Connaissance du système

Fabien Rico

Univ. Claude Bernard Lyon 1

séance 4

Fabien Rico	fabien.rico@univ-lyon1.fr	CM+TD+TP
Jacques Bonneville	jacques.bonneville@univ-lyon1.fr	TP
Adil Khalfa	adil.khalfa@cc.in2p3.fr	TD + TP
Yves Caniou	yves.caniou@univ-lyon1.fr	TP
Dorra Boughzala	dorra.boughzala@ens-lyon.fr	TP

Système et administration

Système d'exploitation

- Savoir comment les choses sont organisées
- Pour comprendre les problèmes
  - Problèmes mémoire
  - Interblocages
- Pour utiliser les caractéristiques du système
  - Communications entre processus
  - Multi-threading
  - Gestion des resources

Système

- Configuration
- Utilisateurs
- outils

Configurations

- Chaque logiciel doit stocker des informations spécifiques de configuration,
- qui doivent être conservées
- Facilement modifiables
- Organisées
  - Centraliser les données de la machine
  - Distribuer celles des utilisateurs
  - Organiser par thème
  - Organiser par logiciel
  - Organiser par programmation ou entreprise
- Pour le stockage on peut utiliser :
  - Un système de fichiers
  - Une base de données
- Et les interfaces ?

Quoi ?

- Comment démarrer (`/boot/`, `c:\boot.ini`)
- Les services : logiciel annexe à démarrer (service d'authentification, serveur d'accès, gestion du matériel, reseaux...)
- Installation, librairies partagées et drivers
- Les utilisateurs, droits
- Configurations des logiciels.
- Configurations spécifiques aux utilisateurs.

Système de fichiers

Sous unix tout est fichier, les périphériques `/dev/`, les variables du système et leurs informations`/sys/` et `/proc/`, la configuration aussi

- `/boot/` pour le démarrage ;
- `/etc/rc.d`, `/etc/init.d`, `/etc/[x]inetd` ou `/etc/systemd` pour les services ;
- `/etc/<nom du logiciel d'installation>`, `/etc/ld.so.conf` (librairies) ;
- `/etc/<nom du logiciel>` par exemple `/etc/X11` pour le serveur graphique ;
- fichiers sur le compte utilisateur par ex.`/home/rico/.subversion` ;

Avantages et inconvénients

Avantages

- Objet simple déjà connu cela permet d'utiliser des techniques éprouvées (cp, rsync, diff, grep, locate...).
- Souple, on peut utiliser le format le plus adapté (langage de programmation de l'application, xml, script).
- Organisation naturelle par répertoires.

Inconvénients

- Lent (à voir)
- Souple (pas de schéma général)
- Difficile à distribuer dans un réseau (nis, ldap, nsswitch.conf)
- Pas forcement adapté :
  - Modifications concurrentes (Fichiers *.d)
  - Droits (Il faut être root)

Modifications concurrentes

Des fichiers différents peuvent avoir besoin de modifier la même configuration.

- Les serveurs doivent modifier la configuration des services
- Beaucoup de logiciels doivent modifier les configurations des utilisateurs (menu variable d'environement, logiciels)

Il est très difficile de modifier un fichiers de façon automatique (à l'installation ou à la désinstallation)

- Les fichiers les plus importants ou liés aux serveurs les plus populaires ont tendance à être coupés en plusieurs morceaux.
  - /etc/apache/http.conf → /etc/apache/conf.d/php.conf, /etc/apache/conf.d/squid.conf,...
  - /etc/modprobe.d/broadcom−wl−blacklist.conf
- Apparition de logiciels de configuration qui gèrent un ensemble de fichiers ... et ont eux même des fichiers de configurations.
  - dhclient (/etc/dhclient.conf) modifie /etc/resolv.conf
  - authconfig (red hat - /etc/sysconfig/authconfig) modifie les configuration de pam, nsswitch nis et libnsswitch −ldap.

La base de registre

- Pour améliorer l'accès aux configurations, on peut utiliser une base de données
- Avec une organisation similaire aux répertoires des systèmes de fichiers
- Chaque répertoire contient des données de différents types.
- C'est la base de registre.

