Introduction aux problématiques architecturales des systèmes d'exploitation

Jean-Baptiste.Yunes@u-paris.fr

0.8/2020

Disclaimer: many of the pictures here presented come from the Internet,
not to overload the slides I assume the fact that no copyright notice is included,
if any is a copyright infringement please tell me.

Préambule

un cours ou une conférence…

inspiré des précédents dispensés par

Ronan Keryell

(now SE @ Xilinx, previously Telecom Bretagne, Silkan, AMD…)

François Armand

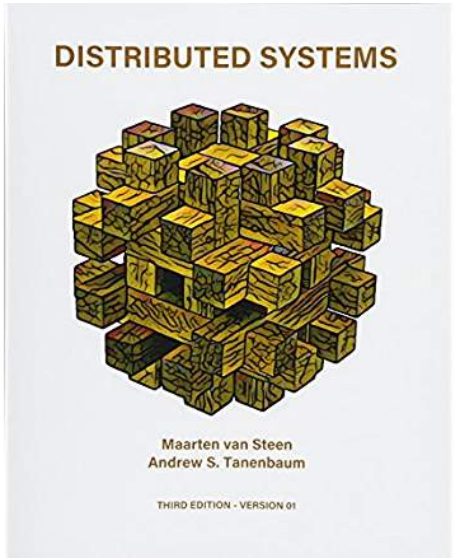
(now CSO @ Gland d'Atome, previously Chorus, Sun, VirtualLogix…)

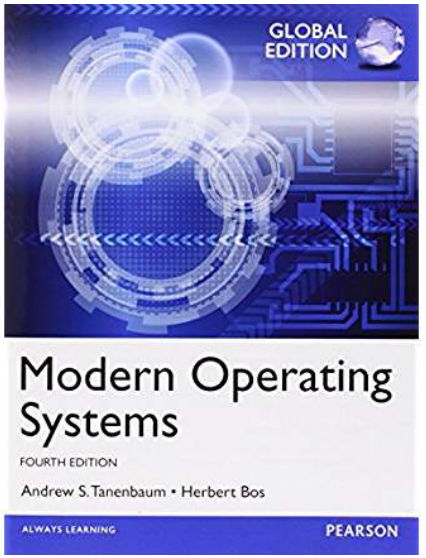
Retenez les concepts et les idées

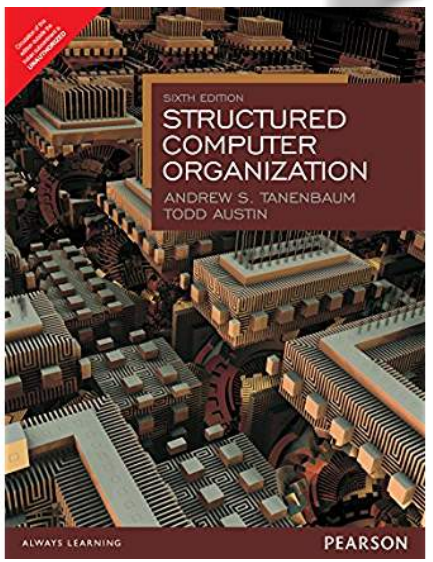
Soyez réactifs

Bibliographie

des ouvrages de Andrew Tanenbaum

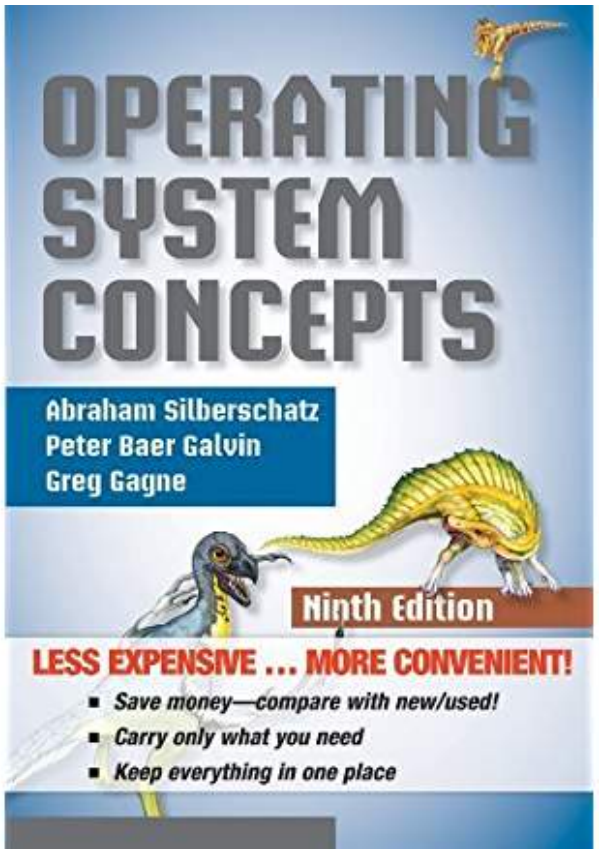


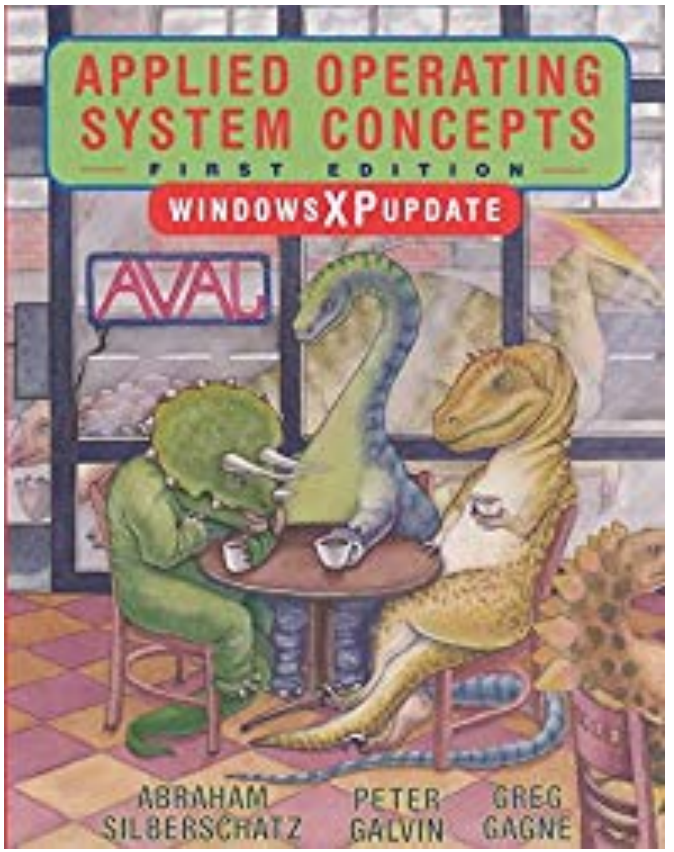




Bibliographie

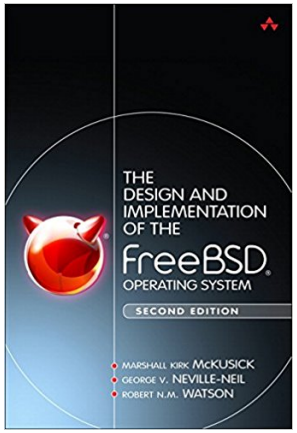
des ouvrages de Abraham Silbershatz

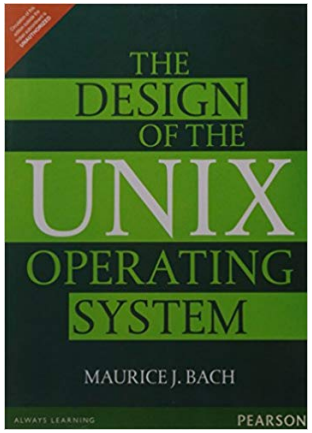


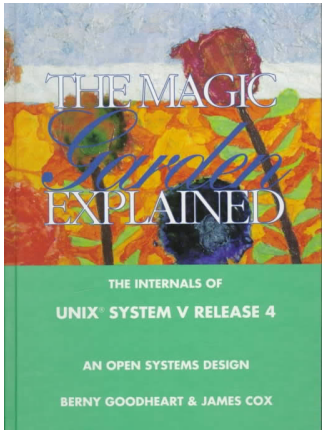


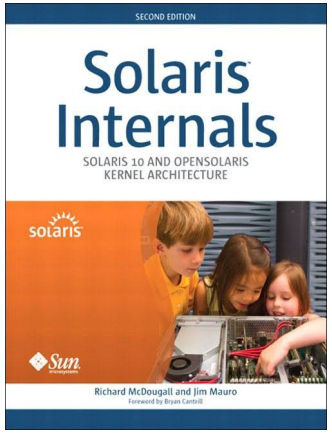
Bibliographie

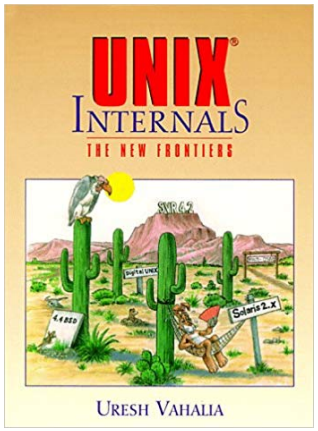
d'autres ouvrages (tout aussi majeurs)

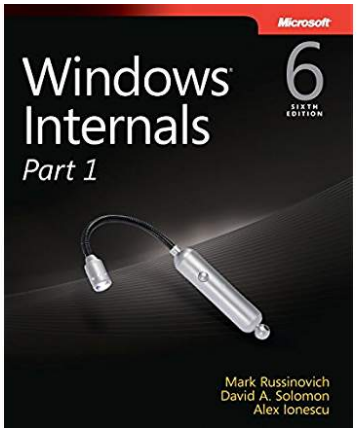


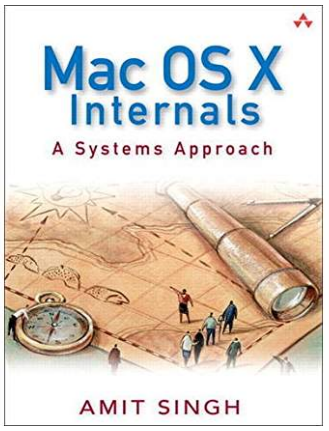












Bibliographie

pas d'intérêt à être exhaustif

de très nombreux ouvrages très intéressants existent

de très (trop) nombreux sites autour du sujet…

Bibliographie

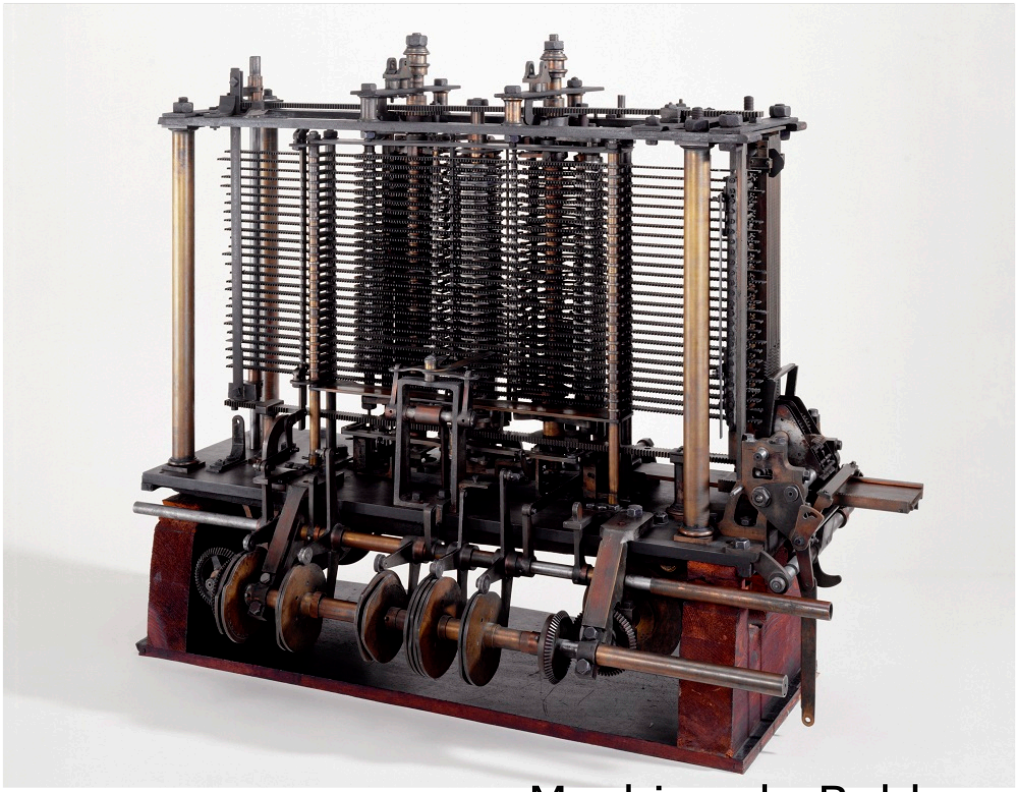
pourquoi autant d'ouvrages sur UNIX ?

- le système d'exploitation majeur des 40 dernières années…

Histoire

computerhistory.org (timeline/computers)

histoire.info.online.fr (<2012)

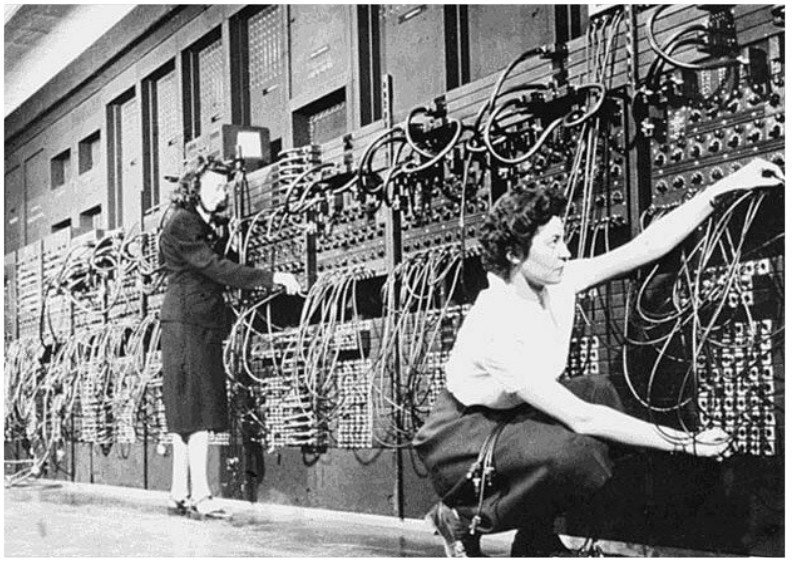


Machine de Babbage

Histoire

1945 — 1955

- Tout faire à la main…
- Tube électronique
- Le système c'était l'humain avec son expertise (inefficace)



Histoire

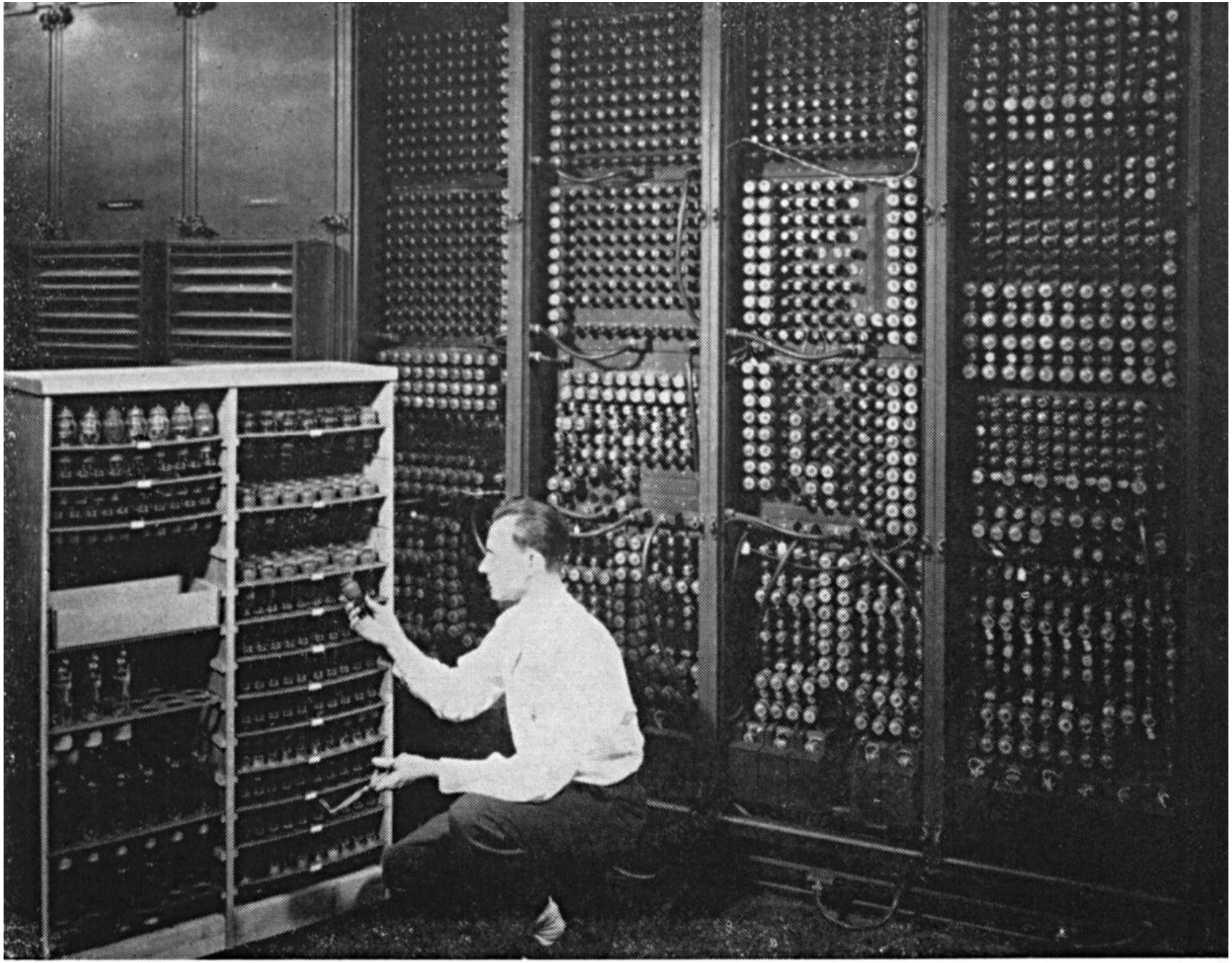
Le tube électronique (ou lampe)

- volumineux
- consommateur en énergie
- fiabilité aléatoire

Histoire

Principes établis par von Neumann, Eckert, Mauchly (1940)

- la mémoire contient à la fois le programme et ses données
- chemin de données
- ENIAC : premier ordinateur entièrement électronique
  - un « monstre » de 30 tonnes!
  - disponibilité proche de 0…
- Attention, de nombreux autres projets ont existé à cette époque (et un peu avant) mais étaient couverts par le secret militaire (Allemagne, Russie, Japon, France, Angleterre)…

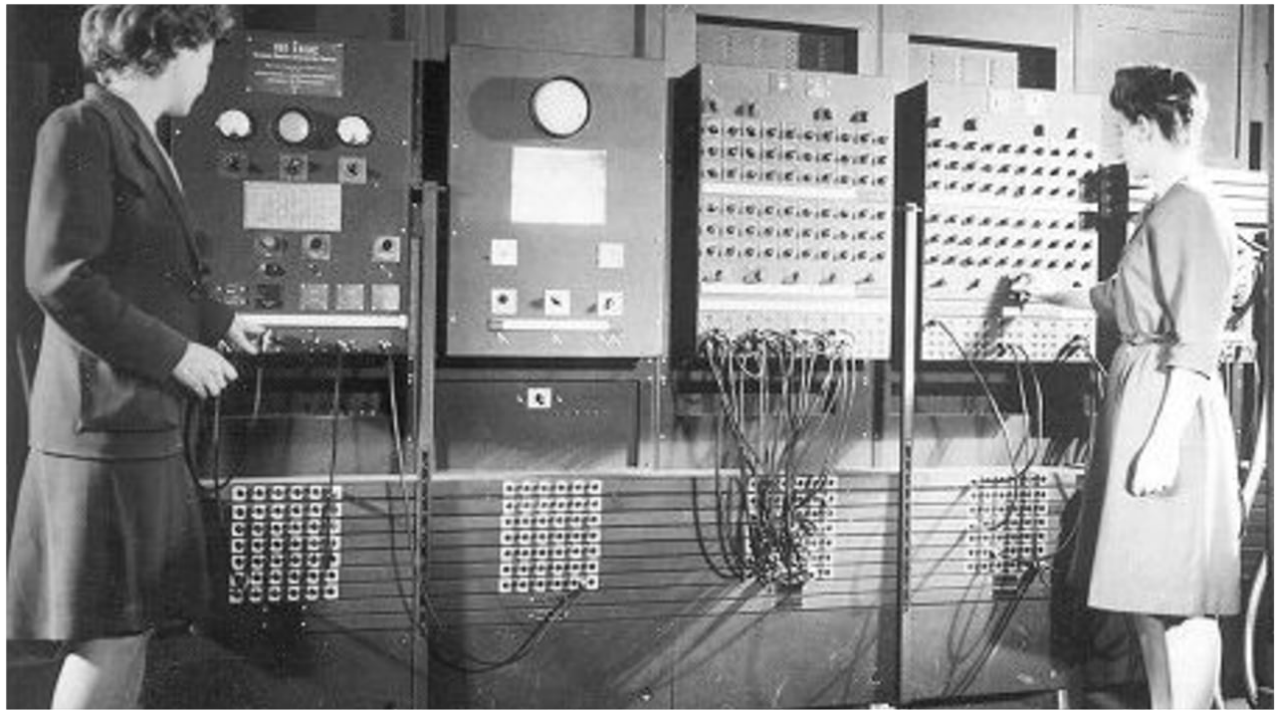


Replacing a bad tube meant checking among ENIAC's 19,000 possibilities.

Histoire

Un tableaux de commande de l'ENIAC

peut-on parler d'ergonomie ? de facilité d'utilisation ?



Histoire

Certaines machines de cette époque disposaient d'un assembleur

- un langage de programmation qui ne nécessitait plus d'effectuer des branchements sans fin ou la saisie de longues suites de 0 et de 1
- mais encore fallait-il transcoder cela…

Histoire

```
0013            RESETA  EQU     %00010011
0011            CTLREG  EQU     %00010001

C003 86 13      INITA   LDA A   #RESETA    RESET ACIA
C005 B7 80 04           STA A   ACIA
C008 86 11              LDA A   #CTLREG    SET 8 BITS AND 2 STOP
C00A B7 80 04           STA A   ACIA

C00D 7E C0 F1           JMP     SIGNON     GO TO START OF MONITOR

                *************************************
                * FUNCTION: INCH - Input character
                * INPUT: none
                * OUTPUT: char in acc A
                * DESTROYS: acc A
                * CALLS: none
                * DESCRIPTION: Gets 1 character from terminal

C010 B6 80 04   INCH    LDA A   ACIA       GET STATUS
C013 47                 ASR A              SHIFT RDRF FLAG INTO CAI
C014 24 FA              BCC     INCH       RECIEVE NOT READY
C016 B6 80 05           LDA A   ACIA+1     GET CHAR
C019 84 7F              AND A   #$7F       MASK PARITY
C01B 7E C0 79           JMP     OUTCH      ECHO & RTS

                *************************************
                * FUNCTION: INHEX - INPUT HEX DIGIT
                * INPUT: none
                * OUTPUT: Digit in acc A
                * CALLS: INCH
                * DESTROYS: acc A
                * Returns to monitor if not HEX input

C01E 8D F0      INHEX   BSR     INCH       GET A CHAR
C020 81 30              CMP A   #'0        ZERO
C022 2B 11              BMI     HEXERR     NOT HEX
C024 81 39              CMP A   #'9        NINE
C026 2F 0A              BLE     HEXRTS     GOOD HEX
C028 81 41              CMP A   #'A
C02A 2B 09              BMI     HEXERR     NOT HEX
C02C 81 46              CMP A   #'F
C02E 2E 05              BGT     HEXERR
C030 80 07              SUB A   #7         FIX A-F
C032 84 0F      HEXRTS  AND A   #$0F       CONVERT ASCII TO DIGIT
```

```
## %bb.0:
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset %rbp, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register %rbp
        subq    $80, %rsp
        leaq    -32(%rbp), %rdi
        leaq    -40(%rbp), %rsi
        movl    $0, -20(%rbp)
        callq   __ZNSt3__16threadC1IZ4mai
LBB0_1:                                 ##
        jmp     LBB0_2
LBB0_2:                                 ##
        movl    _a(%rip), %eax
        cmpl    $1, %eax
        jne     LBB0_10
## %bb.3:                               ##
        movl    _a(%rip), %eax
        cmpl    $2, %eax
        jne     LBB0_10
## %bb.4:                               ##
        movl    _a(%rip), %eax
        cmpl    $3, %eax
        jne     LBB0_10
## %bb.5:                               ##
Ltmp0:
        movq      ZNSt3__14coutE@GOTPCREL
```

Histoire

1955 — 1965

- Le transistor entre dans la danse (1954), puis le circuit intégré (1958)
- Langages de plus haut-niveau (expression facilité des problèmes à résoudre)
  - FORmula TRANslator (1954 — pour IBM 704)
  - ALGOrithmic Language (1958 - indépendant), l'ancêtre des langages modernes
  - LISt Processing (1958 - indépendant)
  - COmmon Business Oriented Language (1959 — indépendant)
- Générations de machines, naissance d'une industrie
  - Atlas : Mémoire Virtuelle
  - UNIVAC, IBM
  - BESYS, ShareOS, CTSS, GCOS, TOPS-10
- Première interfaces

Histoire

Semi-Automatic Ground System

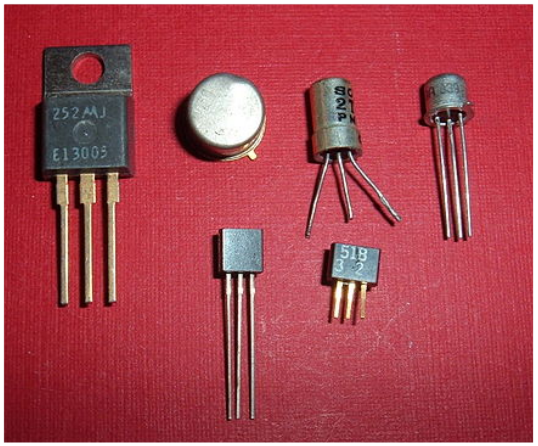
1958 - NORAD

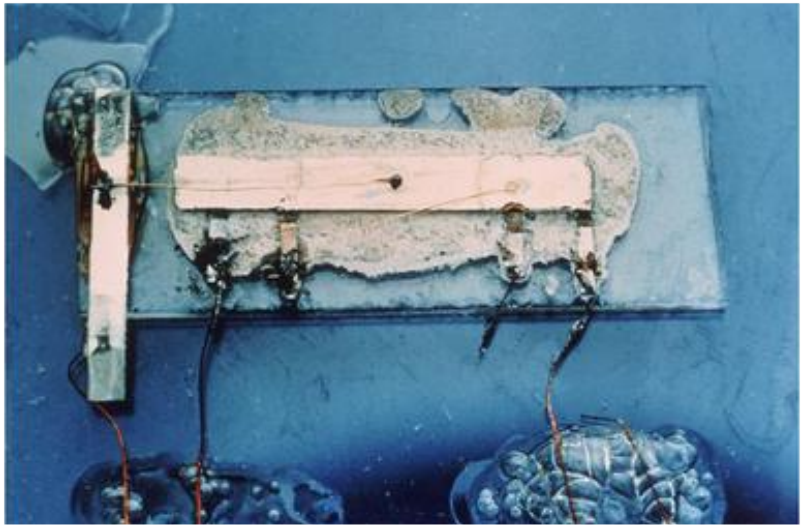
Histoire

Le transistor

ouvre la voie à la miniaturisation et donc l'intégration

augmente très sérieusement la fiabilité des machines





Histoire

IBM 350

Toujours un seul utilisateur

- le mythique ingénieur système

Réservation de la machine (tranches horaires)

Traitements par lots (on fait la queue)

- priorité

Stockage à bande magnétique, cartes, bandes perforées

Mono-programmation

Premiers super-calculateurs

Premiers disques durs

- gestion de la fragmentation

Attention, à cette époque les ordinateurs sont encore des monstres!

Histoire

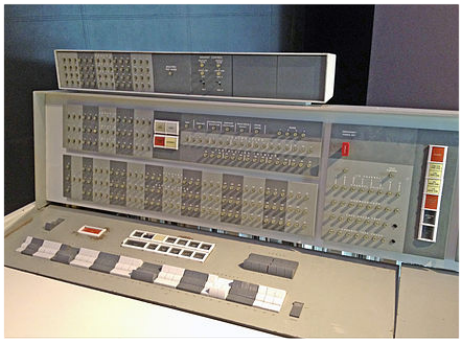
1960 - une année marquante

Multitâche, Multiprocesseur

- Bull Gamma 60, trop avancé (Bull en mourra)

Petits et gros

- Spécialisation des tâches
  - IBM 1401 (gestion des données)
  - IBM 7094 (calcul)



Histoire

1965 — 1985

- Le microprocesseur (1969)
- Industrie en pleine expansion
  - fourmillement de projets, sociétés, etc
- Diffusion du temps partagé, de la multiprogrammation, etc
- Multiplication des OS en tout genres
  - MVS, DOS, CP/M, Multics, Unix, VMS, MS-DOS

Histoire

L'âge d'or des gros systèmes

- multi-programmation
- spooling
- temps-partagé
- disques durs
- mémoire
- écrans graphiques
- modems

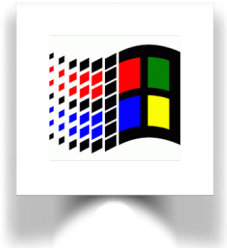
La loi de Moore

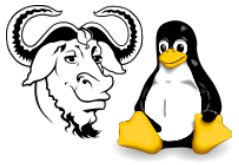
- aujourd'hui on a des problèmes avec les effets quantiques

Histoire

1980 — 2000

- Micro informatique (aka ordinateur personnel)
- Stations de travail
- Écrans graphiques
- Réseau (Internet)
  - donc des OS en conséquence…





Histoire

2000 —

- Ordinateurs portables
- Smartphones
  - l'iPhone (2007)
- Informatique embarquée
  - donc des OS en conséquence

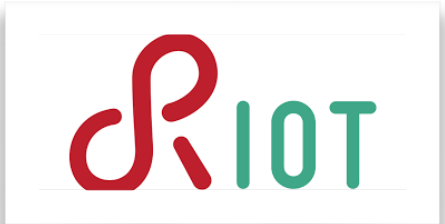
Histoire

2015 —

Internet des objets (IoT)

internet partout et dans tout ?

des OS en conséquence



Histoire

2018 —

Explosion des Smartwatches

des OS en conséquence

Problèmes

Ergonomie de l'usage

Problèmes

- Ergonomie de l'usage
  - au début les problèmes étaient très nombreux
    - mais l'automatisation de tâches d'exploitation est vite devenue cruciale
  - par la suite c'est la variété qui posa problème
    - s'affranchir des détails

Problèmes

Variété

- processeurs
- architectures matérielles
- périphériques

Problèmes

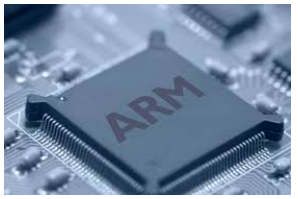
Variété

- **processeurs**
  - représentation des nombres
  - arithmétique
  - jeu d’instructions
- architecture matérielle
- périphériques

Problèmes

Variété

- **processeurs**
  - Intel/AMD
  - ARM
  - MIPS
  - Cell
  - SuperH…
  - + des tas plus ou moins exotiques, en particulier dans l'embarqué…
  - + GPU
- architecture matérielle
- périphériques



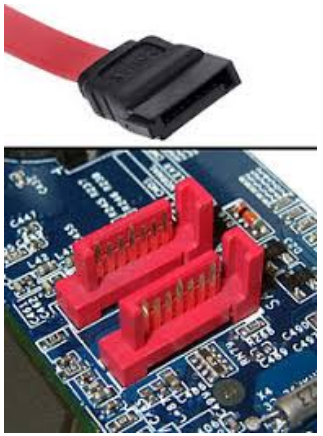
Problèmes

- Variété
  - processeurs
  - **architecture matérielle**
    - bus
    - mémoire
  - périphériques

Problèmes

Variété

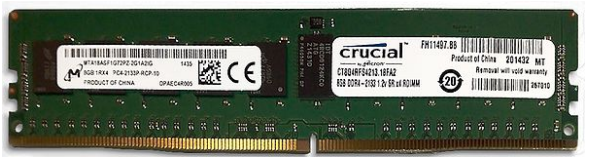
- processeurs
- **architecture matérielle**
  - **bus**
    - ISA, PCI, SATA, SCSI, Thunderbolt…
  - mémoire
- périphériques



Problèmes

Variété

- processeurs
- **architecture matérielle**
  - bus
  - **mémoire**
    - SRAM, SDRAM, DDR, ROM, EEPROM, Flash…
  - périphériques



Problèmes

Variété

- processeurs
- architecture matérielle
- **périphériques**
  - connectique
  - physique

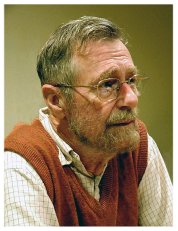
Problèmes

Variété

- processeurs
- architecture matérielle
- **périphériques**
  - mais si, vous savez, le bon «driver»
  - les màj incessantes…

Problèmes

*«The major cause of the **software crisis** is that machines have become several orders more powerful!*
To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem.
In this sense the electronic industry has not solved a single problem, it has only created them, it has created the problem of using its products.»



Edsger Dijkstra, 1972, **1972** ? **1972** ???

Besoins

L'informatique est pervasive

- elle envahit de très nombreux pans de notre vie
  - ordinateur, tablette, smartphone,…
  - voiture, avion, hoverboard, vélo
  - machine à laver,…
  - montre, GPS
  - industrie, administration,…

Les algorithmes sont partie intégrante de nos vies

- au moins dans les pays développés

Besoins

L'informatique est pervasive

- En regardant autour de moi depuis mon canapé, je viens de compter
  - 3 ordinateurs + 1 imprimante
  - 3 smartphones +2 tablettes
  - 1 téléviseur
  - 1 box ADSL + 1 box TV + 1 home cinéma
- Je parie que les 13 appareils ont un OS (même primitif pour TV et HC, en tous cas ils supportent la mise à jour de quelque chose…)
- Si je prend les transports, j'ai un écran dans le bus avec informations, un GPS à bord, des écrans d'informations à la gare, des bornes de validation, des bornes de rechargement, etc
  - Combien d'OS ? Combien de processeurs ?

Besoins

piloter/maintenir un fonctionnement

interagir avec le monde

manipuler des données

assurer sécurité/sureté

Besoins

- Assumer la paresse naturelle
  - de l'informaticien ?
  - sinon à quoi serviraient les machines ?
- Une devise : réutiliser
  - le graal des informaticiens
    - se décline en portabilité, inter-opérabilité, pérennité (le «développement durable» informatique)
- Éviter de réécrire/refaire
  - gain de temps
  - conservation des correctifs précédents
  - hummmm… jusqu'à ce que l'on décide de repartir à zéro (changement de langage, nouvelle conception, etc)

La loi de Moore

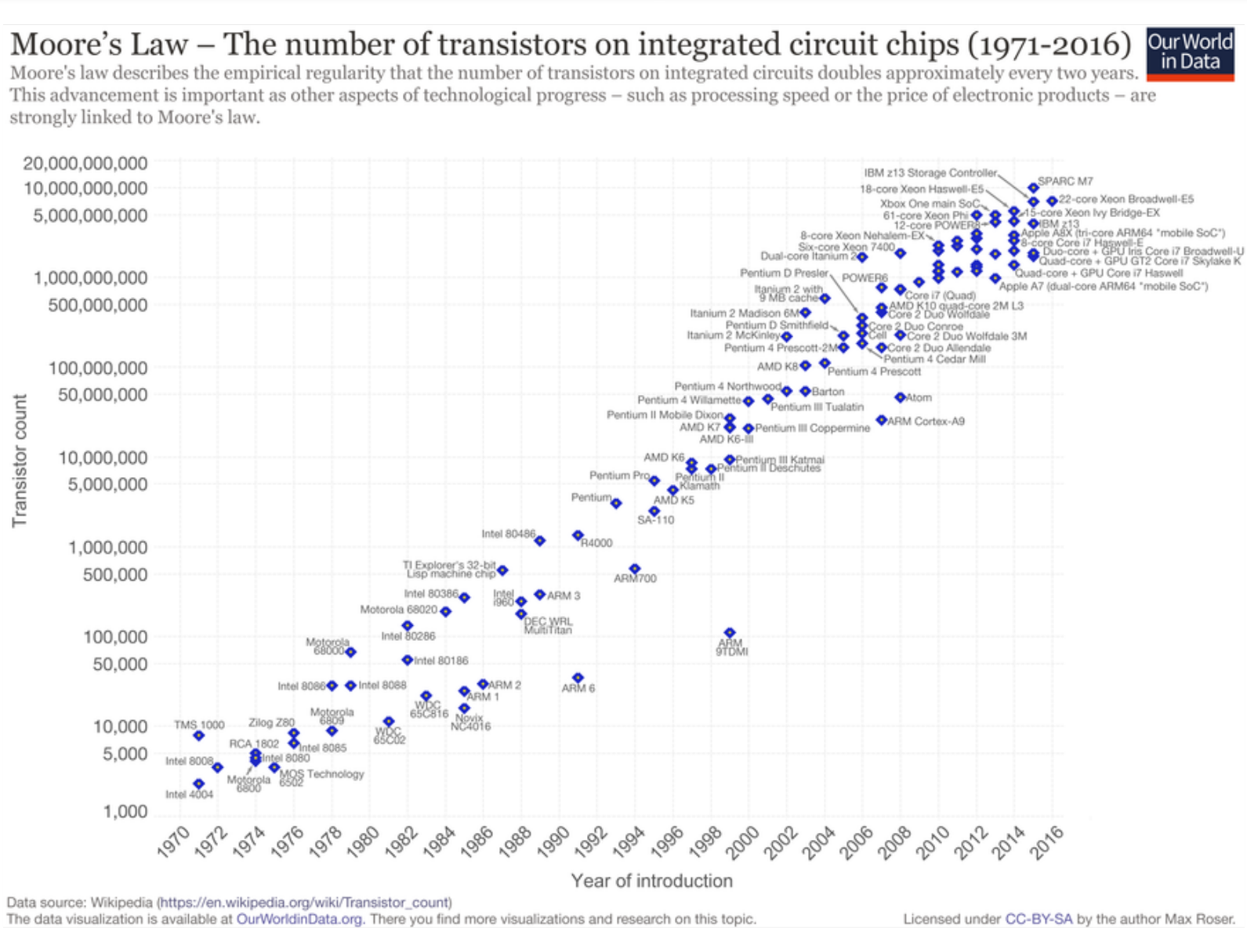
Au sens strict c'est une conjecture

> Le nombre de transistors sur un circuit intégré de surface constante double tous les deux ans
>
> Gordon Moore

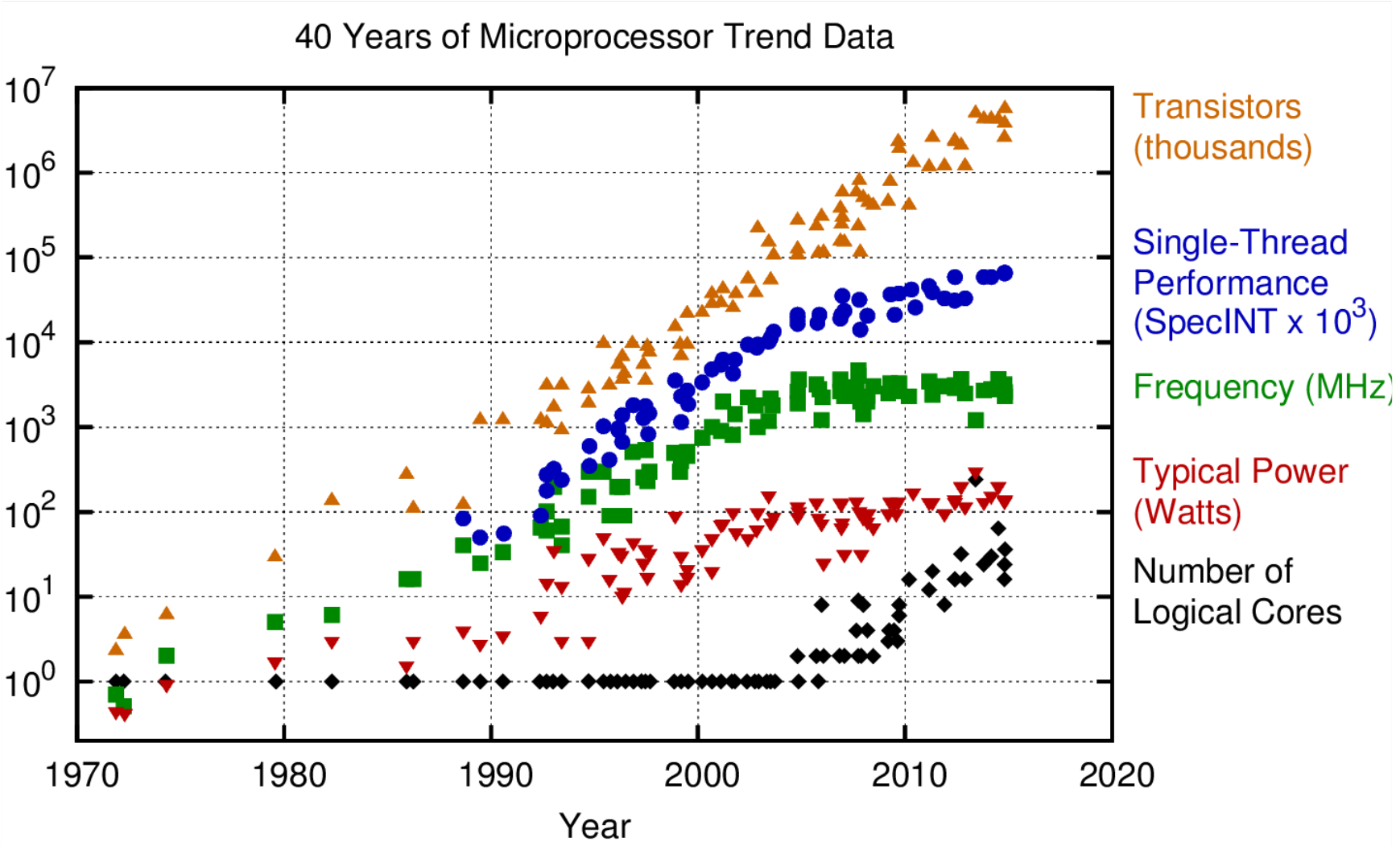
Il existe une seconde loi

> le coût de fabrication suit aussi une loi exponentielle

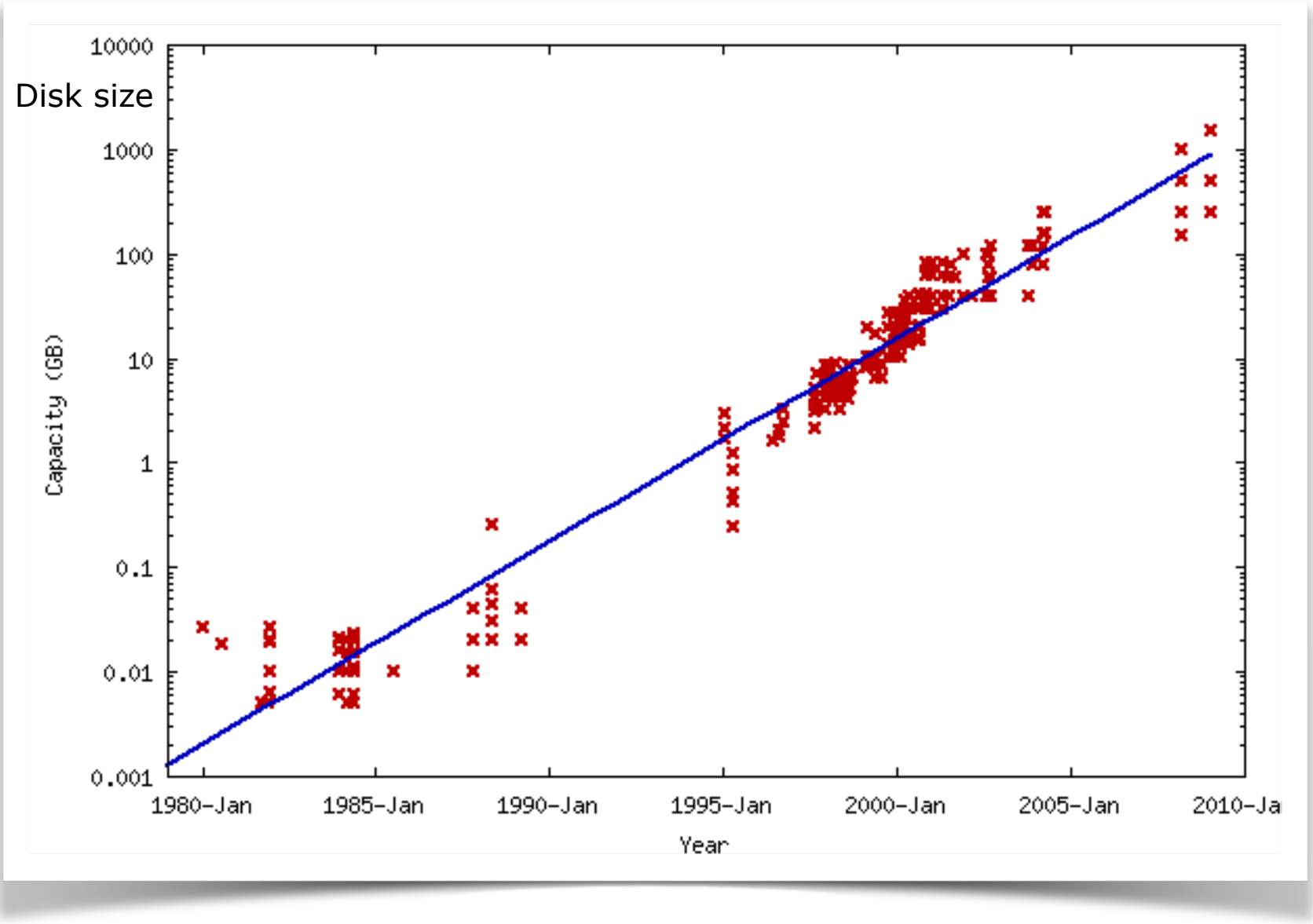
La loi de Moore



La loi de Moore



La loi de Moore



La loi de Moore

Elle oblige à faire évoluer les systèmes d'exploitation

- plus de puissance de calcul
- plus de mémoire
- plus de disque

Les OS ont énormément évolué ces 40 dernières années

- sans que ces évolutions soient nécessairement visibles (vive l'abstraction) aux utilisateurs

La loi de Moore

Elle possède néanmoins ses limites

- 2011 - transistor à 1 électron
- 2012 - transistor à un seul atome

La puissance continue à croître sans être nécessairement liée à l'intégration…

- augmentation du nombre de cœurs
- hyperthreading
- augmentation générale des capacités mémoire (volatile ou permanente)
  - cloud ?
- abandon des disques dur pour le SSD
  - encore de nouveaux OS…

La loi de Moore

Aux origines, un processeur = une seule unité de calcul

Architecture Multiprocesseur

plusieurs processeurs afin d'augmenter le puissance

Architecture Unifiée

plusieurs unité de calcul de nature différente (CUDA)

Architecture HyperThreadée

possède la faculté de basculer d'un calcul à un autre facilement

un seul processeur qui peut-être vu comme plusieurs

Architecture Multicœur

un cœur est une unité de calcul

sur une même puce plusieurs unités de calcul

➥ encore des nouveaux systèmes d'exploitation…

La loi de Moore

Sur ma machine Macbook Pro (2017)

- CPU i7-4870HQ
- 4 cœurs - 8 Threads

```
[host-36-29:~] sysctl hw.physicalcpu
hw.physicalcpu: 4
[host-36-29:~] sysctl -n hw.activecpu
8
[host-36-29:~]
```

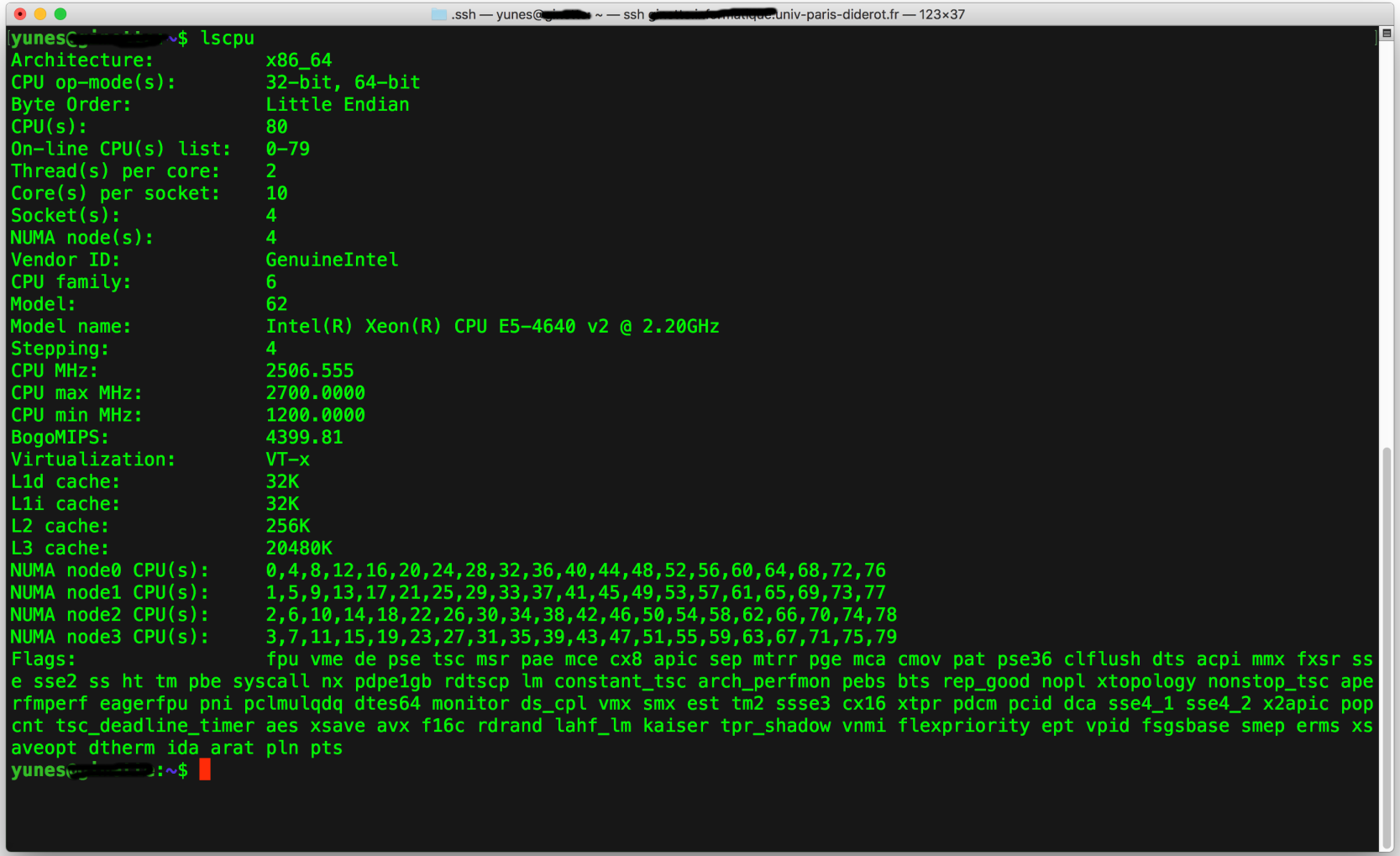
La loi de Moore

Sur une machine virtuelle

```
parallels@ubuntu:/proc/341/ns$ lscpu
Architecture :        x86_64
Mode(s) opératoire(s) des processeurs :32-bit, 64-bit
Boutisme :            Little Endian
Processeur(s) :       2
Liste de processeur(s) en ligne :0,1
Thread(s) par cœur : 1
Cœur(s) par socket : 2
Socket(s) :           1
Nœud(s) NUMA :        1
Identifiant constructeur :GenuineIntel
Famille de processeur :6
Modèle :              70
Nom de modèle :       Intel(R) Core(TM) i7-4870HQ CPU @ 2.50GHz
Révision :            1
Vitesse du processeur en MHz :2494.226
BogoMIPS :            4988.45
Constructeur d'hyperviseur :KVM
Type de virtualisation :complet
Cache L1d :           32K
Cache L1i :           32K
Cache L2 :            256K
Cache L3 :            6144K
Cache L4 :            131072K
Nœud NUMA 0 de processeur(s) :0,1
Drapaux :             fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2
ss ht syscall nx rdtscp lm constant_tsc nopl xtopology nonstop_tsc pni pclmulqdq ssse3 fma cx16 pcid sse4_1 sse4_2 x2api
c movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand hypervisor lahf_lm abm invpcid_single ibrs ibpb stibp kaiser
 fsgsbase tsc_adjust bmi1 avx2 smep bmi2 invpcid xsaveopt dtherm arat pln pts
parallels@ubuntu:/proc/341/ns$
```

La loi de Moore

Sur une machine dans mon Université



Système d'exploitation

Ensemble de programmes permettant de faire fonctionner et contrôler un appareil informatique

- offre une suite de services généraux
- facilite la création de logiciels
- sert d'intermédiaire entre ces logiciels et le matériel
- apporte commodité, efficacité, capacité d'évolution sans remettre en cause l'existant

Système d'exploitation

fournit un environnement pratique pour l'exécution de tâches

assure l'ordonnancement de tâches de calcul

offre un partage raisonné des resources

garanti autant que possible un fonctionnement optimal de la machine

Système d'exploitation

Wikipédia liste les types suivants :

- mono et multitâche
- mono et multiutilisateur
- distribué
- templaté
- embarqué
- temps-réel
- library

Famille de systèmes

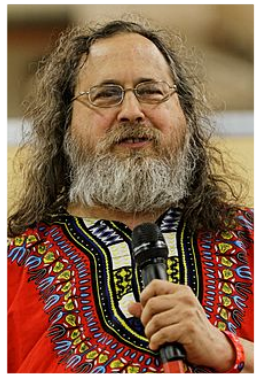
- Plutôt que de système on parle plus souvent de famille
  - Linux, n’est pas un système en tant que tel, cela désigne une variété de systèmes
    - dont le noyau (le kernel) est commun mais constitue déjà une famille
  - idem Unix, Windows, macOS

Famille de systèmes

Dans chaque famille on trouve des variantes :

- architecturales
  - ordinateurs portables, de bureau, serveur, tablette, téléphone, console de jeu, système embarqué
  - processeurs et cœurs, NUMA, SMP, etc
- écosystèmes
  - libre, ouvert, propriétaire
- usage
  - mono/multi utilisateur
  - interface graphique (kde, gnome…)

Libre vs Ouvert



Free Software *vs* Open Source

Richard Stallman lui-même :

«Open source is a development methodology; free software is a social movement.»

Le mouvement libre/ouvert est un mouvement de très grande importance

C'est un mouvement préoccupé par l'accès à la connaissance

OS = VM

Bien avant ce que l'on appelle aujourd'hui des machines virtuelles

> Wikipédia excerpt : «***VM*** *(often:* ***VM/CMS****) is a family of IBM virtual machine operating systems used on IBM mainframes System/370, System/390, zSeries, System z and compatible systems, including the Hercules emulator for personal computers.»*

VM/CMS date de 1972

OS = VM

Un système d'exploitation est une machine virtuelle :

- utilisation plus facile par construction d'abstractions
  - fichiers
  - IG
  - applications simultanées
  - utilisateurs simultanés
  - *invisibilité* des limites matérielles
  - etc.
- on raisonne en étant affranchi des détails
  - *«ma distribution Ubuntu est la même sur mon ordinateur de tel modèle ou tel autre»*

Fonctions d'un OS

Gestion des processeurs

Gestion des exécutions

Gestion des interruptions

Gestion des modes opératoires

Gestion de la mémoire

Gestion des périphériques, entrées/sorties et fichiers

Gestion de la communication

Gestion du réseau

Gestion des utilisateurs

Gestion du système

Fonctions d'un OS

Gestion des exécutions

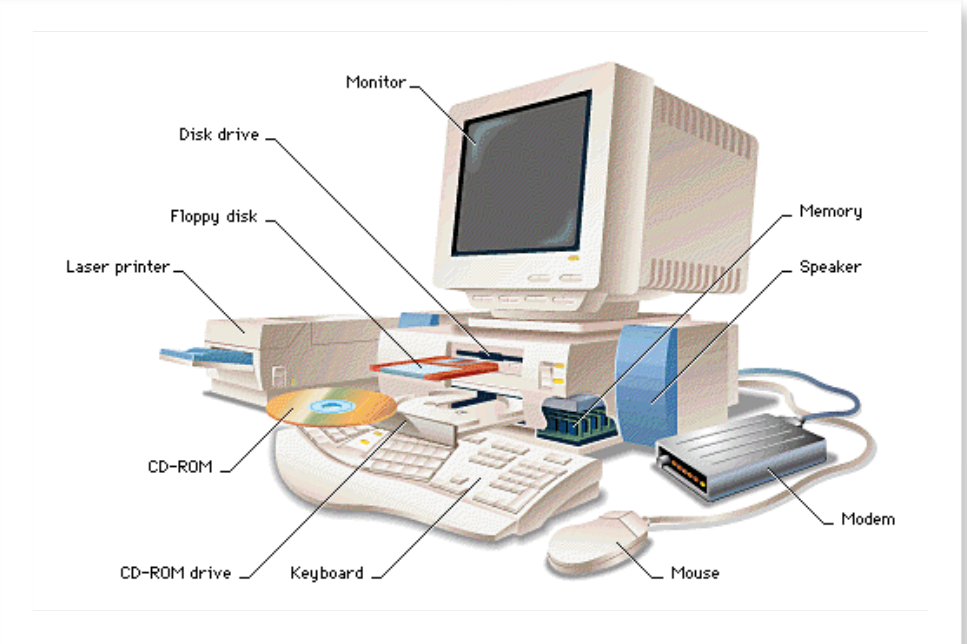
- création, destruction, contrôle
- ordonnancement, synchronisation
- concept d'application ? multitâche ? multithread ?

Fonctions d'un OS

Gestion des interruptions

réaction à l'environnement

souris, clavier, réseau, disque, etc



Fonctions d'un OS

- Gestion des modes opératoires
  - il existe en général plusieurs modes de fonctionnement de la machine, certains ayant accès à plus ou moins de fonctions de base
  - dans de nombreux systèmes, deux modes
    - utilisateur (il ne voit que la VM)
    - noyau (il voit la φM - Physical Machine)

Fonctions d'un OS

- Gestion de la mémoire
  - allocation, libération, contrôle d'accès
  - mémoire virtuelle : pagination, segmentation, swap
  - partage

Fonctions d'un OS

Gestion des périphériques, entrées/sorties et fichiers

- de très nombreuses variétés : audio, vidéo, clavier, souris, USB, Wifi, capteurs, etc
- pilotes (drivers)
- protocoles
- disques physiques, logiques
- fichiers, répertoires
  - création, destruction, modification, contrôle d'accès

Fonctions d'un OS

- Gestion de la communication
  - intra
    - signaux, mémoire partagée, messages
  - extra
    - réseau

Fonctions d'un OS

- Gestion des utilisateurs
  - création, suppression
  - gestion des droits
- Gestion du système
  - configuration
  - mise au point
  - maintenance
  - surveillance
- Un métier : administrateur système

Aspects non fonctionnels

Réactivité

Sûreté de fonctionnement

Tolérance aux pannes

Gestion de l’énergie

Réactivité

Un système temps-réel doit réagir aux stimuli de son environnement dans des délais impartis propres au domaine

- système de freinage, contrôle d'automatisme, vidéo, acquisition données, guidage fusée, etc

Un système temps-réel ne doit pas seulement fournir des données correctes mais en temps et en heure (beaucoup plus difficile)

- résultats corrects si fournis en temps
- les résultats tardifs sont faux!

Contraintes en temps-réel

Dures (hard realtime) ou strictes

- une contrainte est dure si de son non-respect résulte une catastrophe (Kopetz 1977)
  - centrale nucléaire, avion, pacemaker…

Molles (soft realtime) ou souples ou lâche

- les autres…
  - vidéo, audio, jeu en réseau…
  - en général on considère qu'un rattrapage à court terme suffira à compenser (Ex. : lag vidéo…)

difficile!

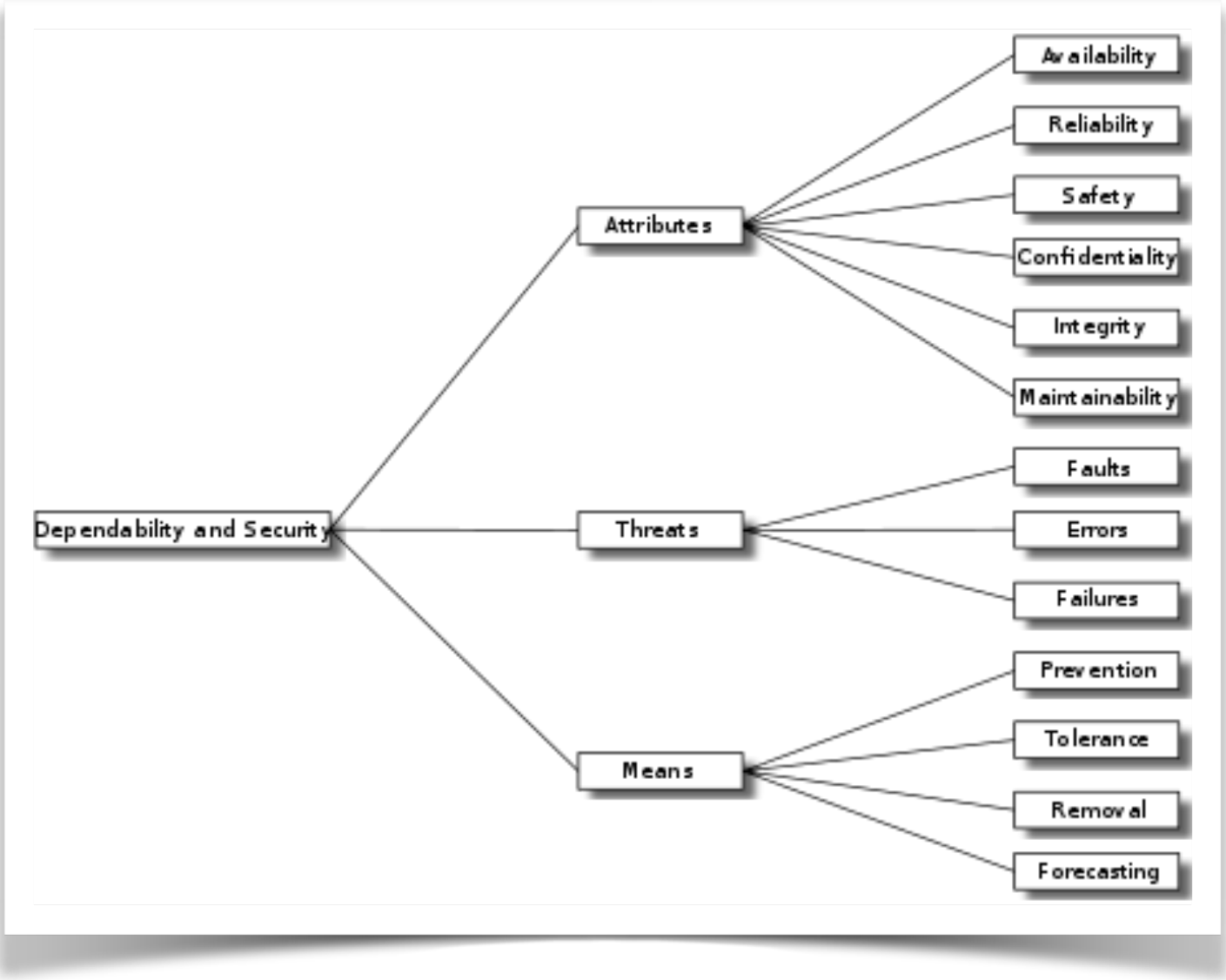
Sûreté de fonctionnement

Sûreté (dependability)

- Disponibilité (availability)
- Fiabilité (reliability)
- Sécurité (safety)
- Confidentialité (confidentiality)
- Intégrité (integrity)
- Maintenabilité (maintenability)

difficile!

Sûreté de fonctionnement



Sûreté de fonctionnement

Disponibilité (availability)

probabilité que le système soit prêt à assurer un service à t

vous savez… la panne de votre fournisseur d'accès Internet ou mobile, frigo, voiture…

Haute disponibilité

> Five nine

Très haute disponibilité

> Six nine

Ultra haute disponibilité

> Seven nine

Disponibilité	Indisponibilité par an	Indisponibilité par mois	Indisponibilité par semaine	Indisponibilité par jour
One nine 90 %	36j	73h	16h	2h
Two nine 99 %	3,5j	7h	1,5h	14m
Three nine 99,9%	9h	44m	10m	1,5m
Four nine 99,99%	1h	4,5m	1m	8,5s
Five nine 99,999%	5m	26s	6s	0,8s
Six nine 99,9999%	31s	2,5s	0,6s	0,08s

Sûreté de fonctionnement

- Disponibilité (availability)
  - une heure d'indisponibilité
    - système financier : perte exprimée en dizaines de millions de $
    - système de paiement par carte : millions de $
    - système de réservation de transport aérien : des centaines de milliers de $
- Vous voyez le problème ?

Sûreté de fonctionnement

Fiabilité (reliability)

probabilité que le système fonctionne correctement à t+Δt, s'il fonctionnait correctement à t

Si ma voiture démarre, est-ce qu'elle va rouler toute la journée ?

Il existerait de nombreuses machines fonctionnelles depuis 30 ans, peut-être plus…

Sûreté de fonctionnement

Sécurité (safety)

- absence de conséquence externe catastrophique

Si mon système s'arrête, est-ce que des gens meurent ?

- pacemaker, avionique, etc.

Sûreté de fonctionnement

Confidentialité (confidentiality)

absence de fuite d’information

La cryptographie à la rescousse ?

Sûreté de fonctionnement

Intégrité (integrity)

- absence d'altération compromettant
  - la cohérence
  - l'honnêteté du service

Le système vous vole t-il votre numéro de carte bleue ?

Ne commande t-il pas 1.000.000 de pizza au lieu d'une les vendredi 13 ?

Sûreté de fonctionnement

Maintenabilité (maintenability)

aptitude à la remise en état

Le système est-il réparable ?

Combien de temps me faut-il pour réparer ?

Résistance aux pannes

- Processeur
  - multi-cœur, intégration élevée
    - 1996 une erreur par rayonnement cosmique/mois/256Mo de mémoire (étude IBM)
    - 2010 une erreur cosmique/jour/4Go!
    - 10 fois plus d'erreurs en montagne qu'à la mer
    - 300 fois plus en avion!
    - Aujourd'hui en avion votre portable est **b**ouché à coup sûr toutes les 5 heures
- Embarquement
  - spatial, environnements hostiles, militaire, etc

Résistance aux pannes

- Super-calculateurs
  - des milliers, centaines de milliers, millions de processeurs! La probabilité que certains tombent en panne chaque jour est plutôt élevée!
    - 6/2018
      - Summit > 2.000.000 cœurs
      - Sunway > 10.000.000 cœurs
      - Sierra > 1.500.000 cœurs
      - Tianhe > 5.000.000 cœurs
    - Note : toutes (ou presque) ces énormes machines utilisent Linux…
      - source TOP500.org

Résistance aux pannes

Question subsidiaire : qui possède la plus grosse puissance de calcul ?

La Chine! très loin devant

États-Unis, très très loin devant

Japon, Royaume-Uni, Allemagne, France…

La Chine en possède plus que tout le reste réuni!

Attention ce sont des estimations…

La France possède (par exemple) le HPC de Bruyères-le-Châtel

http://www-hpc.cea.fr

Résistance aux pannes

- Cloud
  - des milliers/millions, de machines, disques, équipements réseau
  - > 2.500.000 serveurs (estimation 2016)
    - donc 2Eo (Exaoctet, trillion!) de stockage
      - pas loin de la frontière des 64 bits d'adressage!
  - > 100.000 serveurs (estimation 2012)
    - probablement beaucoup plus aujourd'hui, mais combien ?
    - 200Po (Petaoctet, billiard!, estimation 2017)

Résistance aux pannes

Cloud

Microsoft > 1.000.000 serveurs (Ballmer, 2013)

qu'en font-ils ? 😮

amazon inconnu, Ballmer a dit moins qu'eux

? On dit qu'ils atteignent le Yottabyte (quadrillion)…

c'est probablement faux, mais peut-être le Zettabyte (trilliard) ?

c'est seulement 15Po…

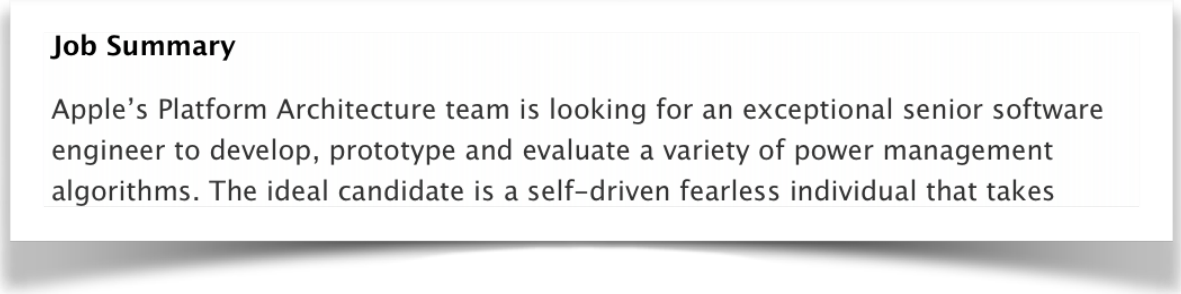
Résistance aux pannes

Le Crédit Agricole aurait 27M de clients

- avec 1 transaction bancaire de 1024o/jour/client
- ~10To de données/an…

Gestion de l'énergie

une problématique récente et cruciale!



conserver un fonctionnement raisonnable tout en économisant l'énergie

- dfs, dynamic frequency scaling
- power saving (cpu certes, mais écrans, disques, clavier éclairé, RAM, etc)

Transparence

Un système d'exploitation rend les choses transparentes à l'utilisateur

C'est transparent à l'utilisateur ⇔ Il ne voit rien,

donc c'est opaque

Hétérogénéité

- fichiers/répertoires : la plupart des utilisateurs ne sont en rien conscient des différences…
- marque/modèle
- etc

Transparence

Un système d'exploitation rend les choses transparentes à l'utilisateur

C'est transparent à l'utilisateur ⇔ Il ne voit rien, donc c'est opaque ⇔ Boîte noire

Transparence

- Localisation
  - L'utilisateur se préoccupe assez peu de l'endroit exact où ses fichiers (données) sont stocké(e)s
    - local ou non : Clouds
      - un ami à moi : « c'est là-haut » (dans les nuages)
    - Mac OS Spotlight, Windows search, Ubuntu dash…
    - le mail… (Savez-vous où sont vos stockés vos mails ?)

Transparence

Migration

- Cloud, serveurs qui se déplacent sans qu'on le sache

Distribution

- Les données peuvent être fragmentées et distribuées à différents endroits

Transparence

Réplication (tolérance aux pannes et performances)

- Savez-vous quel serveur répond lorsque vous faites une recherche google ?
- Cloud
- Vision unique

Transparence

- Caches
  - Données rapatriées au fil de l'eau au plus près de celui qui réclame
    - Transparent ? Vraiment ? Humm…

Transparence

Un graal difficile à réaliser en pratique

Pas toujours aussi transparent

- données perdues lors de crashes
- incohérence de vision entre machines/applications
  - murs facebook ?
- consistence ?
- outils différents pour manipuler les objets
  - cp *vs* scp *vs* ftp
- débit - latence - asynchronisme

Problème classique : le ping-pong (effet désastreux)

Le mythe du SSI

- Single System Image
  - le graal du « distribué » : l'utilisateur ne voit qu'une seule machine
    - ultimement la même machine pour tout le monde
      - en arrière-plan : l'unification des ordinateurs du monde en une même machine
      - la matrice ?

Le mythe du SSI

Les propriétés attendues d'un SSI

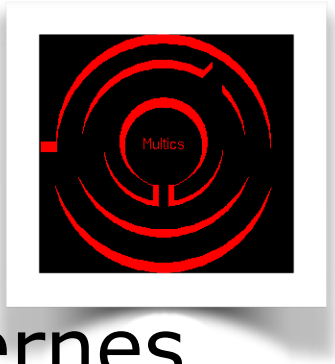
- migration (process migration)
- point de contrôle (process checkpointing)
- un seul espace de processus (single process space)
- un seul espace de stockage (single root)
- un seul espace de périphériques (single I/O space)
- un seul espace de communication (single IPC space)

Le mythe du SSI

Internet/Cloud

un SSI ?

Multics



La pierre angulaire des systèmes modernes

MULTiplexed Information and Computing Service

- création conjointe du MIT, de Bell et General Electrics 1964

Idée : distribuer du «computing» comme on distribue du téléphone ou de l'eau (un SSI ?)

- un super Minitel pour la ville de Boston (une prise dans chaque pièce ?)

Multics

Caractéristiques

- Écrit en PL/1 + assembleur
- Temps partagé
- Shell
- Traitement de texte
- Terminaux distants
  - computer grid?

Multics

Technologie de l'OS (ce que l'on trouve dans tous les OS aujourd'hui)

- anneaux
- multitâche préemptif
- mémoire virtuelle / segmentation
- système de fichiers hiérarchique
- intégrité/sécurité

Multics

N’a jamais vraiment atteint son but

Est resté disponible un peu partout dans le monde (en particulier universitaire) jusqu’à ~1990

Multics

Trop complexe à mettre en œuvre

Trop complexe à utiliser

Une influence immense sur tous les autres OS

Unix est arrivé…

From Multics to

Unix

Thompson & Ritchie (du projet Multics)

développèrent Unics (dans un coin)

en s'inspirant de Multics

tout en en enlevant la lourdeur

idée fixe : rester simple à tous les niveaux

Unics est né puis renommé en Unix

Unix

- Un système de fichiers hiérarchique
- Un multitâche préemptif
- Un interpréteur de commande (shell)
- Une vision unificatrice
  - tout est fichier
    - ce graal n’est quasiment atteint que par Plan 9
  - tout est écrit en C (évolution de B)
    - sauf une portion ridicule du noyau

Unix

1970 — 1980

Le cœur d'Unix tel qu'on le connaît aujourd'hui dans sa philosophie initiale

portable

ouvert aux contributions

1980 —

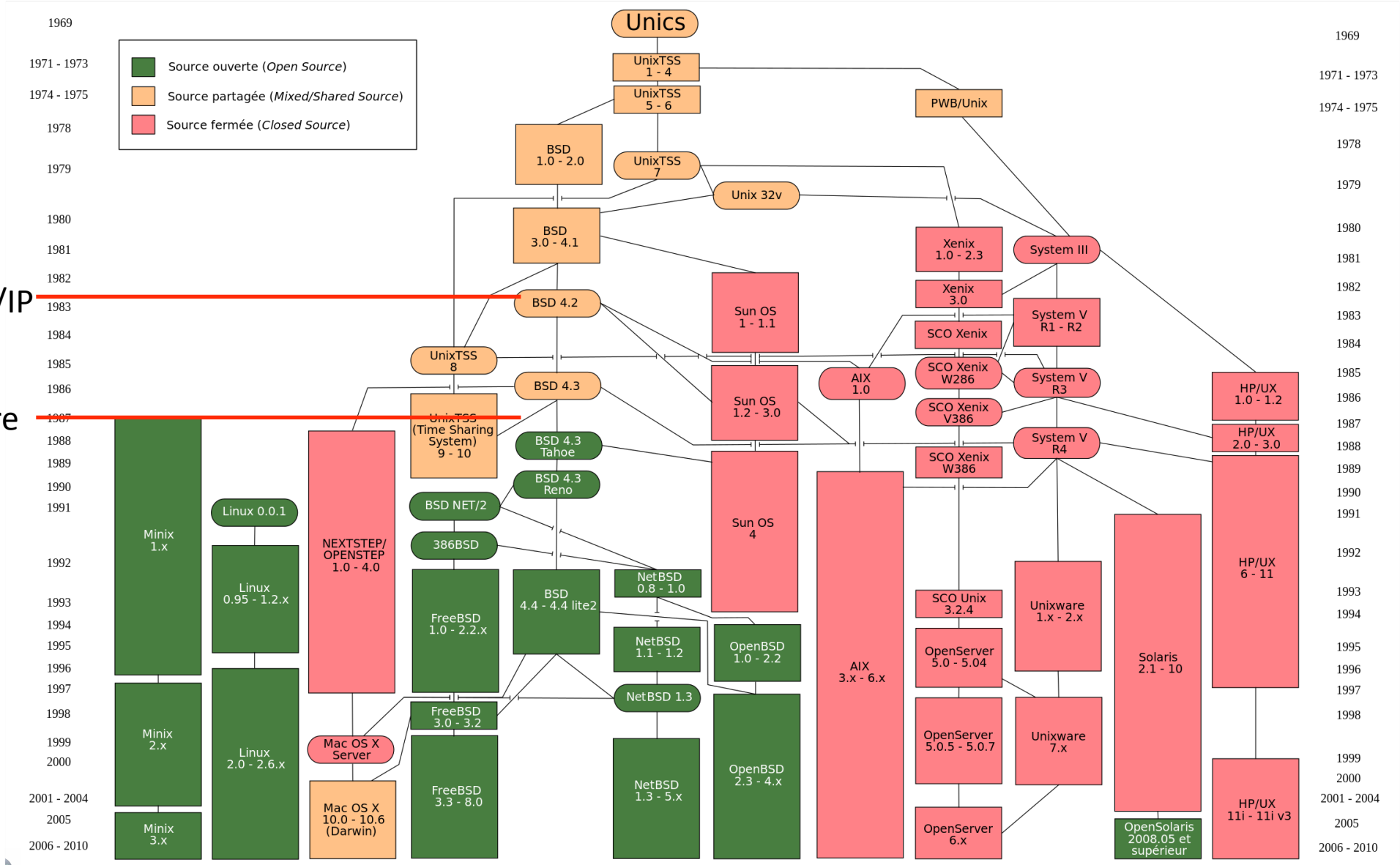
Unix a tout contaminé et on le retrouve un peu partout, hormis les gros systèmes (des survivants) et le monde Microsoft (encore qu'il y a bien eu des tentatives Unixiennes de leur côté aussi)

regain attendu : couche Windows par-dessus Linux ?

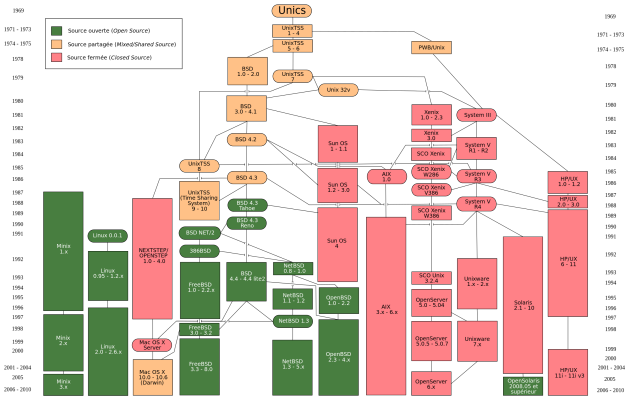
Famille Unix

TCP/IP

Libre



Famille Unix



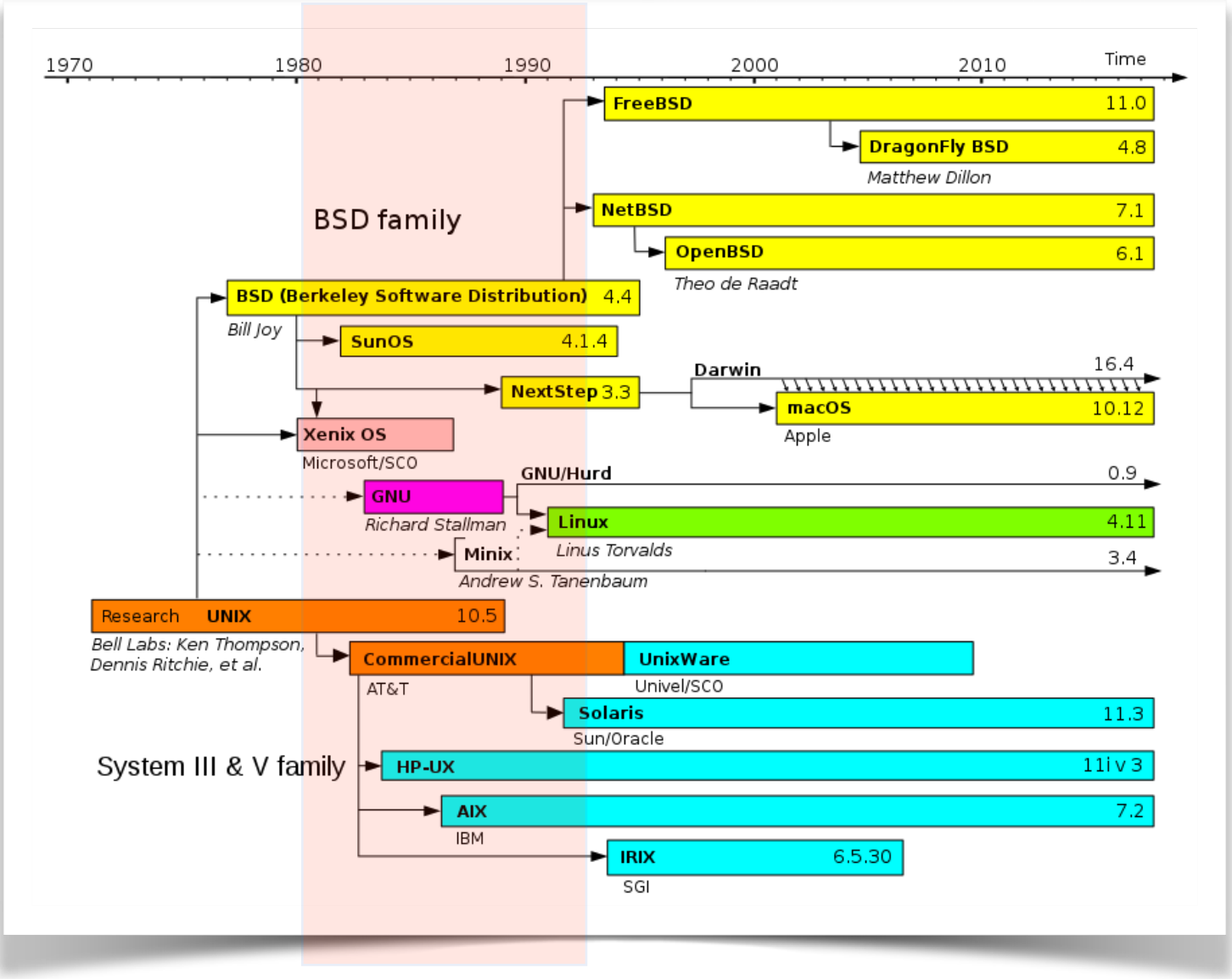
Ne pas se méprendre

- ce diagramme est largement incomplet
- la famille contient probablement des centaines de descendants de l'initial

Les deux grandes sous-familles sont

- System V
- BSD

Unix Wars


1970
1980
1990
2000
2010
Time
FreeBSD 11.0
DragonFly BSD 4.8
Matthew Dillon
BSD family
NetBSD 7.1
OpenBSD 6.1
Theo de Raadt
BSD (Berkeley Software Distribution) 4.4
Bill Joy
SunOS 4.1.4
Darwin
16.4
NextStep 3.3
macOS 10.12
Apple
Xenix OS
Microsoft/SCO
GNU/Hurd
0.9
GNU
Richard Stallman
Linux 4.11
Linus Torvalds
Minix
3.4
Andrew S. Tanenbaum
Research UNIX 10.5
Bell Labs: Ken Thompson, Dennis Ritchie, et al.
CommercialUNIX
UnixWare
AT&T
Univel/SCO
Solaris 11.3
Sun/Oracle
System III & V family
HP-UX 11i v 3
AIX 7.2
IBM
IRIX 6.5.30
SGI

Unix Wars

La guerre des Unix pour la standarisation

- 80’ - ATT «SV is the standard»
- 80’ - BSD «4.2 > V»
- 1984 - X/Open standard
- 1988
  - OSF (Open Software Foundation)
  - UI Unix International
  - IEEE POSIX.1
- 1993 OSF+UI ☛ COSE (X/OPEN)
- 1996 X/OPEN+ OSF ☛ Open Group (SUS - Single Unix Specification) Contrôle POSIX

Linux est à part et ne cherche pas vraiment la standardisation (il ne l’est pas sur divers points), peut-être a être le standard…

La famille System V

- La famille originelle
  - dont descendent AIX (IBM), HP/UX (Hewlett-Packard), Solaris (Sun ☛ Oracle), UnixWare (USL+Novell)
- Code fermé et propriétaire
  - permet un bon contrôle du marché
  - pose des difficultés d'adaptation en concurrence avec l'innovation ouverte (BSD)

La famille System V

Les innovations

- IPC
- Streams
- RFS
- Shared Libraries

La famille BSD

Après un retentissant procès anti-trust l'université de Berkeley

- récupère une licence gratuite
- premier portage sur machine 32bits en 79
- csh
- tcp/ip
- sockets
- ffs

Robustesse…

BSDI créée pour commercialiser BSD en 1994

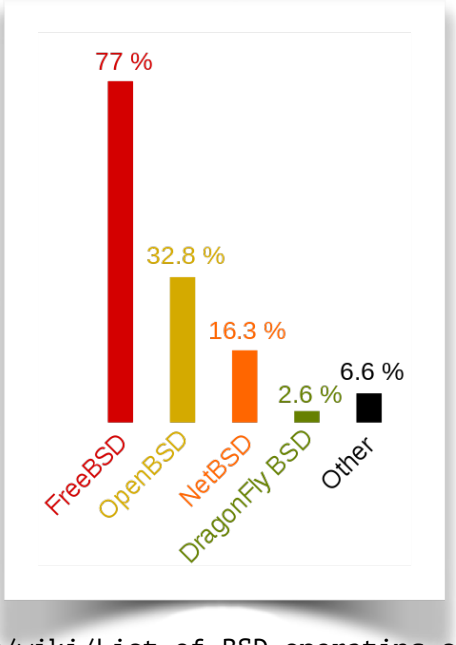
La famille BSD

BSD désigne formellement un système ayant vécu de 1977 à 1995, latest BSD 4.4

Depuis, BSD désigne génériquement les systèmes

- NetBSD
- FreeBSD
- OpenBSD
- DragonFly BSD

Darwin (aka OsX, MacOS) ?



https://en.wikipedia.org/wiki/List_of_BSD_operating_systems

La famille BSD

La famille BSD est parfois considérée comme la famille de meilleure qualité

- influence du modèle de développement ?
  - Open Source vs Free Software ?

