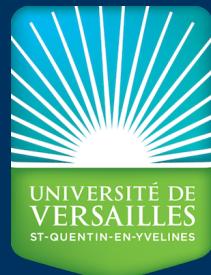


Noyau Linux

M1 - CHPS

Architecture Interne des Systèmes d'exploitations (AISE)

Jean-Baptiste Besnard
<jean-
baptiste.besnard@paratools.com>



Julien Adam
<julien.adam@paratools.com>

Organisation

- Chaque session est découpée en deux parties. Un cours théorique le matin et une mise en pratique l'après-midi (TD) portant sur les connaissances vues le matin.
- Des QCMs sur les bases **importantes** au fil des semaines et portant sur un cours précédent. Le QCM aura toujours lieu durant la matinée
- Un Projet, date de rendu au **19/12/2023 23:59**
- Un Examen final le **22/12/2023 (APM)**

1. Généralités sur les OS et Entrées-Sortie
2. Compilation, Bibliothèques et Layout Mémoire
3. Mémoire partie 2, Layout Binaire, Runtime
4. Virtualisation et Conteneurs
5. Programmation réseau et entrées/sorties avancées
6. Programmation Noyau
7. Scheduling et Temps-Réel
8. Examen Ecrit et Démos

Type d'Examen	Coefficient
QCMs	10 %
Projet	40 %
EXAMEN	50 %

Cours et Corrections



https://github.com/besnardjb/AISE_24

Programme

1. *Généralités sur les OS et Entrées-Sortie*
2. *Compilation, Bibliothèques et Layout Mémoire*
3. *Mémoire, Layout Binaire, Runtime*
4. *Virtualisation et Conteneurs*
5. *Programmation réseau et entrées/sorties avancées*
- 6. Programmation Noyau**
7. Scheduling et Temps-Réel
8. Examen Ecrit et Démos

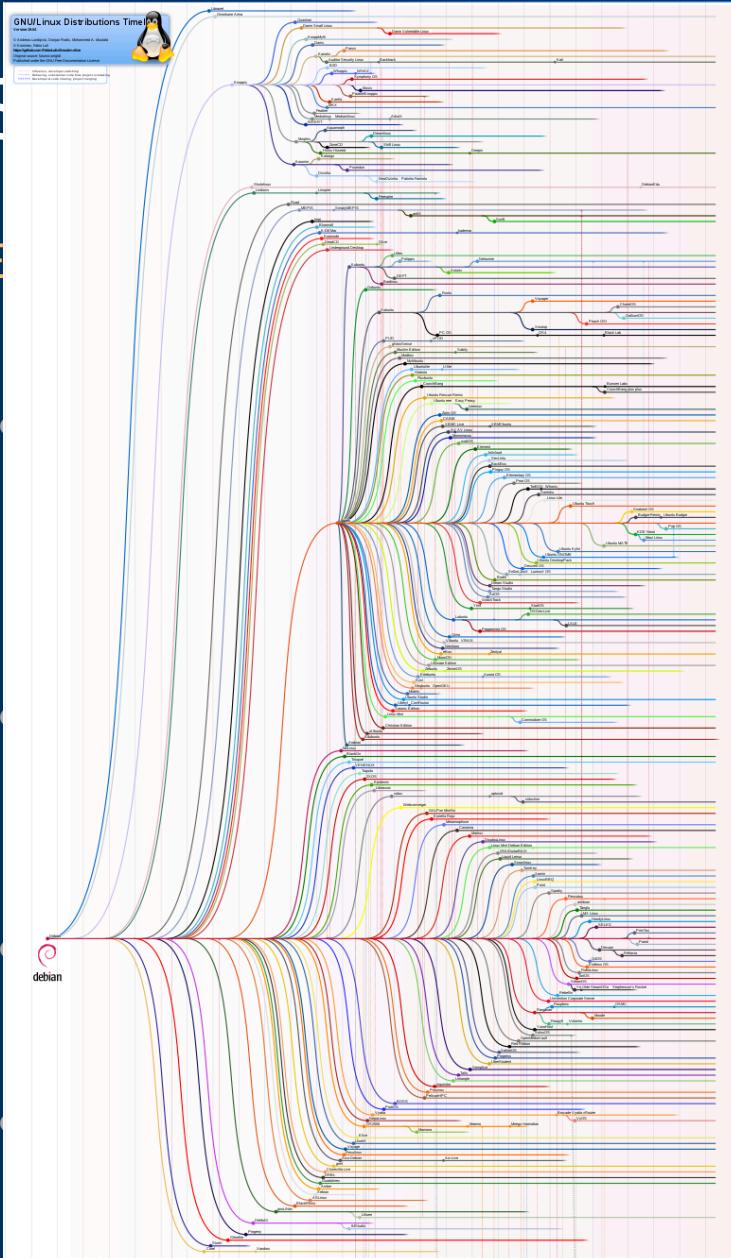
Systèmes d'exploitation (OS)

« Ensemble des programmes qui **dirige** l'utilisation des ressources d'un ordinateur »

- Multi-utilisateur
 - Conçu pour permettre à plusieurs utilisateurs d'interagir simultanément avec le système, avec lequel vient la dimension de « permissions d'accès » et de « temps partagé »
- Multitâche
 - Permet l'exécution simultanée de plusieurs programmes (en apparence), grâce à un changement de contexte très rapide
 - Préemptif vs Coopératif
- Temps-réel
 - Capacité d'un système à respecter un délai de réponse fixé
- Embarqué
 - Grande restriction de ressources (CPU, mémoire...)



« E



ation (

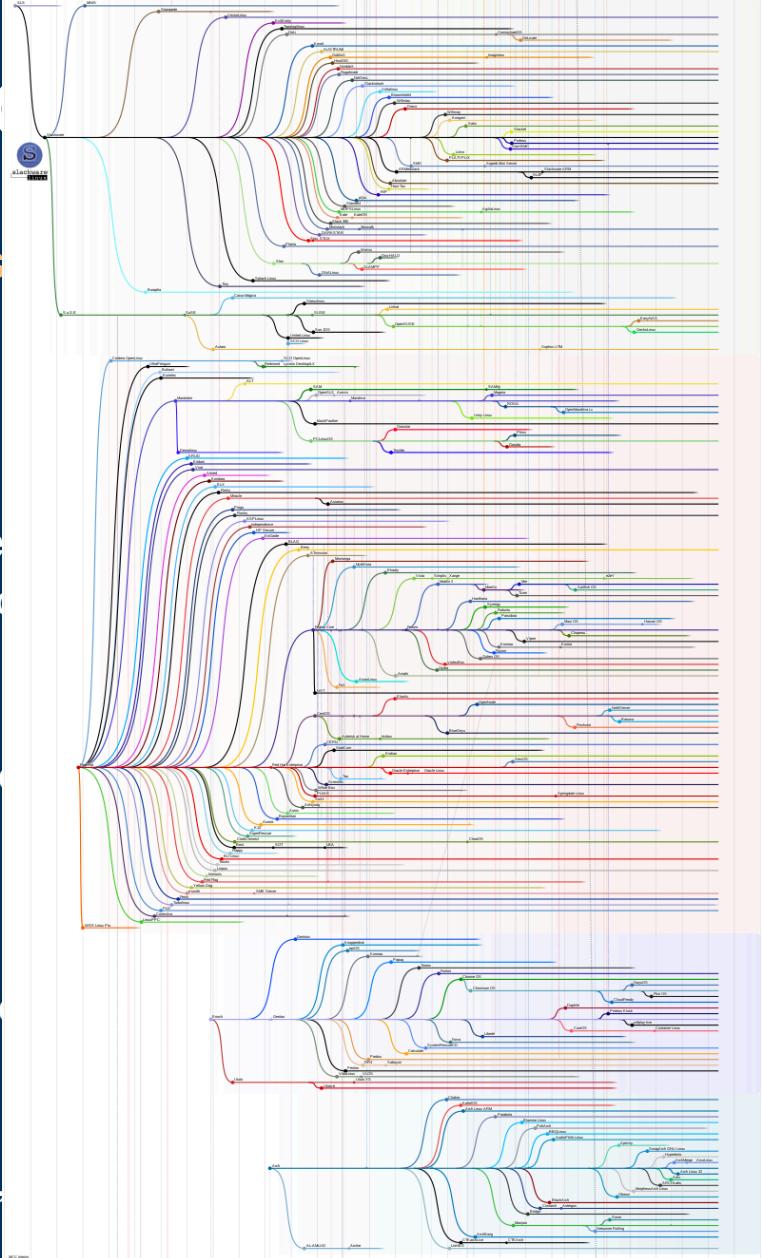
ge l'utilis

eurs utilis
c le systè
permisio
gé »

