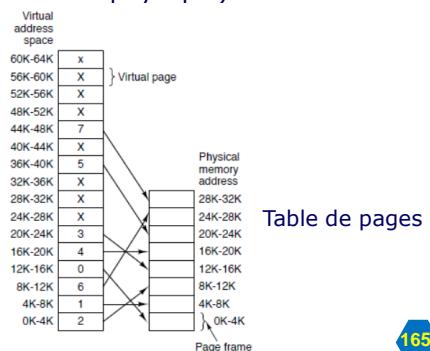
## LA GESTION DE LA MÉMOIRE

### La mémoire virtuelle : Pagination

La plupart des architectures actuellement utilisées reposent sur des processeurs permettant de gérer un espace virtuel paginé : l'espace d'adressage virtuel est divisé en pages, chaque page occupée par un processus est soit en mémoire physique soit dans le disque (va-et-vient).

La correspondance entre une page virtuelle et la page physique à laquelle elle peut être associée (si elle est en mémoire physique) est réalisée en utilisant une table de pages.

Lorsque le processeur doit exécuter une instruction qui porte sur un mot mémoire donné dont il a l'adresse virtuel, il cherche dans la table des pages l'entrée qui correspond à la page contenant le mot. Si la page est présente en mémoire il lit le mot, sinon il déclenche un déroutement de type défaut de page. A la suite de ce déroutement, le système de gestion de la mémoire est appelé afin de charger la page manquante à partir du disque (va-et-vient)



165

# Système d'Exploitation

## LA GESTION DE LA MÉMOIRE

### La mémoire virtuelle : Pagination

Chaque entrée de table de pages consiste en un descripteur de page qui contient généralement ces informations (au moins) :

- ❑ Numéro de la page en mémoire physique (figure) si celle-ci est présente en mémoire.
- ❑ Un bit qui indique la présence (ou non présence) de la page en mémoire.
- ❑ Un bit de modification : qui mémorise le fait qu'une page a été modifiée, ceci permet au gestionnaire de mémoire de voir s'il doit la sauver sur le disque dans le cas où il veut récupérer l'espace qu'elle occupe pour charger une autre page.
- ❑ Un bit de référencement : qui est positionné si on fait référence à un mot mémoire de la page, ceci permet au système de trouver les pages non référencées qui seront les meilleures candidates pour être retirées de la mémoire physique s'il y a besoin et manque de mémoire physique.

166

# Système d'Exploitation

## LA GESTION DE LA MÉMOIRE

### Organisation de la mémoire virtuelle (VM)

- ❑ Le système UNIX fonctionne en mémoire virtuelle paginée. Ceci permet de faire fonctionner des processus demandant une quantité d'espace mémoire supérieure à la mémoire physique installée.
- ❑ Lorsqu'un processus demande l'allocation d'une page de mémoire et qu'il n'y en a pas de disponible en mémoire centrale, le noyau traite un *défaut de page*. Il choisit une page (qui n'a pas été utilisé depuis longtemps) et l'écrit sur une partition spéciale du disque dur.
- ❑ La place libérée est alors attribuée au processus demandeur.
- ❑ Ce mécanisme demande la réservation d'une (ou plusieurs) partition spéciale sur l'un des disques durs, nommée *partition de swap*. La mémoire disponible pour les processus est donnée par la somme de la taille de mémoire physique (RAM) et des partitions de swap. Bien entendu, les performances du système se dégradent lorsque la fréquence des défauts de page augmente ; dans ce cas, il faut augmenter la mémoire physique.
- ❑ Sur un système typique, la partition de swap est deux à trois fois plus grande que la mémoire centrale (exemple : PC avec 32Mo de RAM, partition de swap de 64Mo).

167

# Système d'Exploitation

## LA GESTION DE LA MÉMOIRE

### Organisation de la mémoire virtuelle (VM)

#### Paging

- ❑ Diviser la mémoire (physique et virtuelle) en des morceaux de taille fixe, appelés "pages"
- ❑ Maintenir des "page tables" pour faire l'association entre une page virtuelle et une physique
- ❑ Maintenir une "carte de mémoire" avec des informations sur chaque page physique
- ❑ Utiliser des composants matériels pour accélérer la conversion virtuel → physique

168

# Système d'Exploitation

## LA GESTION DE LA MÉMOIRE

### Organisation de la mémoire virtuelle (VM)

#### Espace d'adressage des processus

```
root@debian:~# ps -al
F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY TIME CMD
0 S 0 1343 1064 0 80 0 - 3128 - pts/0 00:00:03 watch
0 R 0 4472 1463 0 80 0 - 7467 - pts/1 00:00:00 ps
root@debian:~# pmap -x 1343
1343: watch -n 1 free -m
Address Kbytes RSS Dirty Mode Mapping
000056084a8e7000 20 20 0 r-x-- watch
000056084a8e7000 0 0 0 r-x-- watch
000056084aaeb000 4 4 4 r---- watch
000056084aaeb000 0 0 0 r---- watch
000056084aaec000 4 4 4 rw--- watch
000056084aaec000 0 0 0 rw--- watch
000056084b151000 264 216 216 rw--- [anon]
000056084b151000 0 0 0 rw--- [anon]
00007f7634350000 1620 1144 0 r-x-- libc-2.24.so
00007f7634350000 0 0 0 r-x-- libc-2.24.so
00007f76344e5000 2048 0 0 ----- libc-2.24.so
00007f76344e5000 0 0 0 ----- libc-2.24.so
00007f76346e5000 16 16 16 r---- libc-2.24.so
00007f76346e5000 0 0 0 r---- libc-2.24.so
00007f76346e9000 8 8 8 rw--- libc-2.24.so
00007f76346e9000 0 0 0 rw--- libc-2.24.so
```

169

# Système d'Exploitation

## LA GESTION DE LA MÉMOIRE

### La VM, pour quoi faire?

- ❑ Les processus ont un espace d'adressage plus large
- ❑ La VM peut satisfaire des allocations qui dépasseraient la capacité physique: "overcommit"
- ❑ Partage de la mémoire entre plusieurs processus
- ❑ Prétendre satisfaire des allocations et retarder l'allocation réelle autant que possible: "Copy-on-Write" (COW) ou "Lazy Allocation"
- ❑ Prétendre satisfaire des chargements du disque et retarder l'accès réel autant que possible: "demand paging"
- ❑ Servir de cache des accès disque (page cache); pas vraiment VM, mais..
- ❑ Permettre de "mapper" des fichiers à l'espace d'adressage d'un processus: assurer un "zero-copy"

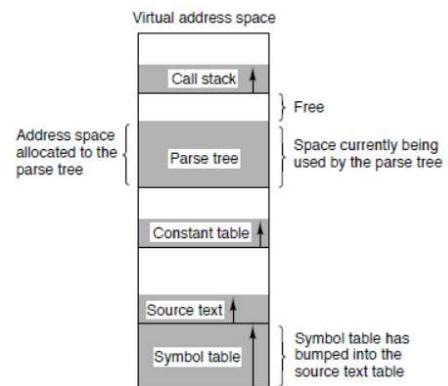
170

# Système d'Exploitation

## LA GESTION DE LA MÉMOIRE

### La segmentation : Implantation de segments purs

- La segmentation de la mémoire permet de traiter la mémoire non plus comme un seul espace d'adressage unique, mais plutôt comme un ensemble de segments (portions de la mémoire de taille variable), ayant chacune son propre espace d'adressage. Ceci permet aux processus de simplifier considérablement la gestion de mémoire propre.
- Ainsi un processus qui utilise différentes tables : table des symboles, code, table des constantes, etc. doit se préoccuper de la position relative de ses tables et doit gérer les déplacements de tables quand celles-ci croissent et risquent de se recouvrir :

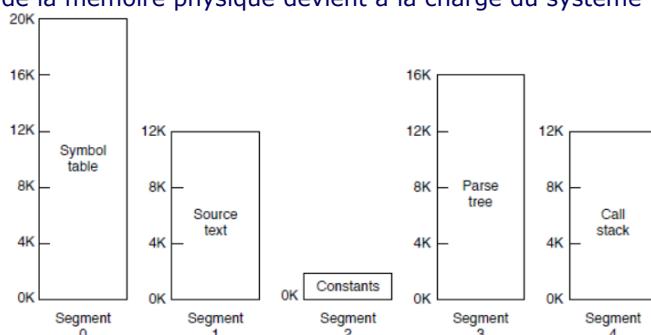


# Système d'Exploitation

## LA GESTION DE LA MÉMOIRE

### La segmentation : Implantation de segments purs

Alors que dans une gestion de la mémoire segmentée, il suffit d'allouer un segment pour chaque table, la gestion de la position des segments dans l'espace d'adressage de la mémoire physique devient à la charge du système :



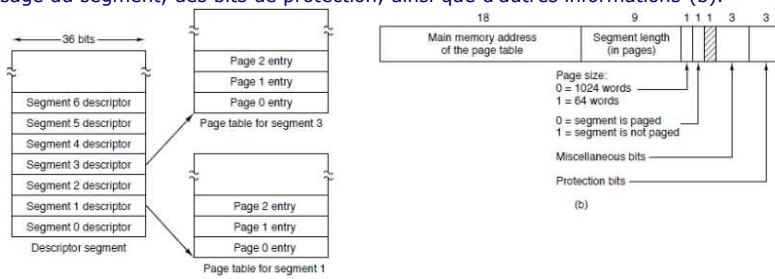
La segmentation permet aussi de faciliter le partage de zones mémoire entre plusieurs processus. L'implémentation d'un système utilisant la segmentation pure (sans pagination) pose le problème de la fragmentation de la mémoire. La fragmentation est due au fait que les segments (contrairement aux pages) sont de taille non fixe.

# Système d'Exploitation

## LA GESTION DE LA MÉMOIRE

### La segmentation : Segmentation avec pagination

- L'apparition du système d'exploitation MULTICS dans les années soixante a permis d'introduire divers concepts novateurs encore utilisés dans les systèmes les plus récents. Parmi ces idées, on trouve le principe de la segmentation avec pagination. En combinant segmentation et pagination MULTICS combine les avantages des deux techniques en s'affranchissant des principaux défauts qu'ils ont : fragmentation de la mémoire pour la segmentation, et espace d'adressage unique pour un système utilisant un adressage virtuel paginé.
- Sous MULTICS chaque segment possède son propre espace d'adressage paginé. Ainsi chaque segment possède sa propre table de pages (a). Le descripteur de chaque segment contient, en plus de l'adresse de la table de pages qui lui est associée, la taille du segment, la taille des pages (1024 mots ou 64 mots), un indicateur permettant de paginer ou non l'espace d'adressage du segment, des bits de protection, ainsi que d'autres informations (b).



173

# Système d'Exploitation

## LA GESTION DE LA MÉMOIRE

### Gestion de la mémoire sous GNU/Linux

- Comme MULTICS, la gestion de la mémoire sous Linux est basée sur la segmentation avec pagination.

### Espace d'adressage d'un processus

- L'espace d'adressage d'un processus comprend plusieurs segments appelés régions (areas) parmi lesquels figurent typiquement :
  - un segment de code ;
  - deux segments de données, un concernant les données initialisées (à partir du fichier exécutable), et l'autre les données non initialisées ;
  - un segment de pile. D'autres segments peuvent se trouver dans l'espace d'adressage du processus, ils concernent généralement les segments des bibliothèques partagées qu'utilise le processus.