Xenix

Microsoft & ATT puis SCO Santa Cruz Operation collaborent pour un portage sur Intel x86

- Xenix
  - File Locks
  - Semaphores
  - EMail sur UUCP

Un Unix à ne pas négliger, longtemps en tête des ventes des Unix!

- Et oui, Microsoft a été un acteur Unix important

SunOS ☛ Solaris

Un cas intéressant!

- SunOS est un BSD
- Solaris un System V

Des innovations intéressantes

- Yellow Pages
- NFS
- VFS
- automount

GNU/Hurd

Richard Stallman (un Torvalds plus extrême avant l'heure)

- souhaite créer un système complètement affranchi du secteur privé
  - de très nombreuses applications et briques diverses sont écrites dans cet esprit
- l'écriture du noyau Hurd est entreprise mais trop tard, Linux emportera le morceau…
  - à la grande fureur de Stallman

Linux

Né un peu par hasard et contre nature en 1991

- challenge : réécrire un OS tout seul
  - Linus Torvalds répond oui, voici…
    - attention sa réponse était largement incomplète mais méritait d'être complétée
      - son appel à contribution fût largement entendu et au-delà de ses espérances

Linux a envahi le monde des OS

- avec ses variantes propres

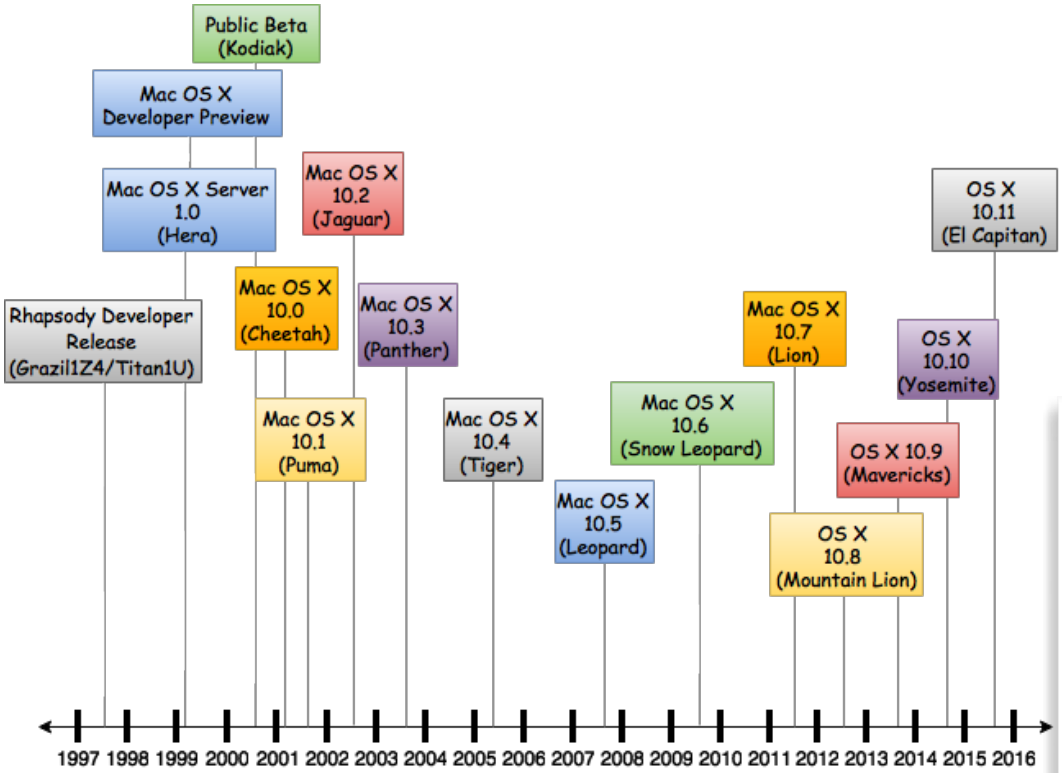
Mac OS X ☛ OS X ☛ macOS

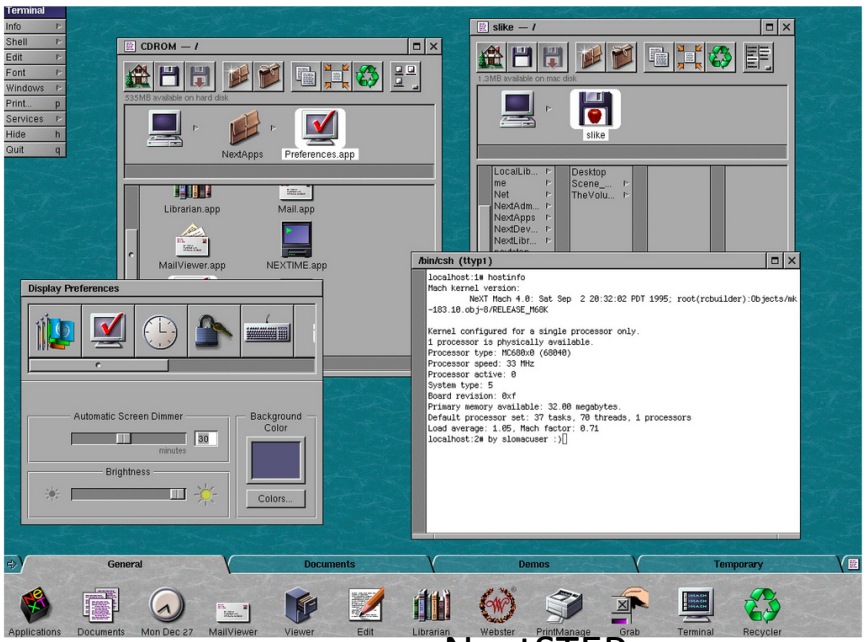
Les successeurs Unixiens du Mac OS classic

Une histoire intéressante :

- Steve Jobs mis à la porte d'Apple créé NeXT
  - NeXTSTEP
    - Mach + BSD + Display PostScript
  - WWW inventé sur NeXT par Tim Berners-Lee (CERN)
- Steve Jobs rappelé à la rescousse pour sauver Apple d'un désastre annoncé
  - classic is dead, NeXTSTEP devient Mac OS X
  - au passage Display PostScript devient Display PDF

Mac OS X ☛ OS X ☛ macOS





NextSTEP

Mac OS X ☛ OS X ☛ macOS

Toutes les innovations majeures de NeXTSTEP sont dans macOS

- l'ergonomie
- les outils
- …

Des dérivés pour IoT

- iOS, tvOS, watchOS

iOS

watchOS

tvOS

Mac OS X ☛ OS X ☛ macOS

- Rappel (important)
  - Apple est resté
    - un fabricant de machines et logiciels
    - un vendeur de machines et logiciels
- Une maîtrise totale de la chaîne
  - hormis le système Darwin (Open Source)
  - y compris la distribution des logiciels, etc
  - tout intégré
- De très nombreuses innovations orientées utilisateur

Le vainqueur est…

Linux a tout envahi

- serveurs, desktops, portables super-calculateurs
- mobiles
- équipements réseau
- voitures
- téléviseurs
- box internet
- etc

C’est quoi un Unix

un Unix ?

- ça dépend de ce que l’on appelle un Unix!
- si l’on regarde à quoi cela ressemble
- si on est strict dans la descendance, mais BSD n’est pas un Unix! Un BSD peut-être ?
- si Unix c’est POSIX, mais en ce cas de nombreux non Unix sont POSIX, y compris Windows qui possédait une API POSIX ou encore VMS…

Unix’s not dead

D’abord parce que de nombreux Unix non Linux continuent d’exister

Ensuite parce que MacOS est un Unix décoré

  un Darwin : MacOS(FreeBSD(Mach))

et puis il y a Linux

et puis la «philosophie» Unixienne a tout imprégné

En dehors d'Unix ?

Si Unix est majeur depuis les années 80

ce n'est pas le seul système existant!

Windows, certes

OS/2, QNX, BeOS, VMS, L4, Mach, Chorus, Minix, MS/DOS, OS/400, etc.

ou de très exotiques

Red Star OS (붉은별)

Les grands jalons technologiques

Transistors ☛ Miniaturisation

Écrans graphiques ☛ Ergonomie

μ-processeurs ☛ Scalable Super Computing

Réseau ☛ Partage des ressources, Cloud

Les grands jalons technologiques

Vos grand-parents n'ont vu (de leurs yeux) leur premier ordinateur que bien tard

Vos parents n'ont expérimenté Internet que tard

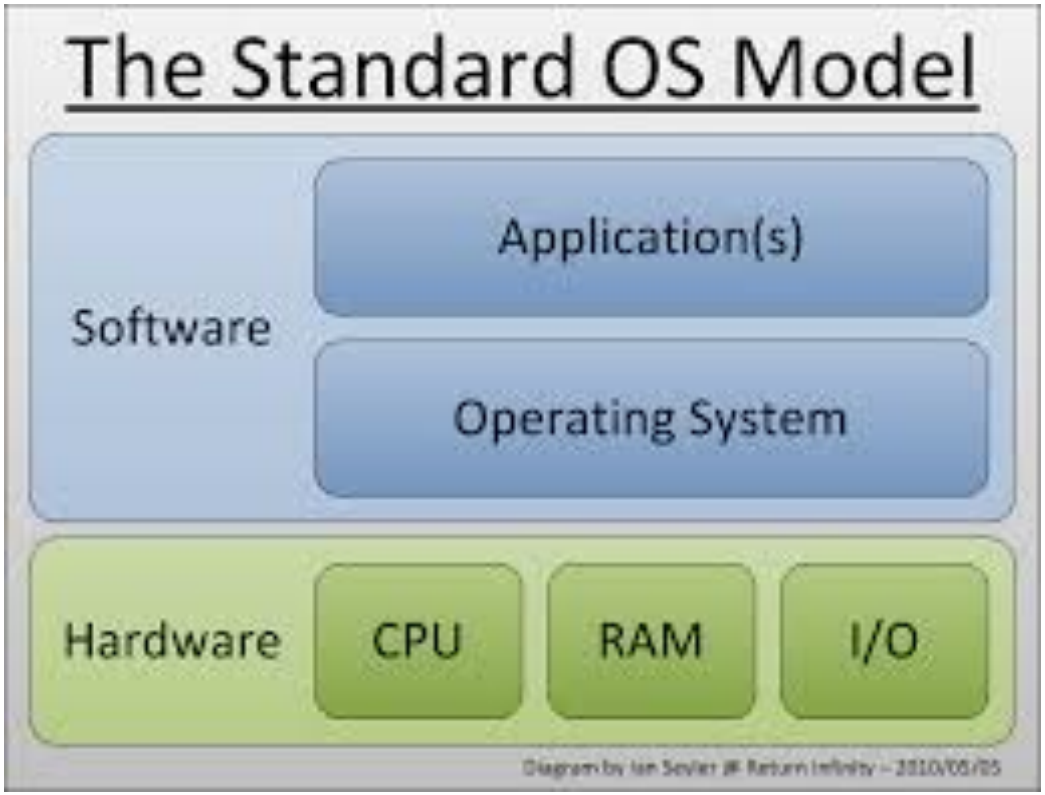
Vous avez vu apparaître les premiers smartphones

Que découvrirez-vous d'incroyable bientôt ?

Architecture système

Repose sur l'articulation de trois composants principaux :

- le matériel
- le noyau
- les applications



Architecture système

Ex. : macOS

macOS Architecture

Cocoa Application – Application User Interface responds to user events , manages app behavior.

App Kit									
Notification Center	Siri	Sharing	Full Screen Mode	Cocoa Autolayout	Popovers	Software Configuration	Accessibility	Apple Script	Spotlight

Media – Plays , records , editing audio-visual media , rendering 2D and 3D graphics.

AV Foundation	Core Animation	Core Audio	Core Image	Core Text	Open AL	Metal	Quartz
Audio playback, editing, analysis & recording.	2D rendering & animation. 3D Transformations.	Audio services for recording, playback and synchronization.	Fast image processing. Uses GPU based acceleration.	Handles Unicode fonts & texts.	Delivers 3D audio. High performance positional playbacks in games.	Portable 3D graphics apps & games. Imaging functions & effects.	macOS graphics, rendering support for 2D content. Event routing & cursor management.

Core Services – Fundamental Services for low level network communication , Automatic Reference Counting , Data Formatting , String Manipulation.

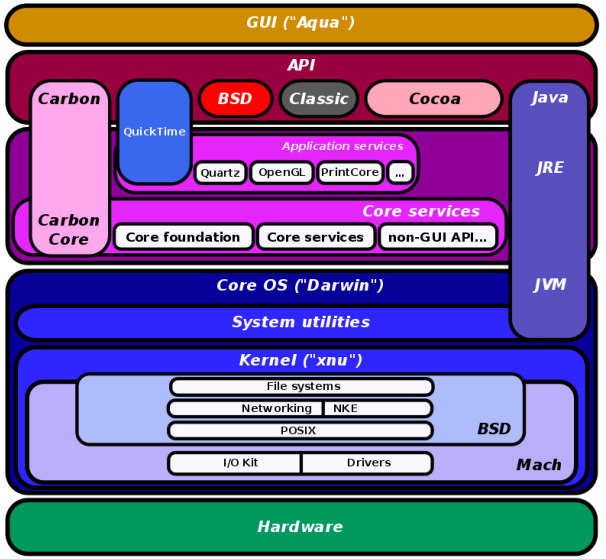
Address Book	Core Foundation	Quick Look	Security	Core Data	Foundation	Social	Webkit
Centralized database for contacts & groups.	Declares C based pro-grammatic interfaces. Data Types & Data Management.	Enables Spotlight & Finder to display thumbnail images.	User authentication, certificates & keys, authorization, keychain services etc.	Data model management & storage, undo/redo, validation of property values.	Swift framework for object behavior, inter-nationalization, data types & data mgmt.	Supports integration with social networking services.	Display HTML content in apps. Contains WebCore and JavaScript Core.

Core OS – Related to hardware and networking. Interfaces for running high-performance computation tasks on CPU or GPU.

Accelerate	Directory Services	Disk Arbitration	Metal	System Configuration
Accelerate complex operations, improve performance using vector unit, supports data parallelism, 3D graphic imaging, image processing.	Provides access to collected information about users, groups, computers, printers in a networked environment.	Notifies when local or remote volumes are mounted and unmounted.	Makes the high-performance parallel processing power of GPUs available to general purpose computing.	Provides access to current network configuration information. Determines reachability of remote hosts. Notifies about changes in network.

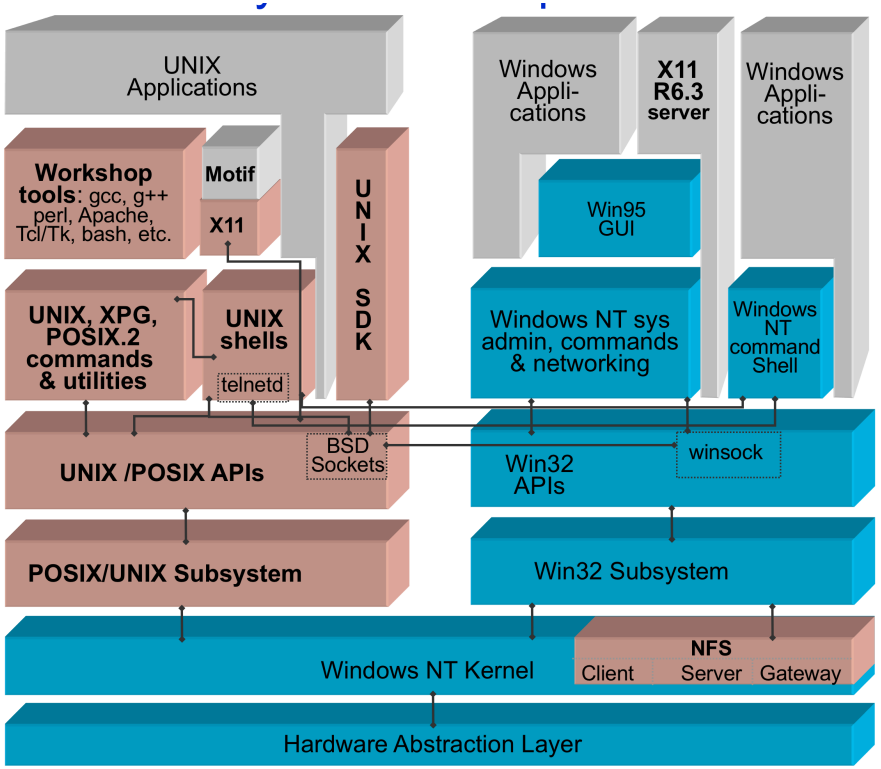
Kernel & Device Drivers – Device drivers & BSD Libraries , low level components. Support for file system security , interprocess communications.

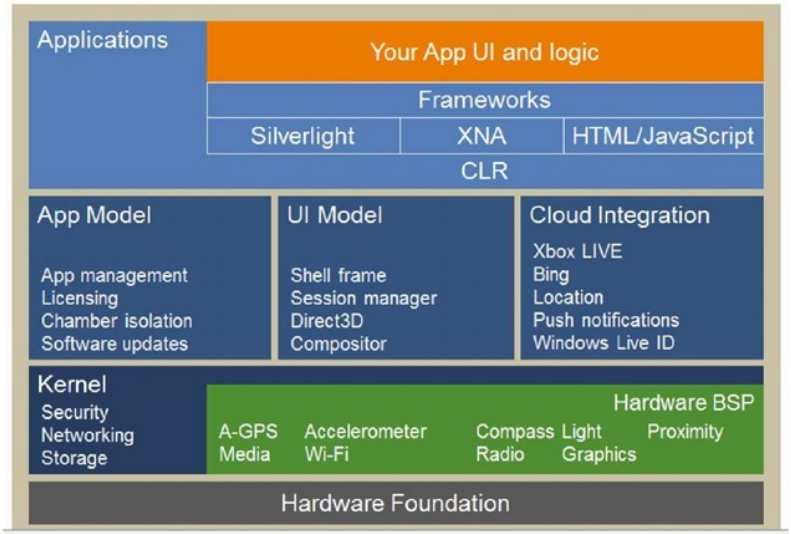
BSD	File System	Mach	Networking
Provides basis for file systems and networking facilities, POSIX thread support, BSD sockets.	Supports multiple volume formats (NTFS, ExFAT, FAT, HFS+, APFS etc.) & file protocols (AFP, NFS etc.)	Protected Memory, Preemptive Multitasking, Advanced Virtual Memory, Real Time Support.	Supports network kernel extensions (NKEs). Create network modules, configure protocol stacks. Monitor and modify network traffic.

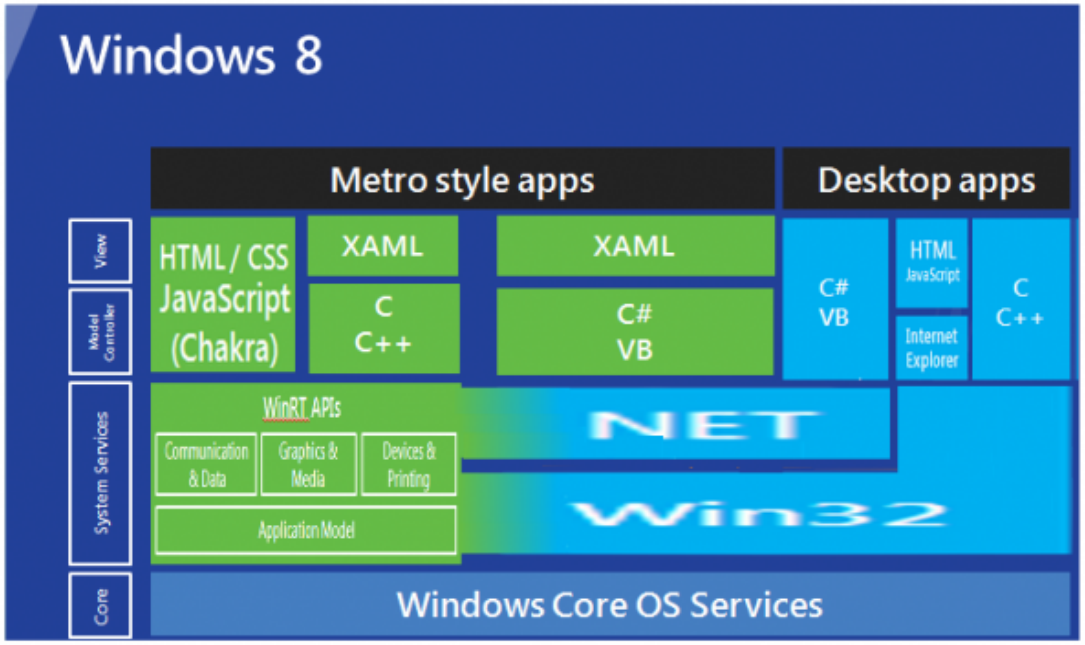


Architecture système

Windows

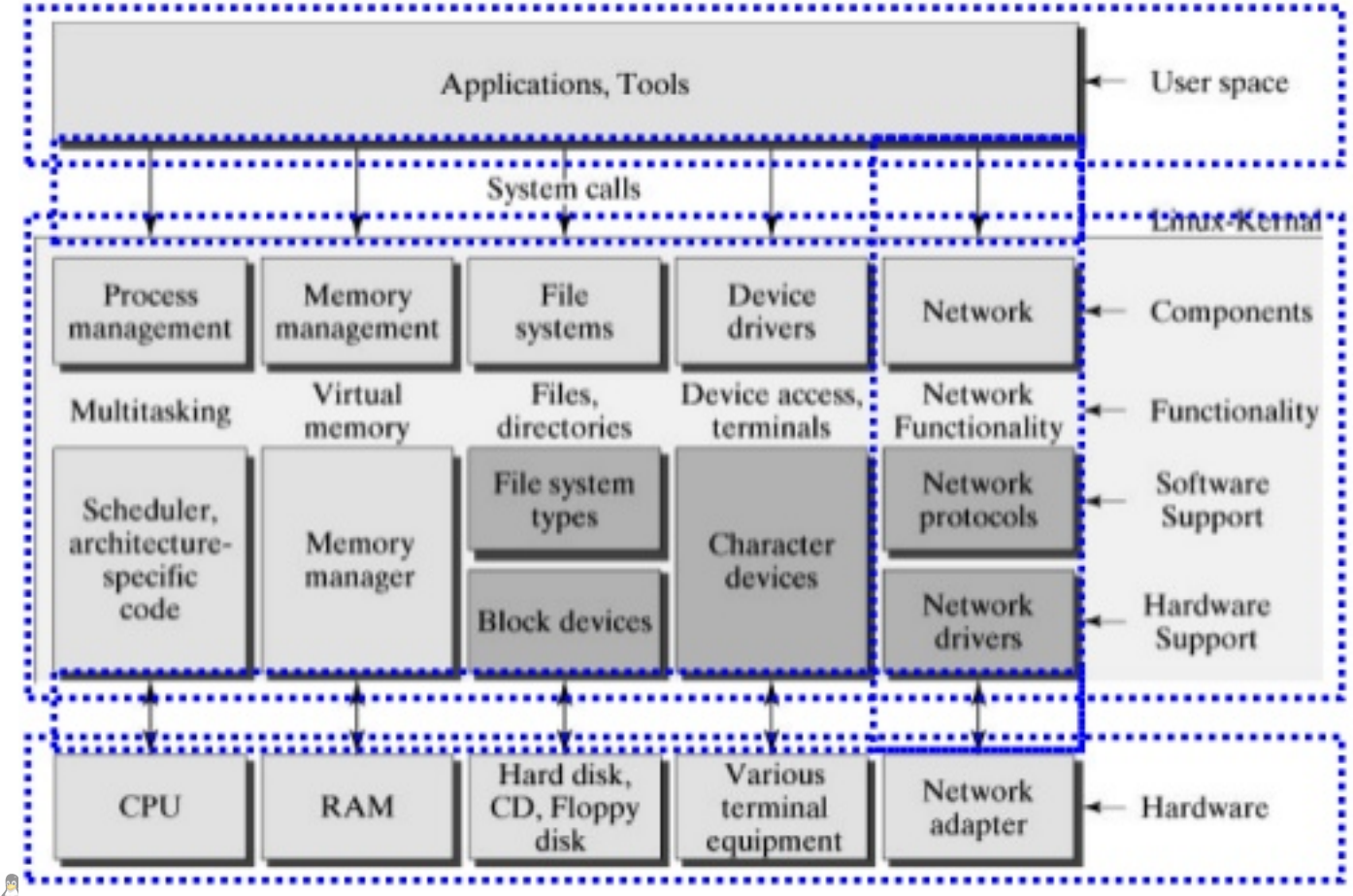


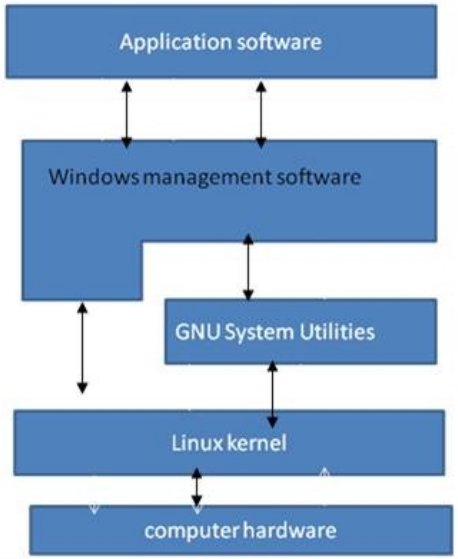


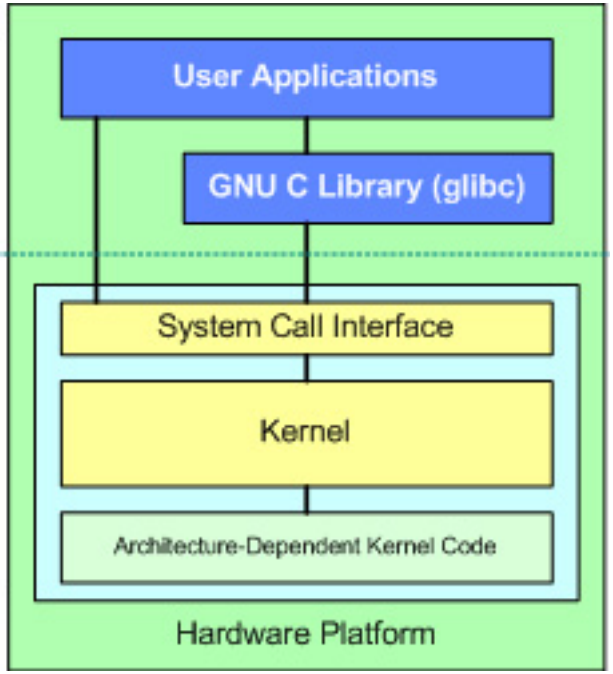


Architecture système

GNU/Linux

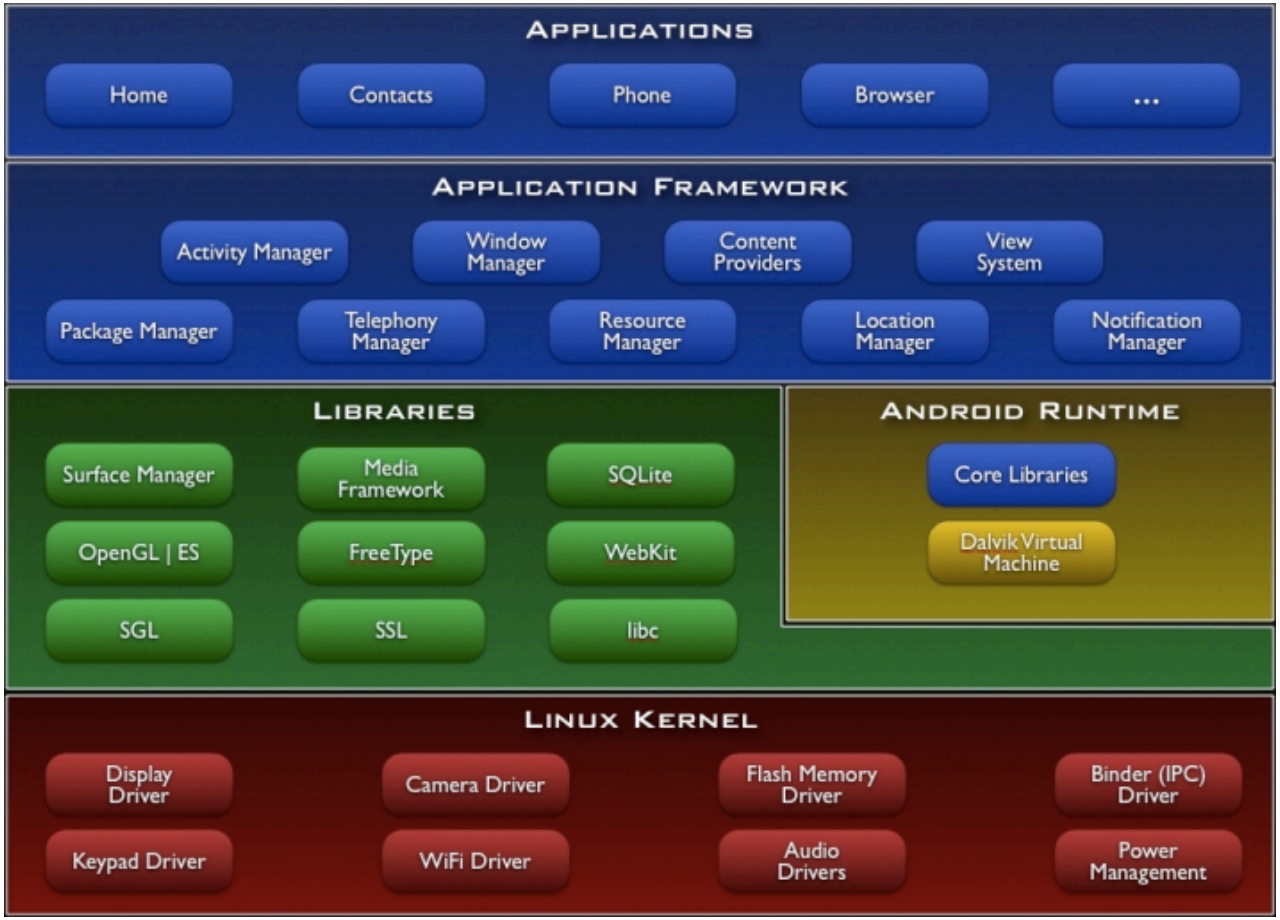






Architecture système

Android/Linux



Architecture Système

Pourquoi ces couches ?

- chaque couche expose au-dessus d'elle de nouvelles abstractions construites (en principe) en ne reposant que sur celles fournies par la couche en dessous d'elle

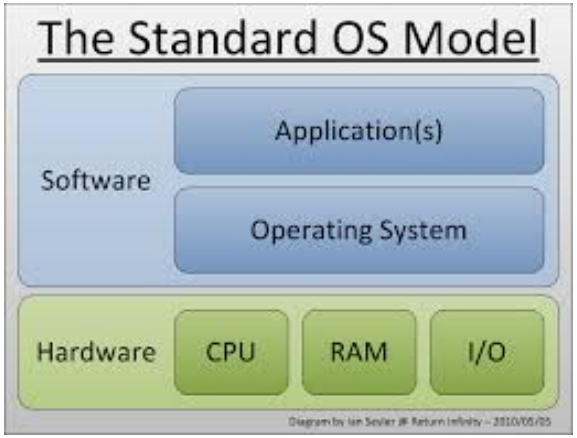
Le couplage faible entre couches

Le modèle en couche n'est pas propre à l'informatique

- mais on l'y trouve un peu partout…

Architecture système

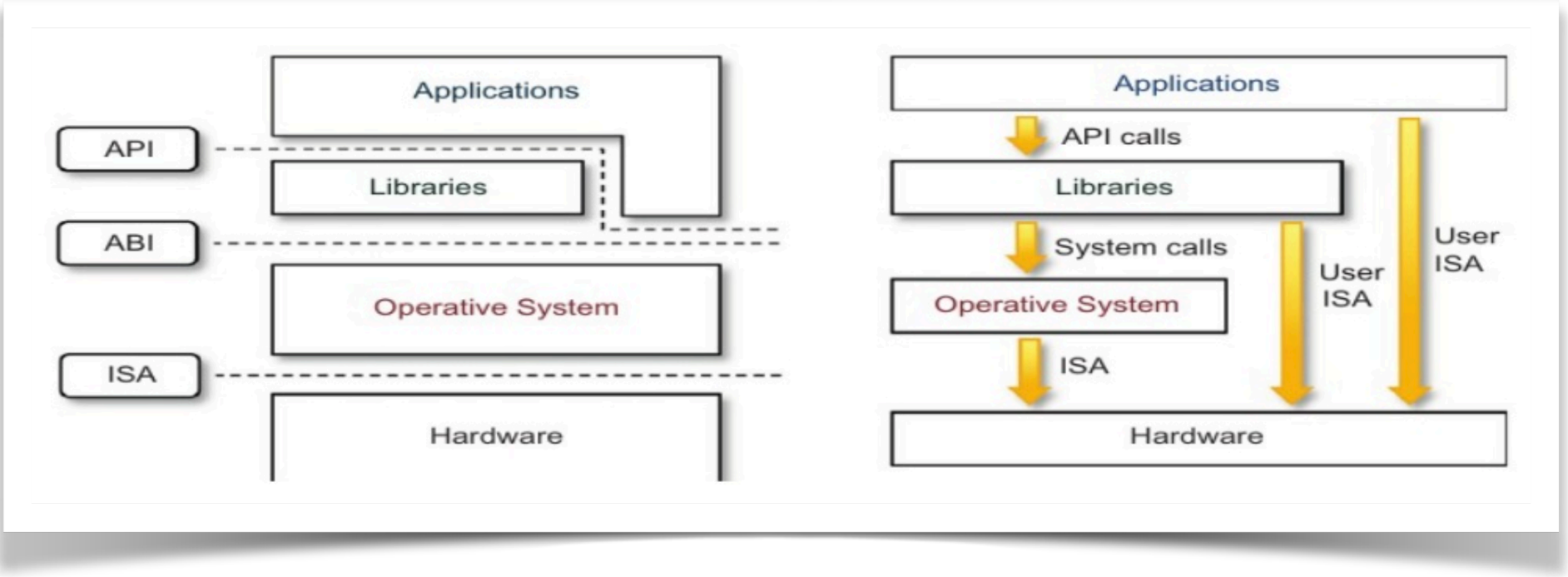
- Modularité
  - les avantages
    - Indépendance de conception
    - Portabilité
    - Interchangeabilité
  - Inconvénients
    - Freine l'innovation
    - Limite l'optimisation



Architecture système

- L’architecture n’est pas neutre
  - Des systèmes de la même famille ne sont pas facilement interchangeables/interopérables
    - La difficile fusion
      - MacOS/iOS,
      - Windows/Windows tablet,
      - Linux/Androïd

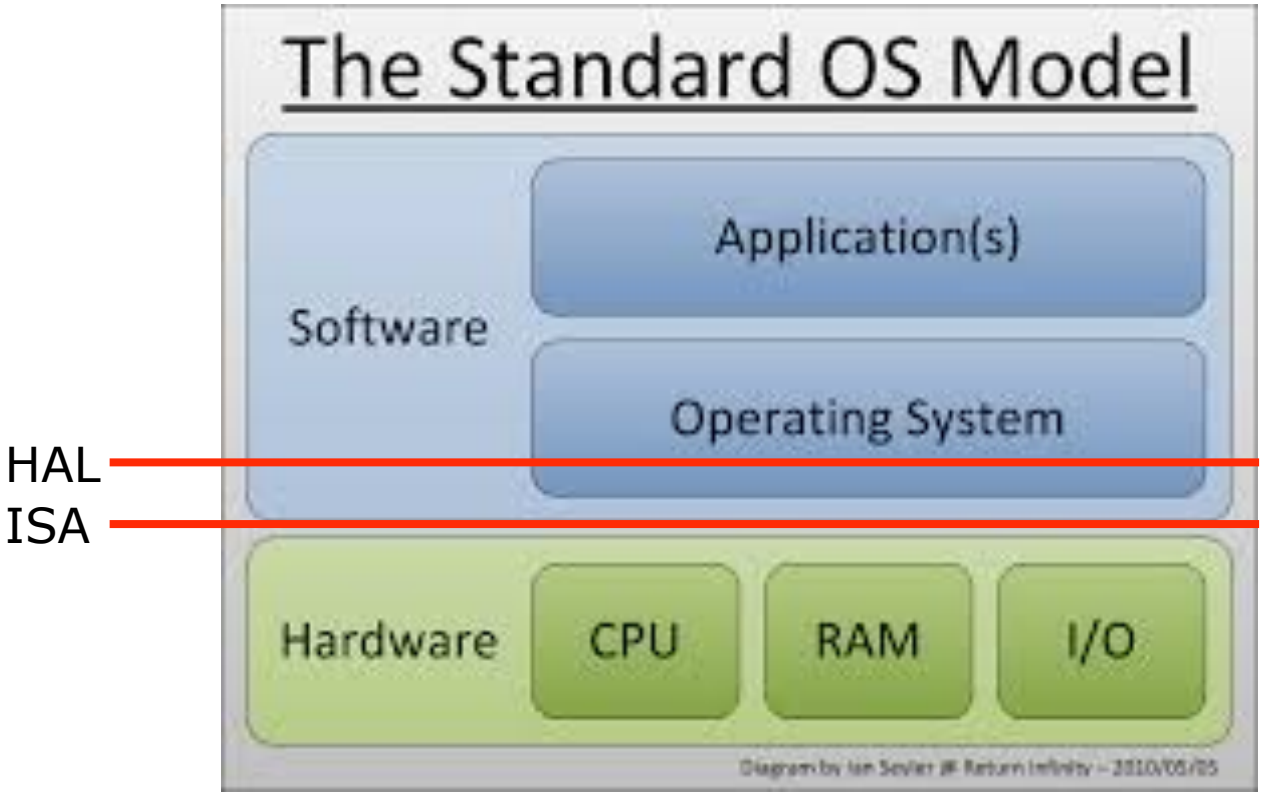
Architecture système



Architecture système

Repose sur l'articulation de trois composants principaux :

- le matériel
- le noyau
- les applications



ISA

Instruction Set Architecture

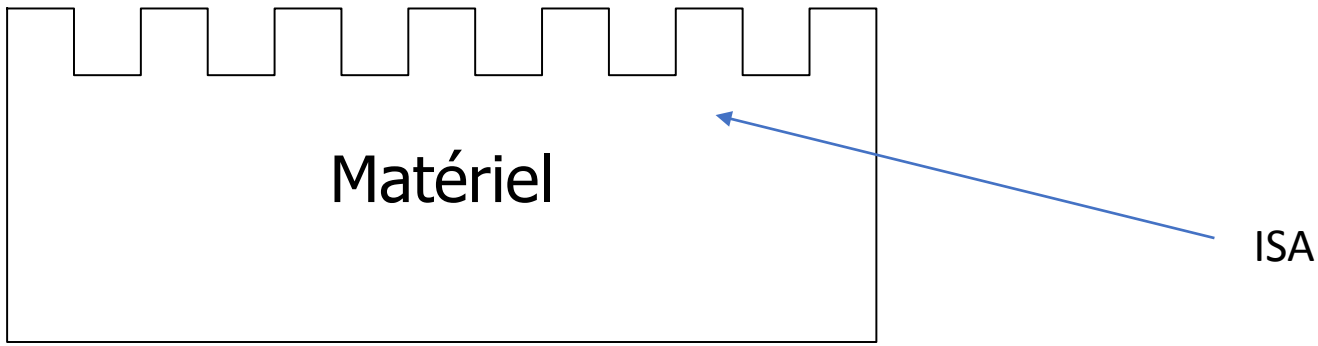
Une interface à la frontière hardware/software

Une architecture abstraite

- décrit l'ensemble de ce que doit connaître un programmeur afin de lui permettre d'écrire des programmes
  - donc principalement le jeu d'instructions de la machine

ISA

Instruction Set Architecture



ISA

- Différents modes d'exécution
  - en général au moins deux
    - pour le kernel (dit mode privilégié)
    - pour les applications (dit mode utilisateur)
  - x86 : 4 rings mais seulement deux utilisés rings 0 et 3
- Selon le mode, toutes les instructions ne sont pas disponibles, toute la machine n'est pas accessible

ISA

Support pour la virtualisation

- Intel-VT
- AMD-V

Faciliter le support machine pour la virtualisation

- en augmenter l'efficacité

3 ISA ?

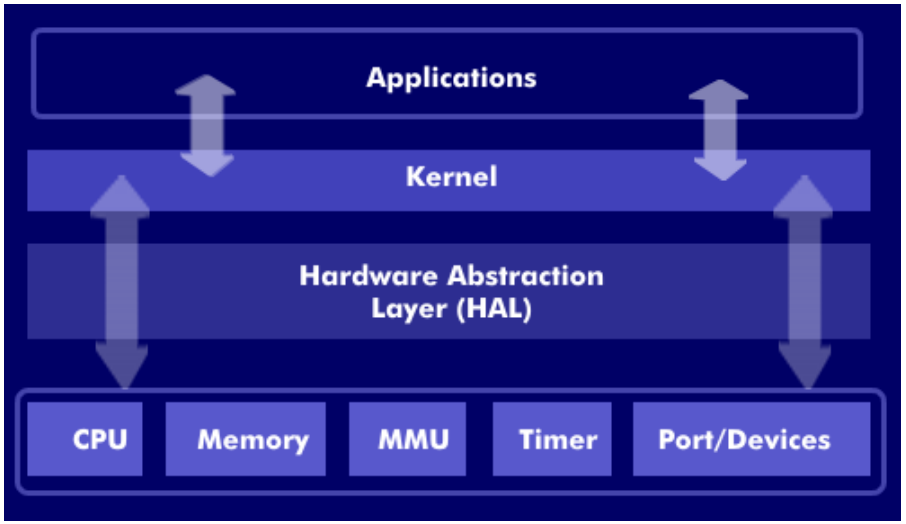
- user, supervisor, hypervisor

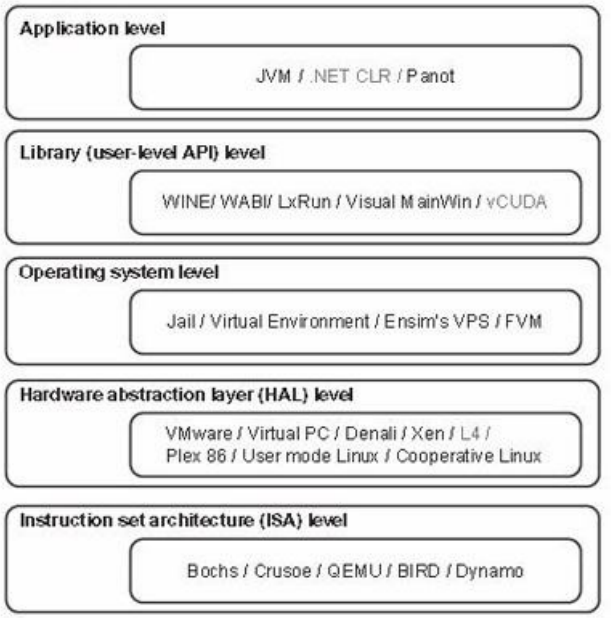
HAL

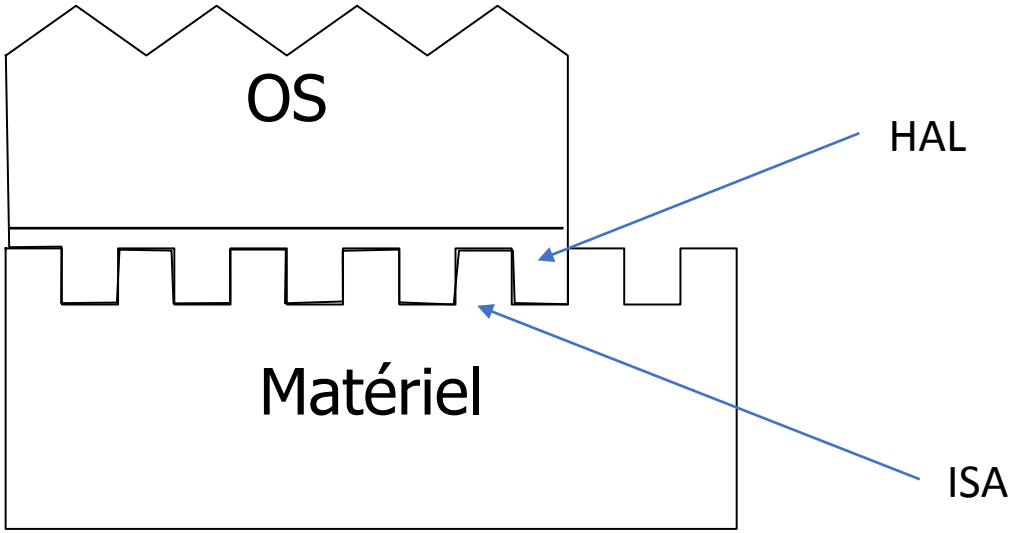
Hardware Abstract Layer

- une couche logicielle permettant de s'affranchir de détails hardware…
- en particulier des périphériques
  - joystick, souris, clavier, audio, disques, etc

HAL


Applications
Kernel
Hardware Abstraction Layer (HAL)
CPU
Memory
MMU
Timer
Port/Devices

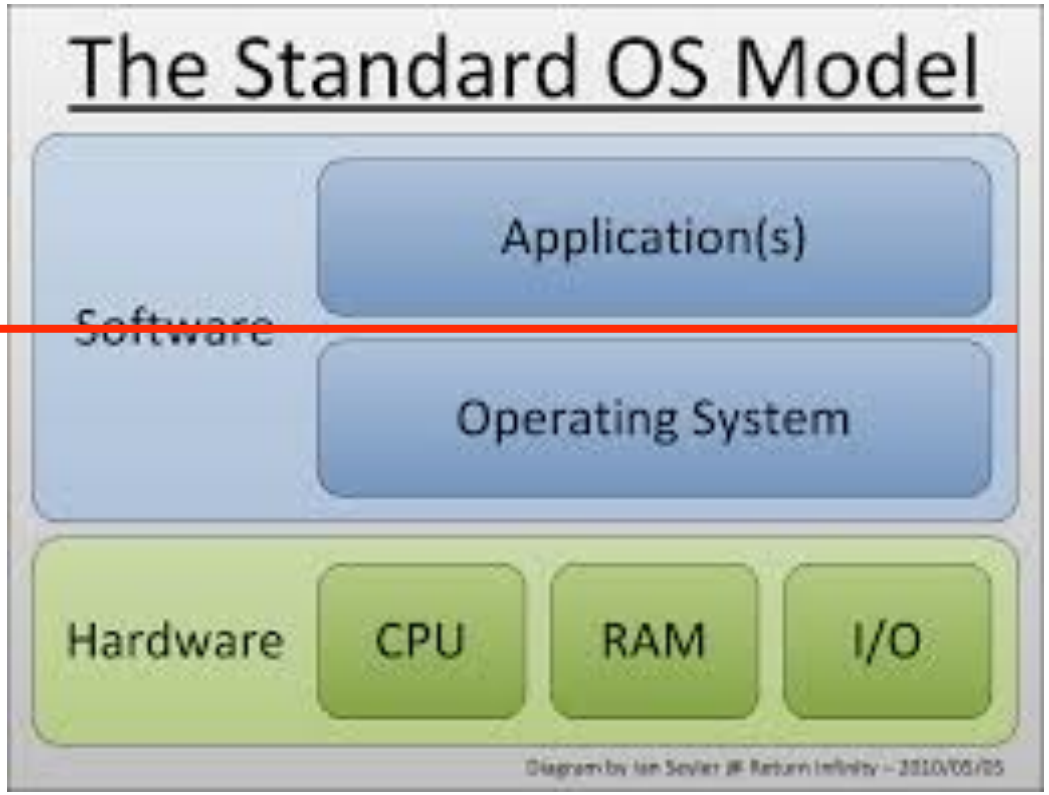

Application level
JVM / .NET CLR / Panot
Library (user-level API) level
WINE/ WABI/ LxRun / Visual MainWin / vCUDA
Operating system level
Jail / Virtual Environment / Ensim's VPS / FVM
Hardware abstraction layer (HAL) level
VMware / Virtual PC / Denali / Xen / L4 / Plex 86 / User mode Linux / Cooperative Linux
Instruction set architecture (ISA) level
Bochs / Crusoe / QEMU / BIRD / Dynamo


OS
HAL
Matériel
ISA

API

Application Programming Interface

API OS



API

C’est la définition des services offerts par le système vus comme des fonctions pouvant être appelées dans un langage

- des manuels de références
  - par exemple man 2 et 3 d’Unix
- des prototypes de fonctions, bibliothèques de code binaire
- Ex. : POSIX
  - Portable Operating System Interface (for uniX)

Permet le portage au niveau du code source

ABI

Application Binary Interface

Portabilité de binaires sur OS différents mais offrant la même API sur des plateformes compatibles (même ISA/HAL)

ABI

Comment passe-t'on des paramètres ?

- les compilateurs génèrent du code pour une ABI

Comment fait-on appel à un service système ?

- numérotation des appels systèmes

Comment invoquer une bibliothèque dynamique ?

Comment invoquer un handler de signal ?

Quelle est la structure d'un exécutable ?

- ELF, mach-O, COFF, a.out, PE

etc.

Portabilité, compatibilité

Compatibilité

« temporelle », ou synonyme de portabilité

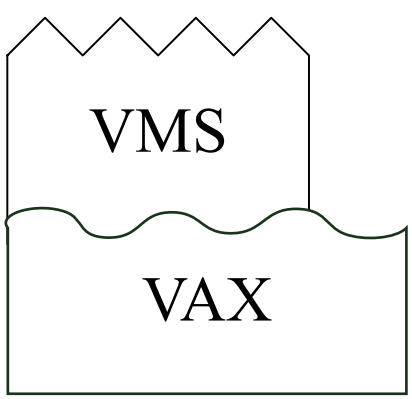
Portabilité

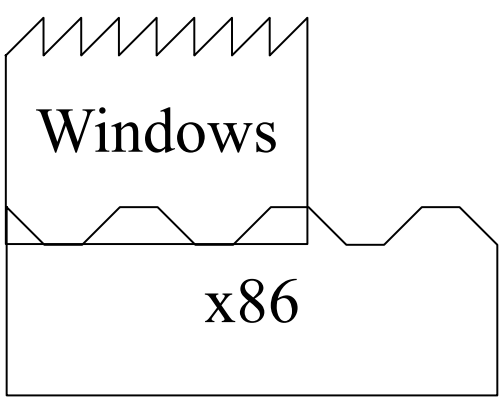
« verticale »

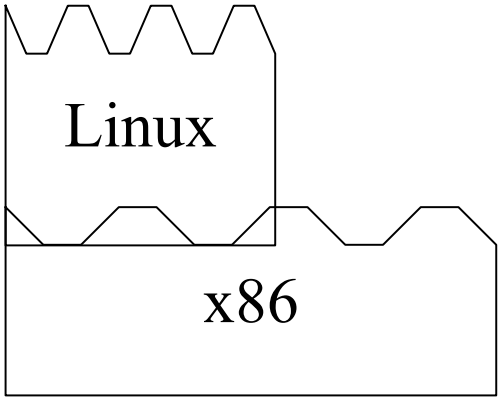
Interopérabilité

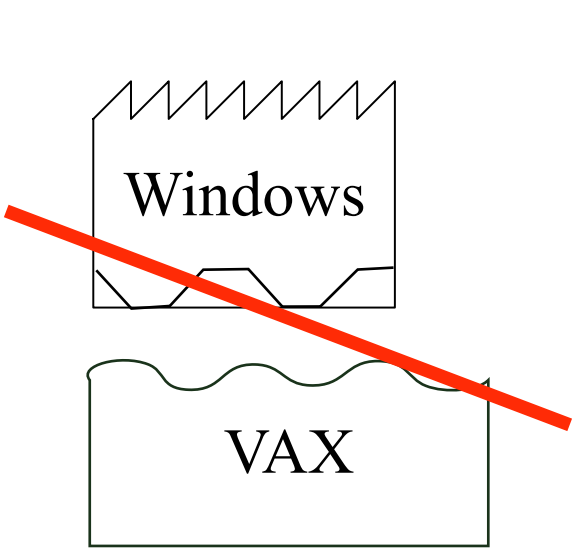
« horizontale », plutôt des problèmes de protocole

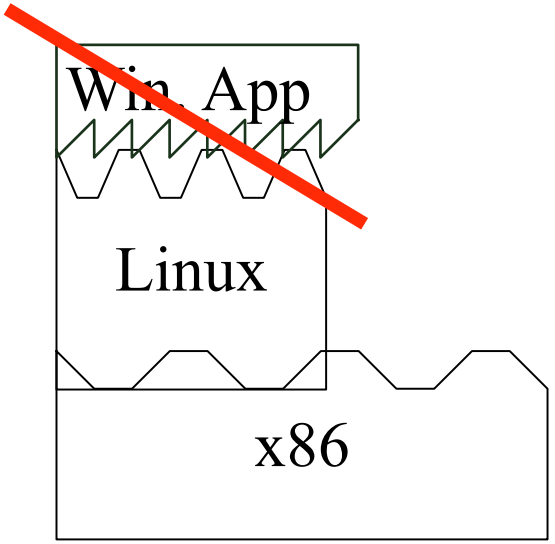
Portabilité, compatibilité











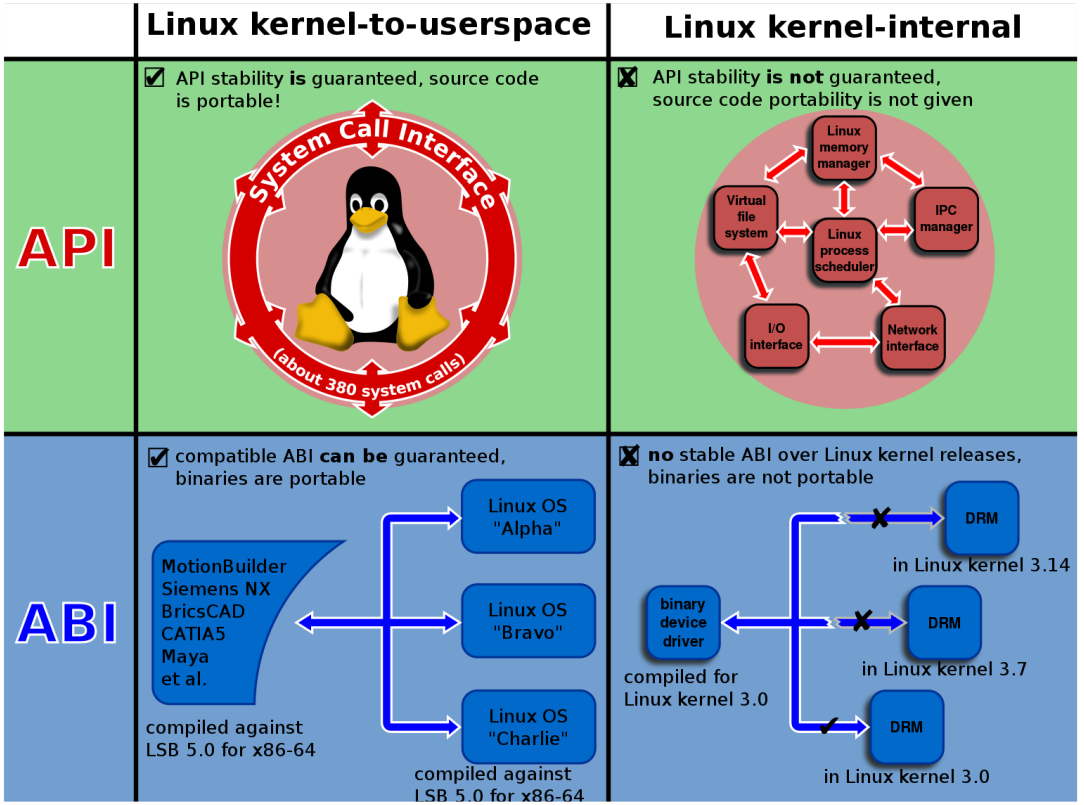
Portabilité

Source ?

une question d'API

Binaire ?

une question d'ABI



Portabilité

Source ?

une question d'API

Ex. : Cygwin ? Simulates Unix env. on Windows

Binaire ?

une question d'ABI

Ex. : flinux ? Run Linux app on Windows

Ex. : Wabi ? Run Windows app on Solaris

Ex. : WINE ? Run Windows app on Linux

Portabilité des OS

- Avec processeurs différents (OS α sur ProcX et ProcY)
  - au niveau du source
    - HAL : OSX sur PowerPC et Intel (phase de transition d'architecture)
  - au niveau du binaire
    - simulation/émulation/virtualisation
- Avec architecture différente (PC x86, Carte x86) :
  - BSP : Board Support Package, abstraction de spécificités

Applications

Pour le monde Unix commun il n'y a pas de concept d'application…

- même si les UI en donnent une certaine illusion
- il n'existe que le concept de processus
  - une approximation : les control groups ?

macOS possède le concept d'application

Windows pas vraiment

Applications

Pour Unix, il n'existe que des programmes qui s'exécutent : un processus

Pour macOS, il existe un niveau où le concept d'application existe

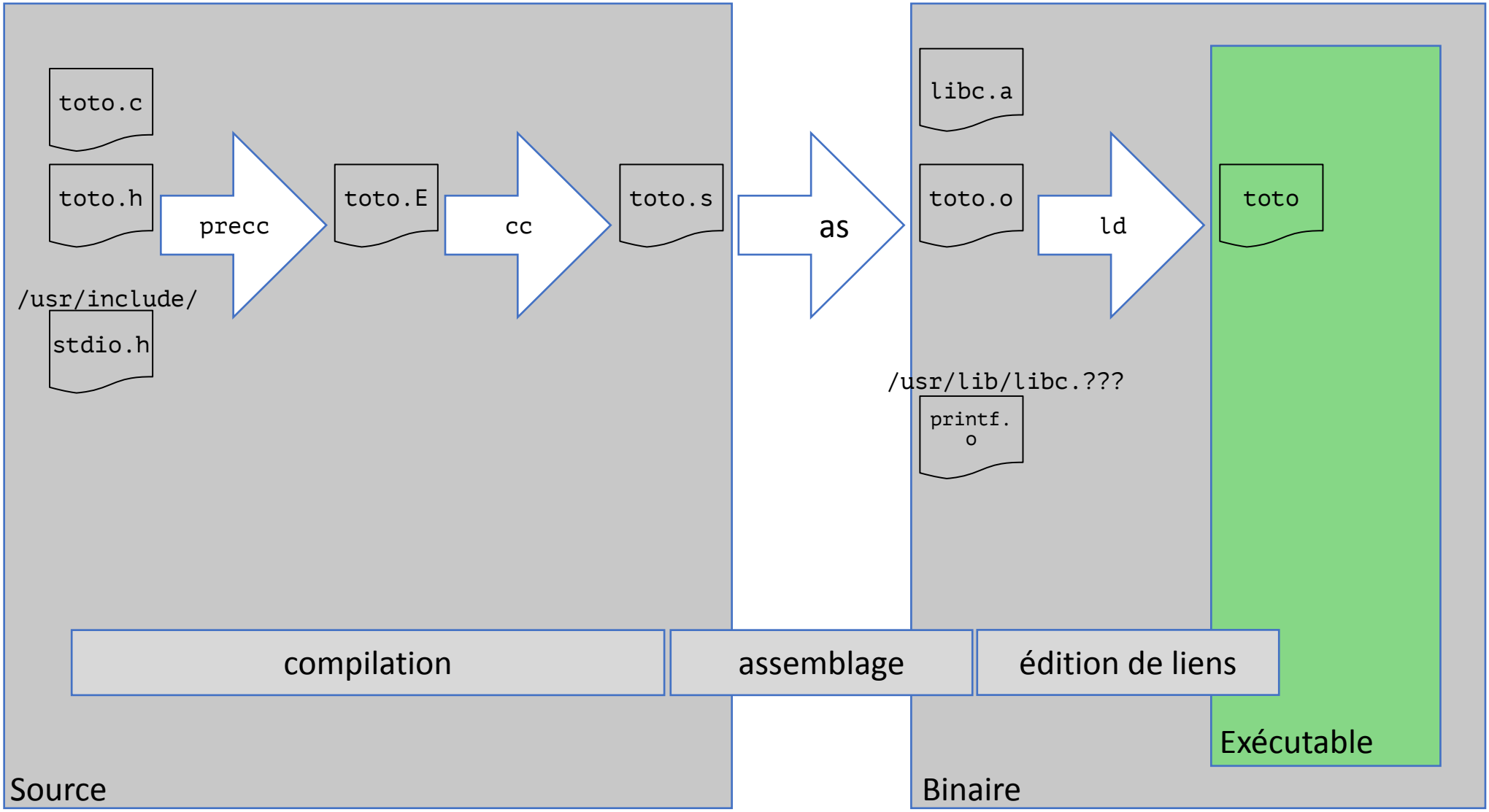
- une application s'exécute ou pas mais pas en plusieurs exemplaires
  - d'autre part elle n'est liée au système que par les bibliothèques standard (pas besoin de charger la bonne lib ou incompatibilité entre libs)
  - elle contient aussi les ressources dont elle a besoin
    - anciennement resource fork du filesystem

Exécutable

Dans tous les cas, il y a au moins du code exécutable à construire

- à partir des sources
- à travers une chaîne de production (dite chaîne de compilation)

Compilation



Exemple (source)

```
#include <stdio.h>
int main() {
    printf("Hello\n");
}
```

Exemple (assembleur)

ISA : Intel

OS/ABI : MacOS

llvm-gcc

```
            .section    __TEXT,__text,regular,pure_instructions
            .macosx_version_min 10, 13
            .globl      _main
            .p2align    4, 0x90
_main:                                  ## @main
            .cfi_startproc
## BB#0:
            pushq       %rbp
Lcfi0:
            .cfi_def_cfa_offset 16
Lcfi1:
            .cfi_offset %rbp, -16
            movq        %rsp, %rbp
Lcfi2:
            .cfi_def_cfa_register %rbp
            subq        $16, %rsp
            leaq        L_.str(%rip), %rdi
            movb        $0, %al
            callq       _printf
            xorl        %ecx, %ecx
            movl        %eax, -4(%rbp)          ## 4-byte Spill
            movl        %ecx, %eax
            addq        $16, %rsp
            popq        %rbp
            retq
            .cfi_endproc

            .section    __TEXT,__cstring,cstring_literals
L_.str:                                 ## @.str
            .asciz      "Hello\n"


.subsections_via_symbols
```

Exemple (assembleur)

ISA : Intel

OS/ABI : Ubuntu

gcc

```
            .file       "toto.c"
            .section    .rodata
.LC0:
            .string     "Hello"
            .text
            .globl      main
            .type       main, @function
main:
.LFB0:
            .cfi_startproc
            pushq       %rbp
            .cfi_def_cfa_offset 16
            .cfi_offset 6, -16
            movq        %rsp, %rbp
            .cfi_def_cfa_register 6
            movl        $.LC0, %edi
            call        puts
            popq        %rbp
            .cfi_def_cfa 7, 8
            ret
            .cfi_endproc
.LFE0:
            .size       main, .-main
            .ident      "GCC: (Ubuntu 4.8.4-2ubuntu1~14.04.3) 4.8.4"
            .section    .note.GNU-stack,"",@progbits
```

Exemple (assembleur)

ISA : Intel

OS/ABI : Windows

mingw (gcc)

```
            .file       "toto.c"
            .def        ___main;
            .scl        2;
            .type       32;
            .endef
            .section .rdata,"dr"
LC0:
            .ascii "Hello\0"
            .text
            .globl      _main
            .def        _main;
            .scl        2;
            .type       32;
            .endef
_main:
            pushl       %ebp
            movl        %esp, %ebp
            andl        $-16, %esp
            subl        $16, %esp
            call        ___main
            movl        $LC0, (%esp)
            call        _puts
            leave
            ret
            .ident      "GCC: (tdm-1) 4.9.2"
            .def        _puts;
            .scl        2;
            .type       32;
            .endef
```

Exemple (exécutable)

dans les trois cas un code assembleur différent donc un code exécutable différent

- ce n'est même pas le même système et donc pas la même ABI ni API

MacOS

```
Mach-O 64-bit executable x86_64
```

Ubuntu

```
ELF 64-bit LSB  executable, x86-64, version 1 (SYSV), dynamically linked (uses shared libs), for
GNU/Linux 2.6.24, BuildID[sha1]=7023dfd9e4ac3387919290ac20ee8a23e66fa386, not stripped
```

Windows 7

```
PE32 executable (console) Intel 80386, for MS Windows
```

Statique vs Dynamique

Le code source ne contient pas la **définition** de la fonction `printf`

- en dynamique les exécutables non plus (par défaut, aujourd'hui)
  - mais contiennent de quoi la retrouver…
- en statique les exécutables sont clos (rarement utilisé aujourd'hui)
  - macOS n'offre plus de liaison statique en standard, ni Solaris d'ailleurs

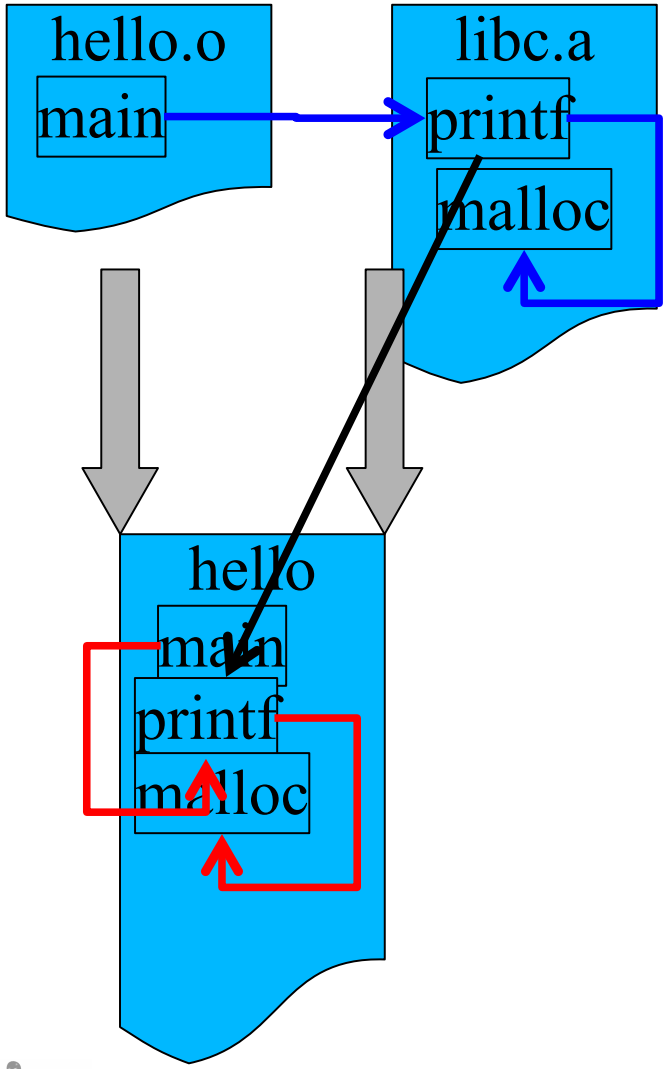
Statique vs Dynamique

Le code source ne contient pas la **définition** de la fonction `printf`

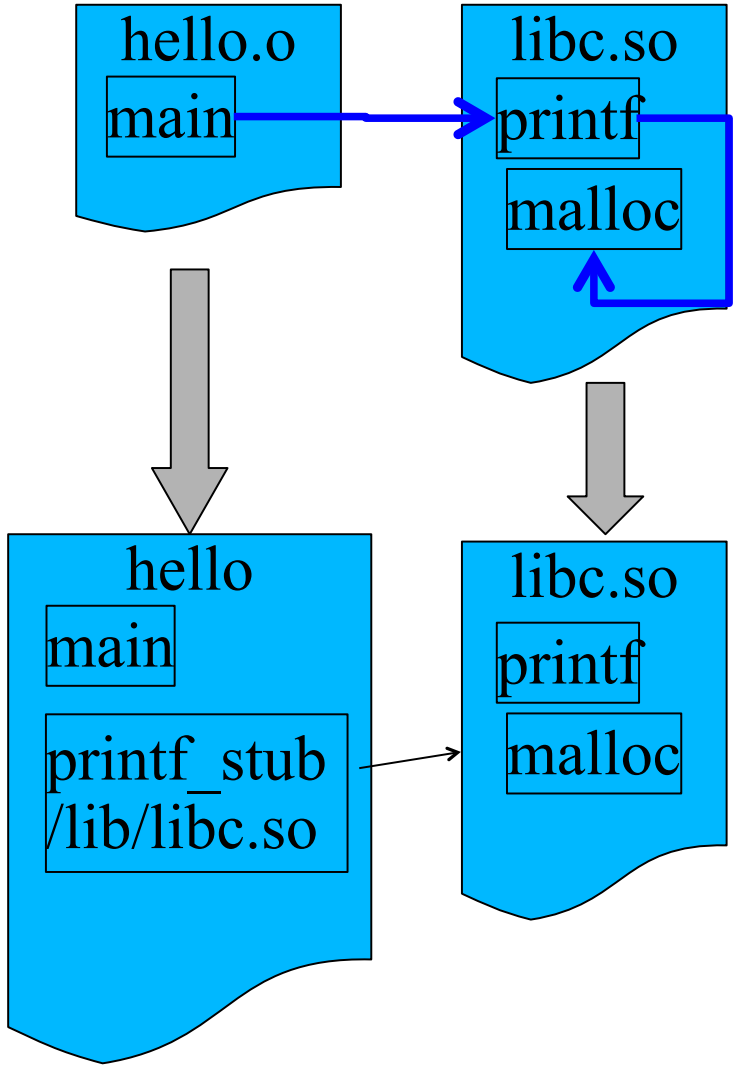
- par contre il contient sa **déclaration**
  - permet au compilateur d'émettre le code permettant d'appeler la fonction même savoir où elle est…
    - l'édition de lien raboutera le tout

Statique vs Dynamique

Statiques



Dynamiques



Bibliothèques dynamiques

Lien avec la bibliothèque libc.so (entre autres…)

```
$ cc -o toto toto.c
$ file toto
toto:  ELF 64-bit LSB  executable, x86-64, version 1 (SYSV),
dynamically linked (uses shared libs), for GNU/Linux 2.6.24,
BuildID[sha1]=7023dfd9e4ac3387919290ac20ee8a23e66fa386, not
stripped
$ size toto
   text        data      bss          dec     hex       filename
   1192         560        8         1760     6e0       toto
```

Bibliothèques statiques

Lien avec la bibliothèque libc.a (entre autres…)

```
$ cc -o toto toto.c -static #ou -Wl,-static
$ file toto
toto:  ELF 64-bit LSB  executable, x86-64, version 1 (GNU/
Linux), statically linked, for GNU/Linux 2.6.24,
BuildID[sha1]=bb6748732f6d7af5f915522eb01ad61106e158f2, not
stripped
$ size toto
   text    data     bss     dec     hex filename
 783139    7532    9600  800271    c360f toto
```

Bibliothèques statiques

duplique le code dans l'exécutable (sur disque et en mémoire)

- avantages : chargement et exécution simple, pas de dépendances, un code autonome
- inconvénients : taille plus importante (assez notable), ne bénéficient pas des mises-à-jour des bibliothèques…

Bibliothèques dynamiques

Dynamiques

- pas de duplication des codes des fonctions
- avantages : taille réduite, bénéfice des mises-à-jour
- inconvénients : nécessiter d'avoir les bibliothèques disponibles, dépendance entre code et bibliothèques, chargement plus complexe

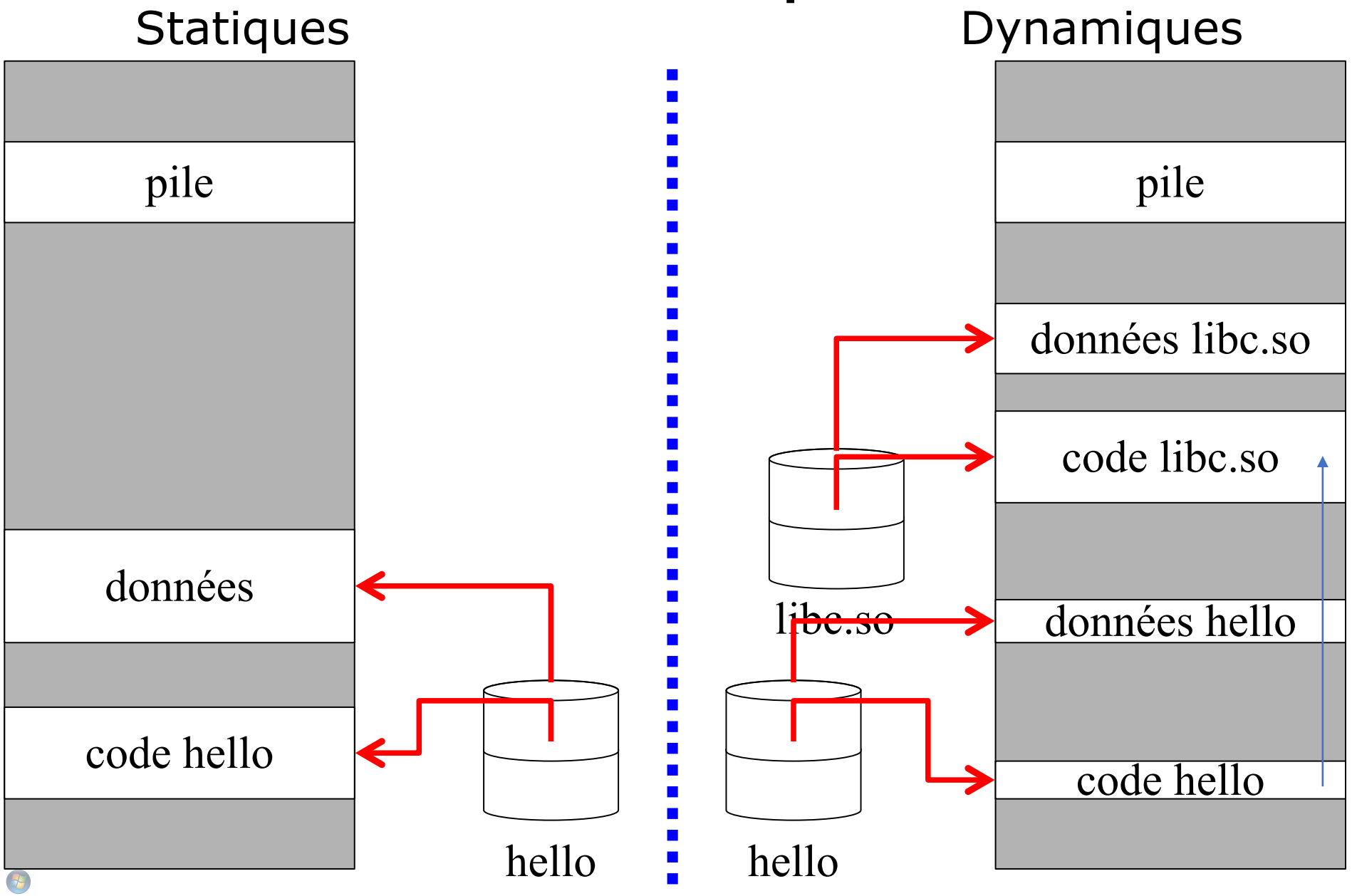
nécessite d'avoir un code de nature particulière : PIC (Position Independent Code)

- option `-fpic` de `gcc`

nécessite un support particulier dans le système

- édition de lien dynamique

Bibliothèques



Liaison dynamique

La liaison dynamique peut-être obtenue par appel explicite

POSIX

`dlopen()` pour «ouvrir» attacher une bibliothèque partagée

`dlsym()` pour obtenir l'adresse d'un symbole de la bibliothèque

Grandes fonctions d'un OS

Gestion mémoire

Gestion des exécutions

Gestion des fichiers

Gestion du matériel

Gestion mémoire

Partage

Protection

Relogement

Organisation Physique

Organisation Logique

Cache

Services mémoire (*nix)

Explicites (sur requête du code)

- Le tas : `malloc/new`
- La mémoire partagée : `mmap`
- La protection : `mprotect`
- Le comportement système : `mlock`
- `exec`

Implicites (appel de fonction, chargement de code)

- chargement de page, défauts de page
- gestion de la pile
- violation des droits

Questions à se poser

Intervalles d'adresses qui peuvent être atteintes ?

Adresses valides ?

Que faire si tentative d'accès à une adresse « interdite » ?

Qu'est ce qui est autorisé avec une adresse valide ?

Besoins en termes de gestion des adresses (point de vue processus ou système) ?

De quelle mémoire physique dispose-t'on ?

