Systèmes d'exploitation

Fonctionnement interne

Jeudi 27 mai et mardi 1er Juin

Thomas Petazzoni – Lolut - http://lolut.utbm.info

Linux Magazine

Série complète sur la construction d'un OS simple : SOS. A partir de ce mois-ci (numéro 62) et pendant 10 ou 11 numéros.

Compilation

Processus de compilation :

- pré-processeur (gcc -E)
  - interprêter toutes les directives # (#include, #define) ...
  - démo
- compilation (gcc -S)
  - transformer le code C en code assembleur pour l'architecture
  - démo
- assemblage (as)
  - transformer le code assembleur en code machine
  - démo + doc Intel
- linkage (ld)
  - rassembler plusieurs fichiers objets
  - démo

Format binaire

Un binaire a une structure :

- Sous Linux : ELF (anciennement a.out)
- Sous Windows : PE

Décomposé en sections

- Code (.text)
- Données (.data)
- Données en lecture seule (.rodata)
- Données non initialisées (.bss)
- Informations pour le linkage dynamique (.got, .plt)
- Sections de relocation (.rel.plt ...)
- Informations de débuggage

Chargement en mémoire

Dans ELF : **Program Header**

- Taille
- Positionnement
- Droits d'accès

Informations interprétées par le système d'exploitation (appel système **exec()**).

Architecture Intel : registres

Registres de travail : eax, ebx, ecx, edx, esi, edi

- ax = 16 bits de poids faible de eax
- al = 8 bits de poids faible de ax
- ah = 8 bits de poids fort de ax

Pointeur de pile : esp

Pointeur de **frame** : ebp

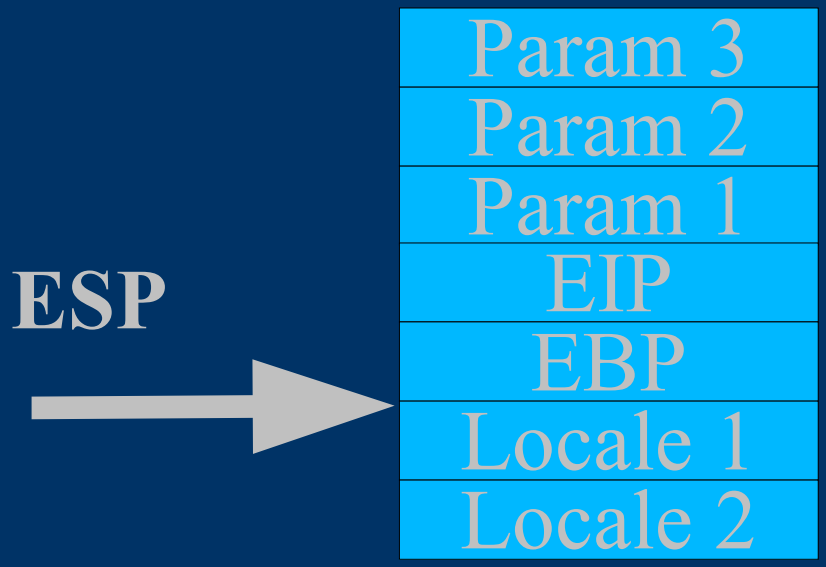
Pointeur d'instruction : eip

Registres de segment : cs, ds, es, fs, gs, ss (16 bits)

Conventions d'appel

Avec **gcc** :

- Passage des paramètres sur la pile
- Allocation des variables locales sur la pile
- Utilisation de **ebp** comme pointeur de frame
- Valeur de retour dans **eax**



Bibliothèques dynamiques / statiques

Statique : tout le code du programme est inclu dans l'éxécutable.
Dynamique : le code des bibliothèques est partagé entre les applications

Mécanisme de relocation.
.got, .plt

Démo compilation statique, dynamique, **ldd,** / proc/self/maps

Rôle de l'OS

- Gérer les ressources
  - Processeur
  - Mémoire
  - Périphériques
- Les uniformiser
- Les rendre accessible aux applications

Différents types de noyau

- **Monolithique** : un seul bloc réalisant toutes les fonctionnalités (ex: Linux)
- **Micro-noyau** : un petit noyau et de nombreux serveurs communiquants entre eux (ex: Hurd)
- Systèmes à **composants**
- ...