174

# Système d'Exploitation

## LA GESTION DE LA MÉMOIRE

### Gestion de la mémoire sous GNU/Linux

#### Espace d'adressage d'un processus

On peut trouver les informations concernant l'espace d'adressage d'un processus en visualisant le contenu du fichier maps du répertoire concernant ce processus dans le système de fichier /proc. Dans l'exemple qui suit, on découvre la liste des segments concernant un processus portant le numéro 1463 associé à un shell /bin/bash

```
root@debian:/proc# ps
 PID TTY TIME CMD
 1463 pts/1 00:00:00 bash
24150 pts/1 00:00:00 ps
root@debian:/proc# cat /proc/1463/maps
00400000-00500000 r-xp 00000000 08:01 650 /bin/
bash
00700000-00703000 r--p 00100000 08:01 650 /bin/
bash
00703000-0070c000 rw-p 00103000 08:01 650 /bin/
bash
0070c000-00716000 rw-p 00000000 00:00 0
00f06000-00f73000 rw-p 00000000 00:00 0 [heap]
]
7f3ca9ada000-7f3ca9ada000 r-xp 00000000 08:01 131104 /lib/
x86_64-linux-gnu/libnss_files-2.24.so
7f3ca9ada000-7f3ca9cda000 ---p 0000a000 08:01 131104 /lib/
x86_64-linux-gnu/libnss_files-2.24.so
7f3ca9cda000-7f3ca9cdb000 r--p 0000a000 08:01 131104 /lib/
```

175

# Système d'Exploitation

## LA GESTION DE LA MÉMOIRE

### Gestion de la mémoire sous GNU/Linux

#### Espace d'adressage d'un processus

Le fichier maps permet ainsi de voir les principaux attributs d'un segment :

- L'adresse de début et de fin du segment dans l'espace d'adressage virtuel.  
On voit dans cette implémentation sur processeur x86 que les adresses virtuelles sont sur 64 bits et permettent donc d'adresser 8 Go.
- Les droits d'accès au segment, le caractère p indique que le segment peut être partagé. A partir des droits on peut en déduire que le premier segment associé à /bin/bash est son segment de code, le second son segment de données (initialisé).
- Le déplacement du début de segment dans l'objet qui lui est associé (le fichier exécutable dans le cas de /bin/bash).
- Le numéro de périphérique contenant l'objet.
- Le numéro d'i-node de l'objet associé au segment (nul s'il n'existe pas).
- Le chemin de l'objet.

176

# Système d'Exploitation

## LA GESTION DE LA MÉMOIRE

### Gestion de la mémoire sous GNU/Linux

#### Gestion des pages et swap

Les tables de pages gérées par Linux sont organisées en trois niveaux :

- ❑ la table globale dont chaque entrée pointe sur une table intermédiaire ;
- ❑ les tables intermédiaires dont chaque entrée pointe sur une table de page ;
- ❑ les tables de pages dont les entrées contiennent les adresses de pages physiques.

Selon la plate forme sur laquelle Linux est implanté, le nombre de niveaux effectifs peut être plus réduit. Sur un processeur de type x86, par exemple, étant donné que les pages sont organisées en deux niveaux seulement, le noyau Linux considère que la table intermédiaire ne contient qu'une seule entrée.

Lorsqu'une page est libre elle est chaînée avec les autres pages libres (pointeurs next et prev de la structure page).

Lorsque Linux manque de mémoire, il traque des pages mémoire. Pour ce faire le processus kswapd (endormi la plupart du temps) est réveillé. Ce processus explore la liste des processus et essaye d'chasser des pages. Si une page est réservée ou verrouillée, ou si elle a été récemment accédée, elle n'est pas chassée. Dans le cas contraire, son état est testé. Si la page a été modifiée, elle doit être sauvegardée sur disque avant d'être libérée (retirée de la mémoire donc du cache).

Ce mécanisme est utilisé également par Linux (depuis la version 2.0) pour les lectures de fichiers.

177

# Système d'Exploitation

## LA GESTION DE LA MÉMOIRE

### Comment afficher l'utilisation mémoire et mémoire libre sur Linux

#### Les utilitaires top

Pour surveiller l'utilisation mémoire, CPU comme souvent, il existe de nombreux utilitaires : **htop**, **atop**, **nmon**, **vtop**, ...

Mais il affiche tous les mêmes informations puisqu'elles proviennent du système.

| top - 13:09:24 up 3:18, 1 user, load average: 0,00, 0,03, 0,01          |      |    |     |        |       |       |   |      |      |         |            |
|-------------------------------------------------------------------------|------|----|-----|--------|-------|-------|---|------|------|---------|------------|
| Tasks: 103 total, 1 running, 102 sleeping, 0 stopped, 0 zombie          |      |    |     |        |       |       |   |      |      |         |            |
| %Cpu(s): 1,7 us, 1,0 sy, 0,0 ni, 97,4 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 s |      |    |     |        |       |       |   |      |      |         |            |
| KiB Mem : 1020368 total, 709904 free, 176484 used, 133980 buff/cache    |      |    |     |        |       |       |   |      |      |         |            |
| KiB Swap: 1046524 total, 1046524 free, 0 used. 703976 avail Mem         |      |    |     |        |       |       |   |      |      |         |            |
| PID                                                                     | USER | PR | NI  | VIRT   | RES   | SHR   | S | %CPU | %MEM | TIME+   | COMMAND    |
| 427                                                                     | root | 20 | 0   | 345800 | 58928 | 29552 | S | 1,0  | 5,7  | 0:36.14 | Xorg       |
| 29112                                                                   | root | 20 | 0   | 44888  | 3628  | 3024  | R | 0,7  | 0,4  | 0:00.04 | top        |
| 29107                                                                   | root | 20 | 0   | 502488 | 34948 | 27148 | S | 0,3  | 3,4  | 0:00.19 | xfce4-ter+ |
| 1                                                                       | root | 20 | 0   | 138920 | 6848  | 5352  | S | 0,0  | 0,7  | 0:01.26 | systemd    |
| 2                                                                       | root | 20 | 0   | 0      | 0     | 0     | S | 0,0  | 0,0  | 0:00.00 | kthreadd   |
| 3                                                                       | root | 20 | 0   | 0      | 0     | 0     | S | 0,0  | 0,0  | 0:00.50 | ksoftirqd+ |
| 5                                                                       | root | 0  | -20 | 0      | 0     | 0     | S | 0,0  | 0,0  | 0:00.00 | kworker/0+ |
| 7                                                                       | root | 20 | 0   | 0      | 0     | 0     | S | 0,0  | 0,0  | 0:01.80 | rcu_sched  |
| 8                                                                       | root | 20 | 0   | 0      | 0     | 0     | S | 0,0  | 0,0  | 0:00.00 | rcu_bh     |
| 9                                                                       | root | rt | 0   | 0      | 0     | 0     | S | 0,0  | 0,0  | 0:00.00 | migration+ |
| 10                                                                      | root | 0  | -20 | 0      | 0     | 0     | S | 0,0  | 0,0  | 0:00.00 | lru-add-d+ |
| 11                                                                      | root | rt | 0   | 0      | 0     | 0     | S | 0,0  | 0,0  | 0:00.09 | watchdog/0 |
| 12                                                                      | root | 20 | 0   | 0      | 0     | 0     | S | 0,0  | 0,0  | 0:00.00 | cpuhp/0    |
| 13                                                                      | root | 20 | 0   | 0      | 0     | 0     | S | 0,0  | 0,0  | 0:00.00 | kdevtmpfs  |

# Système d'Exploitation

## LA GESTION DE LA MÉMOIRE

Comment afficher l'utilisation mémoire et mémoire libre sur Linux

Afficher l'état de la mémoire libre et utilisée

Pour afficher la mémoire utilisé et libre, on peut utiliser la **commande free**. Ce dernier utilise **/proc/meminfo** mentionné ci-dessous pour donner les informations de l'état de la mémoire Linux.

Les options -k, -m, -g permettent d'afficher en Ko, Mo ou Go.

Par exemple :`free -m`

```
root@debian:~# free -m
 total used free shared buff/cache available
Mem: 996 175 690 6 130 68
Swap: 1021 0 1021
root@debian:~#
```

Cela affiche la mémoire totale, la mémoire utilisé et libre et la mémoire partagée.

179

# Système d'Exploitation

## LA GESTION DE LA MÉMOIRE

Comment afficher l'utilisation mémoire et mémoire libre sur Linux

**/proc/meminfo** et **/proc/sys/vm/**

Enfin on peut avoir plus de détails sur l'organisation de la mémoire de Linux en interrogeant directement le noyaux Linux.

**/proc/meminfo** est utilisé par pour signaler la quantité de mémoire libre et utilisée (à la fois physique et swap) sur le système ainsi que la mémoire partagée et les tampons utilisés par le noyau.

Il donne aussi des **statistiques complètes sur la mémoire virtuelle**

**Linux.**

```
root@debian:~# cat /proc/meminfo
MemTotal: 1020368 kB
MemFree: 677144 kB
MemAvailable: 690472 kB
Buffers: 17692 kB
Cached: 123112 kB
SwapCached: 0 kB
Active: 220828 kB
Inactive: 49368 kB
Active(anon): 129844 kB
Inactive(anon): 6384 kB
Active(file): 90984 kB
Inactive(file): 42984 kB
Unevictable: 96 kB
Mlocked: 96 kB
SwapTotal: 1046524 kB
SwapFree: 1046524 kB
```

La configuration de la mémoire virtuelle de Linux se trouve dans **/proc/sys/vm/**

180

# Système d'Exploitation

## LA GESTION DE LA MÉMOIRE

Comment afficher l'utilisation mémoire et mémoire libre sur Linux

/proc/meminfo et /proc/sys/vm/

la commande vmstat -s permet d'afficher des statistiques détaillées

plus d'information sur VM veuillez consulter :

<https://www.kernel.org/doc/html/latest/admin-guide/sysctl/vm.html>

```
root@debian:/proc/sys/vm# ls
admin_reserve_kbytes mmap_min_addr
block_dump mmap_rnd_bits
compact_memory mmap_rnd_compat_bits
compact_unevictable_allowed nr_hugepages
dirty_background_bytes nr_hugepages_mempolicy
dirty_background_ratio nr_overcommit_hugepages
dirty_bytes nr_pdflush_threads
dirty_expire_centisecs numa_zonelist_order
dirty_ratio oom_dump_tasks
dirtytime_expire_seconds oom_kill Allocating_task
dirty_writeback_centisecs overcommit_kbytes
drop_caches overcommit_memory
extrag_frag_threshold overcommit_ratio
hugepages_treat_as_movable page-cluster
hugegetlb_shm_group panic_on_oom
laptop_mode percpu_pagelist_fraction
legacy_va_layout stat_interval
lowmem_reserve_ratio stat_refresh
max_map_count swappiness
memory_failure_early_kill user_reserve_kbytes
memory_failure_recovery vfs_cache_pressure
```

181

# Système d'Exploitation

## LA GESTION DE LA MÉMOIRE

Gestion RAM sous Linux

Utilisation de la RAM en temps réel watch -n 1 free -m

```
Every 1,0s: free -m debian: Sun Nov 19 13:32:14 2023
 total used free shared buff/cache available
Mem: 996 183 660 6 152 67
Swap: 1021 0 1021
```

watch -n 1 cat /proc/meminfo

```
Every 1,0s: cat /proc/meminfo debian: Sun Nov 19 13:34:43 2022
MemTotal: 1020368 kB
MemFree: 670764 kB
MemAvailable: 684996 kB
Buffers: 17828 kB
Cached: 124100 kB
SwapCached: 0 kB
Active: 226464 kB
Inactive: 50044 kB
Active(anon): 135032 kB
Inactive(anon): 6632 kB
Active(file): 91432 kB
Inactive(file): 43412 kB
```

182

# Système d'Exploitation

## LA GESTION DE LA MÉMOIRE

### Gestion RAM sous Linux

#### Nettoyage de la RAM non utilisée

Première méthode : `sysctl -w vm.drop_caches=3`

```
root@debian:~# sysctl -w vm.drop_caches=3
vm.drop_caches = 3
root@debian:~#
```

Seconde méthode:

```
sync && echo 3 | tee /proc/sys/vm/drop_caches
sync; echo 3 > /proc/sys/vm/drop_caches
```

```
root@debian:~# sysctl -w vm.drop_caches=3
vm.drop_caches = 3
root@debian:~# sync && echo 3 | tee /proc/sys/vm/drop_caches
3
root@debian:~# sync; echo 3 > /proc/sys/vm/drop_caches
root@debian:~#
```

**Remarque:** A savoir, ces commandes ne rendront pas le système plus rapide, n'aura aucune incidence sur la stabilité et les performances, le système sera simplement nettoyé des différents caches en RAM non utilisée par le noyau Linux.