Par exemple :

`\HKEY_CLASSES_ROOT \.pdf`	contient le type associé aux fichiers d'extention .pdf
`\HKEY_CLASSES_ROOT \AcroExch.Document\`	contient les icônes et logiciels associés

Stockage

- La base est stockée dans des fichiers « *backend* ».
- Mais, il faut passer par des utilitaires dédiés pour lire/modifier le contenu.
  - Gconf pour gnome (fichier xml)
  - Netinfo pour OS X (fichier de bd)
  - Regedit pour windows (fichier « ruches »)

Exemple : la base windows

- HKEY_LOCAL_MACHINE\HARDWARE la liste du matériel détecté
- HKLM\SYSTEM la configuration du système (fichier %systemroot%\system32\config\SYSTEM)
- HKLM\SAM les comptes (fichier %systemroot%\system32\config\SAM)
- HKLM\Software les logiciels (dont la sous branche classes est HK_CLASSES_ROOT)
- HK_USERS*id* Les données de l'utilisateur (fichier \Document and setting*login*\NTUSER.DAT)

Avantages et inconvénients

Avantages

- Plus rapide,
- Distribution des données souple ⇒ facile à distribuer.
- Permet les recouvrements (HKEY_CURRENT_USER contient HKEY_USERS\<*id de l'utilisateur courant*>)
- Chaque configuration est un objet avec ses propres droits.
- Logiciel spécifique (sauvegarde, versions...)

Inconvénients

- Logiciel spécifique (plus difficile à gérer)
- Chaque configuration est un objet avec ses propres droits (compliqué)
- Modifications concurrentes

Interface

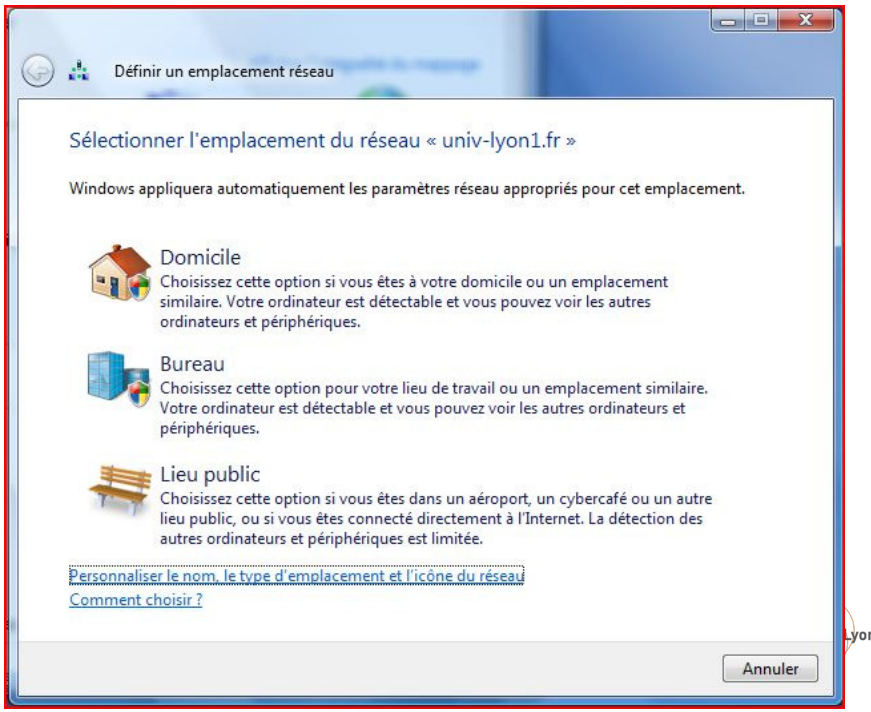
- Modifier directement les fichiers est parfois compliqué,
  - Par exemple pour utiliser l'authentification ldap sous linux il faut configurer :
  - le client ldap.
  - l'agent d'authentification qui doit l'utiliser.
  - le système qui doit remplacer les fichiers d'utilisateurs par un client.
- Pour ça il existe des utilitaires
- Pratique mais l'utilitaire :
  - Ne permet pas de tout configurer (ex : firewall).
  - Ne permet pas toujours de sauvegarder les configurations.
- Effet boite noire.

Assistants

Certains utilitaires sont basés sur des configurations toutes faites qu'on sélectionne en répondant à quelques questions. Ce sont les *Assistants*, ou *Wizards*

- C'est très simple à utiliser tout en permettant de faire des configurations très pointues
- La méthode est de choisir un scénario d'utilisation courant et de l'appliquer. Comme une « hotline » qui a un serveur d'appel automatique...
- ...Comme une hotline c'est souvent très difficile à utiliser.

  - Par exemple pour configurer un ordinateur portable au Nautibus :



Utilisateurs

- L'une des configurations les plus importantes est celle qui gère les utilisateurs
- Il faut
  - permettre l'authentification (généralement par mot de passe) ;
  - gérer les groupes ;
  - gérer les droits sur les différents composants ;
  - conserver les données (`/home/`).
- Ces données peuvent être
  - stockées localement sur la machine ;
  - centralisées sur un serveur ;
  - stockées dans un annuaire.