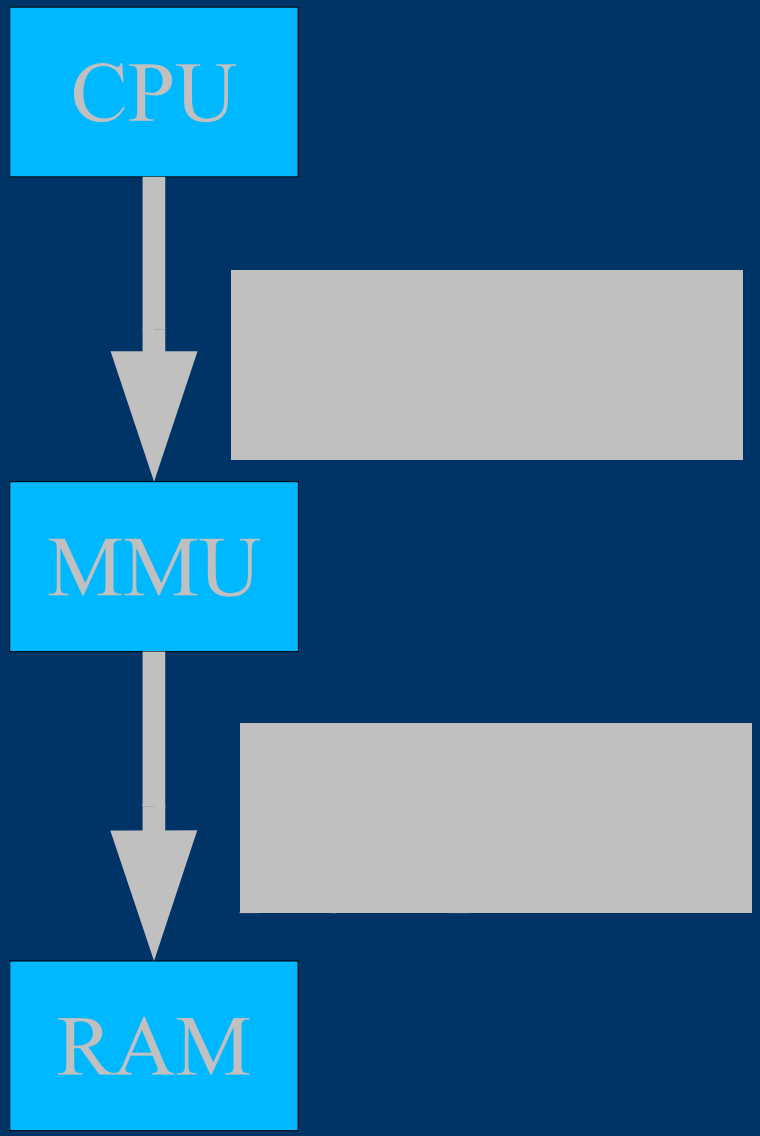
Adresses physiques/effectives

En mémoire :
- Instructions (code)
- Données

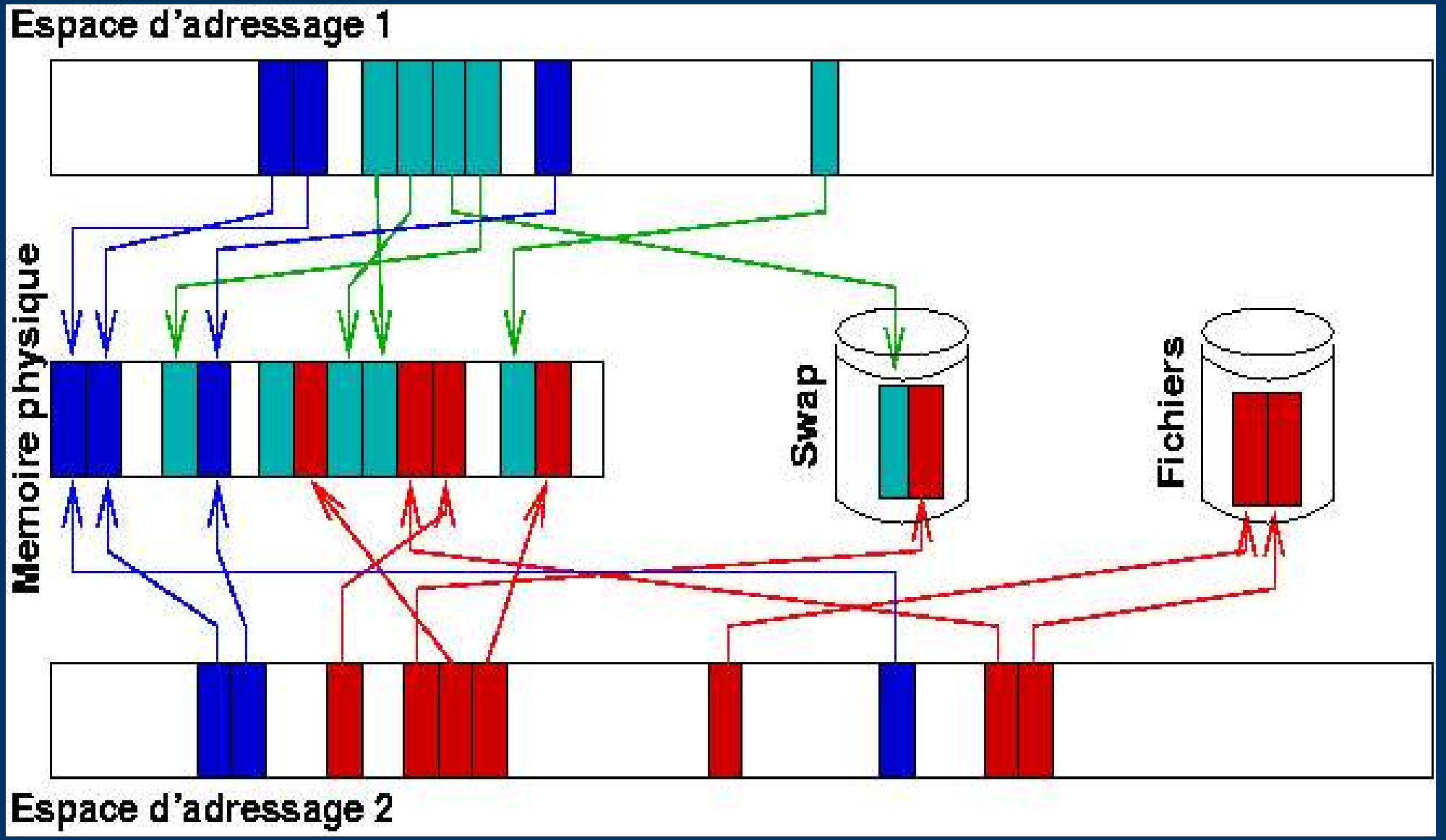
Pour faire du multitâche robuste, 2 types d'adresses :
- Adresses **physiques**
- **Adresses « effectives »**, relatives à l'**espace d'adressage** courant

Conversion **physique => effective** : MMU.

Adresses physiques/effectives



Adresses effectives/physiques



Adresses effectives/physiques

La traduction effectives => physiques s'effectue à l'aide de **tables de traduction,** fournie par le système d'exploitation.