183

# Plan : Système d'Exploitation

1. INTRODUCTION AUX SYSTEMES D'EXPLOITATION
2. LES PROCESSUS
3. LES ENTREES/SORTIES
4. LA GESTION DE LA MÉMOIRE

## 5. LES SYSTEMES DE FICHIERS



184

# Système d'Exploitation

## LES SYSTEMES DE FICHIERS

### Introduction aux fichiers Unix

- ❑ Besoin de mémoriser des informations
  - Photos, PDF, données brutes, exécutables d'applications, le système d'exploitation lui-même, etc.
- ❑ Organisation du stockage sur mémoire de masse
  - Localisation abstraite grâce à un chemin dans une arborescence
  - Unité de base = fichier
- ❑ Exemples de types de systèmes de fichiers
  - NTFS pour Windows, ext2, ext3, ext4 pour Linux, HFSX pour Mac-OS
  - FAT pour les clés USB, ISO pour les CD
  - ... autres types de systèmes de fichiers

185

# Système d'Exploitation

## LES SYSTEMES DE FICHIERS

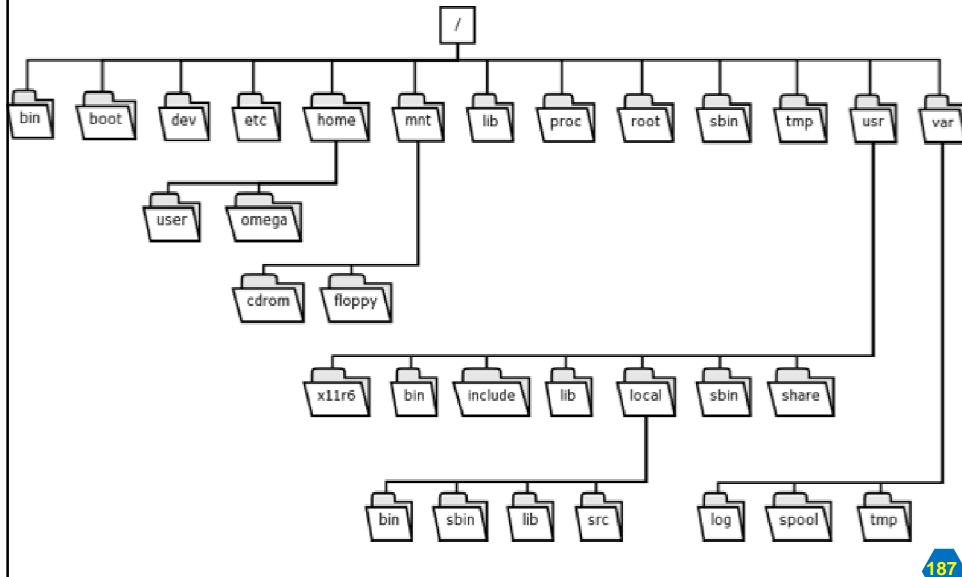
### Introduction aux fichiers Unix

- ❑ Sous les systèmes d'Exploitation UNIX/Linux tout élément est représenté sous forme de fichier. L'ensemble des fichiers est architecturé autour d'une unique arborescence dont la base, appelée racine, est notée «/».
- ❑ Il s'agit d'un Système de Fichiers Hiérarchique (Structure Arborescente: voir Fig.: Exemple d'Arborescence de Fichiers sous UNIX. ).
- ❑ Notons que, le système de fichiers est géré par le noyau. Les fichiers UNIX correspondent soit à des fichiers de données sur disque dur, soit à des répertoires, soit encore à des fichiers spéciaux, (devices) , permettant la gestion de certaines ressources du système.

186

# Système d'Exploitation

## Arborescence de Fichiers sous UNIX



187

# Système d'Exploitation

## Le concept Fichier

### Qu'est ce qu'un fichier ?

- Un **fichier** est une suite d'octets (de 0 à plus de quelques Go). Signalons qu'un Octet, dans les jargons informatiques, n'est qu'une quantité d'information de 8 bits. Pour nous, un octet est équivalent à un caractère.
- Un **fichier** est un objet abstrait à partir duquel on peut lire et écrire des informations sans se soucier de l'emplacement physique du contenu du fichier. On peut classer les fichiers selon les types suivants :

### Fichier classique

- ASCII : peut être affiché et imprimé sans modification, facile à modifier par un éditeur de texte, exemple : .txt, .html, .bat
- Binaire de données : nécessite un programme décodeur, contiennent souvent des données structurées, exemple : .doc, .excel, .gif
- Binaire exécutable : possède une structure binaire qui définit les composantes d'un programme exécutable.

### Fichier spéciaux d'E/S divisé en fichier spéciaux octet et fichier spéciaux bloc

- octet : terminaux, imprimantes, réseaux
- bloc : disque (exemple /etc/hdd1)

188

# Système d'Exploitation

## Introduction aux fichiers Unix : Types de fichiers

Les systèmes UNIX/linux définissent différents types de fichiers :

- Les **fichiers physiques**, enregistrés sur le disque dur. Il s'agit du fichier au sens où on l'entend généralement (fichiers standards);
- Les **répertoires** sont des fichiers (noeuds) de l'arborescence pouvant contenir des fichiers ou d'autres sous-répertoires. Un répertoire contient à minima un répertoire parent (noté ..), correspondant au répertoire de plus haut niveau, et un répertoire courant (noté .), c'est-à-dire lui-même (ou répertoire de travail) ;
- Les **fichiers virtuels** n'ayant pas de réelle existence physique car ils n'existent qu'en mémoire. Ces fichiers, situés notamment dans le répertoire /proc contiennent des informations sur le système (processeur, mémoire, disques durs, processus, etc.) ;
- Les **fichiers de périphériques** (ou fichiers spéciaux), situés dans le répertoire /dev/, correspondent aux périphériques du système. (par exemple, lorsqu'un périphérique d'entrées/sorties permet des opérations comme ouverture, écriture, lecture (terminal, imprimante), on y accède généralement par un fichier spécial (device) )
- Les **liens** sont des fichiers spéciaux permettant d'associer plusieurs noms (liens) à un seul et même fichier. Ce dispositif permet d'avoir plusieurs instances d'un même fichier en plusieurs endroits de l'arborescence sans nécessiter de copie, ce qui permet notamment d'assurer un maximum de cohérence et d'économiser de l'espace disque.

189

# Système d'Exploitation

## Introduction aux fichiers Unix

On distingue deux types de **liens** (links en anglais):

- Les **liens symboliques** représentant des pointeurs virtuels (raccourcis) vers des fichiers réels. En cas de suppression du lien symbolique le fichier pointé n'est pas supprimé. Les liens symboliques sont créés à l'aide de la commande ln -s selon la syntaxe suivante :  
**In -s nom-du-fichier-réel nom-du-lien-symbolique**
- Les **liens physiques** (aussi appelées liens durs ou en anglais hardlinks) représentent un nom alternatif pour un fichier. Ainsi, lorsqu'un fichier possède deux liens physiques, la suppression de l'un ou l'autre de ces liens n'entraîne pas la suppression du fichier. Plus exactement, tant qu'il subsiste au minimum un lien physique, le fichier n'est pas effacé. En contrepartie lorsque l'ensemble des liens physiques d'un même fichier est supprimé le fichier l'est aussi.
- Il faut noter toutefois qu'il n'est possible de créer des liens physiques qu'au sein d'un seul et même système de fichiers. Les liens physiques sont créées à l'aide de la commande ln , (sans option ), selon la syntaxe suivante :  
**In nom-du-fichier-reel nom-du-lien-physique**

190

# Système d'Exploitation

## Le concept Fichier(suite)

### Catalogues (répertoire)

- Lorsque le nombre de fichiers devient élevé, le système d'exploitation a besoin d'une organisation afin de structurer ces fichiers et de pouvoir y accéder rapidement. Cette organisation est effectuée par le biais de répertoires ou catalogues. Un répertoire est une entité créée pour l'organisation des fichiers. En effet on peut enregistrer un grand nombre de fichiers sur un disque dur.
- Les catalogues (**répertoire**) : fichiers spéciaux dont le contenu permet de cataloguer l'arborescence des fichiers sur le disque.  
Un fichier peut être créé de deux façons :
  1. Soit par l'utilisateur dans la plupart des cas ;
  2. Soit par le système. Certains fichiers sont générés par les systèmes ou certains outils tels que les compilateurs.
- Un répertoire est considéré comme un fichier puisqu'il est stocké sur le disque et est destiné à contenir des fichiers.

191

# Système d'Exploitation

## Le concept fichier (suite)

- Sous les systèmes d'Exploitation UNIX/Linux tout élément est représenté sous forme de fichier. L'ensemble des fichiers est architecturé autour d'une unique arborescence dont la base, appelée **racine**, est notée «/».
  - TOUT est fichier (sous Unix), y compris un périphérique.
- Il s'agit d'un Système de Fichiers Hiérarchique (Structure Arborescente: voir **Figure: Exemple d'Arborescence de Fichiers sous UNIX/Linux.** ).
- Notons que, le système de fichiers est géré par le noyau. Les fichiers UNIX correspondent soit à des fichiers de données sur disque dur, soit à des répertoires, soit encore à des fichiers spéciaux, (devices) , permettant la gestion de certaines ressources du système.

192

# Système d'Exploitation

## Les systèmes de fichiers sous Unix/Linux

Le terme **système de fichiers** (abrégé « FS » pour File System) désigne soit :

- l'organisation **hiérarchique** des **fichiers** au sein d'un système d'exploitation (on parle par exemple du file system d'une machine unix organisé à partir de sa racine (/)).
- l'organisation des fichiers au sein d'un volume physique ou logique, qui peut être de différents types (par exemple **NTFS**, **FAT**, **FAT32**, **ext2fs**, **ext3fs**, **ext4fs**, **zfs**, **btrfs**, etc.)
- De façon générale, un système de fichiers ou **système de gestion de fichiers** (SGF) est une façon de stocker les informations et de les organiser dans des **fichiers** (**mémoire de masse :disque dur**, un disque **SSD**, un **CD-ROM**, une **clé USB**, etc.). Une telle gestion des fichiers permet de traiter, de conserver des quantités importantes de **données** ainsi que de les partager entre plusieurs **programmes informatiques**. Il offre à **l'utilisateur** une vue abstraite sur ses données et permet de les localiser à partir d'un chemin d'accès.

193

# Système d'Exploitation

## Les systèmes de fichiers sous Unix/Linux

- Contrairement au système de fichiers Windows, il n'existe pas de lecteurs A:, C:, etc.
- L'entrée du système de fichier se situe à la racine, notée /.
- Ensuite, il existe un certain nombre de répertoires présents par défaut. Chaque répertoire a un rôle bien précis, comme indiqué dans le tableau ci-dessous.
- Les systèmes de fichiers les plus courants sont la fat (disquettes et clefs usb), ntfs (Windows), Ext2, Ext3 et Ext4 (Linux), iso 9660 (cd) et udf (dvd).

### Où est stocké le système de fichiers ?

- Un système de fichiers exploite des blocs, pas des secteurs. Les blocs du système de fichiers sont des groupes de secteurs qui optimisent l'adressage du stockage. Les systèmes de fichiers modernes utilisent généralement des tailles de bloc de 1 à 128 secteurs (512-65536 octets). Les fichiers sont généralement stockés au début d'un bloc et occupent des blocs entiers.

### Systèmes de fichiers et systèmes d'exploitation associés ou compatibles

- Le choix du système de gestion des fichiers se fait principalement en fonction du système d'exploitation. Généralement, les systèmes d'exploitation les plus récents supportent un grand nombre de systèmes de fichiers.

194

# Système d'Exploitation

## Les systèmes de fichiers sous Unix/Linux

### Systèmes de fichiers et systèmes d'exploitation associés ou compatibles

- Le choix du système de gestion des fichiers se fait principalement en fonction du système d'exploitation. Généralement, les systèmes d'exploitation les plus récents supportent un grand nombre de systèmes de fichiers.

### Non journalisés

- ext et ext2 : Extented FS version 2 (Linux, BSD, Windows via un pilote tiers)
- exFAT : Extended File Allocation Table (nouveau système de fichiers proposé par Microsoft pour remplacer la FAT sur les supports amovibles)
- FAT : File Allocation Table (DOS/Windows, Linux, BSD, OS/2, Mac OS X). Se décompose en plusieurs catégories (FAT12 ; FAT16 ; FAT32 ; VFAT ;
- HFS : Hierarchical File System (Mac OS, Mac OS X, Linux)

195

# Système d'Exploitation

## Le système de fichiers ext (Extended file system)

L'**extended file system** ou **ext**, est le premier système de fichiers créé en avril 1992 spécifiquement pour le système d'exploitation **Linux**, il est intégré dès la version 0.96c du noyau Linux. Il a été conçu par Rémy Card pour surmonter certaines limitations du système de fichiers Minix. Il a plus tard été remplacé par ext2 et Xiafs, tous deux en compétition, laquelle a été remportée par ext2 grâce à sa viabilité sur le long terme.

Linux possède donc son système appelé **ext (ext1, ext2, ext3, ext4 ...)** mais peut en gérer d'autres.

L'utilisateur peut accéder sous Linux à d'autres systèmes de fichiers, comme **NTFS**, DOS, Vfat,... provenant d'un périphérique ou importé par le réseau.