Évolution de la gestion mémoire

- Les premiers temps
  - pas d'OS donc pas de gestion mémoire ? C'est sûr ? La pile ? Les overlays ?
- Un OS avec une seule application
  - pas de vrai besoin sauf celui de l'isolation entre les deux parties…
- Un OS et plusieurs applications
  - les besoins sont nettement plus compliqués
    - chargement/déchargement des apps, isolation inter-proc, proc/OS, gestion dynamique…

Besoins complémentaires…

Charger des applications même si l'espace mémoire physique est entièrement occupé

- swap / pagination

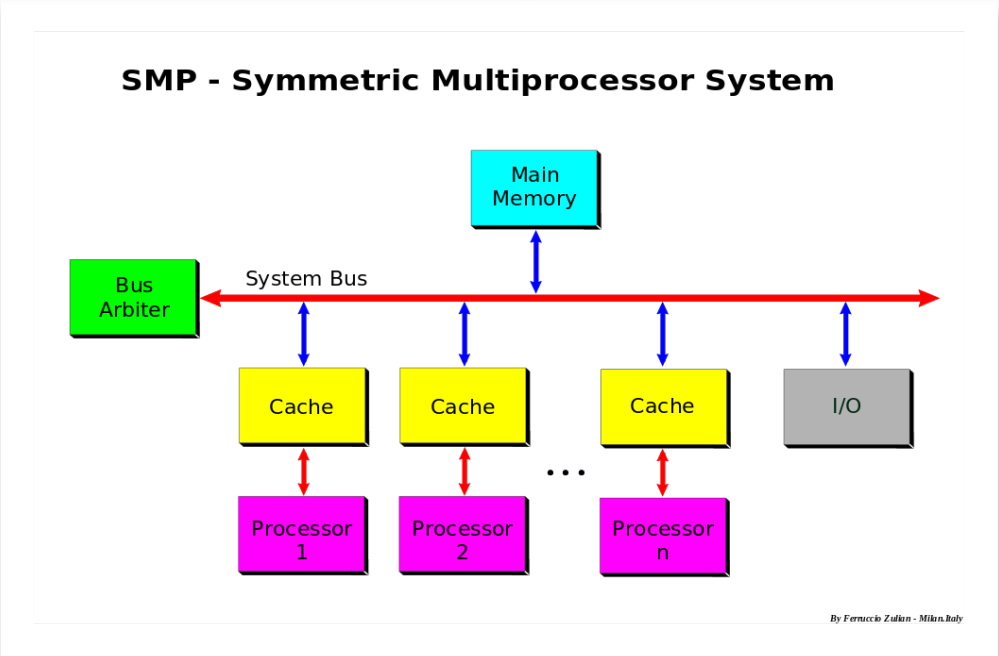
Permettre à une application d'utiliser un espace qui dépasse la taille de la mémoire physique

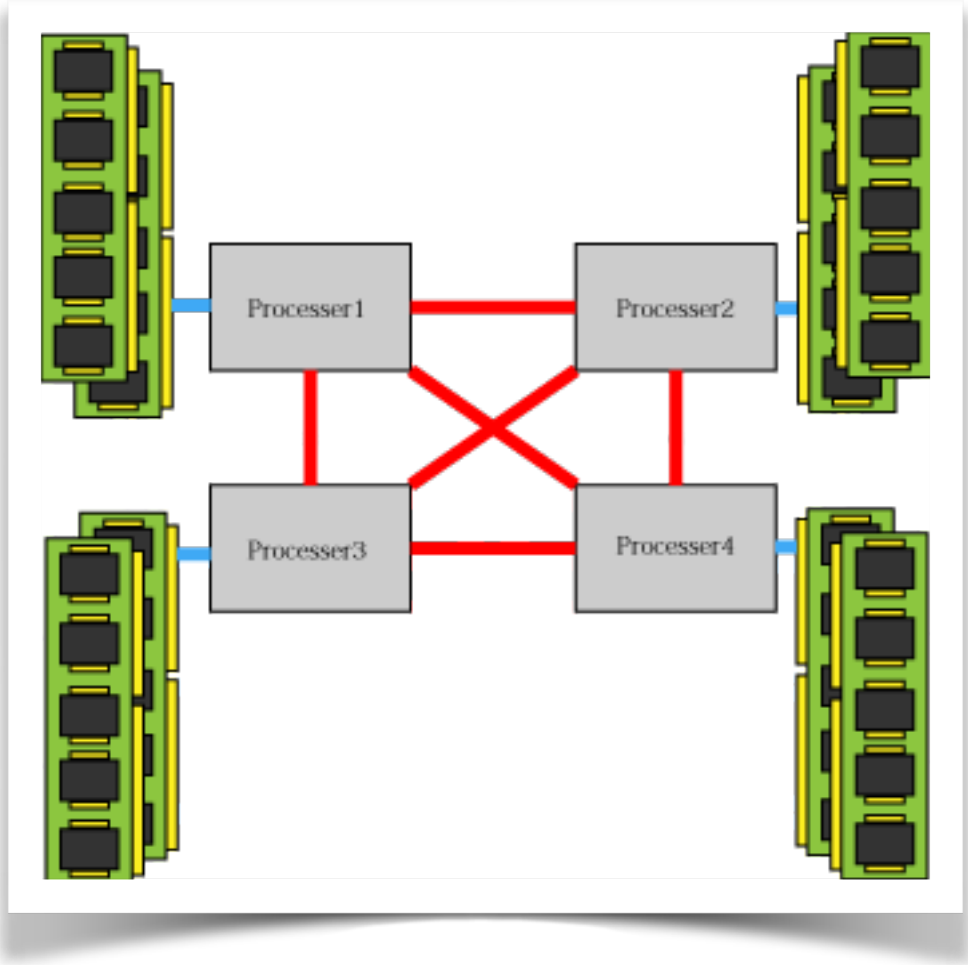
Exploiter une machine qui aurait plus d'espace de mémoire physique qu'un processus n'a d'espace mémoire possible

Besoins complémentaires…

- Support des machines multi-processeurs
  - SMP (Symmetric Multi-Processor)
    - tous les CPU accèdent à la même mémoire, de la même manière
    - attention aux caches…
  - NUMA (Non-Uniform Memory Access)
    - même mémoire mais l'usage n'est pas uniforme (accès dépendant des couples bancs/CPU)
    - attention au placement des données, etc

SMP/NUMA


SMP - Symmetric Multiprocessor System
Main Memory
Bus Arbiter
System Bus
Cache
Cache
Cache
I/O
Processor 1
Processor 2
Processor n
By Ferruccio Zulian - Mikro.Italy


Processer1
Processer2
Processer3
Processer4

Mono-tâche, pas d'OS

Le programme est chargé (on parle de loader si c'est un programme qui s'en charge - une sorte d'OS très primitif) en un point prédéfini

- son édition de lien a eu lieu en connaissance de cause

Nul besoin de service mémoire, le programme est seul sur la machine

- il se débrouille si nécessaire

La machine appartient au programme

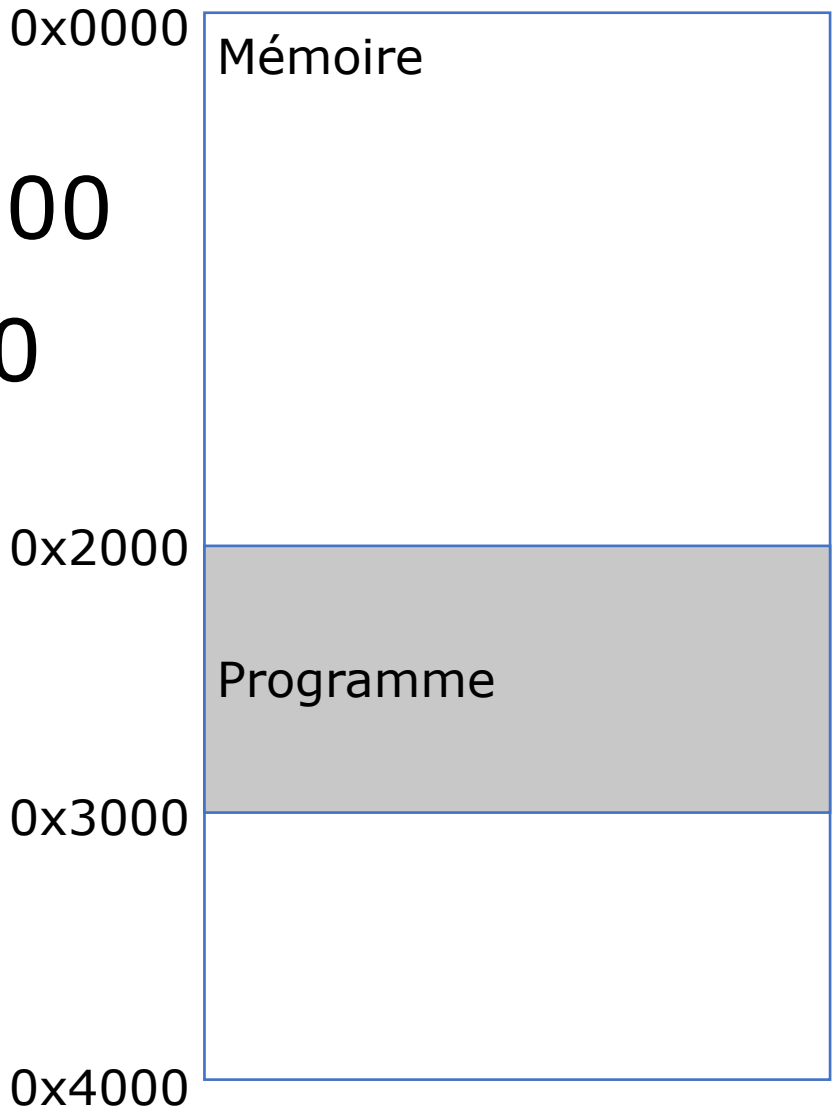
- on fait ce que l'on veut

Mono-tâche, pas d'OS

Mémoire de taille 0x4000

Programme de taille 0x1000

Chargé à l'adresse 0x2000



Mono-tâche, avec OS

- Le programme est chargé en tenant compte de l'existence de l'OS
  - son édition de lien a eu lieu en connaissance de cause
  - ou alors l'OS se charge de son relogement
    - recalcul d'adresses (address translation)
    - code relatif
    - code indexé
- En général, dans ce cas la machine appartient encore au programme
  - on fait ce que l'on veut
  - en particulier
    - on peut utiliser ou non les services de l'OS
    - on peut abîmer l'OS (pas de protection)

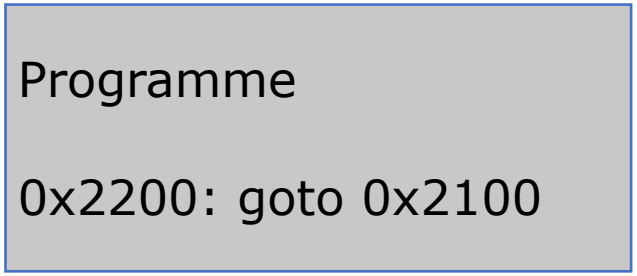
Mono-tâche, avec OS

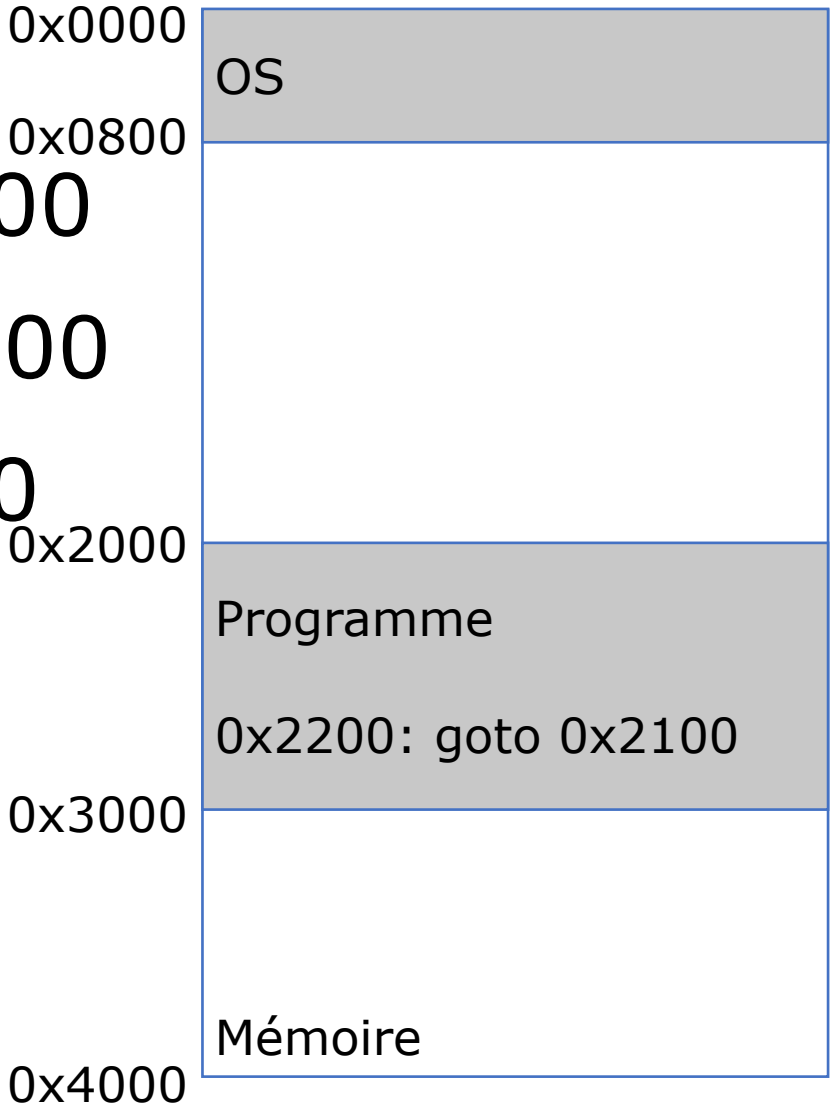
Mémoire de taille 0x4000

OS taille 0x0800 en 0x0000

Programme de taille 0x1000

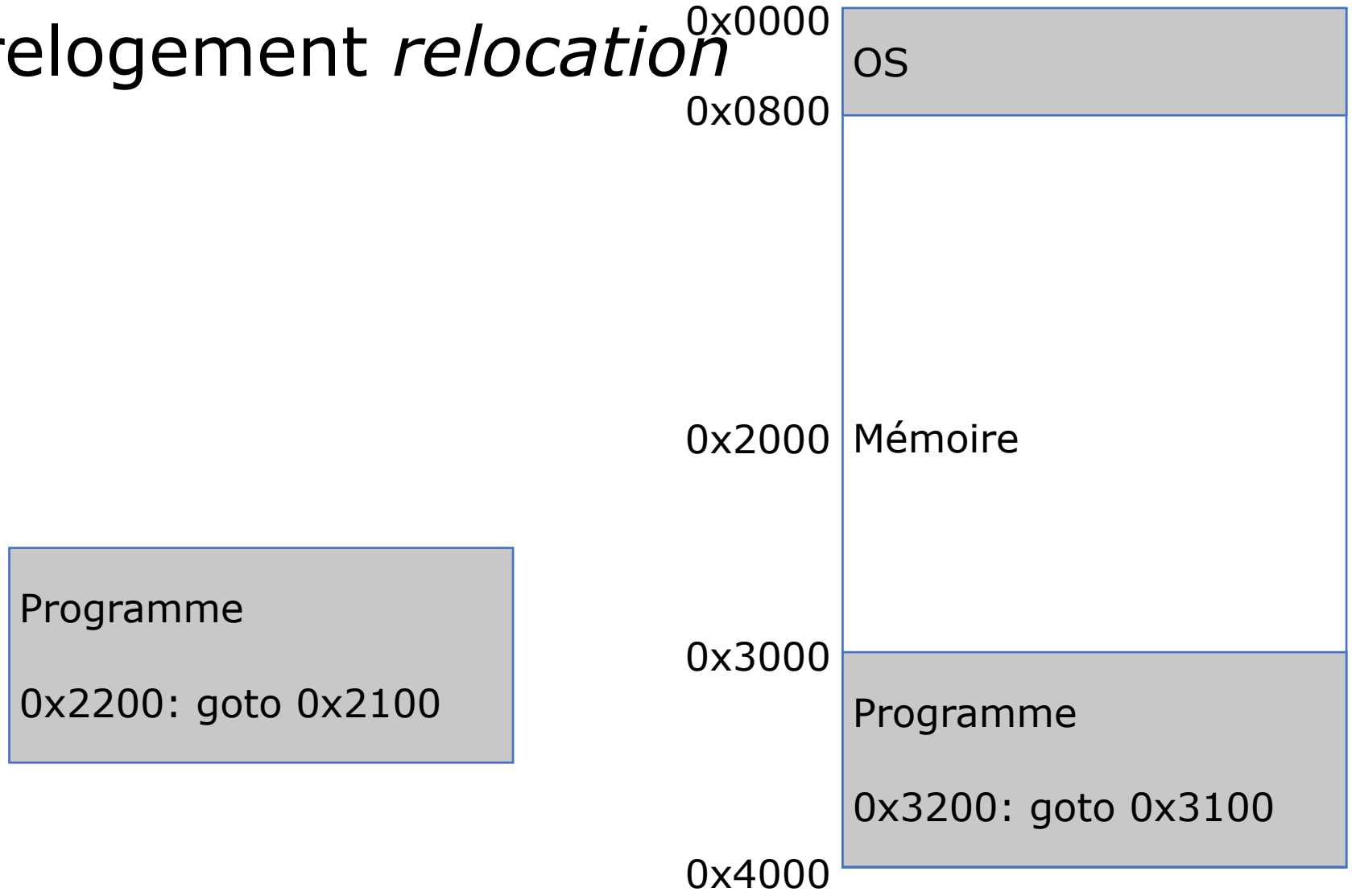
Chargé à l'adresse 0x2000





Mono-tâche, avec OS

Avec relogement *relocation*



Mémoire virtuelle

On a donc deux visions de la mémoire (dans le cas du relogement)

- la vision du programmeur qui fait une hypothèse sur la machine (par exemple elle est pour lui seul)
- la vision système qui doit tenir compte de l'existant (de sa présence en particulier) et qui donc modifie le programme en en tenant compte

Mémoire virtuelle

On a donc deux visions de la mémoire (dans le cas du relogement)

- une vision statique : le programme
- une vision dynamique : le programme qui s'exécute (aka processus)
  - notion d'environnement d'exécution

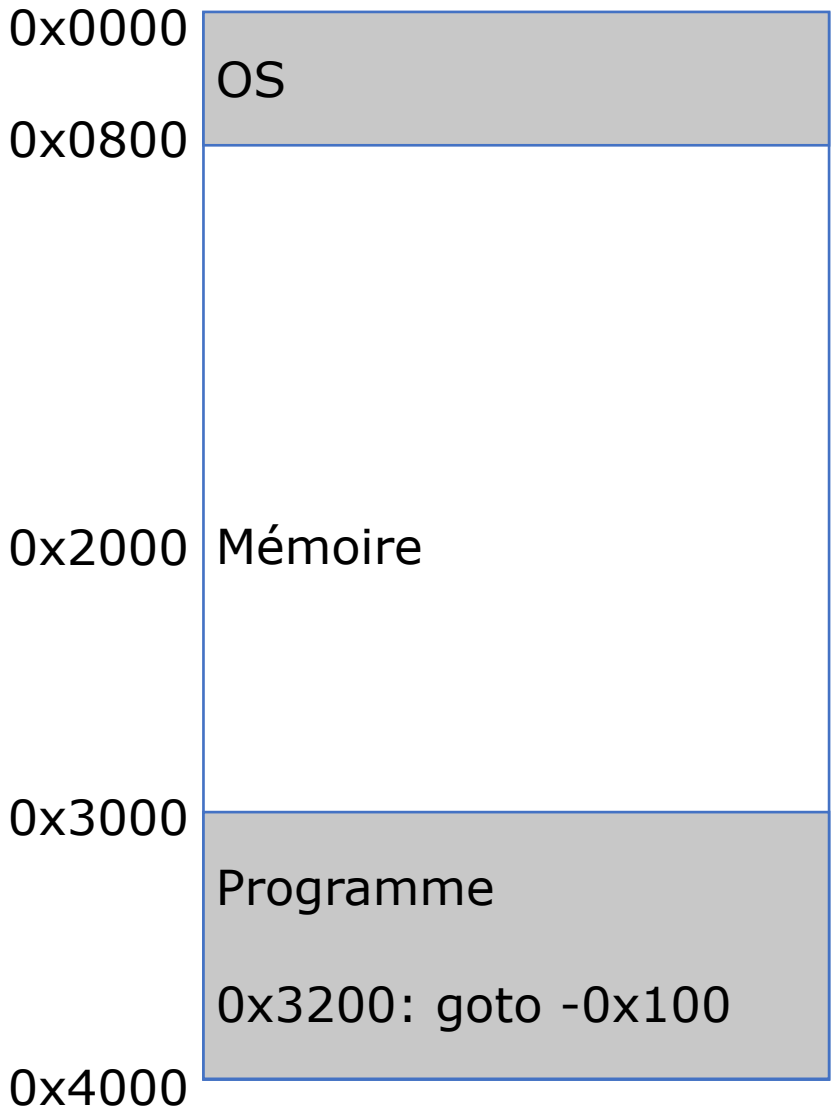
Mémoire virtuelle

On a donc deux visions de la mémoire (dans le cas du relogement)

- une vision logique : le programme qui s'exécuterait à tel endroit
- une vision physique : le programme qui s'exécute à tel endroit

Mono-tâche, avec OS

Avec code relatif



Programme

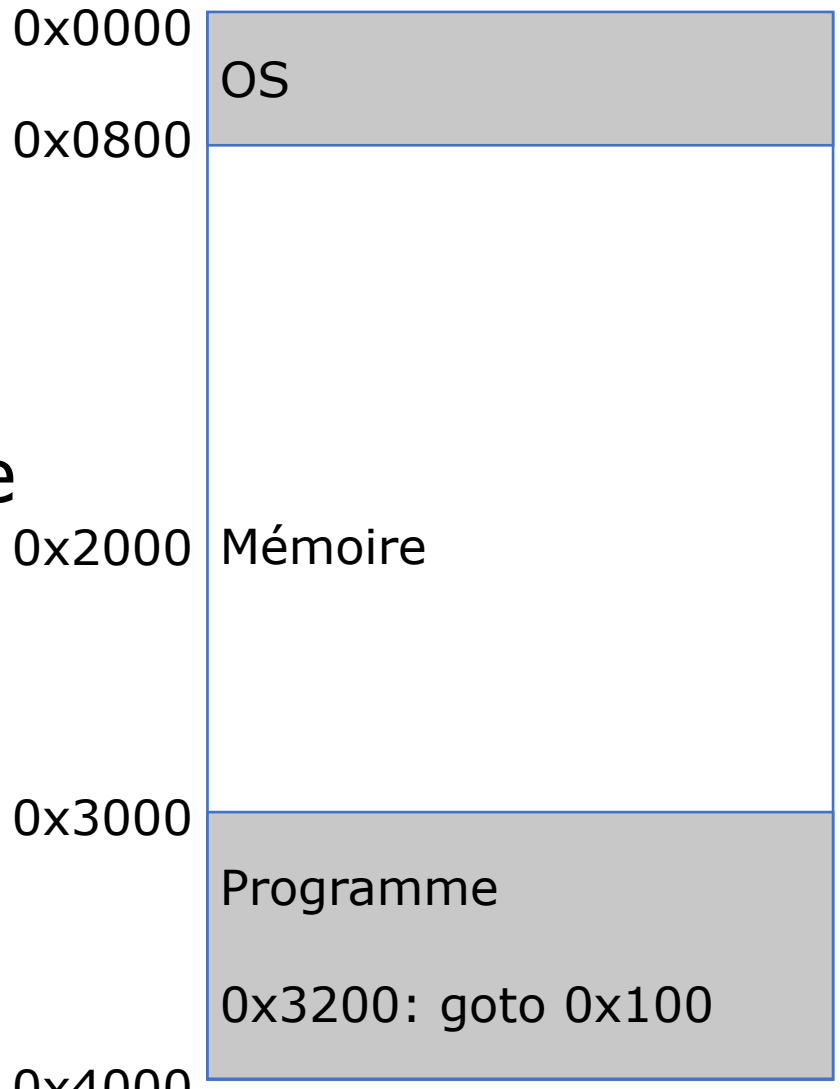
0x200: goto -0x0100

Mono-tâche, avec OS

Avec code indexé

à l'exécution de l'instruction on calcule l'adresse effective par addition de l'adresse émise avec l'adresse de chargement

0x3000+0x0100 = 0x3100



Programme

0x200: goto 0x0100

Mémoire virtuelle

La mémoire virtuelle est celle sur laquelle raisonne le programmeur M_p

La mémoire physique est celle avec laquelle le processeur fonctionne M_ϕ

Le système de mémoire virtuelle réalise donc une fonction $\phi_M : M_p \longmapsto M_\phi$

Un nouveau problème

Maintenant que faire si on souhaite exécuter un programme plus grand que la mémoire physique disponible ?

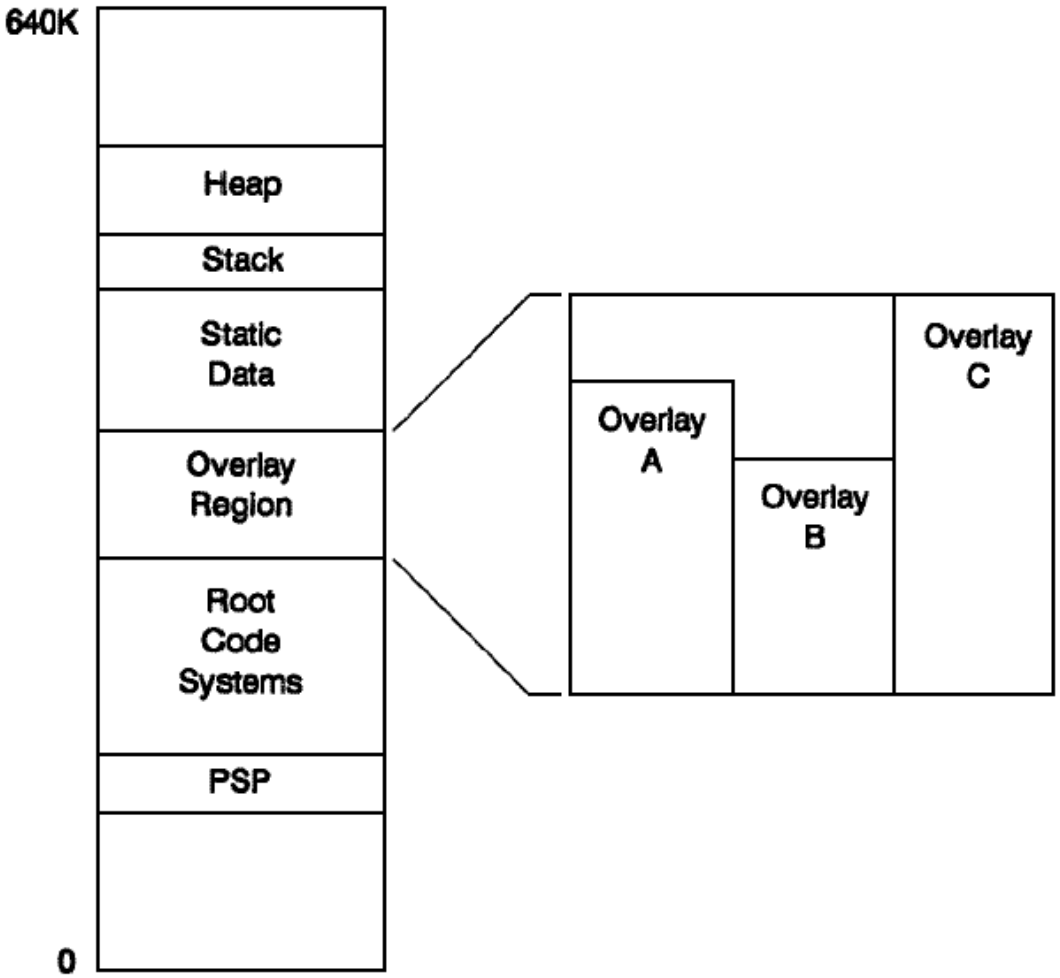
- La réponse primitive est celle des overlays
  - des zones mémoires dédiées à charger/ décharger du code en cours d'exécution

Leur utilisation est délicate et pas seulement techniquement

Les overlays

MS/DOS

Overlays



Multitâche avec OS

Il y a un nécessaire relogement (donc une mémoire virtuelle même primitive)

On peut fournir une isolation

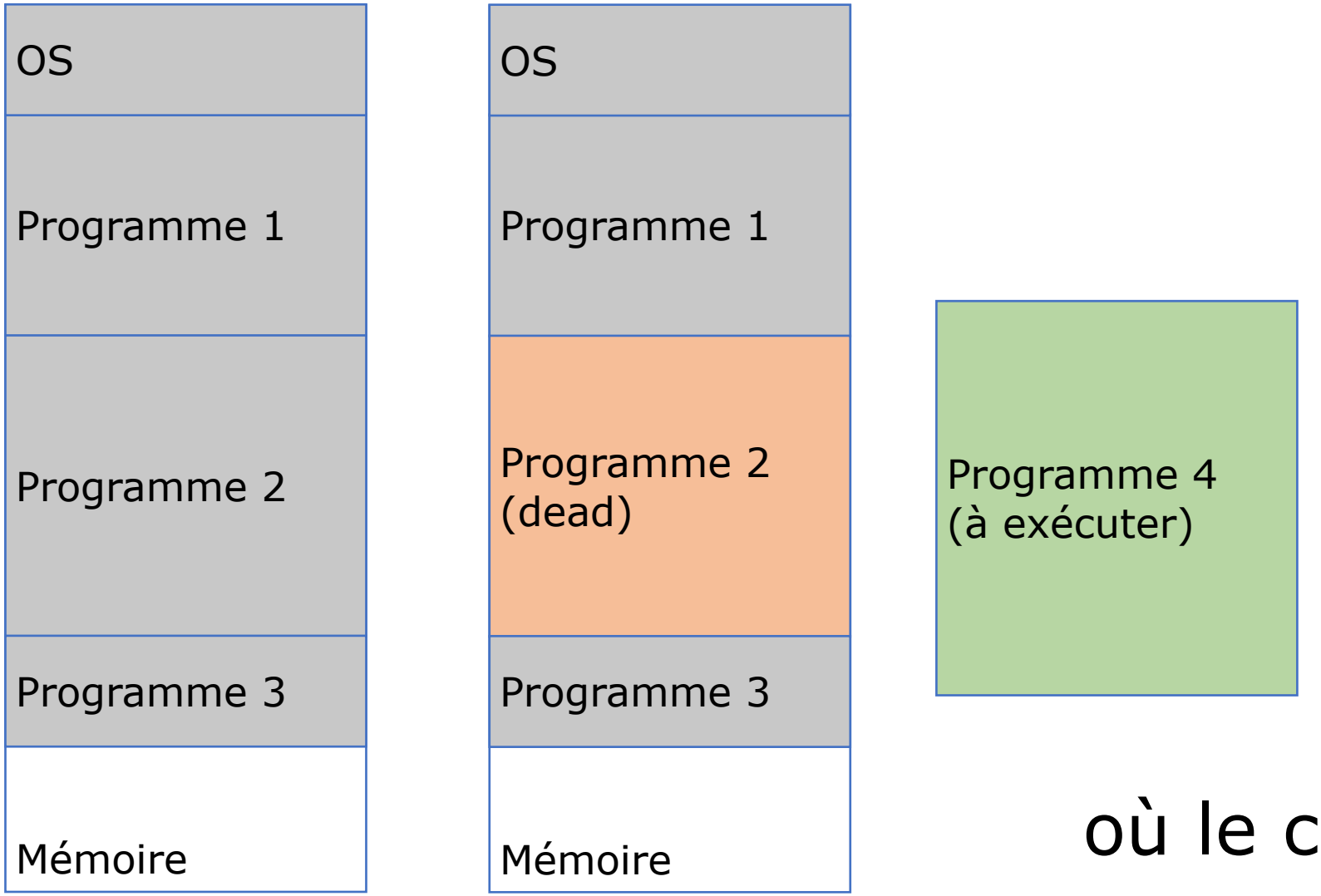
Avantage : on peut utiliser un ordonnanceur

Inconvénient : fragmentation externe

GCOS6

Multitâche avec OS

Fragmentation



où le charger ?

Fragmentation

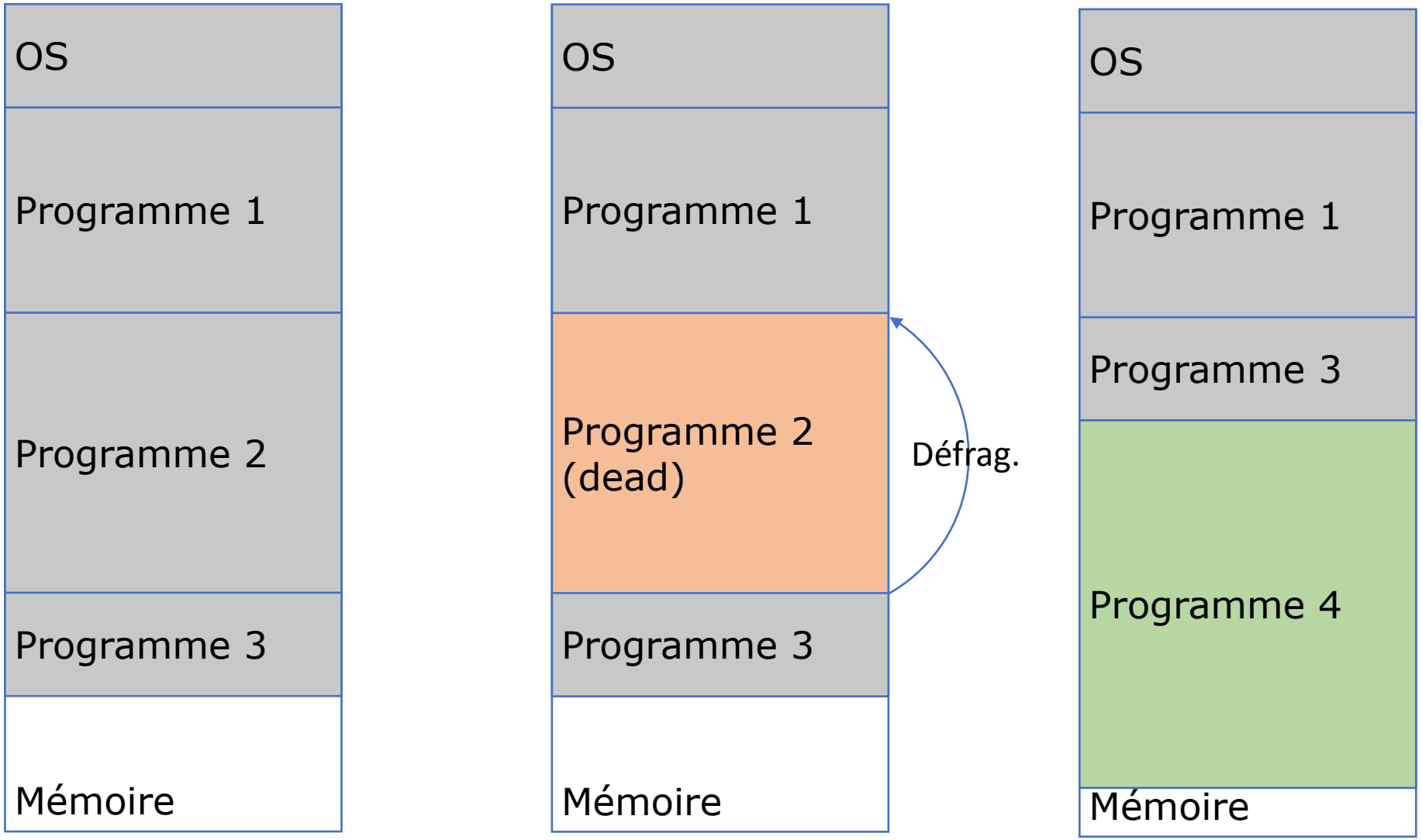
Au cours du temps la mémoire libre se fragmente en bouts de plus en plus petits…

Solution :

- on reloge les programmes de sorte à reconstruire des espaces libres et contigus de grande taille
  - défragmenteur

    - c'est pas une horreur Windowsienne ça ? Si mais c'était pour les disques (même histoire en fait)
    - mon petit doigt (CTO de Glande @Home) m'a raconté qu'il défragmentait sous GCOS6 en faisant les choses à la main : il arrêtait certains processus pour en lancer d'autres…

Fragmentation



Fragmentation

La défragmentation est un procédé coûteux

- arrêter les exécutions
- copier des espaces
- reprendre des exécutions
  - impact sur l'état ?

Si l'on pouvait l'éviter

- ce que MS/DOS n'a pas su faire pour ses disques pendant longtemps…

Segmentation

Les exécutions sont logiquement divisés en différentes sections :

- le code-*text* (instructions)
- la pile-*stack* (fonctions, paramètres)
- le tas-*heap* (allocation dynamique)
- données-*data*/BSS (variables globales)

```
ask> size p
__TEXT   __DATA   __OBJC  others     dec         hex
4096     4096     0       4294971392 4294979584 100003000
```

Segmentation

Plutôt que de considérer un programme comme un seul espace

- Le découper en sous-espaces, les segments
  - code,
  - pile,
  - tas,
  - etc

Segmentation

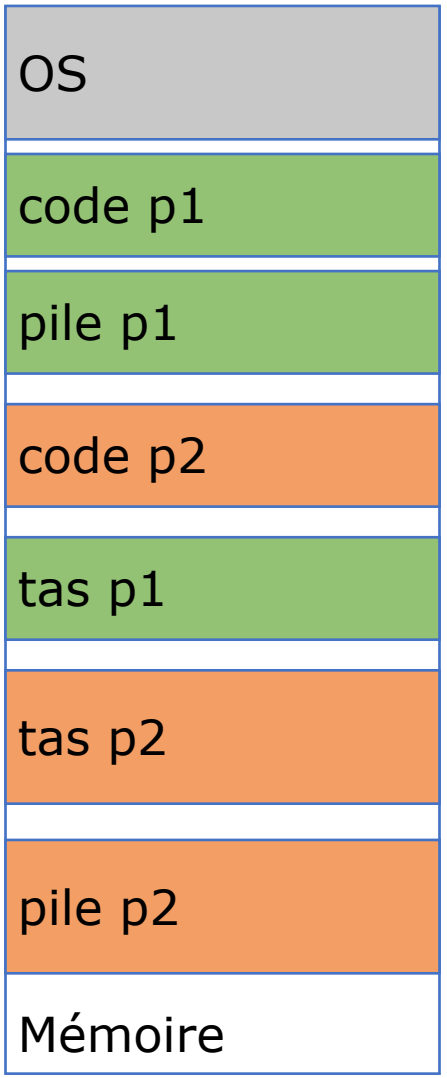
Avantage

- on ne défragmente plus en déplaçant un processus entier mais uniquement des segments (plus petites portions mémoire)
  - défragmentation moins fréquente, moins couteuse
- on peut envisager une gestion différenciée des segment

Conséquence

- un processus n'est plus nécessairement considéré comme un seul espace d'adresse contigu

Segmentation



Segmentation

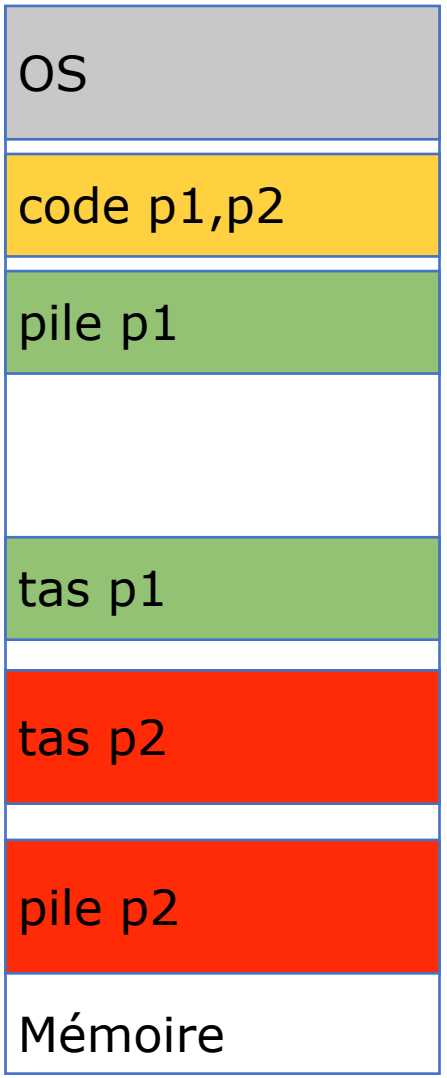
Du coup on peut envisager partager de la mémoire!

Impact sur la représentation dans le noyau ?

On peut plus facilement envisager de la protection mémoire à granularité fine

- code : protégé en écriture
- sortie du segment
  - tiens, tiens, `Segmentation fault`…

Segmentation



Support matériel pour la mémoire virtuelle (simple)

Mécanisme de conversion à la volée des adresses virtuelles utilisées par le processus en adresses physiques

- dans notre cas : simple
  - adresse de base pour la région de code, pile, données…
  - Ex. : registres x86
    - `CS` (Code Segment),
    - `DS` (Data Segment),
    - `SS` (Stack Segment),
    - `ES` (Extra Segment)

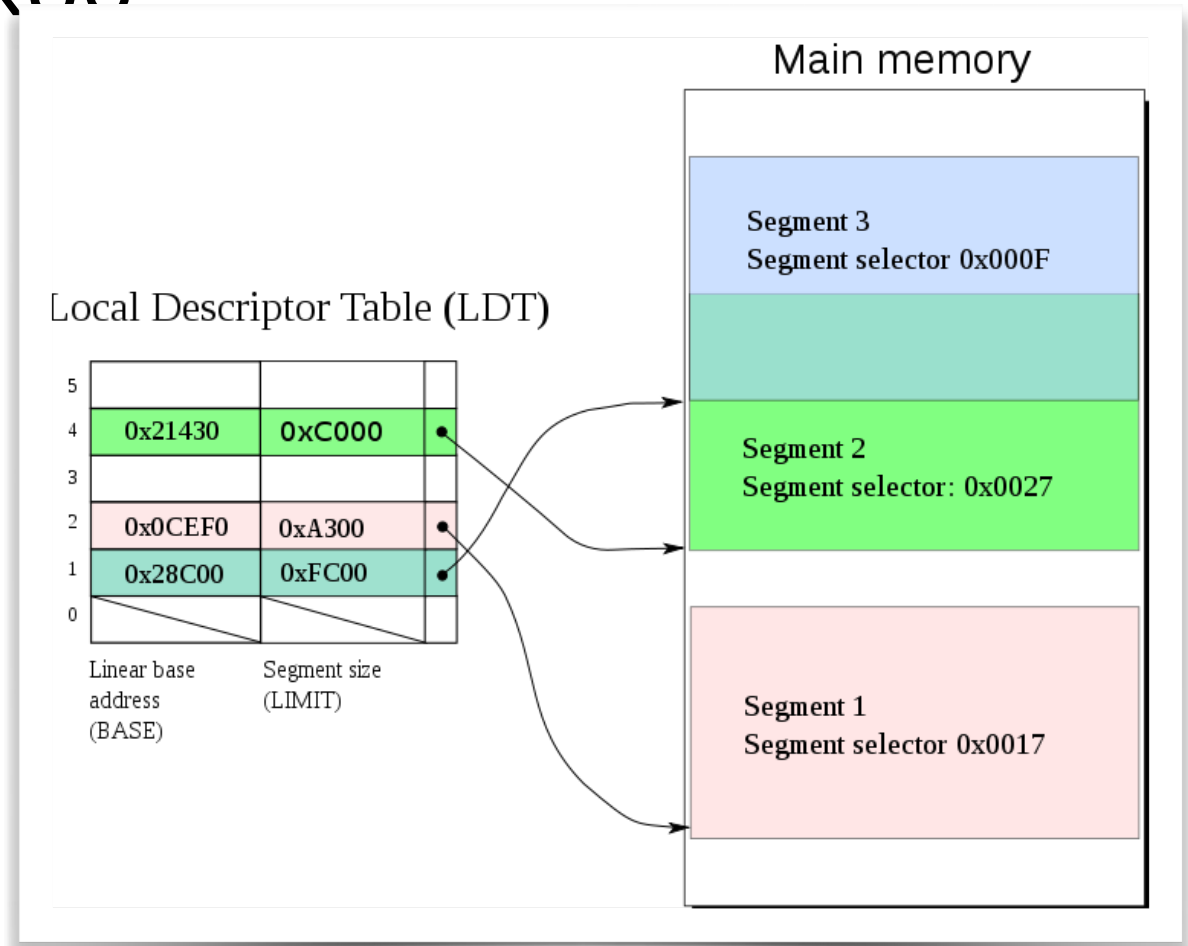
Régions / Segments

Un segment ou une région :

- espace mémoire contigu uniforme du point de vue de sa gestion (droits,…)
- `brk/sbrk`
- `exec`
- `fork`
- `mmap`
- `shmat`

Les segments Intel

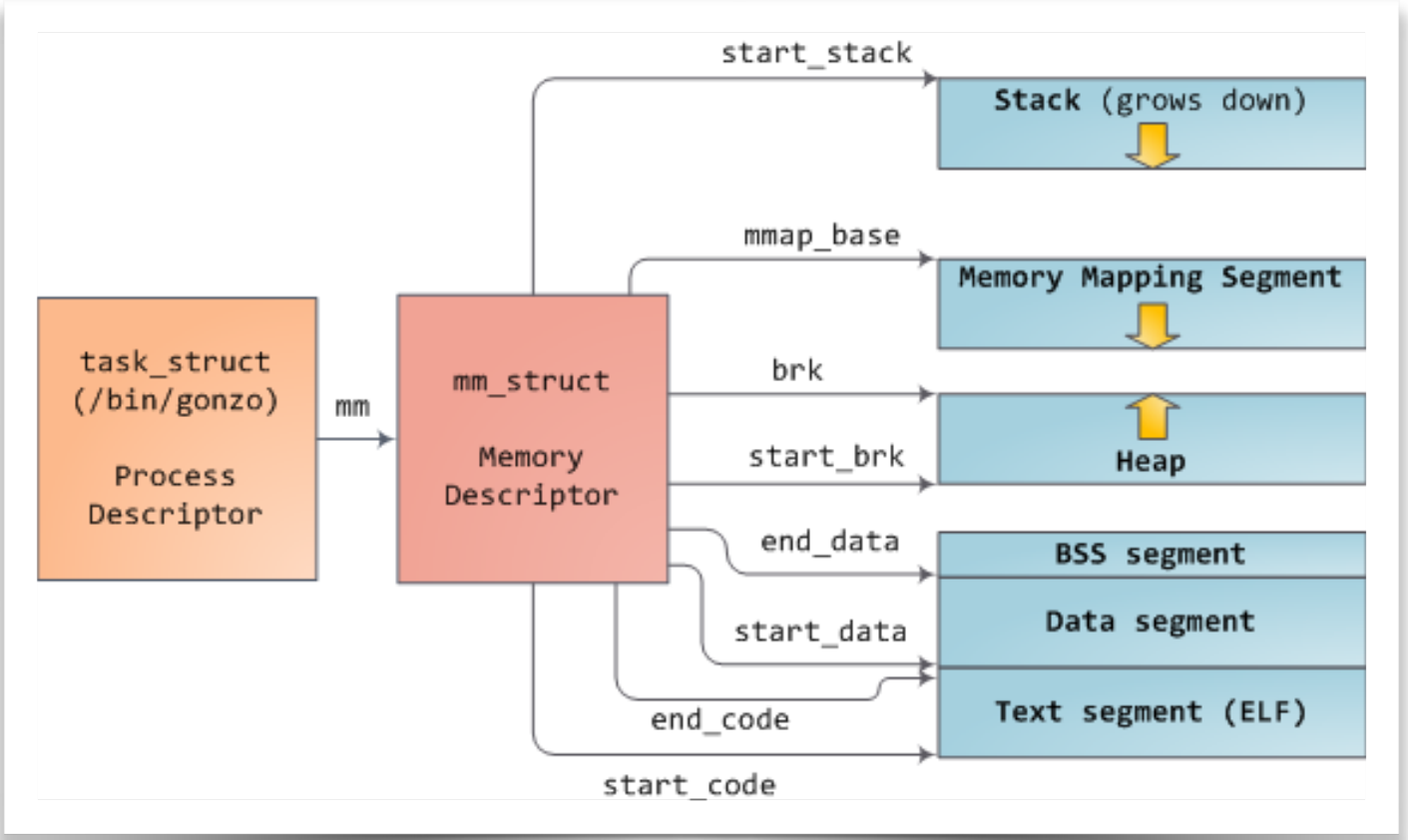
Ce concept est implanté dans la famille des processeurs Intel x86



Régions (Linux)

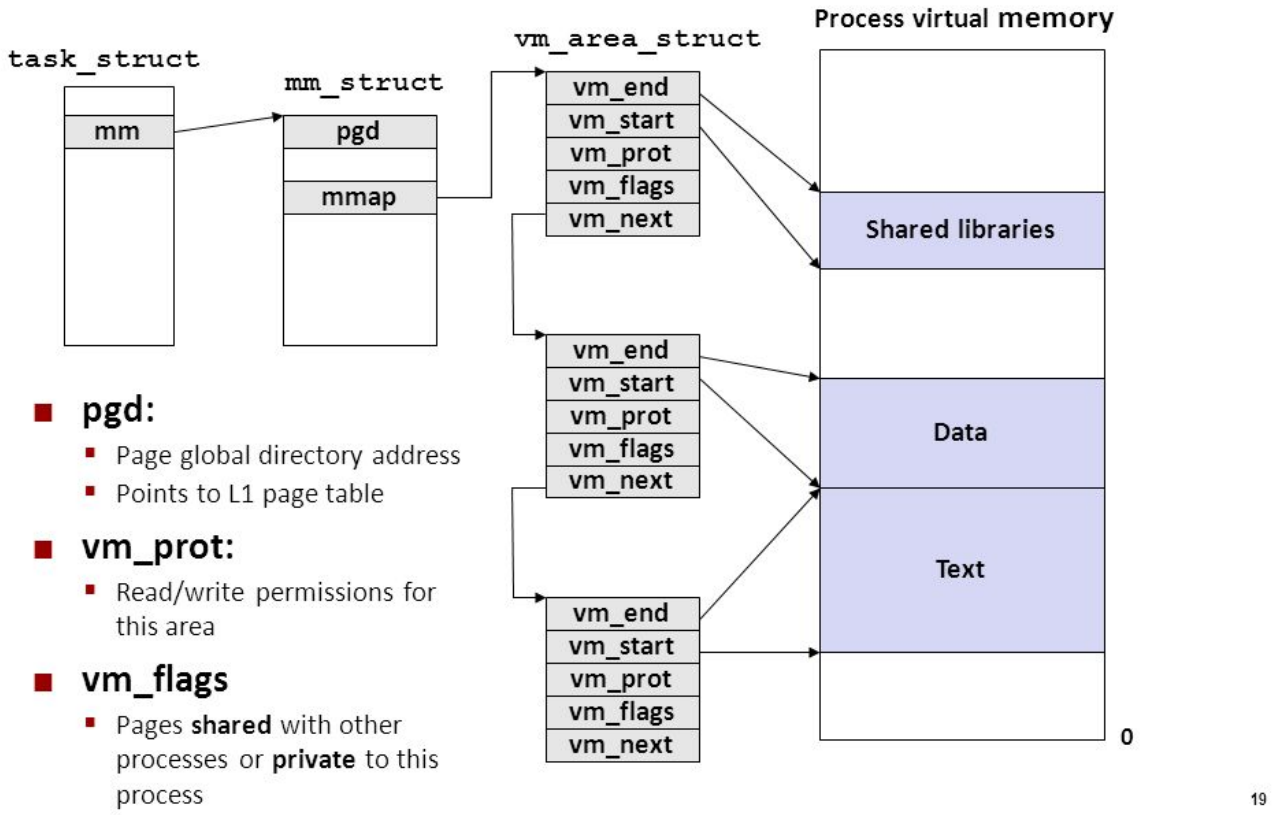
À chaque processus (`task_struct`) est associé la description de son espace d'adressage

`mm_struct`



Régions (Linux)

Chaque segment de son espace d'adressage est décrit par une structure `vm_area_struct`



Régions (Linux)

```
264    struct vm_area_struct {
265            /* The first cache line has the info for VMA tree walking. */
266
267            unsigned long vm_start;          /* Our start address within vm_mm. */
268            unsigned long vm_end;            /* The first byte after our end address
269                                                within vm_mm. */
270
271            /* linked list of VM areas per task, sorted by address */
272            struct vm_area_struct *vm_next, *vm_prev;
```

```
340    struct mm_struct {
341            struct {
342                    struct vm_area_struct *mmap;            /* list of VMAs */
```

linux/mm_types.h

Segmentation

La segmentation est généralement prise en charge par le MPU

- Memory Protection Unit
  - découpage en régions/segments
  - translation d’adresse élémentaire
  - protection

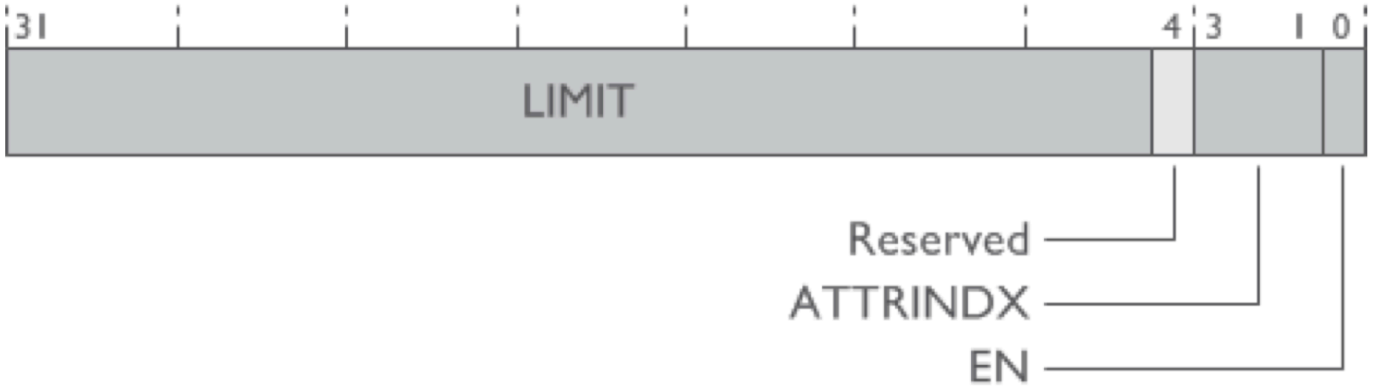
ARM MPU

Table 3-1 MPU registers

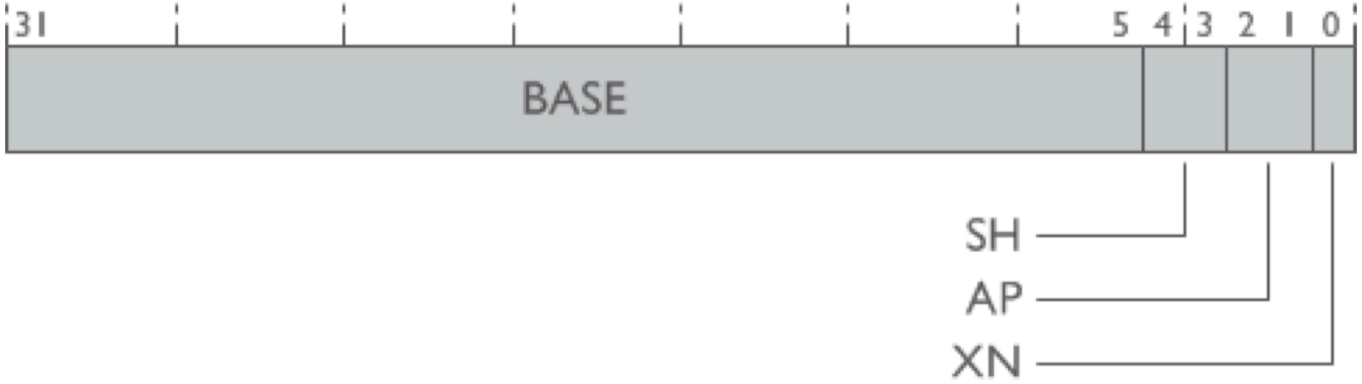
Register	Address	NS Address Alias	Description
MPU_TYPE	0xE000ED90	0xE002ED90	MPU Type Register
MPU_CTRL	0xE000ED94	0xE002ED94	MPU Control Register
MPU_RNR	0xE000ED98	0xE002ED98	MPU Region Number Register
MPU_RBAR	0xE000ED9C	0xE002ED9C	MPU Region Base Address Register
MPU_RLAR	0xE000EDA0	0xE002EDA0	MPU Region Base Limit Register
MPU_RBAR_A1	0xE000EDA4	0xE002EDA4	MPU Region Base Address Register Alias 1
MPU_RBAR_A2	0xE000EDAC	0xE002EDAC	MPU Region Base Address Register Alias 2
MPU_RBAR_A3	0xE000EDB4	0xE002EDB4	MPU Region Base Address Register Alias 3
MPU_RLAR_A1	0xE000EDA8	0xE002EDA8	MPU Region Limit Address Register Alias 1
MPU_RLAR_A2	0xE000EDB0	0xE002EDB0	MPU Region Limit Address Register Alias 2
MPU_RLAR_A3	0xE000EDA8	0xE002EDB8	MPU Region Limit Address Register Alias 3
MPU_MAIR0	0xE000EDC0	0xE002EDC0	MPU Memory Attribute Indirection Register 0
MPU_MAIR0	0xE000EDC4	0xE002EDC4	MPU Memory Attribute Indirection Register 1

ARM MPU

The MPU Region Limit Address Register defines the ending address of an MPU region, region enable, and an indirection index to memory attribute array.



The MPU Region Base Address Register defines the starting address of an MPU region and access permission.



Pagination

On peut voir la pagination comme la segmentation poussée à l'extrême

- pourquoi ne pas découper chaque espace en plus petits bouts ?
  - un petit bout = une page

On peut aussi voir la pagination comme une implémentation d'une mémoire virtuelle

- Φ : Mv $\longmapsto$ Mϕ
- Si cette fonction est définie par morceaux de taille fixe, on fait de la pagination
  - la taille des morceaux définit une page…
    - dans une page, la mémoire virtuelle est une mémoire simplement indexée…

Pagination

Elle fonctionne par ce qu'en général

- on a pas besoin de continuité physique
- à un instant donné, il n'y a pas nécessité d'avoir sous la main l'intégralité de la mémoire d'une exécution (voir **principes de localité** plus loin)
- l'espace d'un processus est généralement ridiculement petit par rapport à l'espace théorique

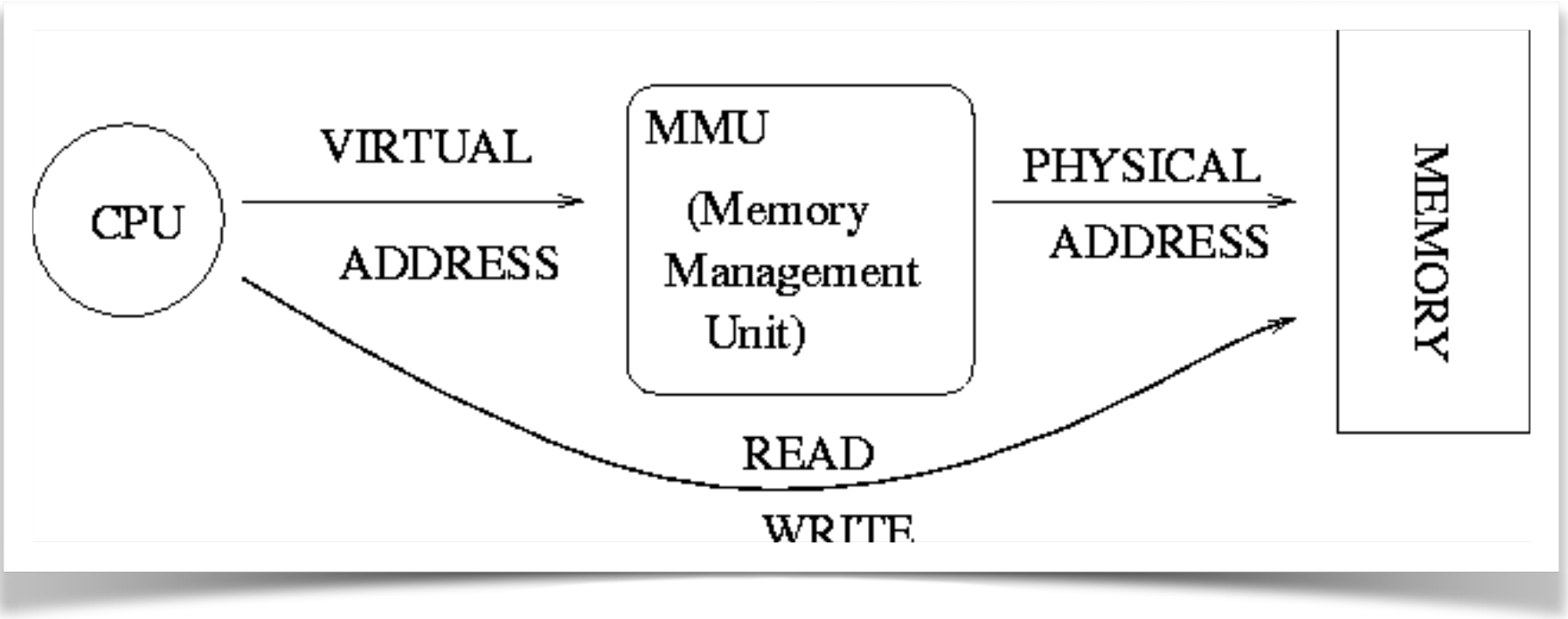
Pagination

La pagination est prise en charge par la MMU

- Memory Management Unit
  - définit la notion de mémoire virtuelle
  - autorise une gestion à granularité fine

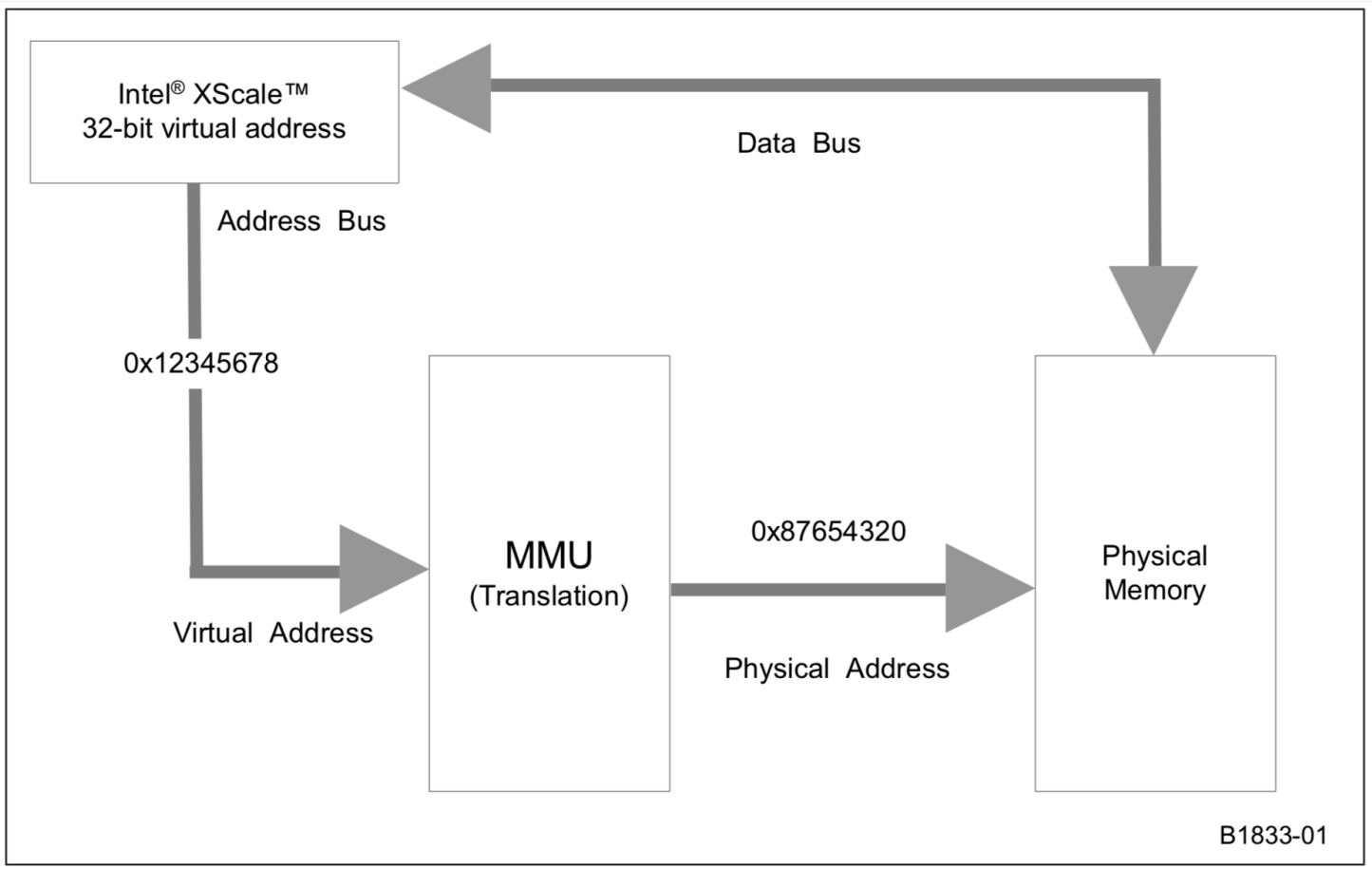
MMU

$\varphi_{address} = MMU(virtual_{address})$



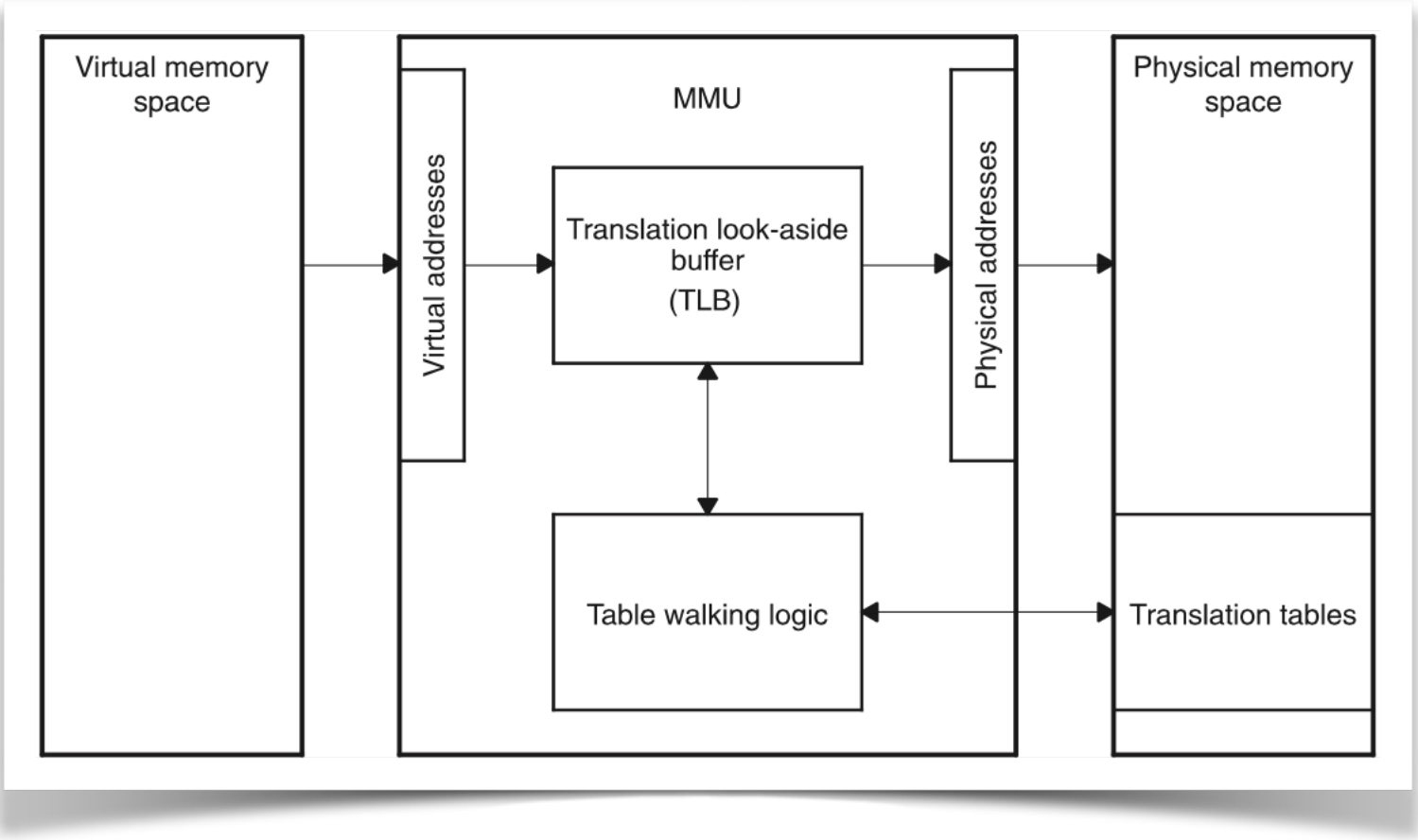
MMU

Schéma fonctionnel (Intel)



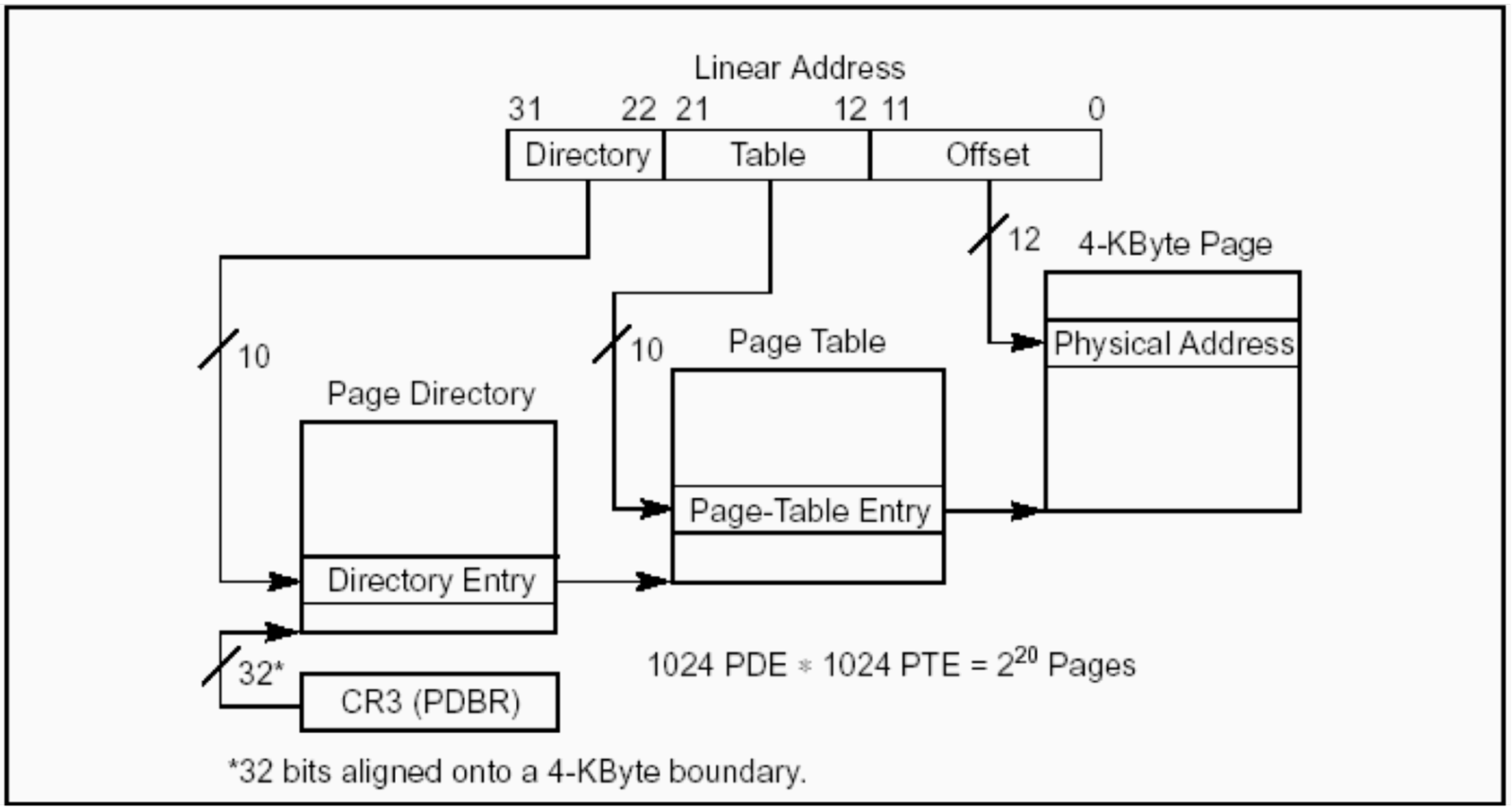
MMU

Schéma fonctionnel

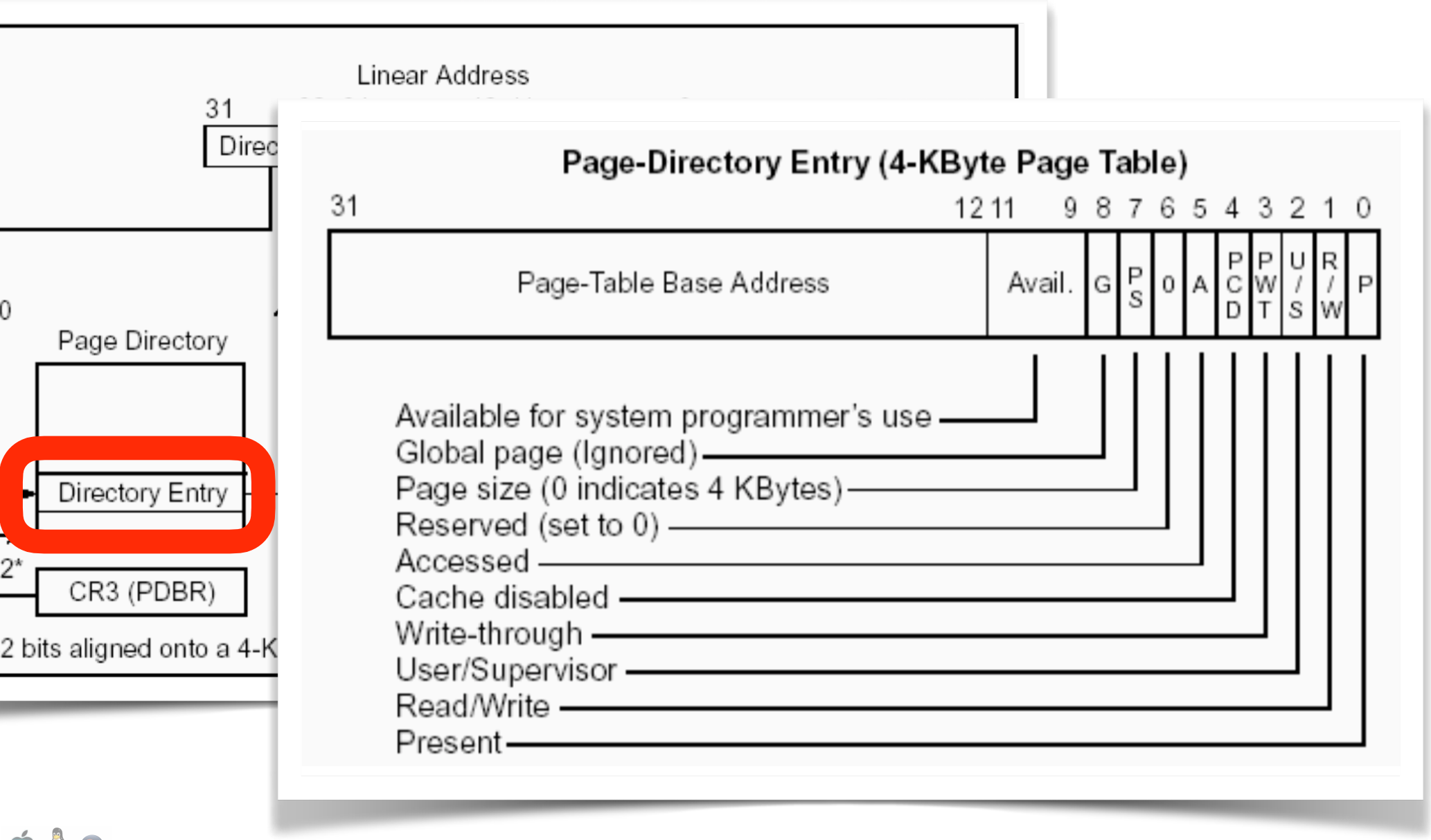


MMU

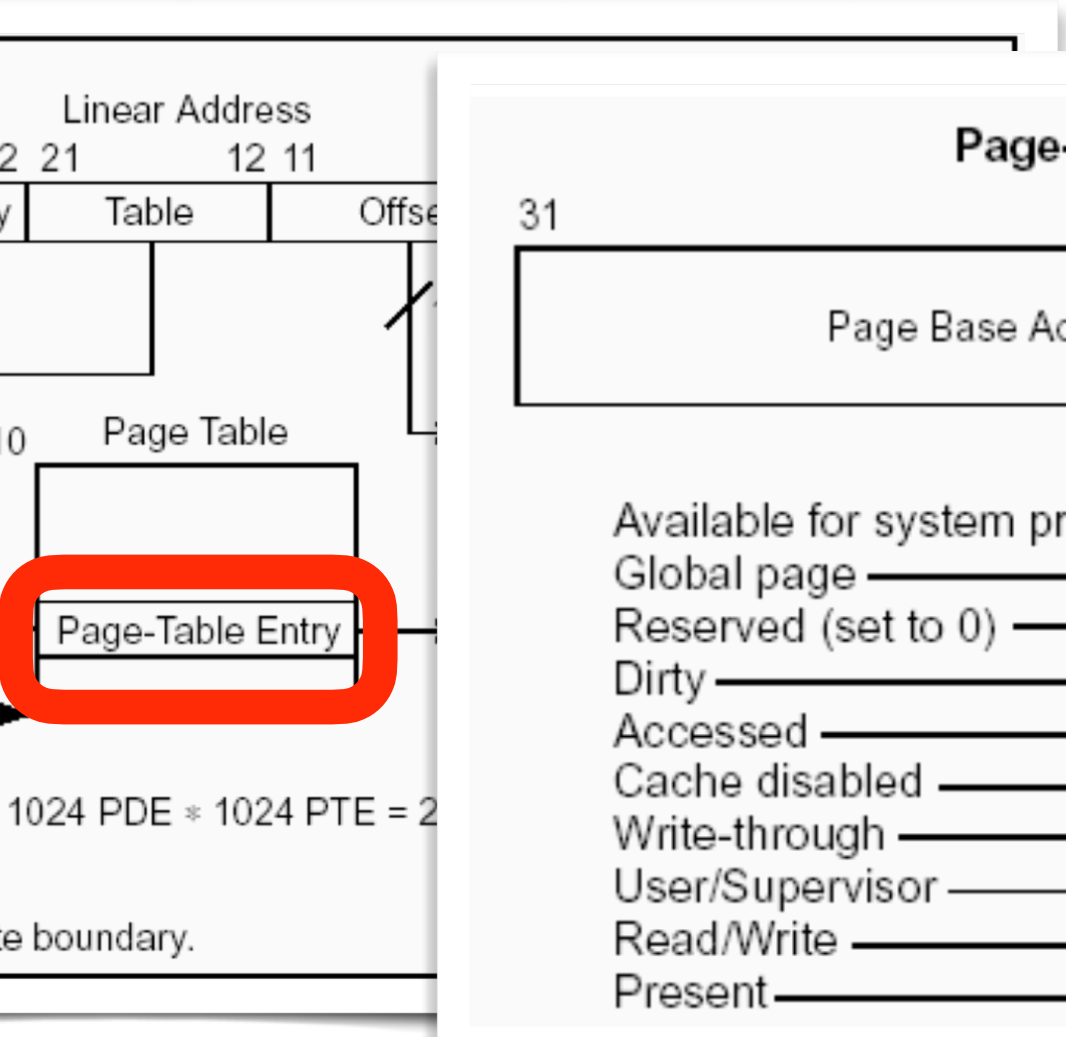
Vue logique Intel IA-32

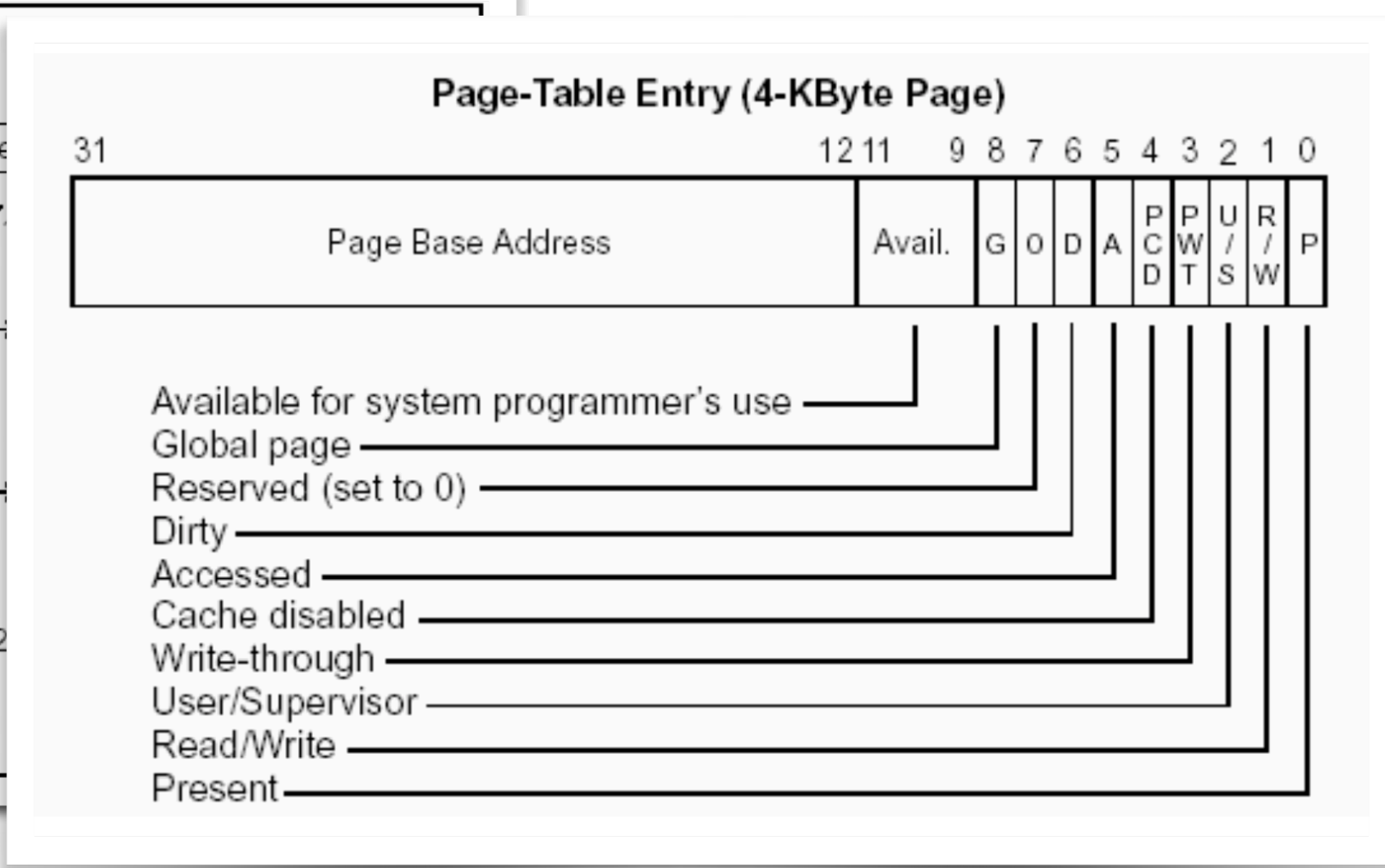


MMU (IA-32)

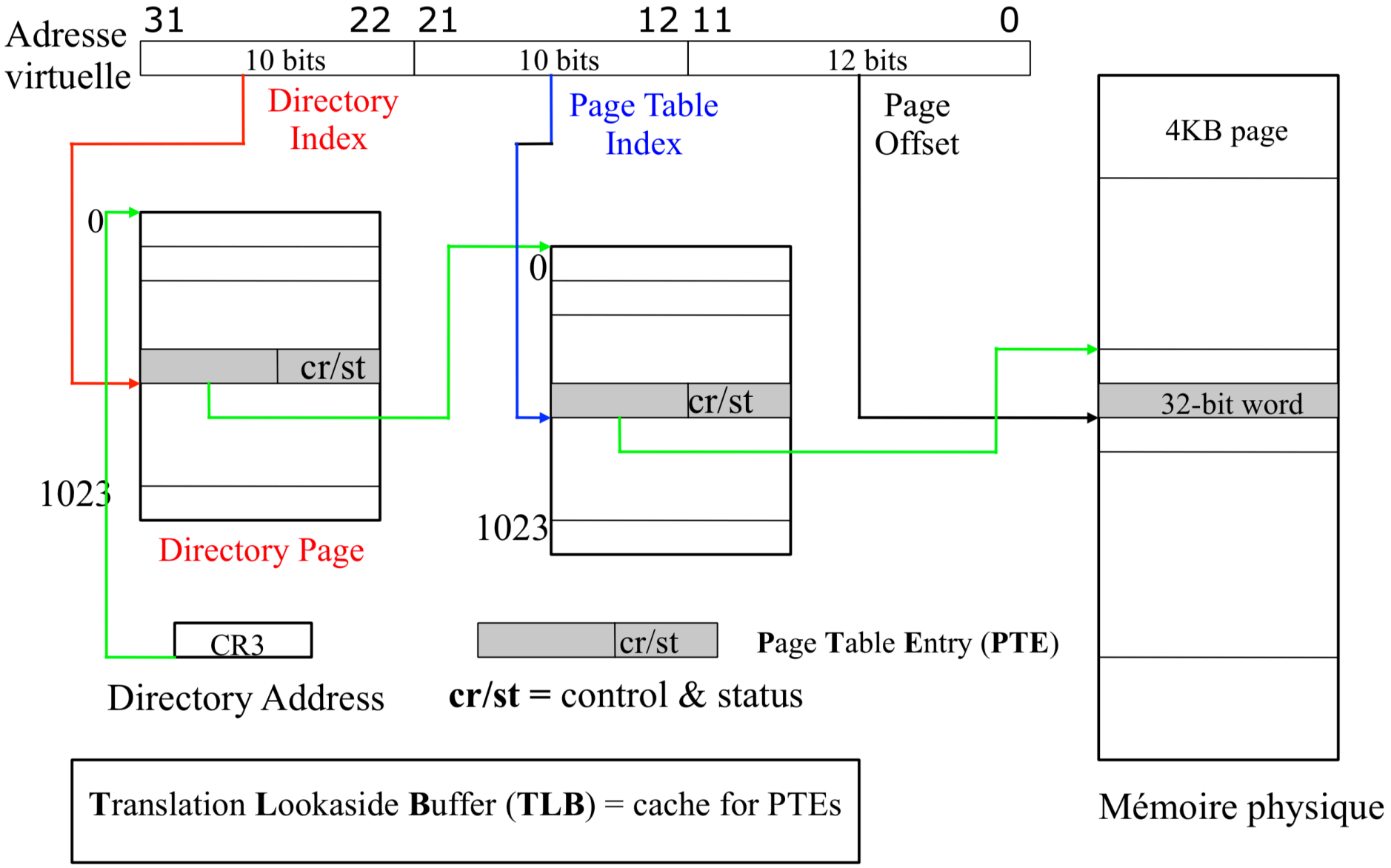

Linear Address
31
Direc
0
Page Directory
Directory Entry
2*
CR3 (PDBR)
2 bits aligned onto a 4-K
Page-Directory Entry (4-KByte Page Table)
31
12 11
9
8
7
6
5
4
3
2
1
0
Page-Table Base Address
Avail.
G
PS
0
A
PCD
PWT
U/S
R/W
P
Available for system programmer's use
Global page (Ignored)
Page size (0 indicates 4 KBytes)
Reserved (set to 0)
Accessed
Cache disabled
Write-through
User/Supervisor
Read/Write
Present

MMU (IA-32)


Linear Address
2 21
12 11
Table
Offse
10
Page Table
Page-Table Entry
1024 PDE * 1024 PTE = 2
e boundary.


Page-Table Entry (4-KByte Page)
31
12 11
9
8
7
6
5
4
3
2
1
0
Page Base Address
Avail.
G
0
D
A
PCD
PWT
U/S
R/W
P
Available for system programmer's use
Global page
Reserved (set to 0)
Dirty
Accessed
Cache disabled
Write-through
User/Supervisor
Read/Write
Present

MMU


Adresse virtuelle
31
22
21
12
11
0
10 bits
10 bits
12 bits
Directory Index
Page Table Index
Page Offset
4KB page
0
cr/st
1023
Directory Page
0
cr/st
1023
32-bit word
CR3
Directory Address
cr/st
Page Table Entry (PTE)
cr/st = control & status
Translation Lookaside Buffer (TLB) = cache for PTEs
Mémoire physique

MMU

Ok 1024 tables de 1024 entrées…

c’est gros!

Oui mais

toutes les entrées ne sont pas utiles

espace à trous, beaucoup d’entrées inutiles

les tables peuvent être en mémoire!

et si elles ne le peuvent pas ?

swap!

MMU

L'espace pour "Hello World!"

Ubuntu 16.04LTS Xenial Xerus

```
ask> cat /proc/24944/maps
00400000-00401000 r-xp 00000000 08:01 140132                             /home/parallels/tmp/p
00600000-00601000 r--p 00000000 08:01 140132                             /home/parallels/tmp/p
00601000-00602000 rw-p 00001000 08:01 140132                             /home/parallels/tmp/p
01378000-01399000 rw-p 00000000 00:00 0                                  [heap]
7efed07cc000-7efed098c000 r-xp 00000000 08:01 2883760                    /lib/x86_64-linux-gnu/libc-2.23.so
7efed098c000-7efed0b8c000 ---p 001c0000 08:01 2883760                    /lib/x86_64-linux-gnu/libc-2.23.so
7efed0b8c000-7efed0b90000 r--p 001c0000 08:01 2883760                    /lib/x86_64-linux-gnu/libc-2.23.so
7efed0b90000-7efed0b92000 rw-p 001c4000 08:01 2883760                    /lib/x86_64-linux-gnu/libc-2.23.so
7efed0b92000-7efed0b96000 rw-p 00000000 00:00 0
7efed0b96000-7efed0bbc000 r-xp 00000000 08:01 2883677                    /lib/x86_64-linux-gnu/ld-2.23.so
7efed0da1000-7efed0da4000 rw-p 00000000 00:00 0
7efed0dbb000-7efed0dbc000 r--p 00025000 08:01 2883677                    /lib/x86_64-linux-gnu/ld-2.23.so
7efed0dbc000-7efed0dbd000 rw-p 00026000 08:01 2883677                    /lib/x86_64-linux-gnu/ld-2.23.so
7efed0dbd000-7efed0dbe000 rw-p 00000000 00:00 0
7ffddc76d000-7ffddc78e000 rw-p 00000000 00:00 0                          [stack]
7ffddc790000-7ffddc793000 r--p 00000000 00:00 0                          [vvar]
7ffddc793000-7ffddc795000 r-xp 00000000 00:00 0                          [vdso]
ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0                  [vsyscall]
```

MMU

18 régions…

Les plus intéressantes

- celles du code, de la pile et du tas
- oublions les autres, elles ne changent rien à notre raisonnement

```
ask> cat /proc/24944/maps
00400000-00401000 r-xp 00000000 08:01 140132       /home/parallels/tmp/p
00600000-00601000 r--p 00000000 08:01 140132       /home/parallels/tmp/p
00601000-00602000 rw-p 00001000 08:01 140132       /home/parallels/tmp/p
01378000-01399000 rw-p 00000000 00:00 0            [heap]
7ffddc76d000-7ffddc78e000 rw-p 00000000 00:00 0    [stack]
```

MMU

Code : trois régions de taille 0x1000

Tas: une région de taille 0x21000

Pile : une région de taille 0x21000

```
ask> cat /proc/24944/maps
00400000-00401000 r-xp 00000000 08:01 140132       /home/parallels/tmp/p
00600000-00601000 r--p 00000000 08:01 140132       /home/parallels/tmp/p
00601000-00602000 rw-p 00001000 08:01 140132       /home/parallels/tmp/p
01378000-01399000 rw-p 00000000 00:00 0            [heap]
7ffddc76d000-7ffddc78e000 rw-p 00000000 00:00 0    [stack]
```

MMU

Ici une page 4Ko = 0x1000

donc espace total occupé par `p` : 69 pages!