Localement

- Les informations sont stockées dans un fichier de la machine
  - `%systemroot%\system32\config\SAM` sous windows
  - /etc/passwd, /etc/shadow et /etc/groups sous linux
- Les fichiers ne contiennent pas directement les mots de passe mais leur empreinte numérique par une fonction de hachage.
  - Pour authentifier un utilisateur le système récupère le mot de passe en clair.
  - Il utilise la même fonction de hachage et compare les résultats.
- Ces fichiers sont critiques pour le système
  - Problème des mots de passe identiques.
  - Problème des mots de passe trop simples.

En réseau

- Dans un réseau local, il est nécessaire centraliser la gestion des utilisateurs.
- On peut modifier les utilitaires qui accèdent aux descriptions des utilisateurs pour qu'ils contactent un serveur.
  - Ex : NIS à chaque accès, le fichier correspondant est demandé au serveur.
  - Le service utilisé pour chaque fichier est géré par le « Name Service Switch » (fichier /etc/nsswitch.conf).
  - Cache local.
- On peut déléguer une partie du travail à un serveur
  - Ex : les domaines windows
  - le PDC fournit l'authentification
  - le reste est fait par des scripts
- Avec ces deux méthodes les informations centralisées sont limitées.

Annuaire

- Un annuaire est une base de données
  - optimisée pour la lecture.
  - Pouvant contenir tout type d'information
  - Avec une organisation hiérarchisée (arbre).
  - Permettant des recherches multiples.
  - Proposant un système d'authentification.
- Par exemple :
  - OpenLdap « Lightweight Directory Access Protocol ».
  - Active Directory qui utilise le protocole de nom ldap.

Ldap/AD

- Les objets sont placés dans une *structure arborescente*.
- La racine de la structure est liée au domaine DNS.
  DC=polytech,DC=upmc,DC=fr
- chaque objet a un nom unique le *distinguished name* ou *dn* faisant apparaître le chemin dans l'arbre
  - OU=comptes,DC=polytech,DC=upmc,DC=fr par exemple l'entité qui rassemble tous les comptes
  - OU=encad,OU=comptes,DC=polytech,DC=upmc,DC=fr par exemple l'entité qui rassemble tous les enseignants
  - CN=rico,OU=encad,OU=comptes, DC=polytech,DC=upmc,DC=fr mon compte
- À chaque objet on associe des données
- Le type des données et leurs positions dans l'arbre sont fixés par des *shémas* (donc identiques entre serveur, mais adaptables).
- Les droits d'accès aux données sont gérés par des *ACL* (Voir plus loin)