Comme pour l'utilisateur tout est fichier, tous les systèmes de fichiers quels que soient leur emplacement physique doivent être intégrés dans l'UNIQUE arborescence logique du système Linux.

Cette arborescence peut donc être construite (et évoluer) à partir de diverses partitions qui peuvent être situées sur plusieurs disques. Cela réalise une intégration et une abstraction plus poussée que dans le monde Windows où les partitions et lecteurs auxquels sont affectées les lettres A: C: D: ... demeurent des entités séparées.

Naturellement la partition sur laquelle est situé le répertoire racine joue un rôle important.

196

# Système d'Exploitation

## Les systèmes de fichiers sous Unix/Linux

### Journalisés

Les systèmes de fichiers journalisés (enregistrent les modifications dans un journal avant de les effectuer sur les fichiers eux-mêmes). Ce mécanisme apporte une plus grande fiabilité, en permettant une récupération des modifications "en cours" en cas d'arrêt intempestif (coupe de courant, plantage système, ...).

- ext3 : Extented FS version 3 - (Linux, BSD)
- ext4 : Extented FS version 4 - (Linux >=2.6.28)
- HFS+ (Mac OS X, Linux)
- **NTFS** : New Technology FileSystem (Windows NT/XP/7/8/10/11, Linux et Mac OS X ) (NTFS est le système de fichiers principal des versions récentes de Windows et Windows Server) (Windows Server 2003, Windows Server 2008, Windows Server 2012, Windows Server 2016, Windows Server 2019 (October 2018) et Windows Server 2022)
- Remarque : Pour plus de détail voir le lien suivant : ([https://fr.wikipedia.org/wiki/Syst%C3%A8me\\_de\\_fichiers](https://fr.wikipedia.org/wiki/Syst%C3%A8me_de_fichiers))

### **Le système de fichiers NTFS (New Technology File System)**

Dans NTFS, l'organisation des fichiers est optimisée sur la partition, d'où une très faible fragmentation. NTFS respecte la norme POSIX. et permet une bonne gestion des disques de grande taille. Il prend également en charge le modèle de sécurité des systèmes Windows.

197

# Système d'Exploitation

## Les systèmes de fichiers sous Unix/Linux

**Le système de fichiers ext2fs** : Il définit quatre catégories de fichiers :

- . les fichiers normaux (textes, exécutables) ;
- . les fichiers répertoires ;
- . les fichiers spéciaux, contenus dans le répertoire /dev, qui sont des points d'accès aux périphériques ;
- . les fichiers liens symboliques qui font référence à d'autres fichiers.

**Ses principales caractéristiques sont les suivantes :**

- . répertoires séparés par des / ;
- . sensible à la casse (aaa <> AAA) ;
- . Fichiers cachés débutant par un « . ».
- . espaces et noms longs acceptés ;
- . défragmentation quasi-inutile.

198

# Système d'Exploitation

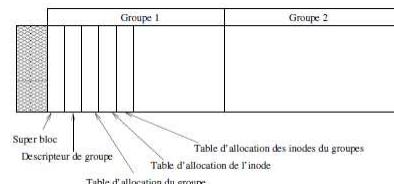
## Les systèmes de fichiers sous Unix/Linux

### Le système de fichiers ext2fs

Une partition ext2fs comporte un secteur d'amorçage et un ensemble de groupes de secteurs de même taille.

Le secteur d'amorçage est le premier sur la partition, est noté secteur 0 et a une taille de 1 Ko. Chacun des groupes de secteurs comprend six parties :

- un super bloc, contenant les informations de structure de la partition ;
- la liste des descripteurs de groupe permettant de localiser sur le disque les informations de chaque groupe ;
- la table d'allocation des blocs du groupe ;
- la table d'allocation des inodes ;
- la table d'allocation des inodes du groupe ;
- les blocs de données.



Les groupes d'ext2fs

199

# Système d'Exploitation

## Les systèmes de fichiers sous Unix/Linux

- **ext3** est un système de fichiers utilisé par GNU/Linux. C'est une évolution de **ext2**, le précédent système de fichiers utilisé par défaut par de nombreuses distributions GNU/Linux.
- **ext3** est une évolution de **ext2** et a pour principale différence l'utilisation d'un fichier journal, lui permettant ainsi d'éviter la longue phase de récupération lors d'un arrêt brutal de la machine.

**ext4 est le successeur du système de fichiers ext3, principalement destiné aux systèmes basés sur GNU/Linux.**

Outre le fait qu'il puisse gérer les volumes d'une taille allant jusqu'à un exaoctet ( $2^{60}$  octets), la fonctionnalité majeure de **ext4** est l'allocation par extent qui permettent la pré-allocation d'une zone contiguë pour un fichier, pour minimiser la fragmentation. L'option extent est active par défaut depuis le noyau Linux 2.6.23 ;

Le système de fichiers **ext4** a une compatibilité ascendante avec **ext3**. C'est-à-dire qu'une partition **ext3** peut toujours être montée comme **ext4**.

### Accès depuis Windows et Mac OS

Le contenu des partitions ou fichiers images de partitions formatés en **ext4** sont accessibles en lecture seule sous Windows en utilisant le logiciel libre **ext2read6** (**ext2explore**) ou **LinuxReader**. Mac OS X supporte les systèmes de fichiers **ext2/3/4** en lecture/écriture à travers le logiciel **Paragon ExtFS**.

200

# Système d'Exploitation

## Les systèmes de fichiers sous Unix/Linux

### Répertoires et hiérarchie des fichiers



- Les fichiers sont organisés en répertoires et sous-repertoires, formant une arborescence (voir Figure: Exemple d'Arborescence de Fichiers sous UNIX, autre exemple type que celui qu'a donné précédemment )
- Dans chaque répertoire, on trouve au moins deux répertoires, nommés . (point) et .. (deux points). Où le premier (répertoire point) permet de référencer le répertoire courant ou le répertoire lui-même, alors que le deuxième (répertoire ..) permet d'accéder au répertoire parent (répertoire du dessus).
- Signalons qu'à chaque instant, toute tâche possède un répertoire de travail ou répertoire courant.
- Pour afficher ce répertoire de travail (répertoire courant) on utilise la commande **pwd**.  

```
root@debian:~# pwd
/root
root@debian:~#
```
- Pour changer le répertoire de travail, il suffit d'utiliser la commande **cd**.

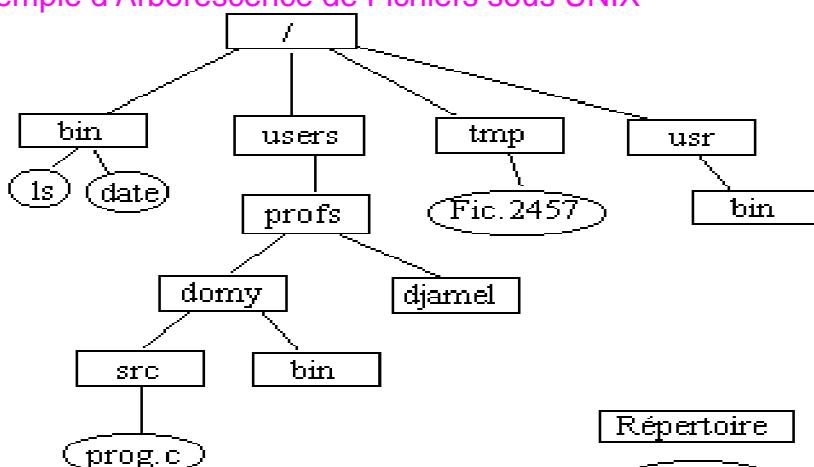
201

# Système d'Exploitation

## LES SYSTEMES DE FICHIERS

### Répertoires et hiérarchie des fichiers

#### Exemple d'Arborescence de Fichiers sous UNIX



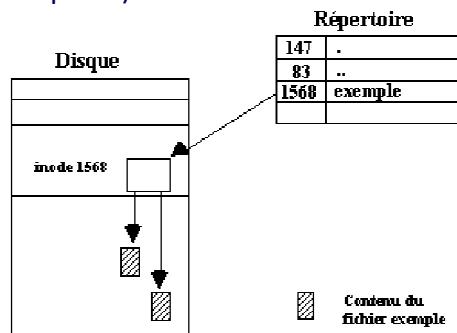
202

# Système d'Exploitation

## Répertoires et hiérarchie des fichiers (suite):

- Un répertoire UNIX/Linux est un catalogue, c'est-à-dire une correspondance entre un nom de fichier et son numéro d'inode (descripteur).

- La création d'un répertoire s'effectue par la commande :  
**mkdir nom\_du\_reperatoire**



- Le répertoire ainsi créé contient 2 entrées :
  - le répertoire parent (celui dans lequel il a été créé) noté ".."
  - le répertoire lui-même noté "."

203

# Système d'Exploitation

## Architecture du système Unix/Linux

- Un **répertoire** n'est rien d'autre qu'un fichier contenant une liste de couples formés d'une chaîne de caractères (appelée **lien physique** ou "nom de fichier") et d'un nombre (numéro de l'i-node décrivant le fichier).
- La désignation d'un fichier est ainsi réalisée en utilisant un lien associé à ce fichier dans un répertoire, ce répertoire étant lui-même désigné par le même mécanisme à partir d'un autre répertoire.
- L'organisation qui s'ensuit est donc arborescente et suppose, pour être opérationnelle, l'existence d'une origine symbolique. Cette origine est appelée racine absolue du système de gestion de fichiers noté (" / ").

## Notion de point de montage

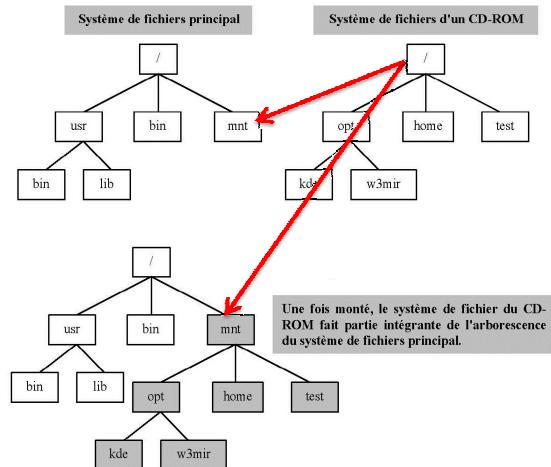
- Les fichiers d'un système UNIX sont organisés dans une arborescence unique. Il est toutefois possible d'avoir plusieurs partitions grâce à un mécanisme appelé **montage**, permettant de raccorder une partition à un répertoire de l'arborescence principale. Ainsi le fait de monter une partition dans le répertoire /mnt/partition rendra l'ensemble des fichiers de la partition accessible à partir de ce répertoire, appelé "point de montage".