e de plusie
grâce à u
rapide

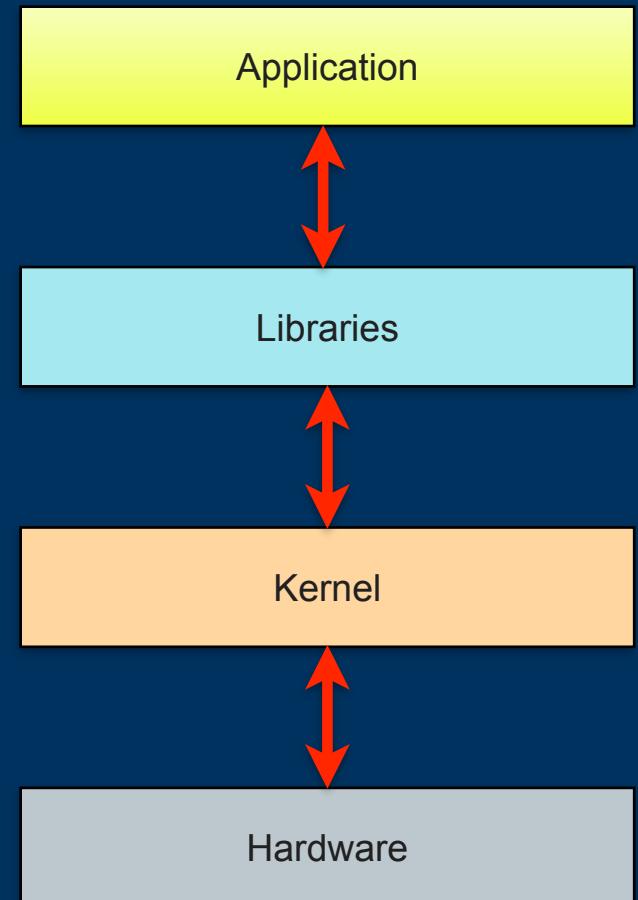
ecter un

ces (CPU,



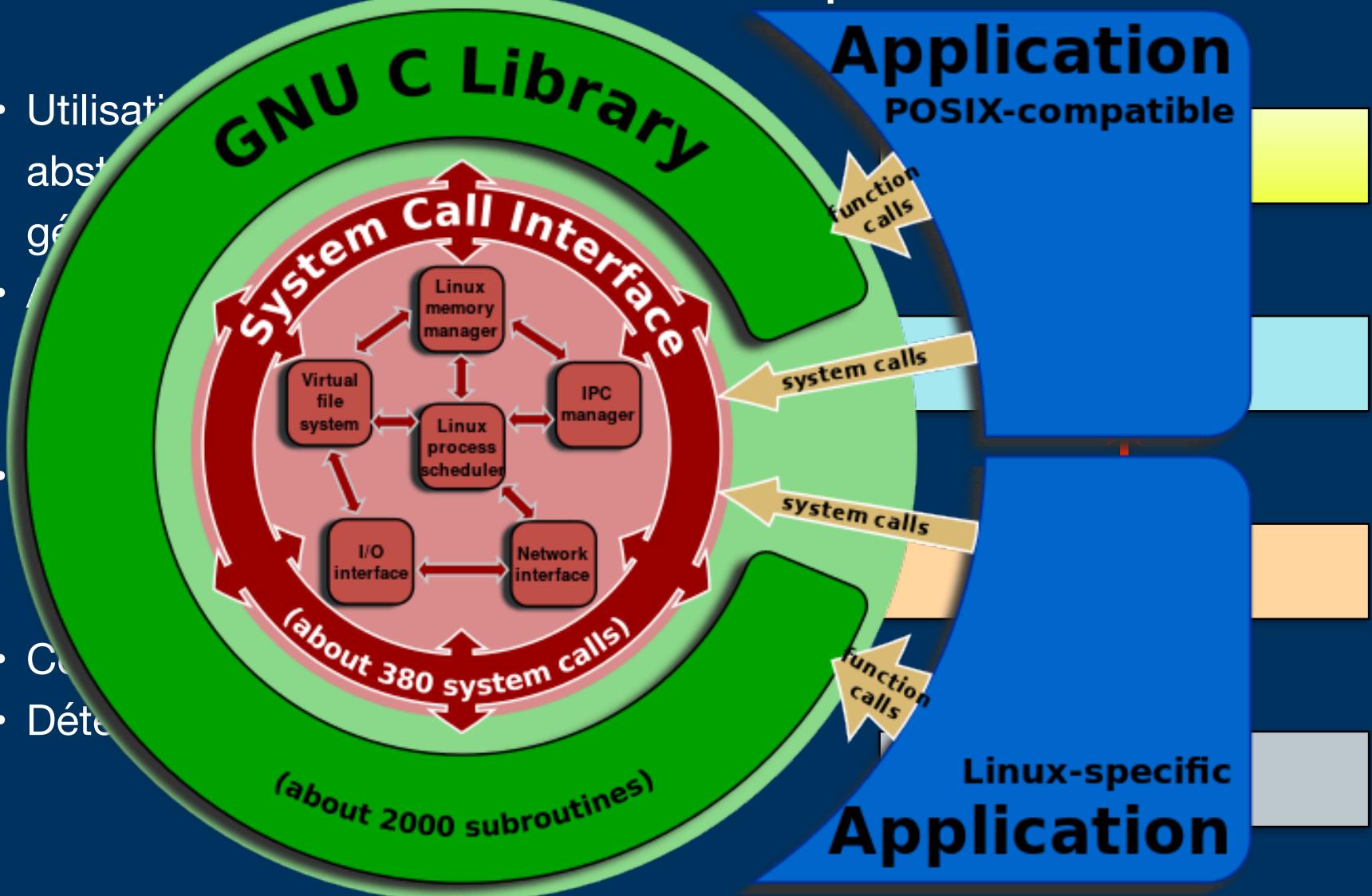
Structure d'un système d'exploitation

- Utilisation des périphériques et abstraction au travers d'interfaces génériques
- Accès aux données par isolation
 - Droits d'accès
 - Privilèges
- Protection des ressources
 - Mémoire
 - CPU
- Contrôle de maintien opérationnel
- Détection d'erreur (résilience)

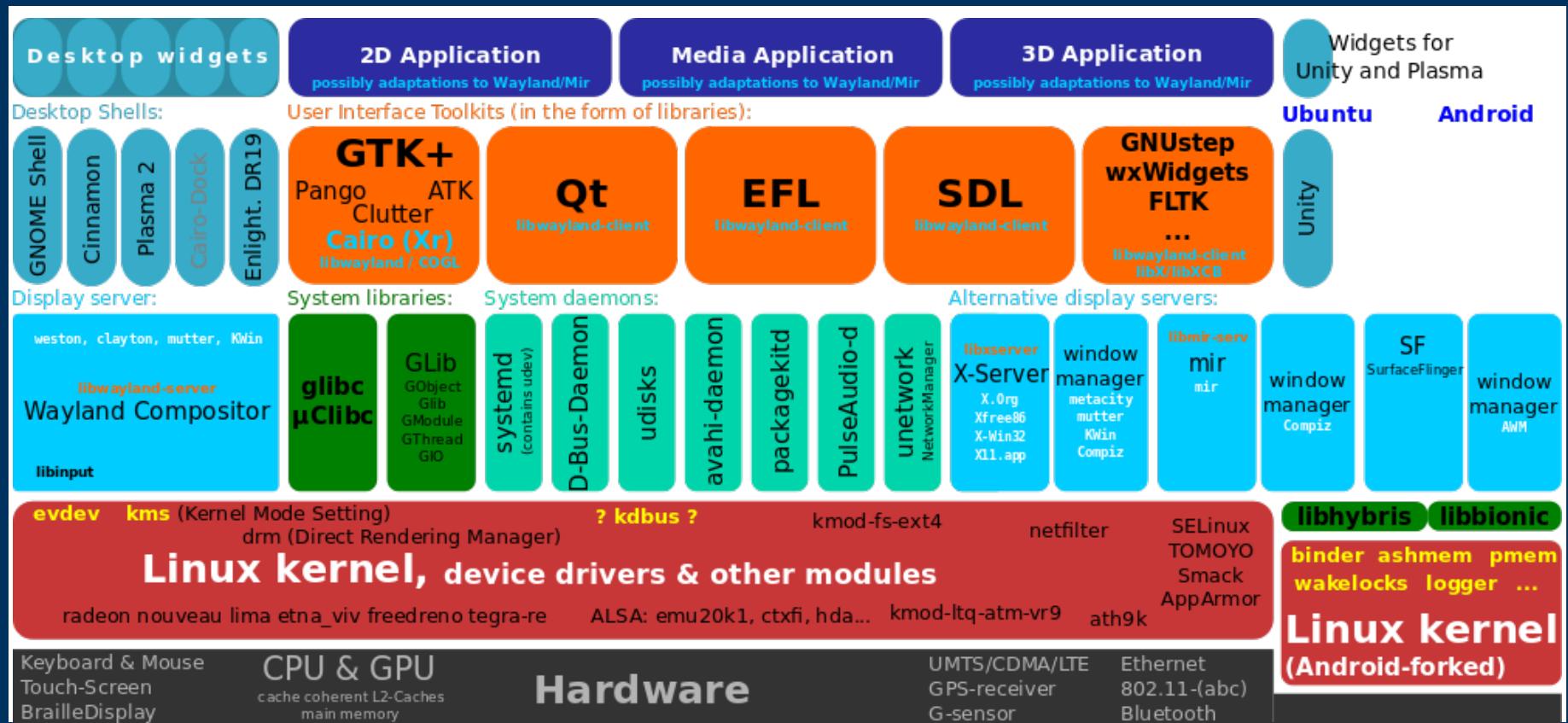


Structure d'un système d'exploitation

- Utilisation d'abstractions générales
- Fonctionnalités
- Configuration
- Détection d'erreurs



« Couches » d'un système d'exploitation sous Linux



« Sticky-bit »

- Sur Fichier
 - Fichier persistant en mémoire
 - Déprécié, voire plus implémentée
- Sur Répertoire
 - Interdit la suppression de fichiers autre que par son propriétaire
 - Utilisé pour les points de montage partagés
- Valeur: 1000 // +t
- Gestion de droits étendus : ACL & SELinux

```
Betelgeuse ~
↳ ls -l / | grep tmp
drwxrwxrwt root root 128 KB Mon Mar  7 10:08:25 2022 tmp
```

Setuid / Setgid

- Transfert de droits entre utilisateurs
- Un programme Set-UID est exécuté **au nom de** l'utilisateur du programme, même si ce n'est pas ce dernier qui l'exécute
- Tous les droits sont propagés (=déguisement)
- Utile quand aucune des autres restrictions n'est adéquate
- Ex: modifier son mot de passe personnel
 - Implique de modifier /etc/shadow, qui contient TOUS les mots de passe utilisateurs
- Cela peut-il poser un problème ?

```
↳ ls -l ./a.out
-rwsrwxr-x. 1 adamj adamj 8296 5 févr. 21:51 ./a.out
```

Et le noyau ?

- Segment privilégié permettant la communication entre logiciel et matériel
- Communication logicielle au travers de « requêtes » émises = les appels systèmes (=*syscalls*)
- Communication matérielle au travers d'un pilote de périphérique (=*driver*), spécifique à un matériel donné. Réalise la conversion s'une interface générique à une interface spécifique
- Assure consistance et cohérence dans un contexte concurrent
- Gestion du partage efficace entre les différentes instances demandeuses de temps de calcul (=ordonnancement)
- Distribution équitable de la mémoire
- **Tous les branchements d'exécution doivent être gérés !**

Depuis le tout début...