Ce qui est ridicule…

- La Page Directory table est probable quasi-vide
- Donc très peu de place occupée pour décrire l'espace utile de `p`…

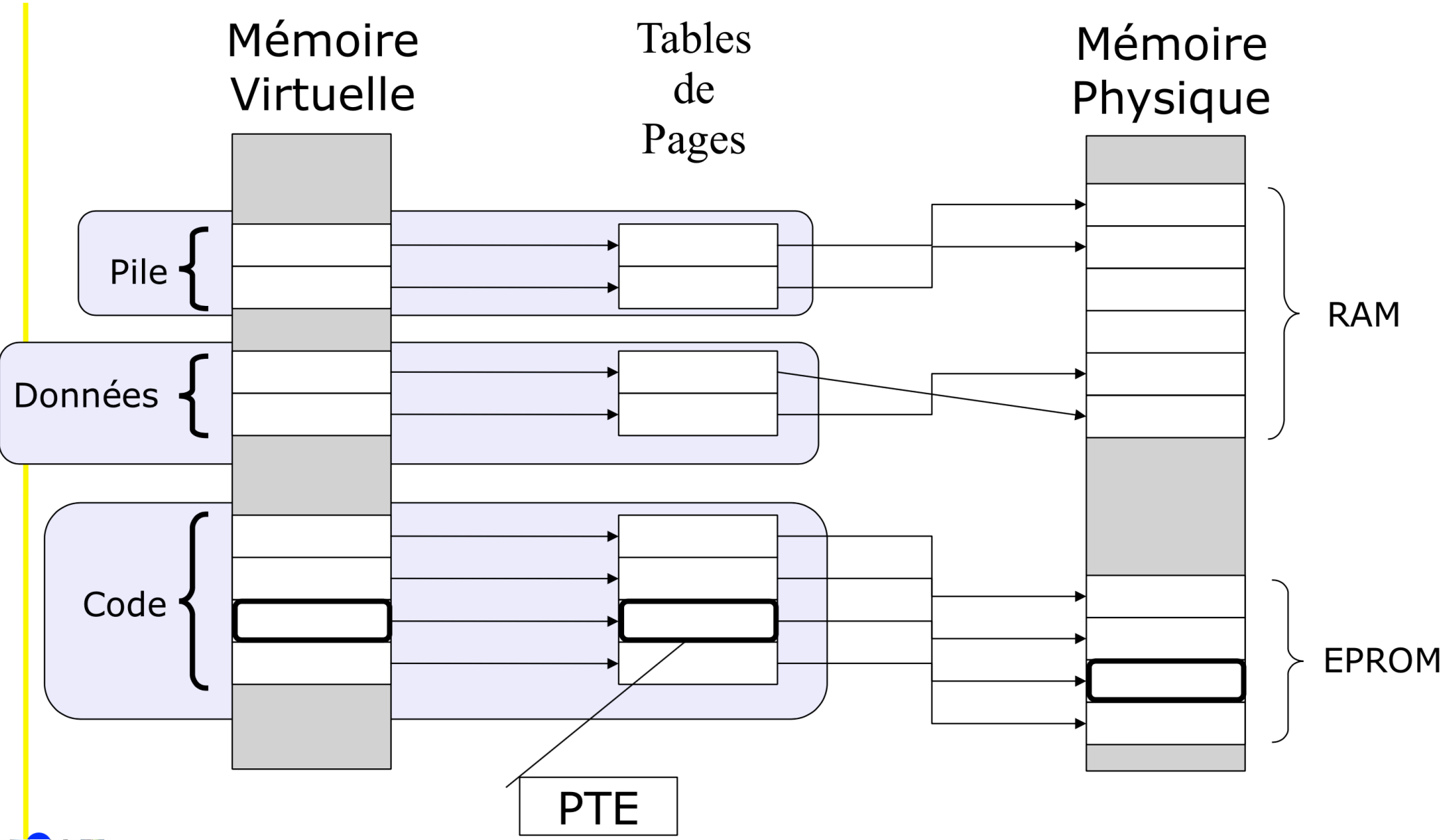
```
ask> cat /proc/24944/maps
00400000-00401000 r-xp 00000000 08:01 140132       /home/parallels/tmp/p
00600000-00601000 r--p 00000000 08:01 140132       /home/parallels/tmp/p
00601000-00602000 rw-p 00001000 08:01 140132       /home/parallels/tmp/p
01378000-01399000 rw-p 00000000 00:00 0            [heap]
7ffddc76d000-7ffddc78e000 rw-p 00000000 00:00 0    [stack]
```

MMU

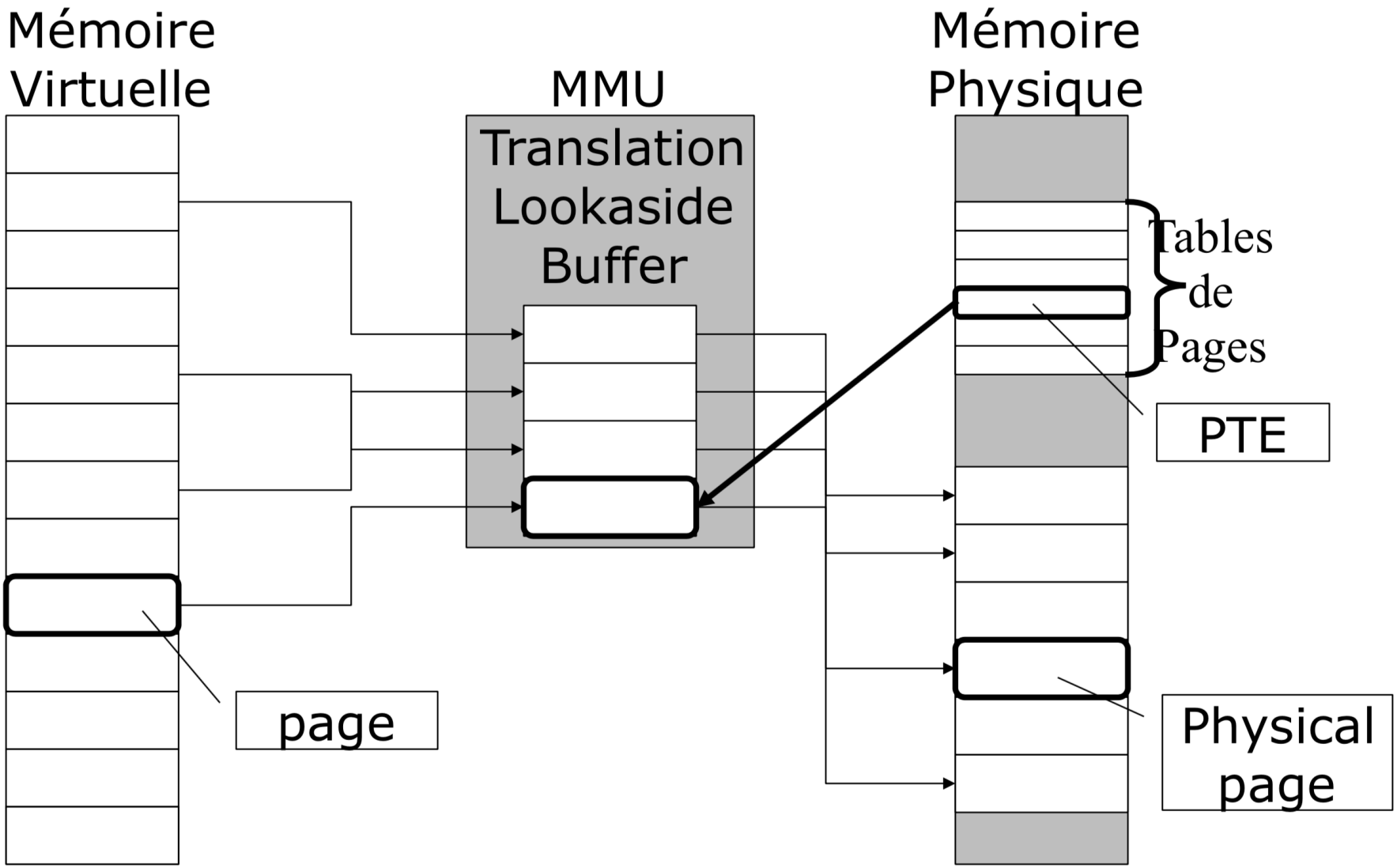
Taille d'une page ?

- Jusqu'ici fixe, quelques Ko
- Aujourd'hui, architecture avec page de taille variable!
  - «Back to the Future» : presque de la segmentation!

Exemple de mapping

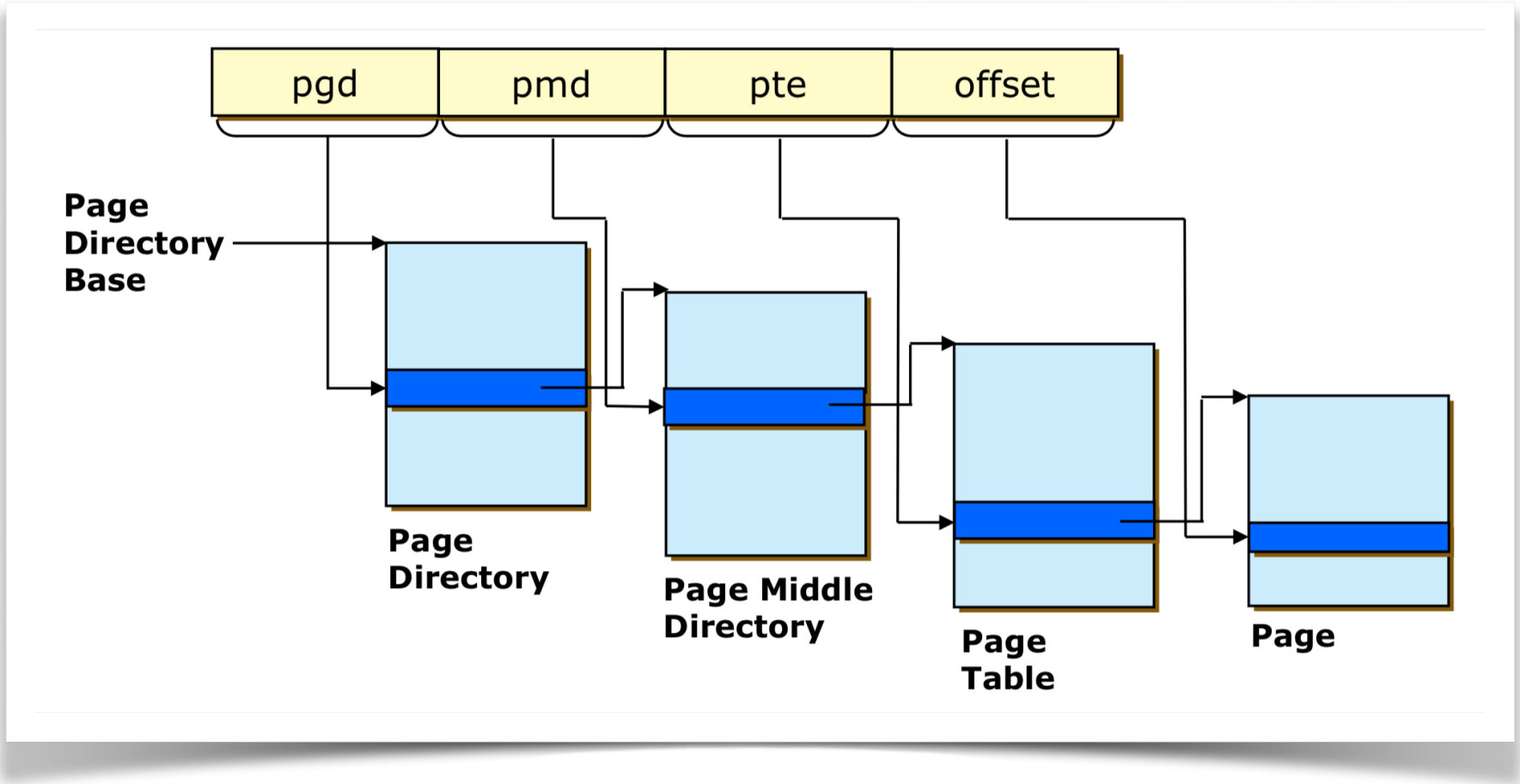


MMU


Mémoire
Virtuelle
MMU
Translation
Lookaside
Buffer
Mémoire
Physique
Tables
de
Pages
PTE
page
Physical
page

MMU

Vue logique Intel IA-64



Gestion mémoire

MPU : Memory Protection Unit

en charge du contrôle des accès à la mémoire

MMU : Memory Management Unit

en charge de la gestion de l'espace d'adressage

Caches Mémoire

But : accélérer les accès mémoire

Basé sur :

- le principe de **localité temporelle**
  - la probabilité d'utiliser au temps t+Δt la donnée utilisée au temps t est d'autant plus élevée que Δt est petit
- le principe de **localité spatiale**
  - la probabilité d'utiliser au temps t+1 une donnée voisine de celle utilisée au temps t est d'autant plus élevée que la distance entre les deux est faible

Caches Mémoire

```
int t[1000];

void f() {
  for (int i=0; i < 1000; i++){
    t[i]++;
  }
}
```

localité temporelle

localité spatiale

Y a t'il une autre localité implicite ici ?

Caches Mémoire

Attention, il s'agit de principes, ce ne sont donc pas des lois universelles…

- Cela dépend fortement de la façon d'utiliser les données
  - En général, ces principes sont respectés « naturellement » ou « instinctivement »
    - mais rien d'obligatoire
      - on peut violer ces principes…
        - code spaghetti, accès randomisé à la mémoire

Caches Mémoire

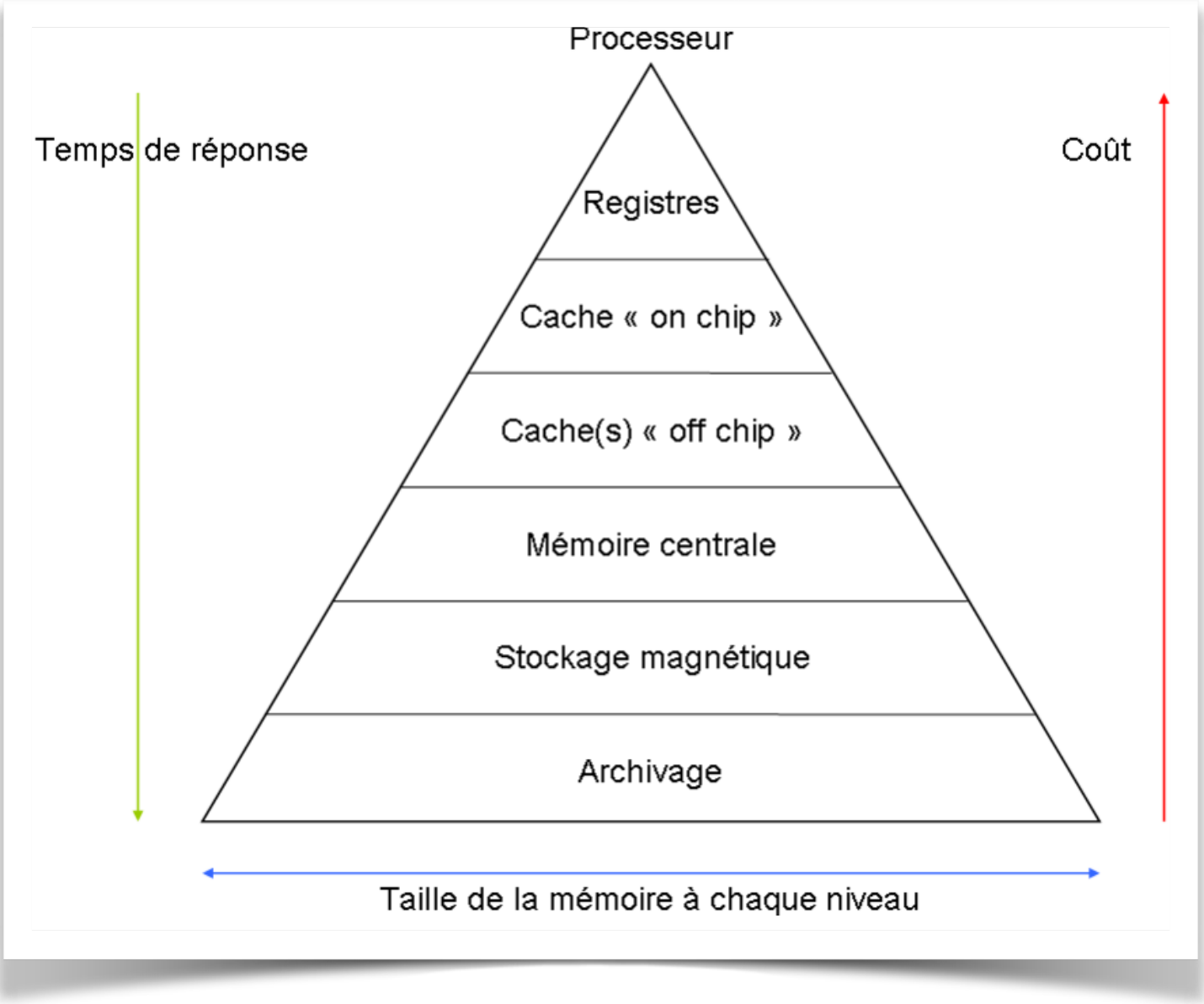
Idée :

- utiliser une hiérarchie mémoire afin d'accélérer les accès à la mémoire fréquemment utilisée
- utiliser une mémoire rapide comme copie d'une portion d'une mémoire plus lente

Ne pas se méprendre :

- plus la mémoire est grande, plus lents sont ses accès…

Hiérarchie mémoire



Cache Mémoire

Où place t-on les choses dans le cache ?

- direct-mapped
- fully-associative
- set-associative

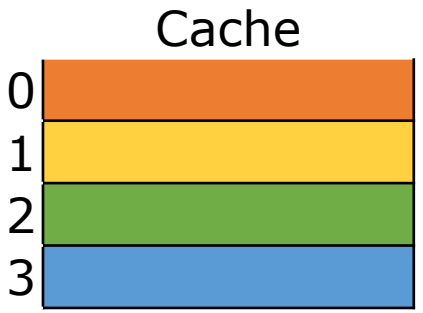
Direct-Mapped Cache

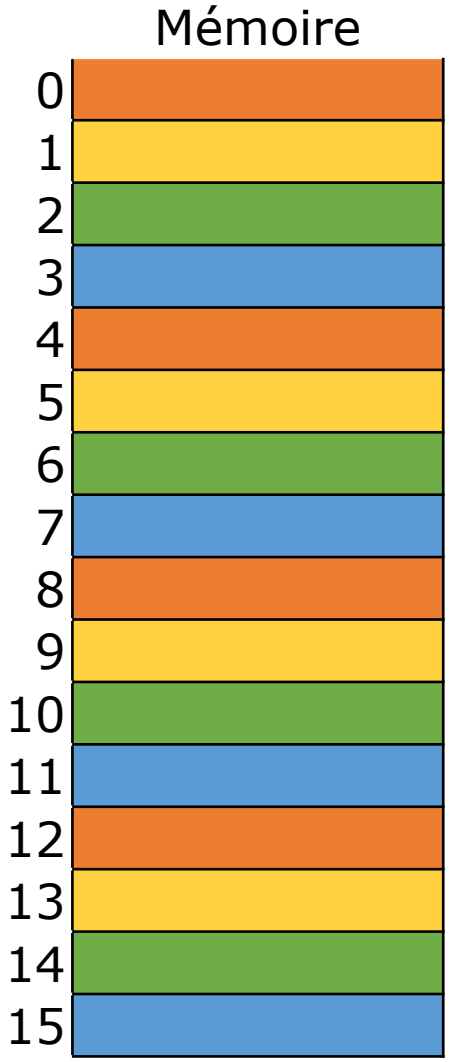
Mémoire de taille 2^n

Cache de taille 2^k, $k \ll n$

Donnée d'adresse x placée

dans le cache en $x\%2^k$

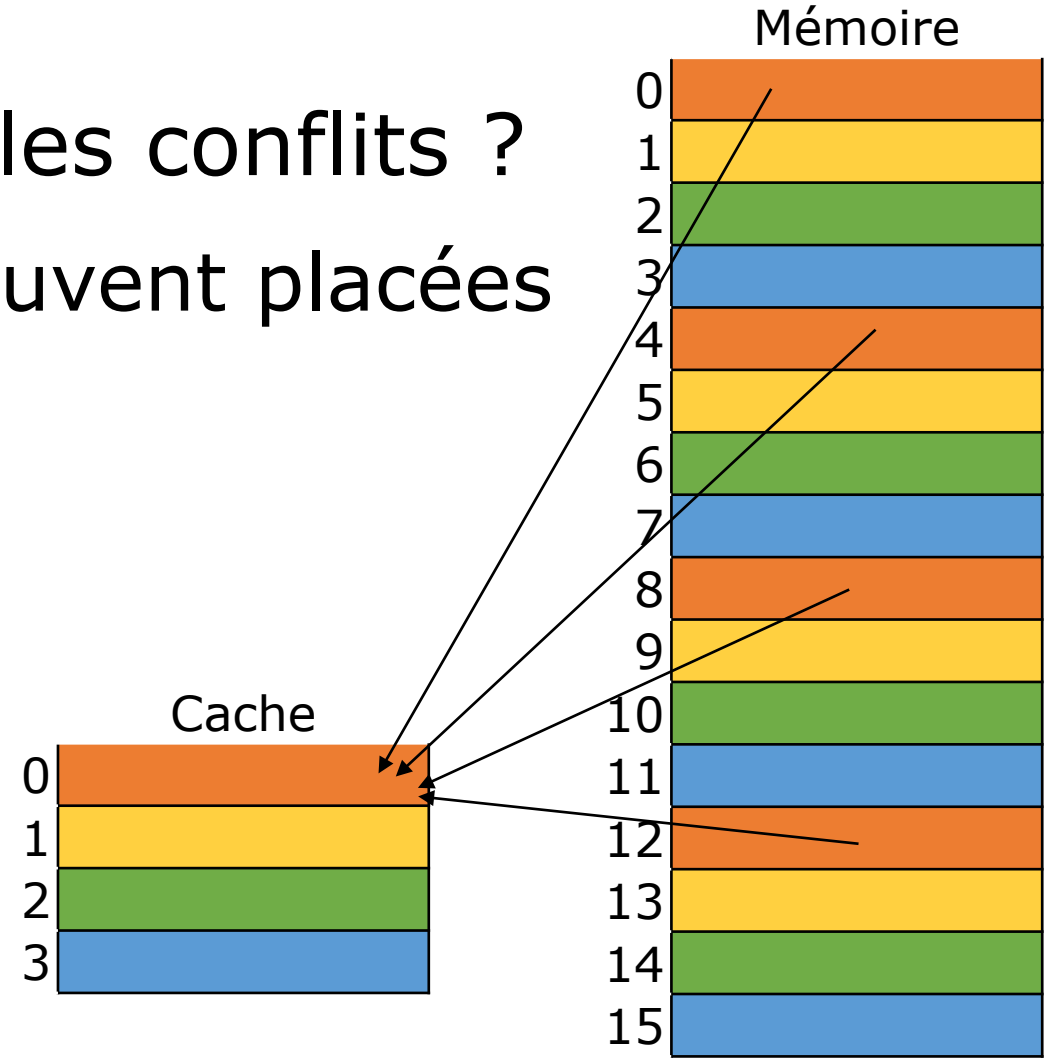




Direct-Mapped Cache

Mais comment gérer les conflits ?

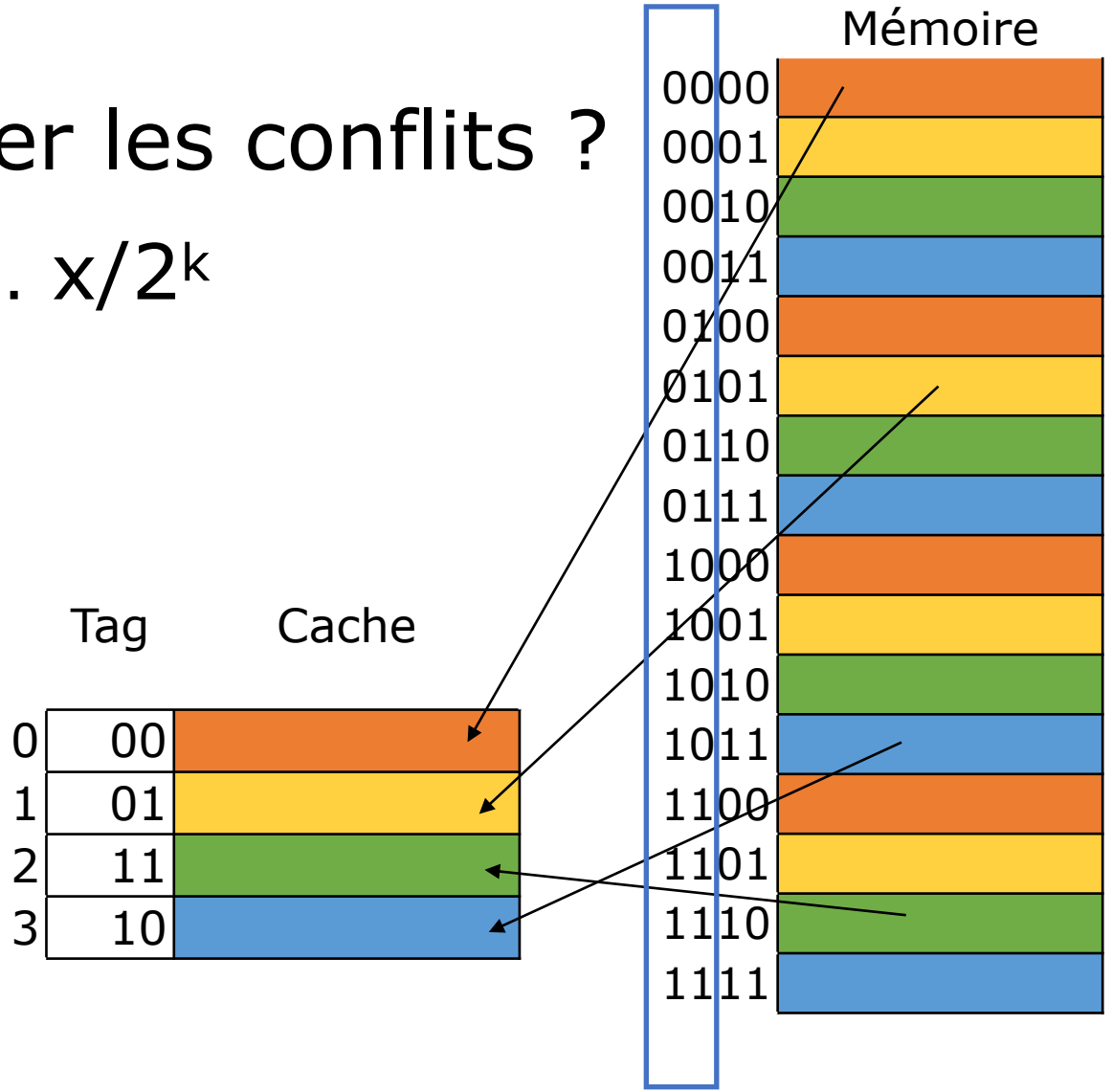
Plusieurs adresses peuvent placées au même endroit…



Direct-Mapped Cache

Mais comment gérer les conflits ?

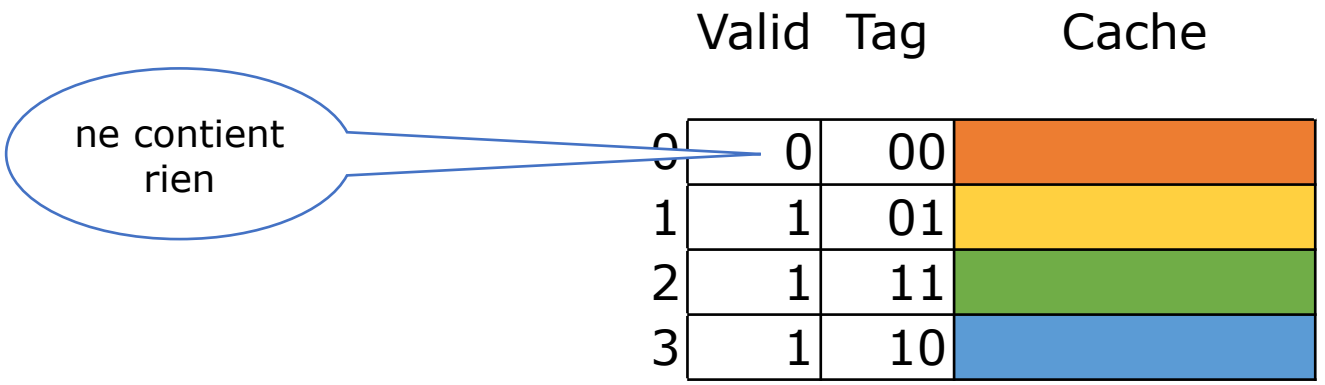
On rajoute un tag… $x/2^k$



Direct-Mapped Cache

Au démarrage (au moins) il faut distinguer une entrée valide d'une entrée ne correspondant à rien

On rajoute des données de contrôle



Cache hit

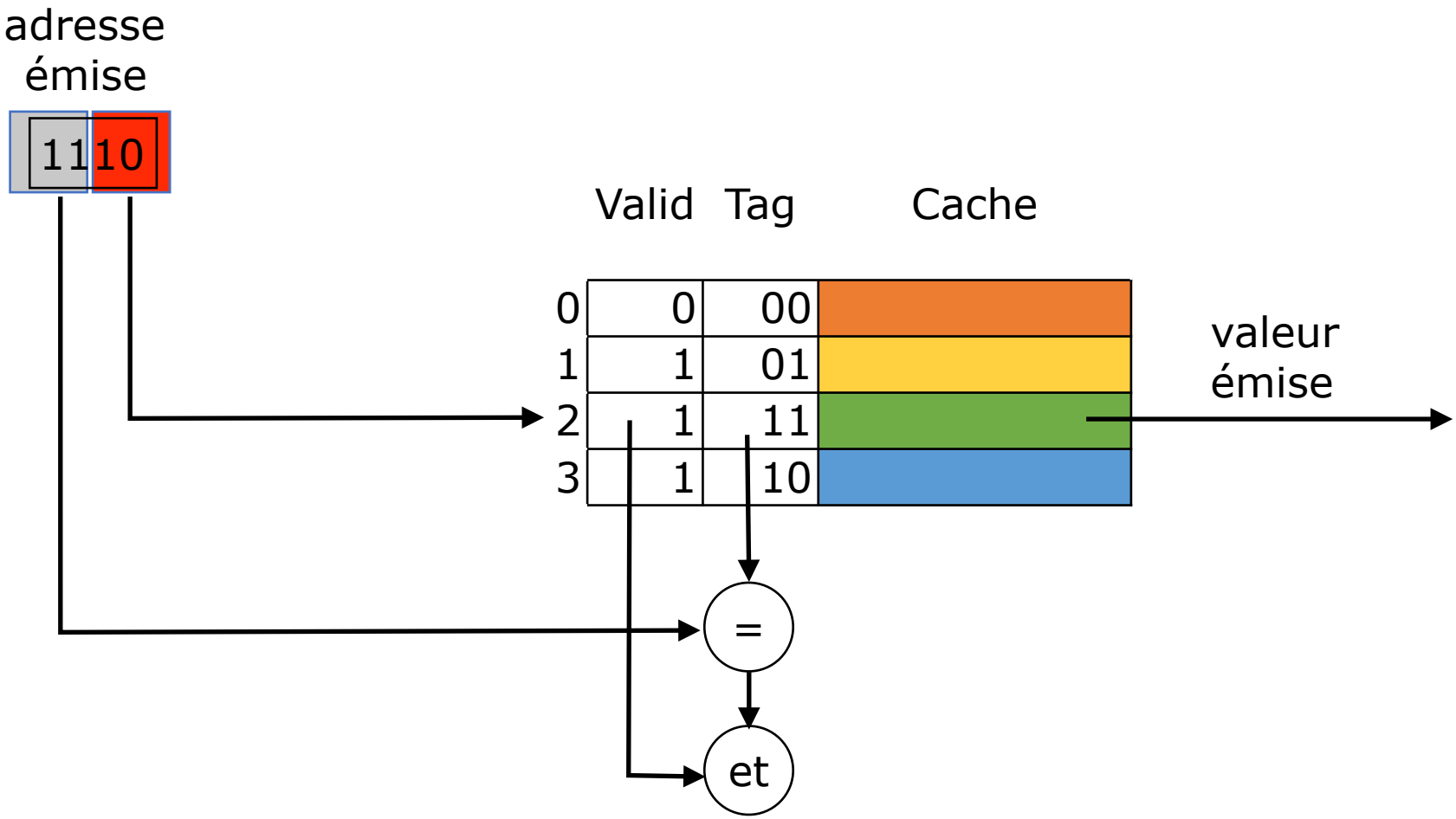
Le processeur émet une adresse x

On isole n-k et k bits

Si les n-k premiers bits correspondent au tag de l'entrée de numéro indiquée par les k derniers bits et que la validité est correcte

- les données correspondantes sont envoyées en retour

Cache hit


adresse
émise
11
10
Valid
Tag
Cache
0
1
2
3
0
1
1
1
00
01
11
10
valeur
émise
=
et

Cache miss

Si le tag ou le bit de validité ne correspondent pas

- l'adresse est émise vers la mémoire, puis la valeur est chargée dans le cache avec son tag et le bit de validité mis-à-jour et la valeur est émise vers le processeur

Direct-Mapped Cache

Une entrée de cache contient

- un bit de contrôle
- n-k bits de tag
- l bits de cache
  - qu'est ce qu'on cache ? octets, 32 bits, 64, combien ?

Il vaut mieux que l soit assez grand devant n-k+1…

- Les principes de localité permettent d'envisager placer un nombre assez grand pour ne pas trop gâcher de mémoire…

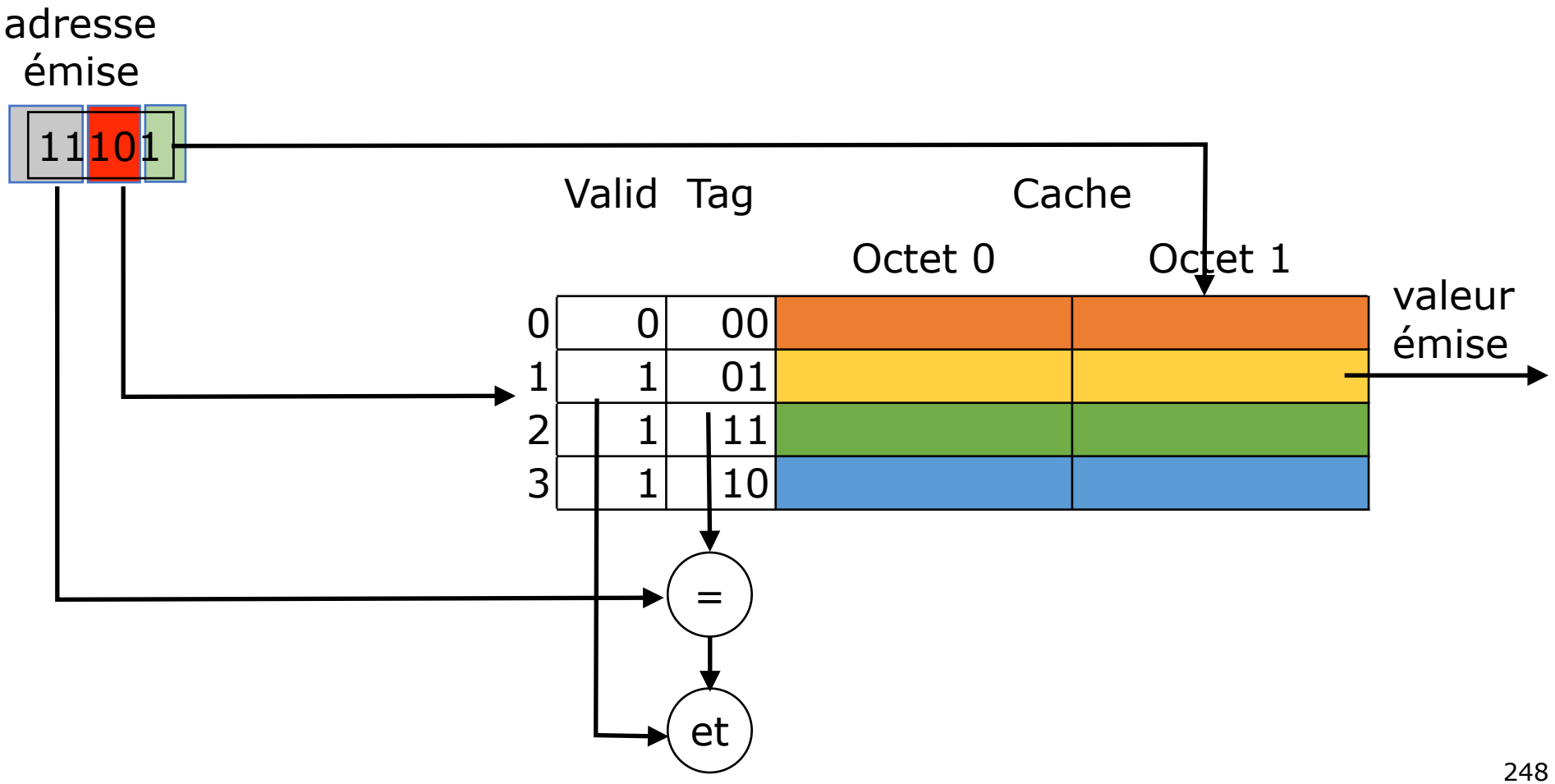
Direct-Mapped Cache

On peut cacher des blocks de mémoire (cache row)

- l'adresse est donc découpée en
  - b bits d'offset dans le bloc
  - k bits d'adresse dans le cache
  - n-k-b bits de tag

Direct-Mapped Cache

Exemple avec des blocs de 2 octets



Direct-Mapped Cache

Inconvénient :

- Si deux adresses correspondant à la même entrée de cache sont émises en alternance, on obtient que des caches misses
  - le cache n’apporte rien
  - pire il coûte…

Fully Associative Cache

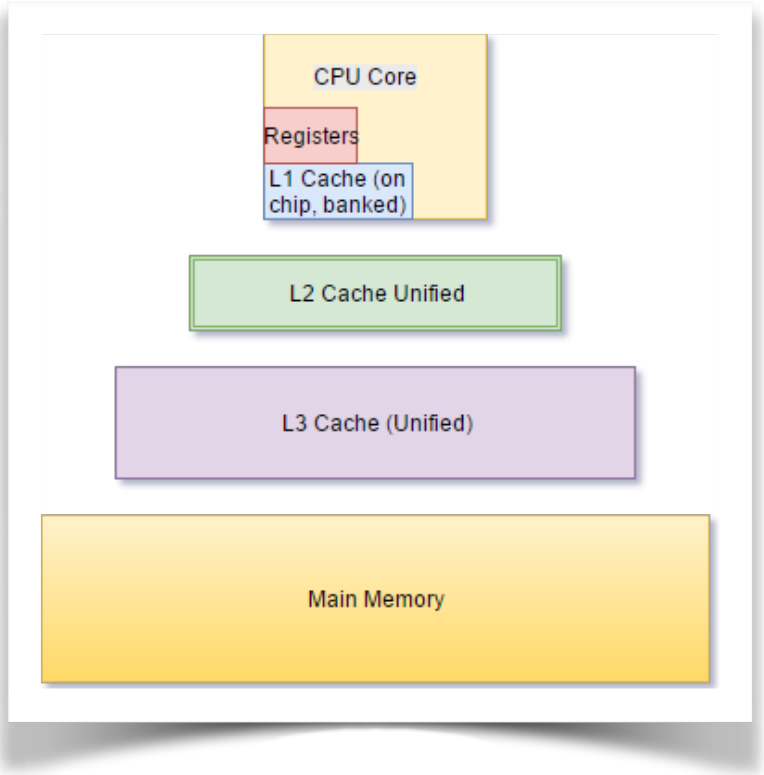
Cette fois à une adresse ne correspond pas un emplacement dédié dans le cache

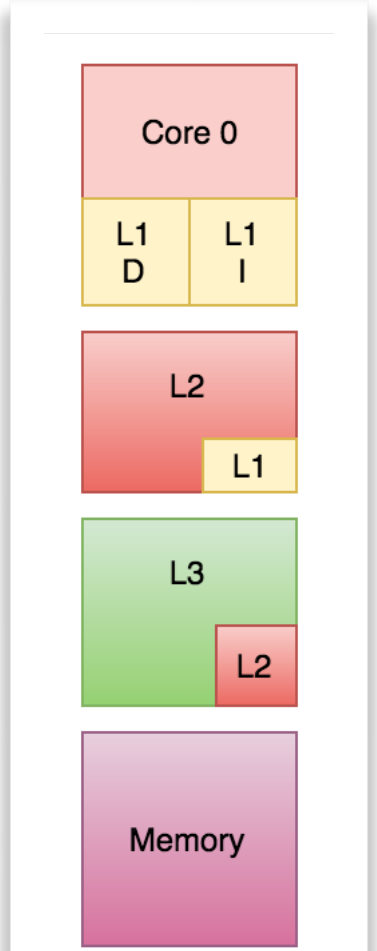
- les données peuvent être placées n'importe où
  - de la sorte le cas défavorable précédent ne peut apparaître (sous cette forme triviale…)
- si le cache est plein il faut trouver de quoi vider une entrée
  - en espérant ne pas vider une entrée qui va être bientôt utilisée
    - le principe de localité temporel à la rescousse!
      - LRU, du contrôle supplémentaire…

Fully Associative Cache

- Inconvénient
  - coûte cher en composants
    - il faut tester toutes les entrées en parallèles!
- La solution est de prendre un intermédiaire
  - un placement direct pour localiser un bloc lequel est associatif
    - ex. : 2^w-way associative

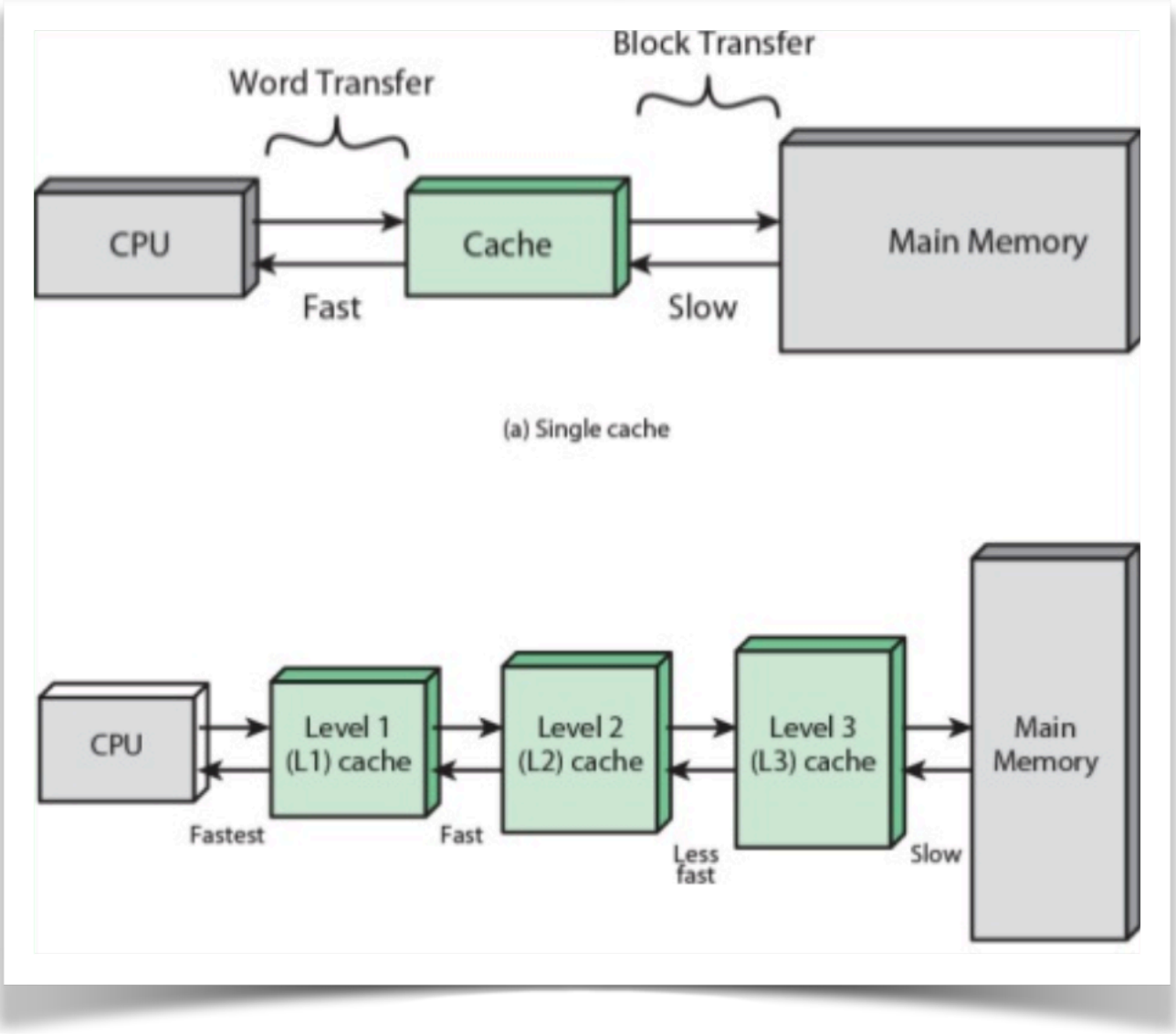
Hiérarchie de caches



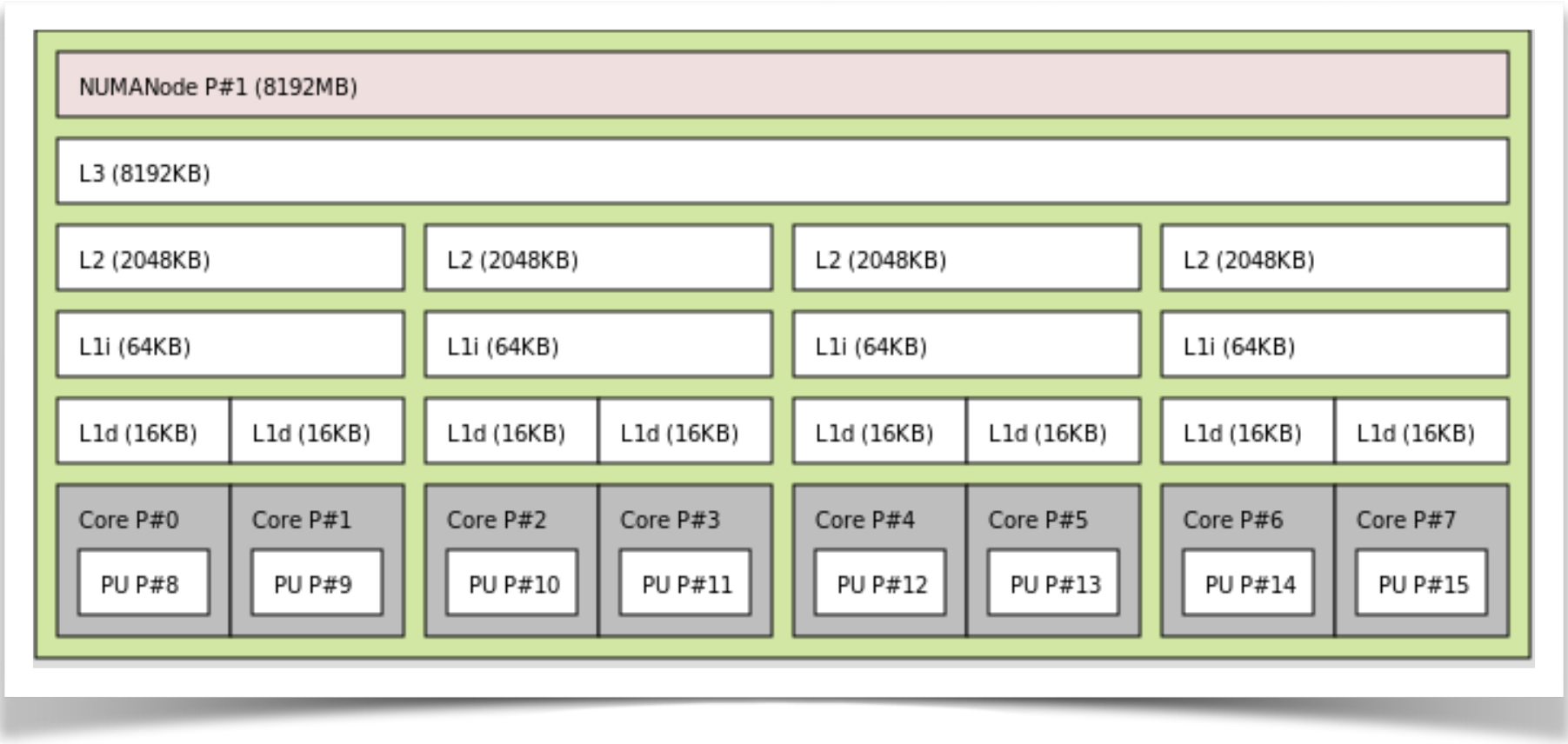


Hiérarchie inclusive

Caches Mémoire



Caches Mémoire



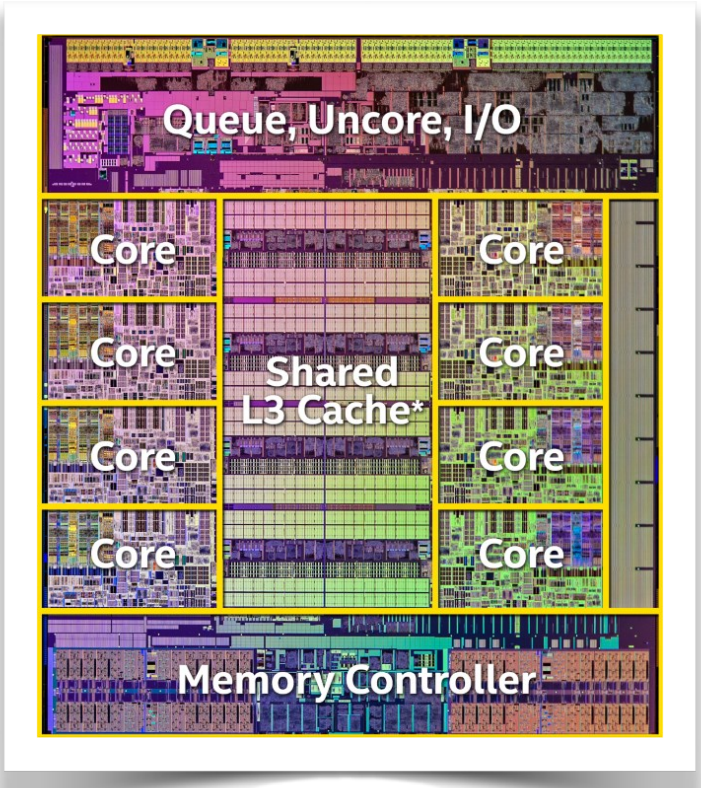
Hiérarchie de caches AMD Bulldozer

Caches mémoire

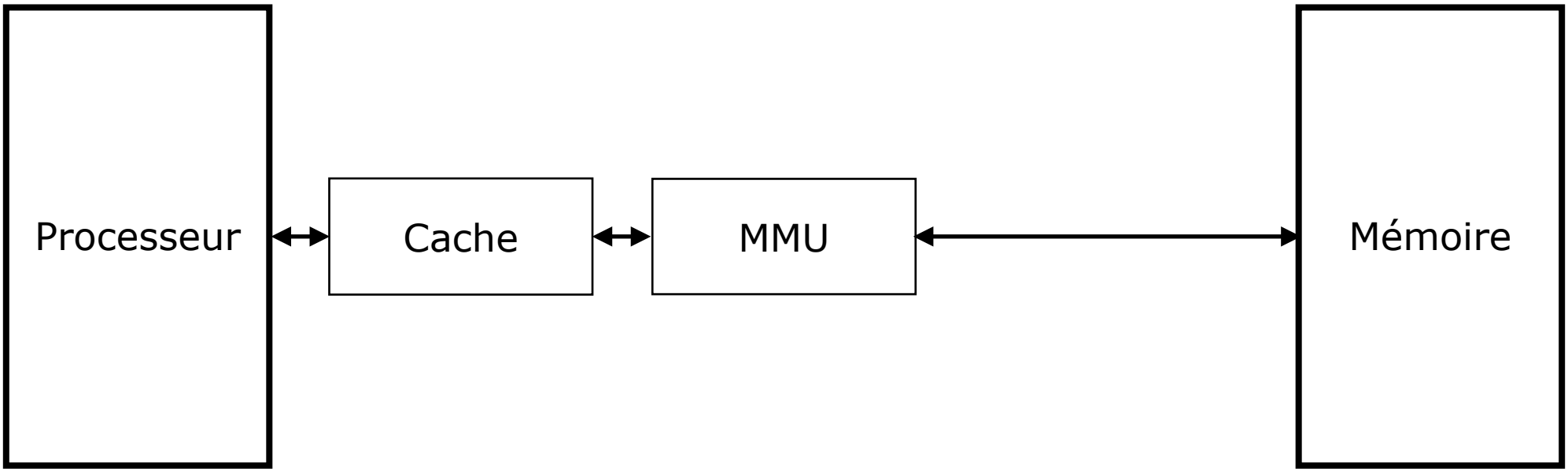
L1 est le cache situé dans le processeur

L2 est le cache externe du processeur

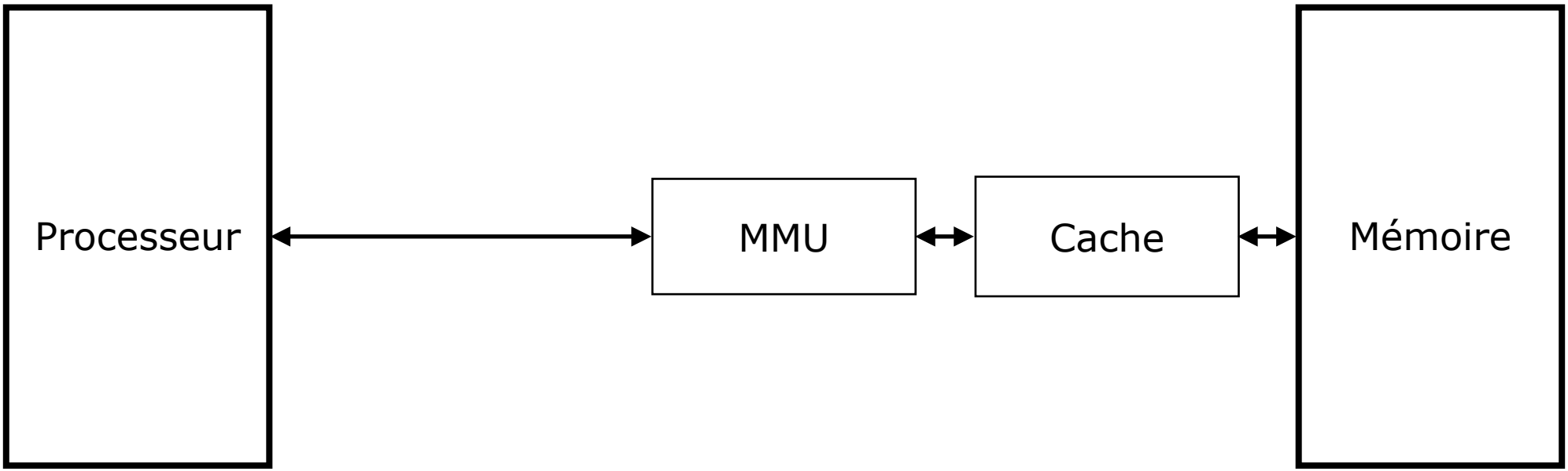
L3 est le cache machine de la mémoire



Cache logique



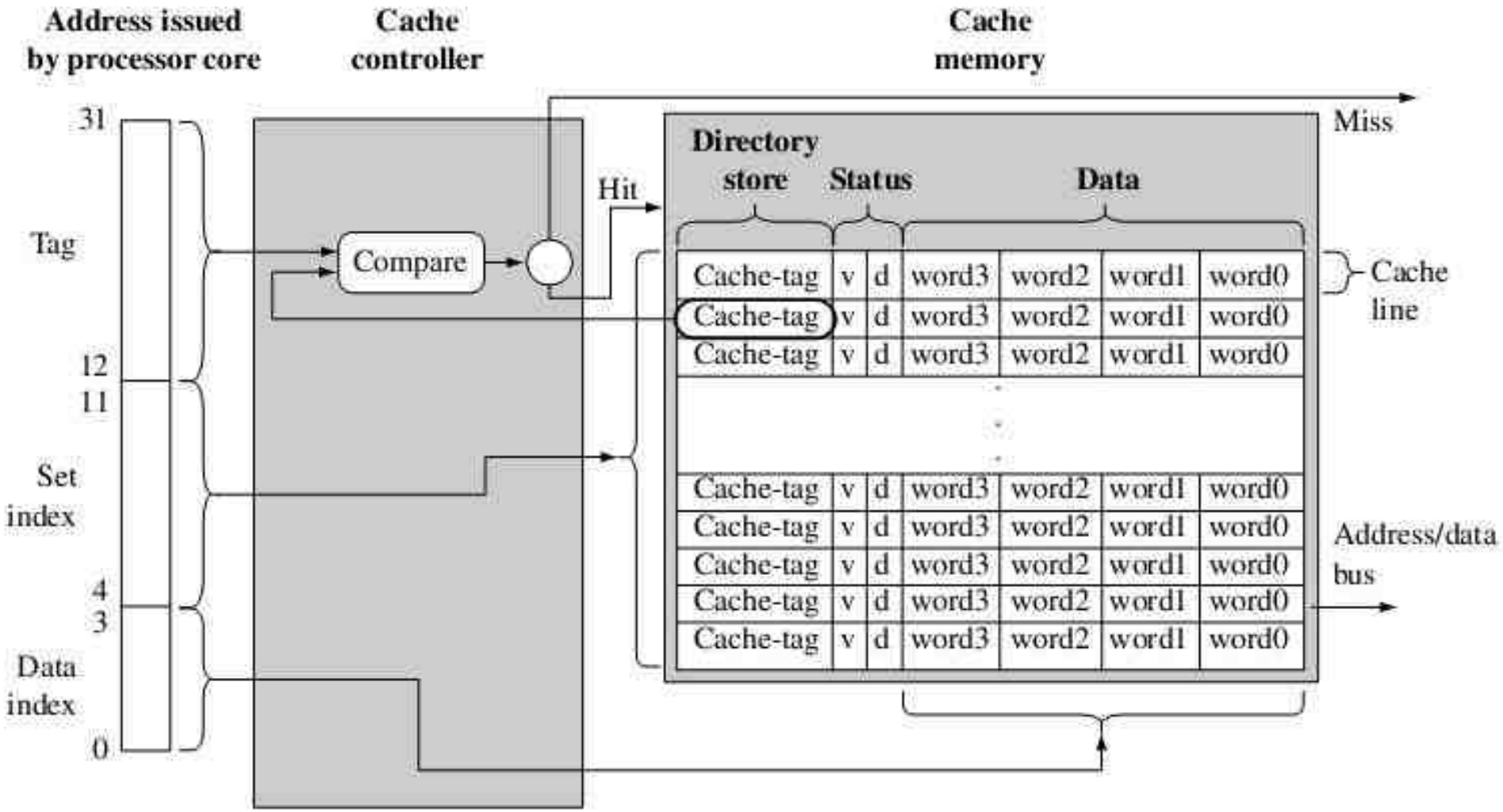
Cache physique



Caches politique de gestion

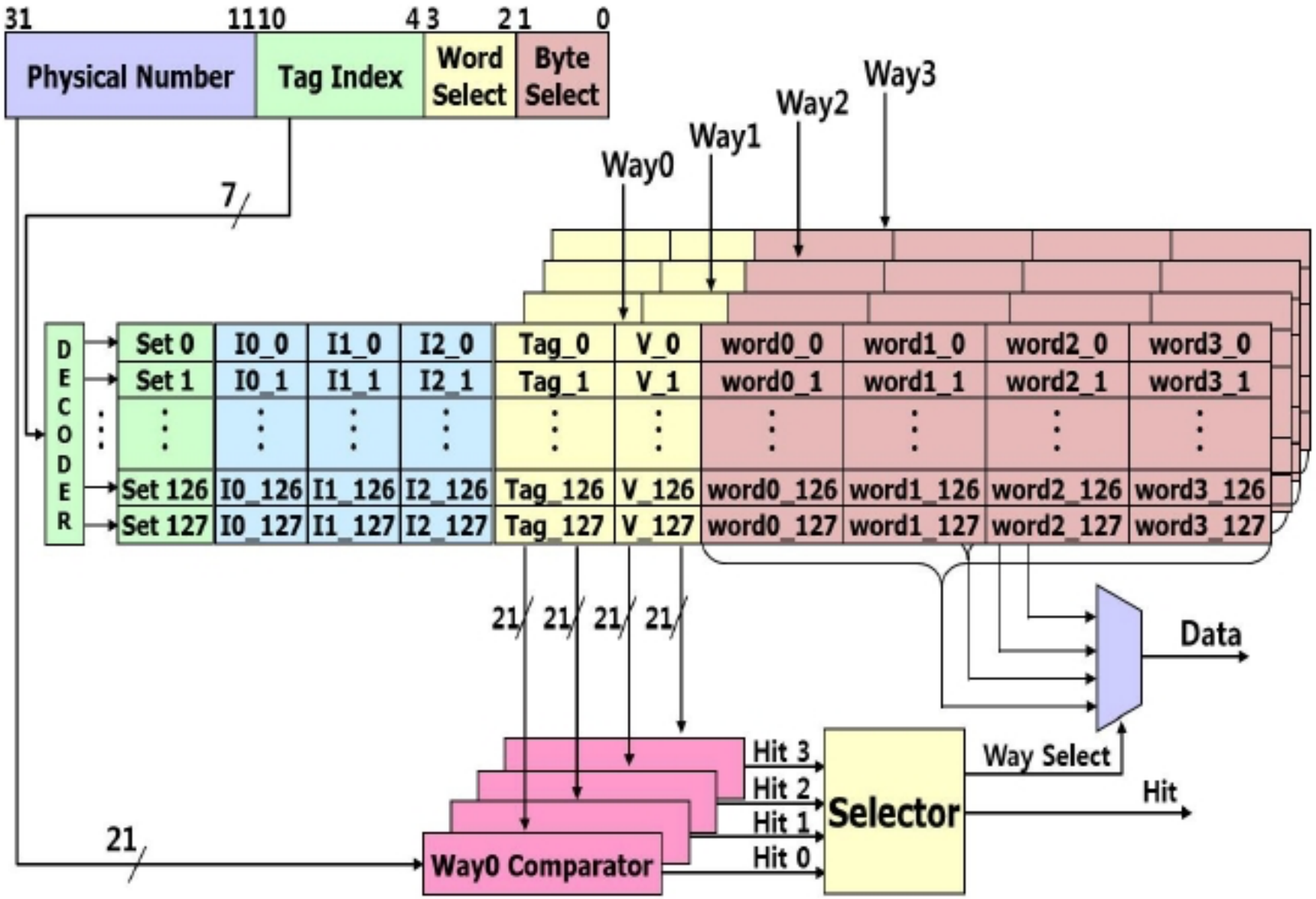
- Writethrough (à travers)
  - les écritures sont immédiatement répertoriées
    - lent…
- Writeback (délayées)
  - écritures effectuées dans le cache mais répercutées sous certaines conditions
    - cohérence…
- Remplacement
  - LRU, Round-Robin, Random

Architecture d'un cache



d = dirty (modifié)
v = valid

Cache Associatif



Contrôle du cache

Opérations

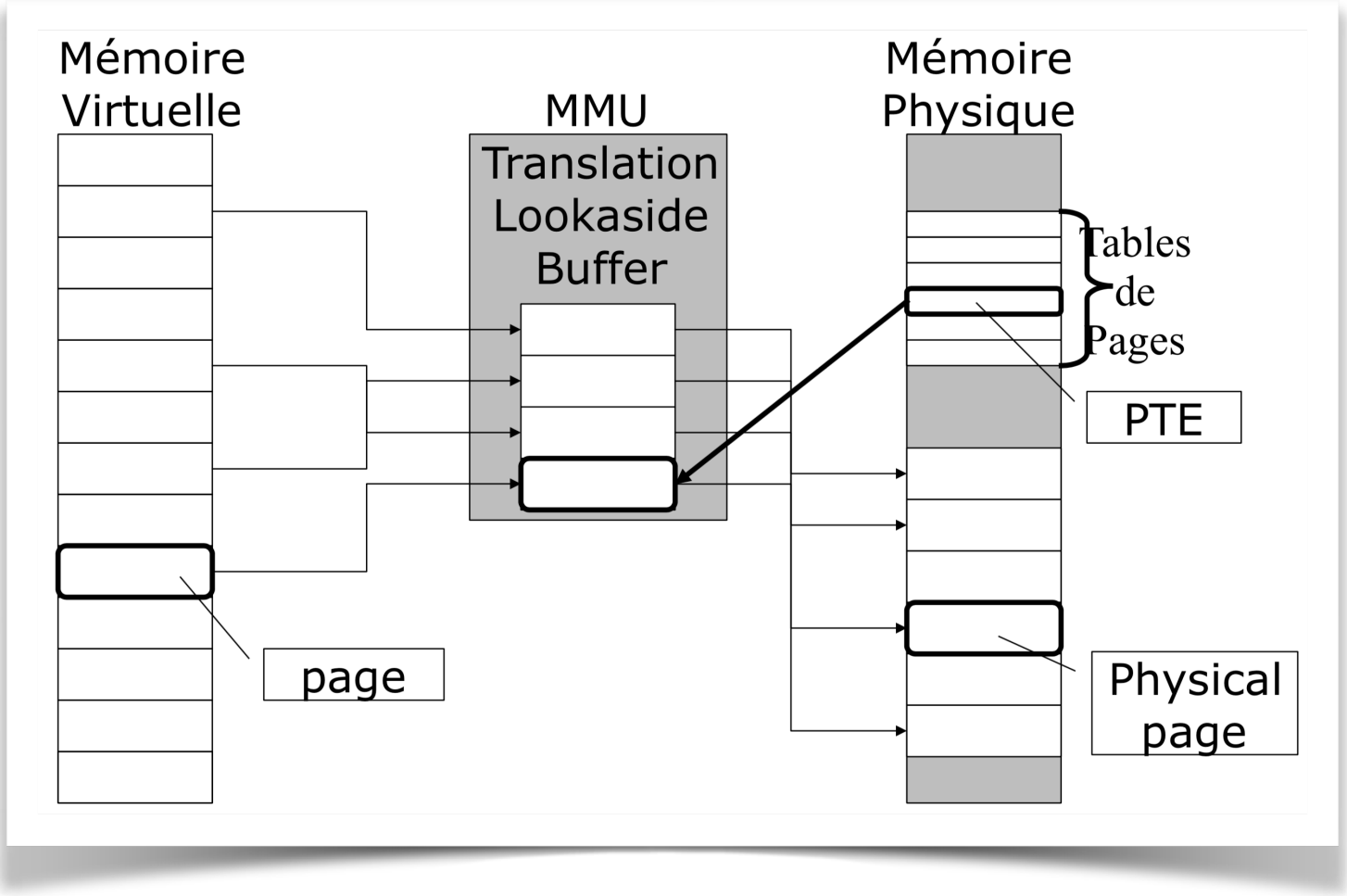
- Flush
  - vider le contenu
- Clean
  - écrire les données modifiées en mémoire

Peuvent être réalisés avec des granularités différentes

Cache code, Cache données…

Retour sur la MMU

TLB ?



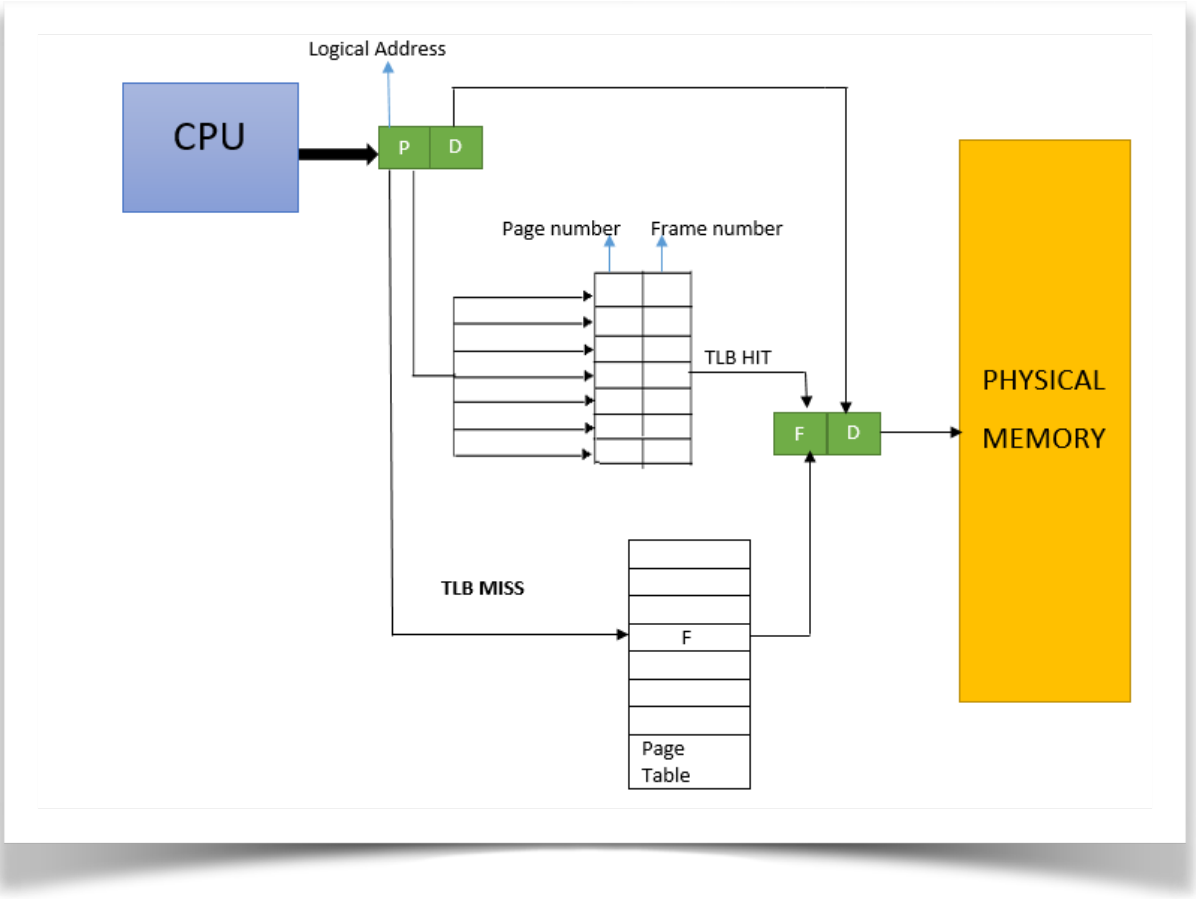
TLB

Translation Lookaside Buffer

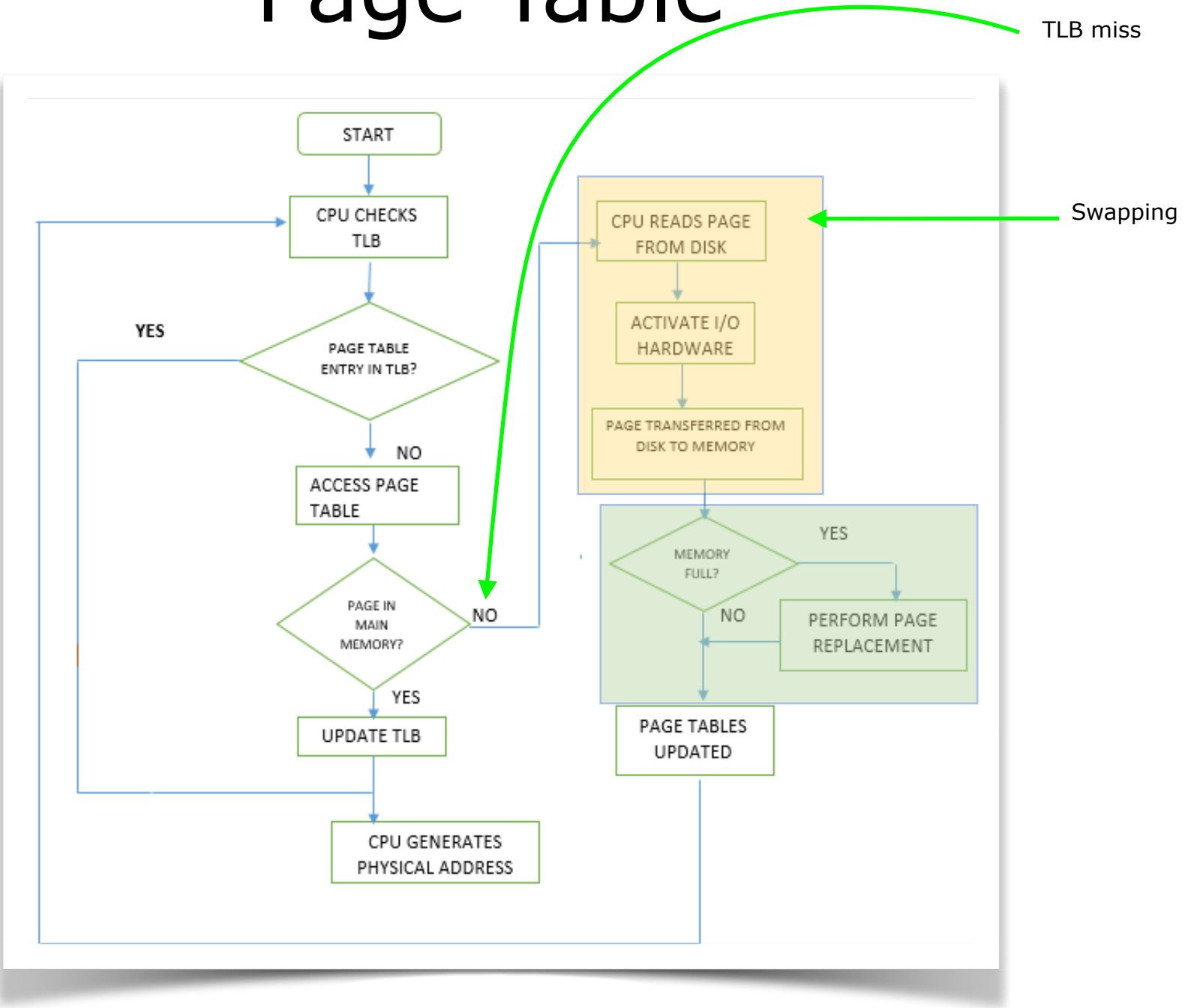
un cache

associatif

Attention : c'est coûteux en matériel



Page Table


TLB miss
Swapping
START
CPU CHECKS TLB
YES
PAGE TABLE ENTRY IN TLB?
NO
ACCESS PAGE TABLE
PAGE IN MAIN MEMORY?
NO
YES
UPDATE TLB
CPU GENERATES PHYSICAL ADDRESS
CPU READS PAGE FROM DISK
ACTIVATE I/O HARDWARE
PAGE TRANSFERRED FROM DISK TO MEMORY
MEMORY FULL?
YES
NO
PERFORM PAGE REPLACEMENT
PAGE TABLES UPDATED

MMU Partage

En mémoire virtuelle le partage de mémoire est (presque) trivialement réalisé

- il suffit qu'une même page physique soit mappée dans deux espaces virtuels différents!
- Unix `mmap / shmat`

Cache en multi-processeur

- Si chacun leur cache (L1, L2)
  - comment gérer la cohérence ?
    - si lecture seule, pas de problème
    - si écriture
      - assurer que la donnée n'est plus présente dans les autres caches
      - cache snooping implémenté en matériel
        - Ex. : ARM SCU (Snoop Control Unit)
        - protocoles : write-invalidate, write-update

Mémoire Intel PAE

- Machines 32 bits
  - limitée à 2^{32} octets d'espace (4Go)
- PAE : Physical Address Extension
  - autorise 2^{36} octets d'espace (64Go)
    - adresses physiques 36 bits
    - adresses virtuelles 32 bits
      - un processus est toujours limité à 4Go mais il y a plus d'espace physique donc moins de collision (trashing)

Mémoire Intel PAE

M_v : 32 bits, M_ϕ : 36 bits

Modification des PTE

- Offset dans page toujours 12 bits
- Adresse de page 24 bits!
- Une entrée de PTE fait 64 bits
  - donc une table ne contient plus que 512 entrée au lieu des 1024… On réduit le tout à 1/4 avec la page directory
    - on rajoute un niveau *Page Directory Pointer* qui contient 4 entrées vers des *Page Directory*, etc

Machines 64 bits

On n'utilise pas encore la totalité de l'espace d'adressage possible

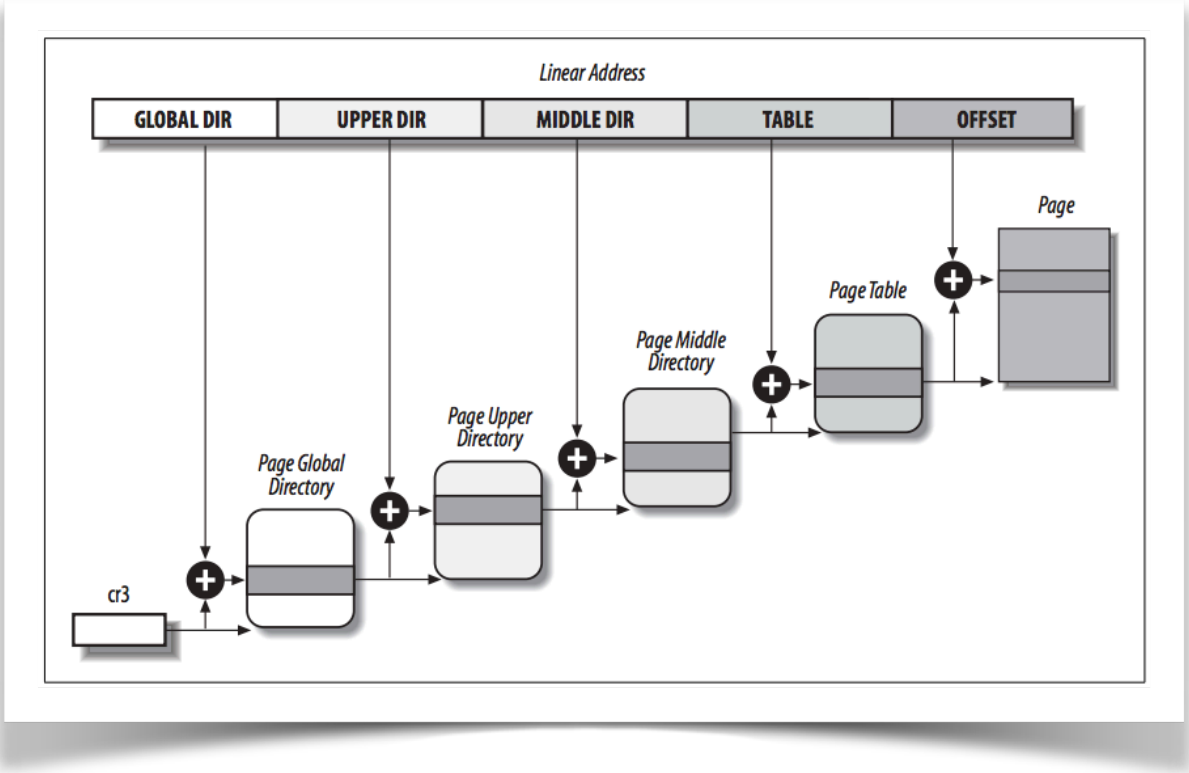
- 1To, adresses sur 40bits
- 256To, adresses sur 48 bits

Nécessité d'augmenter les niveaux d'indirection

Linux 64 bits

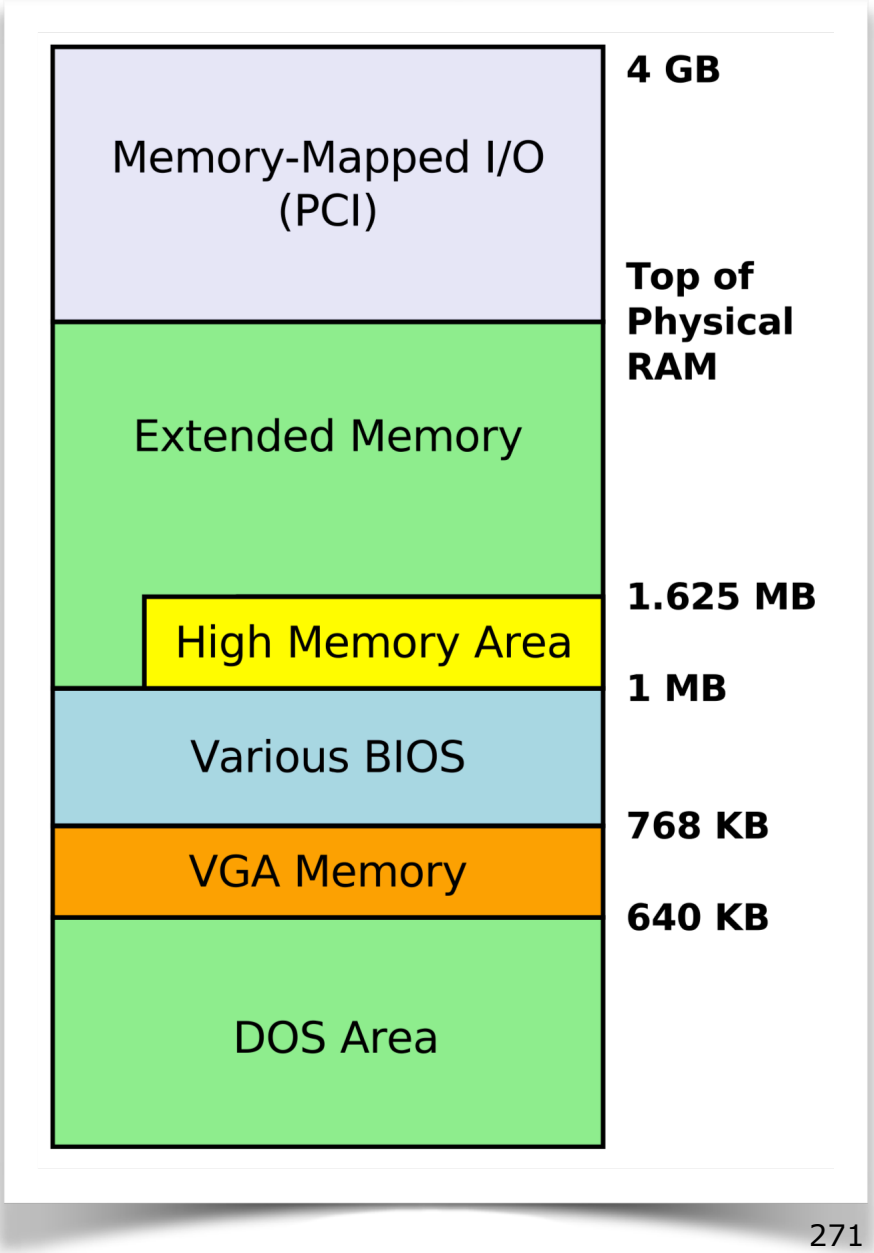
Linux prévoit jusqu'à 4 niveaux d'indirections

en 32 bits, Upper et Middle sont inexistants



x86 Physical Memory

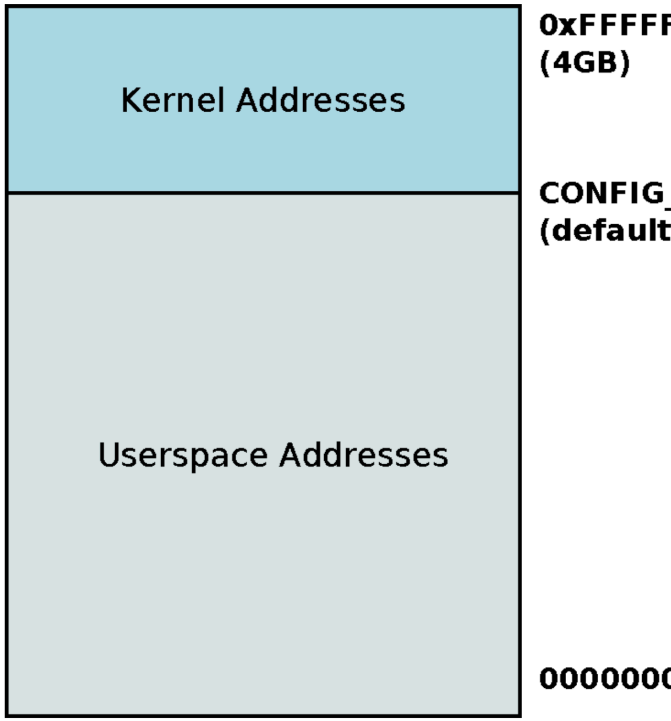
Trop de constructions *ad-hoc*



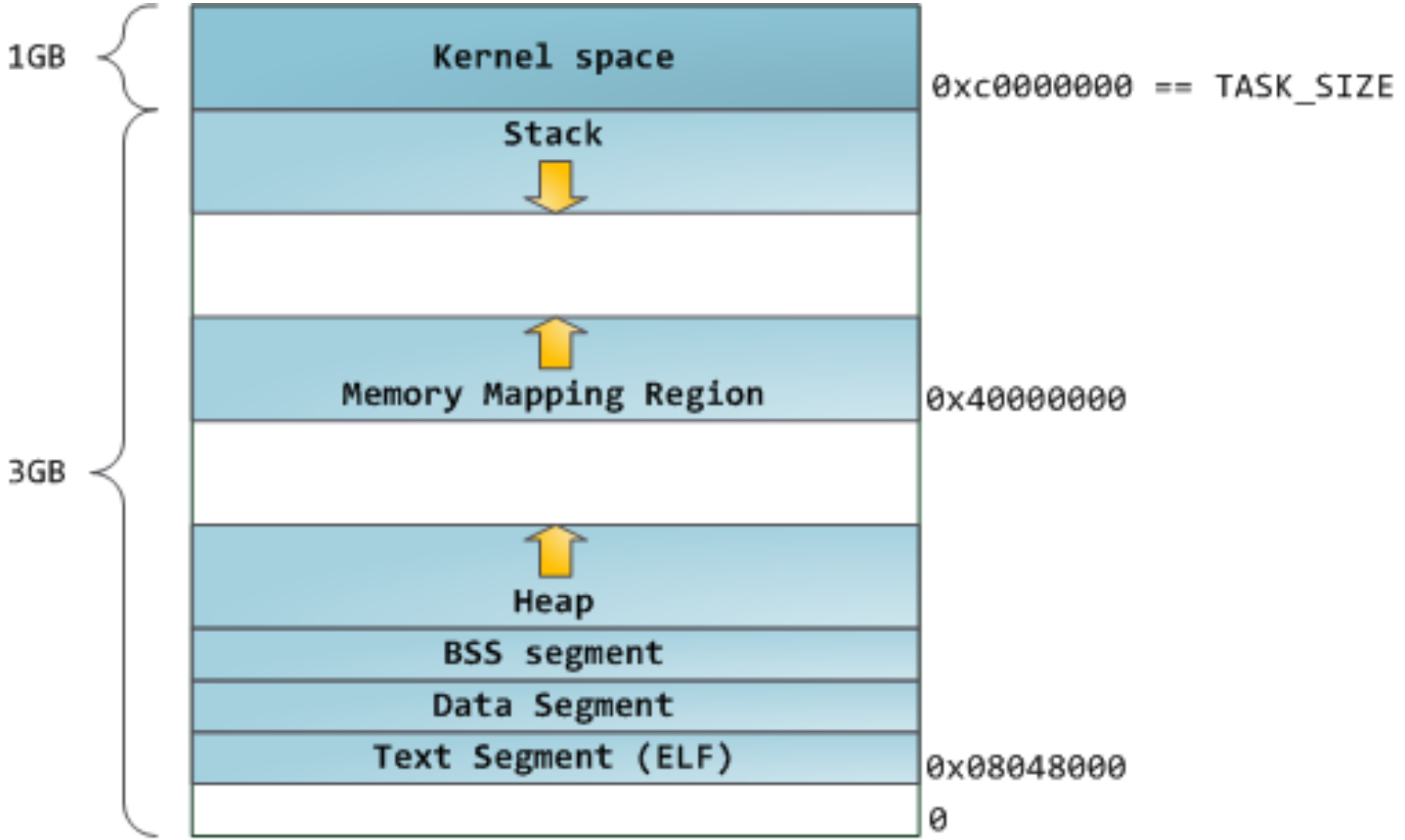
Linux espace virtuel 32bits

Particularité Linux :

- le kernel est en mémoire virtuelle!
- Par défaut 1Go pour le kernel
  - Configurable via `CONFIG_PAGE_OFFSET`, par défaut `CONFIG_VMSPLIT_1G`, etc, à la compilation du noyau



Linux espace virtuel 32bits



Linux espace virtuel 64bits

La coupure est définie par défaut à

- `0x8000000000000000` pour les architectures ARM
- `0xffff880000000000` pour les architectures x86_64

Espaces Linux

Trois espaces sont distingués dans Linux

- noyau
  - espace logique (kernel logical space)
    - espace virtuel avec mapping physique trivial
  - espace virtuel (kernel virtual space)
    - avec pagination
- utilisateur
  - espace virtuel (user virtual space)
    - avec pagination

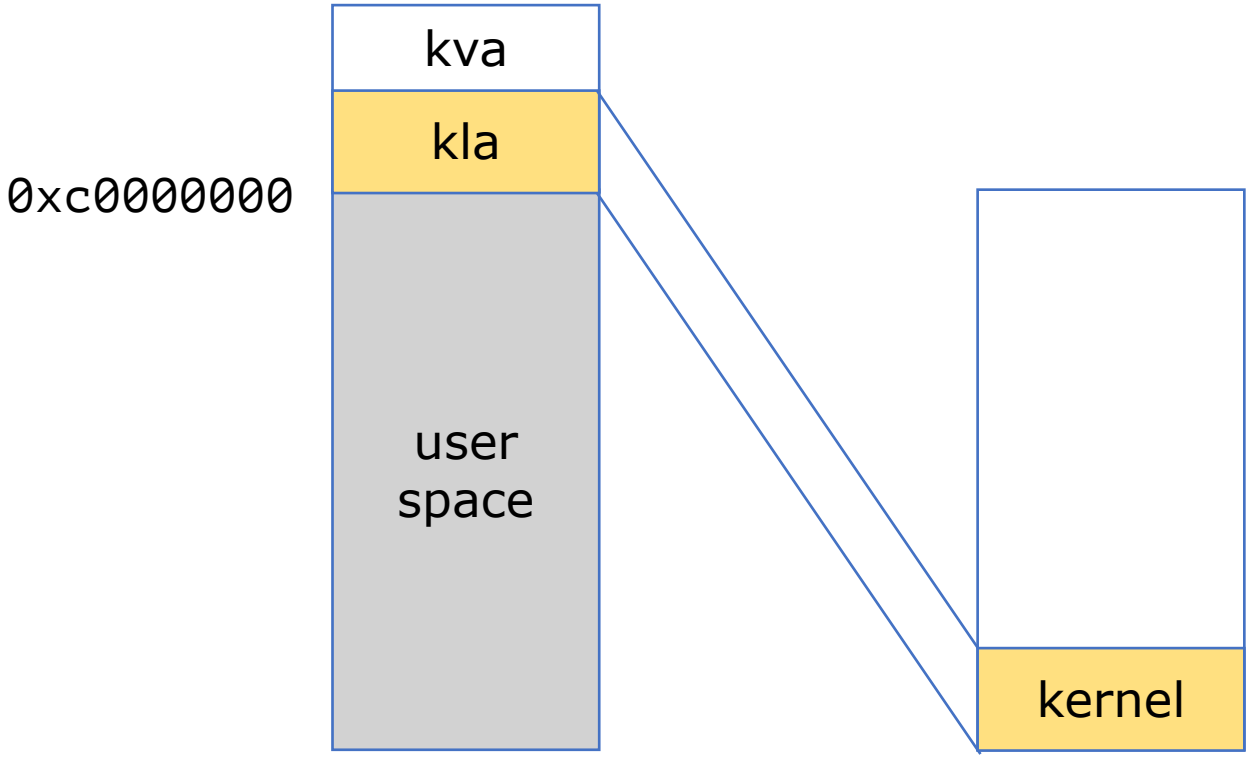
Linux Kernel logical space

- Mappé par translation à l'adresse 0x0 de la mémoire physique
  - donc le noyau ne voit pas toute la mémoire physique dans son logical space
  - les allocations s'y font via `kmalloc()`
- Note : le logical space n'est jamais swappé…
  - parfait pour les DMA…

Linux Kernel virtual space

La mémoire physique au-dessus de l'espace logique noyau est la mémoire noyau virtuelle

vmalloc()



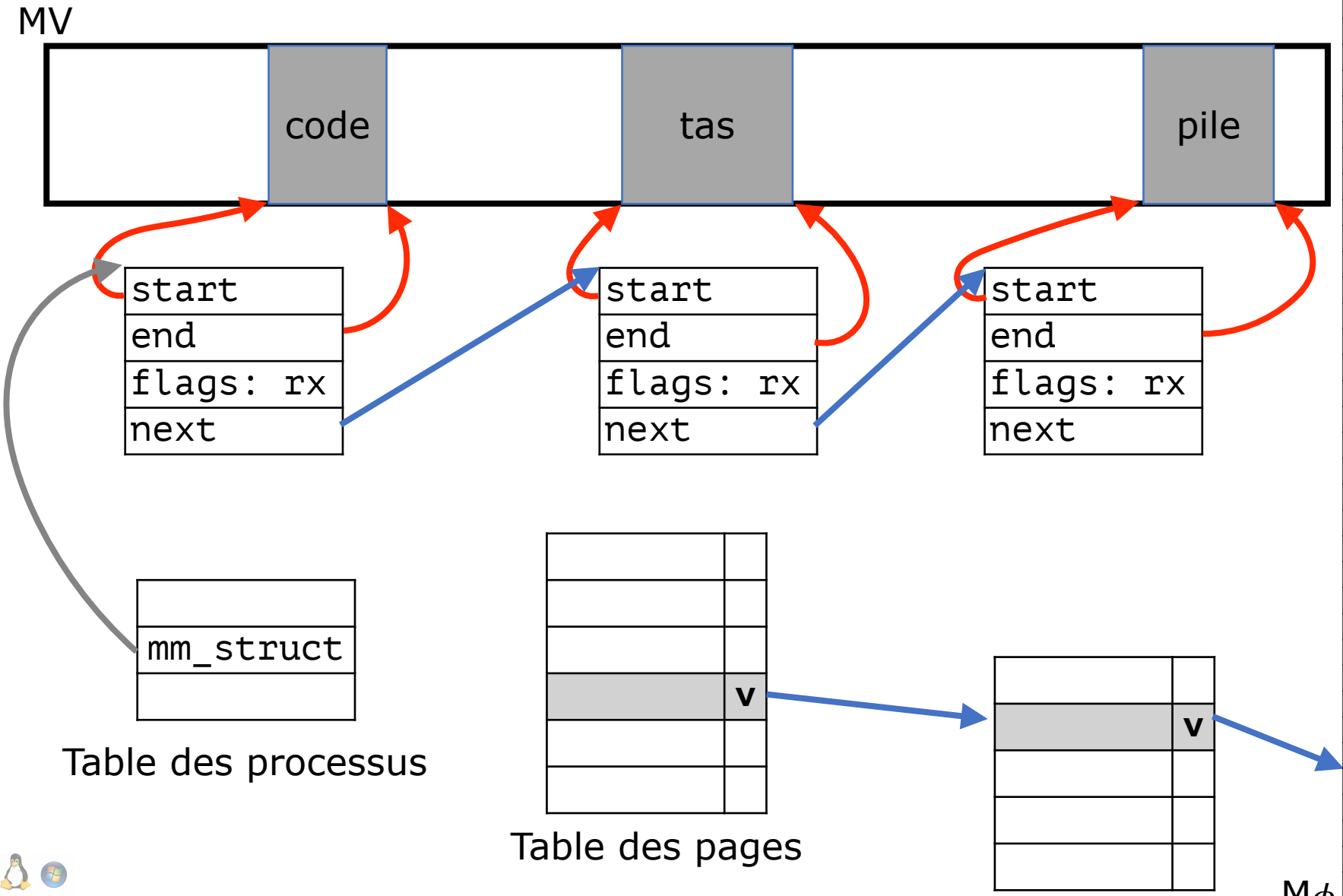
Lazy allocation

- Sur beaucoup de systèmes
  - l'allocation d'une nouvelle page est une allocation virtuelle!
    - l'allocation est réalisée dans l'espace virtuelle mais aucun mapping physique n'est réalisé
      - on défère la véritable allocation au moment où l'on tente d'accéder à la page la première fois
        - ainsi on peut toujours allouer de la mémoire même si l'on en dispose pas assez, pourvu qu'on ne l'utilise pas!