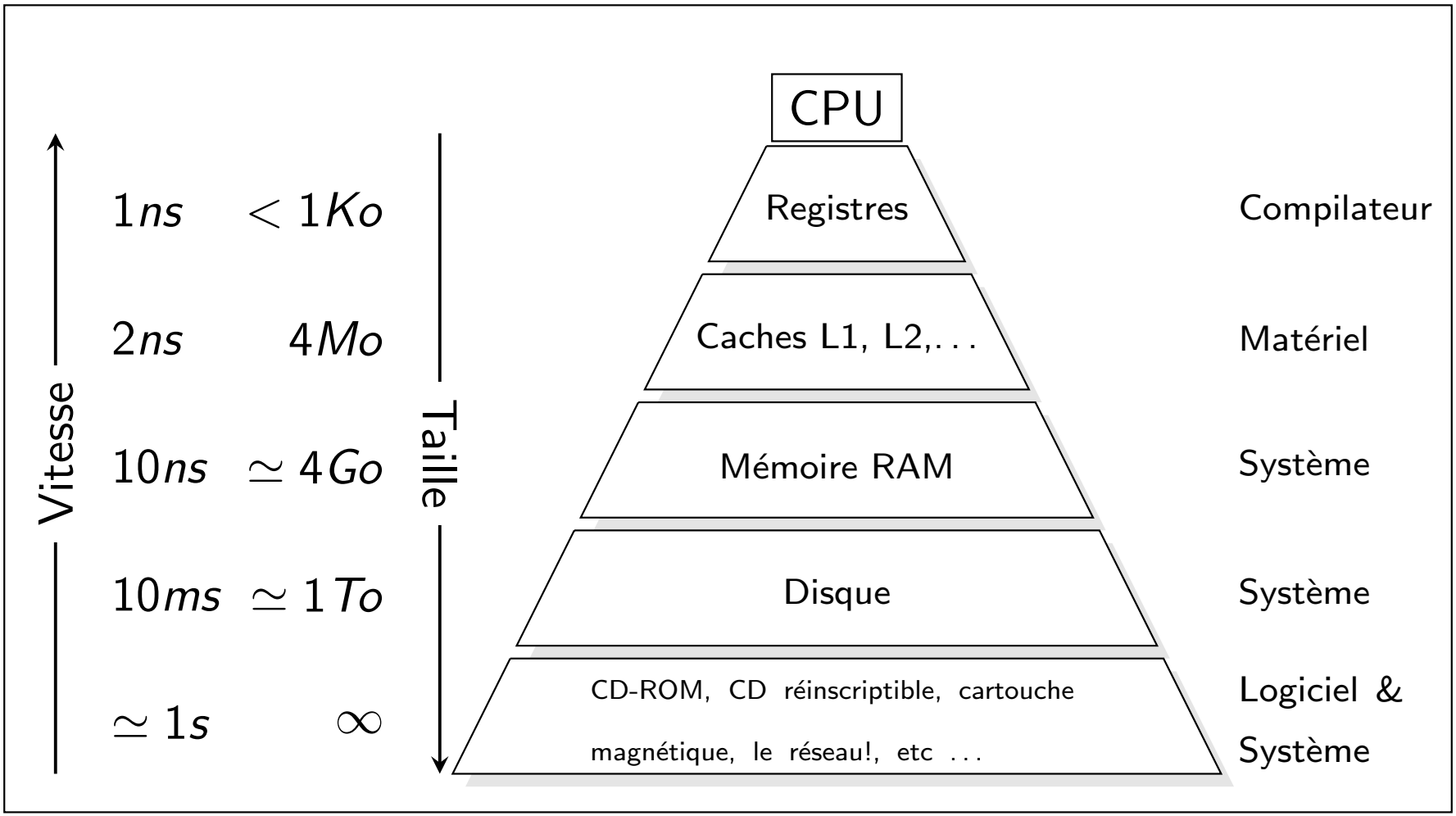
Exemple de données système

```
  # RICO FABIEN, D\C3\A9partement Informatique, UFR Sciences et Technologies, univ-lyon1.fr
# dn en base64
dn:: Q049UklDTyBGQUJJRU4sT1U9RMOpcGFydGVtZW50IEluZm9ybWF0aXF1ZSxPVT1VRlIgU2NpZ
 W5jZXMgZXQgVGVjaG5vbG9naWVzLERDPXVuaXYtbHlvbjEsREM9ZnI=
...
cn: RICO FABIEN
sn: RICO
l: VILLEURBANNE CEDEX
street: Campus DOUA Ouest
description: UFR Sciences et Technologies
postalCode: 69622
physicalDeliveryOfficeName:: IELDonRpbWVudCBOYXV0aWJ1cyA4IEJvdWxldmFyZCAgTmllb
 HMgQk9IUiA=
telephoneNumber: 0472431656
givenName: FABIEN
...
displayName: RICO FABIEN
# mes groupes
memberOf: CN=cloud-user,OU=groups,OU=d64,OU=sim,OU=univ-lyon1,DC=univ-lyon1,DC=fr
memberOf: CN=nautibus.mc-dpt,OU=BV Groupes,OU=CRICampusSud,DC=univ-lyon1,DC=fr
...
# des adresses mail
proxyAddresses: smtp:FABIEN.RICO@adm.univ-lyon1.fr
proxyAddresses: SMTP:fabien.rico@univ-lyon1.fr
...
employeeType: PERSONNEL
# login windows
sAMAccountName: fabien.rico
# info unix
mail: fabien.rico@univ-lyon1.fr
gidNumber: 2000
loginShell: /bin/bash
uidNumber: 56257
unixHomeDirectory: /home/pers/fabien.rico
```

À quoi sert la mémoire ?

- Stocker le code du programme
- Stocker le code des fonctions partagées
- Stocker les variables globales (Tas)
- Stocker les variables locales (Pile)
- Une donnée pour pouvoir être traitée doit être dans un registre du processeur.

Hiérarchie des mémoires



Principe de localité : Les programmes tendent à utiliser des instructions et des données accédées dans le passé (localité temporelle) ou proches de celle-ci (localité spatiale)