204

# Système d'Exploitation

## Architecture du système Unix/Linux

- Sous Windows (NTFS, FS), chaque système de fichiers porte un nom bien précis (Exemple C; D; ...)
- Sous UNIX/Linux, tous les systèmes de fichiers utilisés viennent se « rattacher » (on dit se monter) sur le système de fichiers principal (sur lequel on a booté), il y a hiérarchisation . Exemple:

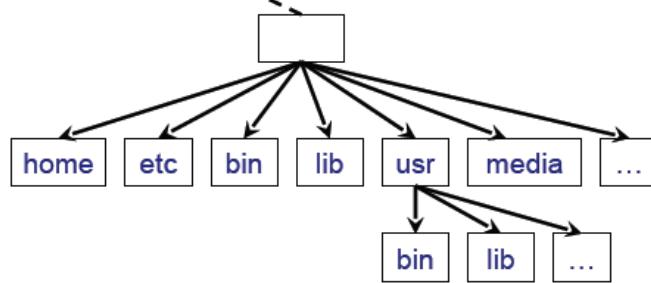


205

# Système d'Exploitation

## Les systèmes de fichiers sous Unix/Linux Arborescence standard des systèmes d'exploitation UNIX

La racine est référencée par le nom vide

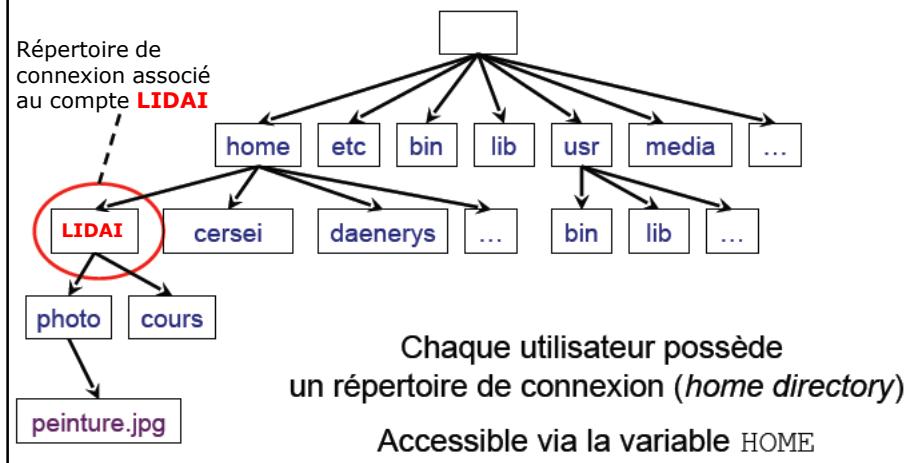


La plupart des systèmes d'exploitation Unix (GNU/Linux, BSD, MacOS...) utilisent une arborescence de base standardisée (seul Windows utilise une arborescence réellement différente)

206

# Système d'Exploitation

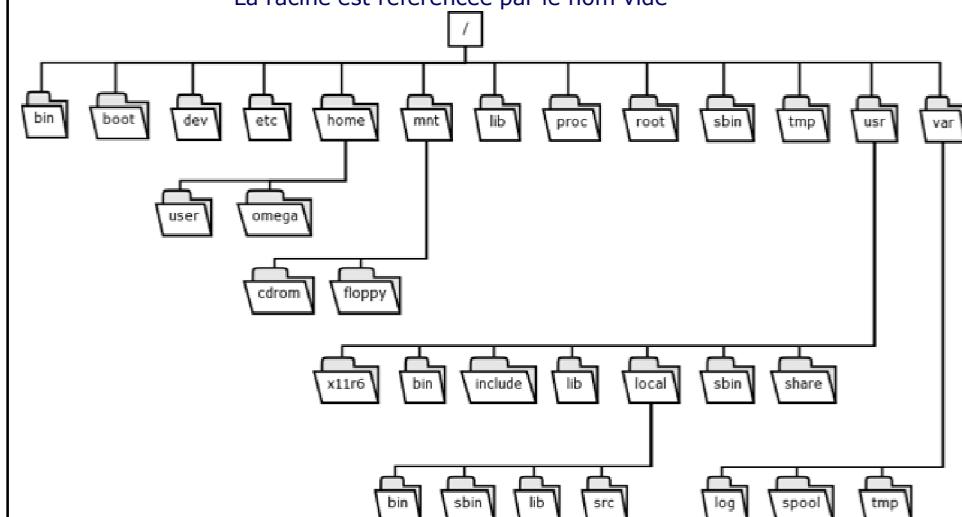
## Les systèmes de fichiers sous Unix/Linux Arborescence standard des systèmes d'exploitation UNIX



# Système d'Exploitation

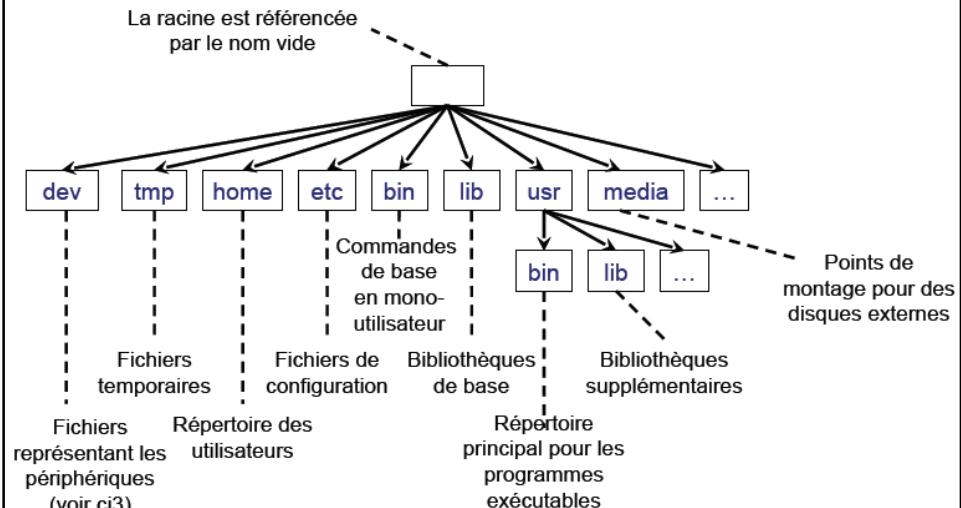
## Les systèmes de fichiers sous Unix/Linux Arborescence standard des systèmes d'exploitation UNIX

La racine est référencée par le nom vide



# Système d'Exploitation

## Les systèmes de fichiers sous Unix/Linux Arborescence standard des systèmes d'exploitation UNIX



# Système d'Exploitation

## Architecture du système Unix/Linux

### Hiérarchie des fichiers/ répertoires sous Unix

Pour assurer la compatibilité et la portabilité, les systèmes UNIX respectent la norme FHS (File Hierarchy Standard). La hiérarchie de base d'un système Unix est la suivante :

**Dans les systèmes basés sur Unix, le système de fichiers contient les répertoires de bases suivant :**

- **/** : la racine, elle contient les répertoires principaux; contient les outils et fichiers de configuration vitaux pour le système.
- **/home** : les répertoires des comptes utilisateurs (destinés à accueillir les fichiers des utilisateurs du système).
- **/bin** : Contient les exécutables essentiels au système, employés par tous les utilisateurs.
- **/boot (noyau)**: Contient les fichiers de chargement du noyau, dont le chargeur d'amorce ou de redémarrage.
- **/dev** : Contient les points d'entrée des périphériques.
- **/etc** : Contient les fichiers de configuration nécessaires à l'administration du système (fichiers *passwd*, *group*, *inittab*, *ld.so.conf*, *lilo.conf*, ...).
- **/lib** : Contient les bibliothèques standards partagées entre les différentes applications du système.

210

# Système d'Exploitation

## Architecture du système Unix/Linux

### Hiérarchie des fichiers/ répertoires sous Unix

- **/mnt** : Permet d'accueillir les points de montage des partitions temporaires (cd-rom, disquette, USB, ...).
- **/proc** : Regroupe un ensemble de fichiers virtuels permettant d'obtenir des informations sur le système ou les processus en cours d'exécution.
- **/root** : Répertoire personnel de l'administrateur root. Le répertoire personnel de l'administrateur est situé à part des autres répertoires personnels car il se trouve sur la partition racine, afin de pouvoir être chargé au démarrage, avant le montage de la partition /home.
- **/sbin** : Contient les exécutables système essentiels (par exemple la commande adduser).
- **/tmp** : contient les fichiers temporaires;
- **/usr** : toutes les applications dont celles utilisateurs.
- **/usr** : Hiérarchie secondaire;
  - /usr/X11R6 : ce répertoire est réservé au système X version 11 release 6;
  - /usr/X386 : utilisé avant par X version 5, c'est un lien symbolique vers /usr/X11R6;
- **/usr/bin** : contient la majorité des fichiers binaires et commandes utilisateurs;

211

# Système d'Exploitation

## Architecture du système Unix/Linux

### Hiérarchie des fichiers/ répertoires sous Unix

- **/usr/include** : contient les fichiers d'en-tête pour les programmes C et C++;
- **/usr/lib** : contient la plupart des bibliothèques partagées du système;
- **/usr/local** : contient les données relatives aux programmes installés sur la machine locale par le root;
  - /usr/local/bin : Binaires des programmes locaux;
  - /usr/local/include : Fichiers d'en-tête C et C++ locaux;
  - /usr/local/lib : Bibliothèques partagées locales;
  - /usr/local/sbin : binaires système locaux;
  - /usr/local/src : Fichiers sources locaux;
  - /usr/sbin : contient les fichiers binaires non essentiels au système réservés à l'administrateur système;
  - /usr/share : réservé aux données non dépendantes de l'architecture;
  - /usr/src : contient des fichiers de code source;
- **/var** : contient des données versatiles telles que les fichiers de bases de données, les fichiers journaux (logs), les fichiers du spouleur d'impression ou bien les mails en attente
- ..., etc.

212

# Système d'Exploitation

## Arborescence & Chemins d'Accès (Absolus ou Relatifs)

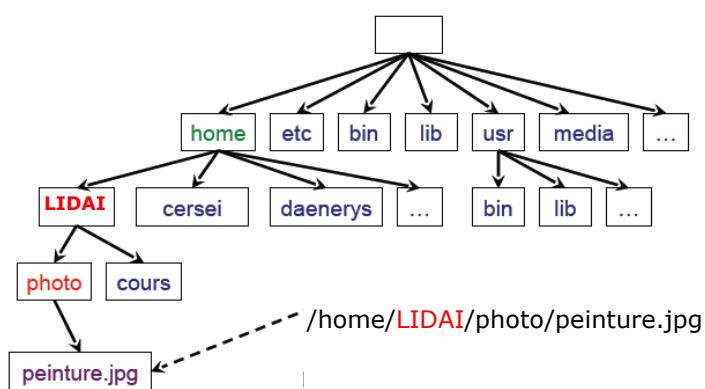
### Chemins absolus et relatifs:

- ❑ Le chemin d'accès à un fichier ou répertoire, constitué d'une suite de répertoires, séparés par des "/".
- ❑ En bash, le séparateur de répertoires est le caractère /
- ❑ Un chemin s'écrit sous la forme a/b/c qui référence
  - >le fichier c ;
  - >se trouvant dans le répertoire b ;
  - >se trouvant lui même dans le répertoire a
- ❑ On peut référencer chaque élément de l'arborescence de deux façons :
  1. par son **chemin absolu** (par rapport à la racine). Un chemin absolu spécifie la suite des répertoires à traverser en partant de la racine, séparés par des caractères /. (*Exemple* : **/users/profs/younes/src/prog.c** Un chemin absolu part de la racine du système de fichiers)
  2. par son **chemin relatif** (par rapport au répertoire courant ou de travail), c'est-à-dire par rapport à l'endroit où l'on se trouve. Tout chemin qui ne commence pas par un caractère / (prononcé **slash**) est interprété comme un chemin relatif (**short pathname**) au répertoire courant. On peut ainsi accéder aux fichiers du répertoire courant en donnant simplement leur nom.
    - >Un chemin relatif part du répertoire de travail du processus

213

# Système d'Exploitation

## Exemple de chemin absolu (1 /2)

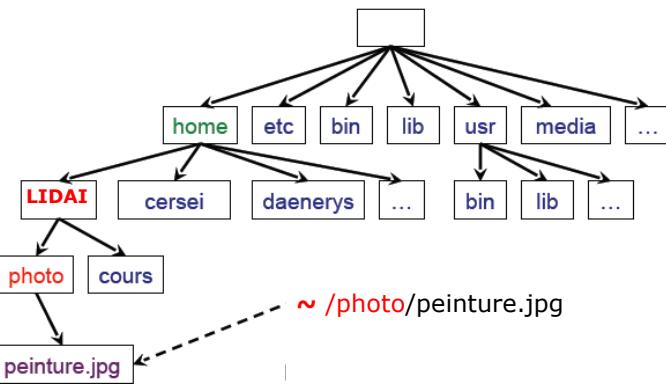


214

# Système d'Exploitation

## Exemple de chemin absolu (2/2)

Un utilisateur peut utiliser ~ comme raccourci pour son répertoire de connexion



Remarque : un utilisateur peut référencer le répertoire de connexion d'un autre utilisateur avec ~name (par exemple ~LIDAI/photo/peinture.jpg)

215

# Système d'Exploitation

## Arborescence & Chemins d'Accès (Absolus ou Relatifs)

Une arborescence UNIX comporte différents types d'éléments :

- des fichiers ordinaires (suite d'octets)
- des répertoires (catalogues)
- des liens symboliques (alias)
- des fichiers dits spéciaux (périphériques et outils de communication),...,etc.

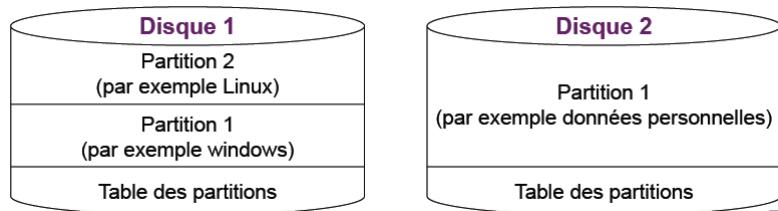
mais l'utilisateur en a une **vision uniforme** (mêmes protections, mêmes commandes, même syntaxe...) grâce à la notion d'inode (descripteur associé à chaque entrée et contenant le nom du propriétaire, les droits d'accès, différentes dates...)