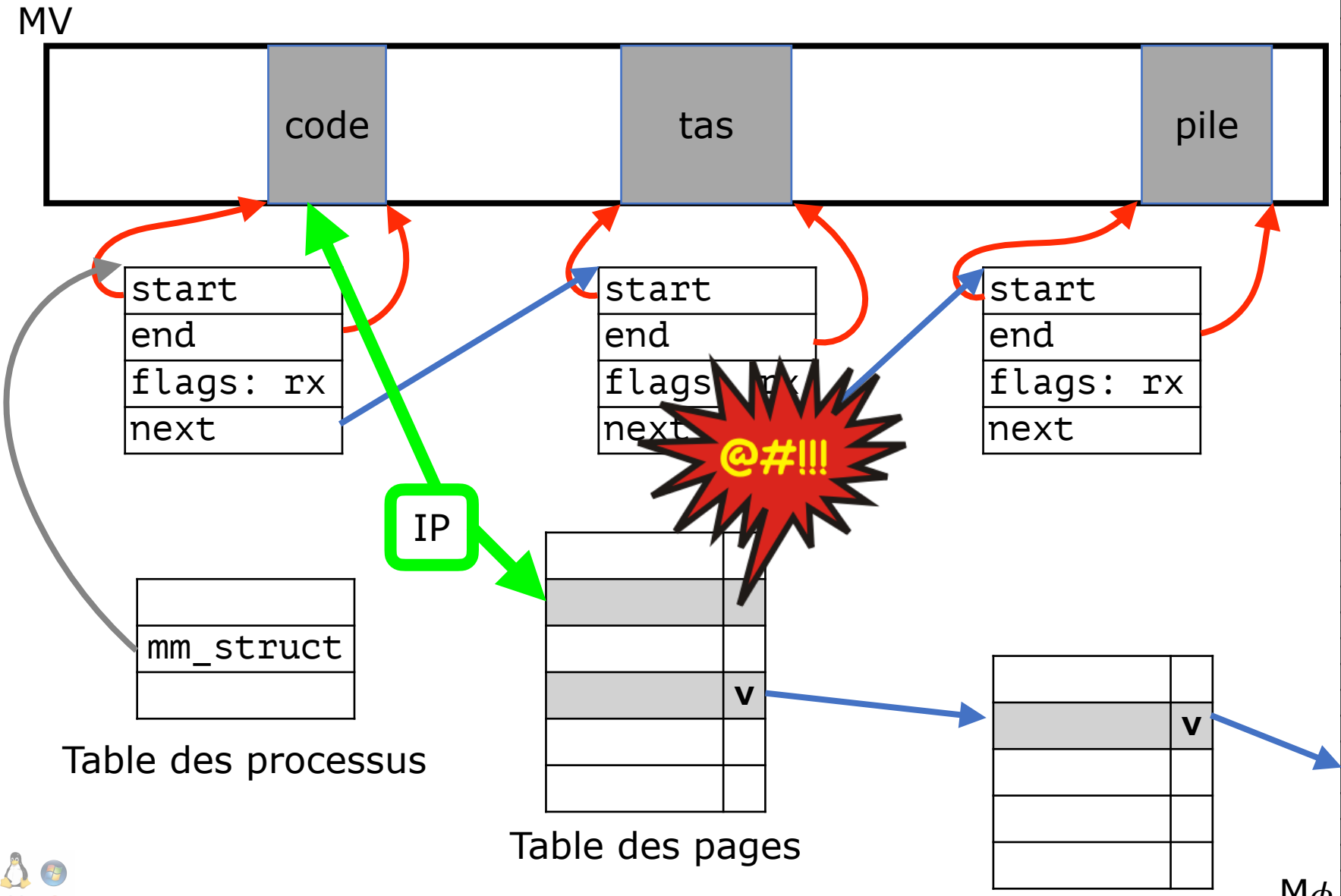
La mémoire virtuelle en pratique

- Nouveau programme «chargé» en mémoire pour exécution
  - Configuration de la MMU, mais pas de pas chargée ne mémoire physique (lazy allocation)
- Quand le processus s'exécute et émet une adresse (IP, Instruction Pointer)
  - Ouille! Pas encore de page physique!
  - Exception…

La mémoire virtuelle en pratique



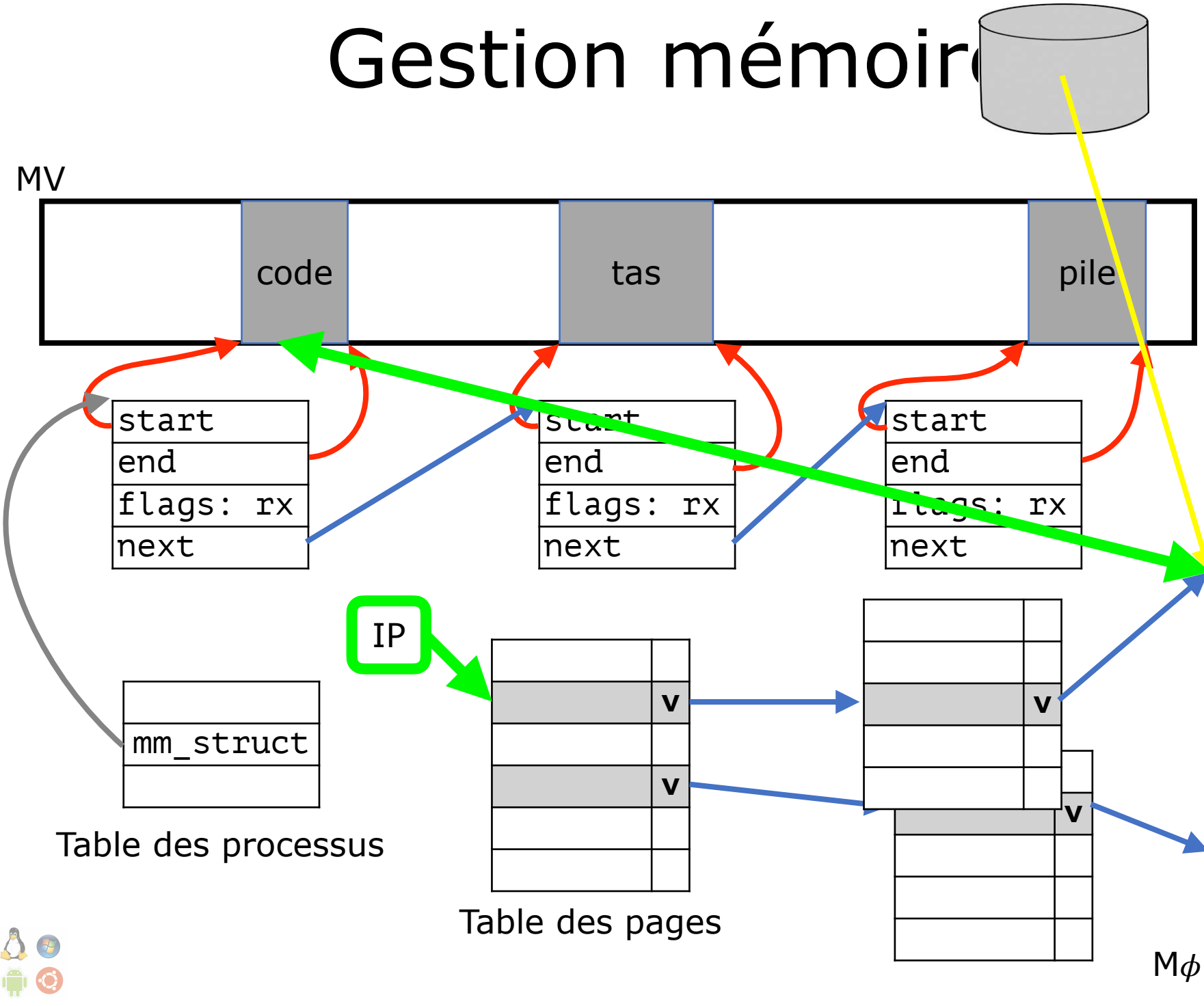
Le défaut de page



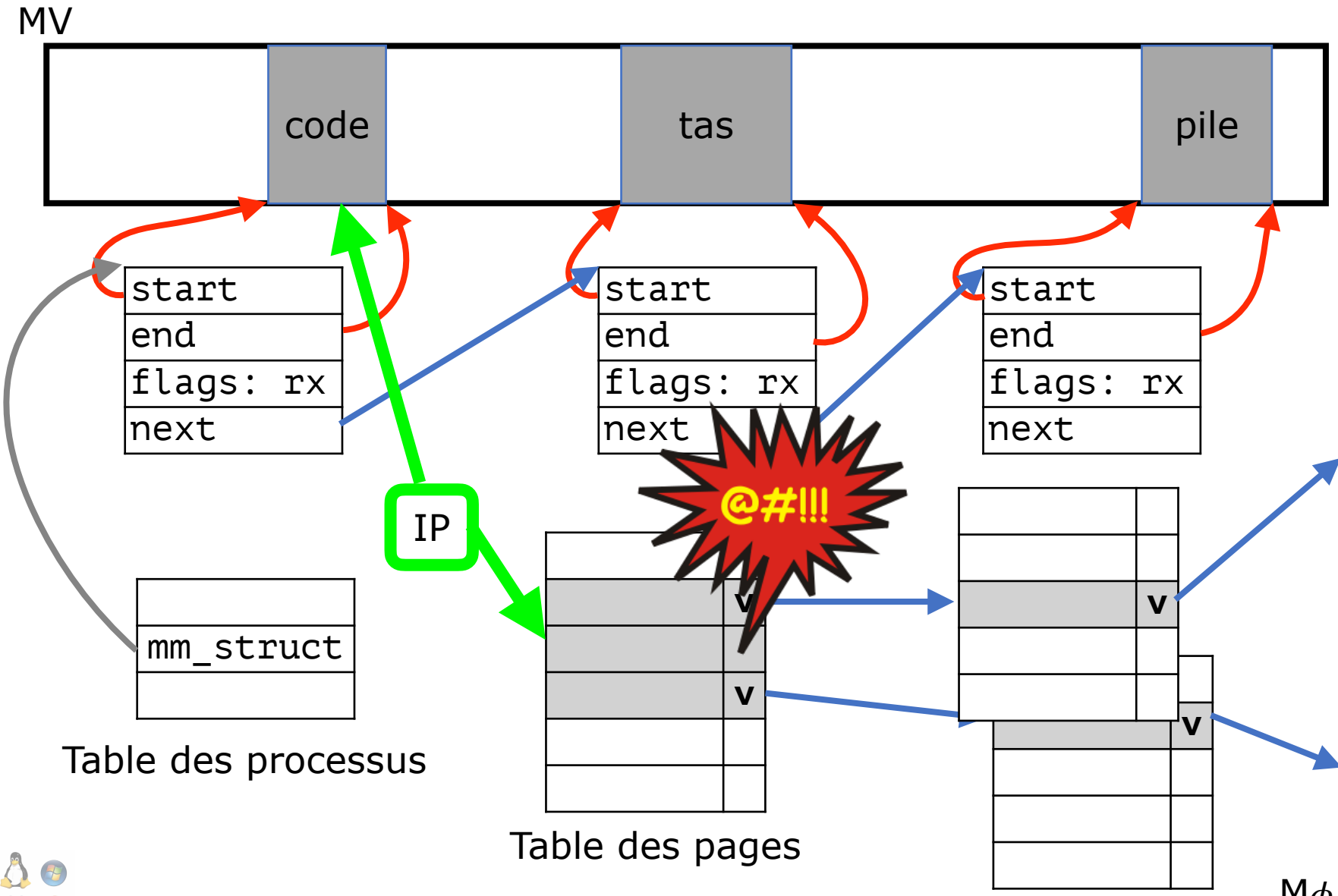
Le défaut de page

- L'OS va traiter l'exception
  - examen de l'adresse
    - si adresse invalide (n'appartient pas à l'espace d'adressage, accès non autorisé)
      - on termine le processus
    - Si adresse valide
      - on retrouve la partie (page) du programme qui correspond
      - on trouve un page de mémoire physique
        - gestion des pages physiques
      - on copie la page disque sur la page mémoire
      - on met à jour les tables
      - on reprend l'exécution de l'instruction fautive

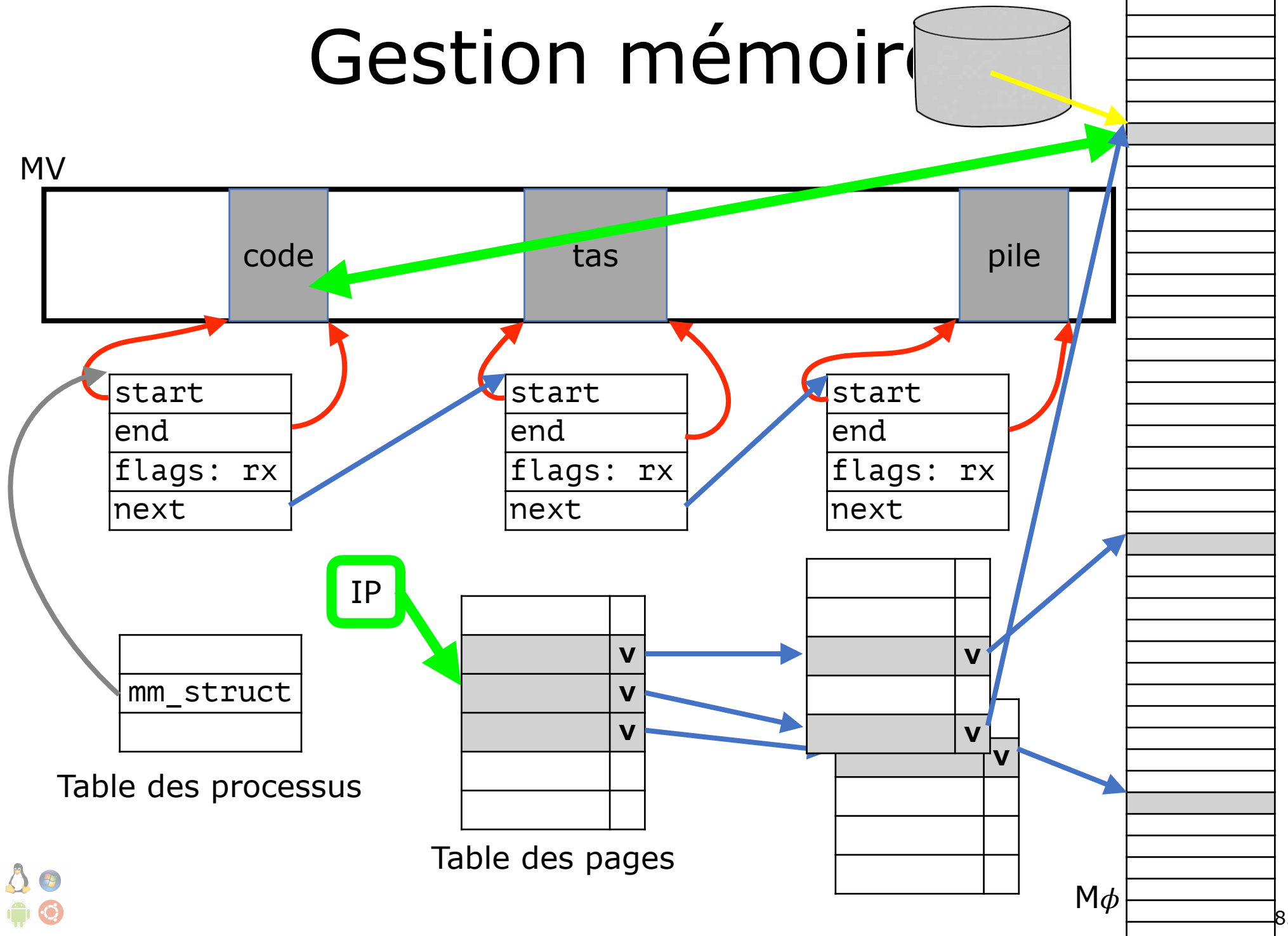
Gestion mémoir



Gestion mémoire



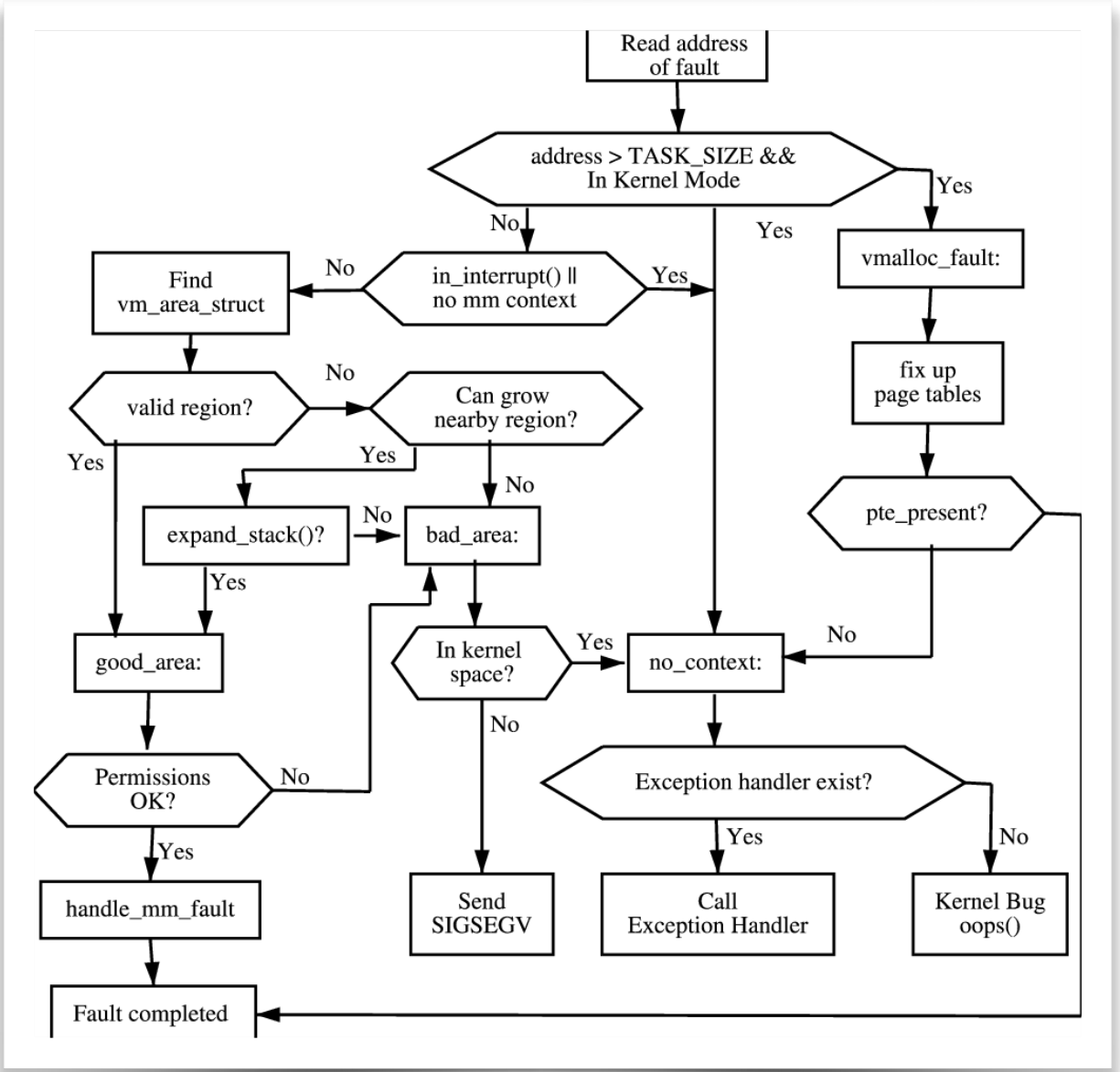
Gestion mémoir



Le défaut de page

- Si l'instruction nécessite une donnée
  - le scénario est le même!
- On continue jusqu'à la terminaison du processus (s'il y en a une!)
  - au cours du temps on remplit donc la table des pages du processus
    - par du code
    - par des données (pile, tas)

Page fault dans Linux



`do_pagefault()` stub de `__do_page_fault()` Linux 4.19 pour x86

Page fault dans Linux

`do_pagefault()` stub de `__do_page_fault()`
Linux 4.19 dans `fault.c`
- `fault.c` est architecture-dépendant évidemment…
  - Linux 4.19 :
    - alpha, arc, arm, arm64, c6x, h8300, hexagon, ia64, m68k, microblaze, mips, nds32, nios2, openrisc, parisc, powerpc, riscv, s390, sparc, um, unicore32, x86, xtensa

Page stealing

Mais que se passe t-il s'il n'y a plus de page physique disponible ?

- page stealing
- on reprend une page physique (à un autre processus ou au même)
  - mise à jour des deux tables concernées

Page stealing

- quid du contenu de la page volée ?
  - on l'écrit sur disque!
    - emplacement réservé sur le disque pour stocker les pages évincées
      - inutile pour les page de code!
      - swap file…
    - modification des entrées de la table des pages en conséquence (page lazy, page mappée, page évincée, localisation de la page)

Page thrashing

A sort une page de B

B sort une page de A

A sort une page de B

à chaque fois la même page en jeu…

- le résultat est désastreux car gérer un défaut de page est très coûteux
  - lire en mémoire est de l'ordre de 10 ns (DDR4)
  - lire sur disque est de l'ordre de 5-10 ms, sur SSD c'est 100μs
    - bref, facteur 10.000 ou 1.000.000!!!!

Page thrashing

- Lutte
  - empêcher certaines pages de sortir peut-être utile
    - concept de working set
      - ensemble de page physiques dédiées à un processus
  - utiliser le principe de localité temporelle
    - LRU

Contraintes

Éviter d'éjecter les pages partagées

Ne pas éjecter les pages sur lesquelles une I/O est en cours

Ne pas éjecter les pages verrouillées

LRU

Least Recently Used

- utiliser un tampon horodateur pour marquer les pages fréquemment accédées et n'évincer que les plus vieilles
  - trop coûteux en pratique!
  - s'approxime par le NRU

NRU

Not Recently Used

- on accède à une page, on la marque (1 bit)
- on écrit dans une page, on la marque (1 bit)
- à intervalle régulier, on efface la marque d'accès de toutes les pages
  - 4 états possibles
    - non accédée, non modifiée
    - non accédée, modifiée
    - accédée, non modifiée
    - accédée, modifiée

NRU

On évince donc les non accédée, non modifiée en priorité, etc

- N’évite pas le thrashing mais en pratique le limite très sérieusement
- Coût assez faible

Page replacement

Linux 4.18

- LRU

BSD

- Second Chance (CLOCK), un NRU incrémental

FreeBsd

FreeBSD distingue 5 type de pages

- Wired
  - page qui ne peuvent être évincées (kernel, locked)
- Active
  - Un équivalent du Working Set, peuvent être évincées mais le risque de thrashing augmente
- Inactive
  - Ne servent plus depuis un moment, peuvent donc être choisies
- Cache
  - Peuvent être évincées sans avoir à les sauvegarder
- Free
  - Le graal

macOS

`vm_stat`

statistiques de mémoire virtuelle

`vmmap`

détails sur la mémoire virtuelle d'un processus

macOS

```
Mach Virtual Memory Statistics: (page size of 4096 bytes)
Pages free:                                57358.
Pages active:                            1604778.
Pages inactive:                          1429509.
Pages speculative:                         77218.
Pages throttled:                               0.
Pages wired down:                        771795.
Pages purgeable:                         236044.
"Translation faults":                  409036052.
Pages copy-on-write:                    15382200.
Pages zero filled:                     256927522.
Pages reactivated:                       549667.
Pages purged:                           4890641.
File-backed pages:                       534292.
Anonymous pages:                        2577213.
Pages stored in compressor:              660138.
Pages occupied by compressor:            253072.
Decompressions:                         8039312.
Compressions:                          10970961.
Pageins:                               13651051.
Pageouts:                                 22355.
Swapins:                                9455713.
Swapouts:                               9846324.
```

Gestion avancée

POSIX realtime extensions prévoit de pouvoir donner au système des indications sur la façon dont on va utiliser certaines plages mémoire

- de la sorte le système peut utiliser des algorithmes différents pour gérer les pages correspondantes

madvise

SYNOPSIS

```
[ADV] ⌦ #include <sys/mman.h>

int posix_madvise(void *addr, size_t len, int advice);
```

POSIX_MADV_NORMAL on ne sait pas

POSIX_MADV_SEQUENTIAL accès séquentiels

POSIX_MADV_RANDOM accès aléatoires

POSIX_MADV_WILLNEED besoin très bientôt

POSIX_MADV_DONTNEED pas besoin avant un moment

madvise

est supporté de façon stricte par macOS

Linux propose des options supplémentaires

- `MADV_REMOVE` (2.6)
  - pour recréer un «trou» dans l'espace
- `MADV_DONTFORK` / `MADV_DOFORK` (2.6)
- `MADV_HWPOISON` (2.6)
  - pour forcer la génération de fautes (certains kernels)
- `MADV_MERGEABLE` / `MADV_UNMERGEABLE` (2.6)
  - instance unique pour pages de même contenu…
- `MADV_SOFT_OFFLINE` (2.6)
  - certains kernels, pour déboguer, modifie l'association physique d'une page

madvise

Linux propose des options supplémentaires

- `MADV_HUGEPAGE` / `MADV_NOHUGEPAGE` (2.6)
  - pour demander l'utilisation de «grosses pages»
- `MADV_DONTDUMP` / `MADV_DODUMP` (3.4)
  - ignorer ces pages lors des core dumps
- `MADV_FREE` (4.5)
  - pour libérer des pages
- `MADV_WIPEONFORK` / `MADV_KEEPONFORK` (4.14)
  - permet d'effacer les pages héritées (pour des raisons de sécurité)

Gestion de processus

Processus

Threads

Processus

C’est quoi ?

une abstraction du système de type VM!

Idée

permettre de raisonner comme si le programme était seul sur la machine

C’est un conteneur

données, programme, exécution

Il évolue dans un contexte

Processus

Le programmeur est affranchi de détails sordides!

- il ne se demande pas
  - combien de mémoire physique est disponible, si tout va tenir
  - où placer les données, le programme, etc.
  - comment on écrit sur le disque, etc.

Les choses sont indépendantes de la réalité (ou presque)

Processus

- Machine Virtuelle
  - s'exécute «seul»
    - processeur pour lui
    - mémoire pour lui
  - instructions
    - mélange instructions machine et services système
- Contexte d'exécution

Contexte

Ensemble des informations détenues par le système sur le processus et qui lui permettent de le manipuler à sa guise

- assurer la cohabitation de plusieurs processus
- assurer l'exécution contextuelle de services (droits)
- assurer la bonne communication
- traçage / comptabilité

Gestion de processus

- Création d'un processus
  - Unix, par clonage *via* `fork()`
  - POSIX, par naissance spontanée *via* `posix_spawn()`
- La vision Unixienne est que les processus ont une génétique
  - l'arborescence des processus

Gestion de processus

L'arborescence des processus

`pstree`

```
parallels@ubuntu:~$ pstree
systemd─┬─ModemManager─┬─{gdbus}
        │              └─{gmain}
        ├─NetworkManager─┬─dhclient
        │                ├─dnsmasq
        │                ├─{gdbus}
        │                └─{gmain}
        ├─accounts-daemon─┬─{gdbus}
        │                 └─{gmain}
        ├─acpid
        ├─agetty
        ├─avahi-daemon───avahi-daemon
        ├─colord─┬─{gdbus}
        │        └─{gmain}
        ├─cron
        ├─cups-browsed─┬─{gdbus}
        │              └─{gmain}
        ├─cupsd───3*[dbus]
        ├─dbus-daemon
        ├─gnome-keyring-d───{gmain}
        ├─gnome-keyring-d─┬─{gdbus}
        │                 ├─{gmain}
        │                 └─{timer}
        ├─irqbalance
        ├─lightdm─┬─Xorg───{Xorg}
        │         └─lightdm─┬─upstart─┬─at-spi-bus-laun─┬─dbus-daemon
        │                   │         │                 ├─{dconf worker}
        │                   │         │                 ├─{gdbus}
        │                   │         │                 └─{gmain}
        │                   │         ├─at-spi2-registr─┬─{gdbus}
        │                   │         │                 └─{gmain}
        │                   │         ├─bamfdaemon─┬─{dconf worker}
        │                   │         │            ├─{gdbus}
        │                   │         │            └─{gmain}
        │                   │         ├─compiz─┬─{dconf worker}
        │                   │         │        ├─{gdbus}
        │                   │         │        ├─{gmain}
        │                   │         │        └─{pool}
        │                   │         ├─dbus-daemon───dbus-daemon───gvfs-udisks2-vo─┬─{gdbus}
        │                   │         │                                            └─{gmain}
        │                   │         ├─dconf-service─┬─{gdbus}
        │                   │         │               └─{gmain}
        │                   │         ├─evolution-addre─┬─evolution-addre─┬─{dconf worker}
        │                   │         │                 │                 └─{evolution-addre}
```

Gestion de processus

L'arborescence des processus

- Les processus sont tous issus d'un processus originel créé au démarrage
  - historiquement `init` issu de SystemV
  - récemment `systemd` pour pallier aux défauts supposés d'`init`
- Les processus sont reliés les uns aux autres
  - Un processus est relié à son processus parent
  - Un processus possède le moyen de retenir l'identité des processus qu'il créé

Création de processus

```c
#include <stdio.h>
#include <unistd.h>
#include <stdint.h>
#include <stdlib.h>

int main() {
  printf("Bonjour je suis %jd\n",(intmax_t)getpid());
  pid_t son = fork();
  switch (son) {
  case -1:
    fprintf(stderr,"Problème\n");
    exit(EXIT_FAILURE);
  case 0:
    printf("Je suis %jd et mon parent est %jd\n",
           (intmax_t)getpid(),(intmax_t)getppid());
    exit(EXIT_SUCCESS);
  default:
    printf("Je suis %jd et mon enfant est %jd\n",
           (intmax_t)getpid(),(intmax_t)son);
    wait(NULL);
    exit(EXIT_SUCCESS);
  }
}
```

Identités

Les processus Unix sont identifiés

`getpid()`

identité de l'appelant

`getppid()`

identité du parent

Identités

sched.h (4.19)

```
struct task_struct {
…
    pid_t           tgid;
    struct task_struct *real_parent;
…
```

sys.c (4.19)

```
SYSCALL_DEFINE0(getpid)
{
        return task_tgid_vnr(current);
}

SYSCALL_DEFINE0(getppid)
{
…
        pid = task_tgid_vnr(rcu_dereference(current->real_parent));
…
        return pid;
}
```

Propriété

Les processus Unix sont la propriété d'utilisateurs

Est donc attachée à chaque processus des informations de propriétés qui permettent de déterminer

- qui possède le processus
- si le processus a le droit d'accéder à une ressource
- si le processus a le droit d'effectuer une opération

Une propriété est un couple

- (propriétaire/groupe)

Propriété

La propriété est fixée au démarrage d'une session

par l'intermédiaire du couple login/password

```
[Estafette:~] id
uid=501(yunes) gid=20(staff) groups=20(staff),505(access_bpf),12(everyone),61(l
ocalaccounts),79(_appserverusr),80(admin),81(_appserveradm),98(_lpadmin),701(co
m.apple.sharepoint.group.1),705(com.apple.sharepoint.group.5),702(com.apple.sha
repoint.group.2),704(com.apple.sharepoint.group.4),33(_appstore),100(_lpoperato
r),204(_developer),250(_analyticsusers),395(com.apple.access_ftp),103(com.apple
.access_screensharing-disabled),399(com.apple.access_ssh),706(com.apple.sharepo
int.group.6)
[Estafette:~]
```

et est héritée…

Propriété

- Deux identités propriétaires!
  - propriétaire réel (real user-id)
    - le véritable propriétaire, c'est-à-dire celui qui possède le processus
  - propriétaire effectif (effective user-id)
    - l'identité utilisée pour vérifier les droits
- En général les deux identités sont identiques
  - mais il est possible d'utiliser temporairement l'identité de quelqu'un d'autre (sous réserve de pouvoir le faire!)

Propriété

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int main() {
  printf("ruid: %d rgid: %d euid: %d egid: %d\n",
         getuid(),getgid(),geteuid(),getegid());
  exit(0);
}
```

```
[Estafette:~/code/ids] ./ids
ruid: 501 rgid: 20 euid: 501 egid: 20
```

Propriété

Le même code exécuté avec

```
-rwsr-sr-x  1 root   wheel  8664  1 nov 10:38 ids2
```

```
[Estafette:~/code/ids] ./ids2
ruid: 501 rgid: 20 euid: 0 egid: 0
```

permet d'obtenir les droits de root/wheel

mais c'est root/wheel qui l'a décidé

c'est à lui de d'autoriser cela avec parcimonie

Propriété

sched.h (4.19)

```
struct task_struct {
…
    const struct cred       *cred;
…
```

cred.h (4.19)

```
struct cred {
…
    kuid_t      uid;        /* real UID of the task */
    kgid_t      gid;        /* real GID of the task */
    kuid_t      euid;       /* effective UID of the task */
    kgid_t      egid;       /* effective GID of the task */
…
```

Création de processus

`fork()`

- un appel, deux retours!
  - un appel depuis un processus
  - clonage par le système dans l'appel
  - deux processus sortent de l'appel
- on distingue l'appelant du nouveau créé par le fait que
  - l'appelant revient de l'appel avec l'identité du nouveau
  - le nouveau revient de l'appel avec la valeur 0 (pour indiquer que c'est lui le nouveau né)

Création de processus

Les deux processus sont à peu près indistinguables… Clonage

Pour cloner un processus, le système clone

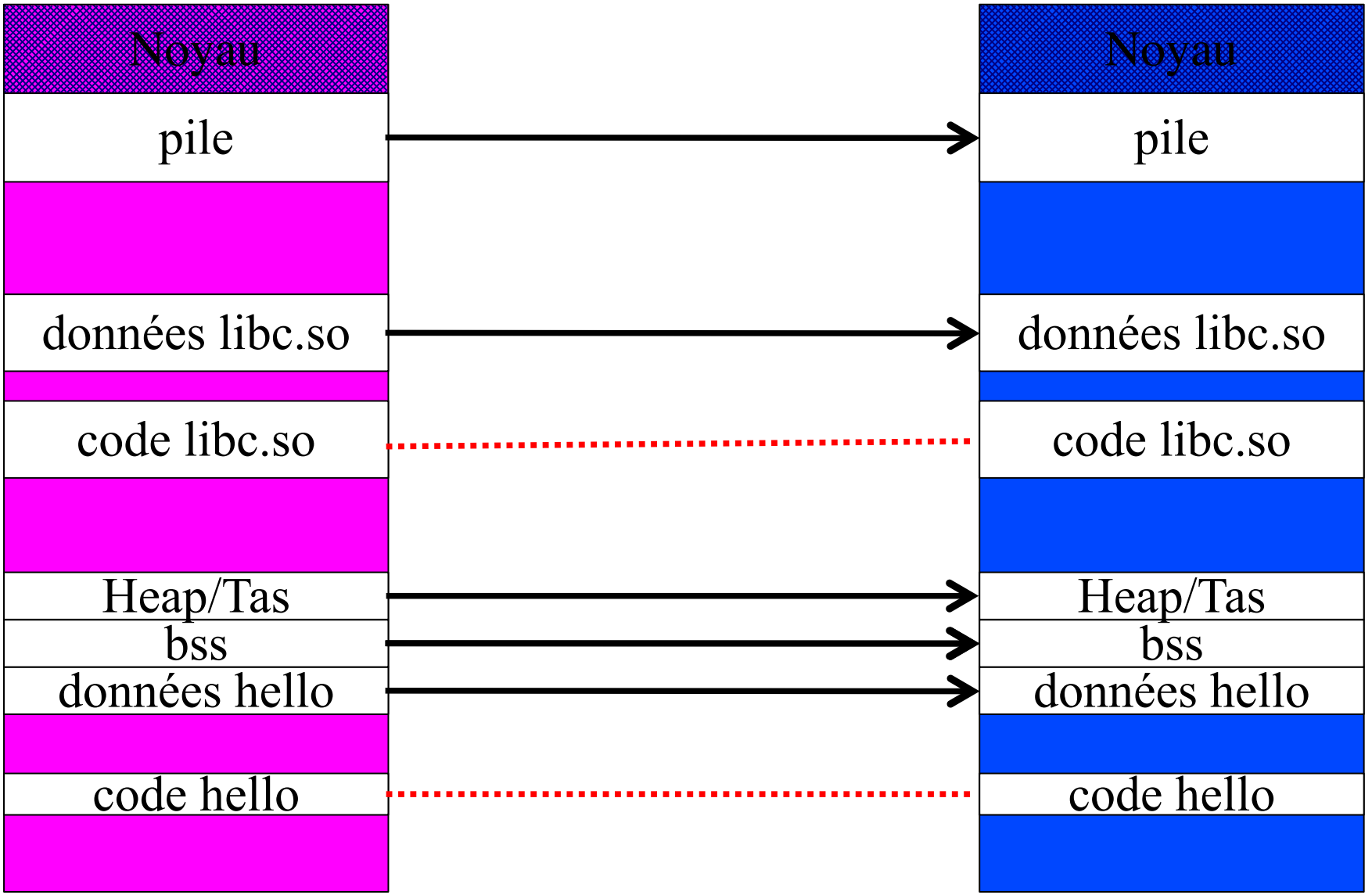
- le contexte (en modifiant de quoi renvoyer une valeur différente)
- l'entrée de la table des processus
- l'espace d'adressage

Linux clone

`clone()` est une fonction Linux qui réalise le clonage de façon fine

```
CLONE(2)                    Linux Programmer's Manual                    CLONE(2)

NAME       top

       clone, __clone2 - create a child process

SYNOPSIS       top

       /* Prototype for the glibc wrapper function */

       #define _GNU_SOURCE
       #include <sched.h>

       int clone(int (*fn)(void *), void *child_stack,
                 int flags, void *arg, ...
                 /* pid_t *ptid, void *newtls, pid_t *ctid */ );

       /* For the prototype of the raw system call, see NOTES */

DESCRIPTION       top

       clone() creates a new process, in a manner similar to fork(2).
```

Création de processus



Création de processus

Le nouveau processus est une copie quasiment stricte

- même adresse pour le code, données, tas, pile…
  - on peut peut-être partager ce qui est partageable

Pas possible si pas une forme d'adressage virtuel

- Sur les machines sans MV, création *via* `posix_spawn()`

Création de processus

Si les processus exécutaient tous le même code

- cela ne serait pas vraiment intéressant

Les processus peuvent muter

- Ils conservent leur identité et leur contexte
  - fichiers ouverts, etc
- Ils changent de code
- `exec*()`

Mutation de processus

```c
#include <stdio.h>
#include <unistd.h>
#include <stdint.h>
#include <stdlib.h>

int main() {
  printf("Bonjour je suis %jd\n",(intmax_t)getpid());
  pid_t son = fork();
  switch (son) {
  case -1:
    fprintf(stderr,"Problème\n");
    exit(EXIT_FAILURE);
  case 0:
    printf("Je suis %jd et mon parent est %jd\n",
           (intmax_t)getpid(),(intmax_t)getppid());
    execlp("ls","ls",NULL);
    exit(EXIT_FAILURE);
  default:
    printf("Je suis %jd et mon enfant est %jd\n",
           (intmax_t)getpid(),(intmax_t)son);
    wait(NULL);
    exit(EXIT_SUCCESS);
  }
}
```

Création/Mutation

Pourquoi avoir séparé les deux opérations ?

- C'est une des forces d'Unix
  - Entre les deux, on peut faire certaines opérations de sorte que le code finalement exécuté par le nouveau processus utilise des caractéristiques héritées depuis le processus d'origine
    - Par exemple : les redirections
      - Facilite l'abstraction des entrées/sorties!

Mutation

`exec*()`

- permet de repartir avec un nouvel espace d'adressage : code, données, etc
- le processus est toujours le même
  - comme lorsqu'on abandonne un livre que l'on a pas fini et que l'on en prend un nouveau…
- on ne revient jamais d'un appel à `exec*()`
  - sauf en cas d'erreur

Création/Mutation

On peut envisager être malin pour le clonage

- Le clonage peut être virtuel
- et passer à la réalité lorsque nécessaire

Copy-on-write

Copy on write

- On marque les pages qui sont à dupliquer et on les dupliquent lorsque nécessaire
  - L'astuce consiste à marquer ces pages en lecture seule
    - Le système sait que c'est un mensonge (car une table des pages n'est jamais protégé en écriture)
    - donc à l'écriture (si un des deux processus tente d'écrire), cela déclenche une erreur (violation d'accès) mais le système la corrige en dupliquant ce qui aurait du l'être

Copy on write

Traitement de l'erreur

- Allocation d'une page libre
- Copie de la page dans la page libre
- Marque les deux pages comme inscriptibles
- Reprise sur erreur

Technique «lazy» ou «retardée»

- on dirait *procastination*

Terminaison

La terminaison peut être obtenue de deux façon :

- par appel à `exit()` ou `_exit()`
- à réception d'un signal

Terminaison

Un appel à `_exit(valeur)` termine un processus de sorte que

- le parent en attente de sa terminaison (`wait`) soit débloqué et puisse recevoir la valeur transmise et l'appelant disparaît alors effectivement
- il soit transformé en zombie afin que son parent puisse prendre connaissance de sa disparition plus tard

Terminaison

Une anomalie durant l'exécution provoque la terminaison

- accès non conforme à une adresse
- division par zéro
- déconnexion
- …

Ces anomalies sont représentées par l'instance d'un signal envoyé

- pour certains signaux il y a génération d'un core, une image de l'espace d'adressage du processus au moment de sa terminaison
  - très utile pour retrouver la cause…

Synchronisation

Un processus en créé un autre souvent pour lui faire réaliser une tâche

- L'appelant peut souhaiter alors savoir si le processus créé a terminé ou non
  - Les deux processus sont indépendants du point de vue de l'exécution…
- L'appelant peut se synchroniser sur la terminaison
  - `wait()`

Synchronisation

`wait()`

permet d'attendre la terminaison d'un processus fils (il peut y en avoir plusieurs)

Le modèle de concurrence correspondant est appelé fork/join

Ex. : Fork/Join pool de Java

Asynchronisme

Le processus parent et son enfant sont liés

Ils ont leur propre exécution

- ☛ que se passe t-il lors de leur disparition ?

  l'enfant meure avant son parent

    zombie (un état supplémentaire)

  le parent avant son enfant

    adoption

Zombie

Un état qui représente un processus terminé

- plus d'espace d'adressage
- plus d'exécution
- ne reste que quelques indicateurs sur sa vie passée (stats, état de terminaison)
  - suffisant pour que le parent puisse déteriner cet état et donc faire disparaître le zombie

Adoption

Mécanisme permettant de raccrocher un processus sans parent à un parent de sorte à préserver la structure arborescente des processus

- en général (mais pas nécessaire) l'adoption est effectuée par un processus dédié (le premier né du système)

Signaux

Un signal est une information porteuse de sens et envoyée par le contexte ou l'environnement d'un processus

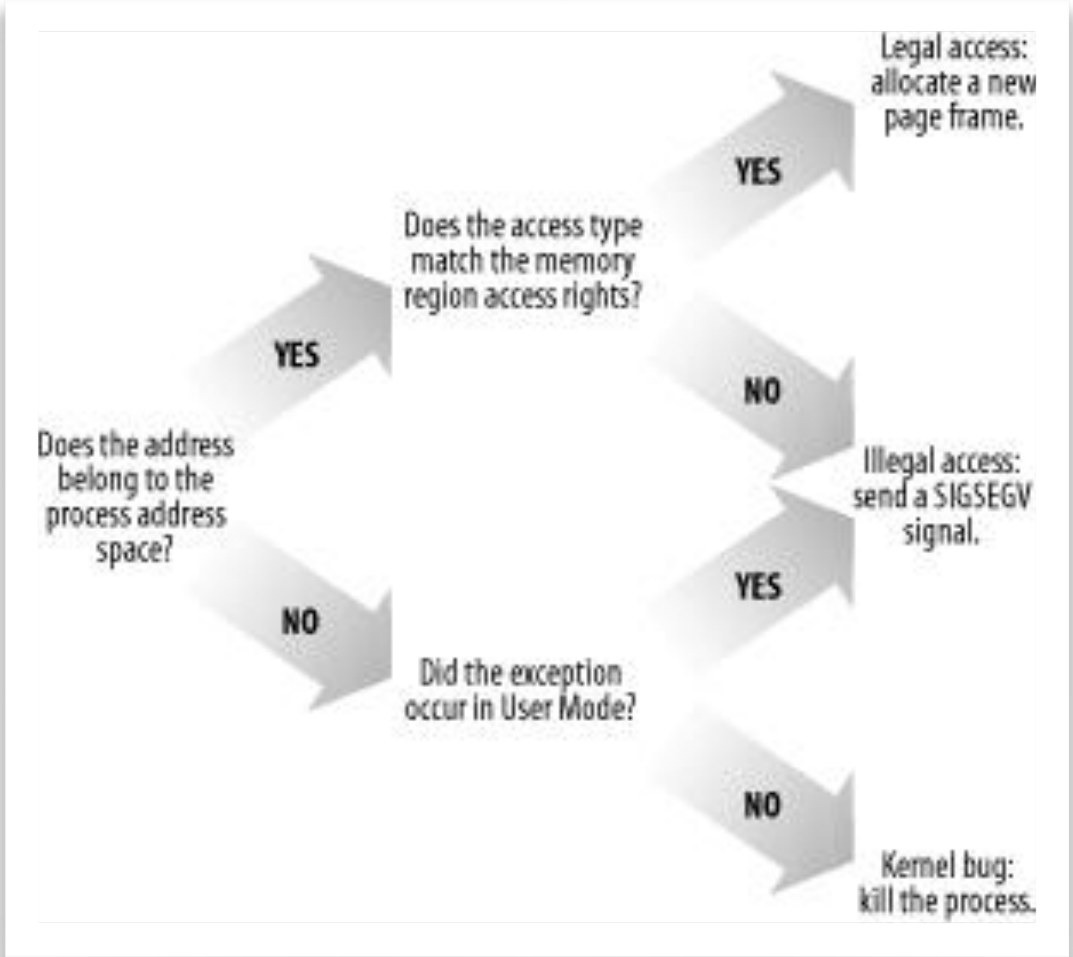
- contexte synchrone d'exécution (ex. : division par 0)
- contexte asynchrone (ex. : déconnexion)

Signaux

No	Name	Default Action	Description
1	SIGHUP	terminate process	terminal line hangup
2	SIGINT	terminate process	interrupt program
3	SIGQUIT	create core image	quit program
4	SIGILL	create core image	illegal instruction
5	SIGTRAP	create core image	trace trap
6	SIGABRT	create core image	abort program (formerly SIGIOT)
7	SIGEMT	create core image	emulate instruction executed
8	SIGFPE	create core image	floating-point exception
9	SIGKILL	terminate process	kill program
10	SIGBUS	create core image	bus error
11	SIGSEGV	create core image	segmentation violation
12	SIGSYS	create core image	non-existent system call invoked
13	SIGPIPE	terminate process	write on a pipe with no reader
14	SIGALRM	terminate process	real-time timer expired
15	SIGTERM	terminate process	software termination signal
16	SIGURG	discard signal	urgent condition present on socket
17	SIGSTOP	stop process	stop (cannot be caught or ignored)
18	SIGTSTP	stop process	stop signal generated from keyboard
19	SIGCONT	discard signal	continue after stop
20	SIGCHLD	discard signal	child status has changed
21	SIGTTIN	stop process	background read attempted from control terminal
22	SIGTTOU	stop process	background write attempted to control terminal
23	SIGIO	discard signal	I/O is possible on a descriptor (see fcntl(2))
24	SIGXCPU	terminate process	cpu time limit exceeded (see setrlimit(2))
25	SIGXFSZ	terminate process	file size limit exceeded (see setrlimit(2))
26	SIGVTALRM	terminate process	virtual time alarm (see setitimer(2))
27	SIGPROF	terminate process	profiling timer alarm (see setitimer(2))
28	SIGWINCH	discard signal	Window size change
29	SIGINFO	discard signal	status request from keyboard
30	SIGUSR1	terminate process	User defined signal 1
31	SIGUSR2	terminate process	User defined signal 2

Exemple

Workflow Linux relatif au traitement des erreurs mémoire



Signaux

À réception d'un signal le processus peut

- laisser le comportement par défaut opérer
  - en général ☛ terminaison du processus
    - `SIG_DFL`
- ignorer le signal
  - `SIG_IGN`
- exécuter un code dédié à la gestion du signal
  - handler

Signaux

NAME

sigaction - examine and change a signal action

SYNOPSIS

```
[CX] ⌦ #include <signal.h>

int sigaction(int sig, const struct sigaction *restrict act,
       struct sigaction *restrict oact); ⌫
```

DESCRIPTION

The *sigaction*() function allows the calling process to examine and/or specify the action to be associated with a specific signal. Th

The structure **sigaction**, used to describe an action to be taken, is defined in the *<signal.h>* header to include at least the follov

Member Type	Member Name	Description
void(*) (int)	*sa_handler*	Pointer to a signal-catching function or one of the macros SIG_IGN or SIG_DFL.
sigset_t	*sa_mask*	Additional set of signals to be blocked during execution of signal-catching function.
int	*sa_flags*	Special flags to affect behavior of signal.

Signaux

L'interface POSIX (moderne) de gestion des signaux permet

- de contrôler finement le comportement vis-à-vis des signaux
  - dont blocage temporaire d'autres signaux durant l'exécution d'un handler
    - auto-concurrence ?

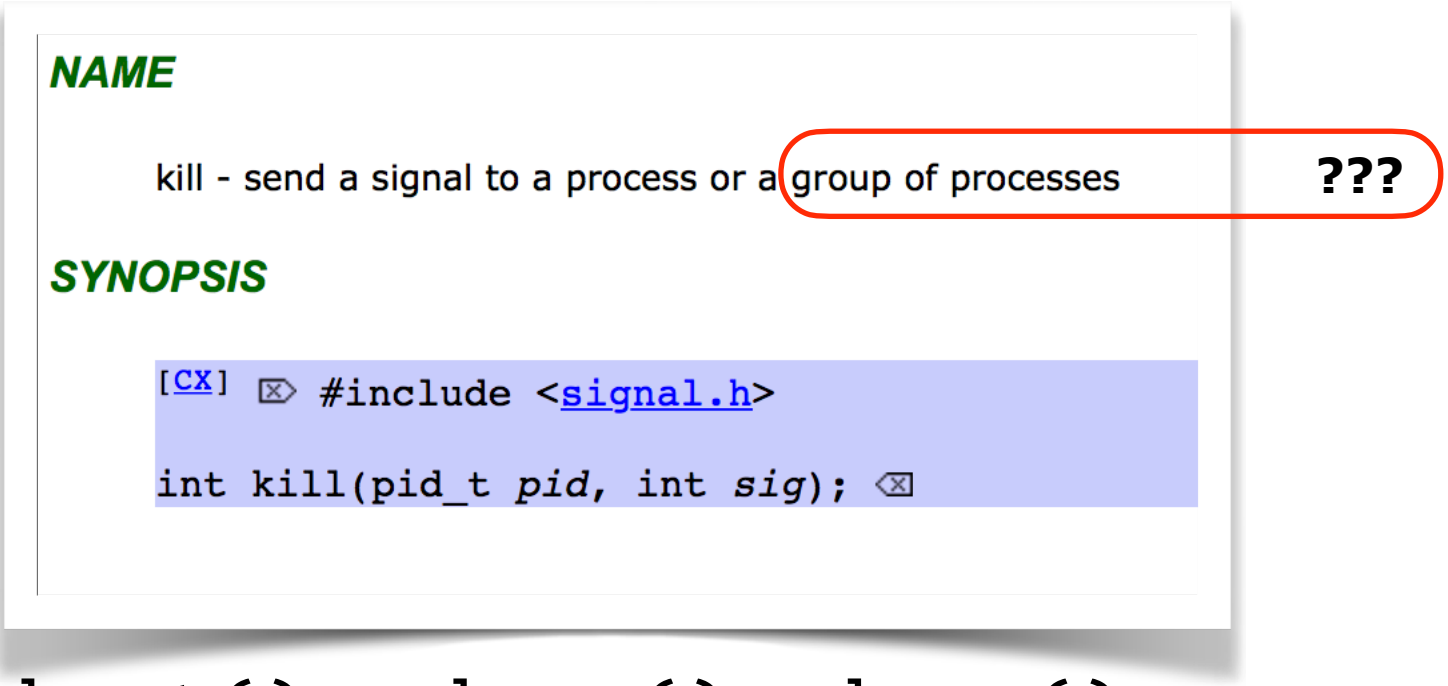
Signaux

`sa_flags` permet de contrôler le comportement à réception du signal

- `SA_NOCLDSTOP`
- `SA_ONSTACK`
- `SA_RESETHAND`
- `SA_RESTART`
- `SA_SIGINFO`
- `SA_NOCLDWAIT`
- `SA_NODEFER`

Signaux

On peut envoyer des signaux (sous certaines conditions — propriété)



NAME

kill - send a signal to a process or a group of processes ???

SYNOPSIS

```
[CX] #include <signal.h>

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

ou via `abort()`, `sleep()`, `alarm()`

Groupes

La notion de groupe de processus permet de manipuler un ensemble de processus logiquement reliés

- une application ?
- une session ?

L'incarnation de cette notion se décline en groupe et session…

- c'est assez complexe à manipuler
  - l'idée générale est qu'il y a un chef (leader) dont le rôle est de «représenter» le groupe

Signaux

Signaux et exec ?

en cas de recouvrement, les captures de signaux sont naturellement (?!) supprimées et le comportement par défaut est rétabli

Tubes anonymes

Des processus peuvent communiquer par le moyen de tube anonymes

NAME

pipe - create an interprocess channel

SYNOPSIS

```
#include <unistd.h>

int pipe(int fildes[2]);
```

DESCRIPTION

The *pipe*() function shall create a pipe and place two file descriptors, one each into the arguments *fildes*[0] and *fildes*[1], that refer to the open file descriptions for the read and write ends of the pipe. Their integer values shall be the two lowest available at the time of the *pipe*() call. The

Tubes anonymes

Le tube est un canal de communication

- unidirectionnel
- capacité limitée
- pas de concept de message
- ☛ les lectures «consomment» les données

Il synchronise les écrivains et les lecteurs

- lecture sur tube vide : attente d'écriture
- écriture sur tube plein : attente de lecture

La synchronisation cale les écrivains/lecteurs sur les plus lents

- la capacité permet un certain amortissement

Tubes anonymes

```
int main() {
  int t[2];
  if (pipe(t)==-1) {
    perror("pipe");
    exit(1);
  }
  switch(fork()) {
  case 0:
    close(t[1]); child(t[0]); close(t[0]);
    exit(0);
  default:
    close(t[0]); parent(t[1]); close(t[1]);
    wait(NULL);
    exit(0);
  case -1:
    perror("fork");
    close(t[0]);
    close(t[1]);
    exit(1);
  }
}
```

```
void child(int d) {
  char c;
  while(read(d,&c,1)>0) {
    printf("J'ai lu %c\n",c);
  }
}

void parent(int d) {
  srandom(time(NULL));
  while(1) {
    char c = (random()%26)+'a';
    write(d,&c,1);
    if (c=='q') break;
  }
}
```

Tubes anonymes

Il y a une véritable sémantique de communication

- une lecture sur un tube vide sans écrivain
  - ➥ «fin de fichier»
- une écriture sur un tube sans lecteur
  - ➥ erreur! Un signal est envoyé `SIGPIPE`

Lourdeur

Le problème est que la création de processus, malgré toutes les optimisations est un mécanisme coûteux

- Or la création de containers d'exécution est très utile
  - Ex. : serveur
    - un serveur est souvent architecturé de sorte qu'une entité attend les connexions entrantes et délègue la réalisation du service à une autre entité

Lourdeur

Il faut trouver un moyen de créer des activités permettant de traiter des requêtes de façon asynchrone sans la lourdeur de la création de processus

- Le thread
  - une exécution au sein d'un processus
- Le processus est donc un container
  - un espace d'adressage
  - des ressources d'exécution

Les processus légers

LWP Lightweight Process

- Construction System V et Solaris

Une construction autorisant du multi-tâche à l'intérieur d'un processus

- Fonctionnent en mode user

Ne sont plus utilisés car comportent certaines lourdeurs

Les threads

POSIX 1995.1c fournit un support standardisé pour les threads

Les threads

`pthread_create()`

fork?

`pthread_exit()`

exit?

`pthread_join()`

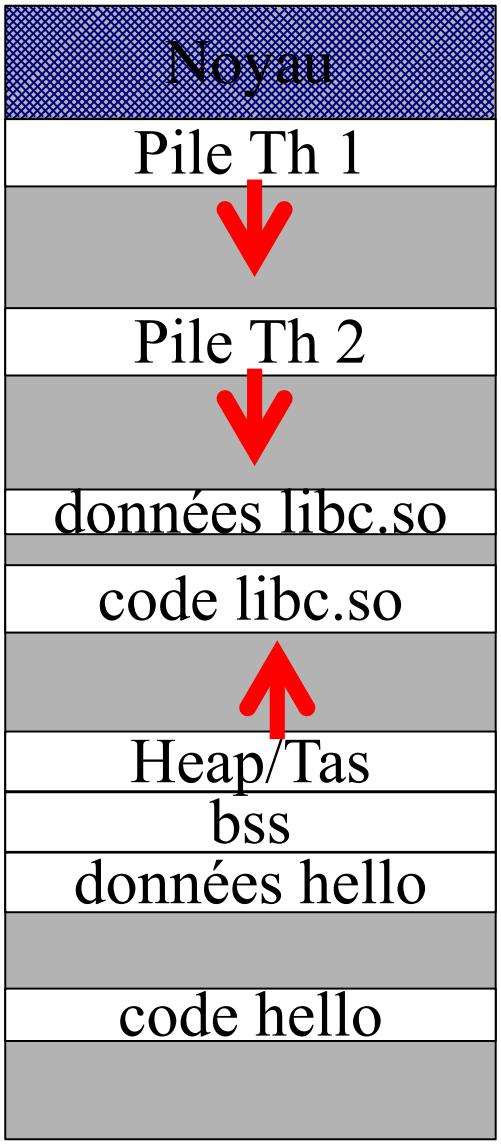
wait?

`pthread_self()`

getpid?

pthread_create

- Espace partagé
- Pas de protection entre piles
- Signaux réceptionnés par n'importe quel thread
  - masquer les signaux dans ceux qui ne veulent pas être perturbés



Threads exemple

```
#include <pthread.h>
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

void *run(void *param) {
  for (int i=0; i<10; i++) {
    usleep(1000000L);
    printf("Yo!\n");
  }
  return NULL;
}

int main() {
  pthread_t p;
  pthread_create(&p,NULL,run,NULL);
  pthread_join(p,NULL);
}
```

Threads

```
Processes: 440 total, 2 running, 438 sleeping, 1614 threads
Load Avg: 1.42, 1.46, 1.51  CPU usage: 0.60% user, 2.66% sys, 96.73% idle
SharedLibs: 278M resident, 63M data, 56M linkedit. MemRegions: 81781 total, 5985M r
PhysMem: 13G used (2049M wired), 3416M unused.
VM: 2957G vsize, 1112M framework vsize, 98961(0) swapins, 174327(0) swapouts.
Networks: packets: 6467141/8903M in, 3801318/311M out. Disks: 4906197/60G read, 197

PID    COMMAND      %CPU TIME     #TH   #WQ  #PORT MEM    PURG   CMPRS  PGRP  PPID
0      kernel_task  17.3 02:57:09 146/8 0    2     1540M+ 0B     0B     0     0
18164  top          3.6  00:02.26 1/1   0    24    6096K  0B     0B     18164 12384
555    Terminal     3.3  02:14.39 9     4    337   155M   23M    2584K  555   1
236    WindowServer 1.7  08:23.56 10    3/1  2605  148M+  31M    37M    236   1
1669   Backup and S 0.5  04:16.16 50    1    322   77M    44K    110M   1669  1
749    Dropbox      0.1  02:31.61 135   2    788   185M   12K    7532K  749   1
17868  com.apple.We 0.1  00:01.06 5     1    166   33M    8192B  0B     17868 1
1684   FinderSyncAP 0.1  00:25.87 3     1    145   6476K  97M    1228K  1684  1
18173  screencaptur 0.1  00:00.03 3     2    52    2632K+ 36K    0B     560   560
485-   EEventManage 0.0  00:42.91 6     1    165   5108K  0B     16K    485   1
706-   Epson Scanne 0.0  00:47.20 3     1    150   6220K  0B     40K    706   1
106    launchservic 0.0  00:58.24 4     4    521   10M    0B     268K   106   1
128    hidd         0.0  02:36.94 6     2    198   3464K  0B     4096B  128   1
1487   Dropbox Web  0.0  00:06.08 5     1    85    4376K  0B     9764K  749   749
78     configd      0.0  00:22.20 8     1    451   8364K  0B     260K   78    1
777    owncloud     0.0  05:31.83 10    2    249+  363M+  12K    13M    777   1
440    universalacc 0.0  00:08.72 4     2    170   4256K  0B     4096B  440   1
1      launchd      0.0  06:06.12 4     4    3090  24M    0B     248K   1     0
851    mysqld       0.0  00:17.61 27    0    45    159M   0B     0B     711   711
90     mds          0.0  03:09.75 9     4    616   77M    0B     2372K  90    1
442    CommCenter   0.0  00:13.39 8     3    285   12M    0B     16K    442   1
458    SafariBookma 0.0  00:16.26 4     2    68    14M    0B     0B     458   1
452    SafariCloudH 0.0  00:16.00 4     3    42    5652K  0B     28K    452   1
656    ViewBridgeAu 0.0  00:04.83 2     1    79    1892K  0B     32K    656   1
377    watchdogd    0.0  00:00.53 4     1    37    712K   0B     28K    377   1
455    gamed        0.0  00:04.40 4     2    120+  7332K+ 12K    2840K  455   1
79     powerd       0.0  00:10.09 2     1    99    2080K  0B     860K   79    1
692    com.apple.ge 0.0  00:02.48 4     2    57+   5604K+ 0B     3148K  692   1
71     fseventsd    0.0  01:02.52 10    1    270   18M    0B     912K   71    1
721    nbagent      0.0  00:05.03 4     1    186   3292K  0B     3572K  721   1
```

Threads

Threads et fork ?

Si un thread appelle fork, le processus est dupliqué mais seul le thread appelant est «copié»

`pthread_atfork` ?

Concurrence, Ordonnancement

E/S - I/O

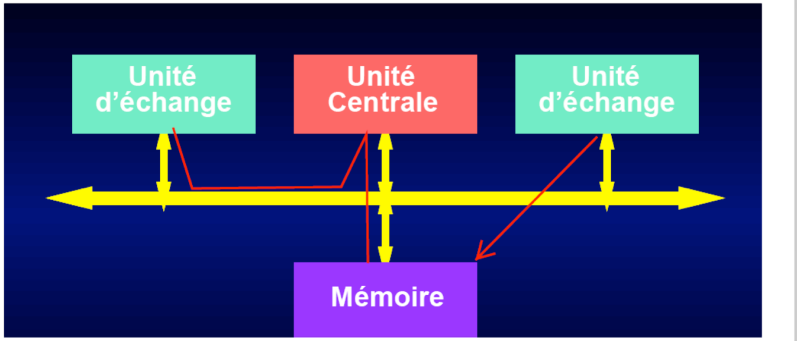
Dans une architecture simple, c'est le processeur qui s'occupe de réaliser les entrées/sorties avec les périphériques

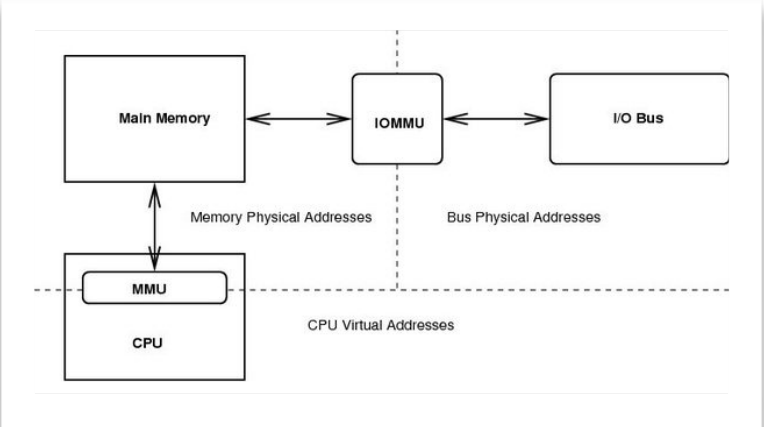
- ☛ il passe donc son temps à attendre les résultats de l'E/S

  Rappel : vitesse processeur / vitesse périphérique

DMA

Dans une architecture DMA, un processeur dédié (contrôleur de périphérique) s'occupe de réaliser l'entrée/sortie





DMA

Dans une architecture DMA

- on décharge le processeur principal qui peut alors être utilisé à d'autres fins
  - exécuter un autre programme en attendant ?

Attention : on introduit de la concurrence…

- le gestionnaire de mémoire doit prendre garde à protéger les pages sur lesquelles des entrées/sorties sont en cours! Verrouillage…

DMA

- Deux techniques
  - le polling
    - le processeur passe son temps à interroger le contrôleur : «est-ce terminé ?» - non, «est-ce terminé ?» - non, etc
      - le polling est intéressant lorsque l'opération concernée est de temps de réalisation court
  - l'interruption
    - le processeur passe à autre chose, le contrôleur déclenche une interruption en fin d'opération qui exécute un code associé dans le système
      - l'interruption est de nature asynchrone!

Interruption

Une **interruption** est la suspension temporaire par un évènement d'une exécution en vue d'exécuter un autre code appelé service d'interruption à la suite duquel (normalement) l'exécution interrompue reprend

- nécessite la gestion d'un contexte d'exécution
  - il faut reprendre l'exécution interrompue dans l'état exact où elle se trouvait

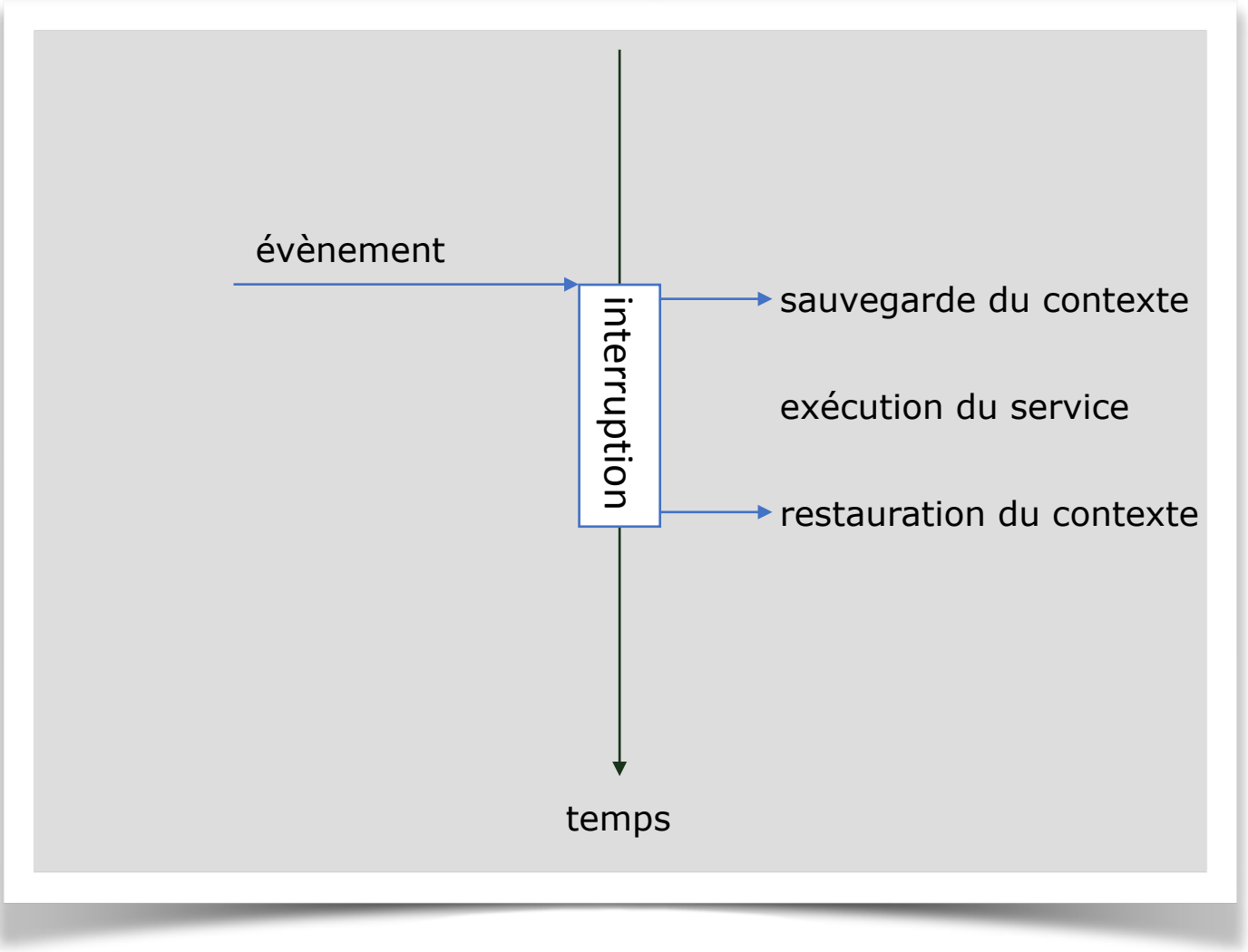
Interruption : appel de fonction asynchrone!

Interruption

On en distingue généralement trois types :

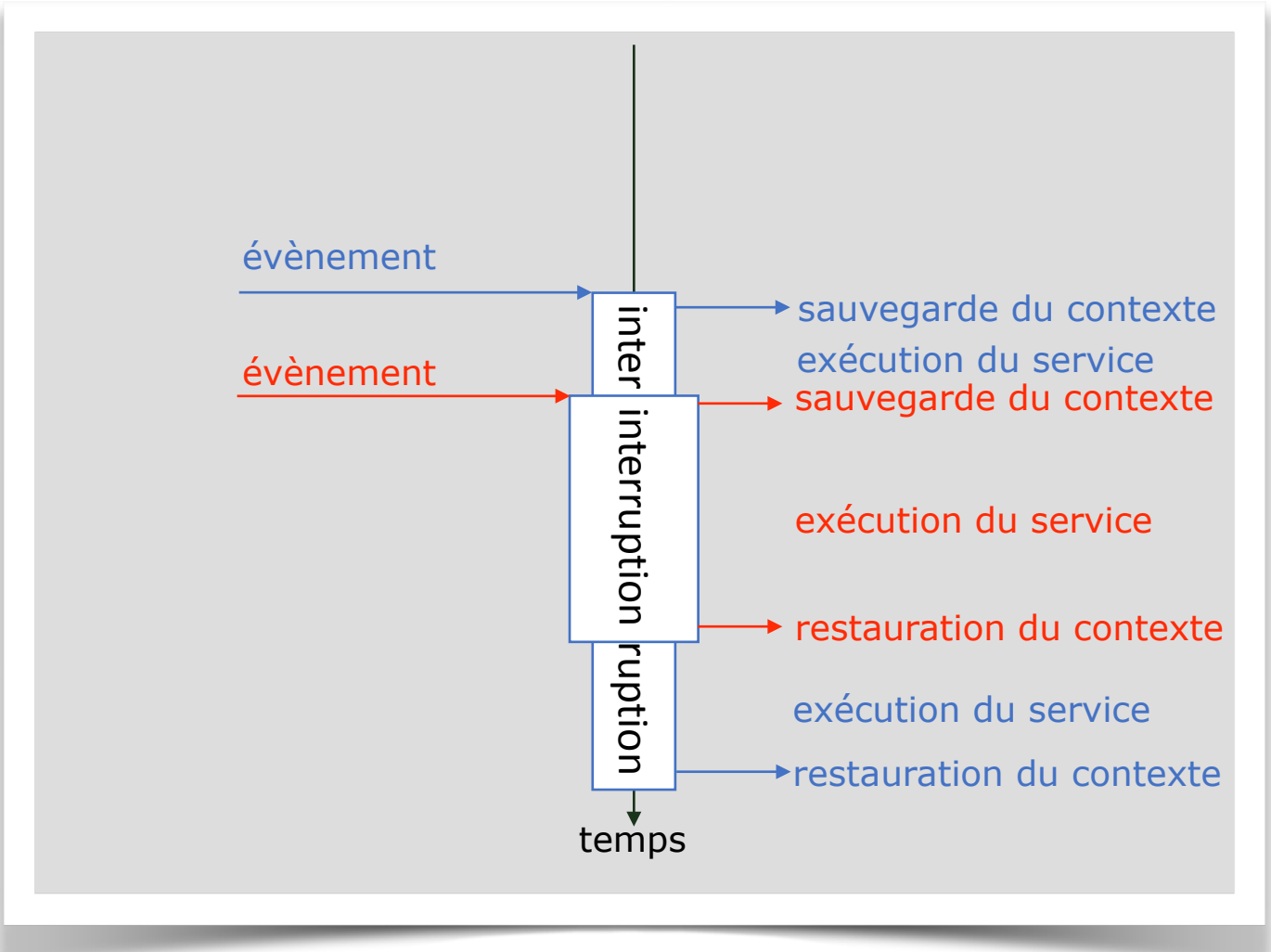
- matérielles : provoquées par un dispositif externe (clavier, souris, etc)
- logicielles : provoquées explicitement par du code
- exceptions : des erreurs
  - fautes : erreur pouvant être réparée, page fault
  - trappes : points d'arrêts, appels systèmes
  - avortements : erreurs d'inconsistance grave

Interruption



Interruption

Problème : l'imbrication!



Interruption

Deux types :

- asynchrones / externes
  - périphériques (clavier, souris, horloge, disque, réseau…)
  - ne sont pas reliées à l'instruction en cours
- synchrones / internes
  - hardware
    - fautes / faults : peuvent être corrigées (page **fault**)
    - trappes / traps : utilisées pour déboguer
    - avortements / aborts : problème majeur (`die()`)
  - software
    - appel au noyau (intr/syscall)

Attention : les signaux sont des interruptions logicielles, des constructions du noyau. Certaines interruptions provoquent l'émission d'un signal (`force_sig`)

Interruption

Contexte d'exécution :

- compteur ordinal (IP)
- sommet de pile
- état du processeur (pipeline, …)
- …

Dépend du processeur!