1. Le bloc d'alimentation délivre la tension
2. Amorçage de la carte-mère (+checks
 - « Anciennement » BIOS (*Basic Input/Output System*)
 - Récemment UEFI
 - Contrôle d'intégrité (« beeps »)
3. Affichage de l'initialisation

Depuis le tout début...

1. Le b
2. Amc

- <<
- R
- C

3. Affic



American
Megatrends

Version 2.17.1245. Copyright (C) 2015 American Megatrends, Inc.

BIOS Date: 07/14/2015 12:00:00 Ver: BLXSV609

Total Memory: 1048576MB

Processor Type: Intel(R) Xeon(R) CPU E7-8870 v3 @ 2.10GHz

CPU ID:306f4

CPU Cores: 18

Total CPUs: 32

Press F12 go to PXE

Press F11 go to Boot Manager(F3 on Remote Keyboard)

Press to enter setup(F4 on Remote Keyboard)

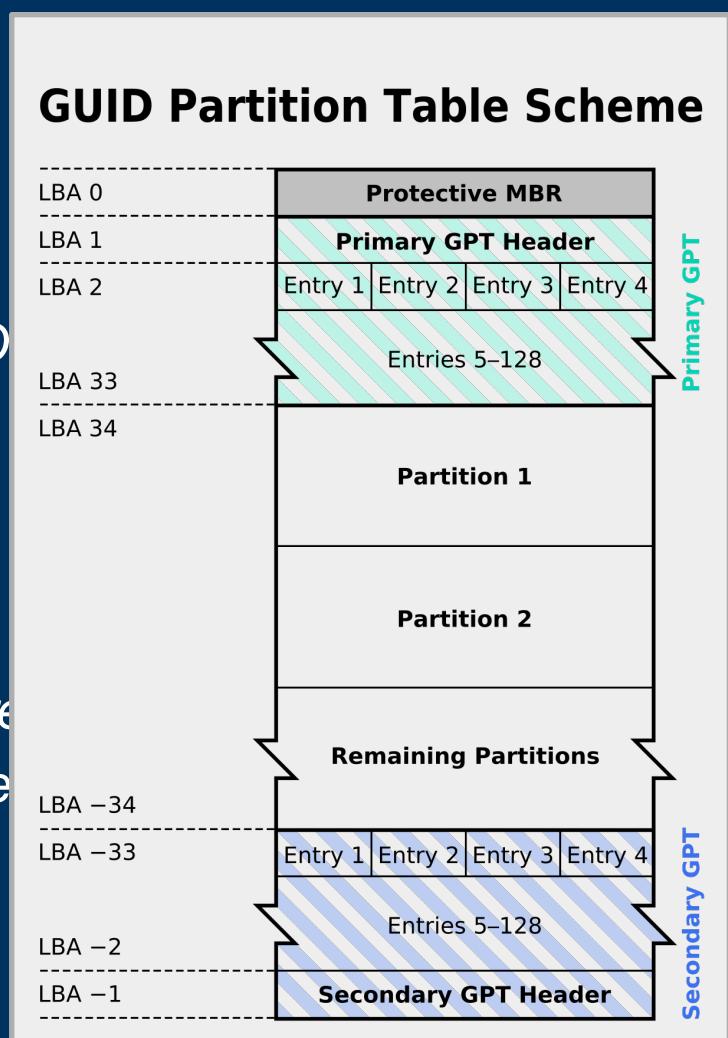
Entering Setup...

Depuis le tout début...

1. Le bloc d'alimentation délivre la tension
2. Amorçage de la carte-mère (+checks)
 - « Anciennement » BIOS (*Basic Input/Output System*)
 - Récemment UEFI
 - Contrôle d'intégrité (« beeps »)
3. Affichage de l'initialisation
4. Chargement de la « zone amorce »
 - « Anciennement » MBR (*master boot record*) »
 - Récemment GPT (*GUID partition Table*)

Depuis le tout début...

1. Le bloc d'alimentation délivre la tension
2. Amorçage de la carte-mère (+checks)
 - « Anciennement » BIOS (*Basic Input/O*)
 - Récemment UEFI
 - Contrôle d'intégrité (« beeps »)
3. Affichage de l'initialisation
4. Chargement de la « zone amorce »
 - « Anciennement » MBR (*master boot record*)
 - Récemment GPT (*GUID partition Table*)



Depuis le tout début...

1. Le bloc d'alimentation délivre la tension
2. Amorçage de la carte-mère (+checks)
 - « Anciennement » BIOS (*Basic Input/Output System*)
 - Récemment UEFI
 - Contrôle d'intégrité (« beeps »)
3. Affichage de l'initialisation
4. Chargement de la « zone amorce »
 - « Anciennement » MBR (*master boot record*) »
 - Récemment GPT (*GUID partition Table*)
5. Le micrologiciel utilise cette table pour lancer le lanceur d'amorçage
 - GRUB2, PXELinux, NTLDLR, winload.exe...

Depuis le tout début...

1. Le bloc d'alimentation délivre la tension

2. Amorçage de la

- « Anciennement
- Récemment
- Contrôle d'initialisation

3. Affichage de l'interface

4. Chargement de

- « Anciennement
- Récemment

5. Le micrologiciel

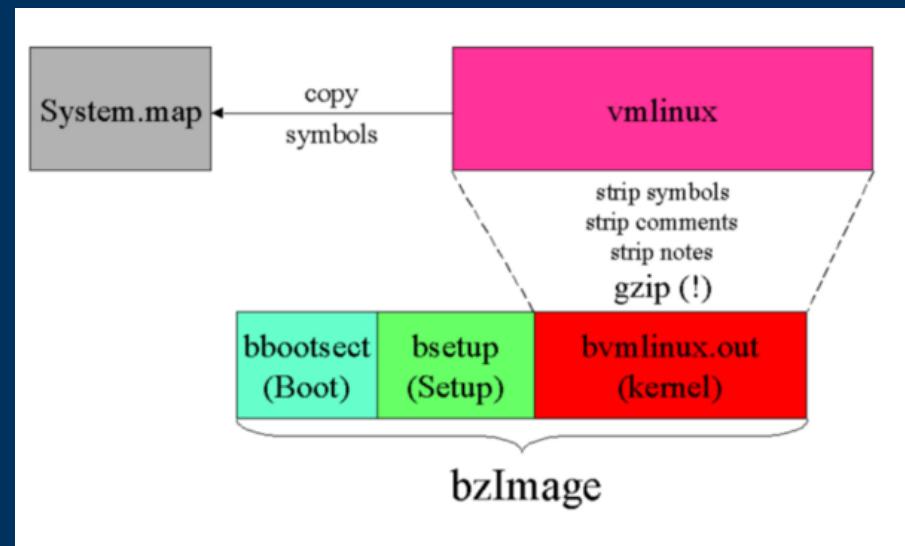
- GRUB2, PXE

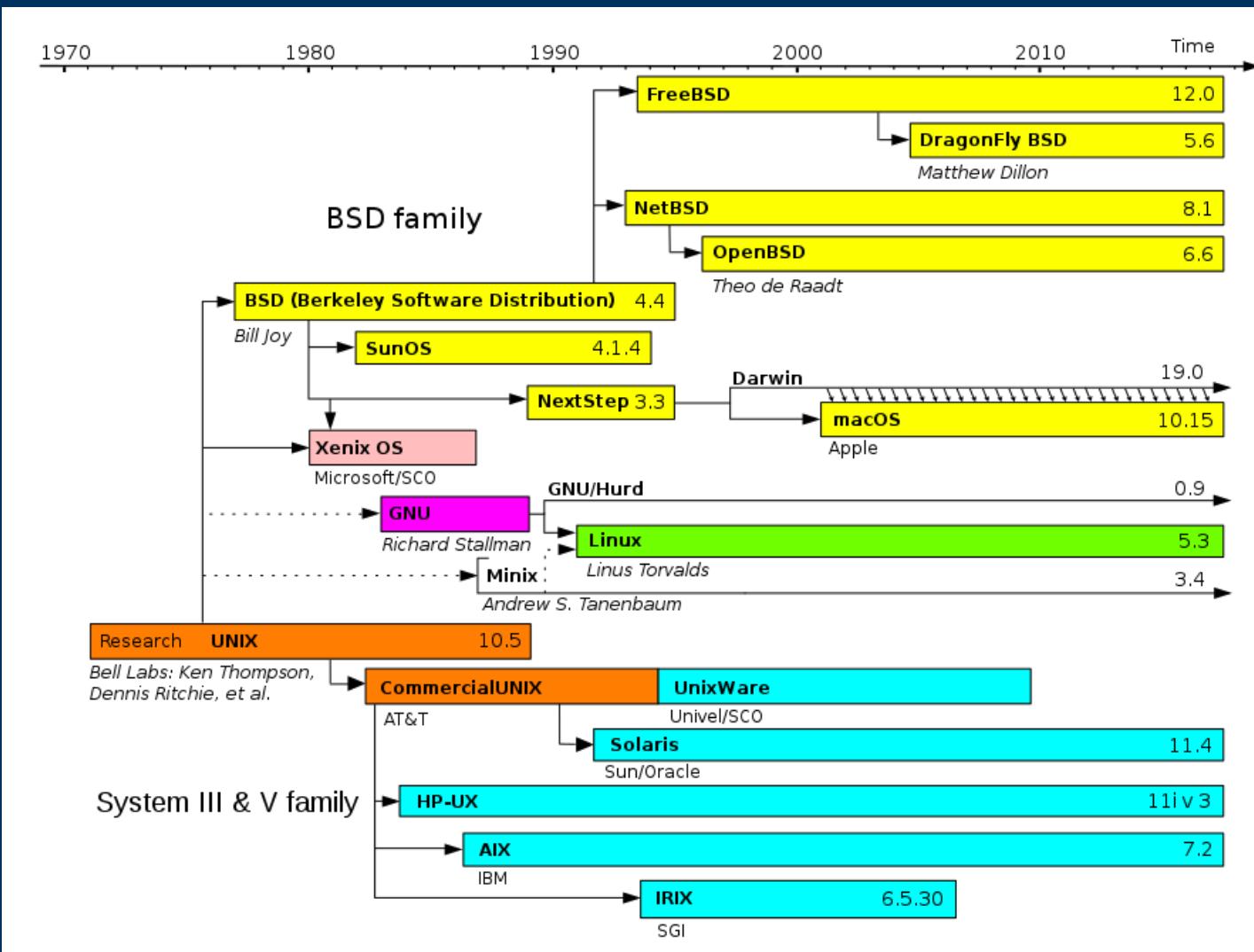


Use the + and - keys to select which entry is highlighted.
Press enter to boot the selected OS, 'e' to edit the commands
before booting or 'c' for a command-line.