216

# Système d'Exploitation

## Organisation des disques

- ❑ Une machine peut posséder plusieurs disques
- ❑ Et chaque disque peut être scindé en plusieurs partitions
  - Utile pour installer plusieurs systèmes d'exploitation ou pour augmenter l'indépendance entre les données utilisateurs et le système d'exploitation
  - Chaque partition possède son système de fichiers indépendant

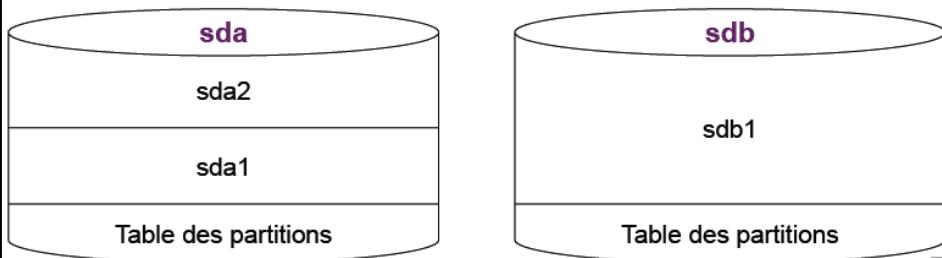


217

# Système d'Exploitation

## Les partitions dans les systèmes UNIX

- ❑ Un disque est identifié par le préfixe sd (*scsi drive*)
- ❑ Les disques sont numérotés a, b, c...
- ❑ Les partitions sont numérotées 1, 2, 3...  
(vous pouvez voir les disques/partitions en faisant ls /dev)



218

# Système d'Exploitation

## Le système de fichiers sur disque (1/2)

### ❑ 3 concepts fondamentaux

- Le bloc : unité de transfert entre le disque et la mémoire  
(souvent 4096 octets)

### ❑ L'inode (*index node*) : descripteur d'un fichier

- Type de l'inode (fichier ordinaire, répertoire, autres)
- Propriétaire, droits, dates de création/modification'accès
- Taille
- Liste des blocs du contenu du fichier
- ...

### ❑ fichier = inode + blocs du fichier

219

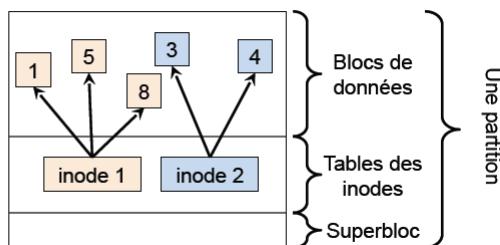
# Système d'Exploitation

## Le système de fichiers sur disque (2/2)

### ❑ Avec ext, utilisé sous GNU/Linux, trois zones principales

- Le superBloc, au début, décrit les autres zones
- La table des inodes contient les inodes (inode 0 = racine)
- La zone des blocs de données contient les données des fichiers

Par exemple,  
contenu de inode 1 :  
4096 octets du bloc 1 puis  
4096 octets du bloc 5 puis  
312 octets du bloc 8

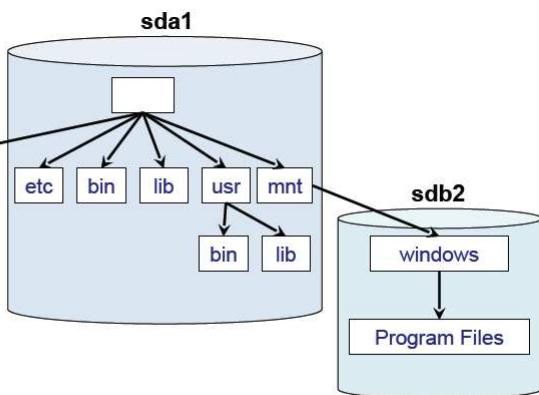
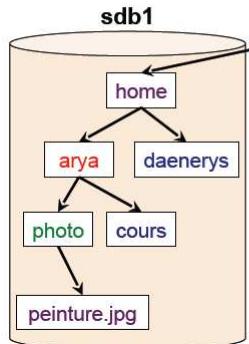


220

# Système d'Exploitation

## Montage d'une partition

Exemple de points de montage



221

# Système d'Exploitation

## Remarques sur le Système de fichiers

- La structure de données d'UNIX se présente sous la forme d'un **arbre inversé** contenant :
  - un **point d'entrée** appelé racine et noté / ;
  - des répertoires (noeuds) pouvant contenir d'autres noeuds et/ou des feuilles ;
  - des fichiers qui constituent les **feuilles** de l'arbre.
- Le caractère / a un double rôle :
  - au début du nom absolu, il symbolise la racine ( / = root).
  - ailleurs, il sert de séparateur de répertoires.
- Il découle de ce qui précède qu'un répertoire Unix n'est jamais vide, puisqu'il contient toujours au moins les deux liens . et .. qui ne peuvent être détruits.
- Notons aussi que chaque utilisateur possède un répertoire particulier que nous appellerons son **répertoire privé, ou répertoire personnel**, qui devient automatiquement son répertoire de travail lors de sa connexion au système. Le symbole ~ permet de faire référence, de manière abrégée, à ce répertoire.
- Pour faire référence au répertoire privé (ou répertoire personnel) d'un autre utilisateur on place son nom après le tilde (~) : ~ login. (exemple : ~ Said).
- certains systèmes de fichiers (par exemple: zfs) peuvent avoir une racine mais peuvent avoir plusieurs : ils permet l'existence de volumes virtuels (datasets sous zfs), qui partagent un même espace physique, mais peuvent être montés à des endroits différents d'une même hiérarchie de fichiers, ce qui permet la déduplication entre des volumes différents et a l'avantage de ne pas imposer à chaque volume virtuel de réservé un espace fixe.

222

# Système d'Exploitation

## Les i-nodes

A tout fichier est attaché un nœud d'index, ou **i-node** contenant des informations sur ce fichier. C'est une structure de quelques dizaines d'octets décrite dans `/usr/include/sys/inode.h` qui contient généralement les champs suivants :

- l'identification du propriétaire et du groupe propriétaire du fichier ;
- le type (fichier ordinaire, spécial, catalogue,...),
- les protections (ou les droits d'accès des différents utilisateurs ),
- les dates de lecture, modification du fichier et modification de l'inode ;
- l'adresse des blocs utilisés pour ce fichier sur le disque ;
- la taille du fichier en octets,
- le nombre de liens physiques (un lien d'un fichier est un autre nom de ce fichier),
- les éléments d'identification du propriétaire et de son groupe,
- l'adresse physique d'implantation sur disque sous forme des adresses de blocs disques.
- l'identification de la ressource associée (fichiers spéciaux).

**Remarque:** Les **noms de fichiers** peuvent contenir n'importe quel caractère (à l'exception du / et - ne peut pas être le premier caractère). Le caractère point (.) n'a pas de signification particulière. Ainsi, **toto.c.bak.old** est un nom acceptable.

223

# Système d'Exploitation

## Protections et Droits d'accès

### Les utilisateurs, l'accès au système

- Chaque utilisateur humain du système doit disposer d'un *compte* protégé par un mot de passe. Lorsque l'on se connecte à la machine, on doit fournir son nom et son mot de passe. Le nom est un pseudonyme attribué une fois pour toute à un utilisateur par l'administrateur du système. Le mot de passe peut être modifié par l'utilisateur aussi souvent qu'il le désire.
- Avec la généralisation des interfaces graphiques, l'accès à un système sous UNIX s'est diversifié et (théoriquement) simplifié. La procédure d'entrée d'un utilisateur dans le système se nomme le **login**. La sortie est donc le **logout**. Lors du login, l'utilisateur devra toujours fournir son **nom d'utilisateur** et **un mot de passe**. Après vérification du mot de passe, le système lance un interpréteur de commande (shell).
- Chaque utilisateur dispose de ses propres fichiers, dont il peut autoriser ou non l'accès aux autres utilisateurs. Il dispose d'un certain nombre de *droits* (accès à certains périphériques, etc).
- Il peut lancer l'exécution de **processus** (le nombre maximal de processus par utilisateur peut être limité sur certains systèmes).

**Les processus lancés par un utilisateur ont les mêmes droits d'accès que lui.**

224

# Système d'Exploitation

## Protections et Droits d'accès

### Connexion et Déconnexion

#### LogIn:

- L'accès au système se fait par la donnée :
- d'un nom d'utilisateur créé par l'administrateur système (ouverture ou création d'un Compte);
- d'un mot de passe :
  - Strictement personnel
  - Garantissant la confidentialité de vos informations
  - Empêche les intrusions en votre nom
  - S'attribuer ou changer de mot de passe à l'aide de la commande : `passwd`
- Après le démarrage d'une session Shell, l'invite de commande (prompt) indique la disponibilité du Shell.

#### LogOut:

Fin de l'utilisation du Shell : Ctrl-d ou (logout ou exit selon le Shell)

225

# Système d'Exploitation

## Protections et Droits d'accès

- Sous Unix/Linux, tout utilisateur possède une identification (**uid**) et appartient à un groupe particulier (**gid**).
  - Toute opération sur un fichier est soumise à droits d'accèsMessage d'erreur « Permission non accordée »
  - Aussi, pour un fichier/répertoire donné, les utilisateurs du système peuvent être classés en trois catégories :
    - le propriétaire du fichier, "**user**", (**u**);
    - les membres du groupe propriétaire du fichier, "**group**", (**g**);
    - les autres utilisateurs (reste des utilisateurs enregistrés sur les machines de réseau), "**others**", (**o**);
  - Tous les utilisateurs, "**all**", (**a**) ou (**u g o**).
- **Trois types d'opérations élémentaires sont contrôlées par le système :**
- la lecture (**r**): lire un fichier / lister un répertoire ;
  - l'écriture (**w**): écrire un fichier / ajouter ou supprimer des fichiers dans le répertoire;
  - et l'exécution (**x**): exécuter un fichier / se déplacer dans le répertoire.

Il y a donc neuf combinaisons possibles, qui sont codées en utilisant 9 bits de l'i-node du fichier. Ces droits (ou permissions) d'accès ne peuvent être changés que par le propriétaire du fichier (ou par son groupe d'utilisateurs ou tous autres s'ils ont les droits d'accès), grâce à la commande **chmod**.

226

# Système d'Exploitation

## Protections et Droits d'accès

- Ce masque de protection est affiché par la commande ls -l sous la forme de neuf lettres : **rwxrwxrwx**, les trois premières correspondent aux permissions de l'utilisateur propriétaire, les trois suivantes aux permissions du groupe propriétaire et les trois dernières aux autres utilisateurs. Si une permission n'est pas accordée, la lettre correspondante est remplacée par un tiret (-) : **rw-r-x---** est un masque de permission assez courant qui signifie que le propriétaire a tous les droits, le groupe propriétaire (les utilisateurs du même groupe) a les droits de lecture et d'exécution et les autres utilisateurs n'ont aucun droit (c'est-à-dire: pas du tout aux autres utilisateurs).

### Le Super-utilisateur:

Afin de permettre l'administration du système, un utilisateur spécial, appelé super-utilisateur, (ou administrateur ou root), est toujours considéré par le système comme propriétaire de tous les fichiers/ répertoires (et aussi des processus).

Il est à signaler que, la personne qui gère le système est normalement la seule à connaître son mot de passe. C'est le seul, normalement, qui peut ajouter des nouveaux utilisateurs au système!