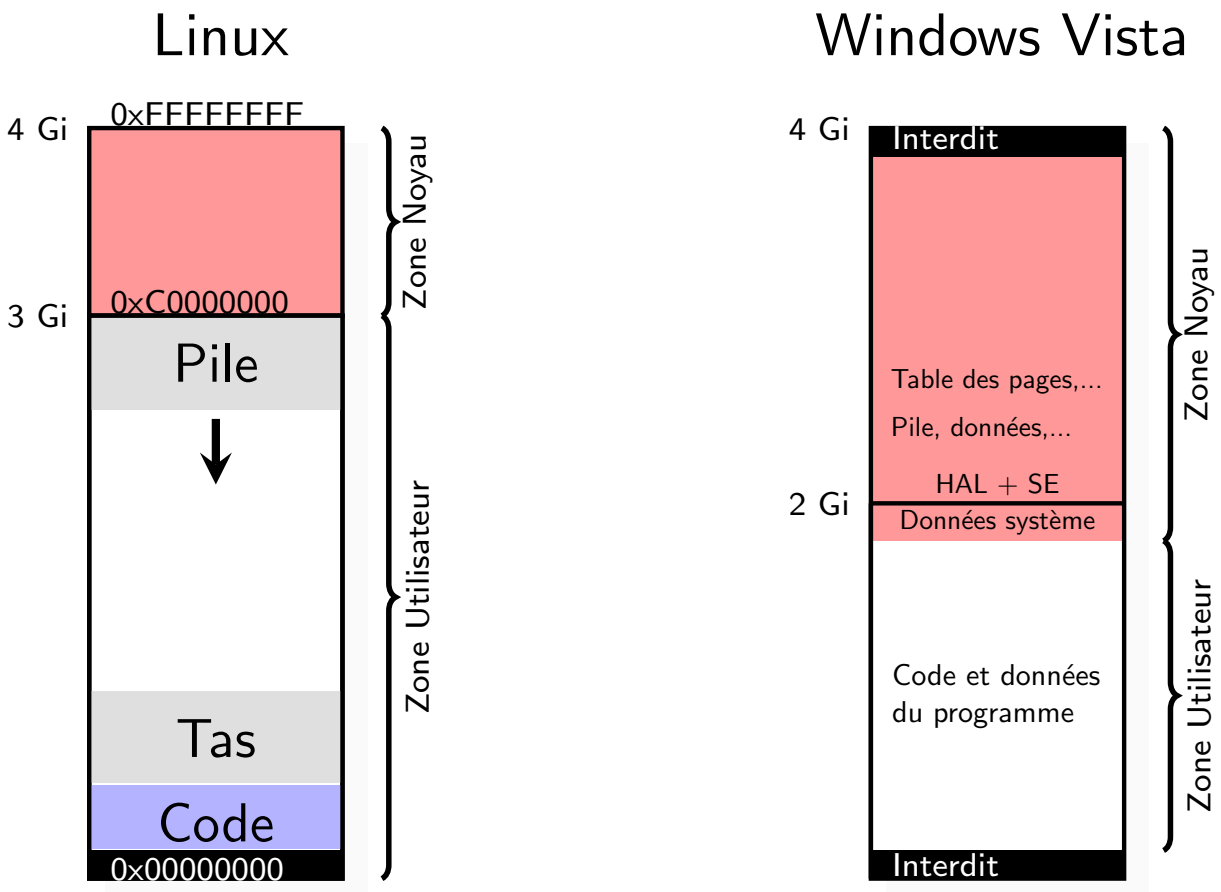
Cercle *vertueux ?*

- Les programmeurs suivent naturellement le principe de localité.
- Donc ceux qui conçoivent les systèmes s'en aperçoivent et l'utilise (mécanismes de pages, overlay...).
- Donc pour garder de bonnes performances les programmeurs suivent ce principe.
- Donc ceux qui conçoivent les processeurs l'utilisent (pipeline, cache mémoire...)
- Donc optimiser un code revient souvent à améliorer sa localité
- ...
- Ce principe prend de plus en plus d'importance.

Mémoire virtuelle

- Idée : le processeur travaille avec des adresses mémoires sans rapport avec les adresses physiques.
- *Capacité d'adressage* : l'ensemble des adresses que l'on peut coder, par ex. 4Go sur un processeur 32 bits (0x00000000 - 0xFFFFFFFF), 16Eo pour un 64 bits.
- *Espace d'adressage* : partie utilisable par le processus.
- L'espace d'adressage est partitionné en segments dépendant du système d'exploitation.
- À chaque lecture mémoire, l'adresse physique est calculée.
- Ce calcul permet de faire des vérifications supplémentaires :
  - présence effective de l'adresse en mémoire ;
  - droits.
- Il faut un matériel dédié la *Memomy Management Unit*.

Espace d'adressage d'un processus



Pagination

Comment faire la correspondance par exemple sur un processeur 32 bits

- La MMU utilise une table de correspondance *@ logique* $\mapsto$ *@ physique*
- La table est stockée en mémoire
- Si on stocke tout, 2^{32} @ à conserver
- $\Rightarrow$ On rassemble les adresses en *pages* (par ex. 4096 octets)
- $\Rightarrow$ 1 Mo à stocker (par processus), la plupart du temps les cases de la table sont vides.
- Donc on utilise plusieurs niveaux

  Ob 0100 1110 :0001 0101 1001 :1001 0101 1001

  Ox 4e : 159 : 959

  `rep` : `page` : `offset`

  rep index dans le répertoire des pages, *page* index dans la table des pages, *offset* position dans la page.