Gestion des interruptions

- Masquer les interruptions
  - les bloquer temporairement car on ne veut pas être interrompu à ce moment-là…
    - auto-masquage ?
- Priorité
  - certains périphériques sont plus critiques que d’autres…
    - horloge plus prioritaire que disque (?)
- Pile d’interruption
  - gérer l’imbrication…

Interruptions Linux

```
parallels@ubuntu: ~
parallels@ubuntu:~$ cat /proc/interrupts
           CPU0       CPU1
  0:          4          0   IO-APIC   2-edge      timer
  1:         11         78   IO-APIC   1-edge      i8042
  8:          1          0   IO-APIC   8-edge      rtc0
  9:       1264        391   IO-APIC   9-fasteoi   acpi
 12:        144       1171   IO-APIC  12-edge      i8042
 14:          0          0   IO-APIC  14-edge      ata_piix
 15:          0          0   IO-APIC  15-edge      ata_piix
 17:        437        396   IO-APIC  17-fasteoi   snd_intel8x0
 18:          0          0   IO-APIC  18-fasteoi   uhci_hcd:usb2
 19:        165        987   IO-APIC  19-fasteoi   ehci_hcd:usb1
 22:          0          0   IO-APIC  22-fasteoi   virtio1
 24:          0          0   PCI-MSI 81920-edge      virtio0-config
 25:         13         21   PCI-MSI 81921-edge      virtio0-input.0
 26:          1          0   PCI-MSI 81922-edge      virtio0-output.0
 27:          0          0   PCI-MSI 487424-edge      xhci_hcd
 28:       6275       7976   PCI-MSI 512000-edge      0000:00:1f.2
 29:     364244          0   PCI-MSI 49152-edge      prl_tg
 30:          4       2166   PCI-MSI 524288-edge      prl_vtg
NMI:          0          0   Non-maskable interrupts
LOC:      32453      33392   Local timer interrupts
SPU:          0          0   Spurious interrupts
PMI:          0          0   Performance monitoring interrupts
IWI:          0          0   IRQ work interrupts
RTR:          0          0   APIC ICR read retries
RES:      54934     400466   Rescheduling interrupts
CAL:       1417       1652   Function call interrupts
TLB:        419        378   TLB shootdowns
TRM:          0          0   Thermal event interrupts
THR:          0          0   Threshold APIC interrupts
DFR:          0          0   Deferred Error APIC interrupts
MCE:          0          0   Machine check exceptions
MCP:          1          1   Machine check polls
ERR:          0
MIS:          0
PIN:          0          0   Posted-interrupt notification event
PIW:          0          0   Posted-interrupt wakeup event
parallels@ubuntu:~$
```

Interruption vs Context switch

- Rien n'oblige à reprendre l'exécution du programme interrompu
  - on peut reprendre l'exécution d'un autre programme
    - la restauration d'un contexte n'implique pas la restauration du dernier contexte sauvé!
- Lorsque l'interruption est celle d'une horloge on parle de changement de contexte / context switch
  - le service consiste juste à sauver le contexte courant et en restaurer un autre
    - la péemption et l'ordonnancement…

Préemption

Propriété d'un système autorisant l'interruption temporaire (ou définitive) d'une tâche en cours afin de réaliser une tâche de priorité égale ou plus élevée

- dans la vie courante : répondre au téléphone, répondre à une question au milieu d'un cours…
- la tâche préemptée reprendra plus tard

La préemption n'intervient qu'à l'aide d'un événement extérieur

- fin d'E/S, horloge,…

Préemption

La préemption (sans soin particulier) peut conduire à des incohérences (concurrence)

- modification de variables qui sont utilisées par le code préempté…
- modification de structures de données utilisées par le code préempté…

Il faut donc parfois se prémunir contre certaines préemptions

- introduction de délai dans la prise en compte des événements

Concurrence (problème)

La concurrence pose problème dans le cas d'un accès concurrent à une ressource

- sécurité
  - coffre-fort d'une banque
- cohérence
  - réservation d'un billet d'avion

Concurrence (problème)

```
#define MAXLOOP 100000000
static volatile int value;

void *my_func(void *arg) {
  for (int i=0; i < MAXLOOP; i++){
    value++;
  }
  return NULL;
}
```

Concurrence (problème)

```
int main() {
   my_func(NULL);
   my_func(NULL);
   printf("Value: %d\n", value);
}
```

```
ask> gcc -O0 -o test test.c
ask> ./test
Value: 200000000
ask>
```

Concurrence (problème)

```
int main() {
  pthread_t th1, th2;
  int res;
  res = pthread_create(&th1,NULL,my_func,NULL);
  res = pthread_create(&th2,NULL,my_func,NULL);
  res = pthread_join(th1, NULL);
  res = pthread_join(th2, NULL);
  printf("Value: %d\n", value);
}
```

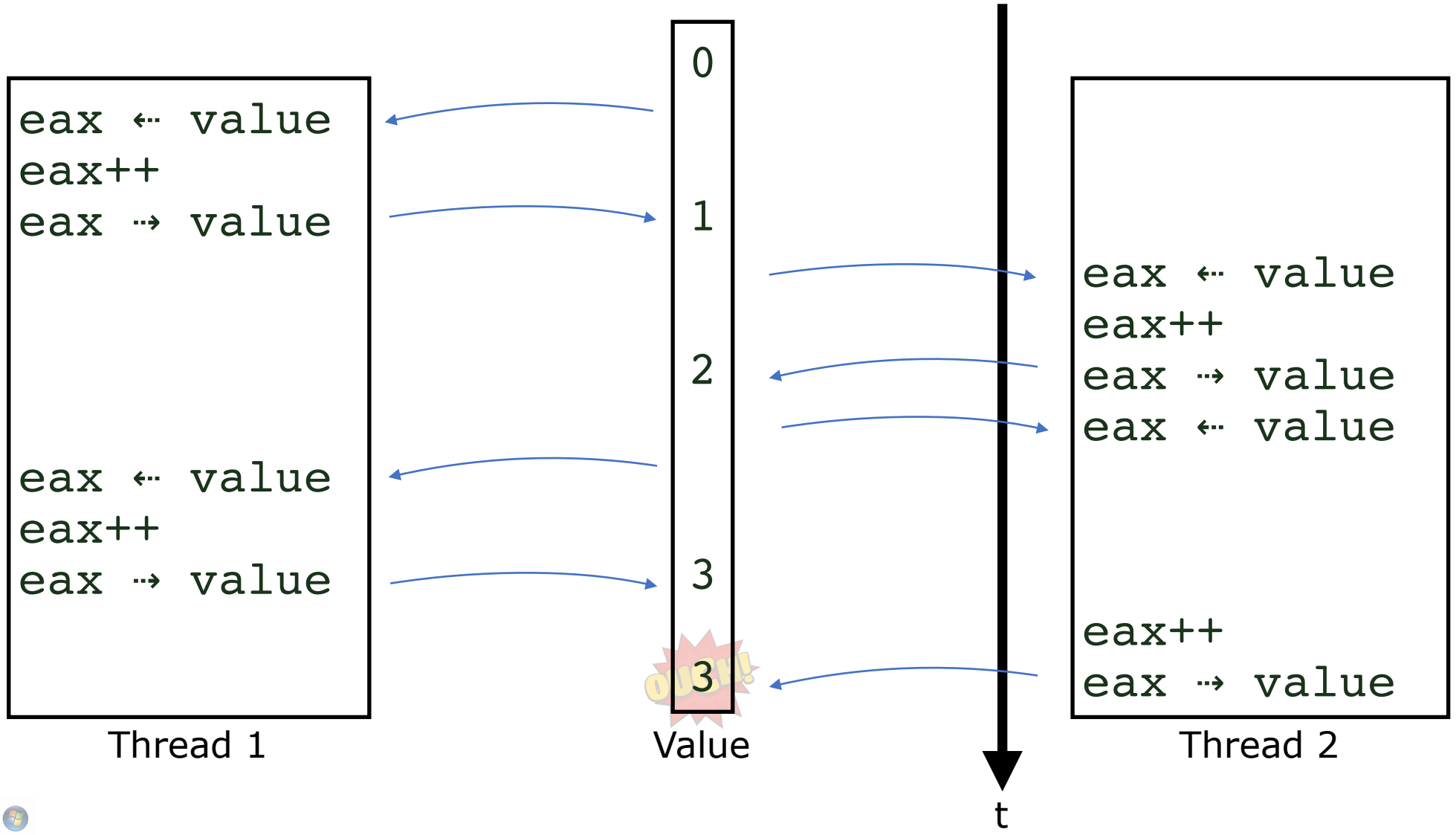
```
ask> gcc -O0 -o test test.c
ask> ./test
Value: 101594476
ask> ./test
Value: 102256442
Ask>
```

Concurrence (problème)

Traduction assembleur (sans optimisation!)

```
LBB0_1:                                 ## =>This Inner Loop Header: Depth=1
        cmpl    $100000000, -12(%rbp)   ## imm = 0x5F5E100
        jge     LBB0_4
## BB#2:                                ##   in Loop: Header=BB0_1 Depth=1
        movl    _value(%rip), %eax      eax = value
        addl    $1, %eax                      eax++
        movl    %eax, _value(%rip)      value = eax
## BB#3:                                ##   in Loop: Header=BB0_1 Depth=1
        movl    -12(%rbp), %eax
        addl    $1, %eax
        movl    %eax, -12(%rbp)
        jmp     LBB0_1
LBB0_4:
        xorl    %eax, %eax
                                        ## kill: %RAX<def> %EAX<kill>
        popq    %rbp
        retq
```

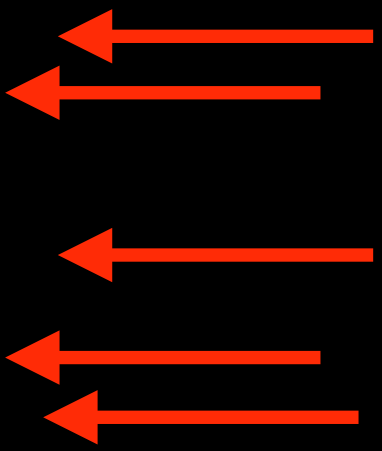
Dépendance vis-à-vis de l'ordonnancement



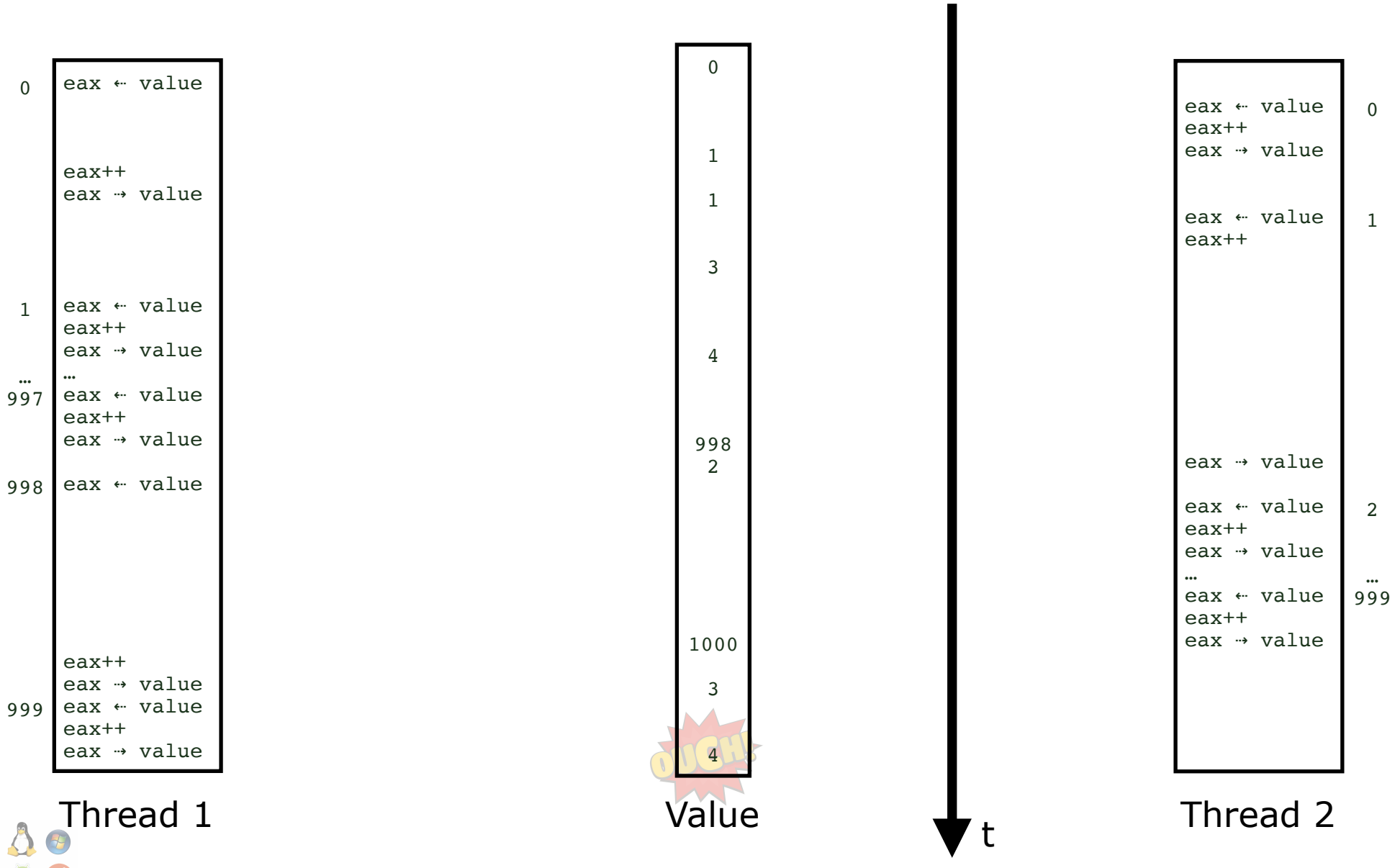
Concurrence (problème)

Pire : on peut « revenir » en arrière!

```
ask> for i in 1 2 3 4 5 6 7 8 9
do
  ./test
done
Value: 104218758
Value: 100816077
Value: 96476592
Value: 99058844
Value: 102678228
Value: 101099097
Value: 94603564
Value: 104799101
Value: 89938246
Value: 95373723
Ask>
```

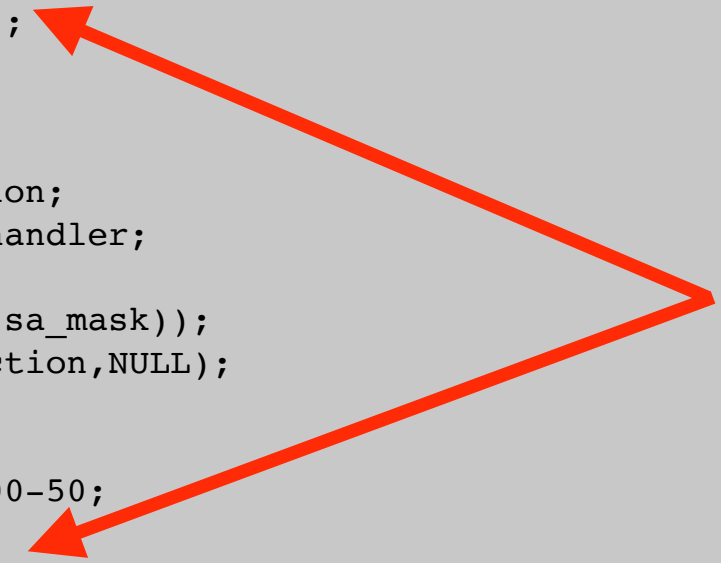


Dépendance vis-à-vis de l'ordonnancement



Concurrence (problème)

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <time.h>

int compte1=1000;
int compte2=0;

void handler(int sig) {
  if (compte1+compte2 != 1000) printf("Bad\n");
  else printf("Good\n");
}

int main() {
  struct sigaction action;
  action.sa_handler = handler;
  action.sa_flags = 0;
  sigemptyset(&(action.sa_mask));
  sigaction(SIGUSR1,&action,NULL);
  srandom(time(NULL));
  while (1) {
    int v = random()%100-50;
    compte1 -= v;
    compte2 += v;
  }
}
```



Concurrence (problème)

```
Good
Good
Good
Good
Good
Good
Bad
```

```
ask> kill -USR1 94767
ask> kill -USR1 94767
ask> kill -USR1 94767
ask> kill -USR1 94767
ask> kill -USR1 94767
ask> kill -USR1 94767
ask> kill -USR1 94767
```

Concurrence

On ne peut pas ne pas s'en préoccuper

Pour les applications :

- un processus mono-thread pas de problème
  - Euh… délivrance des signaux ?
- plusieurs threads, il faut agir

Pour le noyau

- la concurrence est rude!
  - les processus, les threads, les interruptions, les tâches internes…
  - big lock, fine locking

Rappels

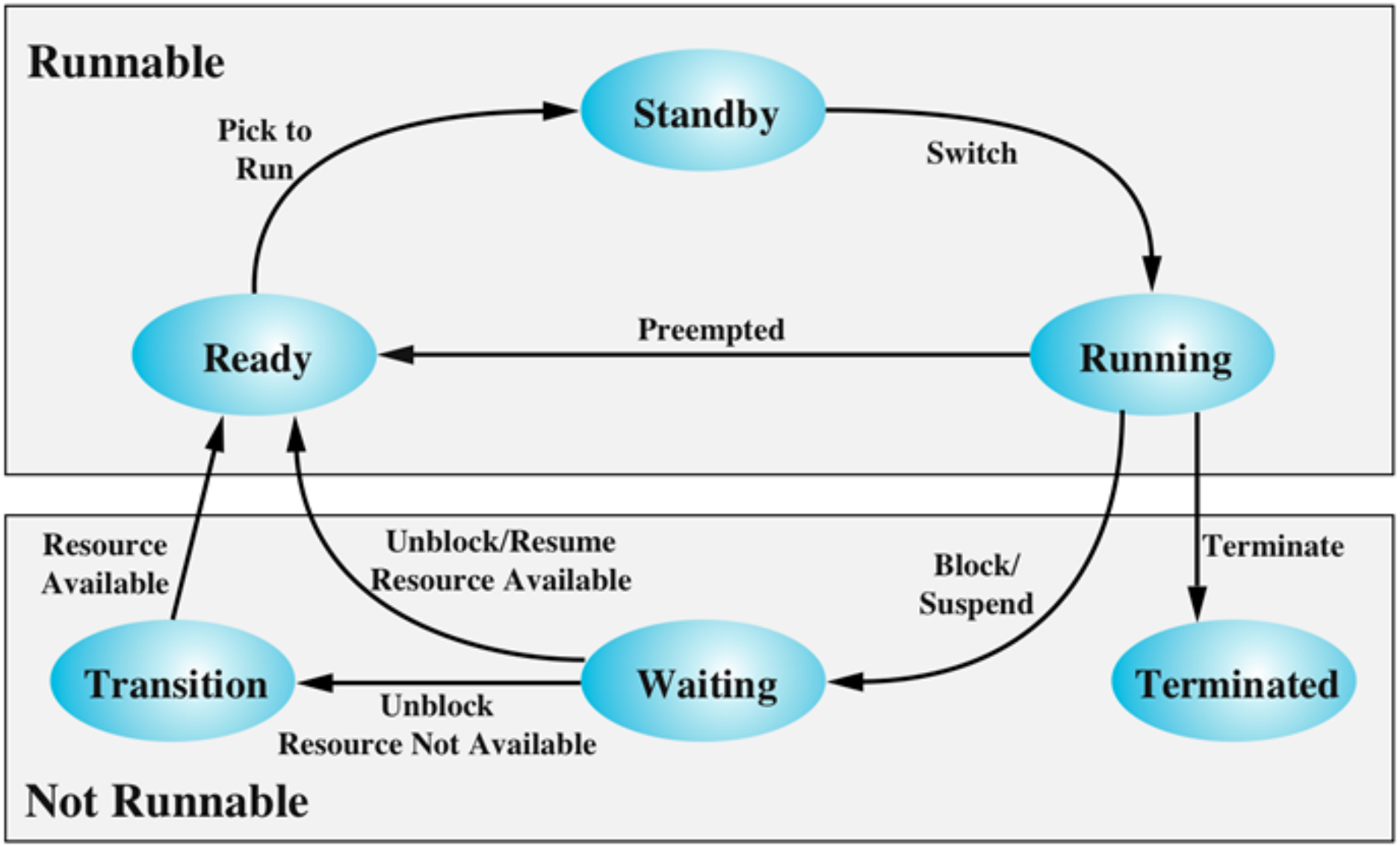
- On n'exécute pas les threads les uns après les autres
  - le temps du batch est révolu (sûr ?)
- On a en général suffisamment de processeurs (cœurs, hyperthreads)
- Dans le cas monoprocesseur (pour rester simple)
  - un composant du noyau attribue à chaque thread l'exclusivité de la ressource processeur pour un temps donné limité (quantum, slice)
    - ce composant est l'ordonnanceur

Si la machine dispose de *c* ressources de calculs, et qu'il y a *t* threads

- si $t \leq c$ chaque thread peut recevoir la ressource processeur dont il a besoin
- is $t > c$ à un instant donné *c* thread peuvent utiliser la ressource et $t-c$ sont en attente…

On a donc dans le noyau des indications sur l'état d'un thread…

États d'un processus (Windows)



États d'un processus (Linux)

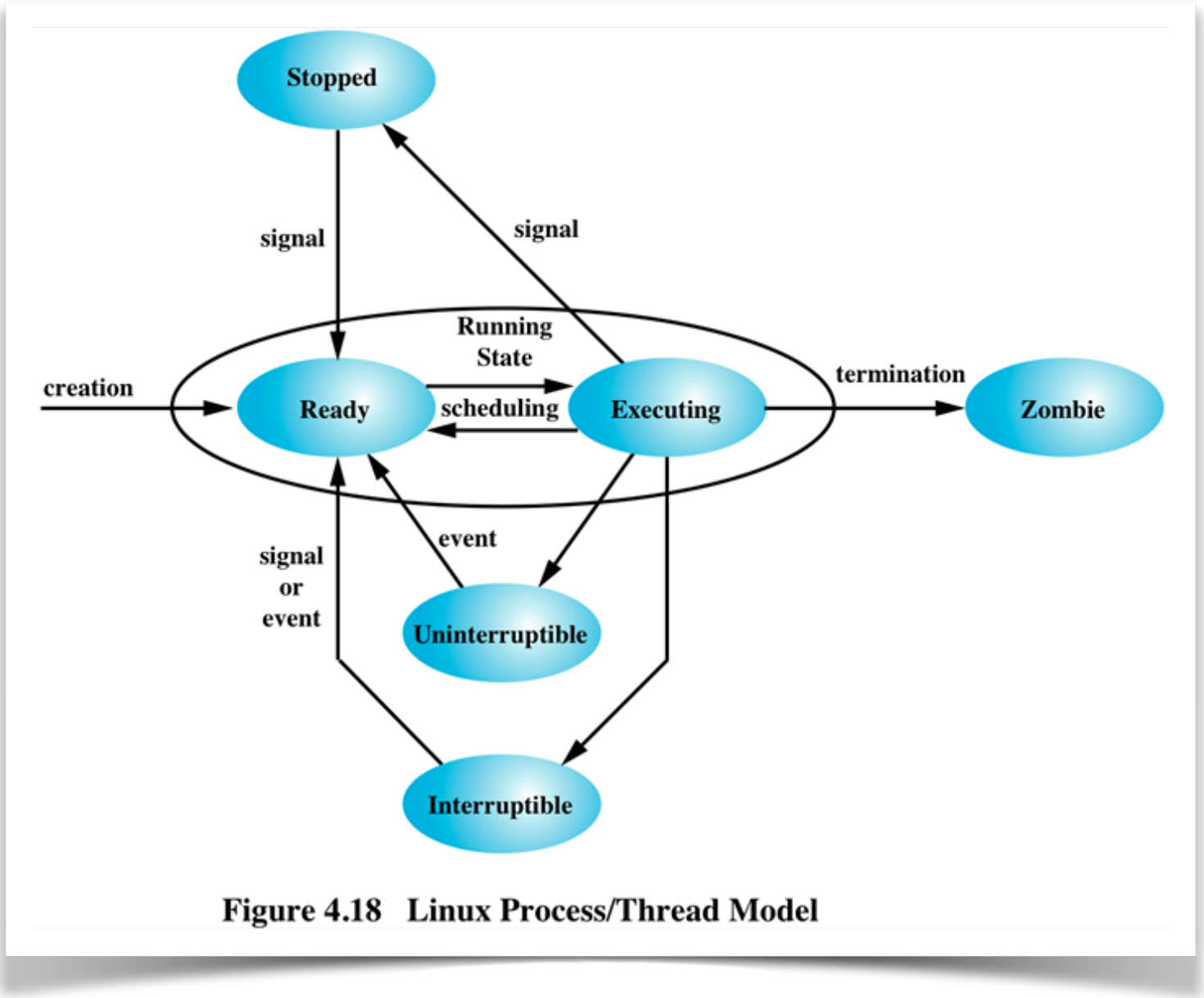


Figure 4.18 Linux Process/Thread Model

États d'un processus (Unix)

zOMBIE

slEEPING

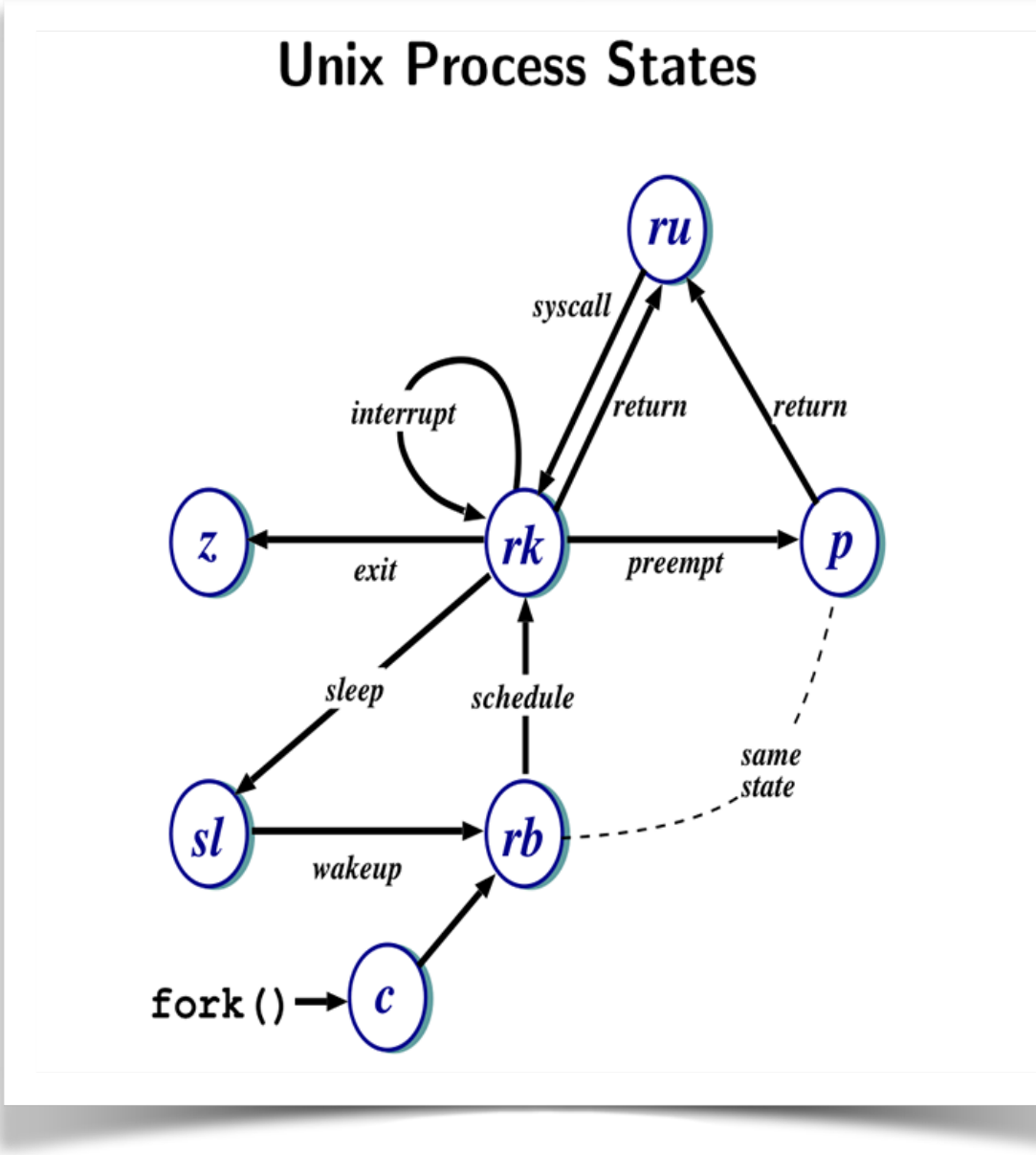
rUNNING uSER

cREATED

rUNNING kERNEL

pREEMPTED

rUNNAbLE



États d'un processus (Unix)

macOS

```
[Estafette:~/Documents/Formation/TELECOM/code/threads] ps -lea
  UID   PID  PPID       F CPU PRI NI        SZ    RSS WCHAN   S    ADDR       TTY      TIME CMD
    0     1     0    4004   0  37  0   4374628  25476 -       Ss      0        ??  6:16.94 /sbin/launchd
  501 18694     1    4004   0   4  0   4331956  16360 -       Ss      0        ??  0:00.06 /System/Library/Fr
    0 12383   555    4106   0  31  0   4324944   3580 -       Ss      0   ttys002  0:00.08 login -pf yunes
  501 12384 12383    4006   0  31  0   4334952   5176 -        S      0   ttys002  0:00.80 -zsh
    0 18817 12384    4106   0  31  0   4268280   1316 -       R+      0   ttys002  0:00.00 ps -lea
    0 16667   555    4106   0  31  0   4341328   6404 -       Ss      0   ttys003  0:00.04 login -pf yunes
  501 16668 16667    4006   0  31  0   4334952   5948 -       S+      0   ttys003  0:00.77 -zsh
```

Solution pour la concurrence

Mutex

Sémaphores

Spin-locks

Mémoire transactionnelle

Algorithmes lock-free

Transactions (BD pas OS)

Mutex

- MUTual EXclusion ou verrou (lock)
  - outil de sérialisation pour l'exécution d'une section critique (portion de code critique)
- Ce n'est pas un outil de synchronisation (entente entre acteurs), il n'y qu'une file d'attente
  - C'est similaire à une clé pour entrer dans une pièce
    - Si vous obtenez la clé vous entrez et en sortant vous redéposez la clé
    - Si vous n'avez pas la clé, vous attendez qu'elle revienne
- Le mutex c'est la gestion de l'entrée dans un coffre-fort de banque
  - On assure qu'une seule personne est présente dans la salle des coffres

Mutex

Services POSIX

- `pthread_mutex_init`,
- `pthread_mutex_lock`,
- `pthread_mutex_unlock`,
- `pthread_mutex_trylock`

Concurrence (solution)

```
#define MAXLOOP 100000000
static volatile int value;

void *my_func(void *arg) {
  pthread_mutex_init(…);
  for (int i=0; i < MAXLOOP; i++){
    pthread_mutex_lock(…);
    value++;
    pthread_mutex_unlock(…);
  }
  return NULL;
}
```

Concurrence (solution)

- L'utilisation de primitives de synchronisation a un impact sur les performances
  - mais l'on ne peut s'en passer
    - maîtriser les performances dans un contexte concurrent est assez difficile

Mutex

- Propriétés
  - Intra- ou Inter-processus (Attribut `PTHREAD_PROCESS_SHARED`)
  - Récursif (réentrant)
  - Détection verrouillage mortel ?
- Problème
  - Inversion de priorité
- Plus le mutex possède de propriétés, plus il coûte cher!
- Support hardware : Test And Set
  - Ex. : Intel `CMPXCHG`, CoMPare and eXCHanGe

Sémaphore

Le sémaphore est un mécanisme de signalisation entre acteurs utilisant une ressource

- Les opérations de base sont P (puis-je?) et V (vas-y!)
- P(s) : prend le sémaphore s'il est libre et en attente sinon
- V(s) : signale ceux qui attendent que la voie est libre

Une valeur est associée à un sémaphore : le nombre d'exemplaires de la ressource…

- Modèle producteur-consommateur

P() V()

Deux opérations atomiques…

```
V() :
  cpt = cpt+1
  Si ∃p∈attente reveiller(p)
```

```
P() :
  Si cpt<0 attendre
  cpt = cpt-1
```

Concurrence au sein du noyau

processus/threads invocation concurrente du noyau

- multi-processeur/multi-cœurs/hyper-thread
- mono-processeur : thread en attente I/O et autre thread qui fait un appel système

threads et interruption

interruption et interruption

…

Linux et la concurrence noyau

- 2.4
  - Big kernel lock : un seul verrou pour tout le noyau
    - efficacité réduite
      - Loi d'Amdahl (parallélisation)
      - très mauvais pour le temps-réél
- 2.6
  - verrouillage de granularité fine (chaque objet ?)
    - un travail sur le noyau long et fastidieux (mais payant)
      - atomic data types, spin-locking, r/w locks, semaphores

Préemption et concurrence au sein du noyau

se protéger contre les interruptions

- masquer/démasquer les interruptions
- à manipuler avec précaution et sur des sections de code les plus courtes possibles

sinon (et parallèlement)

- exécuter le moins de code possible sous interruption
- si possible différer ce qui peut l'être dans des threads

sans ces précautions : interrupt storm, live locking

Ordonnanceurs

- Time driven
  - L'ordonnanceur détermine quand il est invoqué
    - généralement à intervalles constants (horloge)
    - overhead de l'ordonnancement
- Event driven
  - déclenchement sur des événements
    - E/S (restauration vs context-switch)

FIFO

First In First Out

aka FCFS : First Come First Serve

Basique :

un nouveau job est placé en fin de file

le job à exécuter est retiré en début de file

Avantage :

overhead minimal, l'algorithme est assez trivial

FIFO

Trois jobs arrivent au temps 0

- Job 1 : 0-50
- Job 2 : 50-60
- Job 3 : 60-67

Job	Durée	Arrivée
1	50	0
2	10	0
3	7	0

Métrique temps d'attente moyen

(0+50+60)/3 = 36,6…

Métrique temps d'exécution moyen

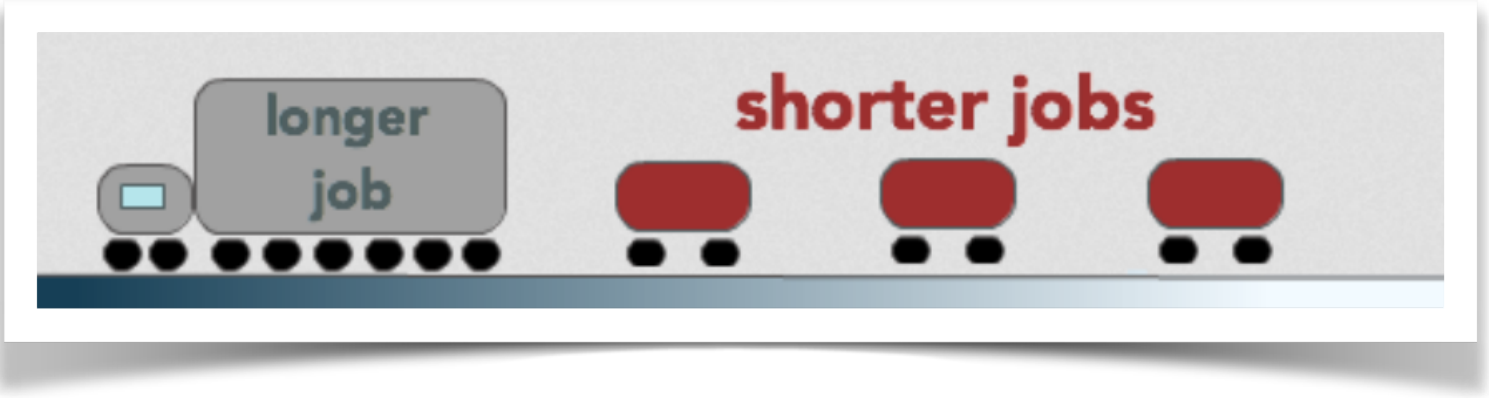
(50 + 50+10 + 60+7)/3 = 59

Pas terrible…surtout rapporté au temps de calcul moyen (50+10+7)/3 = 22,3…

FIFO

Trop d'inconvénients :

- mauvais temps d'attente moyen
- mauvaise gestion des ressources
- effet « convoi »



SJF

Shortest Job First

- on fait passer les tâches les plus courtes avant les autres
  - des résultats optimaux (sous certaines conditions)

SJF

Trois jobs au temps 0

- Job 1 : 17-67
- Job 2 : 7-17
- Job 3 : 0-7

Temps d'attente moyen

- (0+7+17)/3 = 8

Temps d'exécution moyen

- (7 + 17 + 67)/3 = 30

Pas mal!

Job	Durée	Arrivée
1	50	0
2	10	0
3	7	0

SJF

Inconvénient :

- connaître le temps de calcul à l'avance…
  - ce n'est pas irréaliste mais pas toujours possible
  - comment peut-on connaître le temps à l'avance ?
    - et la complexité algorithmique ???
- n'est pas fair, des tâches peuvent s'entendre pour en empêcher d'autres de passer…

Il en existe des variantes : SRTF qui utilise de la préemption

Préemption

Propriété d'un ordonnanceur

- acte d'arrêter l'exécution d'une tâche sans son consentement puis de la reprendre plus tard

Gestion des états courants des exécutions

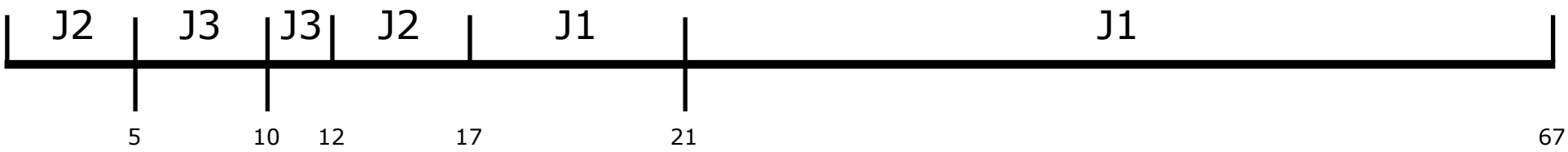
- concept de contexte…
- dans la vie courante : marque-page pour interrompre/reprendre une lecture…

SRTF

Shortest Remaining Time First :

- des préemptions peuvent apparaître à différents moments
  - à la suite de la préemption, le job pour lequel il reste le moins de temps d'exécution est élu

SRTF



Préemptions

Temps d'attente moyen

(0+5+17)/3 = 7,3…

Temps d'exécution moyen

(17 + 11 + 67) = 31,6…

en SJF on aurait

(0+10+17)/3 = 9 et (10+16+67)=31

Job	Durée	Arrivée
1	50	0
2	10	0
3	7	1

Préemption au temps
5
10
21

SRTF

Inconvénient :

- n’est pas fair car des tâches peuvent s’entendre pour empêcher des tâches de passer…

Time slicing

Découpage du temps

Le temps est vu comme une succession d'intervalles d'exécution (quantum/slice) qui sont distribués aux différents processus au cours du temps

C'est un mode de partage du temps

Le time slicing est une technique de partage temporelle de ressources

Ex. : le téléphone, internet…

Time slicing

Permet d'envisager de l'ordonnancement fair

- on peut imaginer obliger chaque tâche à avancer un peu de temps en temps

Tourniquet (Round Robin)

C’est un FIFO avec préemption :

- lorsque le quantum (slice) est épuisé le processus est placé en fin de file et le suivant est pris en tête de la file d’attente
  - équitable mais pas optimal
  - problèmes avec le temps-réel

Tourniquet (Round Robin)

Job	Exécution	Début
1	3	0
2	4	0
3	2	0

Quantum = 1

J1, J2, J3, J1, J2, J3, J1, J2, J2

Quantum = 2

J1, J1, J2, J2, J3, J3, J1, J2, J2

Quantum = 3

J1, J1, J1, J2, J2, J2, J3, J3, J2

Quantum = ∞

FIFO ordinaire!

Tourniquet (Round Robin)

Partage du temps équitable entre processus de même priorité

- partage avec priorité entre processus de priorité différentes…
- pas optimal pour le temps-réel

Le choix du quantum est important

- trop petit : overhead
- trop grand : effet convoi

Une bonne valeur permet

- de faire passer rapidement les petites tâches sans pour autant trop ralentir les longues tâches

Ordonnanceur Unix

L'ordonnanceur traditionnel Unix est un tourniquet avec priorité

- faire passer dans le tourniquet plus fréquemment les processus de plus haute priorité

Une priorité dynamique est calculée à chaque slice

- attention, petite valeur de priorité signifie priorité élevée
  - la valeur peut être vue comme une pénalité…

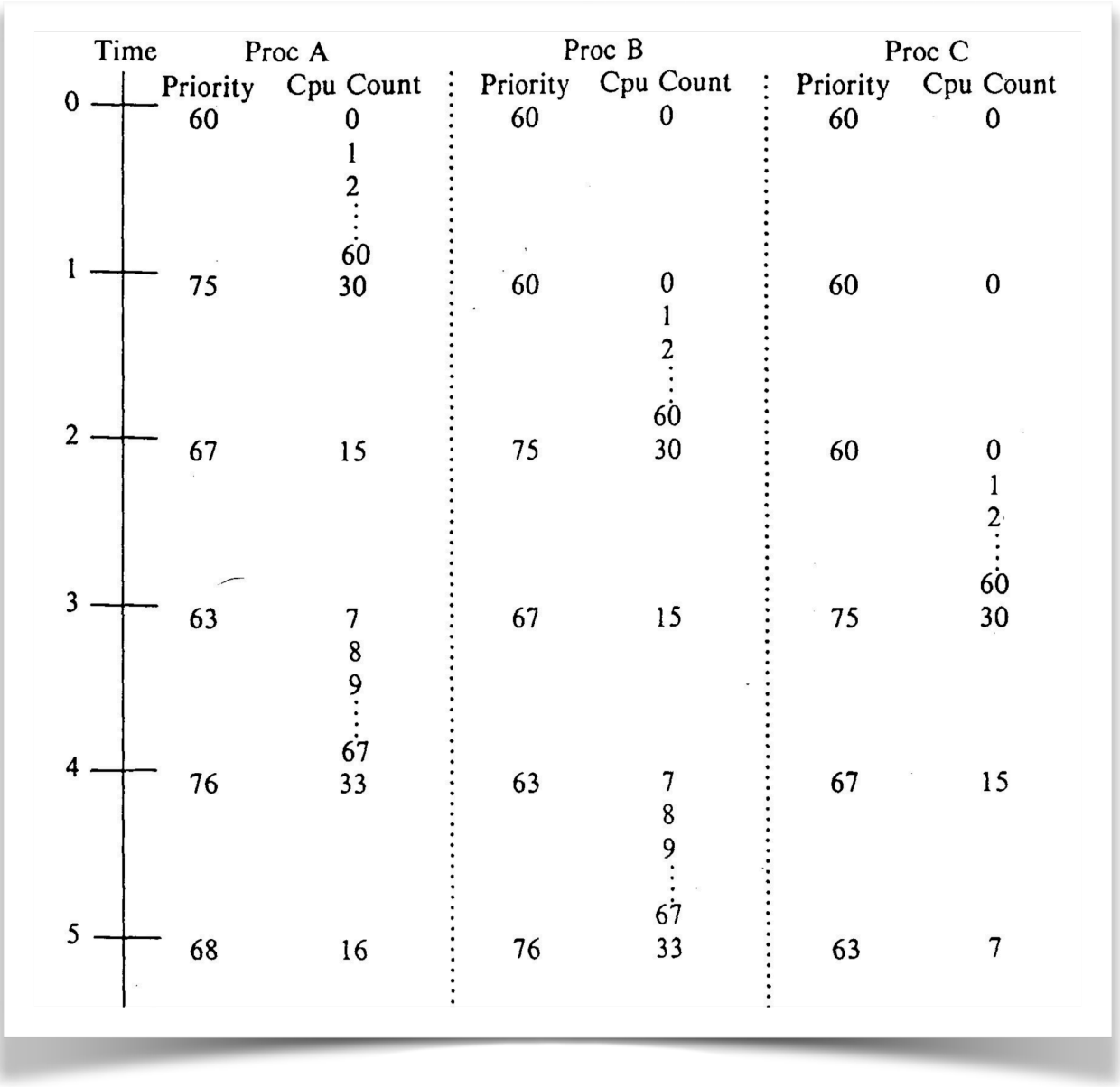
Ordonnanceur Unix

La valeur de la priorité à chaque slice est égale à la priorité de base additionnée de la moitié du temps CPU amorti reçu par le passé

Le temps CPU amorti est augmenté de la durée de CPU reçu si le processus s'exécute

Le temps CPU amorti est divisé par 2 à chaque slice sinon

Ordonnanceur Unix



Time	Proc A		Proc B		Proc C	
	Priority	Cpu Count	Priority	Cpu Count	Priority	Cpu Count
0	60	0	60	0	60	0
		1				
		2				
		⋮				
		60				
1	75	30	60	0	60	0
				1		
				2		
				⋮		
				60		
2	67	15	75	30	60	0
						1
						2
						⋮
						60
3	63	7	67	15	75	30
		8				
		9				
		⋮				
		67				
4	76	33	63	7	67	15
				8		
				9		
				⋮		
				67		
5	68	16	76	33	63	7

Ordonnanceur Unix

12 slices

8 pour P1

2 pour P2

2 pour P3

Slice	Processus 1		Processus 2		Processus 3	
	Priorité	CPU	Priorité	CPU	Priorité	CPU
1	40	0-60	60	0	60	0
2	55	30-90	60	0	60	0
3	62	45	60	0-60	60	0
4	51	22-82	75	30	60	0
5	60	41	67	15	60	0-60
6	50	20-80	63	7	75	30
7	60	40-100	61	3	67	15
8	65	50	60	1-61	63	7
9	52	25-85	75	30	61	3
10	61	42	67	15	60	1-61
11	50	21-81	63	7	75	30
12	60	40-100	61	3	67	15

Ordonnanceur Unix

L'amortissement permet au processus de descendre sa valeur de priorité (augmenter sa priorité)

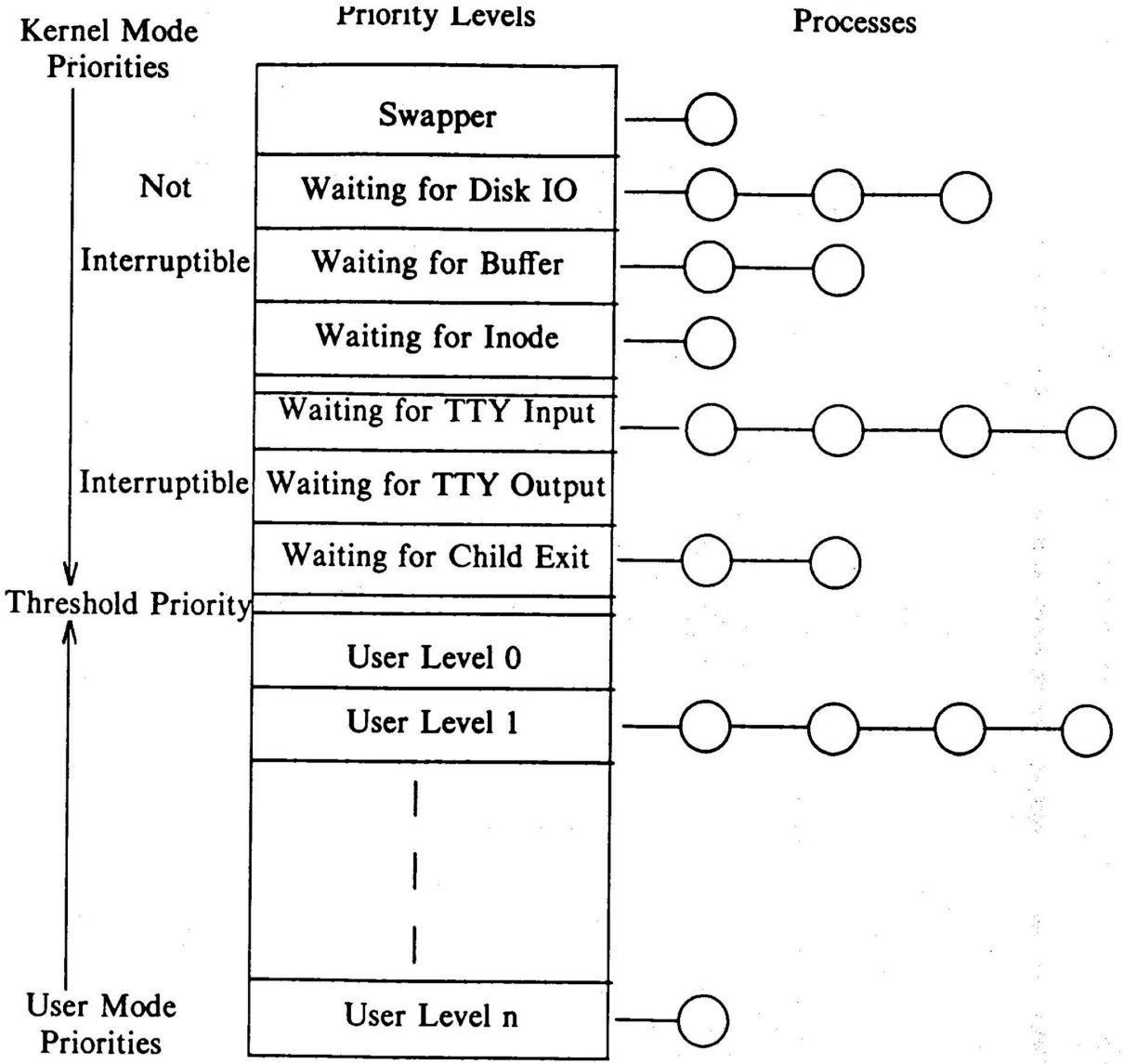
- il change de file d'attente

Les files d'attentes sont gérées comme FIFO

Il existe aussi des files d'attente non ordonnancées

- pour stocker des processus non éligibles qui sont en attente d'évènements (saisie clavier, réponse réseau, etc.)

File d'attentes



Priorité Unix

Le nice Unix est la priorité de base

`nice()`/`setpriority()` permettent de modifier la priorité de la tâche appelante

- on peut diminuer la priorité mais pas l'augmenter
- seuls les tâches avec le privilège adéquat peuvent augmenter leur priorité

Priorité Unix

Il ne faut pas se méprendre

- la priorité est relative, les valeurs absolues ne sont pas pertinentes
  - en particulier, un processus seul s'exécutera à la même vitesse quelque soit sa priorité

Priorité Unix

Le nice Unix est assez peu utilisé (voir jamais)

- à titre personnel, il doit y avoir plus de 20 ans que je ne l'ai pas utilisé
- l'ordonnanceur est suffisamment équitable
- il est exceptionnellement rare de lancer des tâches véritablement longues et pénalisantes

commande `renice`

Round Robin

Avantage :

équitable

Inconvénient :

pas optimal pour les tâches temps-réel

Earliest Deadline First

Quantum = 1

J1, J1, J1, J1, J2, J2, J2, J2, J3x10, J1x6

Quand J2 arrive il reste 29 pour J1, J2 préempte

Quand J3 arrive il reste 28 et 27 pour J1, J2 on continue avec J2…

Quand J2 termine il reste 25 pour J1 et 24 pour J3…

Job	Arrivée	Exécution	Contrainte
1	0	10	33
2	4	3	28
3	5	10	27

Earliest Deadline First

Intéressant car optimal mais…

- en cas de surcharge, effet avalanche : tout le monde rate sa deadline!
- Que se passe t-il si les threads mentent sur leur deadline ?
- il faut donc
  - basculer vers un autre ordonnanceur
  - éliminer des tâches
  - dégrader le service

Earliest Deadline First

Attributs des threads à leur création

Ancien nom

SCHED_EDF

Nouveau nom

SCHED_DEADLINE

sched.h

```
struct sched_attr {
    u32 size;

    u32 sched_policy;
    u64 sched_flags;

    /* SCHED_NORMAL, SCHED_BATCH */
    s32 sched_nice;

    /* SCHED_FIFO, SCHED_RR */
    u32 sched_priority;

    /* SCHED_DEADLINE */
    u64 sched_runtime;
    u64 sched_deadline;
    u64 sched_period;
};
```

L'inversion de priorité

- La gestion des priorités n'est aussi simple
  - il est possible dans le cadre de la concurrence d'obtenir des inversions de priorité
    - c'est dans certains cas particulièrement embêtant
      - Ex. : le processus de commande d'arrêt du train se fait voler la priorité par le processus de gestion du chauffage

L'inversion de priorité

A (priorité élevée)

B (faible priorité)

A attend le verrou X et relâche le processeur

B est élu et prend le verrou Y

X est disponible, A est élu, prend X, attend Y et relâche le processeur

B peut utiliser le processeur…

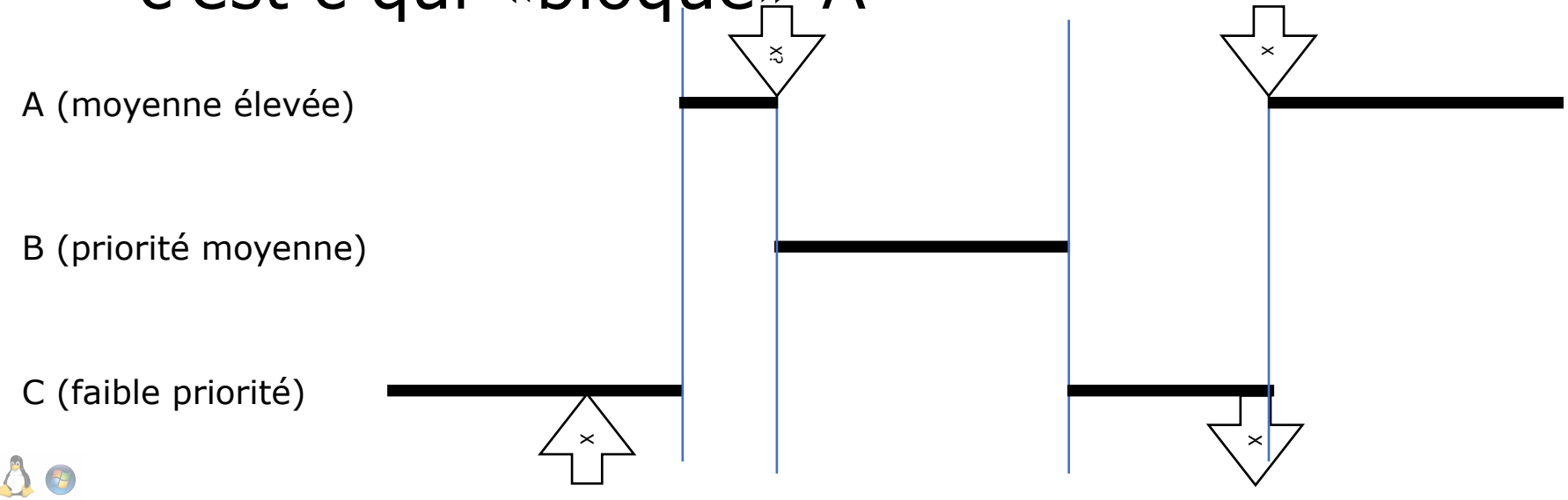
Y a permis à B de voler la priorité à A

L'inversion de priorité

Le processus de priorité la plus élevée passe en dernier

on peut rajouter des tas de processus intermédiaires qui seront élus…

c'est c qui «bloque» A



L'héritage de priorité

Pour résoudre (en partie) le problème on peut temporairement attribuer à C la priorité de A

- afin qu'il libère le verrou au plus vite…

L'ordonnanceur devient assez complexe à mettre en œuvre

- il n'existe pas de solution simple pour résoudre le problème définitivement…

Random boosting

- Technique utilisée par Windows
  - augmenter violemment et de façon aléatoire la priorité des tâches en section critique
    - ainsi on espère qu'elles ne garderont pas trop longtemps leurs verrous

Unix temps-réel

IEEE 1003.1b-1993 POSIX Realtime extension

- VxWorks
- QNX
- μC/OS
- LynxOS
- Linux-rt
- RTLinux

Priorités temps-réel

Linux offre deux classes de tâches

- en temps partagé
  - SCHED_IDLE
  - SCHED_BATCH
  - SCHED_NORMAL
- en temps réel
  - SCHED_FIFO
  - SCHED_RR
  - SCHED_DEADLINE (>3.14)

Priorités temps-réel

En FIFO une tâche rend la main

- d'elle-même `sched_yield()` , `pthread_yield()`
- accès à une ressource
- préempté par tâche RT de plus haute priorité

En RR une tâche rend la main

- d'elle-même `sched_yield()` , `pthread_yield()`
- accès à une ressource
- préempté par tâche RT de plus haute priorité
- fin du quantum

Priorités temps-réel

- Il faut être avoir des privilèges particuliers pour créer une tâche temps-réel
  - pourquoi ?
    - monopolise la ressource processeur!
      - l'interactivité de la machine est réduite à néant ou presque
    - une solution consiste à sortir du mode temps-réel à intervalles réguliers pour une durée déterminée
      - mais on est plus vraiment dans le temps réel…
        - implanté dans les dernières version de Linux

Linux et TR

Linux est-il temps-réel ?

est-il adapté pour permettre à des applications avec contraintes TR de s'exécuter ?

Linux, Préemption

- Possibilité de permettre à une tâche prioritaire de démarrer immédiatement son exécution
- A l'opposé, imposer d'attendre que l'exécution en cours atteigne un point où l'on peut procéder à l'ordonnancement
- Linux 2.4:
  - Peu de points de préemption... Exécution en mode système ressemble à une « grosse section critique », ordonnancement lors du retour vers le mode utilisateur

Linux 2.6

- Incorporation de certaines des modifications contenue par le patch PREEMPT-RT
  - Ordonnanceur O(1)
  - Amélioration des points d'ordonnancement internes
  - High-Precision Timers
- Reste entre autres:
  - Traitement des interruptions

Améliorer la Préemption

- Minimiser:
  - La durée des sections critiques (pendant lesquelles les interruption sont masquées par le code de base)
    - Réduit la latence de prise en compte
  - Le traitement effectué sous interruption
    - Pendant que l'on traite une interruption, soit toutes les interruptions sont masquées, soit l'interruption que l'on traite est masquée.
    - Ex: si le traitement d'un interruption horloge prend plus d'un « tick », le système ne sera pas à l'heure et inutilisable.
- Permettre un ordonnancement en retour d'interruption

Linux interruptions

- SoftIRQ:
  - Faire le minimum dans l'ISR (Interupt Service Routine)
  - Faire le travail réel et « long » après avoir acquitté l'interruption, avant de reprendre le travail interrompu
  - Possiblement faire faire ce travail par une thread dédiée `ksoftirqd`, haute priorité
- Tasklet
  - On peut avoir plusieurs softIRQ simultanément (1/CPU)
  - On a une seule tasklet (pour un IRQ donnée), CPU variable

Linux, PREEMPT-RT

- Permettre à des threads définies par l'utilisateur de s'exécuter à des priorités plus élevées que le traitement des interruptions.
- Crée un thread « noyau » par niveau d'interruption.
- L'administrateur système peut ajuster les priorités.
- Augmente la charge du système: passage obligatoire par des threads, coût d'ordonnancement.

Linux, PREEMPT-RT

- Attention! Un marteau est un très bon outil, à condition de savoir s'en servir et de ne pas se taper sur les doigts!
- Le noyau 2.4 n'est pas préemptif,
  - Pas possible de tourner de véritable apps TR dures
- Certaines des modifications ont été incorporées au noyau 2.6
  - Reste toujours la nécessité d'un patch suivant les besoins applicatifs

Linux 2.6

- Ordonnanceur en O(1)
  - Remplacé par CFS depuis 2.6.23
- Offre 3 modes de préemption (config)
  - Defaut (pas de préemption)
  - Préemption Volontaire
  - Préemption « involontaire »
  - Encore insuffisant (y compris pour audio)
- 2.6.18 : héritage de priorité
- 2.26.21: High-resolution timers (Posix) ~1µ sec.

Patch PREEMPT-RT

- Introduit un niveau de préemption « complète »
- Permet des latences encore plus courtes ~100µsec
- Impact global sur performances
  - On passe plus temps en changement de contexte
- Problèmes:
  - Pas dans l'arbre référence (résistance!)
  - Pilotes de périphériques à adapter (quid si binaire?)

Linux 3.14

Temp-réel

Ordonnanceur EDF avec `SCHED_DEADLINE`

Toutefois le manuel indique

```
0. WARNING
==========


 Fiddling with these settings can result in an unpredictable or even unstable
 system behavior. As for -rt (group) scheduling, it is assumed that root users
 know what they're doing.
```

Ordonnancement multi-processeur

Répartir la charge en processeurs

Migration des threads ?

- faire suivre les données en cache…
- défaut de cache donc exécution ralentie!

Concept d'affinité (processor affinity/CPU pinning)

- lier un thread à un processeur
- lier un groupe de thread à un processeur

Affinité

BSD

`pthread_setaffinity_np`

Linux

`sched_setaffinity`

macOS

`thread_policy_set`

Ordonnancement multi-processeur

- Et la file des processus ?
  - file unique ?
    - prendre des précautions, section critique donc goulot d'étranglement
- En pratique
  - mélange de files par processeur (short-term scheduling) et file commune (long-term scheduling)

Gestion de fichiers

Gestion de fichiers

Besoins :

- petits fichiers (texte)
  - édition
- moyen (audio)
  - lecture séquentielle
- gros (vidéo)
  - lecture séquentielle
- gros (base de données)
  - lecture/écriture *aléatoire*

Gestion de fichiers

Besoins :

- le fichier comme container de données
  - propriété, droit d'accès
  - mode d'accès
    - lecture, écriture, ajout
- tolérance aux pannes
  - RAID
- sauvegarde/restauration
  - Apple time machine

Gestion de fichiers

Besoins :

- le fichier comme objet de communication
  - Unix pipes
- alias
  - Unix links, Windows shorcuts, Apple alias
- le fichier comme mode d’accès
  - périphériques, mémoire, processus
  - Linux procfs

Fichiers spéciaux (Unix)

Disques :

`/dev/sd* /dev/disk*`

Terminaux :

`/dev/tty* /dev/pty* /dev/pts/*`

Souris

`/dev/input/mouse*`

Source aléatoire

`/dev/random /dev/urandom`

Puit sans fond

`/dev/null`

Source nulle

`/dev/zero`

Mémoire

`/dev/ram* /dev/mem*`

Volumes périphériques (Windows)

Vus comme des périphériques :

- `CON:` console
- `AUX:` auxiliaire
- `LST:` imprimante
- `PRN:` imprimante qui numérote les lignes
- `NUL:` puit sans fond
- `EOF:` source infinie de caractère EOF (0x1A)
- `INP:` entrée
- `OUT:` sortie

ne sont plus utilisés depuis longtemps…

Gestion de fichiers

Besoins :

- plusieurs espaces de stockage sur un disque physique
- une seul espace de stockage sur plusieurs disques
- des espaces de stockages dynamiques

Gestion de fichiers

Besoins (récents) :

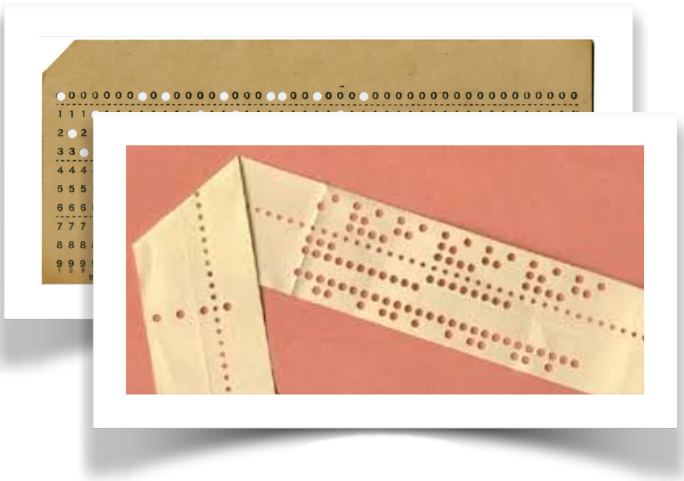
- journalisation
- indexation
- cryptage

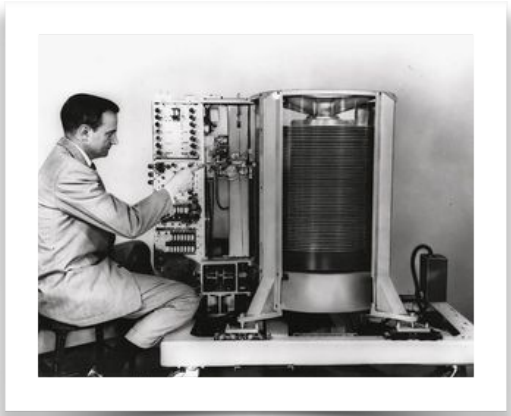
Gestion de fichiers

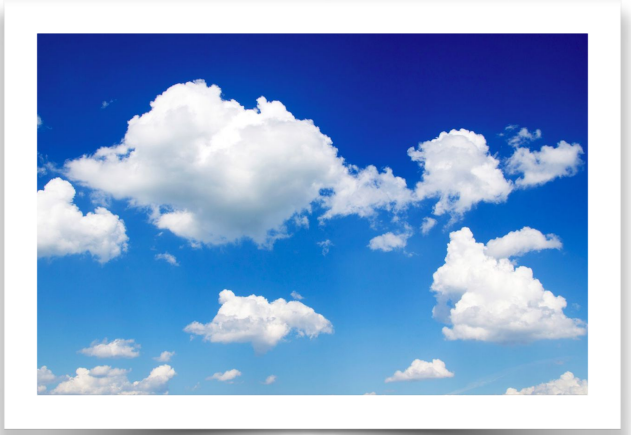
Des filesystems en pagaille :

- historiques : ufs, fat, hfs…
- plus récents : ext4, apfs, ntfs, xfs, zfs…
- réseau : nfs 3, nfs 4, SMB, Lustre…
- flash/SSD : ffs2, jffs, TrueFFS
- exotiques : procfs, sysfs

Technologies







Stockage

Lakers vs Spurs ? HDD vs SSD ?

	HDD	SSD
Énergie	6/7W	2/3W Gain 30mn batterie
Coût	0.03€/Go	0.2€/Go
Capacité	4-8To	1-2To
Taux de panne	1%/an	0.1%/an
Vitesse écriture	100Mo/s	500Mo/s
Sensibilité environnement	magnétisme, mécanique	N/A

Technologie

Auparavant séquence de données

À partir des disques durs

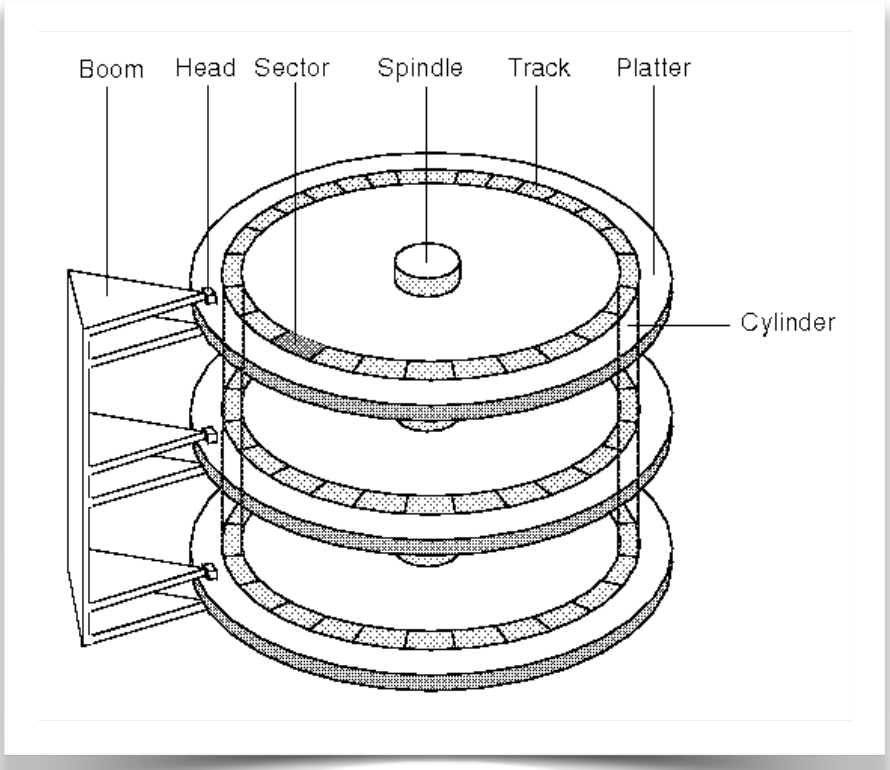
- Concept de bloc de données
  - Y compris flash et ssd!

Les filesystems cherchent à réintroduire de la linéarité (fichier=séquence de données) où il n'y en a pas

- souvenez-vous de la pagination…
  - page/bloc

Disque

- Secteur
  - Unité de lecture/écriture
    - Amortissement du temps d'accès
  - Taille courante
    - 512o HDD
    - 2048o CDROM
    - 4096 HDD récents



Problème

lire un octet ?

$\Longrightarrow$ lire un secteur entier

écrire un octet

$\Longrightarrow$ lire un secteur, modifier, réécrire

Problème

- l'accès est non constant
  - se déplacer vers une donnée précise ne prend pas un temps constant selon l'endroit d'où l'on part
    - tête des disques durs…inertie, distance…

Utilisation de l'espace

Partionnement

Agrégation

Partitionnement

Découpage logique de l'espace physique

- pour faire apparaître un disque comme plusieurs
  - comment le découper ?
    - secteurs/cylindres/plateaux, logique

Pourquoi ?

- isolation des données/robustesse
- facilité de gestion/sauvegarde

Partitionnement

- Le partitionnement de bas-niveau
  - Historiquement décrit dans le MBR Master Boot Record
    - secteur d'initialisation qui contient l'amorce et la table primaire de partition
      - 4 partitions maximum et pas plus de 2To
  - Plus récent GPT GUID Partition Table
    - GPT est un sous-ensemble de UEFI

Partionnement

- MBR
  - raisonne sur cylindre-tête-secteur (CHS cylinder-head-sector)
- GPT
  - raisonne sur LBA logical block addressing
  - idée s'affranchir des CHS
    - il existe des conversions de CHS à LBA

Description d'une partition

GUID partition entry format

Offset	Length	Contents
0 (0x00)	16 bytes	Partition type GUID
16 (0x10)	16 bytes	Unique partition GUID
32 (0x20)	8 bytes	First LBA (little endian)
40 (0x28)	8 bytes	Last LBA (inclusive, usually odd)
48 (0x30)	8 bytes	Attribute flags (e.g. bit 60 denotes read-only)
56 (0x38)	72 bytes	Partition name (36 UTF-16LE code units)

Description d'une partition

Windows	Microsoft Reserved Partition (MSR)	E3C9E316-0B5C-4DB8-817D-F92DF00215AE
	Basic data partition[g]	EBD0A0A2-B9E5-4433-87C0-68B6B72699C7
	Logical Disk Manager (LDM) metadata partition	5808C8AA-7E8F-42E0-85D2-E1E90434CFB3
	Logical Disk Manager data partition	AF9B60A0-1431-4F62-BC68-3311714A69AD
	Windows Recovery Environment	DE94BBA4-06D1-4D40-A16A-BFD50179D6AC
	IBM General Parallel File System (GPFS) partition	37AFFC90-EF7D-4E96-91C3-2D7AE055B174
	Storage Spaces partition	E75CAF8F-F680-4CEE-AFA3-B001E56EFC2D
Linux	Root partition (32-bit ARM)[32]	69DAD710-2CE4-4E3C-B16C-21A1D49ABED3
	Root partition (64-bit ARM/AArch64)[32]	B921B045-1DF0-41C3-AF44-4C6F280D3FAE
	Data partition[33]	A2A0D0EB-E5B9-3344-87C0-68B6B72699C7
		AF3DC60F-8384-7247-8E79-3D69D8477DE4
	Swap partition	0657FD6D-A4AB-43C4-84E5-0933C84B4F4F
	Logical Volume Manager (LVM) partition	E6D6D379-F507-44C2-A23C-238F2A3DF928
	`/home` partition[32]	933AC7E1-2EB4-4F13-B844-0E14E2AEF915
	`/srv` (server data) partition[32]	3B8F8425-20E0-4F3B-907F-1A25A76F98E8
macOS Darwin	ZFS[h]	6A898CC3-1DD2-11B2-99A6-080020736631
	Apple RAID partition	52414944-0000-11AA-AA11-00306543ECAC
	Apple RAID partition, offline	52414944-5F4F-11AA-AA11-00306543ECAC
	Apple Boot partition (Recovery HD)	426F6F74-0000-11AA-AA11-00306543ECAC
	Apple Label	4C616265-6C00-11AA-AA11-00306543ECAC
	Apple TV Recovery partition	5265636F-7665-11AA-AA11-00306543ECAC
	Apple Core Storage (i.e. Lion FileVault) partition	53746F72-6167-11AA-AA11-00306543ECAC

Description d'une partition

macOS

- `diskutil`

freeBSD

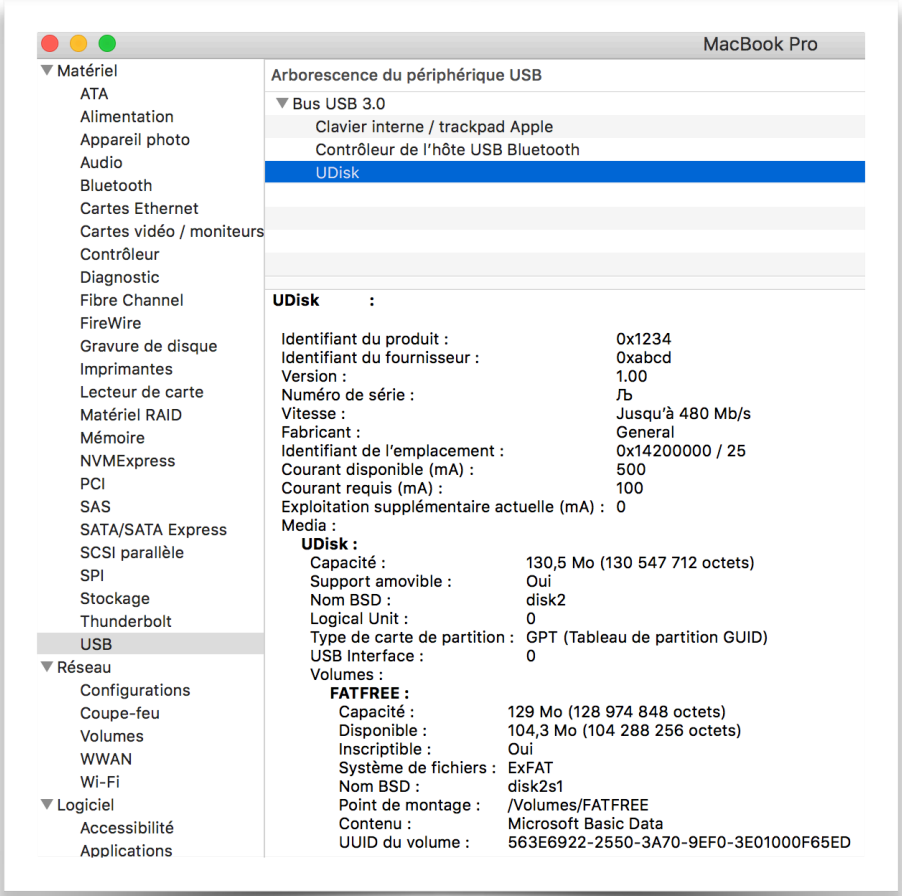
- `gpt`

Linux

- `blkid`, `lsblk`

Description d'une partition

Examen d'une clé USB




```
[Estafette:/tmp] diskutil info disk2s1
   Device Identifier:        disk2s1
   Device Node:              /dev/disk2s1
   Whole:                    No
   Part of Whole:            disk2

   Volume Name:              FATFREE
   Mounted:                  Yes
   Mount Point:              /Volumes/FATFREE

   Partition Type:           Microsoft Basic Data
   File System Personality:  ExFAT
   Type (Bundle):            exfat
   Name (User Visible):      ExFAT

   OS Can Be Installed:      No
   Media Type:               Generic
   Protocol:                 USB
   SMART Status:             Not Supported
   Volume UUID:              563E6922-2550-3A70-9EF0-3E01000F65ED
   Disk / Partition UUID:    316E5925-40E5-4A42-A314-CF5729019D1E
   Partition Offset:         1048576 Bytes (2048 512-Byte-Device-Bl

   Disk Size:                129.0 MB (128974848 Bytes) (exactly 25
   Device Block Size:        512 Bytes

   Volume Total Space:       128.8 MB (128778240 Bytes) (exactly 25
   Volume Used Space:        24.0 MB (23998464 Bytes) (exactly 4687
   Volume Free Space:        104.8 MB (104779776 Bytes) (exactly 20
   Allocation Block Size:    4096 Bytes

   Read-Only Media:          No
   Read-Only Volume:         No

   Device Location:          External
   Removable Media:          Removable
   Media Removal:            Software-Activated
```

Agrégation

«Inverse» de la partition, plusieurs espaces vus comme un seul

- agrégation physique, logique
  - on peut ensuite partitioner, etc

Pourquoi ?

- Obtenir un grand espace à l'aide de petits

LVM Logical Volume Management

- ZFS, Btrfs, VxVM (Veritas), LDM (MS), FusionDrive (Apple)

Agrégation

Linux

lvm Logical Volume Manager

Tolérance aux pannes

- Pour résister aux pannes on utilise de la redondance
  - une cinquième roue pour pallier aux crevaisons dans une voiture
    - pour les disques, idem
- Attention, un cinquième roue ne suffit pas toujours…
  - deux crevaisons simultanées, successives…
- Probabilité/statistique
- Modèle de panne

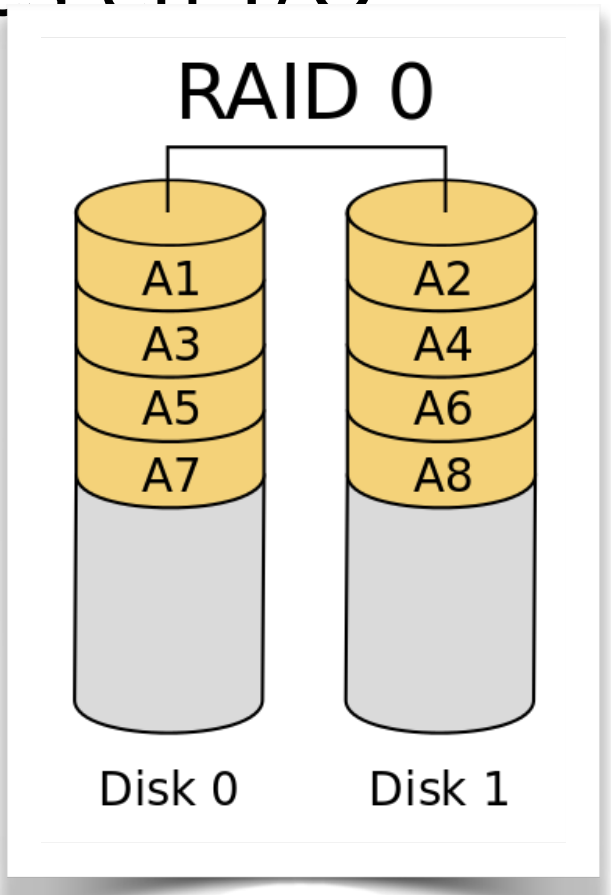
Tolérance aux pannes

- RAID Redundancy of Inexpensive Array Disk
  - l'idée originale est d'agréger des disques afin d'obtenir des avantages divers
    - pas forcément la tolérance aux pannes…
- On parle de virtualisation de stockage

RAID 0

Ce niveau n'offre pas de tolérance aux pannes, il permet d'obtenir des gains intéressants de performances en I/O

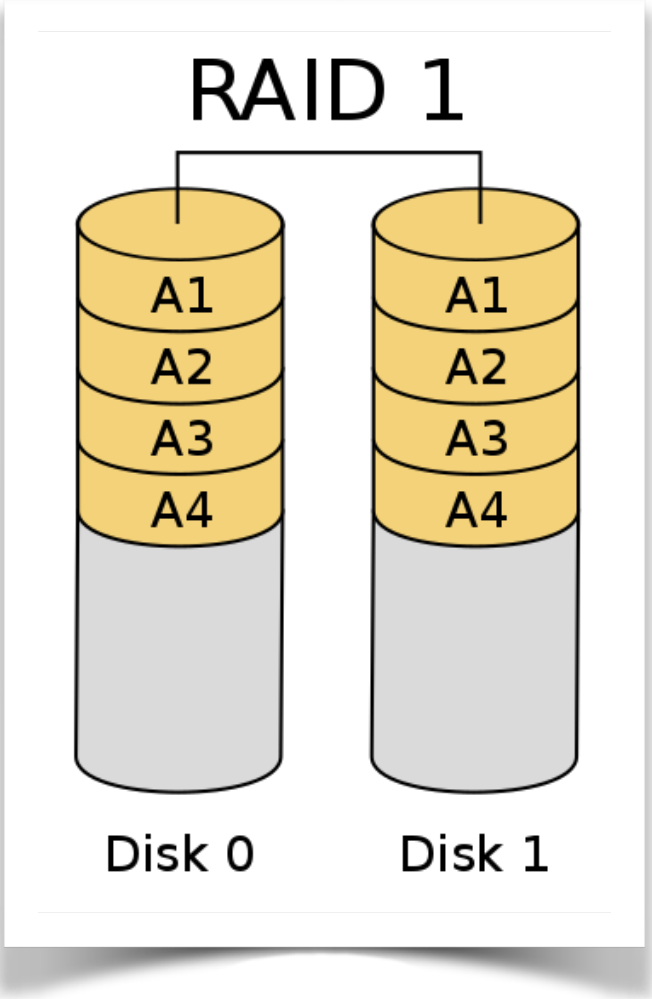
- parallélisme…



RAID 1

La redondance de secours…

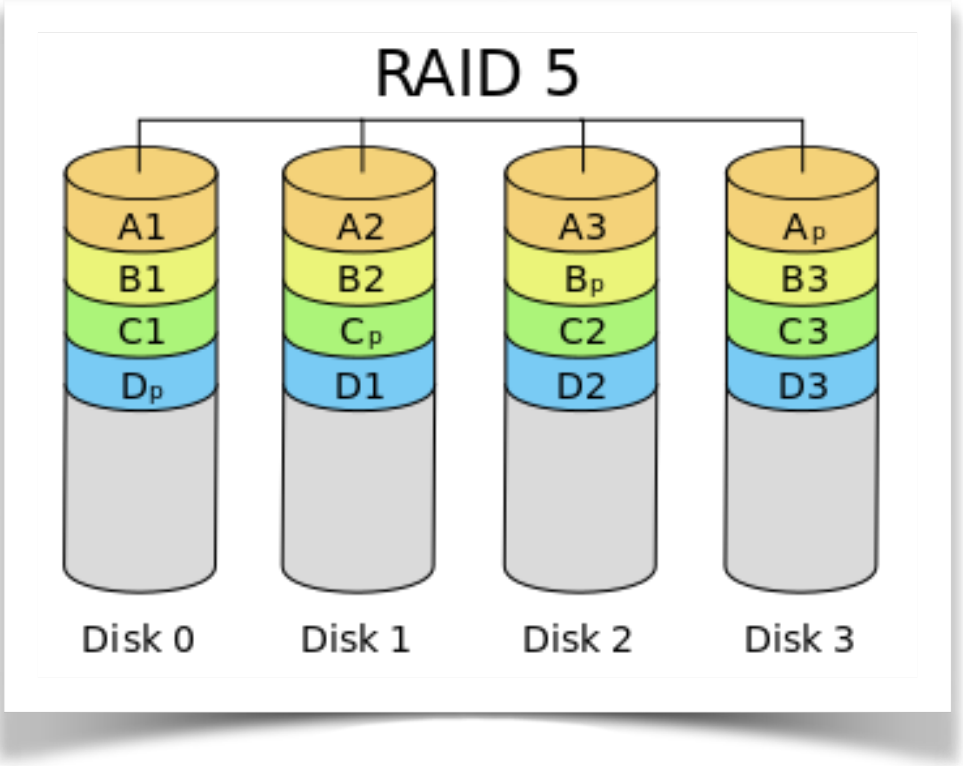
aucun gain d’espace!



RAID 5

L'agrégation répartie du RAID 0 avec le contrôle d'erreur

$$X_p = X_1+X_2+X_3+\ldots+X_n$$



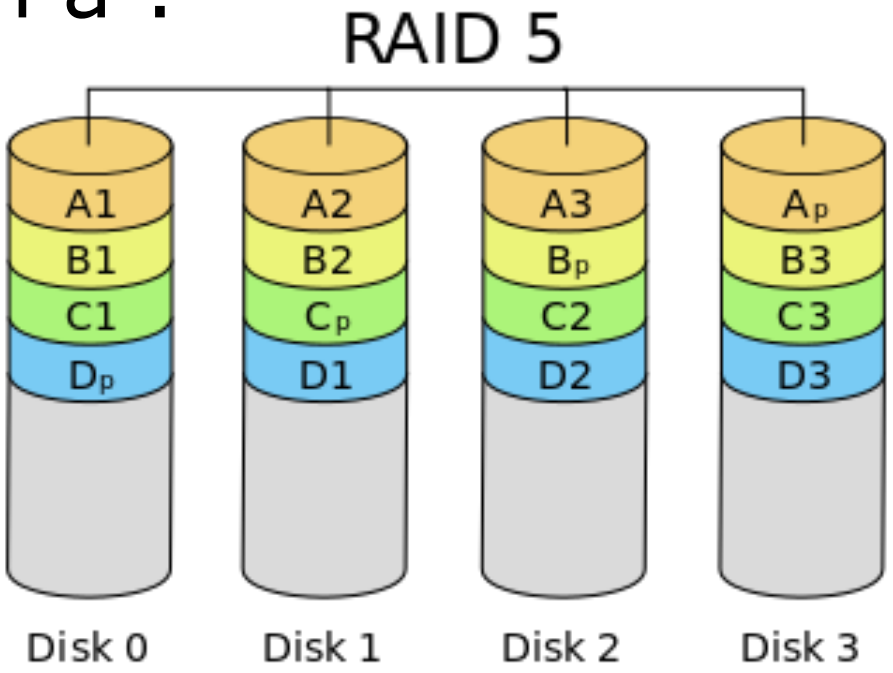
RAID 5

Crash du disk 2 ?

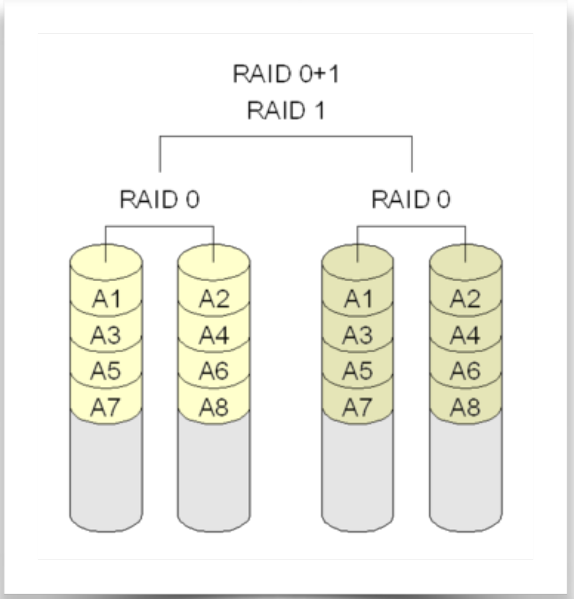
$A_p = A_1+A_2+A_3$?

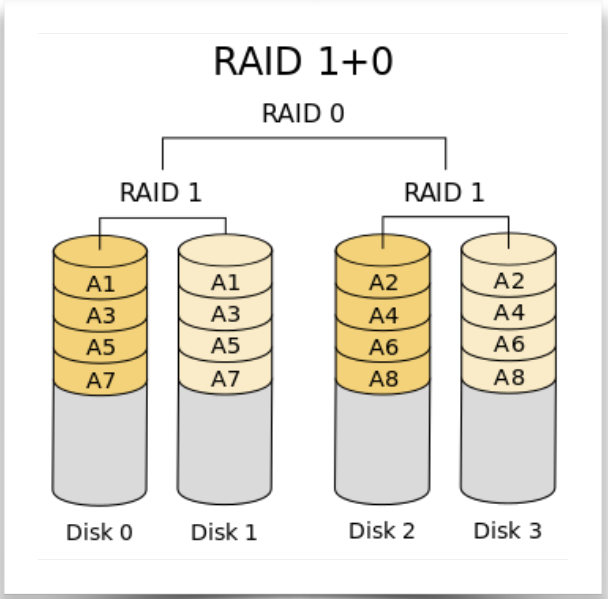
Comme X+X = 0 on a :

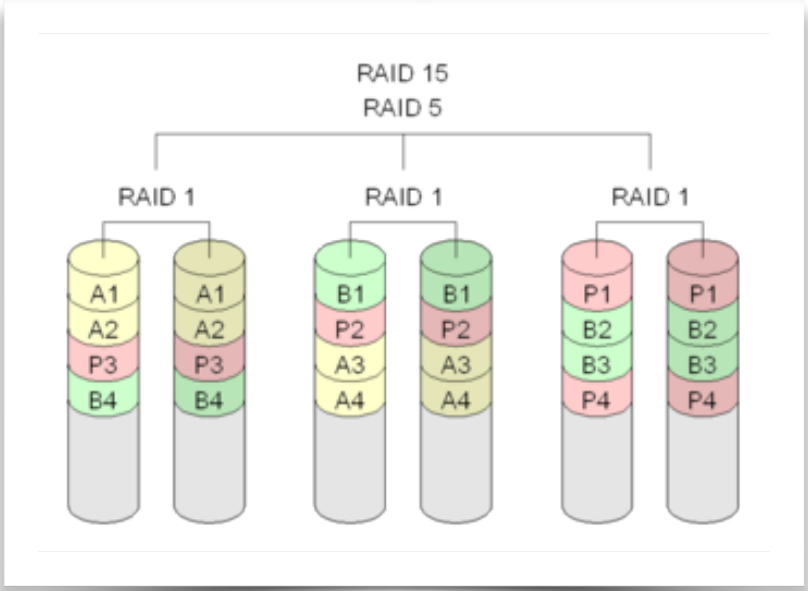
$A_3 = A_P+A_1+A_2$

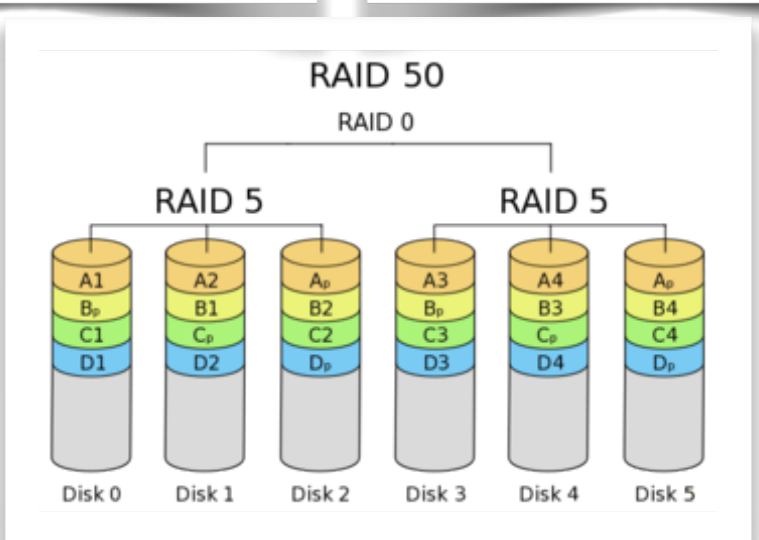


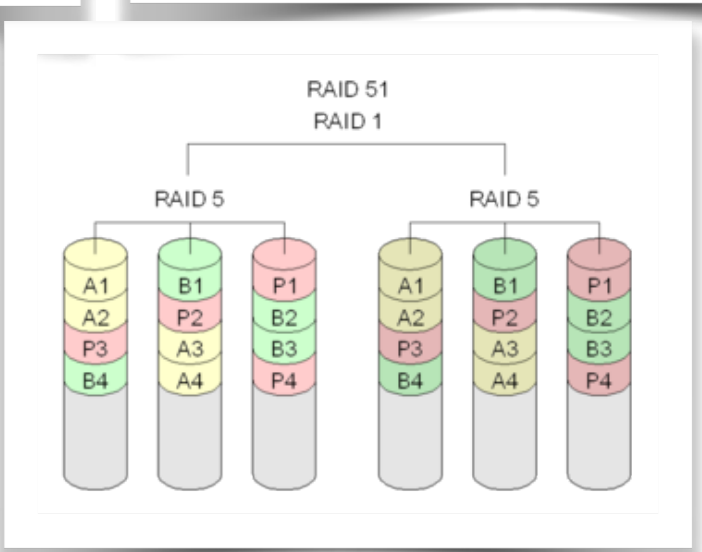
RAID combinés









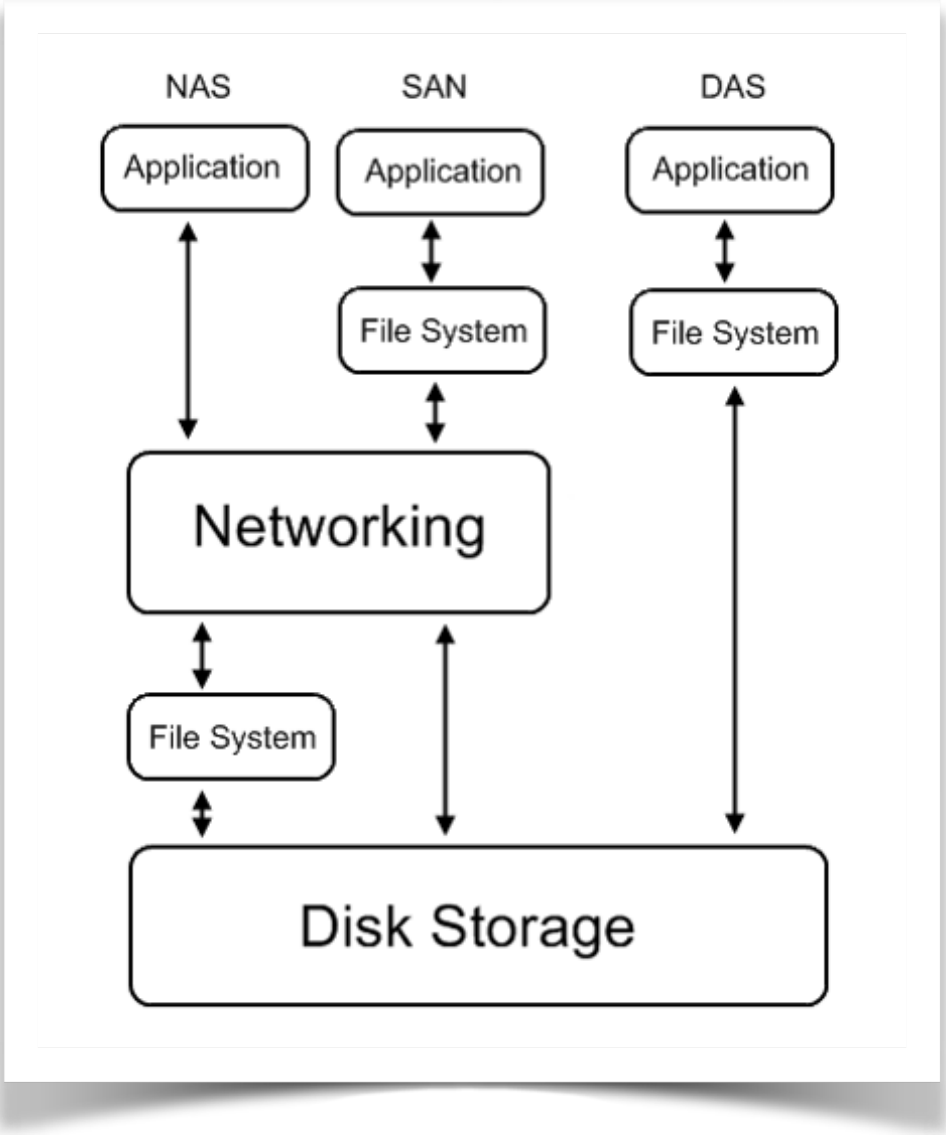


Raid

Les pièges

- plus de disque ☛ plus de pannes!
- grappe de disque identiques! Donc même modèle, même âge, etc.
- ne pas négliger la sauvegarde, même avec du RAID!

DAS, SAN, NAS…


NAS
SAN
DAS
Application
Application
Application
File System
File System
Networking
File System
Disk Storage

Vision utilisateur

Historiquement, aucune structure

Depuis Multics, vision hiérarchique

- macOS
  - folders and documents
- Windows, Unix
  - directories and files

Pour macOS certains documents sont constitués de plusieurs fichiers…

- Bundles

Windows, Unix sont plus terre-à-terre

Vision utilisateur

Problème :

- nommage
  - filename
- désignation
  - filepath

Arborescence

Unix :

vision unifiée, un seul arbre de répertoires et fichiers

Windows :

plusieurs arbres (A:, B:, C:, etc) chacun contenant des répertoires et fichiers

macOS :

au niveau Unix, vision Unix unifiée

au niveau macOS, vision «Appareils» pour représenter les dispositifs physiques clairement séparés (clé USB, disque externe, etc)

Arborescence

- Une partition contient une arborescence autonome
  - qu'on la voit séparée des autres ou non est un autre problème
  - le principe est toujours le même
    - répertoire : fichier particulier désignant d'autres fichiers
    - fichier : espace contenant des données
      - le répertoire est donc un fichier contenant des données particulières pour le système
        - c'est un objet sensible! De sa cohérence dépend la cohérence de l'arborescence…

Vision Unix

Il n'y a donc ultimement que des conteneurs de données

- Unix appèle cela un inœud (inode)
- une partition contient donc des inœuds

L'espace des inœuds est plat

- ce sont les répertoires qui en donne une vision hiérarchique

Vision Unix

- Un inœud
  - des méta données
    - le descriptif du contenu
      - taille, type
    - des propriétés associées au contenu
      - droits d'accès, dates diverses
  - de quoi accéder aux données (le contenu)
    - retrouver la localisation des données

Vision Système

SGF Système de Gestion de Fichiers

- organiser le stockage physique
- fournir les services utilisateurs
  - open, read, write, close

Ext2, Ex3, Ext4

Ext4

système de fichiers installé par défaut dans les distributions récentes de Linux (fragmentation)

Ext3

extension de Ext2 (journalisation)

Ext2 (Rémy Card, UMPC)

système de fichier historique de Linux (Ext est une version préliminaire)

largement inspiré de FFS, Fast File System (le FS de BSD)

Ext2

Les idées Unixiennes

- les fichiers sont représentés par des inœuds
- les répertoires sont des inœuds particuliers contenant une liste d'entrées désignant d'autres inœuds
- tous les périphériques sont représentés par des inœuds (vision unifiée)
  - lire le clavier c'est comme lire un fichier
  - écrire à l'écran c'est comme écrire dans un fichier

Ext2

À la construction de la partition (`mkfs` ?) on choisit la taille d'un bloc

- bloc unité minimale manipulée par le système de gestion de fichier
  - à ne pas confondre avec le secteur
    - secteur : unité minimale du dispositif physique
    - bloc : unité minimale du dispositif logique

Ext2

`mke2fs -b block-size`

1Ko, 2Ko, 4Ko, 8Ko sont les valeurs possibles les plus communes

petite valeur : meilleur utilisation de l'espace mais performances moindres

grande valeur : moins bonne utilisation de l'espace (fragmentation) mais meilleures performances

Ext2

`mke2fs -N number-of-inodes`

- nombre de fichiers maximum que la partition supportera

petite valeur : peu de gros fichiers

grande valeur : beaucoup de petits fichiers

Déterminer cette valeur ?

- sauf expertise, une valeur par défaut est calculée

Ext2

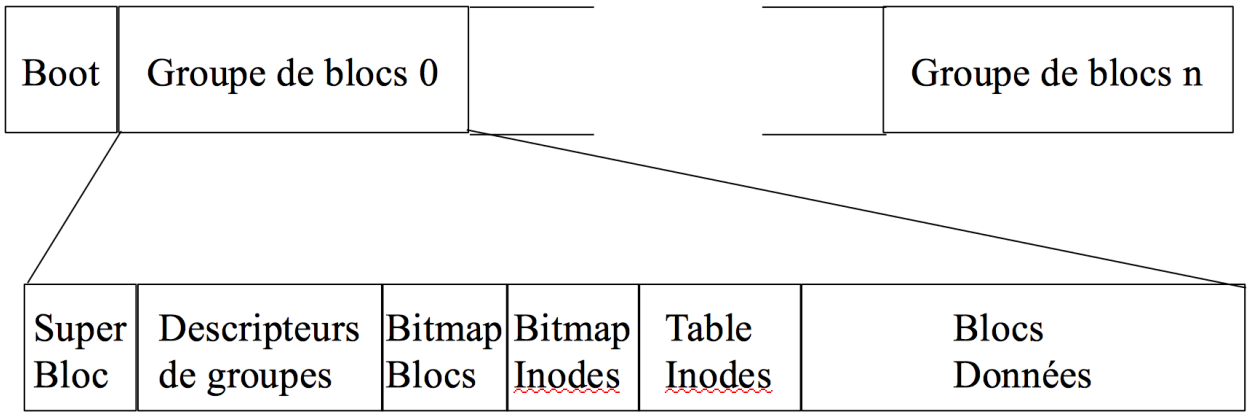
df