227

# Système d'Exploitation

## Les protections des fichiers (Droits d'accès aux fichiers)

- A la création d'un compte d'utilisateur, l'administrateur affecte celui-ci à un groupe. Chaque groupe possède un nom. Un utilisateur peut changer de groupe par la commande **newgrp**.  
On peut donner des droits d'accès particuliers à certains fichiers pour les membres d'un même groupe.
- Au total, il existe 3 types d'accès : propriétaire (**u** = user), groupe (**g** = group), autres (**o** = other). L'ensemble est désigné par **a** (all).
- L'expression des droits nécessite 12 bits :
- Pour un fichier, **r** = droit de lecture, **w** = droit d'écriture, **x** = droit d'exécuter (1 = oui, 0 = non)
- Pour un répertoire,
  - r** : on peut consulter la liste des fichiers qui y sont contenus
  - w** : on peut créer ou effacer des fichiers à l'intérieur
  - x** : on peut utiliser ce répertoire en argument d'une commande et s'y positionner
- Ces 9 bits sont précédés par 3 autres bits pour compléter la description des protections :

228

# Système d'Exploitation

# Architecture du système Unix/Linux

## **Les protections des fichiers (Droits d'accès aux fichiers)**

Les droits des fichiers d'un répertoire peuvent être affichés par la commande **ls -l**. Les droits d'accès apparaissent alors comme une liste de 10 symboles. :

**drwxr-xr-x** Le premier symbole peut être « - », « d », soit « l », entre autres . Il indique la nature du fichier :

- - : fichier ordinaire
  - d : répertoire
  - l : lien symbolique
  - c : périphérique de type caractère
  - b : périphérique de type bloc
  - p : pipe (FIFO) "tube" ou "tuyau" en anglais ou pipeline aussi en français
  - s : socket

Suivent ensuite 3 groupes de 3 symboles chacun, indiquant si le fichier (ou répertoire) est autorisé en lecture, écriture ou exécution. Les 3 groupes correspondent, dans cet ordre, aux droits du propriétaire, du groupe puis du reste des utilisateurs. Ce sont ces lettres qui sont utilisées pour symboliser les dites permissions. Si la permission n'est pas accordée, la lettre en question est remplacé par « - ». Si l'on reprend les lettres données pour lecture/écriture/exécution (read/write/execute), nous obtenons : **rwx<sup>229</sup>**

# Système d'Exploitation

Architecture du système Unix/Linux

## **Les protections des fichiers (en octal)**

# Système d'Exploitation

## Les protections des fichiers (Droits d'accès aux fichiers)

- le bit "set-uid" : quand il vaut 1 pour un fichier exécutable, le processus d'exécution a les droits du propriétaire du fichier et non de l'utilisateur qui le lance
- le bit "set-gid" : même rôle, mais relativement au groupe
- l'attribut spécifique : outre la valeur "-" pour les fichiers ordinaires, il peut prendre la valeur :
  - - : fichier classique
  - d : répertoire
  - l : lien symbolique
  - s : socket
  - ...
- La commande **ls (list)** avec l'option **I (long)**, (**ls -l**), permet d'afficher les protections de fichiers (ou les droits d'accès à un fichier). Alors que, la commande **chmod** permet de modifier les protections (les droits d'accès) d'un fichier.

231

# Système d'Exploitation

## Les protections des fichiers (Droits d'accès aux fichiers) Droits d'accès en octal

En octal, chaque « groupement » de droits (pour user, group et other) sera représenté par un chiffre et à chaque droit correspond une valeur :

r (read) = 4  
w (write) = 2  
x (execute) = 1  
- = 0

### Par exemple,

Pour **rwx**, on aura :  $4+2+1 = 7$

Pour **rw-**, on aura :  $4+2+0 = 6$

Pour **r--**, on aura :  $4+0+0 = 4$

Ce qui permet de faire toutes les combinaisons :

|                                          |
|------------------------------------------|
| 0 : --- (aucun droit)                    |
| 1 : --x (exécution)                      |
| 2 : -w- (écriture)                       |
| 3 : -wx (écriture et exécution)          |
| 4 : r-- (lecture seule)                  |
| 5 : r-x (lecture et exécution)           |
| 6 : rw- (lecture et écriture)            |
| 7 : rwx (lecture, écriture et exécution) |

232

# Système d'Exploitation

## Les protections des fichiers (Droits d'accès aux fichiers)

### Droits d'accès en octal

#### Exemple :

Soit le répertoire LIDAI avec les permissions suivantes:  
drwxr-xr-x

On peut changer ces permissions en rwx r-x --- en utilisant la commande chmod et les permissions en octal comme suit :

En octal, on aura **750** : rwx r-x --- 7(4+2+1) 5(4+0+1) 0(0+0+0)

Pour mettre ces permissions sur le répertoire on taperait donc la commande :

**chmod 750 LIDAI**

Modification sur un fichier existant :

**chmod droit fichier**

# Système d'Exploitation

## Architecture du système Unix/Linux

### Les protections des fichiers (Droits d'accès aux fichiers)

#### Exemple 2:

```
root@debian:/home# mkdir LIDAI
root@debian:/home# cd LIDAI
root@debian:/home/LIDAI# touch lic_lidai.txt
root@debian:/home/LIDAI# ls -ls
total 0
0 -rw-r--r-- 1 root root 0 nov. 18 13:28 lic_lidai.txt
root@debian:/home/LIDAI#
```

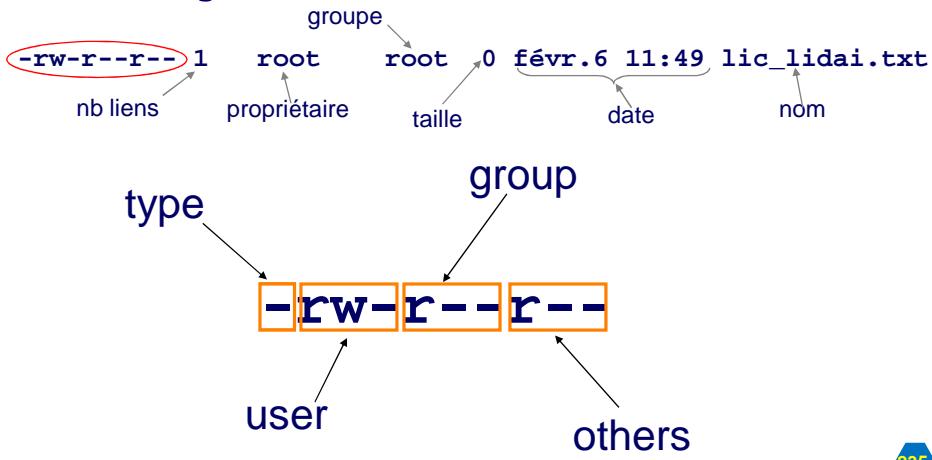
- **Lic\_lidai.txt** est un fichier. C'est le premier caractère qui donne le type de fichier/répertoire (le - indique qu'il s'agit d'un fichier ordinaire).
- Son propriétaire est root, du groupe root
- les protections **-rw-r--r--** sont à interpréter selon les indications ci-dessus.
- La commande **ls - ias** permet l'affichage des numéros d'inodes du répertoire.

# Système d'Exploitation

## Les protections des fichiers (Droits d'accès aux fichiers)

### Synthèse

- Affichage à l'aide de la commande: `ls -l`



235

# Système d'Exploitation

## Les protections des fichiers (Droits d'accès aux fichiers)

```
root@debian:/# ls -ls
total 84
4 drwxr-xr-x 2 root root 4096 mars 28 2021 bin
4 drwxr-xr-x 3 root root 4096 mars 28 2021 boot
0 drwxr-xr-x 17 root root 3060 nov. 18 10:17 dev
4 drwxr-xr-x 98 root root 4096 nov. 18 09:49 etc
4 drwxr-xr-x 26 root root 4096 nov. 18 13:27 home
0 lrwxrwxrwx 1 root root 29 mars 28 2021 initrd.img -> boot/initrd.img-4.9.0-6-amd64
0 lrwxrwxrwx 1 root root 29 mars 28 2021 initrd.img.old -> boot/initrd.img-4.9.0-6-amd64
4 drwxr-xr-x 15 root root 4096 mars 28 2021 lib
4 drwxr-xr-x 2 root root 4096 mars 28 2021 lib64
4 -r----- 1 root root 932 mars 22 2022 libncurses5-dev_6.3-2_amd64.deb
16 drwx----- 2 root root 16384 mars 28 2021 lost+found
4 drwxr-xr-x 3 root root 4096 mars 28 2021 media
4 drwxr-xr-x 4 root root 4096 avril 10 2021 mnt
4 drwxr-xr-x 2 root root 4096 mars 28 2021 opt
0 dr-xr-xr-x 117 root root 0 nov. 18 09:49 proc
4 drwx----- 19 root root 4096 nov. 18 09:50 root
0 drwxr-xr-x 20 root root 600 nov. 18 09:50 run
4 drwxr-xr-x 2 root root 4096 mars 28 2021 sbin
```

236

# Système d'Exploitation

## Droits d'accès initiaux

Masque de droits d'accès !retirés! à la création de tout fichier

❑ Commande **umask (user mask)**

❑ Le masque est donné en octal (base 8) avec 3 chiffres (u, g, o)

❑ En standard, masque par défaut = 022

➤ r = 100 en binaire = 4 en octal, w = 010 = 2

➤ Si droits retirés --- -w- -w-, alors droits appliqués rw- r-- r--

➤ La plupart des programmes n'ajoutent déjà pas le droit x

```
0 -rw-r--r-- 1 root root 0 nov. 18 13:28 lic lidai.txt
root@debian:/home/LIDAI#
```

Modification du masque grâce à la commande umask

➤ Attention : umask sans effet rétroactif sur les fichiers préexistants

➤ Attention : umask n'a d'effet que sur le bash courant

237

# Système d'Exploitation

## Droits d'accès initiaux

❑ Commande **umask (user mask)**

**umask (user file creation mode mask)**, masque de création de fichier par l'utilisateur) est un attribut des processus Unix, ainsi que la commande POSIX qui permet de modifier cet attribut. Le **umask** définit les permissions par défaut d'un répertoire ou d'un fichier créé. La syntaxe de la commande est la suivante : **umask x**

Où x est un nombre exprimé sous forme octale qui déterminera les permissions par complétion de 0666 pour les fichiers et de 0777 pour les répertoires qui seront créés ultérieurement par les appels systèmes creat(2) et mkdir(2), c'est-à-dire que les permissions seront obtenues par l'opération binaire 0777 (ou 0666).

Le umask le plus courant est 0022. Il consiste à supprimer les droits d'écriture pour les membres du groupe et les autres sur les fichiers. En effet :

```
0 -rw-r--r-- 1 root root 0 nov. 18 13:28 lic lidai.txt
root@debian:/home/LIDAI#
```

238

# Système d'Exploitation

## Droits d'accès initiaux

### Commande umask (*user mask*)

La commande umask peut être introduite avant chaque création de fichier ou de répertoire, mais une ligne est habituellement insérée dans le fichier d'initialisation du shell.

```
$ umask 0022
$ mkdir repertoire
$ touch fichier
$ ls -l
drwxr-xr-x 2 user user 512 Jan 1 23:59 repertoire -rw-r--r-- 1 user
user 0 Jan 1 23:59 fichier
```

```
0 -rw-r--r-- 1 root root 0 nov. 18 13:28 lic lidai.txt
root@debian:/home/LIDAI#
```

239

# Système d'Exploitation

## Gestion des utilisateurs sous Unix/Linux

### Création/suppression d'un compte utilisateur : useradd

La commande « useradd » permet d'ajouter un nouvel usager.

#### Syntaxe : **useradd [options] user**

- adduser : lien symbolique sur useradd

#### Exemple

La commande suivante crée le compte zouhair avec la plupart des options de base précisées.

**useradd -m -u 1010 -s /bin/bash -d /home/zouhair -c "Compte de zouhair" zouhair**

La commande « userdel » modifie les fichiers des comptes du système, en supprimant toutes les entrées qui se réfèrent à login.

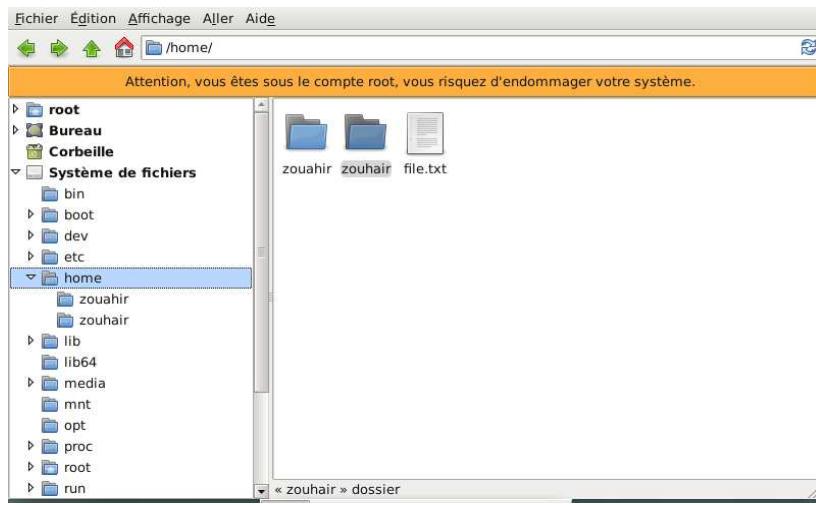
- Les fichiers présents dans le répertoire personnel de l'utilisateur seront supprimés en même temps que le répertoire lui-même. Les fichiers situés dans d'autres systèmes de fichiers devront être recherchés et éliminés manuellement.