Chargement du noyau

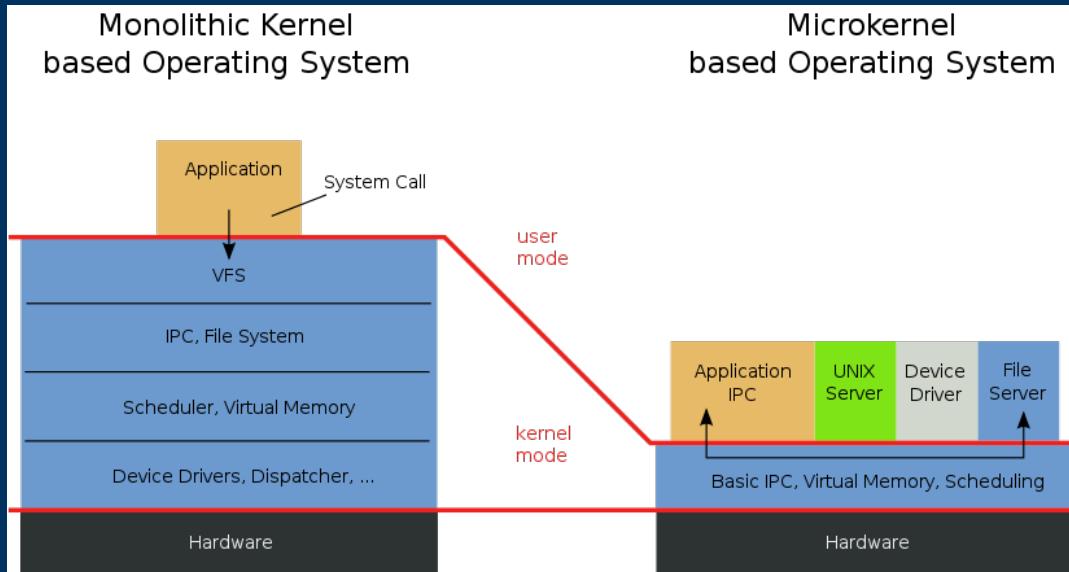
1. Le noyau est souvent stocké compressé. Il est généralement « rangé » dans /boot (c'est là que le chargeur d'amorçage cherche par défaut)
2. Décompression de vmlinu-\$ (version) avec zlib
3. Mise en place:
 1. Filesystem en RAM (initramfs)
 2. Initialisation mémoire (pagination)
 3. Initialisation du hardware
4. `start_kernel()`





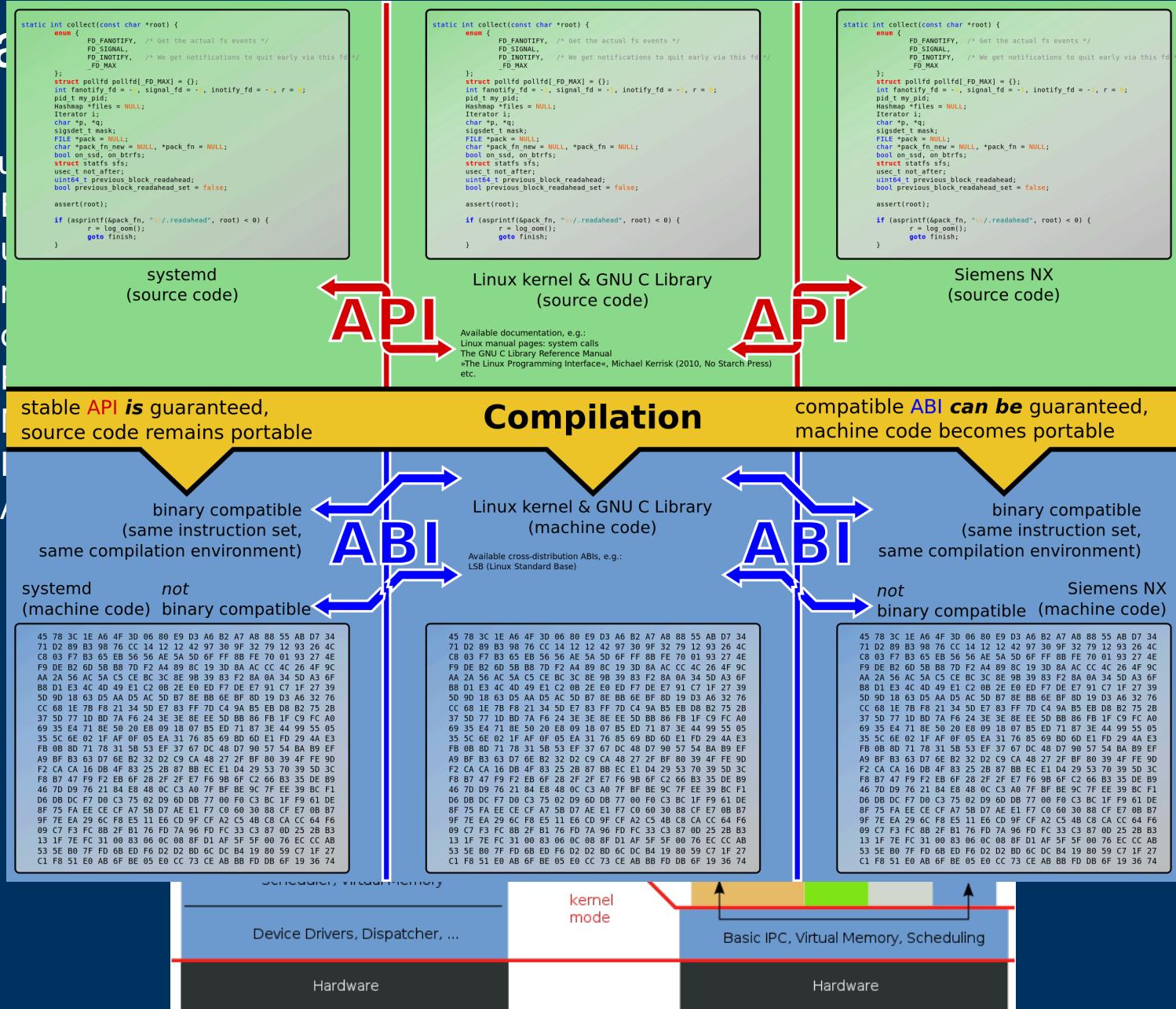
Noyaux monolithiques vs micro-noyaux

- Noyau monolithique:
 - Ensemble du code regroupé dans un seul bloc, évoluant en espace noyau et faisant office d'interface complète.
 - Faible latence
 - Modularité maintenant éprouvée
 - Intolérant aux bugs !
 - ABI moins portable
- Micro-noyau:
 - Un maximum de services du noyau sont déportés dans l'espace utilisateur
 - Le cœur du noyau est restreint au minimum
 - Tolérant aux pannes/bugs
 - Peut avoir une plus grande latence
 - Grande flexibilité / portabilité