```
parallels@ubuntu:~$ df -i
Sys. de fichiers  Inœuds IUtil.  ILibre IUti% Monté sur
udev                121520    514  121006    1% /dev
tmpfs               126528    696  125832    1% /run
/dev/sda1          4128768 193300 3935468    5% /
tmpfs               126528      9  126519    1% /dev/shm
tmpfs               126528      6  126522    1% /run/lock
tmpfs               126528     16  126512    1% /sys/fs/cgroup
```

```
parallels@ubuntu:~$ df -k
Sys. de fichiers blocs de 1K   Utilisé Disponible Uti% Monté sur
udev                   486080         0    486080   0% /dev
tmpfs                  101224      6344     94880   7% /run
/dev/sda1            64891708   4391480  57180884   8% /
tmpfs                  506112       212    505900   1% /dev/shm
tmpfs                    5120         4      5116   1% /run/lock
tmpfs                  506112         0    506112   0% /sys/fs/cgroup
```

Ext2

- Partition Ext2
  - Boot block
  - Superblock
  - Table des groupes de blocs
  - Groupes de blocs



- Pour des raisons de sécurité
  - chaque groupe de bloc contient une copie du superbloc et de la table des groupes!

Ext2

Logiquement

Physiquement

Ext2

Superblock

ext2.h (4.18.14)

```
struct ext2_super_block {
     __le32    s_inodes_count;     /* Inodes count */
     __le32    s_blocks_count;     /* Blocks count */
     __le32    s_r_blocks_count;   /* Reserved blocks count */
     __le32    s_free_blocks_count;/* Free blocks count */
     __le32    s_free_inodes_count;/* Free inodes count */
     __le32    s_first_data_block; /* First Data Block */
     __le32    s_log_block_size;   /* Block size */
     __le32    s_log_frag_size;    /* Fragment size */
     __le32    s_blocks_per_group; /* # Blocks per group */
     __le32    s_frags_per_group;  /* # Fragments per group */
     __le32    s_inodes_per_group; /* # Inodes per group */
     __le32    s_mtime;            /* Mount time */
     __le32    s_wtime;            /* Write time */
     __le16    s_mnt_count;        /* Mount count */
     __le16    s_max_mnt_count;    /* Maximal mount count */
     __le16    s_magic;            /* Magic signature */
     __le16    s_state;            /* File system state */
     __le16    s_errors;           /* Behaviour when detecting errors */
     __le16    s_minor_rev_level;  /* minor revision level */
     __le32    s_lastcheck;        /* time of last check */
     __le32    s_checkinterval;    /* max. time between checks */
     __le32    s_creator_os;       /* OS */
     __le32    s_rev_level;        /* Revision level */
     __le16    s_def_resuid;       /* Default uid for reserved blocks */
     __le16    s_def_resgid;       /* Default gid for reserved blocks */
     __le32    s_first_ino;        /* First non-reserved inode */
     __le16   s_inode_size;        /* size of inode structure */
```

Ext2

La table de description des groupes est constituée d'entrées

- Descripteur de groupe

ext2.h (4.18.14)

```
struct ext2_group_desc
{
    __le32  bg_block_bitmap;        /* Blocks bitmap block */
    __le32  bg_inode_bitmap;        /* Inodes bitmap block */
    __le32  bg_inode_table;     /* Inodes table block */
    __le16  bg_free_blocks_count;/* Free blocks count */
    __le16  bg_free_inodes_count;/* Free inodes count */
    __le16  bg_used_dirs_count; /* Directories count */
    __le16  bg_pad;
    __le32  bg_reserved[3];
};
```

Ext2

Blocks bitmap, Inodes bitmap

- le rôle de ses deux tables est de maintenir la liste d'occupation des blocs et inœuds

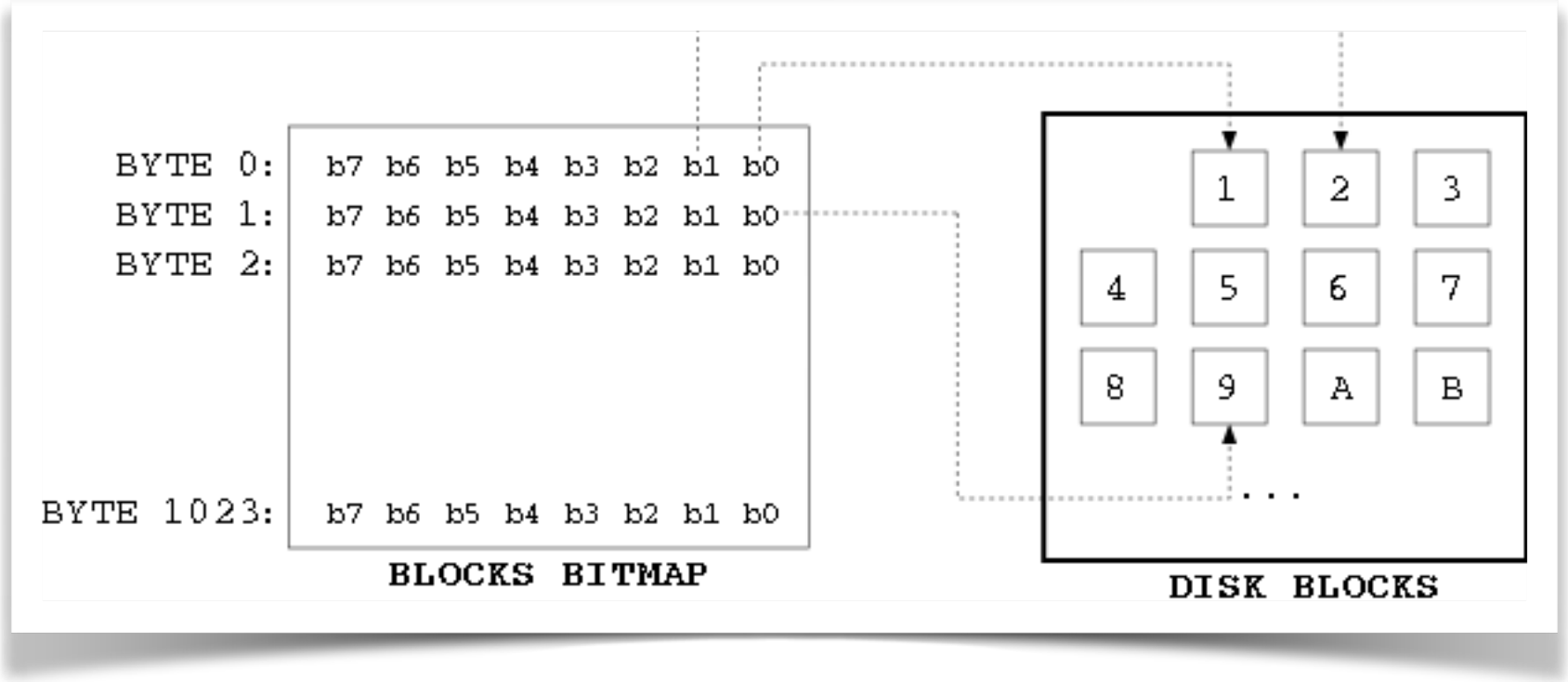
Créer un fichier nécessite obtenir un inœud non actuellement attribué

Ajouter des données à un inœud nécessite d'obtenir un bloc ne contenant actuellement pas de données associées à un fichier

Ext2

L'occupation étant binaire : occupé ou libre

un bit par bloc/inœud suffit



Ext2

inœud

15

ext2.h (4.18.14)

```
struct ext2_inode {
     __le16    i_mode;         /* File mode */
     __le16    i_uid;          /* Low 16 bits of Owner Uid */
     __le32    i_size;         /* Size in bytes */
     __le32    i_atime;        /* Access time */
     __le32    i_ctime;        /* Creation time */
     __le32    i_mtime;        /* Modification time */
     __le32    i_dtime;        /* Deletion Time */
     __le16    i_gid;          /* Low 16 bits of Group Id */
     __le16    i_links_count;  /* Links count */
     __le32    i_blocks;       /* Blocks count */
     __le32    i_flags;        /* File flags */
     __le32    l_i_reserved1;
     __le32    i_block[EXT2_N_BLOCKS];/* Pointers to blocks */
     __le32    i_generation;   /* File version (for NFS) */
     __le32    i_file_acl;     /* File ACL */
     __le32    i_dir_acl;      /* Directory ACL */
     __le32    i_faddr;        /* Fragment address */
     __u8      l_i_frag;       /* Fragment number */
     __u8      l_i_fsize;      /* Fragment size */
     __u16     i_pad1;
     __le16    l_i_uid_high;   /* these 2 fields    */
     __le16    l_i_gid_high;   /* were reserved2[0] */
     __u32     l_i_reserved2;
};
```

Ext2

`i_mode` combinaison

- type
- droits d'accès
  - 9 bits «`rwxrwxrwx`»
  - +3 spéciaux su, sg, sticky

ext2.h (4.18.14)

```
enum {
    EXT2_FT_UNKNOWN     = 0,
    EXT2_FT_REG_FILE    = 1,
    EXT2_FT_DIR         = 2,
    EXT2_FT_CHRDEV      = 3,
    EXT2_FT_BLKDEV      = 4,
    EXT2_FT_FIFO        = 5,
    EXT2_FT_SOCK        = 6,
    EXT2_FT_SYMLINK     = 7,
    EXT2_FT_MAX
};
```

Ext2

`i_mode`

`EXT2_FT_REG_FILE`

fichier ordinaire (sans sens particulier pour le système)

`EXT2_FT_DIR`

répertoire

`EXT2_FT_CHRDEV, BLKDEV`

fichier spécial pour représenter un périphérique, souris ou disque…

`EXT2_FT_FIFO`

fichier spécial permettant de communiquer d'un processus à un autre

`EXT2_FT_SOCK`

fichier spécial permettant de communiquer d'un processus à l'autre de façon symétrique à la manière du réseau

`EXT2_FT_SYMLINK`

fichier spécial permettant de désigner un autre fichier (alias)

Ext2

Horodatage

`i_atime`

dernier accès au contenu de l'inœud

très coûteux, Linux s'autorise à certaines optimisations (`strictatime`, `relatime`, `nodiratime`, `noatime` au montage), idem autres Unix

`i_mtime`

dernière modification du contenu de l'inœud

`i_ctime`

dernière modification des métadonnées de l'inœud

`i_dtime`

libération de l'inœud

Attention, ce sont les dates internes à Ext2, elles ne sont pas nécessairement toutes visibles à l'utilisateur… (dtime?)

Ext2

`i_links_count`

- on peut remarquer que les inœuds ne portent pas de nom!
- L'espace de nommage est contenu dans la structure interne des répertoires
- Cette valeur indique combien d'entrées de répertoires désigne cet inœud
  - Les inœuds ne sont jamais directement supprimés
    - Effet secondaire de la disparition des noms et clôture des fichiers ouverts

Ext2

`i_blocks[15]` ???

- Un inœud ne contient pas ses données associées (son contenu)
  - Une raison simple : la longueur de son contenu est variable au cours du temps!
- Un inœud contient des métadonnées permettant de retrouver les données du contenu
  - C'est à quoi servent ces numéros de blocs
    - Pourquoi 15 ? Les fichiers sont-ils limités à 15*1024o ?
      - Non évidemment, c'est donc un poil plus subtil

Ext2

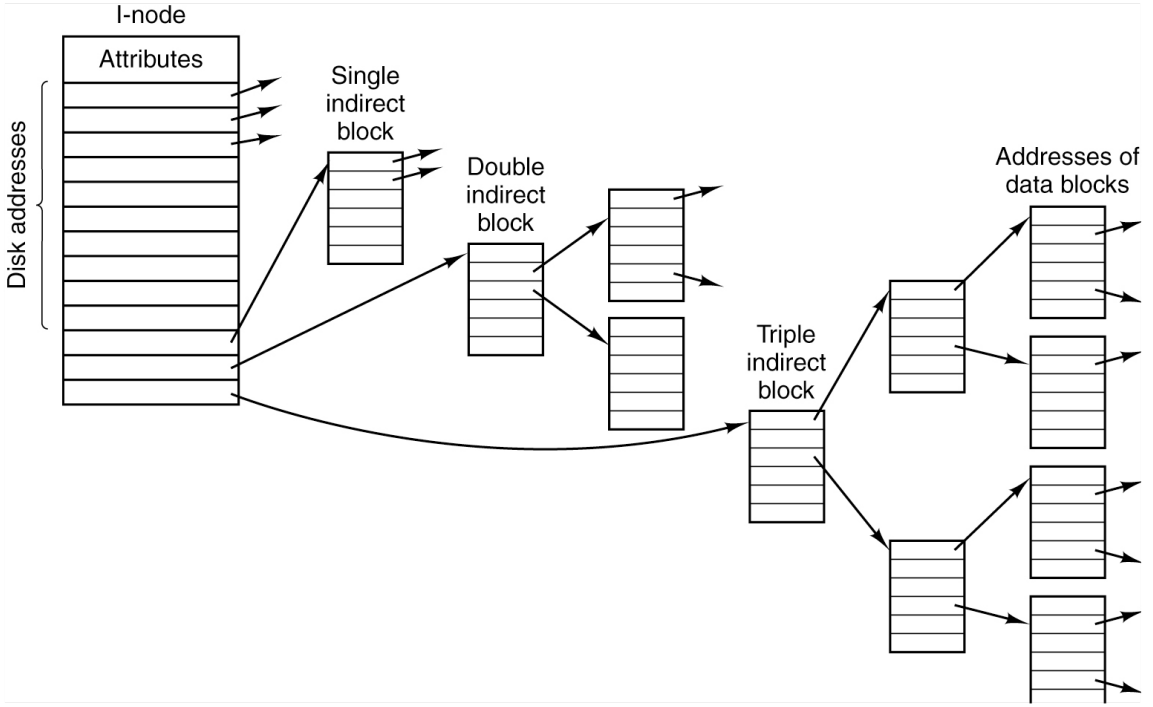
`i_blocks[15]` ???

L'implémentation de la fonction

$\phi_b : B_l \longmapsto B_\phi$

comme pour la mémoire virtuelle!

Ext2



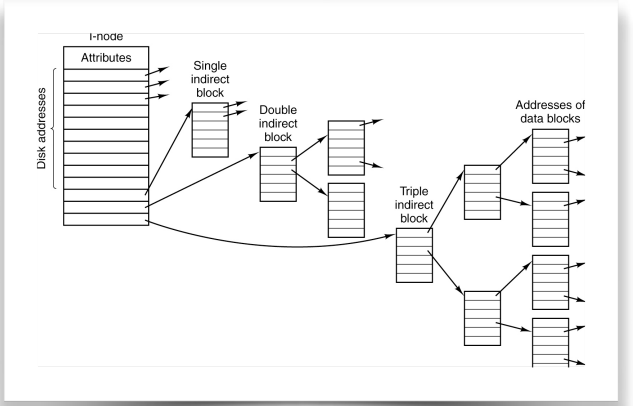
12 blocs «directs»

1 bloc d'indirection de niveau 1

1 bloc d'indirection de niveau 2

1 bloc d'indirection de niveau 3

Ext2



12 blocs «directs»

12 blocs de données

1 bloc d'indirection de niveau 1

1 bloc contenant les adresses de blocs de données

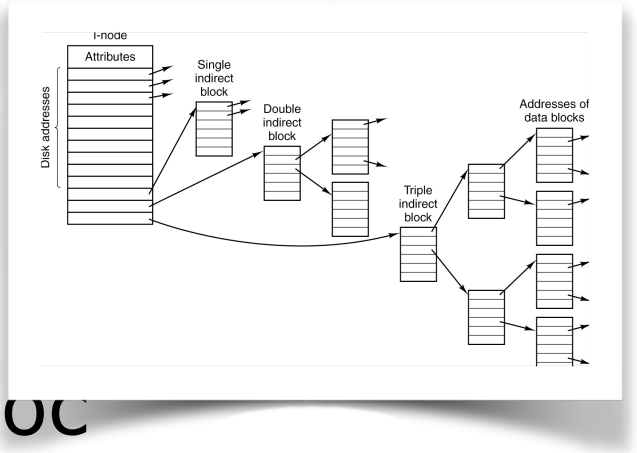
1 bloc d'indirection de niveau 2

1 bloc contenant les adresses de blocs contenant les adresses de blocs de données

1 bloc d'indirection de niveau 3

…

Ext2



Au fur et à mesure que le fichier devient grand on utilise d'abord les blocs directs, puis le bloc d'indirection 1, etc.

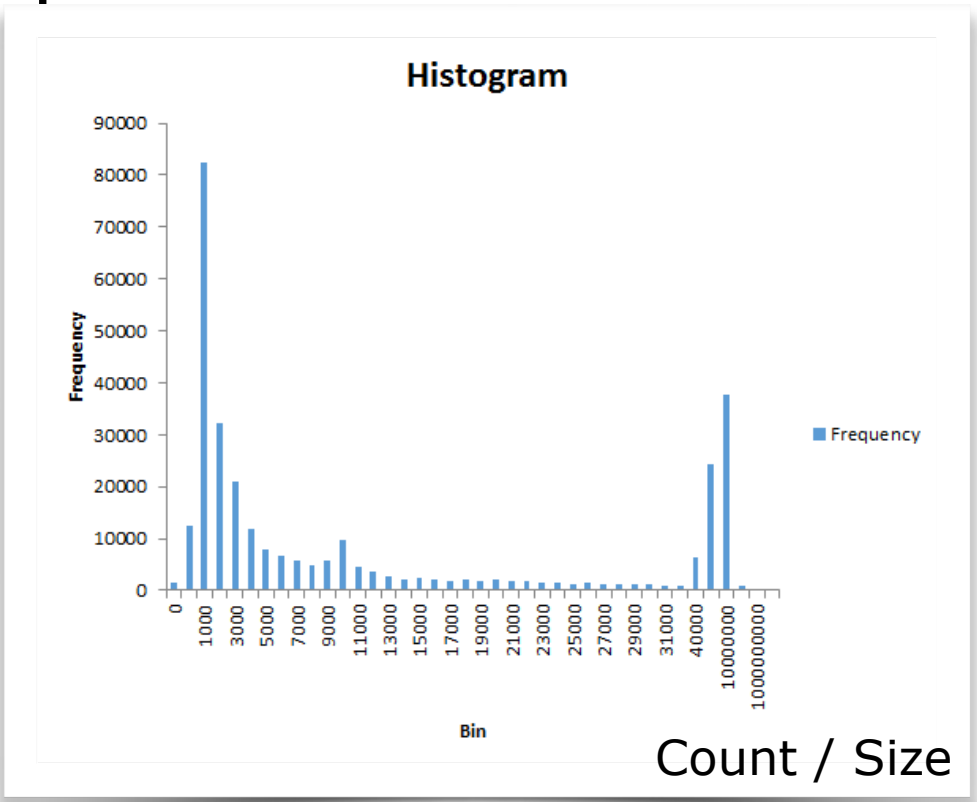
Pour des blocs de taille 4Ko, des adresses de blocs de taille 4o, quelle est la taille maximale possible d'un fichier Ext2 ?

Quelle est la taille d'un fichier n'utilisant pas d'indirection ? l'indirection de niveau 1 ? niveau 2 ?

Ext2

Les indirections n'introduisent que très peu de pénalité

- d'abord parce que les petits fichiers ne sont en rien pénalisés
- qu'il y a du cache…



Count / Size

Ext2

- Il est facile de créer des fichiers à trous
  - au même titre que dans la mémoire virtuelle l'espace est naturellement à trous
    - Linux `fallocate()` permet de recréer des trous lorsque nécessaire…
- Problème avec les trous
  - réservation de l'espace ou non ?
    - `posix_fallocate()`

Ext2

Répertoires

- fichiers dont le contenu est plutôt critique
  - espace de nommage des inœuds
- le contenu est un liste d'entrées
  - essentiellement (nom,n° d'inœud)
- accès contrôlés différemment par l'API
  - `mkdir()`, `opendir()`, `readdir()`, `closedir()`, etc

Ext2

entrée de répertoire

ext2.h (4.18.14)

```
struct ext2_dir_entry {
    __le32  inode;          /* Inode number */
    __le16  rec_len;        /* Directory entry length */
    __le16  name_len;       /* Name length */
    char    name[];         /* File name, up to EXT2_NAME_LEN */
};
```

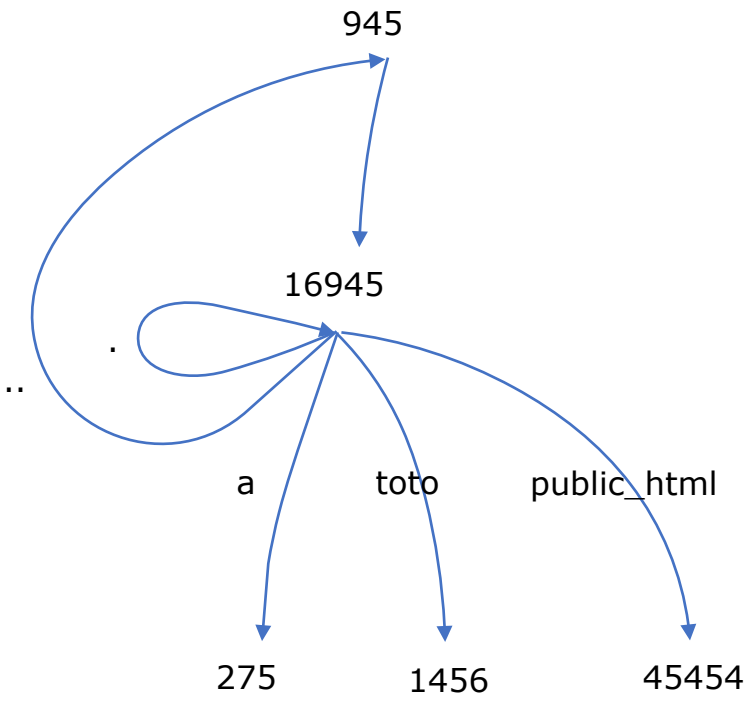
toujours deux entrées . et . .

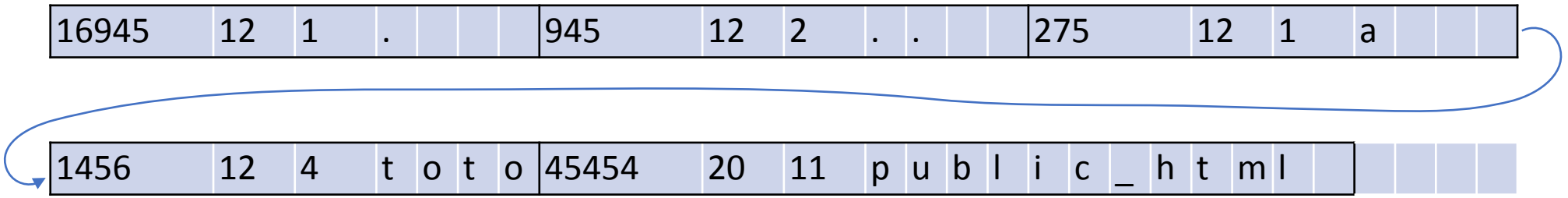
. le nom du répertoire lui-même

. . le nom du répertoire qui contient celui-ci

Ext2

Ex. : un répertoire





Résolution des chemins

Retrouver un inœud à partir d'un chemin absolu :

- décomposer le chemin en composants (/)
- partir de l'inœud racine et retrouver dans ce répertoire le couple correspondant
- répéter

Fonctions

- `filename_lookup()` / `path_lookupat()` dans `namei.c` Linux 4.19

Résolution des chemins

Lorsqu'il y a montage celui-ci mappe un couple vers un couple constitué de la paire (p,nr)

Lorsque le chemin est relatif, on part d'un répertoire particulier (le répertoire dit courant)

Ext2

- Comment supprime t-on une entrée de répertoire ?
  - On la fusionne avec l'entrée précédente
    - Il suffit d'augmenter la taille de l'entrée précédente
- Comment détecte t-on la fin de la liste ?
  - Les entrées de répertoires sont callées de sorte à ce que leur taille totale soit multiple de la taille d'un bloc
- Comme ajoute t-on une entrée ?
  - On cherche une entrée dans laquelle il y a de la place et on «défusionne»
  - Sinon on agrandit à la fin!

Ext2

La structure précédente se révèle trop inefficace lorsque le nombre d'entrée est trop important

- On positionne un indicateur sur le répertoire
  - `EXT2_INDEX_FL`
    - dans ce cas il y a un système d'indexation
  - `EXT2_BTREE_FL`
    - dans ce cas la structure est celle d'un b-arbre
    - structure de données ayant de bonnes caractéristiques de complexité algorithmique

Ext2

L'arborescence d'une partition démarre avec l'inœud de numéro 2

- qui est un répertoire et son propre père…
- l'inœud de numéro 1 sert à stocker les bad blocks
- l'inœud de numéro 0 n'est pas utilisé, car 0 est une valeur sentinelle utilisée dans de nombreuses fonctions

D'autre inœuds sont réservés selon les versions…

Ext2 : Allocation des inœuds

- Pour les répertoires
  - Si le répertoire père est /
    - on le place dans un groupe relativement libre
  - Sinon dans le groupe de son père, sinon où l'on peut
- Pour les fichiers
  - Groupe du père tant que possible
  - Sinon où l'on peut

Ext2 : Allocation des blocs

On essaie d'allouer des blocs consécutifs

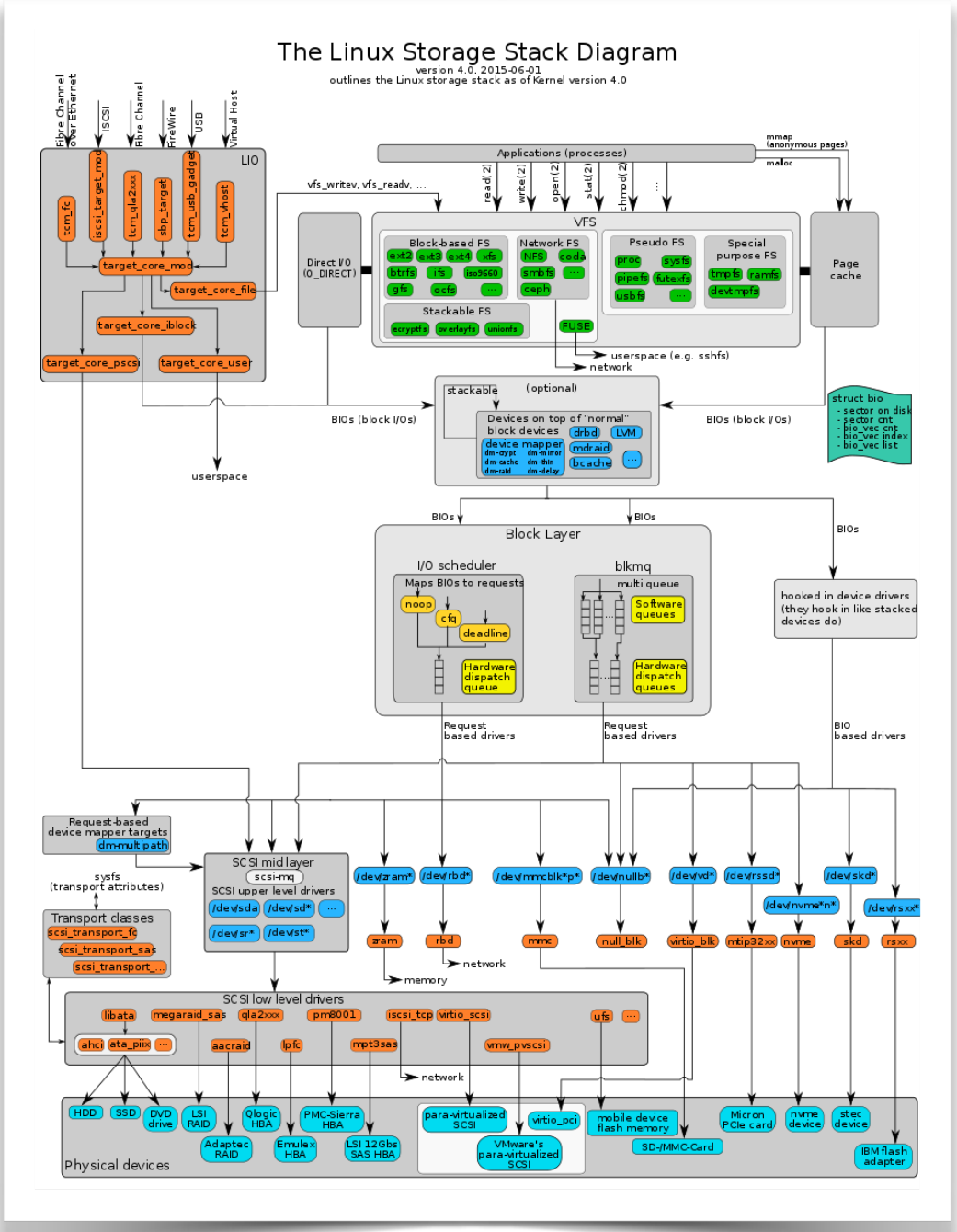
On essaie de rester dans le même groupe

À défaut on essaie d'allouer là où il y a au moins 8 blocs libres consécutifs

On réalise de la préallocation

relâchée à la fermeture effective…

Linux FS stack



Fragmentation

Deux types :

- interne
  - due à la granularité de l'allocation physique (allocation par blocs vs vision, utilisateur à l'octet près)
- externe
  - décorrélation entre espace logique et espace physique

Le problème existait déjà avec le mémoire!

Fragmentation

Des bons algorithmes permettent de lutter efficacement contre les effets induits par la fragmentation externe

- violation de la localité, etc
  - mauvais algorithmes ➥ defrag!

read-ahead

- la lecture ou l'écriture étant souvent séquentielle
  - lorsqu'on utilise un bloc logique N, le système charge en avance le bloc logique N+1
  - très efficace

Fragmentation

- Il n'y a pas de moyen véritable de lutter contre la fragmentation interne
  - Le problème est surtout important pour les petits fichiers
    - astuce NTFS ?
    - BSD fragment ?
      - sous-découpage des blocs, gestion plus complexe
    - Est-ce que ça vaut le coup ?

NTFS

Une astuce rigolote

Pour les tout petits fichiers (minuscules)

stockage des données dans le nœud

Versioning File Systems

- VMS / Files-11
  - une ouverture en écriture provoquait la création d'une nouvelle version du fichier
    - `file.txt;1 file.txt;2…`
- macOS
  - un mécanisme similaire à TimeMachine permet d'obtenir des versions de documents
    - pour les applications conformes au mécanisme
      - plus de «Save as…»

Snapshoting

- Une sorte de TimeMachine à un bas niveau
  - idée similaire au Copy-On-Write
    - création du snapshot par protection en écriture
      - on peut donc stocker le snapshot tranquillement
    - durant ce temps les écritures provoquent des copies des données à modifier (une copie pour le snapshot, une copie pour le système en cours d'exécution)

Log-structured File System

- Très utile!
  - permet de restaurer facilement l'état d'un système de fichiers
  - permet de gérer efficacement des systèmes de fichiers sur des supports type SSD
    - la lecture est simple, pas la réécriture
      - éviter de réécrire et en même temps diffuser les écritures équitablement sur tout l'espace physique
        - nombre de réécritures limité (physiquement)

HFS+

- L'ancien système de fichier d'Apple
- Des ressources fork!
  - Plusieurs ensembles de données pour un même fichier!
    - pourquoi ?
      - données (au sens habituel) + données associées (icônes, etc)
    - Abandonné et remplacé par le concept de Bundle

Alias mac vs Unix

- Liens symboliques Unix
  - fichier contenant un chemin d'accès à un autre fichier
    - pis-aller historique pour obtenir des alias franchissant les partitions
      - pourquoi les liens ne le peuvent-ils pas ?
    - Inconvénient : si la destination bouge…
- Alias macOS
  - fichier contenant l'identité d'un autre fichier
    - un hybride entre le hardlink et softlink
- En général on recommande de ne pas trop utiliser les alias

APFS

Le dernier système de fichier d'Apple

Clonage des fichiers

- Copy-On-Write

Snapshots

Chiffrement

Support d'Unicode

Renommage atomique

Apple Time Machine

L'idée est de créer des images temporelles d'une partition

- entre deux images temporelles, il n'y a que peu de modifications (si le temps écoulé est relativement court)
- on ne *stocke* que les différences…
- utilisation des liens pour reconstituer une image apparemment complète

Apple Time Machine

Très pratique

Peu gourmand en espace

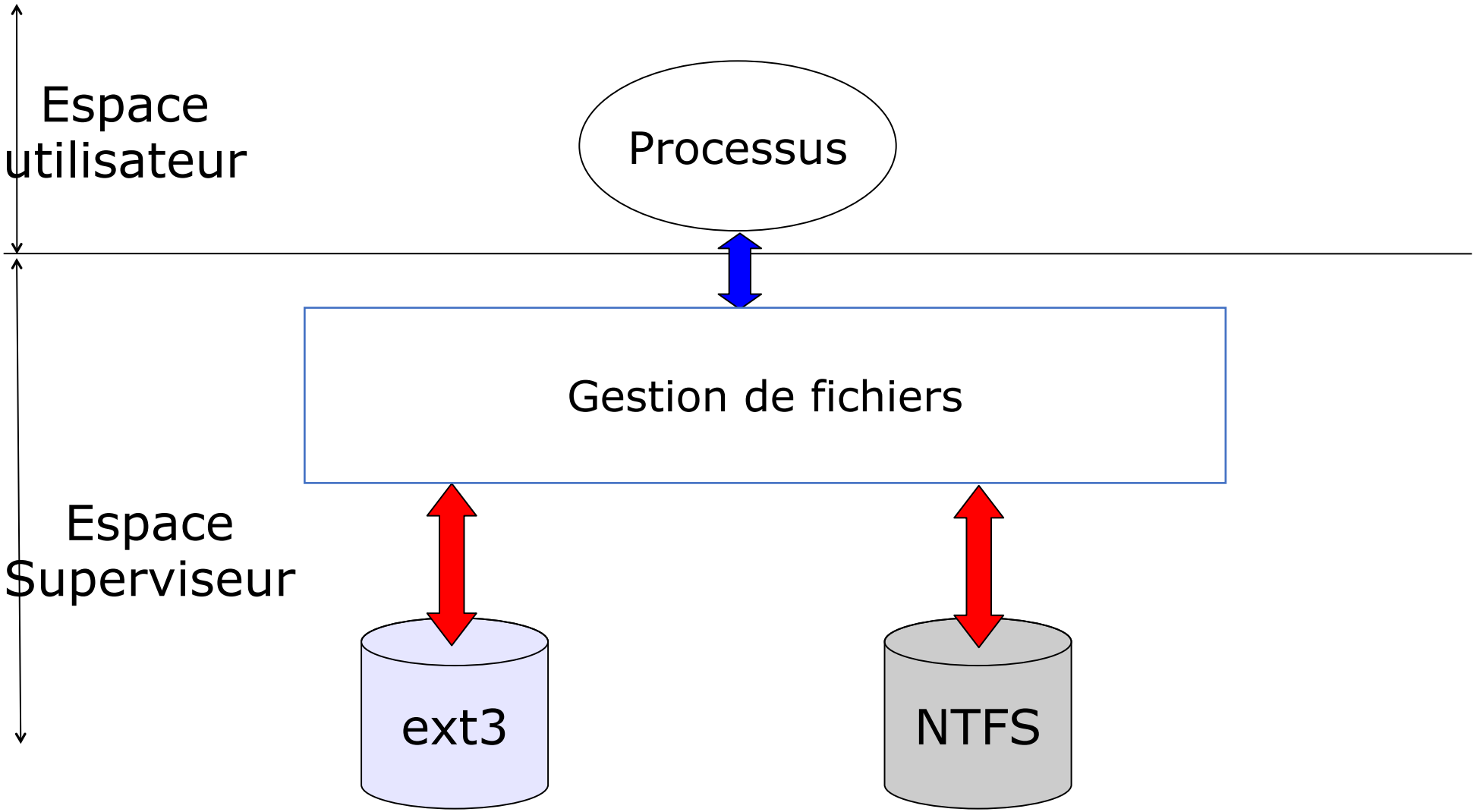
VFS

La couche VFS Virtual File System (apparue avec SunOS 2.0) permet d'abstraire les particularités sous-jacentes

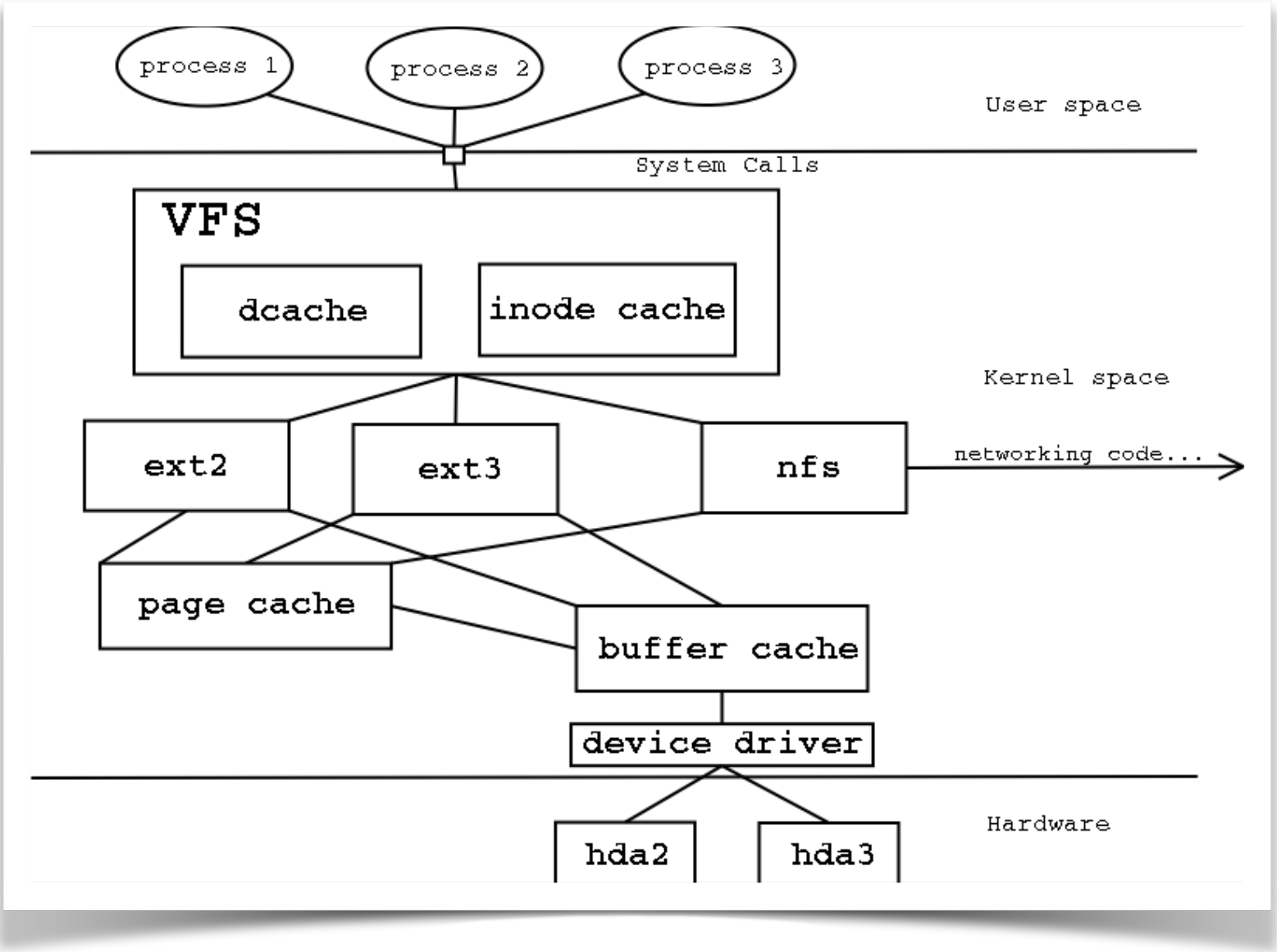
- le système de fichiers concret doivt simplement fournir des données exploitables pour VFS
  - inœuds
  - entrées de répertoires
- si certaines données ou services sont impossibles à fournir, on trouve des alternatives (valeurs par défaut, échec des appels)
  - Ex. : FAT ne supporte pas les liens symboliques…

C'est une vision objet, mais en C…

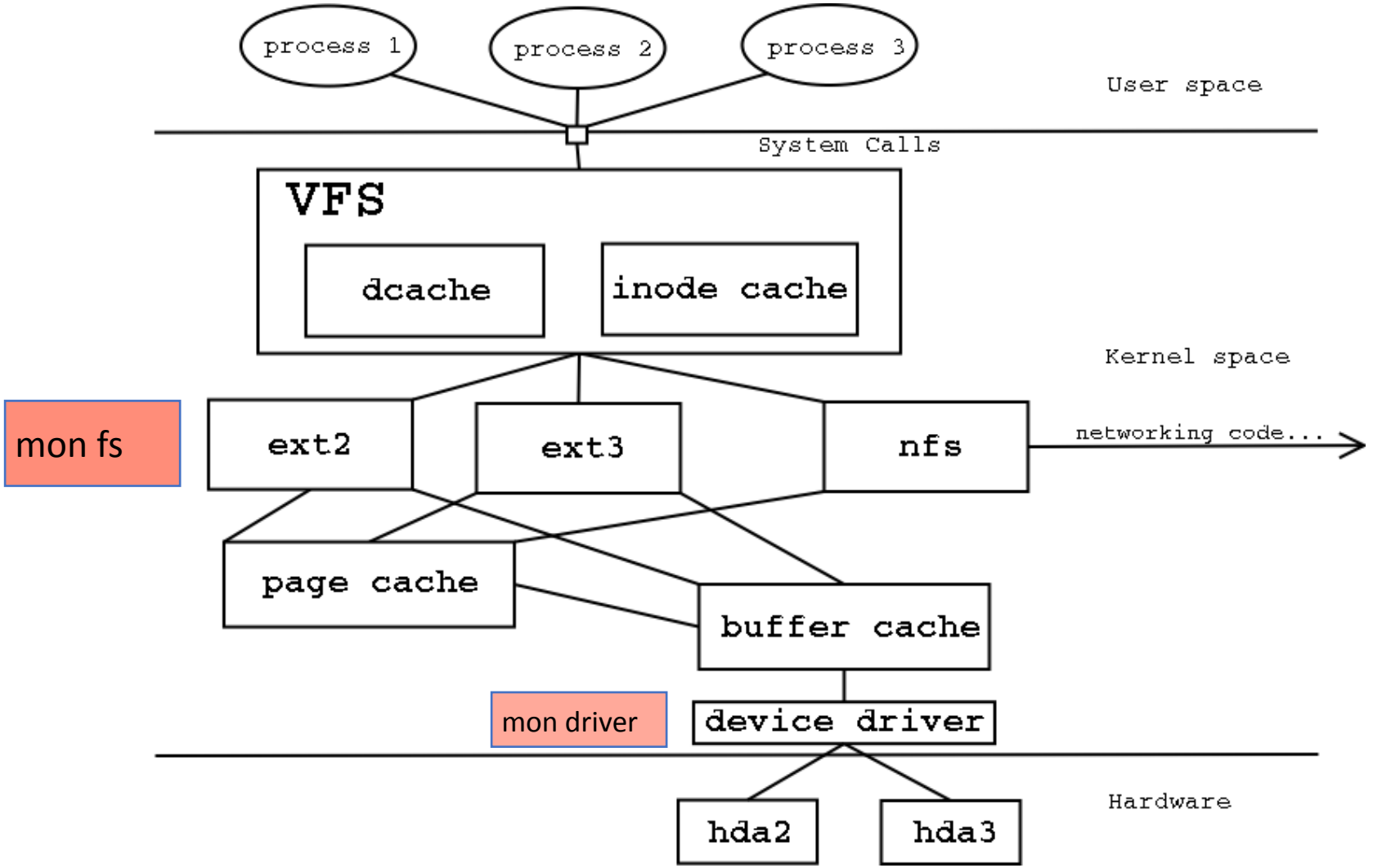
VFS


Espace utilisateur
Processus
Gestion de fichiers
Espace Superviseur
ext3
NTFS

Structure plus détaillée



Structure plus détaillée



VFS

fs.h (4.18.14)

```
struct file_operations {
        struct module *owner;
        loff_t (*llseek) (struct file *, loff_t, int);
        ssize_t (*read) (struct file *, char __user *, size_t, loff_t *);
        ssize_t (*write) (struct file *, const char __user *, size_t, loff_t *);
        ssize_t (*read_iter) (struct kiocb *, struct iov_iter *);
        ssize_t (*write_iter) (struct kiocb *, struct iov_iter *);
        int (*iterate) (struct file *, struct dir_context *);
        int (*iterate_shared) (struct file *, struct dir_context *);
        __poll_t (*poll) (struct file *, struct poll_table_struct *);
        long (*unlocked_ioctl) (struct file *, unsigned int, unsigned long);
        long (*compat_ioctl) (struct file *, unsigned int, unsigned long);
        int (*mmap) (struct file *, struct vm_area_struct *);
        unsigned long mmap_supported_flags;
        int (*open) (struct inode *, struct file *);
        int (*flush) (struct file *, fl_owner_t id);
        int (*release) (struct inode *, struct file *);
        int (*fsync) (struct file *, loff_t, loff_t, int datasync);
        int (*fasync) (int, struct file *, int);
        int (*lock) (struct file *, int, struct file_lock *);
        ssize_t (*sendpage) (struct file *, struct page *, int, size_t, loff_t *, int);
        unsigned long (*get_unmapped_area)(struct file *, unsigned long, unsigned long, unsigned long, unsigned long);
        int (*check_flags)(int);
        int (*flock) (struct file *, int, struct file_lock *);
        ssize_t (*splice_write)(struct pipe_inode_info *, struct file *, loff_t *, size_t, unsigned int);
        ssize_t (*splice_read)(struct file *, loff_t *, struct pipe_inode_info *, size_t, unsigned int);
        int (*setlease)(struct file *, long, struct file_lock **, void **);
        long (*fallocate)(struct file *file, int mode, loff_t offset, loff_t len);
        void (*show_fdinfo)(struct seq_file *m, struct file *f);
#ifndef CONFIG_MMU
        unsigned (*mmap_capabilities)(struct file *);
#endif
        ssize_t (*copy_file_range)(struct file *, loff_t, struct file *, loff_t, size_t, unsigned int);
```

Montage

Unix propose à l'utilisateur une vision unifiée des fichiers, une seule arborescence

- même si les fichiers sont contenus dans des partitions différentes

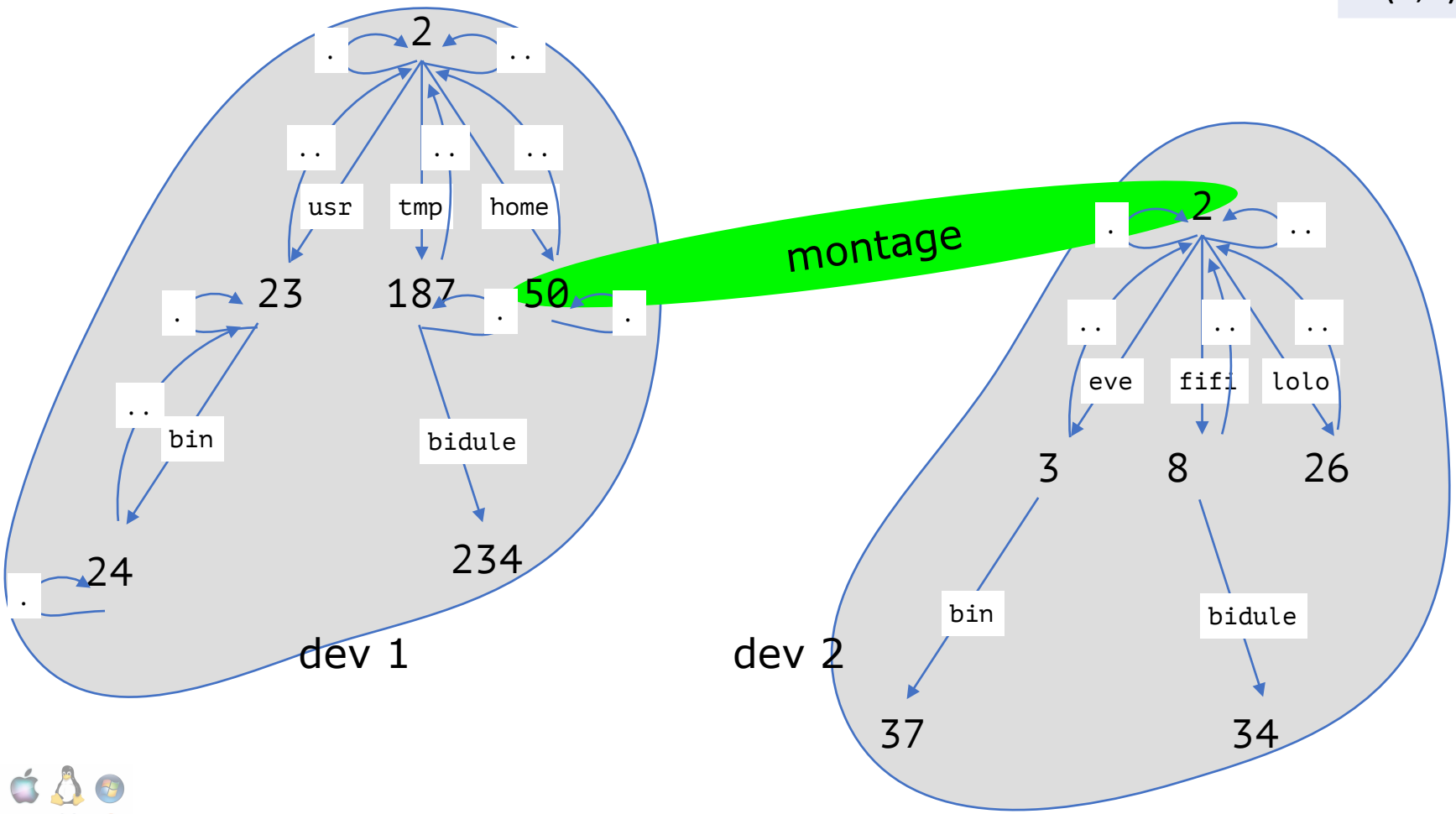
Les arborescences sont greffées les unes aux autres à l'exécution

- `mount`

Montage

Table de montage

monté	sur
(2,2)	(1,50)



Montage

```
<lucy-2-[13:56]-[~]> df
Filesystem                1K-blocks       Used  Available Use% Mounted on
udev                       65957692          0   65957692   0% /dev
tmpfs                      13194064       2008   13192056   1% /run
/dev/sdb2                  50557408    5579008   42340580  12% /
tmpfs                      65970312          0   65970312   0% /dev/shm
tmpfs                          5120          0       5120   0% /run/lock
tmpfs                      65970312          0   65970312   0% /sys/fs/cgroup
/dev/sda1                  52428800   17991664   33515712  35% /client
/dev/sda3                  51343840   18106120   30599896  38% /lucien-root
/dev/sda2                  51343840   10853180   37852836  23% /lucien-home
/dev/sdb1                    523248        144     523104   1% /boot/efi
```

```
<lucy-11-[17:30]-[~/.ssh]> mount
udev on /dev type devtmpfs (rw,nosuid,relatime,size=65957692k,nr_inodes=16489423,mode=755)
tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,size=13194064k,mode=755)
/dev/sdb2 on / type ext4 (rw,relatime,errors=remount-ro,data=ordered)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,size=5120k)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=755)
/dev/sda1 on /client type btrfs (rw,relatime,space_cache,subvolid=259,subvol=/lulu)
/dev/sda3 on /lucien-root type ext4 (rw,relatime,data=ordered)
/dev/sda2 on /lucien-home type ext4 (rw,relatime,data=ordered)
/dev/sdb1 on /boot/efi type vfat
(rw,relatime,fmask=0077,dmask=0077,codepage=437,iocharset=ascii,shortname=mixed,utf8,errors=remount-ro)
```

Montage

Un fichier est identifiée de façon unique dans une partition par son inœud

Dans un Unix en exécution c'est un couple (périphérique,n°inœud) qui identifie un fichier

Contexte Fichiers/Processus

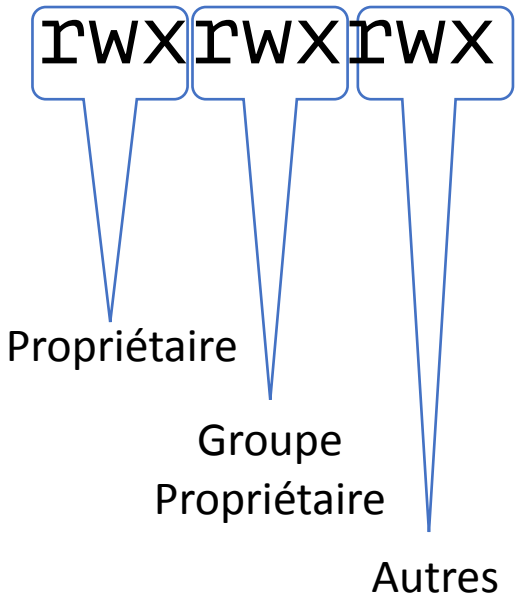
Les fichiers sont propriété d'un propriétaire et d'un groupe

Les processus aussi

la combinaison des deux détermine l'accès aux fichiers d'un processus

Droits d'accès

Droits d'accès de base d'un fichier, 9 bits



`r` : lecture

`w` : écriture

`x` : exécution

Droits d'accès

Attention, pour un fichier :

lecture : autorisation de consulter le contenu

écriture : autorisation de modifier le contenu

exécution : si exécutable (objet ou script) peut-être exécuté

Droits d'accès

Attention, pour un répertoire :

lecture : autorisation de consulter le contenu, donc consulter le contenu du répertoire

écriture : autorisation de modifier le contenu, donc de supprimer ou créer un fichier!

traversable : peut-être utilisé dans un chemin d'accès…

Le droit de supprimer un fichier n'est pas lié au fichier lui-même!!!

- on ne supprime jamais de fichier directement
  - garbage collection…

Droits d'accès

Le processus est propriété de (u,g)

Le fichier est propriété de (u',g')

Si u=u', ce sont les droits «propriétaire» qui s'appliquent

Si u≠u' et g=g', ce sont les droits «groupe propriétaire»

Si u≠u' et g≠g', ce sont les droits «autres»

Attention aux cas bizarres, type suid/sgid/sticky…

Droits d'accès

namei.c (Linux 4.19)

```
static int acl_permission_check(struct inode *inode, int mask)
{
        unsigned int mode = inode->i_mode;

        if (likely(uid_eq(current_fsuid(), inode->i_uid)))
                mode >>= 6;
        else {
                if (IS_POSIXACL(inode) && (mode & S_IRWXG)) {
                        int error = check_acl(inode, mask);
                        if (error != -EAGAIN)
                                return error;
                }

                if (in_group_p(inode->i_gid))
                        mode >>= 3;
        }
        /*
         * If the DACs are ok we don't need any capability check.
         */
        if ((mask & ~mode & (MAY_READ | MAY_WRITE | MAY_EXEC)) == 0)
                return 0;
        return -EACCES;
}
```

Droits d'accès

Ce type de droits d'accès n'est pas toujours suffisant

ACL Access Control List

idée : fournir un mécanisme plus fin permettant de donner un (ou des) droit(s) particulier à un (ou des) utilisateur(s) en particulier

ACLs

Par chmod étendu (Ex. macOS)

```
[Estafette:/tmp/toto] ls -ail titi
8619487659 -rw-r--r--   1 yunes  wheel    0  6 nov 16:19 titi
[Estafette:/tmp/toto] chmod +a "dupont allow write" titi
[Estafette:/tmp/toto] ls -ail titi
8619487659 -rw-r--r--+  1 yunes  wheel    0  6 nov 16:19 titi
[Estafette:/tmp/toto] ls -aile titi
8619487659 -rw-r--r--+  1 yunes  wheel    0  6 nov 16:19 titi
 0: user:dupont allow write
[Estafette:/tmp/toto] chmod +a "dupont deny delete" titi
[Estafette:/tmp/toto] ls -aile titi
8619487659 -rw-r--r--+  1 yunes  wheel    0  6 nov 16:19 titi
 0: user:dupont deny delete
 1: user:dupont allow write
```

ACLs

macOS possède une collection riche de permissions utilisables avec les ACLs :

- tout objet :
  - `delete`
  - `readattr`, `writeattr`, `readextattr`, `writeextattr`, `readsecurity`, `writesecurity`
  - `chown`
- répertoires seulement :
  - `list`, `search`, `add_file`, `add_subdirectory`, `delete_child`
- tout sauf répertoires :
  - `read`, `write`, `append`, `execute`

plus des contrôles de l'héritage (répertoires uniquement) :

- `file_inherit`, `directory_inherit`, `limit_inherit`, `only_inherit`

ACLs

Par commandes spéciales (Ex. Linux)

```
parallels@ubuntu:/tmp/toto$ ls -ail titi
3408016 -rw-rw-r--+ 1 parallels parallels 0 Nov  6 16:23 titi
parallels@ubuntu:/tmp/toto$ setfacl -m user:dupont:w titi
parallels@ubuntu:/tmp/toto$ ls -ail titi
3408016 -rw-rw-r--+ 1 parallels parallels 0 Nov  6 16:23 titi
parallels@ubuntu:/tmp/toto$ getfacl titi
# file: titi
# owner: parallels
# group: parallels
user::rw-
user:dupont:-w-
group::rw-
mask::rw-
other::r--

parallels@ubuntu:/tmp/toto$ setfacl -x user:dupont titi
parallels@ubuntu:/tmp/toto$ getfacl titi
# file: titi
# owner: parallels
# group: parallels
user::rw-
group::rw-
mask::rw-
other::r--

parallels@ubuntu:/tmp/toto$
```

Attributs étendus

Certains systèmes/filesystem supporte des attributs étendus

- par exemple dans Linux :
  - exclure un fichier d'une sauvegarde
  - écriture synchronisée forcée
- par exemple macOS :
  - rien de normalisé, chaque application peut définir/utiliser les attributs qu'elle souhaite
    - mise en quarantaine (standard macOS)

Attributs étendus

Linux :

`chattr`, `lsattr`

macOS :

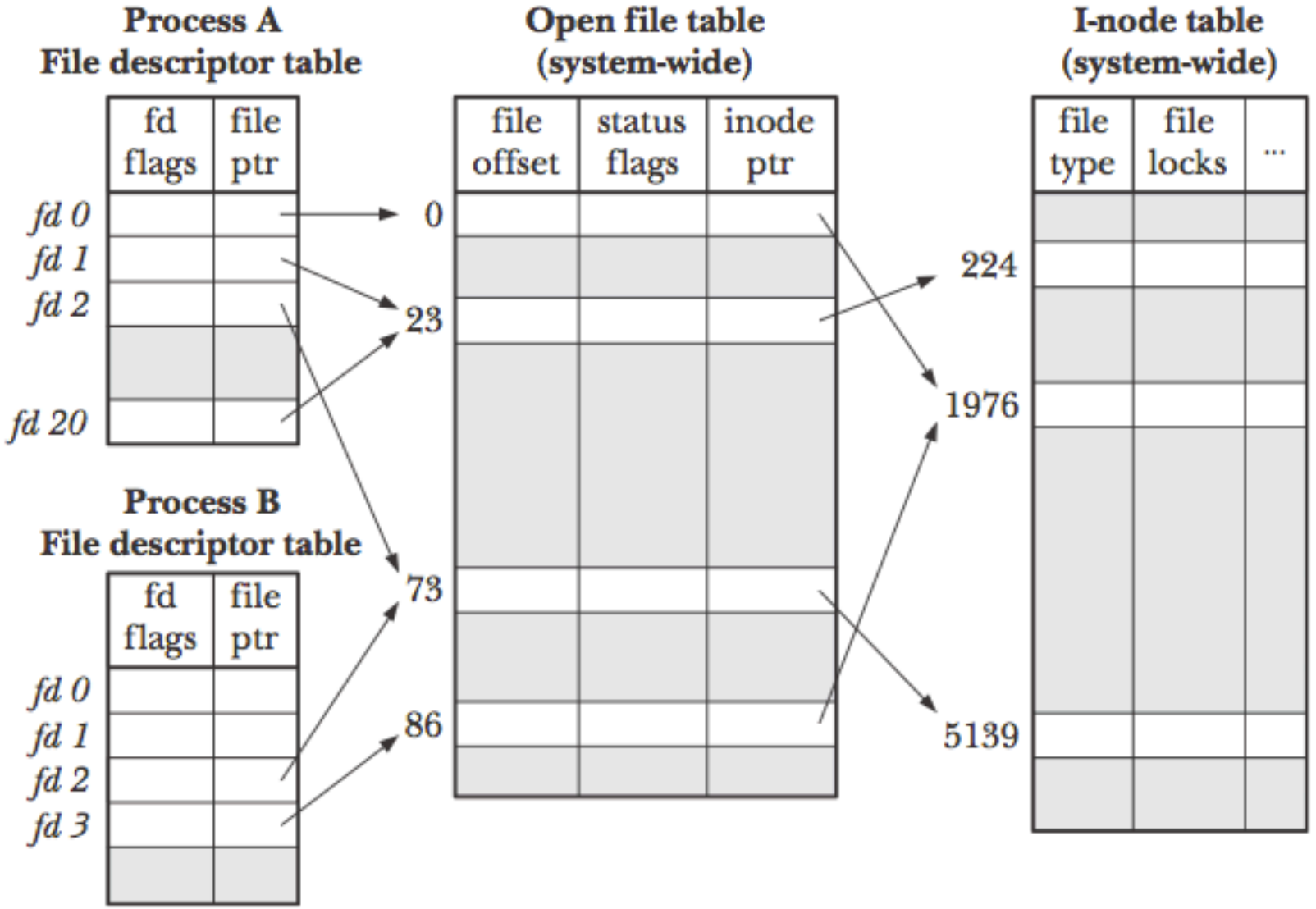
`xattr`

Contexte Fichiers/Processus

Dans la description système d'un processus on a (entre autres) :

- le répertoire courant
- les fichiers ouverts

Contexte Fichiers/Processus



Contexte Fichiers/Processus

sched.h (4.18.14)

```
struct task_struct {
...
    /* Open file information: */
    struct files_struct      *files;
...
```

fdtable.h (4.18.14)

```
struct files_struct {
        atomic_t count;
        bool resize_in_progress;
        wait_queue_head_t resize_wait;

        struct fdtable __rcu *fdt;
        struct fdtable fdtab;
        spinlock_t file_lock ____cacheline_aligned_in_smp;
        unsigned int next_fd;
        unsigned long close_on_exec_init[1];
        unsigned long open_fds_init[1];
        unsigned long full_fds_bits_init[1];
        struct file __rcu * fd_array[NR_OPEN_DEFAULT];
};
```

fs.h (4.18.14)

```
struct file {
    union {
        struct llist_node   fu_llist;
        struct rcu_head     fu_rcuhead;
    } f_u;
    struct path         f_path;
    struct inode        *f_inode;
    const struct file_operations  *f_op;
    spinlock_t          f_lock;
    enum rw_hint        f_write_hint;
    atomic_long_t       f_count;
    unsigned int        f_flags;
    fmode_t             f_mode;
    struct mutex        f_pos_lock;
    loff_t              f_pos;
    struct fown_struct  f_owner;
    const struct cred   *f_cred;
    struct file_ra_state     f_ra;
    u64             f_version;
    void            *private_data;
    struct address_space     *f_mapping;
    errseq_t        f_wb_err;
}
```

Contexte Fichiers/Processus

`fcntl()` qui permet de modifier les caractéristiques

- associées au descripteur
  - `F_DUPD`
  - `F_GETFD`, `F_SETFD`
    - `FD_CLOEXEC`
- au fichier ouvert correspondant au descripteur
  - `F_GETFL`, `F_SETFL`
    - `O_APPEND`, `O_DSYNC`, `O_NONBLOCK`, `O_RSYNC`, `O_SYNC`

Contexte Fichiers/Processus

`fcntl()` permet aussi de manipuler le verrouillage d'un fichier

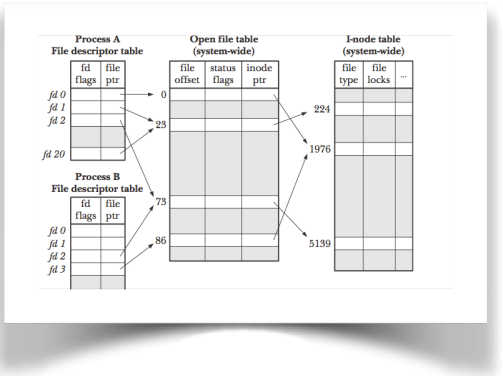
`F_GETLK`, `F_SETLK`, `F_SETLKW`

`F_RDLCK`, `F_UNLCK`, `F_WRLCK`

Attention : le verrouillage n'est qu'indicatif (en général)

Contexte Fichiers/Processus

- `0, 1, 2`
  - `STDIN_FILENO, STDOUT_FILENO, STDERR_FILENO`
  - `stdin, stdout, stderr`
  - `cin, cout, cerr`
  - `System.in, System.out, System.err`
- Héritage de la table des descripteurs lors d'un `exec*()`
  - redirections



Unification mémoire/fichier

Pagination et découpage en blocs ont un fonctionnement très similaire

La pagination fait appel au swap (mémoire mappée sur disque)

Pourquoi ne pas réaliser l'inverse ?

- ☛ fichier mappé en mémoire

mmap

Obtention du mapping d'un bout de l'espace d'adressage (addr,addr+len) et d'un bout de fichier (off,off+len)

NAME

mmap - map pages of memory

SYNOPSIS

```
[MC3] #include <sys/mman.h>

void *mmap(void *addr, size_t len, int prot, int flags,
       int fildes, off_t off);
```

DESCRIPTION

The *mmap*() function shall establish a mapping between a process' address space and a file, shared memory object, or [TYM] typed memory object.

mmap

POSIX prévoit 4 types de protection :

- `PROT_NONE` (guard pages ?)
- `PROT_READ`
- `PROT_WRITE`
- `PROT_EXEC`

mmap

POSIX prévoit 3 types de mapping :

`MAP_PRIVATE`

les modifications de la mémoire sont privées et ne sont pas répercutées sur le fichier

`MAP_SHARED`

les modifications sont partagées (visibles)

`MAP_FIXED`

l'adresse est fixée par l'appelant et non choisie par le système. Usage réservé à des cas très particuliers

mmap

Certains OS proposent d'autres options :

macOS :

`MAP_ANON(YMOUS)`

pour obtenir des pages ne correspondant pas à un fichier. C'est une forme d'allocation mémoire

`MAP_HASSEMAPHORE`

si le segment de mémoire est prévu pour contenir des sémaphores

`MAP_NOCACHE`

pour indiquer que ces pages peuvent être réclamées sans dommage pour les performances

mmap

Certains OS proposent d'autres options :

Linux :

- MAP_32BIT (2.6)
  - pour forcer le mapping dans les 4 premiers Go de l'espace
- MAP_FIXED_NOREPLACE (4.17)
  - comme FIXED mais interdit le mapping sur une région déjà mappée
- MAP_GROWSDOWN
  - pour un mapping permettant une augmentation automatisée du segment (similaire à une pile)
- MAP_HUGETLB, MAP_HUGE_2MB, MAP_HUGE_1GB (3.8)
  - pour obtenir un mapping géré à l'aide de «grosses» pages (Linux supporte les pages de taille variable)
- MAP_LOCKED (2.5)
  - pages non swapables
- MAP_NORESERVE
  - ne réserve pas de swap pour ces pages (peut conduire à des échecs en écriture)

mmap

Certains OS proposent d'autres options :

Linux :

- MAP_POPULATE (2.5)
  - pages pré-chargées et non pas à la demande
- MAP_STACK (2.6)
  - pour obtenir des pages utilisables pour des piles de threads
- MAP_SYNC (2.6)
  - pour obtenir la persistance des modifications (write-through), sur certains kernels uniquement
- MAP_UNINITIALIZED
  - pour obtenir des pages non effacées (plus efficace mais présente un certain danger)
- MAP_ANON(YMOUS)
  - pour obtenir des pages ne correspondant pas à un fichier. C'est une forme d'allocation mémoire

Tubes nommés

Les tubes anonymes ne permettent de communiquer qu'entre processus qui utilisent la propriété d'héritage du fork()

Il est possible d'obtenir le même type de communication via des tubes nommés :

- possèdent une entrée dans le système de fichier (ils sont donc nommés)
- possèdent un mécanisme de synchronisation à l'ouverture

Tubes nommés

À l'ouverture

- en lecture, s'il n'y a pas d'écrivain déclaré, la demande est bloquée
- en écriture, s'il n'y a pas de lecteur déclaré, la demande est bloquée

Il y a donc une synchronisation à la manière du téléphone où il est nécessaire d'établir la liaison avant de communiquer

Attention : les choses sont légèrement différents dans le cas non-bloquant

E/S non-bloquantes

- Le mode simple d'entrées/sorties est le mode bloquant
  - on effectue une requête et on attend le résultat
    - problème : si la requête ne peut être satisfaite dans l'instant, on est bloqué jusqu'à ce que les conditions de réalisation soient satisfaites
      - ce n'est pas toujours souhaitable, on a peut-être mieux à faire en attendant, quitte à y revenir plus tard…

E/S non-bloquantes

Le mode non bloquant est celui dans lequel

- si la requête peut-être satisfaite on la satisfait (mais on est bloqué jusqu'à la fin de la requête)
- si la requête ne peut-être satisfaite, la requête est annulée et on en est prévenu

Ex. : réseau : rien à lire ? Attendre ou pas ?

E/S non-bloquantes

Le mode peut-être choisi à l'acquisition de la ressource ou en cours d'utilisation

Dans le mode non-bloquant les opérations renvoie une erreur (`-1`) et positionne le statut d'erreur (`errno`) à `EAGAIN`

Pour les ouvertures en mode non-bloquant (par exemple tube) alors un écrivain sans lecteur provoque une erreur alors q'un lecteur sans écrivain obtient l'ouverture

E/S Asynchrones

Ne pas confondre mode non-bloquant et mode asynchrone

Le mode asynchrone correspond au scenario suivant :

- on envoie la requête sans être bloqué
- on continue à faire autre chose tandis que le système prend en charge la requête
- lorsqu'on est intéressé par la terminaison de la requête on peut questionner le système sur son état (en mode bloquant ou non éventuellement!)
  - polling («c'est fini ?», «c'est fini ?»…)

E/S Asynchrones

`aio_read`, `aio_write`

lancer une requête de lecture ou écriture

`aio_cancel`

annuler une requête

`aio_error`

pour tester le status courant de l'opération

`EINPROGRESS` (en cours)

`aio_return`

pour récupérer le résultat (uniquement si `aio_error` renvoie `0`)

Multiplexage des E/S

On peut avoir besoin de contrôler plusieurs E/S et en particulier des canaux de communication

- `poll`
- `select`

ou alors utiliser des threads :-)

Communication

IPC System V

- Tubes (déjà vus)
- Sémaphores (déjà vus ?)
- Segments de mémoire partagée (déjà vus ?)
- Files de messages

Communication

Les sémaphores System V sont plus «puissants» (P/V étendus, contrôle d'accès) mais un peu plus complexes à utiliser

- Les sémaphores POSIX sont plus performants

Les segments de mémoire partagés (shm) System V sont aussi plus versatiles

- Est-ce encore vrai ?

Communication

Les files de messages

POSIX et SysV

des messages (unités de lecture/écriture)

lectures destructrices

priorités

SGF distribué

Propriétés attendues :

- transparence de l'accès
- transparence de la localisation
- nommage indépendant de la localisation
- tolérance aux pannes
- mobilité des fichiers

SGF distribué

Espace de nommage uniforme ?

- si oui, transparence de la localisation

Serveur avec ou sans état ?

- des requêtes SGD nécessitent de conserver un état, qui le fait :
  - client ? serveur plus simple
  - serveur ? plus complexe et en cas de plantage plus problématique…

Sémantique des écritures

- à la Unix (tout est tout de suite visible ?) coûteux
- « session » à la fermeture ou de temps en temps…

NFS

Network File System (inventé par Sun)

Concurrents : AFS, RFS

Idée : délocaliser les fichiers de façon transparente

NFS conception

Non restreint à Unix

Protocole indépendant du matériel

Sans état

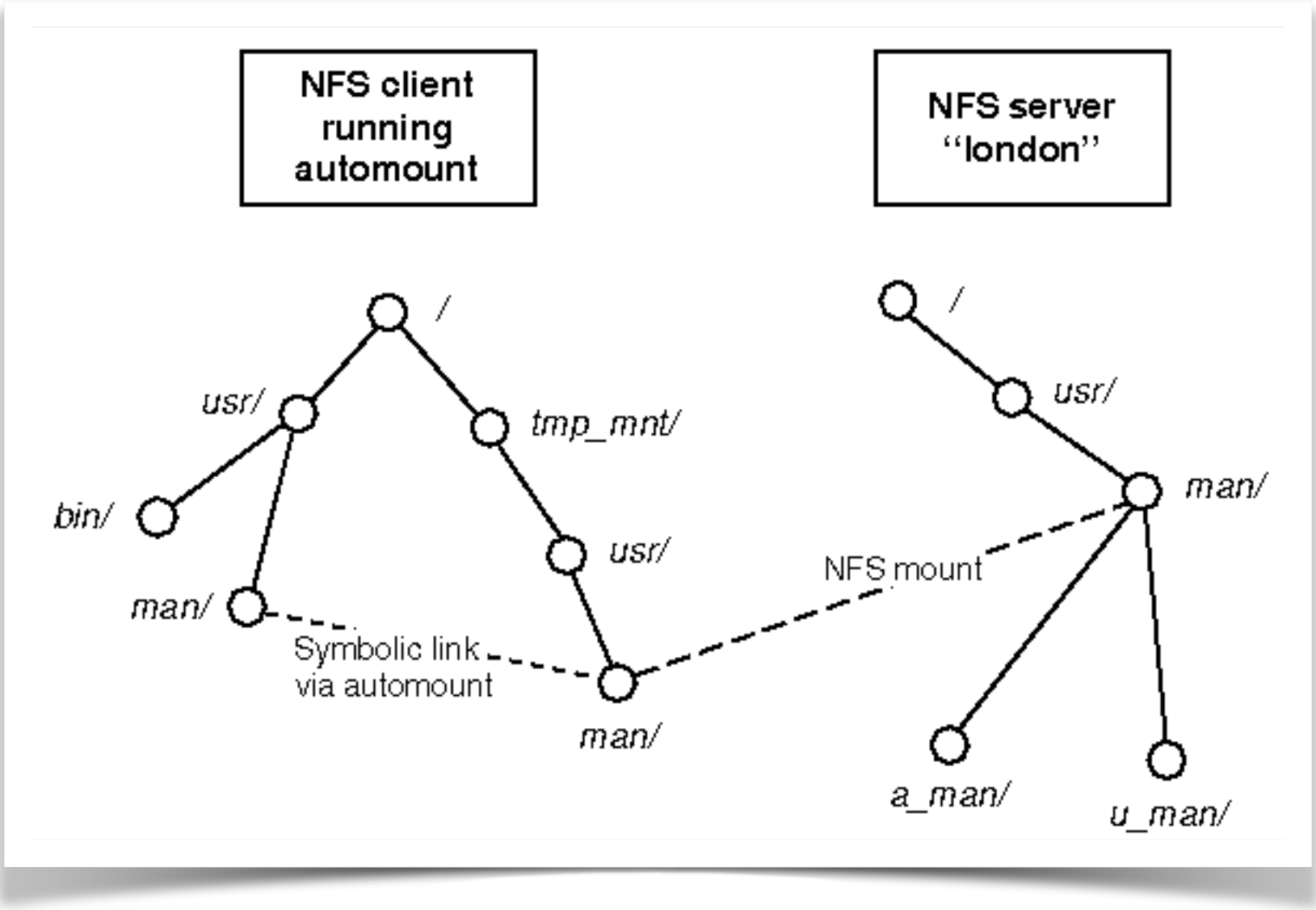
Accès transparent(?) pour les utilisateurs

Sémantique Unix

Performances comparables à des disques locaux

Indépendance de la couche transport

NFS montage

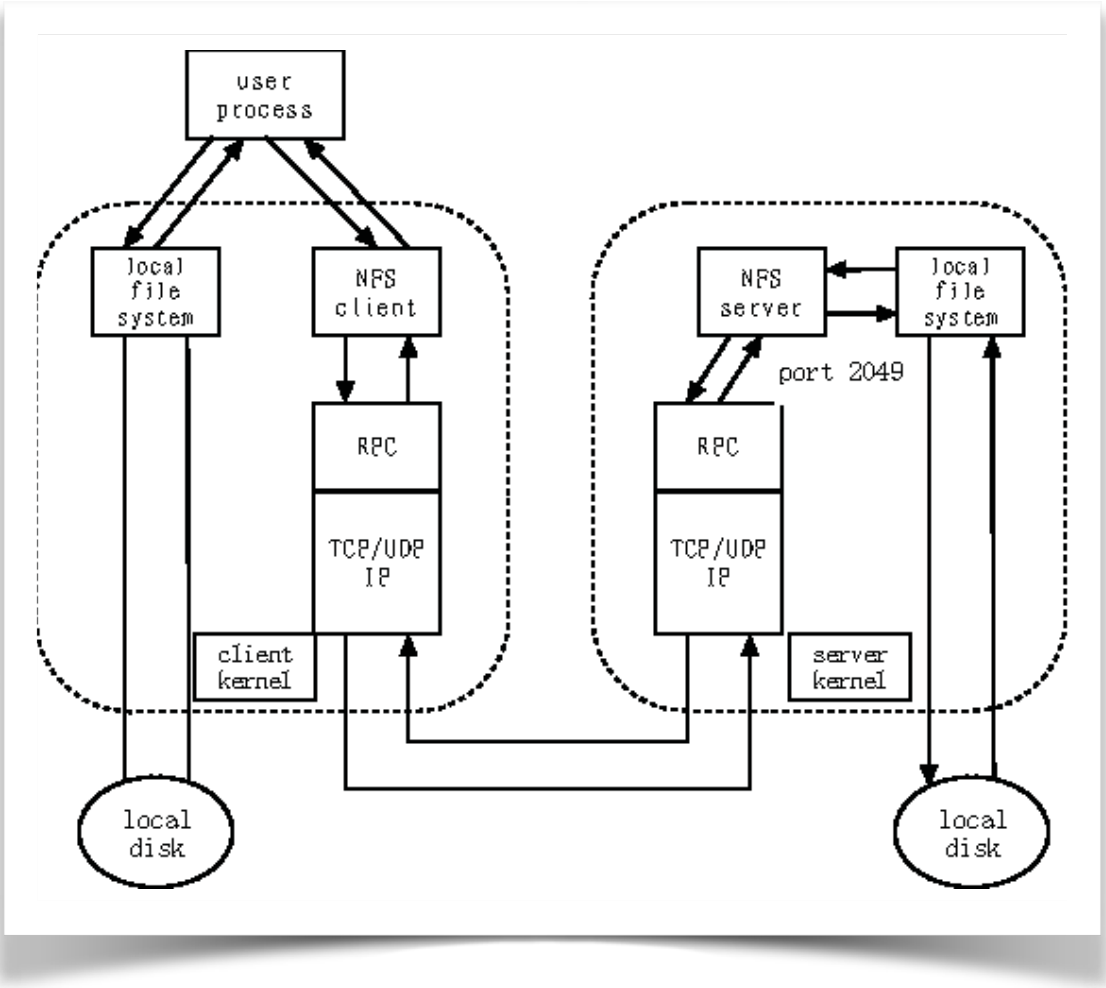

NFS client running automount
NFS server "london"
/
usr/
tmp_mnt/
bin/
usr/
man/
Symbolic link via automount
man/
/
usr/
man/
NFS mount
a_man/
u_man/

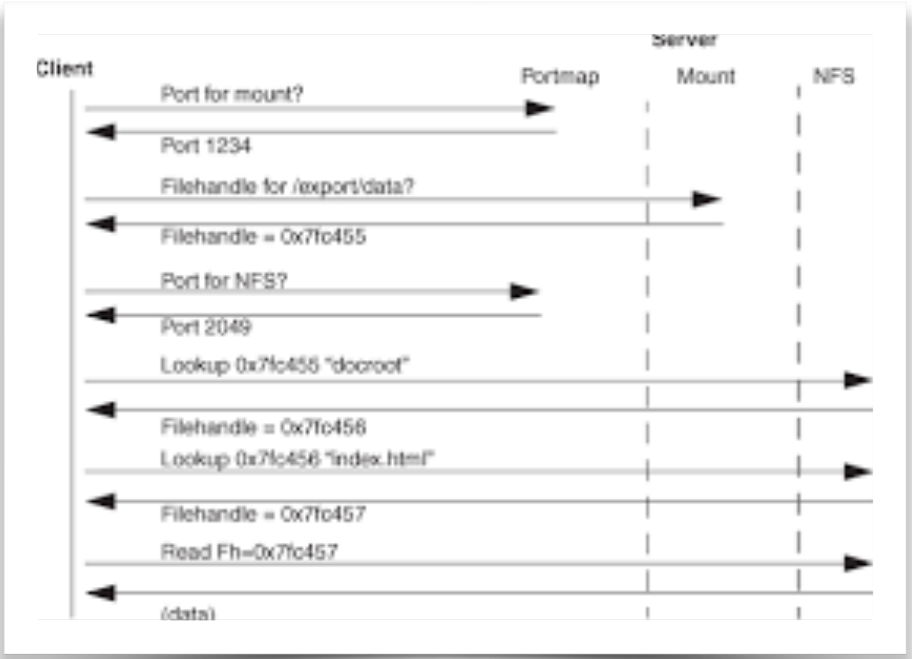
NFS Montage