240

# Système d'Exploitation

## Gestion des utilisateurs sous Unix/Linux

### Création/suppression d'un compte utilisateur : useradd



241

# Système d'Exploitation

## Gestion des utilisateurs sous Unix/Linux

### Création/suppression d'un compte utilisateur : useradd

Principales options (cumulables) :

| option | Rôle                                                                                                                                      |
|--------|-------------------------------------------------------------------------------------------------------------------------------------------|
| -u     | Précise l'UID numérique de l'utilisateur. Autrement l'UID est calculé selon les règles du fichier <b>login.defs</b> et les UID existants. |
| -g     | Précise le groupe principal par GID ou par son nom (variable GROUP).                                                                      |
| -G     | Précise les groupes additionnels (secondaires, de l'utilisateur) séparés par des virgules (variable GROUPS).                              |
| -d     | Chemin du répertoire personnel.                                                                                                           |
| -c     | Un commentaire associé au compte.                                                                                                         |
| -k     | Chemin du répertoire contenant le squelette de l'arborescence du répertoire utilisateur.                                                  |
| -s     | Shell par défaut de l'utilisateur (variable SHELL). L'utilisateur peut le changer via la commande <b>chsh</b> .                           |
| -m     | création du répertoire personnel et recopie du /etc/skel dans le répertoire personnel.                                                    |
| -p     | mot de passe déjà crypté avec crypt.                                                                                                      |
| -e     | date d'expiration du login (YYYY-MM-DD)                                                                                                   |
| -f     | délai avant verrouillage si mot de passe non changé                                                                                       |

242

# Système d'Exploitation

## Gestion des utilisateurs sous Unix/Linux

### Modification d'un compte utilisateur : usermod

La commande « usermod » modifie les informations d'un compte utilisateur.

Syntaxe : **usermod [options] user**

- Ne pas modifier l'uid pendant que l'utilisateur exécute une application
- Accepte les mêmes options que la commande useradd
- Mais dispose aussi des options particulières

| option     | Rôle                                                                                                                                                 |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| -L         | Lock du compte (verrouillage), comme passwd -l.                                                                                                      |
| -U         | Unlock du compte (déverrouillage), comme passwd -u.                                                                                                  |
| -e <n>     | Exire: le mot de passe expire n jours après le 01/01/1970.                                                                                           |
| -u <UID>   | Modifie l'UID associé au login. Le propriétaire des fichiers appartenant à l'ancien UID a u sein du répertoire personnel est modifié en conséquence. |
| -l <login> | Modifie le nom de login.                                                                                                                             |
| -m         | Move : implique la présence de -d pour préciser un nouveau répertoire personnel. Le contenu de l'ancien répertoire est déplacé dans le nouveau.      |

243

# Système d'Exploitation

## Gestion des utilisateurs sous Unix/Linux

### Actions de l'utilisateur

L'utilisateur dispose de certaines actions sur les informations de son compte :

- changer son shell de connexion,
- changer ses informations personnelles,
- changer de groupe principal,
- prendre l'identité de quelqu'un d'autre.

La commande **chsh** est invoqué pour modifier le shell qui est appelé lors de la connexion de l'utilisateur.

- -s : Indique le shell de connexion désiré.
- -l : Affiche la liste des shells mentionnés dans /etc/shells

Le commentaire du fichier /etc/passwd peut être modifié par l'utilisateur à l'aide de la commande **chfn**.

La commande **newgrp** qui permet de changer de groupe principal , L'équivalent de la commande su pour les groupes.

La commande **su** permet de change d'identité et ainsi d'exécuter un processus avec une autre identité.

244

# Système d'Exploitation

## Gestion des utilisateurs sous Unix/Linux

### Changement de mot de passe

La commande **passwd** permet de changer un mot de passe.

- Tout utilisateur a le droit de changer son mot de passe, dans le délai précisé par le 4 champ de /etc/shadow.

Tous les champs de /etc/shadow peuvent être modifiés par la commande **passwd**.

| option | Rôle                                                                                                                                                 |
|--------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| -l     | Lock : verrouille le compte en rajoutant un ! devant le mot de passe crypté.                                                                         |
| -u     | unlock: déverrouille le compte. Il n'est pas possible de déverrouiller un compte qui n'a pas de mot de passe, il faut utiliser en plus -f pour cela. |
| -d     | (root) Supprime le mot de passe du compte.                                                                                                           |
| -n <j> | (root) Durée de vie minimale en jours du mot de passe.                                                                                               |
| -x <j> | (root) Nombre de jours avant avertissement.                                                                                                          |
| -w <j> | Move : implique la présence de -d pour préciser un nouveau répertoire personnel. Le contenu de l'ancien répertoire est déplacé dans le nouveau.      |
| -i <j> | (root) Délai de grâce avant désactivation si le mot de passe est expiré.                                                                             |
| -S     | (root) Statut du compte.                                                                                                                             |

A

245

# Système d'Exploitation

## Gestion des utilisateurs sous Unix/Linux

### Commandes de gestions des groupes

La commande «groupadd» permet de créer un groupe sur le système.

Syntaxe :groupadd [options] user

Accepte l'argument **-g pour préciser un GID précis**

La commande « groupmod» permet de modifier un groupe. Ses paramètres sont les suivants :

| option    | Rôle                                                                                          |
|-----------|-----------------------------------------------------------------------------------------------|
| -n <nom>  | Renomme le groupe.                                                                            |
| -g <GID>  | Modifie le GID. Attention, le groupe d'appartenance des fichiers concernés n'est pas modifié. |
| -A <user> | Ajoute l'utilisateur spécifié dans le groupe (groupe secondaire).                             |
| -R <user> | Supprime l'utilisateur spécifié du groupe.                                                    |

246

# Système d'Exploitation

## Gestion des utilisateurs sous Unix/Linux

### Commandes de gestions des groupes

- La commande « groupdel » supprime un groupe.
  - Le groupe ne peut pas être supprimé si c'est un groupe principale d'un utilisateur.
- La commande « groups » permet de vérifier l'appartenance à un groupe
  - affiche les groupes auxquels l'utilisateur appartient
- La commande newgrp permet de changer à titre temporaire de groupe principal
  - À condition que le nouveau groupe précisé soit un groupe secondaire de l'utilisateur et/ou que l'utilisateur dispose du mot de passe du groupe.
  - Les modifications sont temporaires, le fichier des mots de passe n'est pas modifié.

247

# Système d'Exploitation

## Gestion des utilisateurs sous Unix/Linux

### Configuration avancée : /etc/default/useradd

Le fichier `/etc/default/useradd` contient un certain nombre de variables définissant les règles par défaut à appliquer à la création d'un compte.

- son groupe; la racine de son répertoire personnel (là où celui-ci sera situé), s'il est actif ou non, le shell, son ou ses groupes secondaires,
- l'endroit où est situé le squelette des comptes (structure de base d'un répertoire utilisateur), la création ou non d'un spool (dépôt) de courrier, etc

```
root@debian:~# cat /etc/default/useradd
Default values for useradd(8)
#
The SHELL variable specifies the default login shell on your
system.
Similar to DSHHELL in adduser. However, we use "sh" here because
useradd is a low level utility and should be as general
as possible
SHELL=/bin/sh
#
The default group for users
100=users on Debian systems
Same as USERS_GID in adduser
This argument is used when the -n flag is specified.
The default behavior (when -n and -g are not specified) is to create a
primary user group with the same name as the user being added to the
```

# Système d'Exploitation

## Quelques Commandes usuelles

- Il existe de nombreuses commandes de base décrites dans la section du Manuel de référence ou accessibles par la commande : `man nom_de_commande`
- Syntaxe générale: [commande] [- option] [paramètres (ou arguments)]
  - `pwd`: chemin absolu au répertoire courant;
  - `cd`: déplacer dans répertoire ;
  - `ls`: liste le contenu d'un répertoire (`ls -l` (`l=long`), `ls -a` ; `ls -a -l` ; `ls -al`);
  - `mkdir`: crée un répertoire;
  - `rm`: supprimer un fichier/répertoire;
  - `rmdir`: supprimer un répertoire vide;
  - `cp`: copie un fichier/répertoire;
  - `mv`: déplace/renomme un fichier/répertoire;
  - `ln`: lien d'un fichier (`ln -s`: faire une copie logique du fichier, en général on utilise les liens (links) symboliques (option `-s`));
  - `ps`: liste les processus;
  - `chmod`: modifie les droits d'accès:
    - ✓ Symbolique: `chmod [utilisateur] +/- [droit] [fichier]`  
(exemple: `chmod ug+x`);
    - ✓ Absolu: `chmod xxx [fichier]` ( $0 \leq x \leq 7$ );

249

# Système d'Exploitation

## Les outils UNIX/Linux

- ❑ UNIX est livré avec un grand nombre de programmes utilitaires.  
Ces programmes sont très divers, mais surtout orientés vers le traitement de fichiers de textes et le développement de logiciels.  
Tout système UNIX inclut normalement un compilateur C.
- ❑ Les utilitaires les plus importants sont les suivants :
  - Interpréteurs de commandes (nommés shells), permettant l'accès d'un utilisateur au système. Les shells sont assez sophistiqués et s'apparentent à de véritables langages de programmation interprétés;
  - Commandes de manipulation de fichiers ;
  - Commandes de gestion des processus ;
  - Éditeurs de texte (`vi`, `Emacs/Xemacs`, ..., etc.);
  - Outils de développement : compilateurs, débugueurs, analyseurs lexicaux et syntaxiques, etc.

250

## Références Bibliographique

1. Linux - Administration système et exploitation des services réseau, Philippe BANQUET et Sébastien BOBILLIER, Edition Eni, 2014.
2. Autheentification Liinux // MacOSX , Emmanuel Blindauer, Direction Informatique, Université de Strasbourg
3. José Ángel Herrero Velasco, "Computer System Design and Administration", Department of Computer and Electrical Engineering.
4. [http://chloecabot.com/mmi/M1203/synthese\\_unix.html](http://chloecabot.com/mmi/M1203/synthese_unix.html)
5. <http://web.mit.edu/rhel-doc/3/rhel-rg-fr-3/s1-pam-sample-simple.html>
6. <http://fantasyzone.free.fr/Documentation/Formations/Linux/S%Écurit%É/debdapauth.html>
7. <https://info.blaisepascal.fr/nsi-systemes-dexploitation>
8. [http://mauran.perso.enseeiht.fr/pages/cours/Systemes\\_Centralises/td1in/td001.html](http://mauran.perso.enseeiht.fr/pages/cours/Systemes_Centralises/td1in/td001.html)
9. <https://www.formatux.fr/formatux-securite/module-020-pam/index.html>
10. <http://www-igm.univ-mlv.fr/~dr/XPOSE2003/augereau/2.html>
11. <https://pub.phyks.me/sdz/sdz/les-acl-access-control-lists-sous-linux.html>
12. <http://bjobard.perso.univ-pau.fr/Cours/ISE/TP5.html>
13. <https://opensharing.fr/commandes-linux-getent>
14. [http://www.minetti.org/wiki/Linux:Configuration\\_de\\_NSS/PAM\\_pour\\_une\\_authentification\\_via\\_un\\_LDAP](http://www.minetti.org/wiki/Linux:Configuration_de_NSS/PAM_pour_une_authentification_via_un_LDAP)
15. <https://developpement-informatique.com/article/475/pluggable-authentication-modules-pam>
16. <http://eole.ac-dijon.fr/documentations/2.5/complètes/HTML/ModuleAmonEcole/co/10-IntroClientsLinux.html>
17. <https://docplayer.fr/1680951-Protocole-aaa-principes-et-implantations.html>
18. Introduction aux systèmes d'exploitation, Jalil BOUKHOBZA, Lab-STICC, univ-brest.
19. Source: « Systèmes d'exploitation », Andrew Tanenbaum, 2ème édition, Pearson Education 2001
20. <https://www.malekal.com/comment-utiliser-redirections-entrée-sortie-pipe-linux/>
21. Cours Unix/Linux, Pr. EL.M. EN-NAIMI, cours Licence & Master, Département Génie Informatique, FST de Tanger.