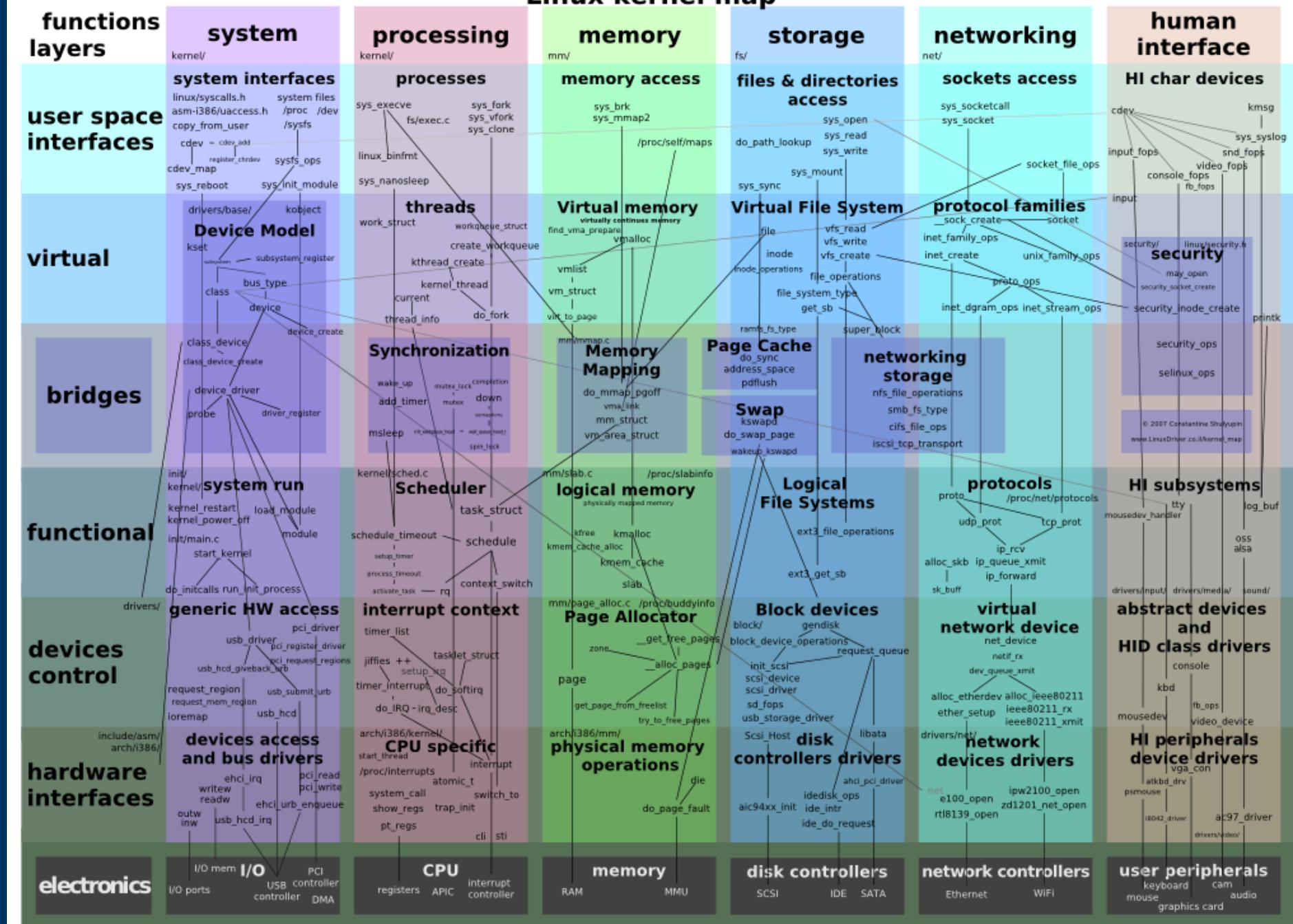


Noya

- Noya

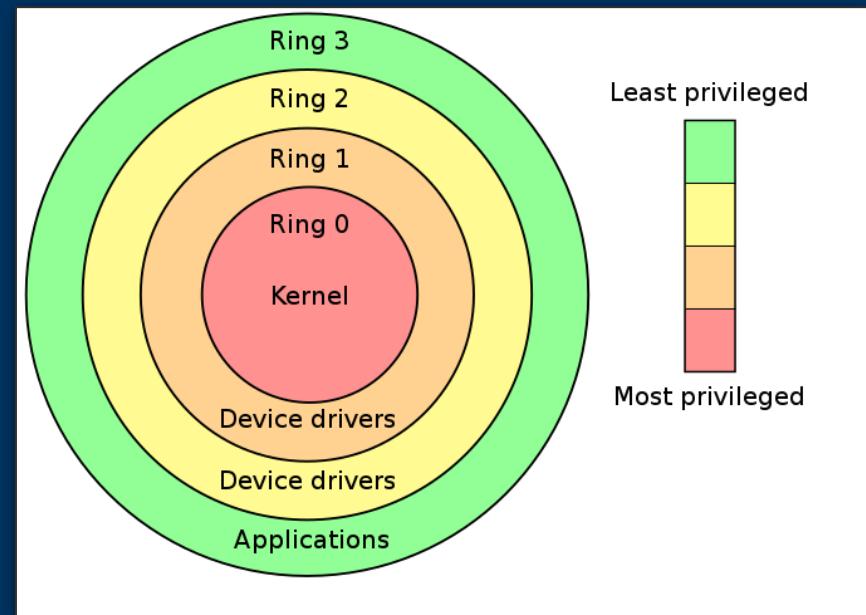


Linux kernel map



Noyaux et anneaux de protection

- Autre nom: *Hierarchical Protection domains*
- Fournit plusieurs niveaux d'accès aux ressources, de manière hiérarchique (l'accès à un niveau privilégié permet aussi l'accès à tous les niveaux moins privilégiés)
- X86 vient avec 4 niveaux, ring0 est dédié au noyau, et ring3 à l'espace utilisateur
- User mode
- Supervisor mode
 - Mode d'exécution privilégiée
 - **Rien à voir avec root...**
- « Hypervisor » mode: les machines virtuelles d'aujourd'hui encapsulent généralement l'OS invité dans son propre set, créant plus profondément ses propres niveau



Noyaux et capacités

- Autre système de sécurité mis en place par la majorité des noyaux
- Permet d'éviter un « tout-ourien » pour les processus non-privilégiés, à grain plus fin que les anneaux de protection
- *Ex: Comment avoir la possibilité de tracer un programme sans être root/dans le noyau ?*
- Les accès sont rangés par groupes de privilèges
- Un programme peut recevoir l'autorisation d'utiliser les permissions d'un groupe => **une capacité**

```
↳ getcap `which ping`  
/usr/bin/ping = cap_net_admin,cap_net_raw+p
```

```
cap_audit_control  
cap_audit_read  
cap_audit_write  
cap_block_suspend  
cap_chown  
cap_dac_override  
cap_dac_read_search  
cap_fowner  
cap_fsetid  
cap_ipc_lock  
cap_ipc_owner  
cap_kill  
cap_lease  
cap_linux_immutable  
cap_mac_admin  
cap_mac_override  
cap_mknod  
cap_net_admin  
cap_net_bind_service  
cap_net_broadcast  
cap_net_raw  
cap_setfcap  
cap_setgid  
cap_setpcap  
cap_setuid  
cap_sys_admin  
cap_sys_boot  
cap_sys_chroot  
cap_syslog  
cap_sys_module  
cap_sys_nice  
cap_sys_pacct  
cap_sys_ptrace  
cap_sys_rawio  
cap_sys_resource  
cap_sys_time  
cap_sys_tty_config  
cap_wake_alarm
```

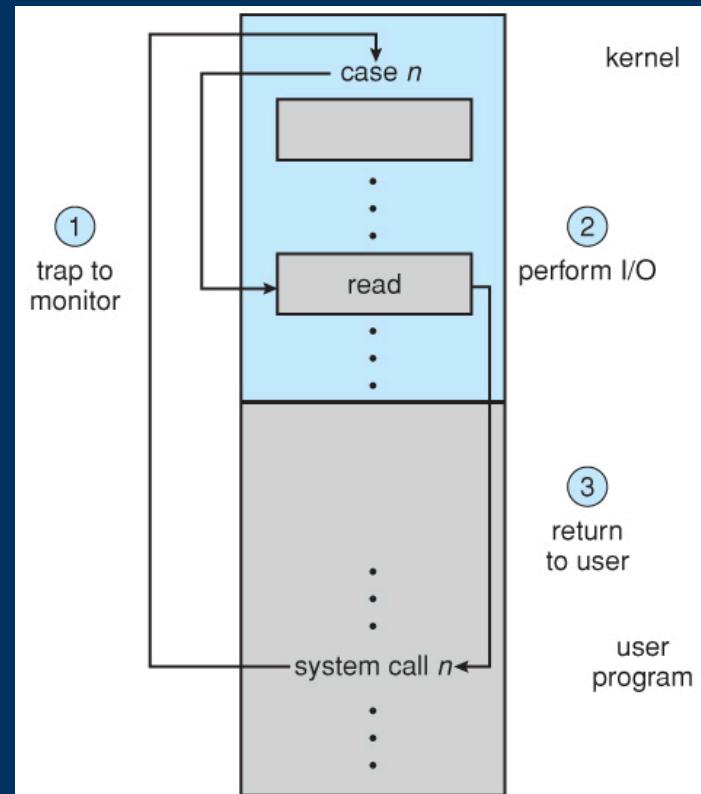
Noyaux et capacités

- En ligne de commande
 - Lister les capacités: getcap <fichier>
 - Changer les capacités: setcap <capacité> <fichier>
- Une capacité est composé de deux termes:
 - Un nom (« cap_net_raw », « cap_sys_module »...)
 - Un modificateur (=, +, -)
 - Une permission
 - p: peut être utilisé (=permitted)
 - e: effectivement utilisé (=effective)
 - i: Héritée lors d'un appel à fork() / exec() (=inherited)

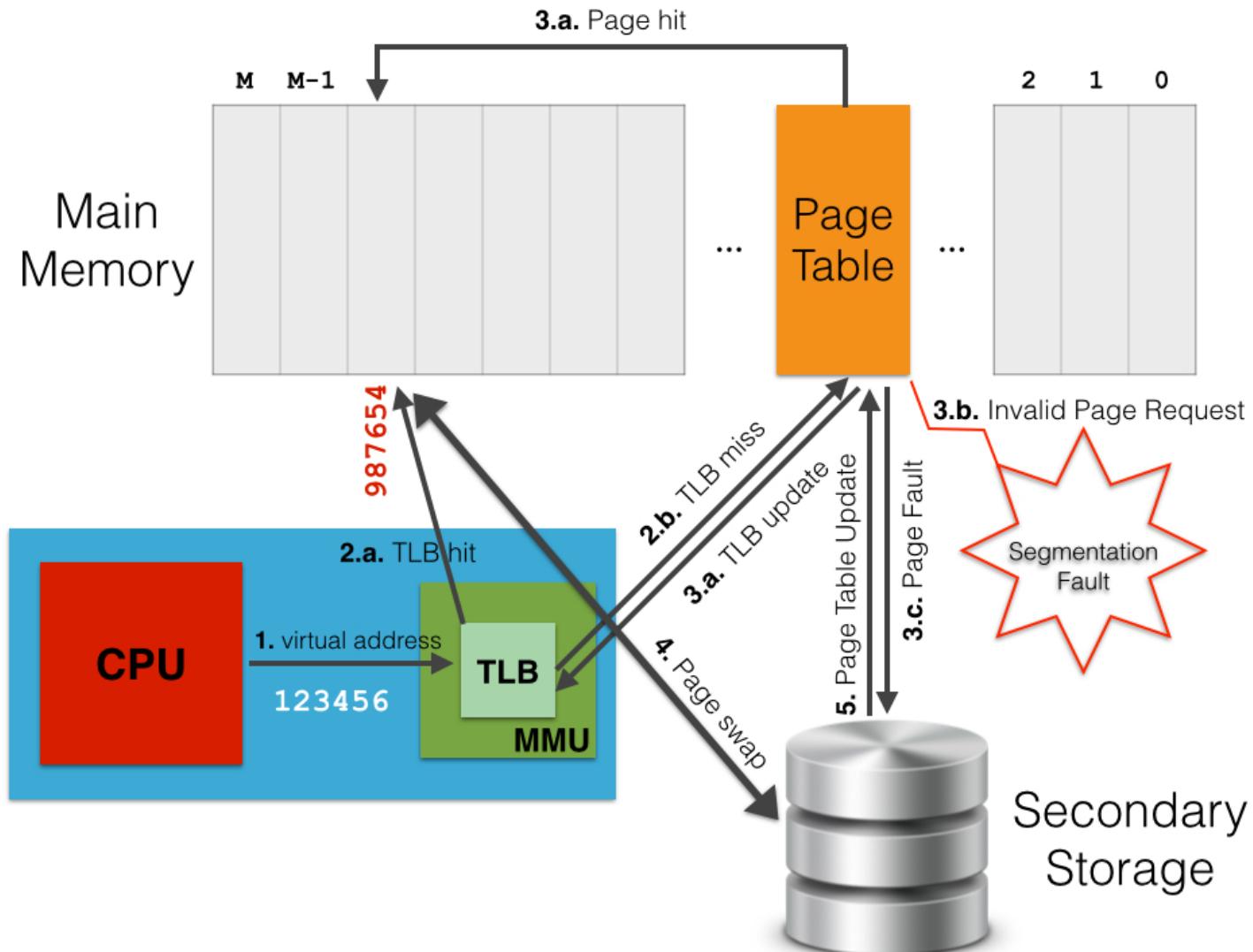
```
(root) → linux-5.5.7 setcap cap_net_raw+ep /bin/ping && ping google.com
PING google.com (216.58.213.142) 56(84) bytes of data.
64 bytes from par21s03-in-f14.1e100.net (216.58.213.142): icmp_seq=1 ttl=50 time=3.75 ms
64 bytes from par21s03-in-f14.1e100.net (216.58.213.142): icmp_seq=2 ttl=50 time=2.64 ms
64 bytes from par21s03-in-f14.1e100.net (216.58.213.142): icmp_seq=3 ttl=50 time=2.70 ms
^C
--- google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 2.649/3.034/3.753/0.510 ms
```