```
<lucy-1-[18:24]-[~]> df
Filesystem                    1K-blocks       Used  Available Use% Mounted on
…
edo:/vol/etudiants/master2 7140383168 1802946176 5337436992  26% /info/master2
edo:/vol/etudiants/master1 7140383168 1802946176 5337436992  26% /info/master1
edo:/vol/ens               1020054784  231864384  788190400  23% /info/ens
```

```
<lucy-1-[18:24]-[~]> mount
…
edo:/vol/etudiants/master2 on /info/master2 type nfs (rw,…,vers=3,…)
edo:/vol/etudiants/master1 on /info/master1 type nfs (rw,…,vers=3,…)
edo:/vol/ens on /info/ens type nfs (rw,…,vers=3,…)
```

NFS Protocol





NFS

Versions :

- v2 : la version de base
- v3 : 64 bits
- v4 : stateful

SMB

Server Message Block (aka CommonInternetFileSystem)

- protocol stateful

Implémentation libre

- Samba

WebDAV

Web Distributed Authoring and Versioning

Une extension de HTTP pour accéder à des fichiers distants

Filesystem réactifs

C’est la propriété d’un système de gestion de fichiers que de permettre de générer des évènements sur les opérations effectuées

- évènement ☛ code exécuté

Usages

- cloud
- indexation

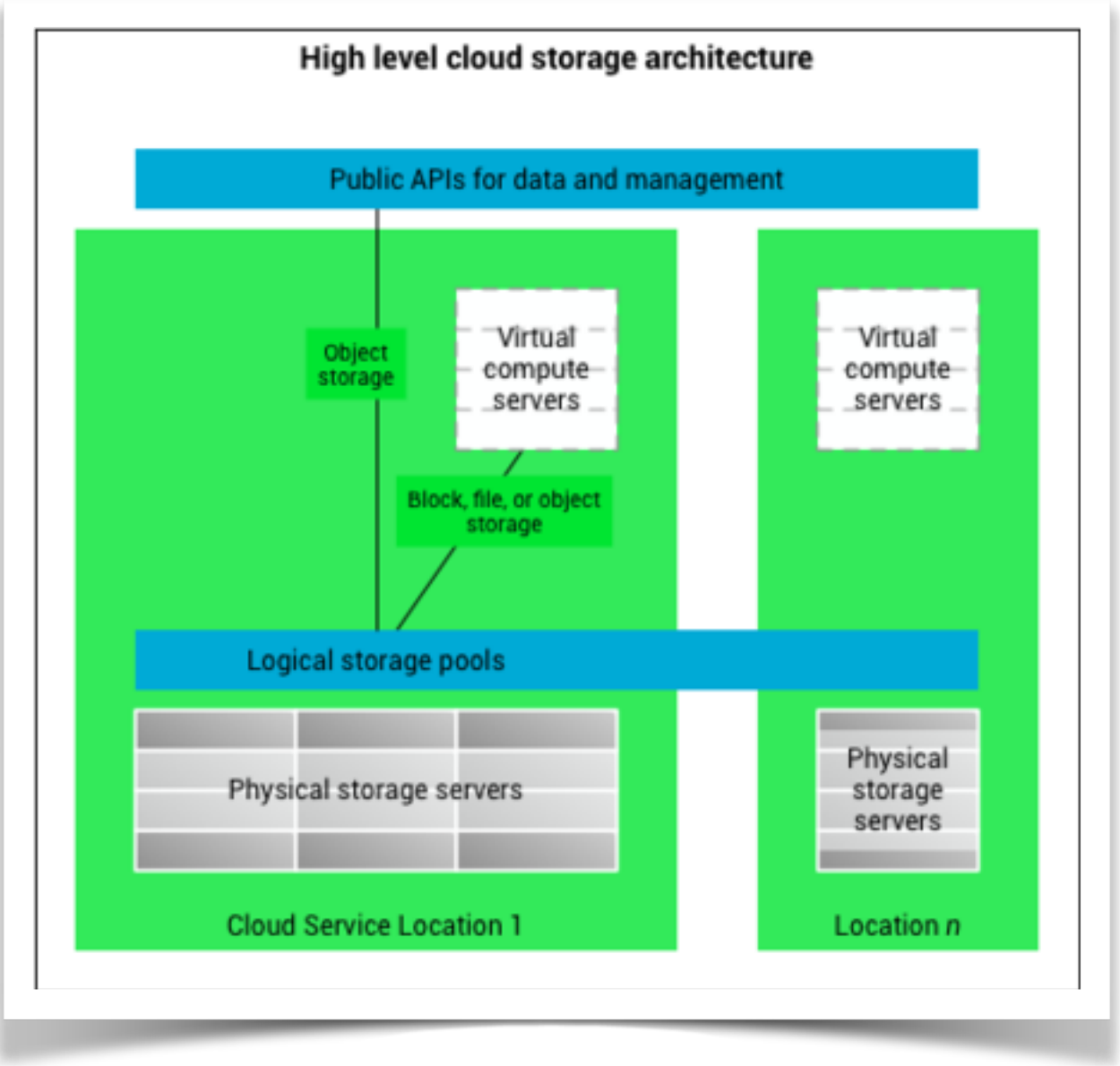
Filesystem réactifs

BSD : kqueue

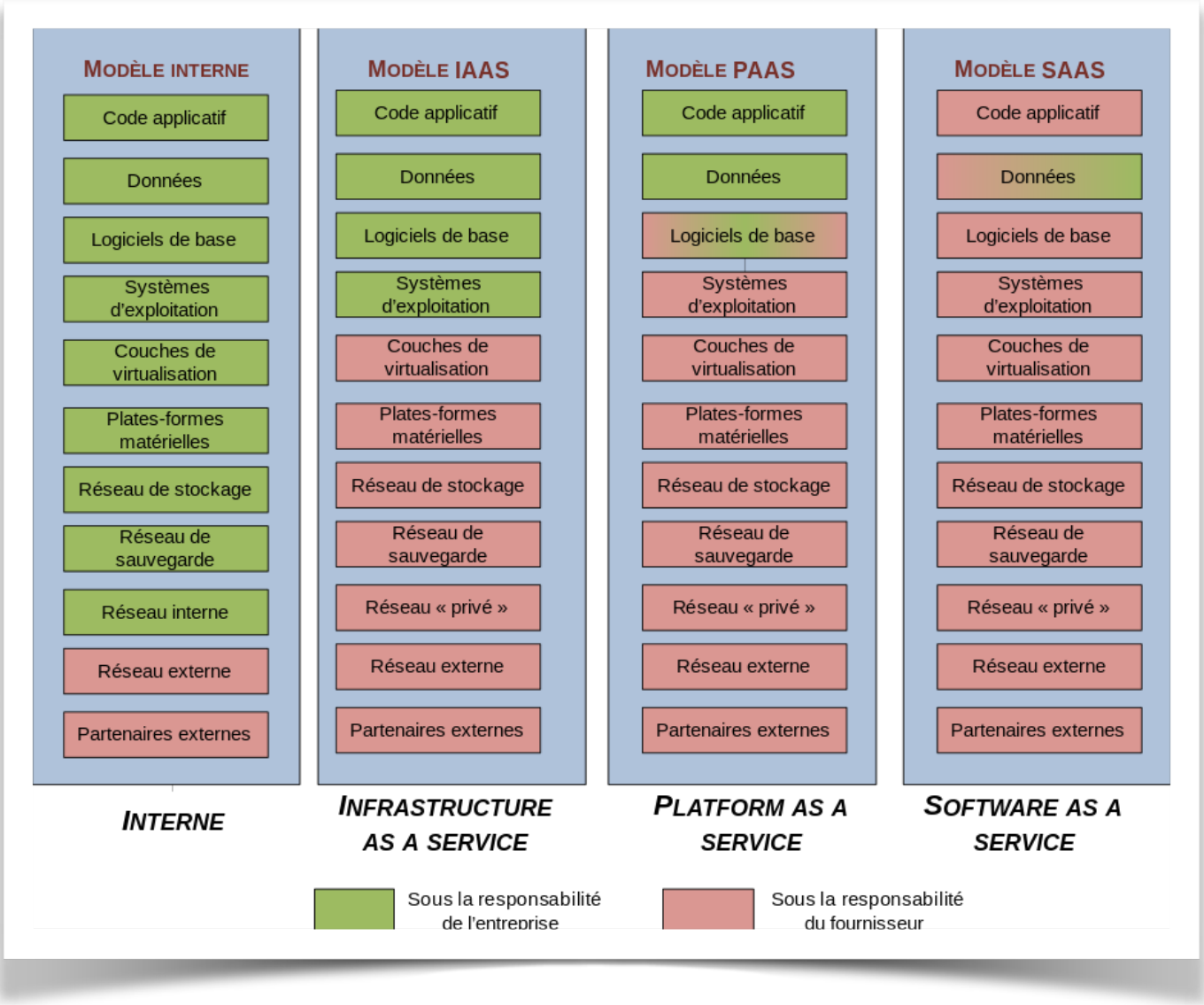
Linux : inotify, fswatch

Apple : kqueue & FSEvents

Cloud


High level cloud storage architecture
Public APIs for data and management
Object storage
Virtual compute servers
Virtual compute servers
Block, file, or object storage
Logical storage pools
Physical storage servers
Physical storage servers
Cloud Service Location 1
Location n

Cloud


MODÈLE INTERNE
Code applicatif
Données
Logiciels de base
Systèmes d'exploitation
Couches de virtualisation
Plates-formes matérielles
Réseau de stockage
Réseau de sauvegarde
Réseau interne
Réseau externe
Partenaires externes
INTERNE
MODÈLE IAAS
Code applicatif
Données
Logiciels de base
Systèmes d'exploitation
Couches de virtualisation
Plates-formes matérielles
Réseau de stockage
Réseau de sauvegarde
Réseau « privé »
Réseau externe
Partenaires externes
INFRASTRUCTURE AS A SERVICE
MODÈLE PAAS
Code applicatif
Données
Logiciels de base
Systèmes d'exploitation
Couches de virtualisation
Plates-formes matérielles
Réseau de stockage
Réseau de sauvegarde
Réseau « privé »
Réseau externe
Partenaires externes
PLATFORM AS A SERVICE
MODÈLE SAAS
Code applicatif
Données
Logiciels de base
Systèmes d'exploitation
Couches de virtualisation
Plates-formes matérielles
Réseau de stockage
Réseau de sauvegarde
Réseau « privé »
Réseau externe
Partenaires externes
SOFTWARE AS A SERVICE
Sous la responsabilité de l'entreprise
Sous la responsabilité du fournisseur

Cloud

Seafile™

ownCloud

SCALITY

Dropbox

iCloud

OneDrive

SPIDEROAK

hubiC

Multi-Cloud

NooBaa

Indexation de fichiers

macOS : Spotlight

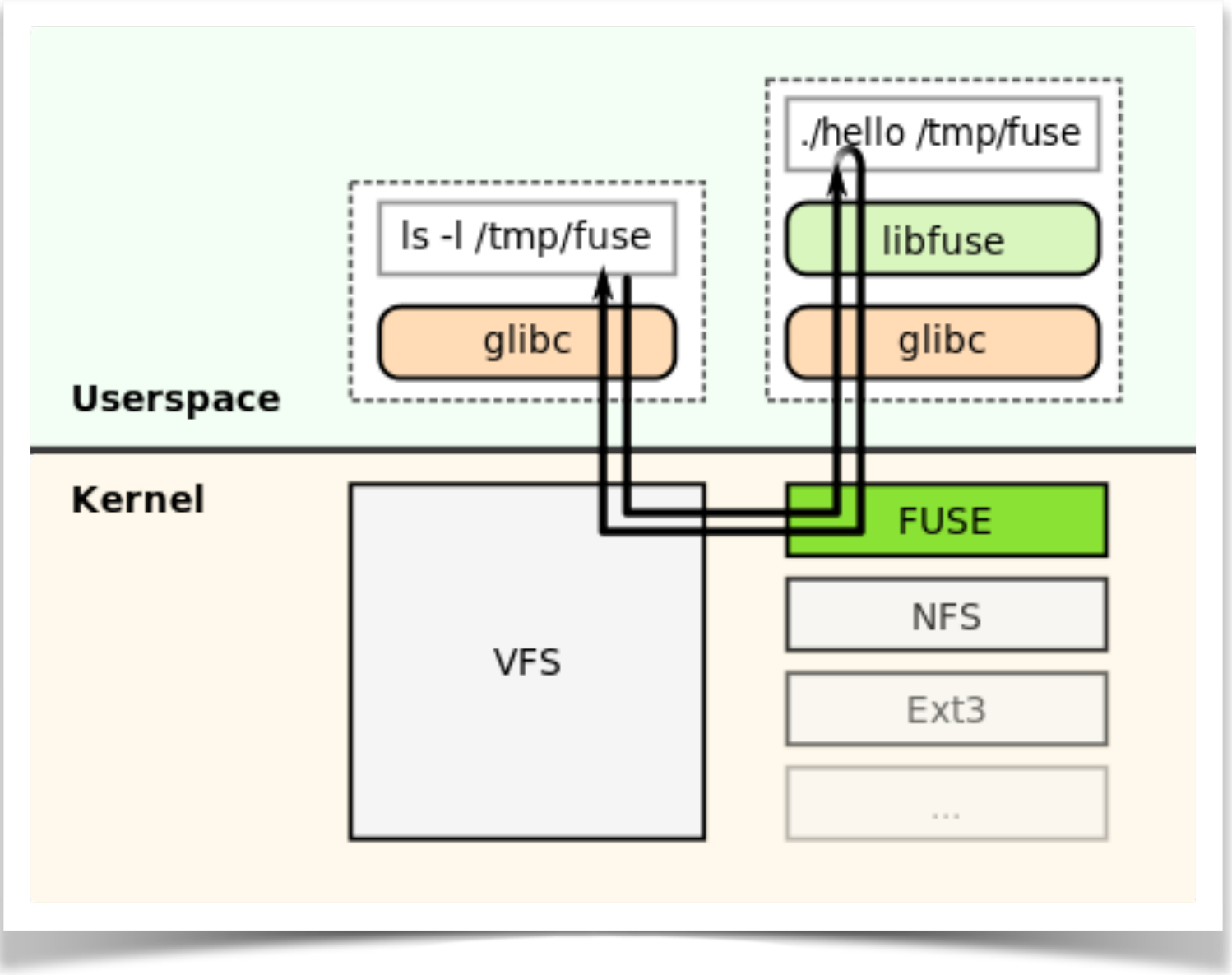
Linux : Tracker, Cerebro

FUSE

Filesystem in USErspace

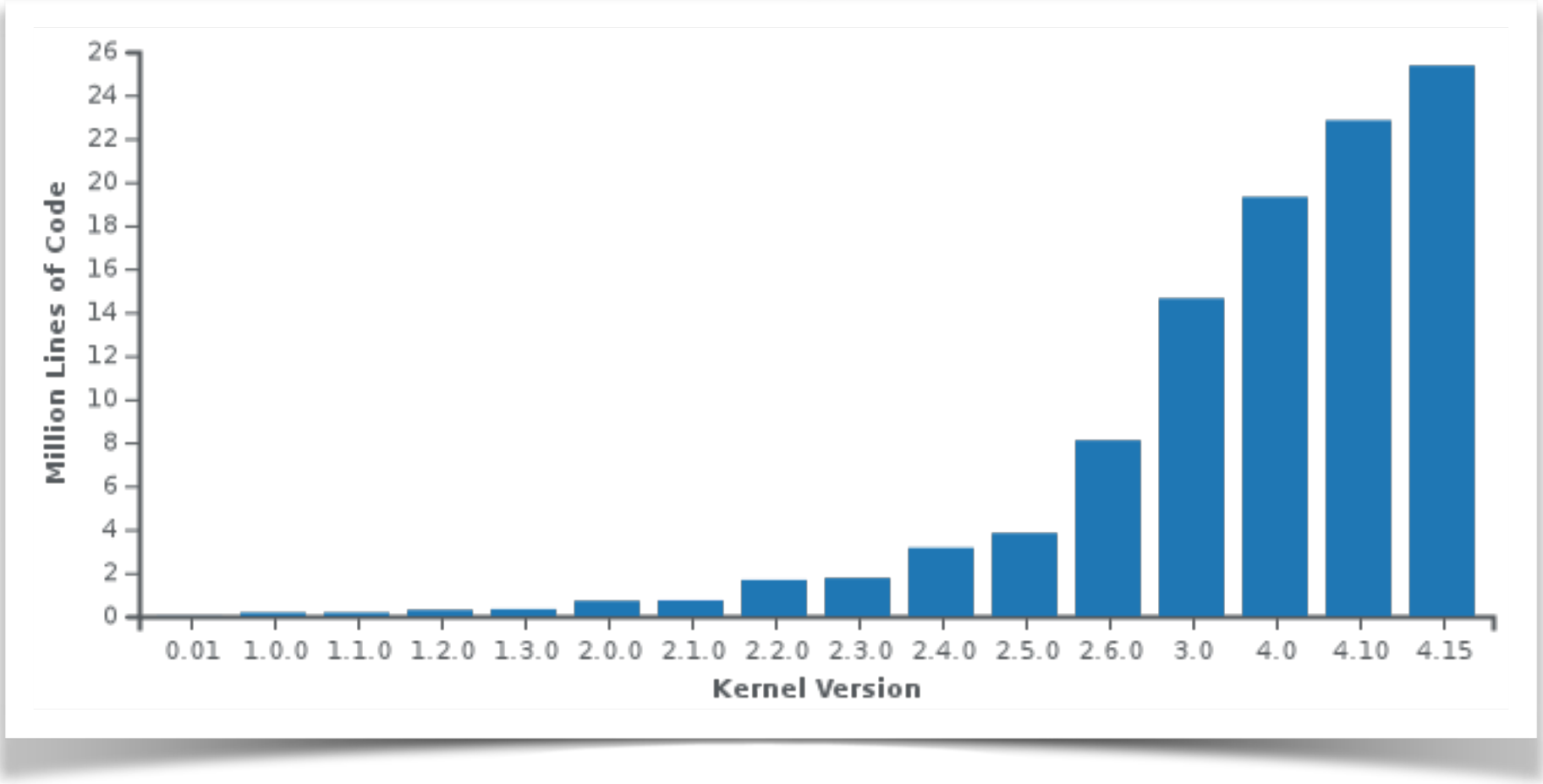
- les systèmes de fichiers correspondent à des modules noyau
- FUSE permet de renvoyer les requêtes correspondant à des E/S vers des processus utilisateurs permettant de les prendre en charge!

FUSE


./hello /tmp/fuse
ls -l /tmp/fuse
libfuse
glibc
glibc
Userspace
Kernel
FUSE
VFS
NFS
Ext3
...

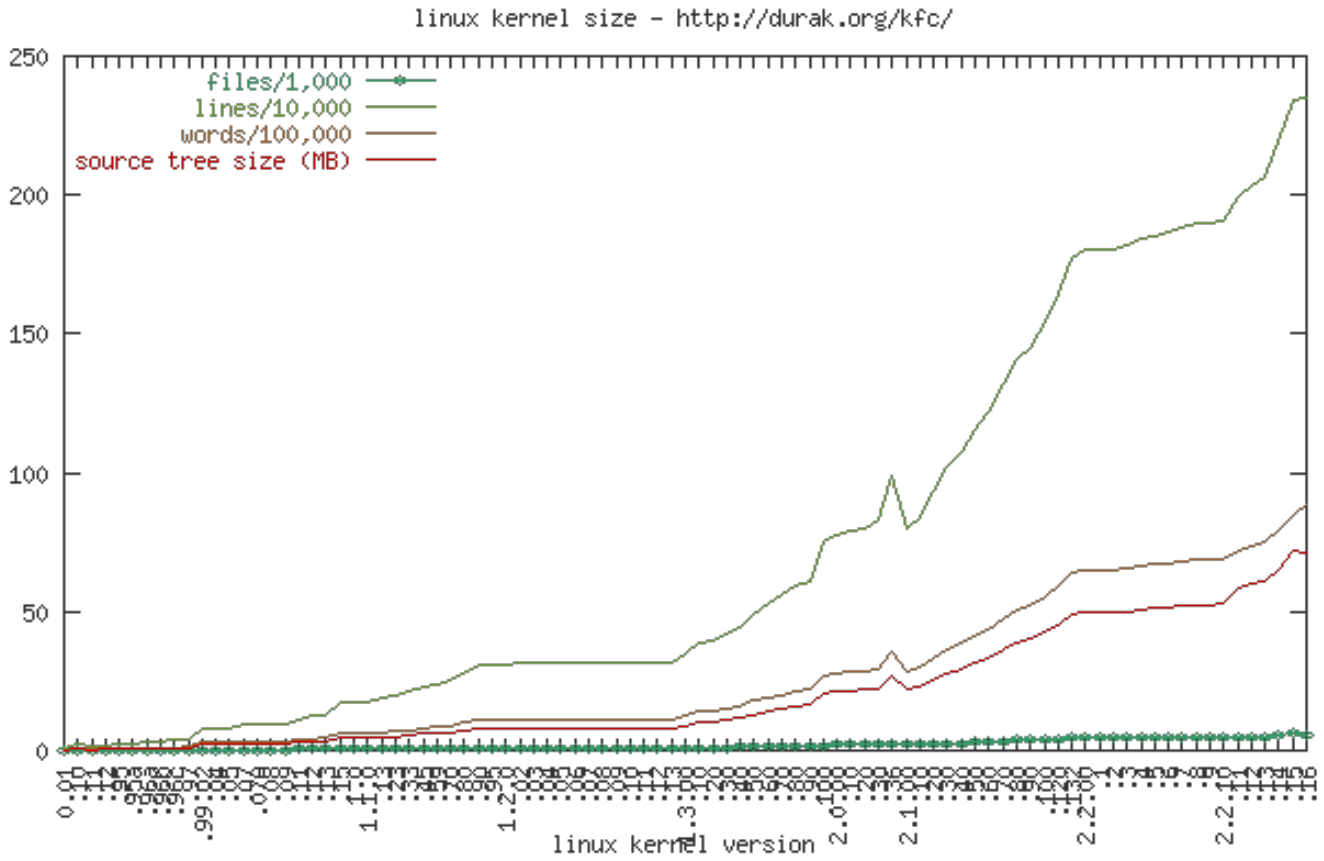
slim fast?

Les systèmes ont tendance à l'embonpoint



slim fast?

Les systèmes ont tendance à l'embonpoint



slim fast?

Les systèmes ont tendance à l'embonpoint

- Monolithes
- deux solutions :
  - modularité
  - micro-noyau

Modularité

Modules Linux

- kernel objects
- lsmod, insmod, rmmod, modprobe,

Kernel Extensions macOS

- kext bundles

Modules BSD

- kernel objects
- kldload, kldunload, kldstat

LKM

«What exactly is a kernel module?
Modules are pieces of code that can be loaded and unloaded into the kernel upon demand.
They extend the functionality of the kernel without the need to reboot the system.
For example, one type of module is the device driver, which allows the kernel to access hardware connected to the system.
Without modules, we would have to build monolithic kernels and add new functionality directly into the kernel image.
Besides having larger kernels, this has the disadvantage of requiring us to rebuild and reboot the kernel every time we want new functionality.»

Excerpt from LKM Programming Guide, 2007
Peter Jay Salzman, Michael Burian, Ori Pomerantz

LKM

Un module n'est pas un programme

Il doit s'intégrer au noyau

On doit l'enregistrer dans le noyau

- `module_init`, `module_exit`

Un module peut faire ce qu'il veut

- mais il travaille dans l'espace du noyau!
- Plus compliqué à programmer mais pas particulièrement difficile non plus

LKM

Différents «types» de modules

- device
- procFS
- syscall hooks
- interrupt catcher
- …

Windows Kernel Extensions

Deux kits à disposition :

- IFSKit
  - File System extensions
- DDK
  - Device Drivers

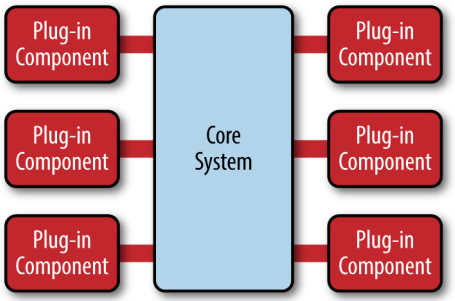
kext

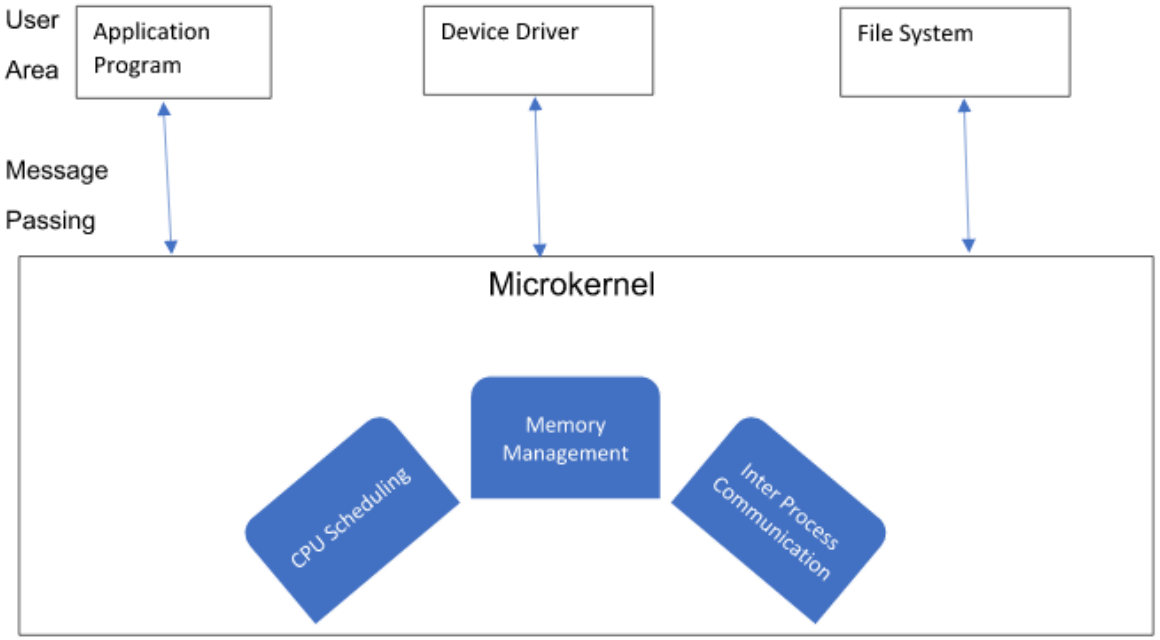
kextstat

Micro-Noyau

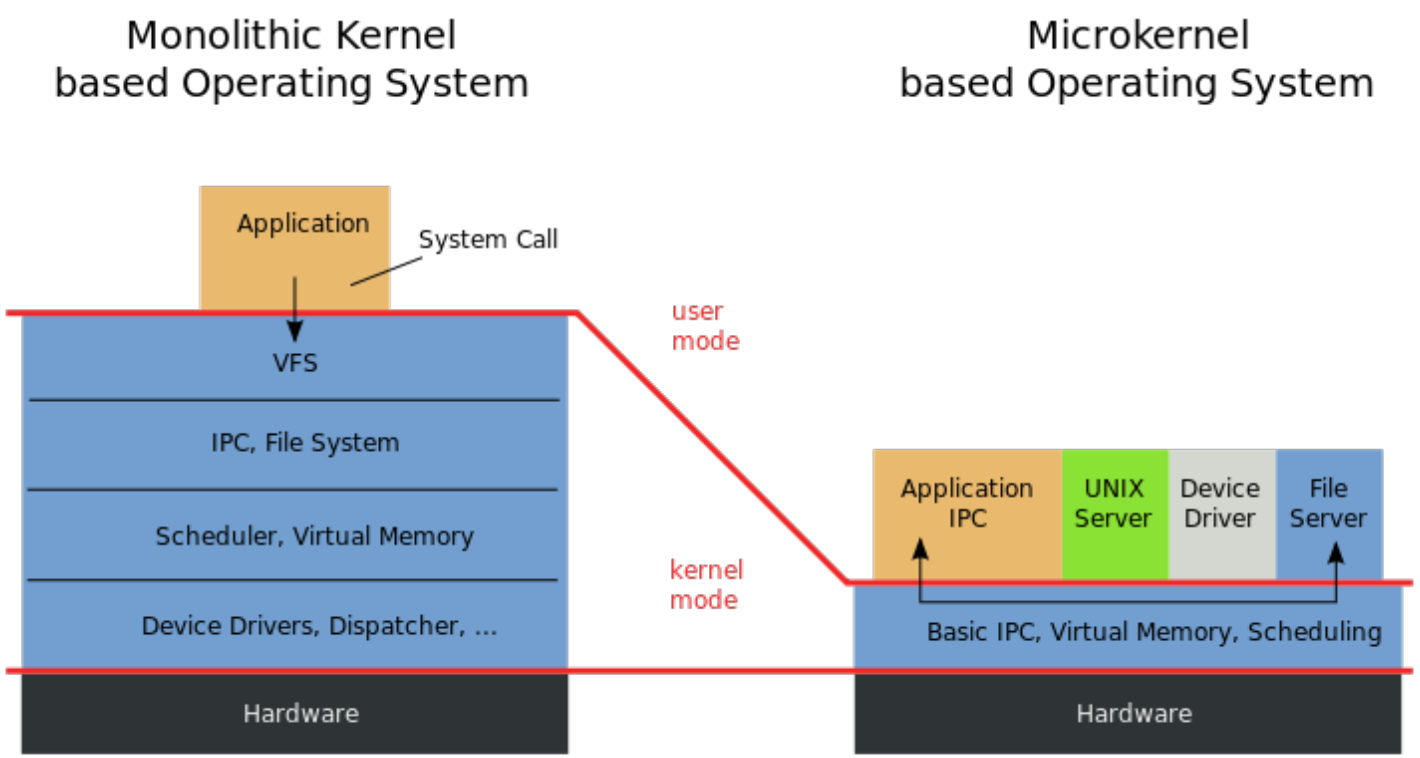
Idée : ne fournir que les services fondamentaux/essentiels

- gestion mémoire
- ordonnancement CPU
- IPC





Micro-Noyau


Monolithic Kernel
based Operating System
Microkernel
based Operating System
Application
System Call
VFS
IPC, File System
Scheduler, Virtual Memory
Device Drivers, Dispatcher, ...
Hardware
user
mode
kernel
mode
Application
IPC
UNIX
Server
Device
Driver
File
Server
Basic IPC, Virtual Memory, Scheduling
Hardware

Micro-noyau

L'idée de base est d'utiliser de la communication entre les entités et non des appels de fonction…

- de la sorte les entités peuvent être dans le noyau ou non cela ne change pas la façon de les utiliser
  - en allégeant le noyau on créé donc des micro-noyaux
    - les avantages sont nombreux, notamment de grandes parties du «système» peuvent être placées en mode utilisateur!
      - le micro noyau n'est rien en tant que tel, on construit au-dessus des personnalités qui peuvent cohabiter!

Mach

Chorus OS

Noyaux désagrégés ?

LegoOS : a disseminated, distributed OS for hardware resource disaggregation

Shan et *al.*, OSDI'18

GUI

Depuis le mitan des années 90 on dispose d'écran graphiques

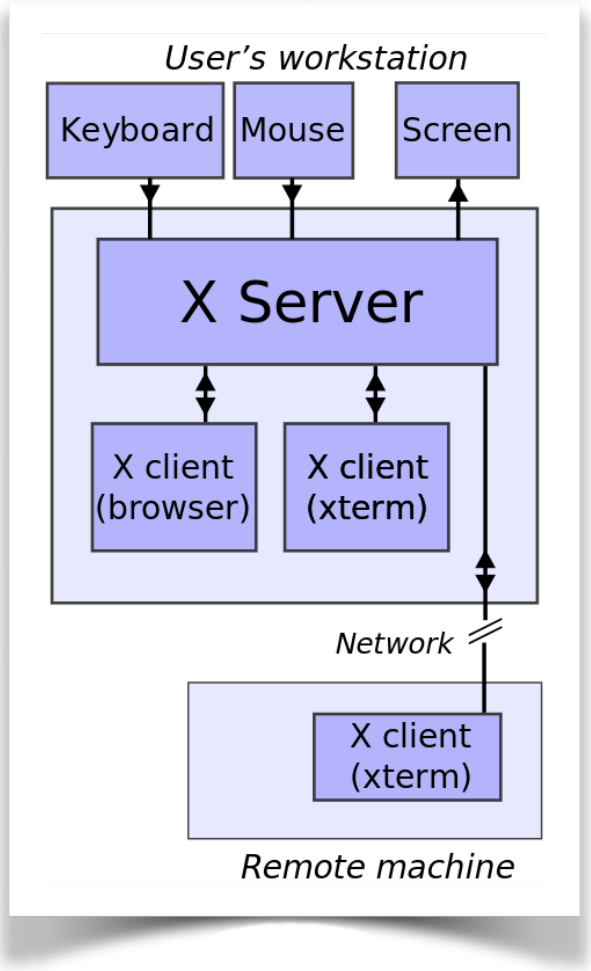
- le développement des interfaces graphiques
  - raster

X11

- X Window System
  - X (1983) ➥ X10 (1986) ➥ X11 (1987) ➥ X11R7.7 (2012)
- C'est LE modèle d'UI d'Unix
  - modèle client-serveur
- L'écran est géré par un serveur, le serveur X avec lequel on interagit via un protocole
  - le protocole X
    - le majeur de la version est le protocole X11 donc…
- Sa succession et en discussion depuis plusieurs années

X11

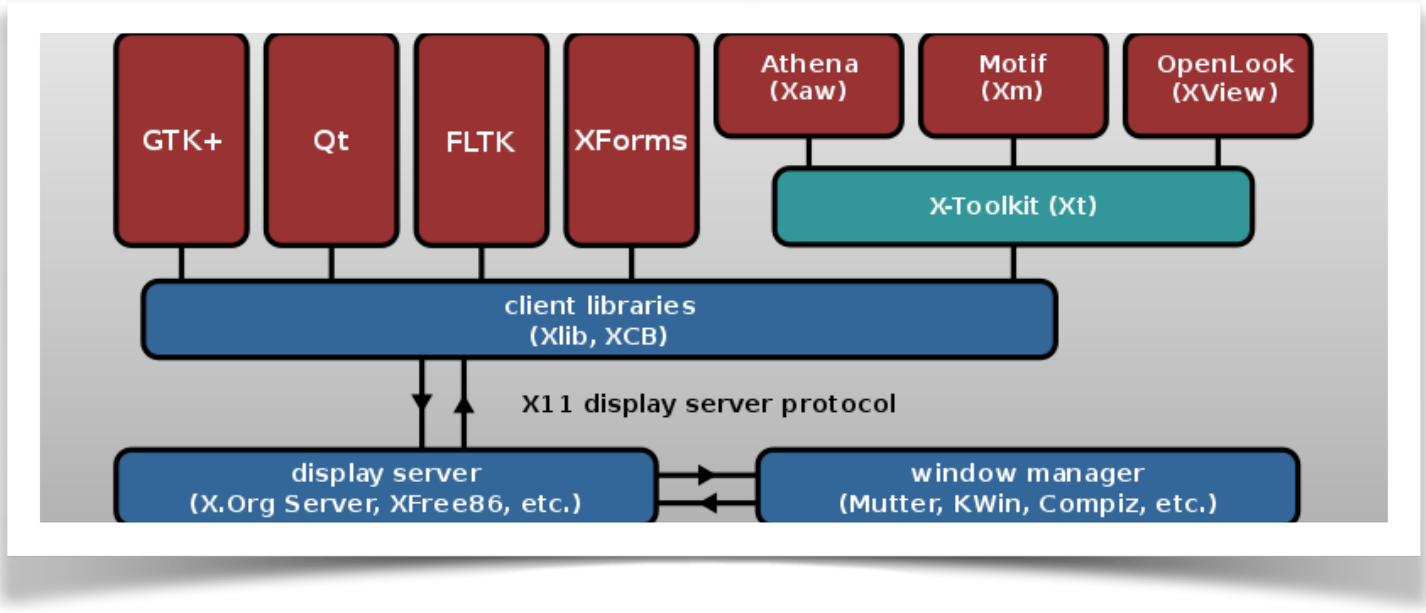
- Déport
  - calcul
  - affichage
- Le modèle du serveur X
  - plusieurs écrans (255?)
  - une souris
  - un clavier



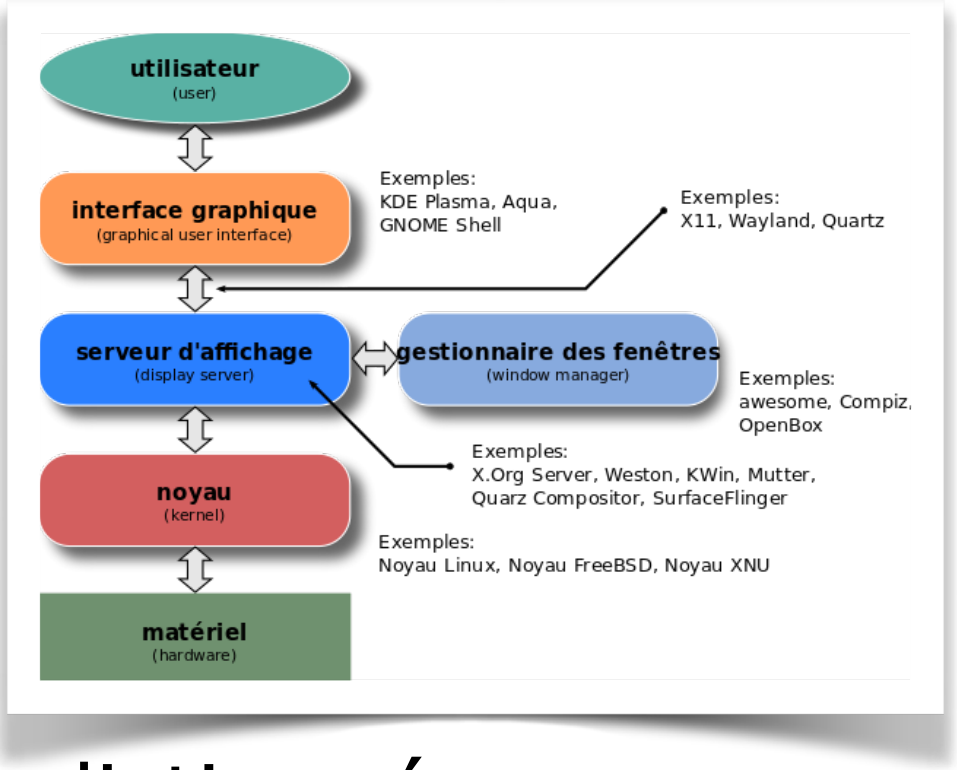
X11

X11 accessible via Xlib est un peu brut

Construction de toolkit (objet)



X11



Parmi les clients l'un est distingué

- le window manager
  - responsable de l'intégrité de la gestion «globale» des fenêtres

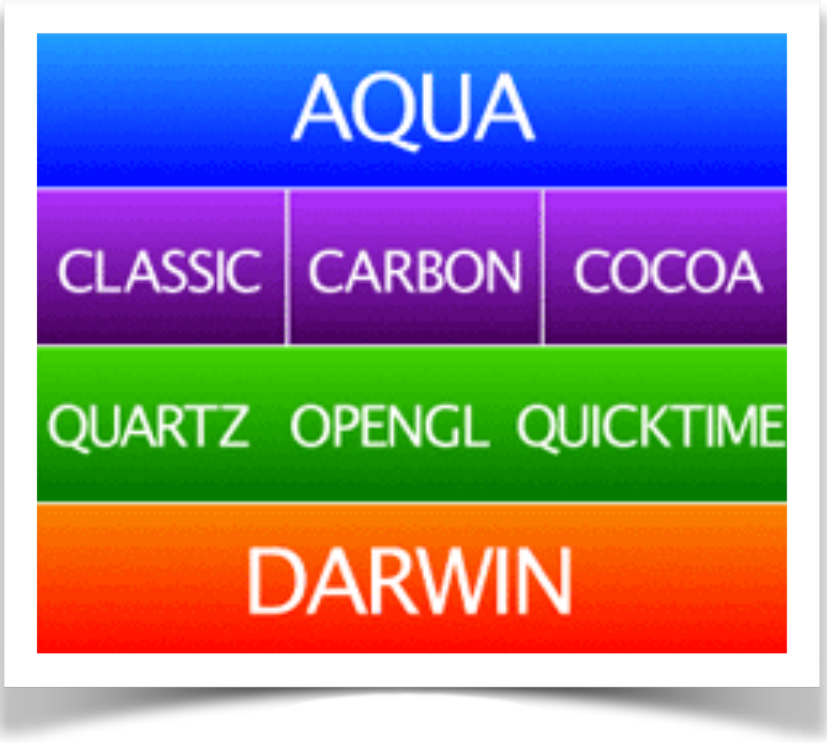
X11

- macOS
  - XQuartz
- Windows
  - *via* Cygwin par exemple

macOS

Le système de fenêtrage natif est assez monolithique et n'utilise pas le modèle client-serveur de X11

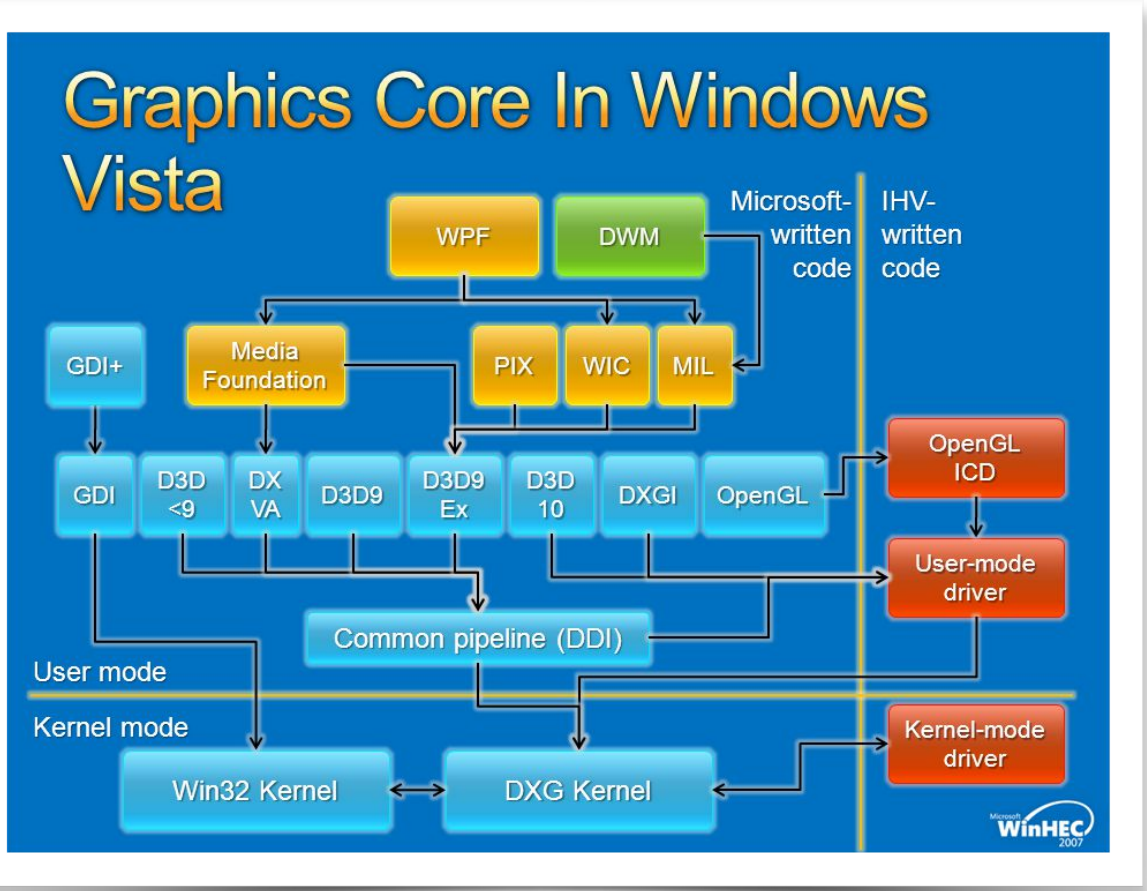
- Quartz le bas-niveau
- Aqua les abstractions



Windows

DWM

Desktop Window Manager



Virtualisation

«Virtualization refers to technologies that are designed to provide a layer of abstraction between layers of hardware and software, as well as layers of software that enables a more simplified interaction with it.»

Paulus Kampert, 2010, Master's Thesis

Virtualisation

Cas d'utilisation : le Data center

- répartir la charge dynamiquement sur l'ensemble des resources physiques
- minimiser le nombre de resources physiques en autorisant leur découpage en resources logiques
- création dynamique de resources logiques

Virtualisation

Cas d'utilisation : la station de travail

- autoriser à exécuter plusieurs OS en même temps
  - éviter le multiboot…

Virtualisation

Cas d'utilisation : le mobile

- autoriser à exécuter plusieurs OS en même temps
  - un pour la téléphonie (en général avec des caractéristiques temps-réel)
  - un pour le *smart* (les apps)

Virtualisation

Cas d'utilisation : l'automobile

- autoriser à exécuter plusieurs OS en même temps
  - un pour le contrôle (en général avec des caractéristiques temps-réel)
  - un pour le *smart* (radio, lcd, GPS, etc)

Virtualisation

Cas d'utilisation : la migration

autoriser l'exécution d'applications écrites pour une version plus ancienne de l'OS sur une nouvelle version

Ex. : MacOS Classic on OSX (supporté pendant plusieurs années) 68k sur PPC, MacOS early versions on MacOS late versions PPC sur Intel.

Virtualisation

Cas d'utilisation : la conservation

- autoriser à exécuter d'anciens OS/logiciels conçus sur des plateformes disparues
  - Vous connaissez MAME ?
    - Plus de machines
    - Ri code sources
      - reste l'émulation logicielle permettant d'exécuter d'anciennes ROM/EEPROM

Virtualisation

3 classes :

- Agrégation
  - N resources vues comme 1
- **Partition**
  - **1 resources vue comme N**
- Émulation
  - resource X vue comme resource Y

Virtualisation

Les bénéfices :

- minimisation des resources physiques
- économie d'énergie
- gestion facilitée (gestion logicielle vs physique)
- tolérance aux pannes : temps de recouvrement, sauvegardes

Virtualisation

Virtual Networking?

UML?

IBM/VM?

Virtual Server?

VMware?

Transitive QuickTransit?

Java, JVM?

Application Virtualization?

Pascal Pcode?

Platespin?

AMD SVM?

Virtual Solutions?

TransMeta Crusoe?

SIMICS?

Softricity?

QEMU?

KVM?

Dynamo?

VirtualPC?

Virtual Reality?

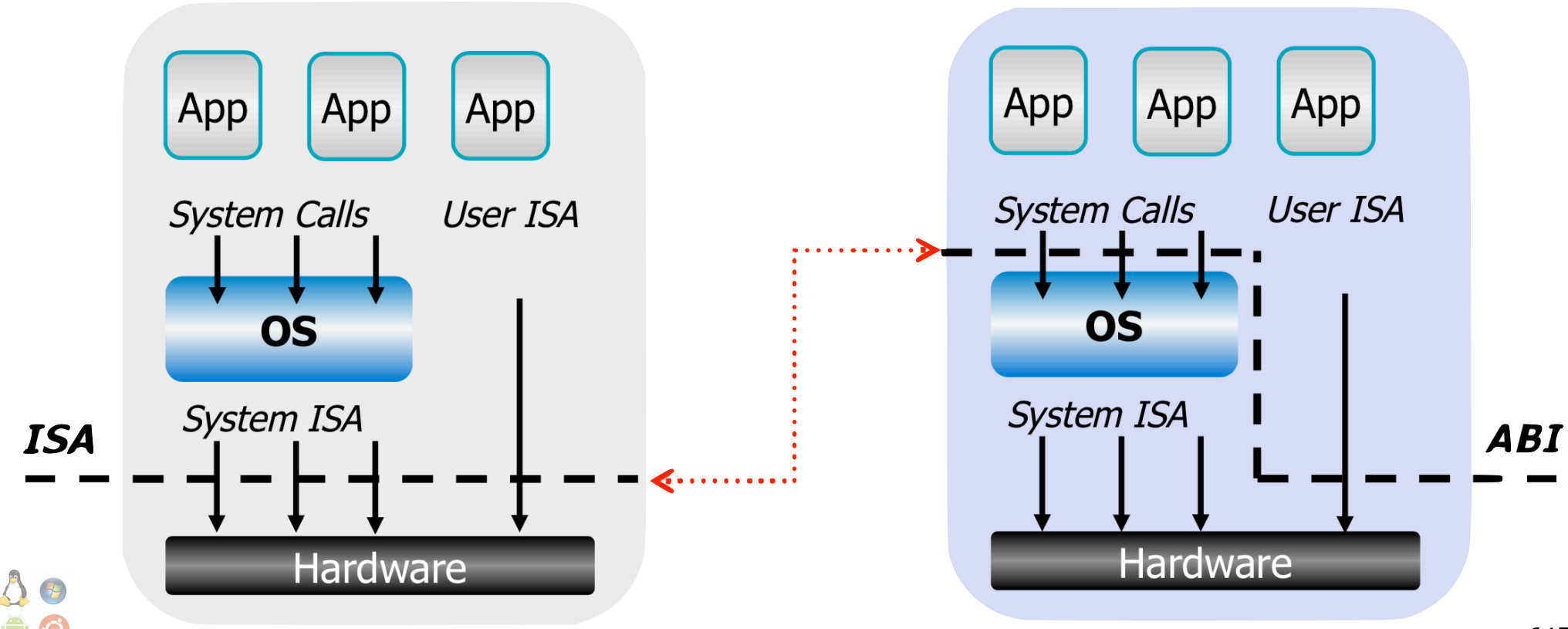
Virtual Storage?

FX!32?

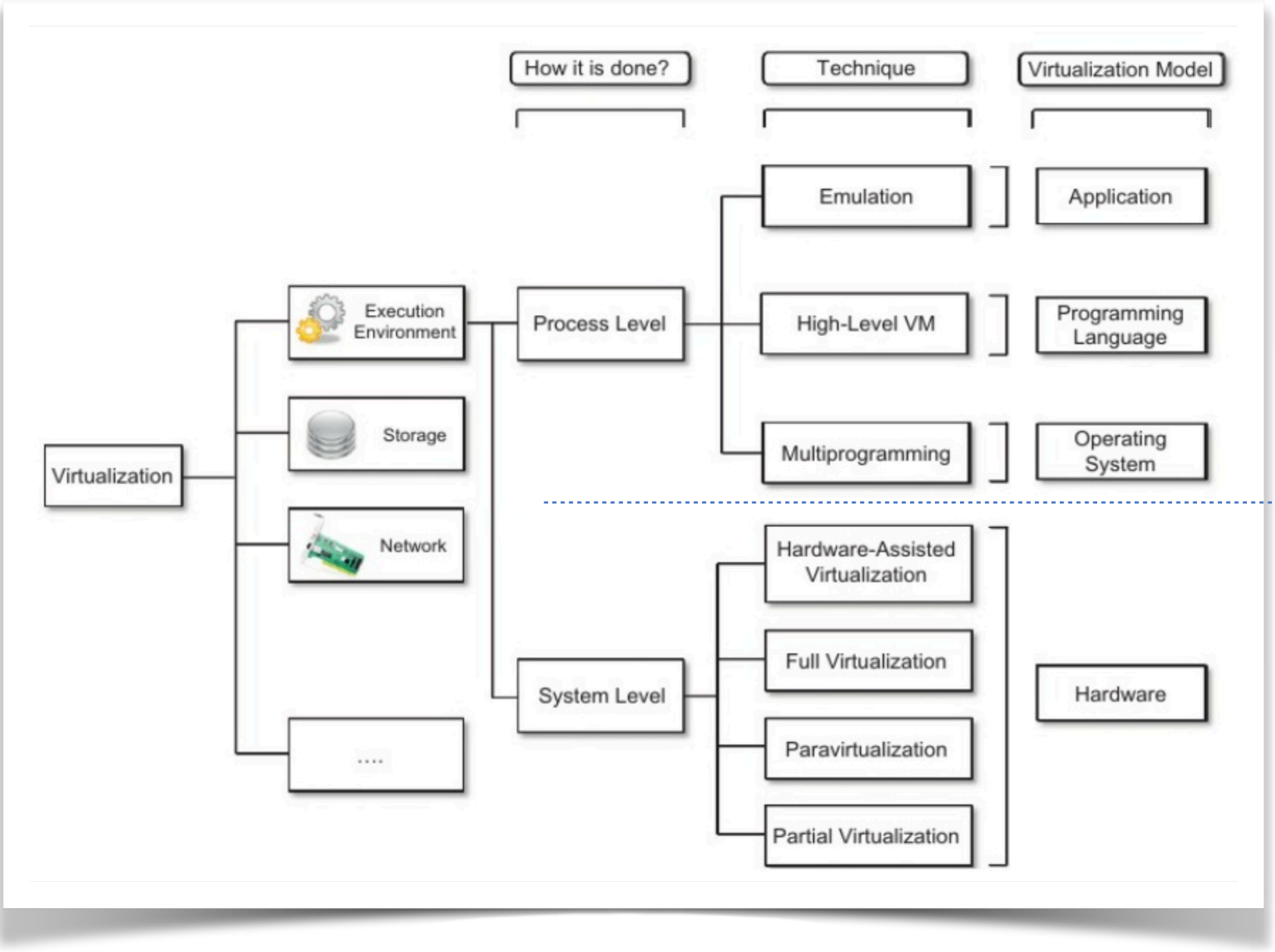
Virtualisation

System level VM : ISA

Process level VM : ABI



Virtualisation



Virtualisation système

Problèmes :

- Les OS sont conçus pour tourner seuls sur une machine
- Il faut partager les resources sensibles : CPU, mémoire, périphériques, etc

Résolution des conflits dans le moniteur/ hyperviseur

- détection des instructions sensibles puis simulation de leur comportement

Virtualisation système

Problème :

exécuter un OS dans un mode moins privilégié que celui pour lequel il a été conçu

Virtualisation transparente :

pas de modification de l'OS

processeur virtualisable (Intel VT-x, AMD-V)

traduction dynamique

Para-virtualisation

modification du HAL de l'OS

Virtualisation système

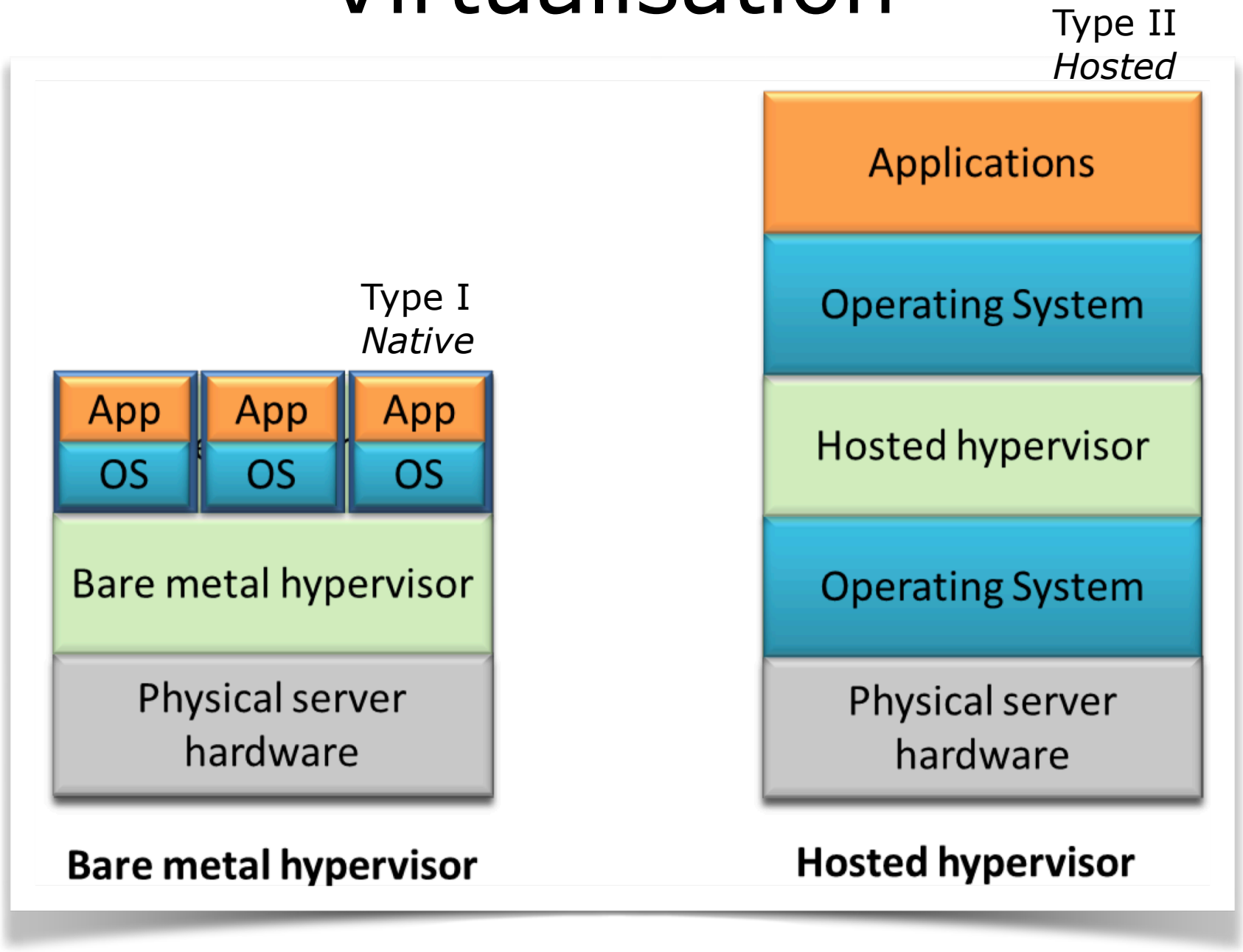
Hyperviseur :

- recrée un environnement d'exécution

Deux types :

- Type I
  - directement sur la machine physique (Bare Metal, native)
- Type II
  - par-dessus un OS (Hosted)

Virtualisation



Virtualisation système

VMM (Virtual Machine Monitor ou Hyperviseur), trois composants :

- Dispatcheur
  - multiplexe les VM
- Allocateur
  - gère les ressources associées à une VM
- Interpréteur
  - réalise les instructions privilégiées de la VM

Virtualisation système

Critère de qualité des VMM

équivalence ou fidélité : même comportement fourni que si exécuté directement

contrôle ou sûreté : les resources doivent être sous le contrôle total du moniteur

efficacité ou performance : un maximum d'instruction à exécuter doivent être exécutée nativement

«Formal requirements for virtualizable third generation architectures»
Communication of the ACM, Popek et Goldberg, 1974

Virtualisation système

Rappel (?) :

ISA 4 catégories d'instructions

non privilégiées : ordinaires

privilégiées : celles qui génère une faute en user mode

contrôles (sensible) : celles qui modifient la configuration des resources

comportementales (sensible) : dont le résultat dépend de la configuration des resources

Virtualisation

Quelles instructions doivent se comporter différemment en user mode et kernel mode ?

> «In all three cases, only privileged applications (usually the OS kernel) may modify IF. Note that this only applies to protected mode code. (Real mode code may always modify IF.)
> CLI and STI are privileged instructions, which trigger a general protection fault if an unprivileged application attempts to execute it, while POPF will simply not modify the IF flag if the application is unprivileged.»
>
> Wikipedia, «Interrupt flag»

Virtualisation

Quelles instructions doivent se comporter différemment en user mode et kernel mode ?

- MMU
  - CR3 devrait renvoyer des infos virtuelles et pas physiques dans le cadre de la virtualisation…

Virtualisation système

Théorème :

Pour toute architecture conventionnelle un Hyperviseur peut être construit si l'ensemble des instructions sensibles est un sous-ensemble des instructions privilégiées

En effet, on peut les trapper en mode utilisateur et donc opérer la correction nécessaire

Full virtualization

Lorsque l'OS peut-être exécuté sans aucune modification

- nécessite une émulation complète de la machine

Difficulté

- interception des instructions privilégiées

Full virtualization

Hyperviseur dans le mode le plus bas

OS invité au niveau supérieur

L'hyperviseur effectue (si nécessaire) une traduction dynamique et trappe les instructions privilégiées

Virtualisation

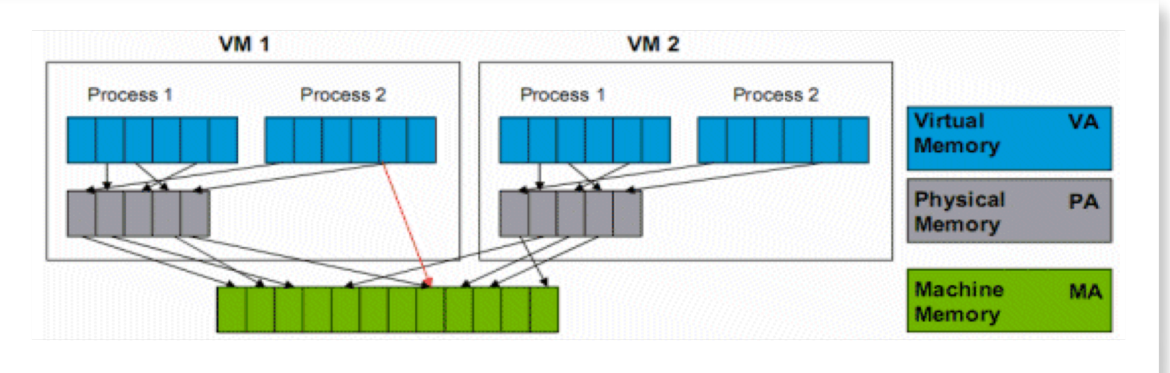
- La traduction dynamique
  - le moniteur «réécrit» les instructions privilégiées qui auraient été exécutées silencieusement
    - cache de traduction
    - impact mémoire
    - problème de déterminisme temporel
- Complexe à réaliser

Hardware-assisted Virtualization

- Support machine pour la réalisation de la virtualisation
  - permet d'éviter ou simplifier l'émulation de certaines instructions privilégiées
    - Intel VT-x, AMD-V

Hardware-assisted Virtualization

- MMU distinction entre
  - Mémoire Virtuelle (process)
  - Mémoire Physique (VM)
  - Mémoire Machine (Machine)
- TLB ASID tag (Address Space IDentifier)



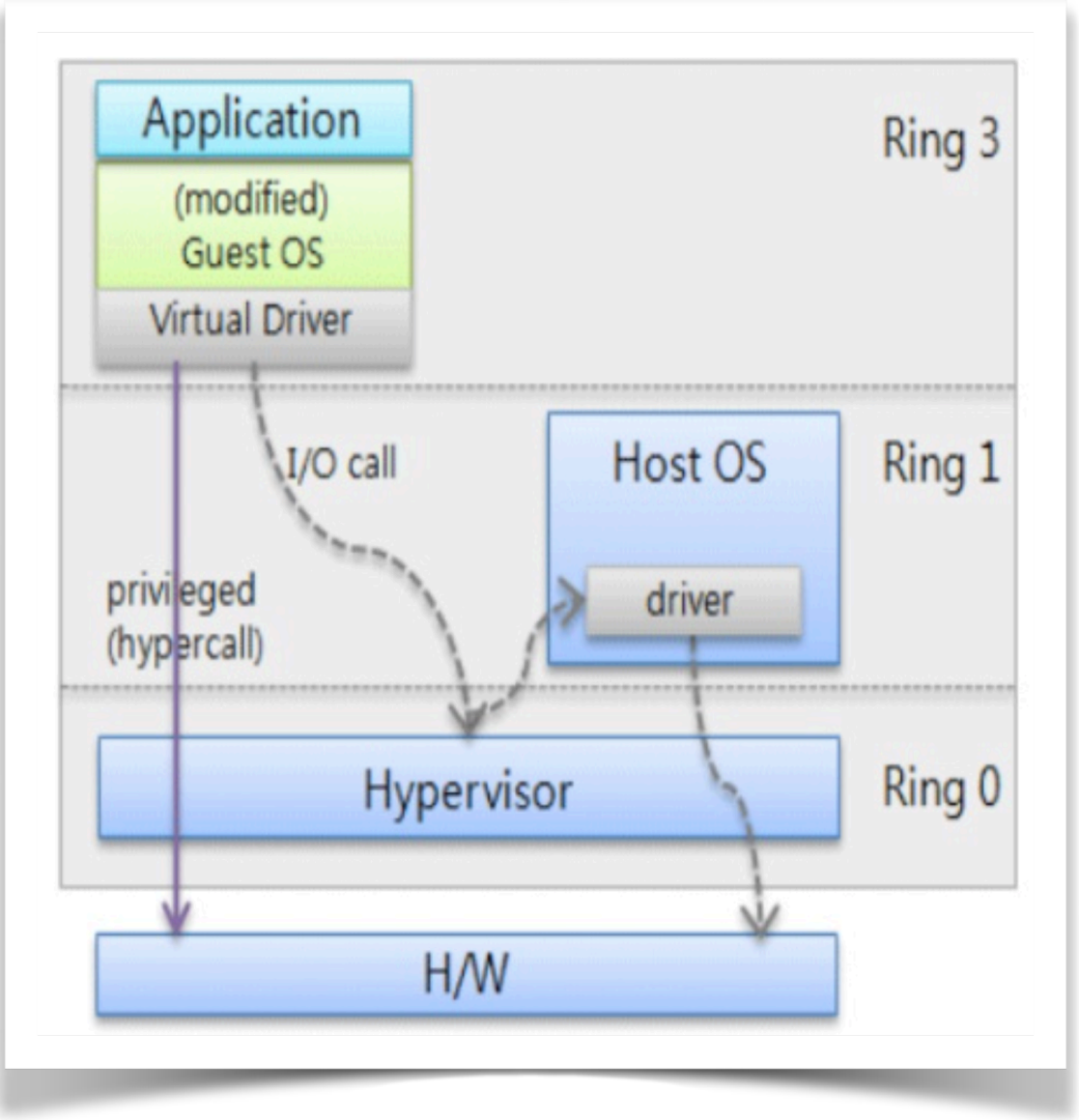
Hardware-assisted Virtualization

- La machine offre un niveau inférieur au niveau 0 dans lequel s'exécute l'hyperviseur
  - le contrôle de la virtualization en est donc très simplifié puisque l'OS s'exécute dans son mode d'origine!

Virtualisation

- Para-virtualisation
  - modifier le HAL du système invité
  - OS s'exécute en mode moins privilégié que d'habitude (ring 1 intel, user mode ARM)
- Meilleures performances
  - mais intrusif!
    - une solution employée que lorsqu'il n'y a pas de support hardware et que la modification de l'OS est possible

Para-virtualization



Partial virtualization

Ne fournit qu'une émulation partielle du monde physique

- un OS ne peut probablement pas s'y exécuter
- mais certaines applications (celles qui n'utilisent que la partie émulée)

Virtualisation

La virtualisation des périphériques : le réseau!

- Différents modes
  - NAT
    - translation d'adresses/ports, l'invité est invisible de l'extérieur (avantage ? inconvénient ?)
  - Bridged
    - multiplexage dans le driver hôte où le moniteur
  - Internal
    - simulation d'un réseau entre VMs
  - Host
    - simulation d'un réseau entre l'hôte et les VMs

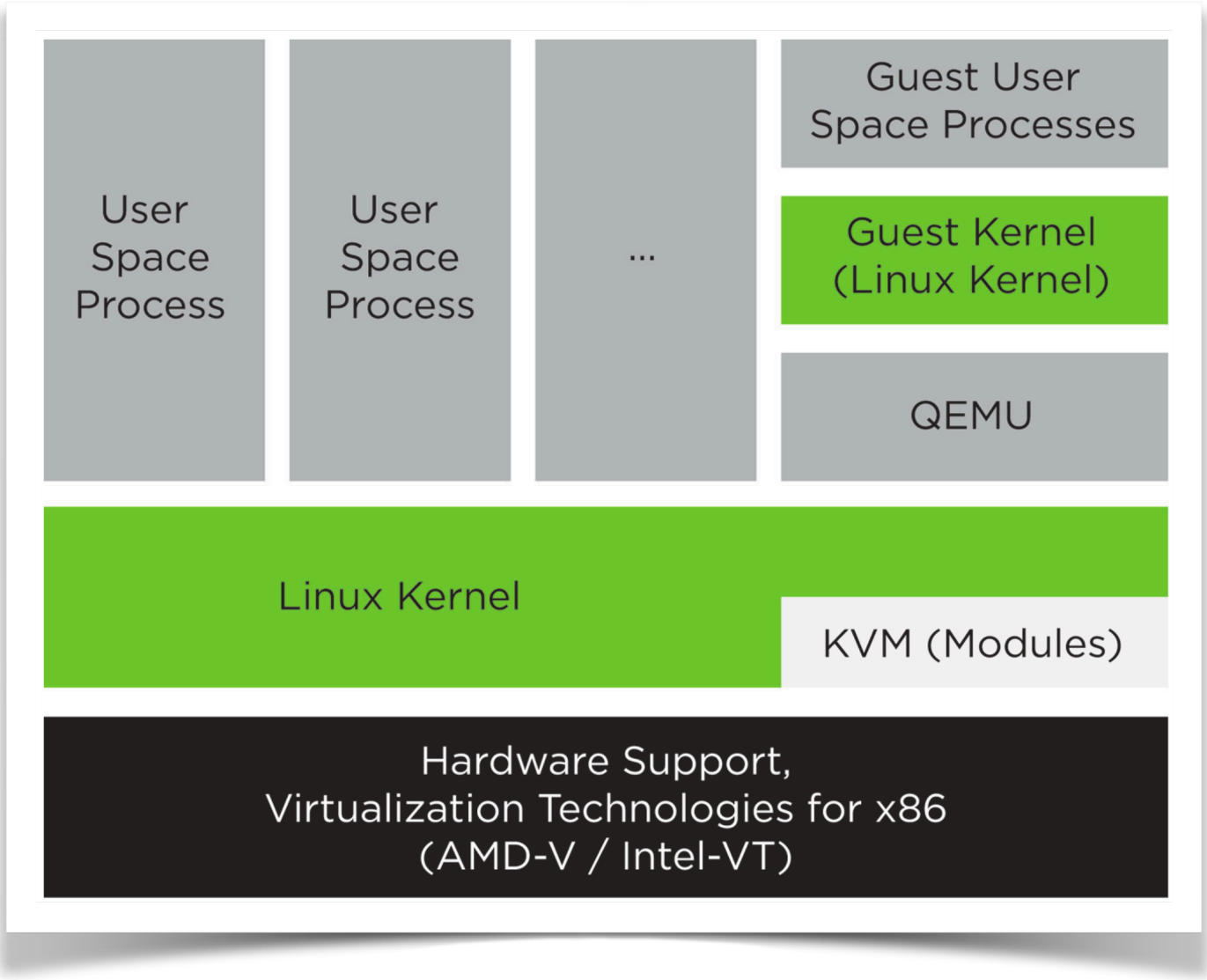
KVM

Kernel-based Virtual Machine

KVM est un module noyau Linux (mais existe sur d'autres : FreeBSD) permettant au noyau de se comporter comme un Hyperviseur de virtualisation complète

KVM est un logiciel d'émulation basé sur QEMU et qui permet d'émuler une machine

KVM


User Space Process
User Space Process
...
Guest User Space Processes
Guest Kernel (Linux Kernel)
QEMU
Linux Kernel
KVM (Modules)
Hardware Support, Virtualization Technologies for x86 (AMD-V / Intel-VT)

QEMU

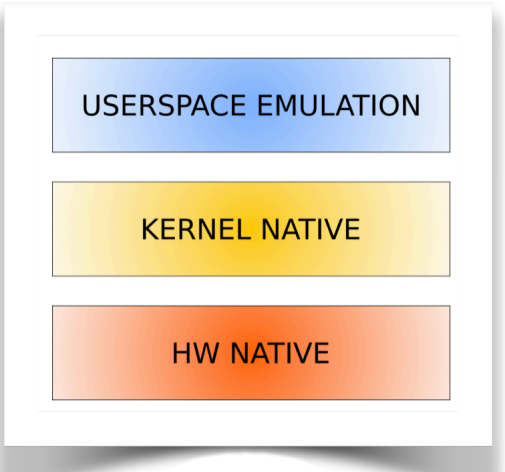
Quick EMUlator

un hyperviseur de machine virtuelle utilisant la traduction dynamique mais aussi le support natif de virtualisation si possible

QEMU

Plusieurs modes de fonctionnement

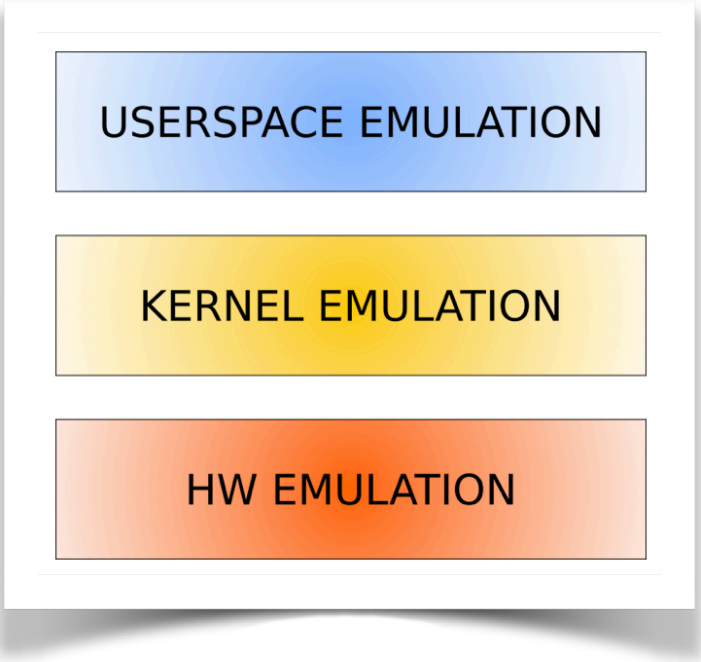
- user-mode emulation
  - exécution d'applications pour une architecture différente, un programme pour Linux/ARM exécuté sur Linux/Intel par exemple.
  - émulation du CPU, les appels systèmes sont les mêmes



QEMU

Plusieurs modes de fonctionnement

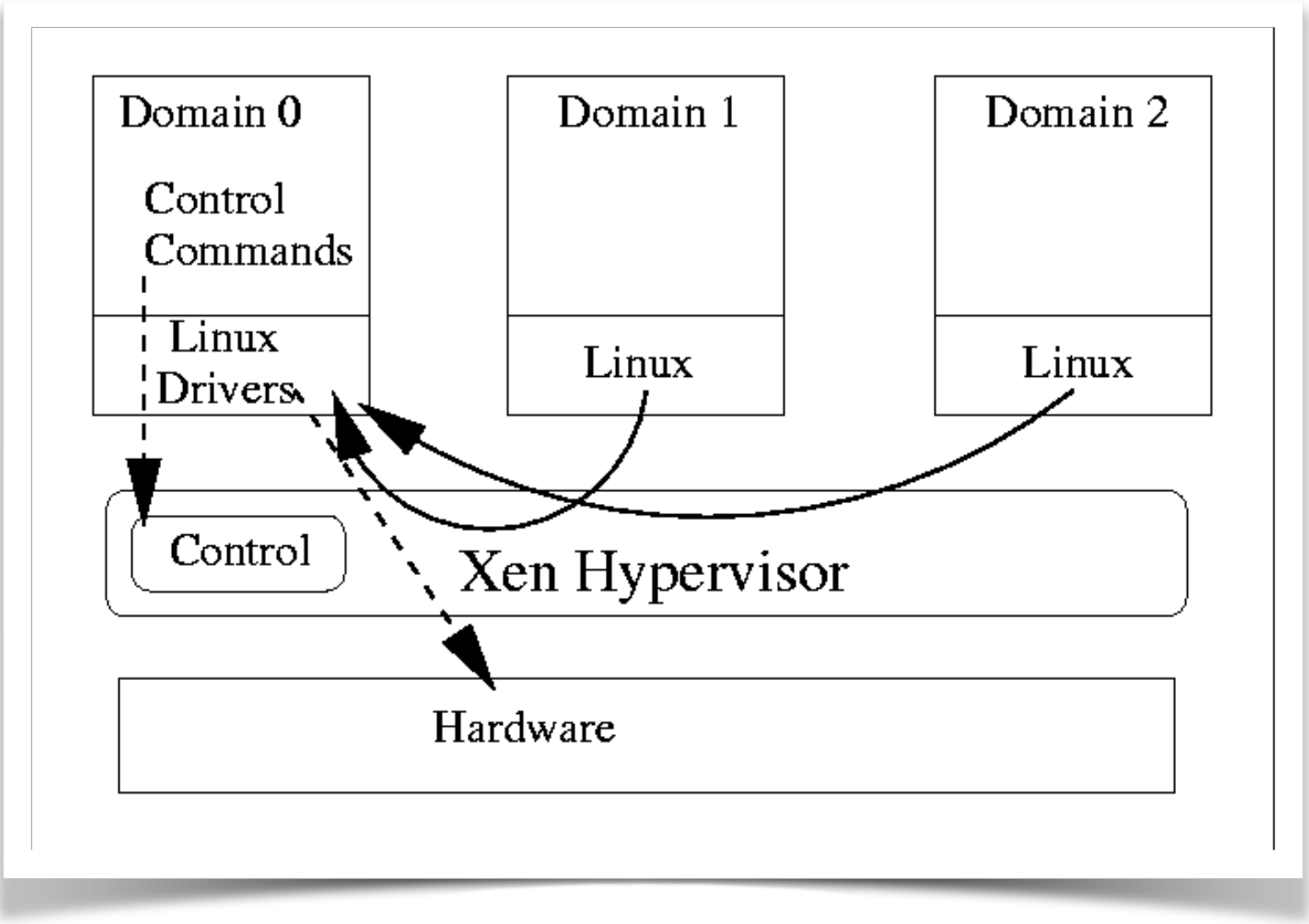
- system emulation
  - émulation d'une machine complète



QEMU

- Plusieurs modes de fonctionnement
  - kvm/xen hosting
    - émulation du hardware, mais l'exécution est sous le contrôle de KVM/Xen
- Xen
  - Hyperviseur de virtualisation

XEN


Domain 0
Control
Commands
Linux
Drivers
Domain 1
Linux
Domain 2
Linux
Control
Xen Hypervisor
Hardware

Virtualization process level

Programming language virtualization

Java, .net

- Une machine virtuelle jvm, CLI
- Interprétation d'un bytecode
- Interface native vers l'hôte
  - couplage relativement faible
- JIT ?
  - Traduction dynamique!

Question : JavaOS ? Is Java an OS ?

Virtualization process level

Application virtualization

il s'agit d'émuler l'environnement d'exécution

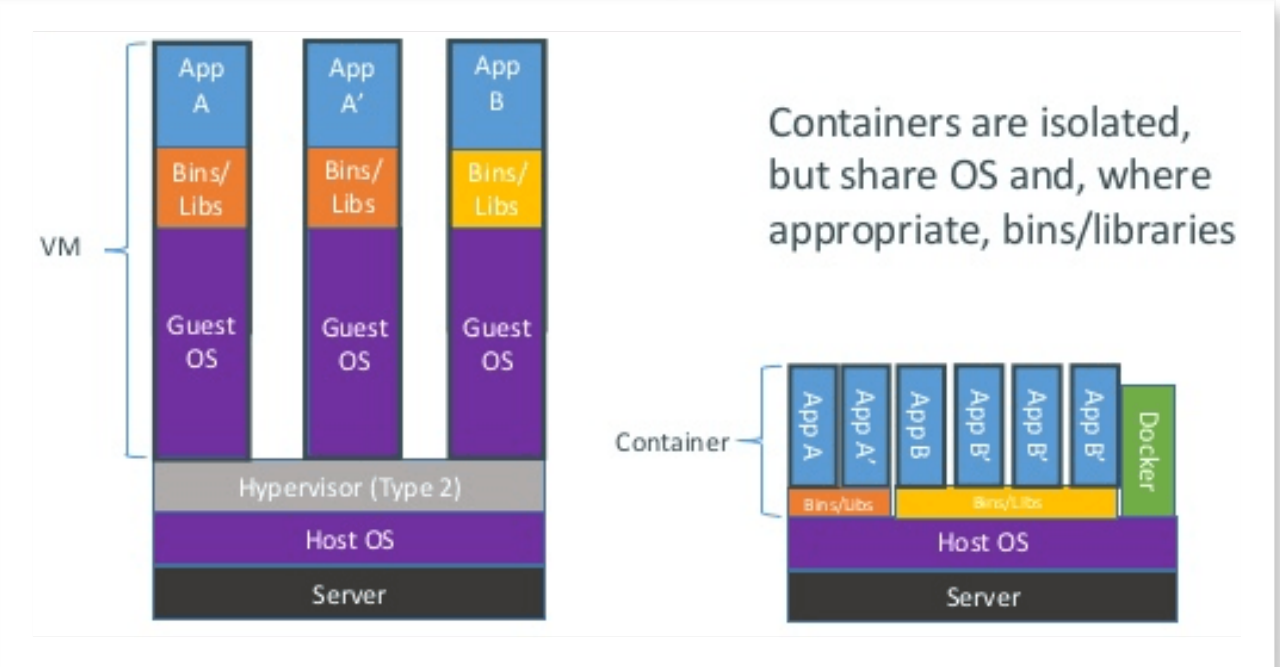
deux techniques :

interprétation : très très coûteux

traduction dynamique : cache de traduction

Containers

- Operating system level
  - Containers
    - un cas de virtualisation dans le but d'isoler et contrôler l'exécution
      - chroot
      - jails
      - cgroups
      - Docker



Containers

- chroot
  - une idée présente dès le début d'Unix!
    - confiner/limiter l'accès au système par l'accès au filesystem
      - CHange ROOT
        - n'est pas totalement sûr, certaines manipulations permettent de sortir du confinement…

Containers

- Jails
  - un chroot étendu
    - confinement strict dans une sous-partie du FS
    - un hostname+adresse IP pour l'isolation réseau
    - un exécutable à exécuter dans la prison

Containers

- Linux cgroups (Control GROUPS)
  - permet de contrôler l'accès aux ressources d'un ensemble de processus
    - isolation
    - gestion facilitée
- Linux namespaces
  - permet d'identifier des ensembles (processus, fichiers, etc)

Docker