Ex. de pagination (suite)

La MMU du pentium permet une table de page avec 2 indirections :

- Pour lire l'adresse 0x4e159959
- La MMU contient un registre donnant l'adresse de la *table des répertoires de pages*.
- On trouve la table des pages à l'index $0x4e = 78$ de la table des répertoires de page
- On trouve la page à l'index $0x159 = 345$ de la table des pages
- La page se trouve en mémoire physique à une adresse 0x?????000 le mot mémoire demandé est à l'adresse physique 0x?????959
- Si on ne trouve pas la page, il y a interruption
  - Si la page existe mais n'est pas en mémoire *Défaut de page*
  - Si la page n'existe pas ou est interdite le processus est signalé (SIGSEG)
- Pour changer de contexte, il faut changer le registre d'adresse du répertoire de page.

Remarque

- En réalité, plusieurs accès mémoire par accès demandé
  - Il faut un cache de traduction pour améliorer le *TLB* ≪ Translation Lookaside Buffer ≫
  - Équilibre taille des tables/nombre d'indirections
- L'adresse d'une page n'utilise qu'une partie d'un mot mémoire (20 bits/32), les bits restant servent à stocker des informations :
  - La page peut-elle être mise en cache ?
  - La page existe ?
  - La page est-elle en accès lecture/écriture/exécution ?
  - La page est modifiée par rapport au disque (Dirty bit) ?
  - La page a-t'elle été accédée récemment ?
  - ...

Avantage de la pagination

- Les processus ont tous un même espace mémoire.
- Permet la séparation des processus.
- Permet le partage de mémoire :
  - Entre processus pour créer un moyen de communication.
  - Partage de zones mémoires contenant le code des bibliothèques partagées.
- Copie à la demande de la mémoire d'un processus lors d'un **fork**.
  - Le processus fils recopie la table des pages du processus père en marquant chaque page en lecture seule.
  - C'est l'écriture qui déclenche réellement la copie.
- Initialisation à zéro sur le même principe.

On complique un peu

Les algorithmes précédants ne sont que des simplifications :

- Besoin de pages contiguës (driver).
- Certaines pages ne peuvent être libérées :
  - lorsqu'un processus opère une lecture sur le disque, il est endormi mais ses pages doivent rester ;
  - les pages du système.
- Certaines pages sont plus ≪ difficiles ≫ à libérer
  - pages partagées ;
  - page modifiées.

  Le système doit éviter de les libérer.
- Contraintes externes :
  - temps réel (ne pas swapper un processus critique) ;
  - consommation d'un portable (pas d'écriture régulière sur le disque).

Multiprogrammation et mémoire

- En général, les cadres libres sont gérées de manières globales pour tous les processus.
- Une processus qui demande plus de mémoire aura plus de pages qui lui sont affectées.
- Il existe cependant des limites d'utilisation (ulimit, cgroups, ...)
- Comment être sur qu'un processus a suffisamment de mémoire ?

Quelle est la définition de ≪ suffisamment de mémoire ≫ ?

Ensemble de travail

Définition

Par le principe de localité, un processus sur une période courte utilise un sous ensemble limité de ses pages. C'est l'*ensemble de travail* (*working set*).

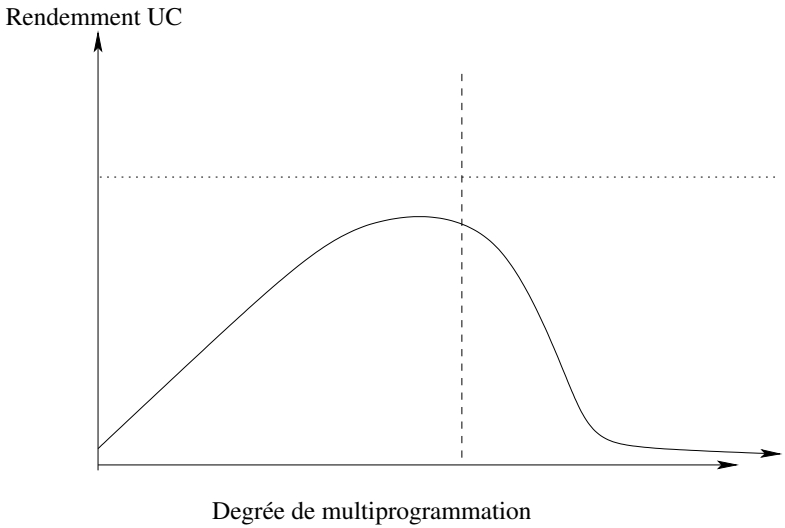
- Un processus qui accède à la mémoire va faire de nombreux défauts de page jusqu'à ce qu'il reconstitue son *espace de travail*.
- On peut éviter ces défauts de pages si on charge ces pages automatiquement.