Composants noyau

- System Call Interface (SCI)
 - Couche d'abstraction espaces utilisateur/noyau permettant aux processus réguliers d'interagir avec le noyau et ses ressources privilégiées
- Process Management (PM)
 - Crée / supprime les processus
 - Support de la communication inter-processus
 - Gestion du PCB dans l'espace noyau de l'espace d'adressage
 - Ordonnancement sur CPU

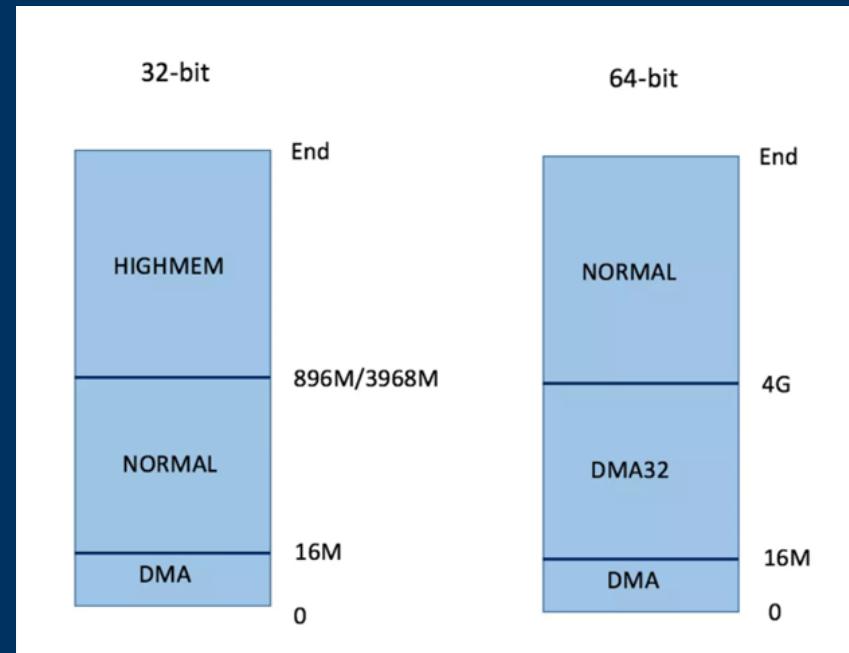


Noy



Noyau & Mémoire

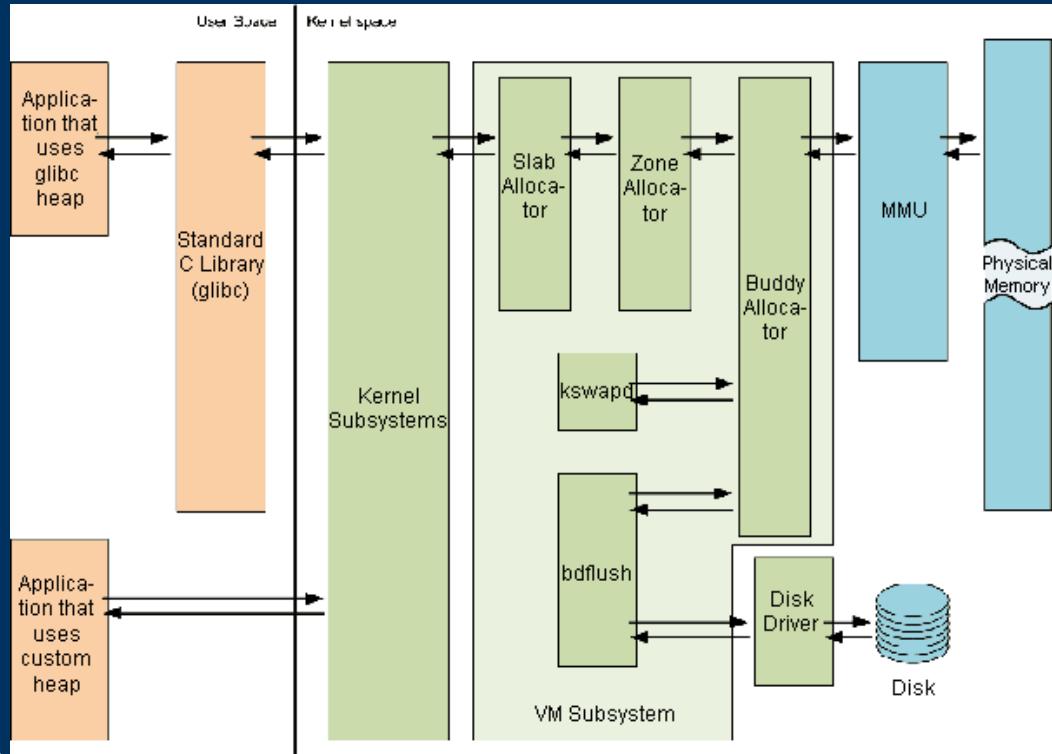
- Contrairement à la mémoire virtuelle, le noyau ne considère pas la mémoire physique comme linéaire. Découpage:
 - Noeuds
 - Zones:
 - DMA[32]: Mémoire dédiée pour accès direct (aujourd'hui à disposition des Devices)
 - HIGHMEM: Mémoire non persistante dans l'adressage du noyau (*obsolète*)
 - NORMAL: mémoire adressable



```
→ ~ cat /proc/buddyinfo
Node 0, zone      DMA      0      0      0      0      0      0      0      0      1      1      3
Node 0, zone    DMA32      4      2      1      1      4      4      2      3      4      3      864
Node 0, zone   Normal  970    412    252   186   112    41     15     14      3     19    594
```

Noyau & Mémoire

- Petites allocations:
Kmalloc/kfree (< 1 page)
- Grosses allocations:
vmalloc/vfree
- Allocation de pages:
alloc_pages/free_pages
- 3 allocateurs : SLAB, SLOB,
SLUB
- Slab allocator : méthode de
cache pour l'allocation de
petites structures
- Buddy allocator: rapide &
efficace mais beaucoup de
fragmentation



`GFP_USER` - Allocate memory on behalf of user. May sleep.

`GFP_KERNEL` - Allocate normal kernel ram. May sleep.

`GFP_ATOMIC` - Allocation will not sleep. May use emergency pools.

`GFP_HIGHUSER` - Allocate pages from high memory.

`GFP_NOIO` - Do not do any I/O at all while trying to get memory.

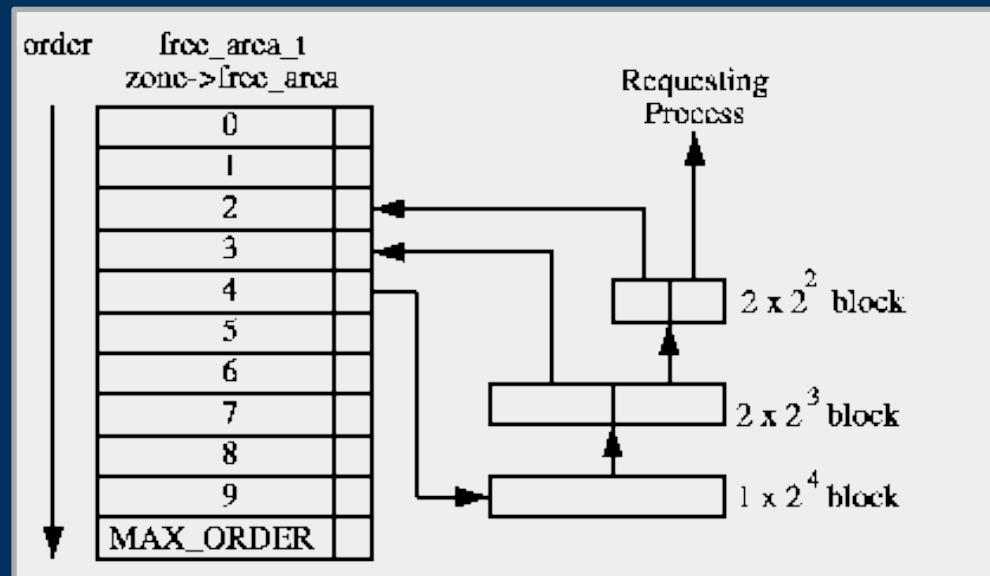
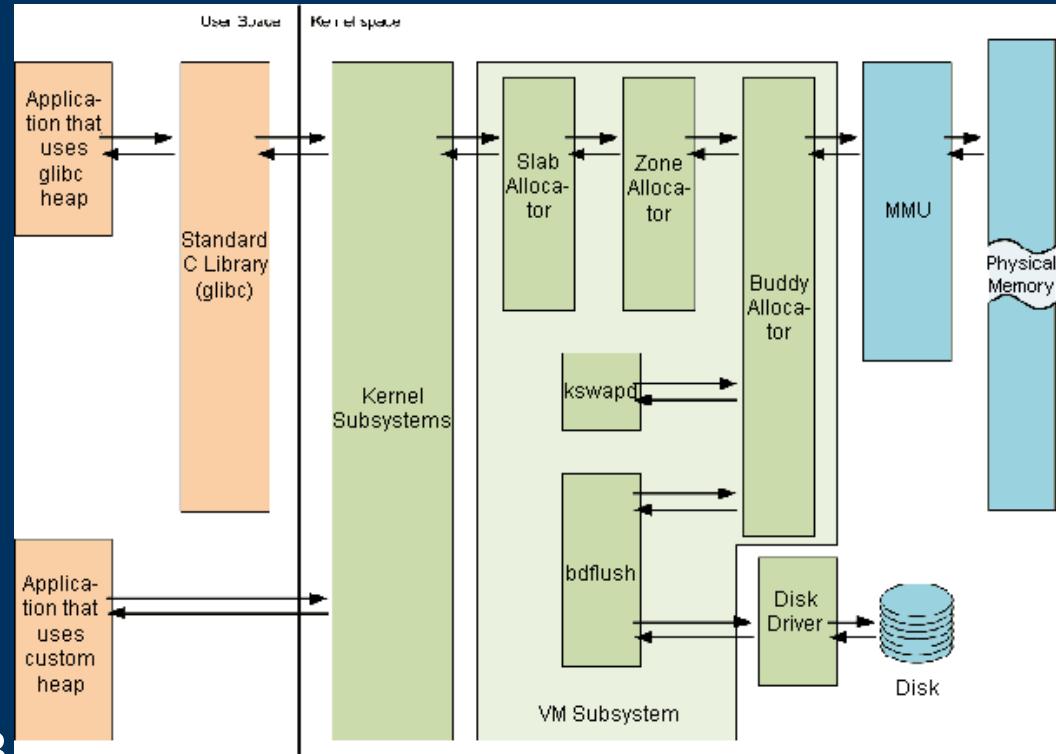
`GFP_NOFS` - Do not make any fs calls while trying to get memory.

`GFP_NOWAIT` - Allocation will not sleep.

`GFP_THISNODE` - Allocate node-local memory only.

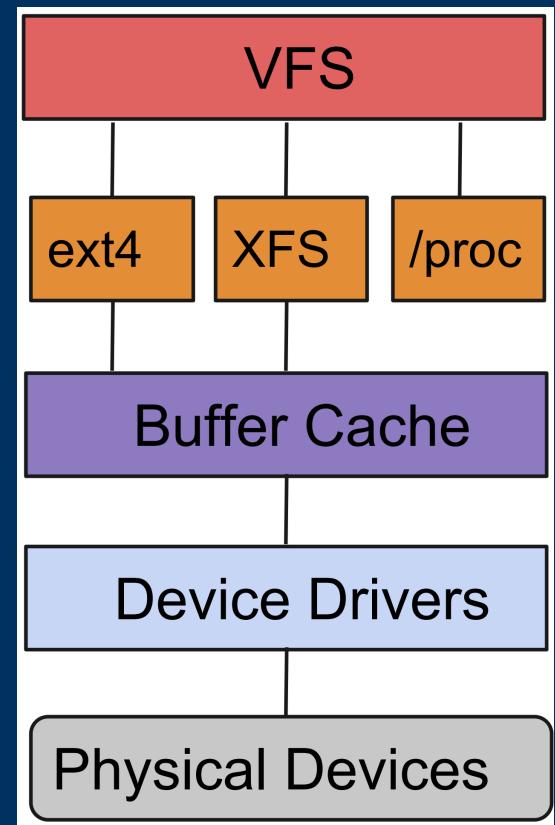
Noyau & Mémoire

- Petites allocations:
Kmalloc/kfree (< 1 page)
- Grosses allocations:
vmalloc/vfree
- Allocation de pages:
alloc_pages/free_pages
- 3 allocateurs : SLAB, SLOB, SLUB
- Slab allocator : méthode de cache pour l'allocation de petites structures
- Buddy allocator: rapide & efficace mais beaucoup de fragmentation



Composants noyau

- Memory Management (MM)
 - Allocation, gestion mémoire (virtuelle <-> physique)
 - Support du swap
 - Table des pages
 - TLB
- Network
 - Protocoles réseau
 - Couche d'abstraction entre applications et interfaces réseau.
- Device Drivers (DD)
 - Gestion des périphériques I/O



Device drivers

- Un « device driver » agit comme un opérateur permettant le contrôle d'un type donné de périphériques attachés à la machine
 - Ex: un disque, une clé USB, un lecteur CD, Terminal
- La liste des devices est disponible dans **/dev**
- Numérotation des devices
 - Majeur : valeur fixée par le noyau pour tous les devices utilisant le même driver
 - Mineur: Valeur arbitraire gérée par le driver pour différencier les différents devices sous sa responsabilité
- Création via « mknod » : `mknod /dev/mydev c 42 1`

```
Betelgeuse ~
└─ MAJOR=42; cat /usr/src/linux-headers-4.9.0-12-common/include/uapi/linux/major.h | grep "$MAJOR"
Betelgeuse ~
└─▶
```

« Device Drivers » : types

- « **Chardev** » : le flux est transmis, un octet à la fois, se comportant comme des pipes ou des ports séries. C'est le type d'interface convenant pour une communication série (une information à la fois)
 - Entrée clavier
 - Carte son
 - Carte vidéo
- « **blockdev** »: données cachées par bloc, permettant aux opérations une manipulation en groupe
 - Point de montage disque
 - Flux réseau

```
↳ ls -l /dev/ | egrep "^\b"  
brw-rw---- 1 root disk    8,   0 févr. 19 08:38 sda  
brw-rw---- 1 root disk    8,   1 févr. 19 08:38 sda1  
brw-rw---- 1 root disk    8,   2 févr. 19 08:38 sda2  
brw-rw---- 1 root disk    8,   3 févr. 19 08:38 sda3  
brw-rw---- 1 root disk    8,   4 févr. 19 08:38 sda4  
brw-rw---- 1 root disk    8,  16 févr. 19 08:38 sdb  
brw-rw---- 1 root disk    8,  17 févr. 19 08:38 sdb1  
brw-rw---- 1 root disk    8,  32 févr. 19 08:38 sdc  
brw-rw---- 1 root disk    8,  48 févr. 19 08:38 sdd  
brw-rw---- 1 root disk    8,  64 févr. 19 08:38 sde  
brw-rw---- 1 root disk    8,  80 févr. 19 08:38 sdf  
brw-rw----+ 1 root cdrom  11,   0 févr. 19 08:38 sr0
```

```
↳ ls -l /dev/ | egrep "^\c.*tty.*"  
crw-rw-rw- 1 root tty      5,   2 mars  9 17:07 ptmx  
crw-rw-rw- 1 root tty      5,   0 mars  9 16:24 tty  
crw--w---- 1 root tty     4,   0 févr. 19 08:38 tty0  
crw--w---- 1 root tty     4,   1 mars  5 11:04 tty1  
crw--w---- 1 root tty     4,  10 févr. 19 08:38 tty10  
crw--w---- 1 root tty     4,  11 févr. 19 08:38 tty11  
crw--w---- 1 root tty     4,  12 févr. 19 08:38 tty12  
crw--w---- 1 root tty     4,  13 févr. 19 08:38 tty13  
crw--w---- 1 root tty     4,  14 févr. 19 08:38 tty14  
crw--w---- 1 root tty     4,  15 févr. 19 08:38 tty15  
crw--w---- 1 root tty     4,  16 févr. 19 08:38 tty16  
crw--w---- 1 root tty     4,  17 févr. 19 08:38 tty17  
crw--w---- 1 root tty     4,  18 févr. 19 08:38 tty18  
crw--w---- 1 root tty     4,  19 févr. 19 08:38 tty19  
crw--w---- 1 root tty     4,   2 févr. 19 08:38 tty2  
crw--w---- 1 root tty     4,  20 févr. 19 08:38 tty20
```

Ordonnancement de processus

- Principe de préemption:
 - Méthode permettant de suspendre l'exécution d'un programme afin d'invoquer un ordonnanceur de tâche pour déterminer qui doit s'exécuter ensuite
 - S'oppose au principe coopératif, où le système « attend » gentiment que le processus veuille bien passer la main
- Est réalisé au moyen d'interruption, **à tout moment** (kernel ou userspace)
- Cette approche permet une distribution plus juste des ressources en fonction d'un ensemble de critères
 - Besoin de réactivité immédiate
 - Attente d'I/O (=I/O bound)
 - Besoin en calcul (=CPU bound)
 - Temps passé sur CPU vs temps passé à attendre le CPU
- La majorité des systèmes industriels reposent aujourd'hui sur un noyau préemptif.

Ordonnancement de processus

- Pour s'adapter au mieux à l'évolution de la charge, la politique d'ordonnancement (scheduling) doit être dynamique
- Le temps d'usage d'un CPU est divisé en *slices*
- L'idée est d'avoir une répartition **juste** des slices sur les processus exécutables
- Sous Linux, avec un noyau préemptif:
 - Une priorité « statique » est attribué à chaque processus lorsqu'il démarre en fonction d'un ensemble de critères (utilisateur, programme)
 - Une priorité « dynamique », qui change au cours de la vie du processus est appliquée périodiquement par le scheduler en fonction de son activité (selon l'ordonnancement choisi)

Ordonnancement de processus

man shed

- **Politiques traditionnelles:**

- **SCHED_NORMAL | SCHED_OTHER:** Politique par défaut, « Completely Fair Scheduler », chaque tache dispose d'une quantité égale de temps CPU.
 - Famine mis en défaut par un incrément à chaque fois qu'un processus est prêt mais non exécuté
- **SCHED_IDLE** (2.6): Jobs à basse priorité, lorsque le CPU « a le temps » (nice() inefficace)
- **SCHED_BATCH** (2.6): Similaire à SCHED_OTHER, mais considère les threads CPU-bound (non-interactifs) et réalise les ajustements nécessaires pour ne pas trop les pénaliser
- Attribut modifiable via:
 - `sched_setaffinity()`
 - `nice()`

chrt -p \$PID

- **Politiques temps-réel:**

- **SCHED_FIFO**: premier arrivé, premier servi, sans prendre en compte une priorité dynamique.
- **SCHED_RR**: chaque processus ne peut garder l'accès au CPU que pour N quantum, après quoi il est remis en fin de file (à priorité constante)
- **SCHED_DEADLINE** (3.14): Le processus qui doit compléter avant la prochaine période passe en premier