$\Rightarrow$ un bon algorithme de remplacement conserve les page qui sont dans l'ensemble de travail d'un processus et libère rapidement les autres.

Écroulement

- Si le nombre de pages des ensembles de travail des processus prêts est supérieur à la mémoire physique, le système génère de nombreux défauts de pages

$\Rightarrow$ il y a risque d'**écroulement** ou **trashing**



Description de la mémoire d'un processus

Chaque processus doit avoir une structure pour décrire les différents segments :

- Fichiers mappé en mémoire
  - segment de code ;
  - bibliothèques partagées.
- Segments *anonymes*
  - pile ;
  - tas.
- Partages

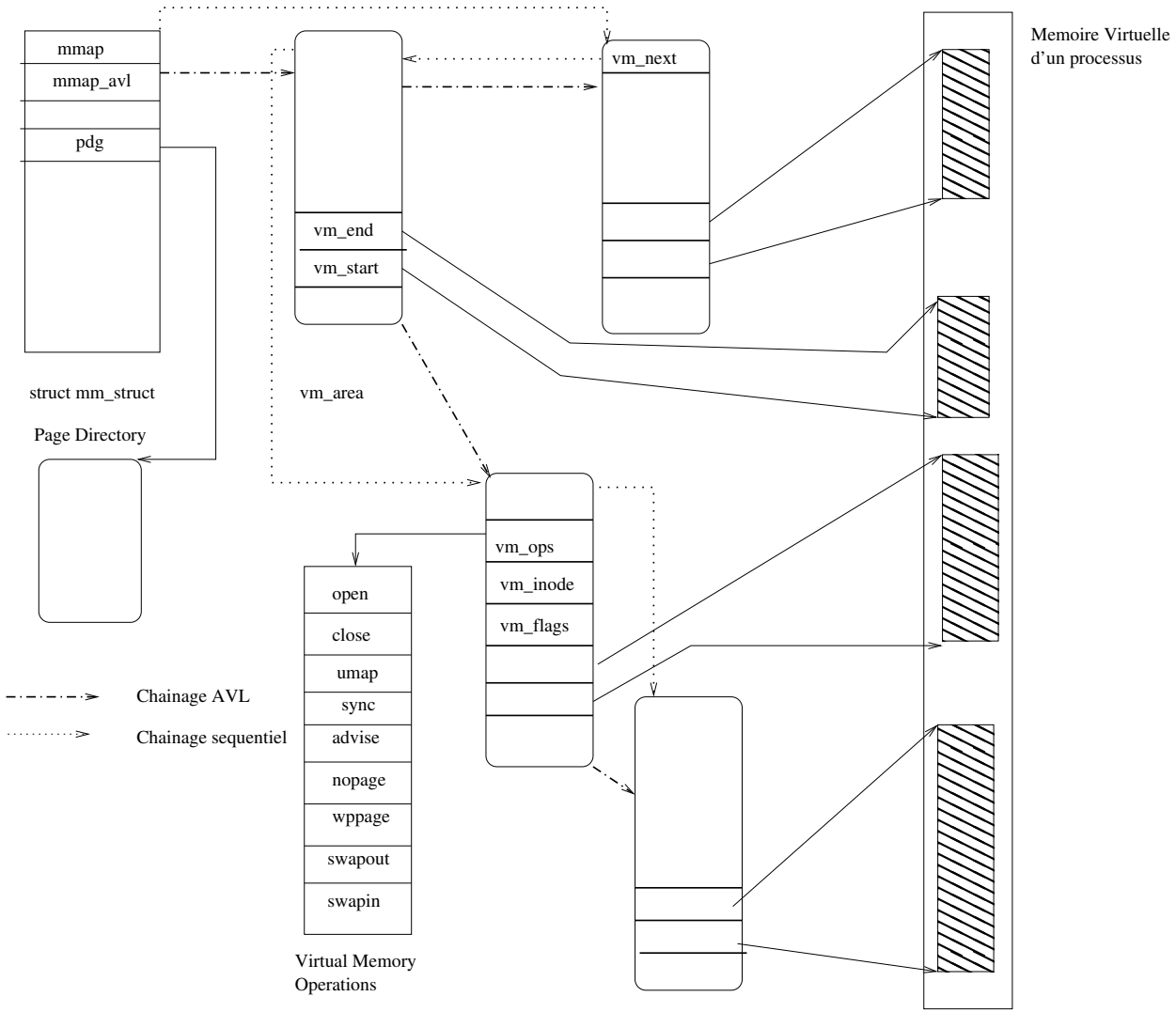
Voir la commande pmap ou le contenu du fichier /proc/pid_du_processus/maps

Description de la mémoire d'un processus

Le mappage mémoire d'un processus est une structure qui contient tous les différents segments.