Compiler le noyau Linux

- Un seul site web: <https://kernel.org>
- Récupération de l'archive **et de sa signature.**
- Vérification de l'archive pour corruption !
 - gpg2 --keyserver https://pgp.mit.edu/ --search-keys gregkh@kernel.org
 - gpg2 --verify <linux-archive>.tar.sig
- Décompression de l'archive
 - tar xf <linux-archive>.tar

```
↳ tree -d -L 1
.
├── arch
├── block
├── certs
├── crypto
├── Documentation
├── drivers
├── fs
├── include
├── init
├── ipc
├── kernel
├── lib
└── LICENSES
  └── mm
  └── net
  └── samples
  └── scripts
  └── security
  └── sound
  └── tools
  └── usr
  └── virt
22 directories
```

22 directories

```
gpg: assuming signed data in 'linux-5.5.7.tar'
gpg: Signature made ven. 28 févr. 2020 17:24:41 CET
gpg:           using RSA key 647F28654894E3BD457199BE38DBBDC86092693E
gpg: Good signature from "Greg Kroah-Hartman <gregkh@linuxfoundation.org>" [unknown]
gpg:           aka "Greg Kroah-Hartman <gregkh@kernel.org>" [unknown]
gpg:           aka "Greg Kroah-Hartman (Linux kernel stable release signing key)
<greg@kroah.com>" [unknown]
```

Compiler le noyau Linux

- Un seul fichier de configuration: **.config**
- Généré par « make <...>config » (ex: make menuconfig)
- Possibilité de copier celui du noyau courant: **/boot/config-***
- Compilation & installation du noyau
 - `make -j<X>`
 - `make modules_install`
 - `make install`
- Durée dépendante de votre configuration et de vos ressources
 - De quelques minutes à plusieurs heures
- C'est tout !

Importing 9349506041576182222

```
disk vdev '/dev/gptid/6ca110ab-3952-11e9-8cc1-0024ecf12fe7': best uberblock found for spa $import.
disk vdev '/dev/gptid/e5c1478c-90a8-11ea-a2bb-0024ecf12fe7': best uberblock found for spa mpool.
txg condensing: txg 9309306, msp[97] 0xfffffff80246ff9000, vdev id 1, spa mpool, smp size 54472, segments
txg condensing: txg 9309306, msp[97] 0xfffffff80261221c00, vdev id 0, spa mpool, smp size 49168, segments
txg condensing: txg 9309306, msp[51] 0xfffffff8024685c800, vdev id 5, spa mpool, smp size 33408, segments
txg condensing: txg 9309306, msp[52] 0xfffffff80246858c00, vdev id 5, spa mpool, smp size 33936, segments
txg condensing: txg 9309306, msp[122] 0xfffffff80246b43000, vdev id 3, spa mpool, smp size 44688, segments
txg condensing: txg 9309306, msp[121] 0xfffffff80246b46c00, vdev id 3, spa mpool, smp size 37472, segments
txg condensing: txg 9309306, msp[123] 0xfffffff80246d66400, vdev id 2, spa mpool, smp size 48360, segments
spa=mpool async request task=32panic: Solaris(panic): zfs: freeing free segment (offset=3435973836800 size=4096)
```

cpuid = 1

KDB: stack backtrace:

```
db_trace_self_wrapper() at db_trace_self_wrapper+0x2b/frame 0xfffffe2022481550
vpanic() at vpanic+0x177/frame 0xfffffe20224815b0
panic() at panic+0x43/frame 0xfffffe2022481610
vcmn_err() at vmn_err+0xcf/frame 0xfffffe2022481740
zfs_panic_recover() at zfs_panic_recover+0x5a/frame 0xfffffe20224817a0
range_tree_remove_impl() at range_tree_remove_impl+0x100/frame 0xfffffe2022481860
space_map_load_callback() at space_map_load_callback+0x73/frame 0xfffffe2022481890
space_map_iterate() at space_map_iterate+0x253/frame 0xfffffe2022481910
space_map_load() at space_map_load+0x91/frame 0xfffffe2022481960
metaslab_load() at metaslab_load+0x65/frame 0xfffffe2022481990
metaslab_preload() at metaslab_preload+0x89/frame 0xfffffe20224819c0
taskq_run() at taskq_run+0x10/frame 0xfffffe20224819e0
taskqueue_run_locked() at taskqueue_run_locked+0x154/frame 0xfffffe2022481a40
taskqueue_thread_loop() at taskqueue_thread_loop+0x98/frame 0xfffffe2022481a70
fork_exit() at fork_exit+0x83/frame 0xfffffe2022481ab0
fork_trampoline() at fork_trampoline+0xe/frame 0xfffffe2022481ab0
--- trap 0, rip = 0, rsp = 0, rbp = 0 ---
```

KDB: enter: panic

[thread pid 0 tid 101780]

Stopped at kdb_enter+0x3b: movq \$0,kdb_why

LKM: Loadable Kernel Module

- Chargement dynamique de modules compilés
- Extension des capacités du noyau
- Tout en gardant le cœur minimal
- Économie d'espace mémoire
- Force une pratique modulaire
- Facilité de débug (sans reboot machine obligatoire)
- Principaux usages:
 - Device drivers: extension de support de nouveaux périphériques
 - Filesystem drivers: extension du support de systèmes de fichiers
 - Syscalls : étend les capacités du noyau en introduisant de nouveaux sys calls
 - Network Drivers: Gestion dynamique de nouvelles technologies réseaux sans avoir à recompiler un noyau entier

Écrire un module noyau: les bases

- Avoir les headers du noyau cible
 - apt install linux-headers-<...>
 - yum install kernel-headers-<...>
 - ls /usr/src/linux<...>/
- Code de base -> init & fini
- Makefile
- Tests
 - modinfo mymodule.ko
 - insmod mymodule.ko
 - rmmod mymodule.ko
 - dmesg || /var/log/kern.log

```
obj-m+=mymodule.o
all:
    make -C /lib/modules/$(shell uname -r)/build/ M=$(PWD) modules
clean:
    make -C /lib/modules/$(shell uname -r)/build/ M=$(PWD) clean
```

```
MODULE_LICENSE("GPL");
MODULE_AUTHOR("adamj");
MODULE_DESCRIPTION("A First module");
MODULE_VERSION("1.0");

static int __init mymod_init(void){
    printk(KERN_INFO "MyMod: Hello there !\n");
    return 0;
}

static void __exit mymod_exit(void){
    printk(KERN_INFO "MyMod: Goodbye !\n");
}
module_init(mymod_init);
module_exit(mymod_exit);
```

DKMS

- Compilation d'un module est dépendante du noyau cible.
- Comment survivre à une mise à jour noyau ?
 - => Recompile, à chaque mise à jour, la liste des modules
- DKMS (Dynamic Kernel Module Support), conçu par Dell en 2003
 - `dkms add -m mymodule -v 1.0.0 -c /dkms.conf`
 - `dkms remove -m mymodule -v 1.0.0 --all`
 - `dkms build -m mymodule -v 1.0.0 -k <kernel_version>`
 - `dkms install/uninstall`
 - `dkms status`
- Description du module dans `dkms.conf`, explicitant comment construire ce module noyau.
- 3 préfixes utilisées par DKMS (par défaut, voir `/etc/dkms/framework.conf`):
 - `/usr/src`: code sources des modules
 - `/lib/modules`: Modules compilés (souvent dans un répertoire « updates »)
 - `/var/lib/dkms`: artifacts de modules (logs, *.ko originaux...)

```
↳ sudo awk -F\' '/menuentry / {print $2}' /boot/grub2/grub.cfg
CentOS Linux (3.10.0-514.6.1.el7.x86_64) 7 (Core)
CentOS Linux (3.10.0-514.2.2.el7.x86_64) 7 (Core)
CentOS Linux (3.10.0-327.36.3.el7.x86_64) 7 (Core)
CentOS Linux (3.10.0-327.36.2.el7.x86_64) 7 (Core)
CentOS Linux (3.10.0-327.36.1.el7.x86_64) 7 (Core)
CentOS Linux (0-rescue-a6dd0bd0c8fc4cba95b0a0996a0e1078) 7 (Core)
```

DKMS: Example de configuration

Configuration d'un driver réseau Linux pour supporter des périphériques Wireless

```
PACKAGE_VERSION="1.0.4"
PACKAGE_NAME="ipw2200"
MAKE[0]="make -C ${kernel_source_dir} SUBDIRS=${dkms_tree}/${PACKAGE_NAME}/${PACKAGE_VERSION}/build modules HOSTAP_SRC=${kernel_source_dir}/3rdparty/hostap/"
CLEAN="make -C ${kernel_source_dir} SUBDIRS=${dkms_tree}/${PACKAGE_NAME}/${PACKAGE_VERSION}/build clean"
BUILT_MODULE_NAME[0]="$PACKAGE_NAME"
BUILT_MODULE_NAME[1]="ieee80211_crypt_ccmp"
BUILT_MODULE_NAME[2]="ieee80211_crypt"
BUILT_MODULE_NAME[3]="ieee80211_crypt_tkip"
BUILT_MODULE_NAME[4]="ieee80211_crypt_wep"
BUILT_MODULE_NAME[5]="ieee80211"
DEST_MODULE_LOCATION[0]="/kernel/drivers/net/wireless/ipw2200/"
DEST_MODULE_LOCATION[1]="/kernel/drivers/net/wireless/ipw2200/"
DEST_MODULE_LOCATION[2]="/kernel/drivers/net/wireless/ipw2200/"
DEST_MODULE_LOCATION[3]="/kernel/drivers/net/wireless/ipw2200/"
DEST_MODULE_LOCATION[4]="/kernel/drivers/net/wireless/ipw2200/"
DEST_MODULE_LOCATION[5]="/kernel/drivers/net/wireless/ipw2200/"
MODULES_CONF_ALIAS_TYPE[0]="eth"
REMAKE_INITRD="no"
AUTOINSTALL="yes"
```