- Chaque segment est une zone contiguë de la mémoire virtuelle.
- La structure doit être efficace pour un petit nombre de segment (liste) et un grand nombre (arbre AVL).
- Structure indépendante du matériel
  - **MachOS, Linux 2.2** : les zones de mémoires virtuelles, *vm area* ;
  - **Windows** : les Virtual Address Descriptor *VAD*.

VM Area


mmap
mmap_avl
pdg
struct mm_struct
Page Directory
vm_end
vm_start
vm_area
vm_next
Memoire Virtuelle
d'un processus
vm_ops
vm_inode
vm_flags
open
close
umap
sync
advise
nopage
wppage
swapout
swapin
Virtual Memory
Operations
Chainage AVL
Chainage sequentiel

Organisation des répertoires

Le système de fichier est séparé en plusieurs parties :

- Les répertoires utilisateurs `c:\\Users` ou `/home/`
- Les répertoires des programmes `c:\\ProgramFiles` ou `/usr/`
- Les répertoires de configuration `/etc/`
- Les répertoires virtuels `/proc/` ou `/sys/`
- Les accès au matériel `/dev/`

Répertoires virtuels

Ce sont des systèmes de fichiers spéciaux qui ne correspondent pas à des données sur le disque mais en mémoire.

- `/proc/` contient des informations sur les processus, les variables du système d'exploitation que l'on peut modifier.
  - `/proc/sys/net/ipv4/ip_forward` le système accepte-t-il de transmettre des paquets ipv4 ?
  - `/proc/4256/cmdline` la ligne de commande utilisée par le processus 4256.
- `/sys/` contient des informations sur chaque driver du noyau
  - `/sys/class/thermal/thermal_zone4/temp` température d'un processeur ;
  - `/sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq` fréquence courante du processeur 0.

Répertoire dev

Ce répertoire contient les accès aux périphériques disponibles :

- `/dev/disk/by...` les partitions des disques
- `/dev/sda?` le disque principal
- `/dev/pts/..` les terminaux
- `/dev/random` un faut périphérique relié au générateur pseudo aléatoire du système.

Résolution de problème

« *Tout programme non trivial possède au moins un bug.* »

Corollaire de la loi de Murphy.

- Il est donc nécessaire de savoir trouver et corriger les problèmes.
- Les systèmes donnent beaucoup d'informations qui généralement permettent de trouver la solution.
- Mais il faut savoir où chercher :
  - *Historique des événements* (logs) du système.
  - Service en mode *debug*.
  - Utilitaires.

Historique des événements

- Comme tous programmes, les services systèmes rendent compte de leurs actions.
- Ces messages sont centralisés et rassemblés /var/log/.
  - /var/log/message pour la plupart des logs.
  - /var/log/httpd/* ou /var/log/apache/* pour le serveur web
  - Xorg.0.log pour le serveur graphique
  - ...
- Il est souvent très instructif de suivre les logs système pour voir en temps réel les effets d'une action.

  ```
  tail -f /var/log/httpd/error.log
  ```
- Tous les services ont dans leur configuration un *niveau de log* qui permet d'augmenter le nombre d'information disponible.

Message d'erreur

- Généralement, les services sont lancés en tâche de fond, dans un mode complexe (multithread/multiprocessus,. . .).
- Mais ils peuvent être lancer en avant-plan pour la correction de problèmes.
  - dhcp −f
  - httpd −X
  - slapd −d 3
  - . . .
- Cela permet de les lancer dans un debugger, ou d'obtenir tous les message d'erreur de l'application.

Outils

- Debugger dbg, ddd, kdbg.
- Pour un script utiliser l'option −x qui affiche les commandes avant de les exécuter.
- Utilitaires d'écoute sur le réseau : tcpdump, wireshark.
- strace qui affiche les appels systèmes d'un programme.
- ltrace qui affiche les appels à une librairie et leur parametres.

Règles élémentaires

- Stopper les services de cache nscd et de sécurité firewall , selinux .
- Rechercher les options de sécurité par défaut.

Conclusion

- Administration
  - Il faut comprendre de qu'on fait.
  - Il faut être capable de l'adapter.
  - Savoir où trouver l'information.