Sur x86, deux mécanismes se combinent :

- segmentation
- pagination

Petit retour sur x86

Deux modes de fonctionnement du processeur :

- **réel**
  - disponible depuis le 8086, 20 bits d'adresse, 1Mo RAM
  - segmentation fixée (segments de 64 Ko)
- **protégé**
  - disponible depuis le 386, 32 bits d'adresse, 4 Go RAM
  - segmentation “custom”

x86 : segmentation (1)

Segments définis par des tables :

- GDT (Global Descriptor Table)
- LDT (Local Descriptor Table)

Adresse définie par **segment:offset**, segment étant un sélecteur de segment dans une des tables.

Les GDT et LDT contiennent des descripteurs de segments.

x86 : segmentation (2)

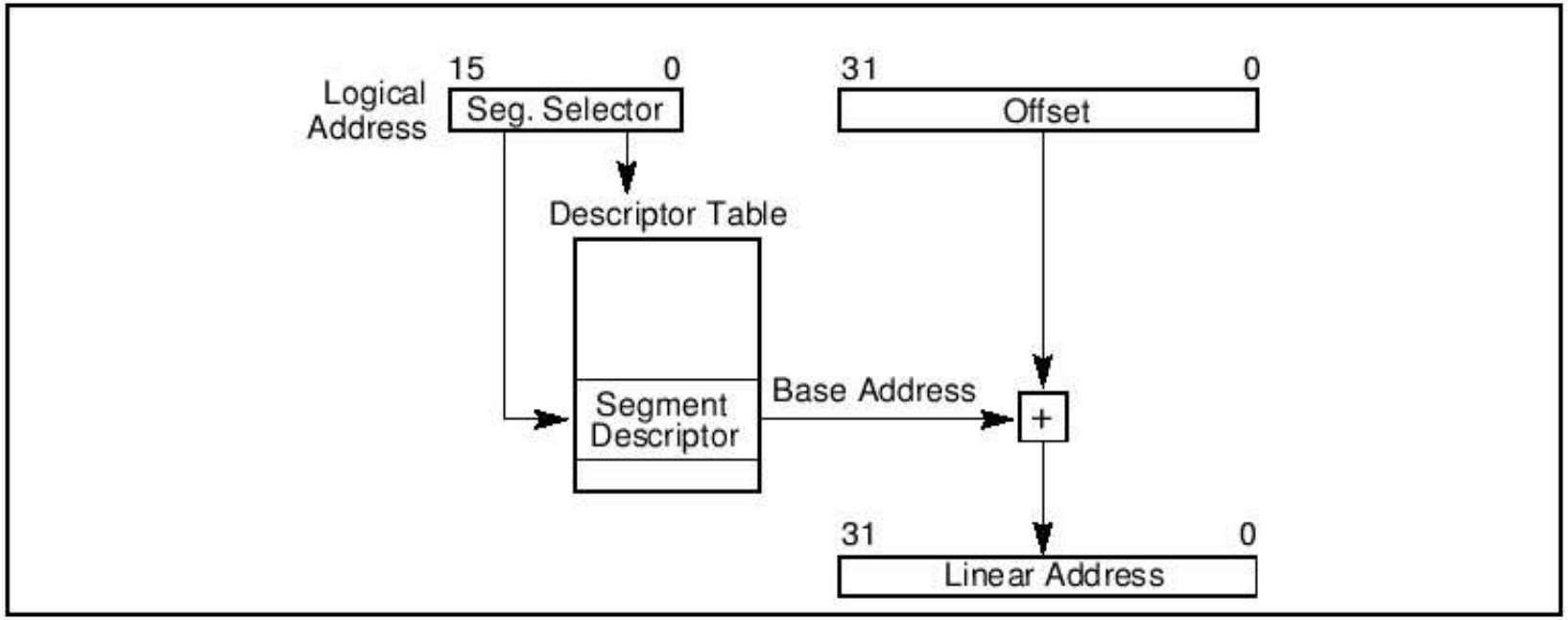


Figure 3-5. Logical Address to Linear Address Translation

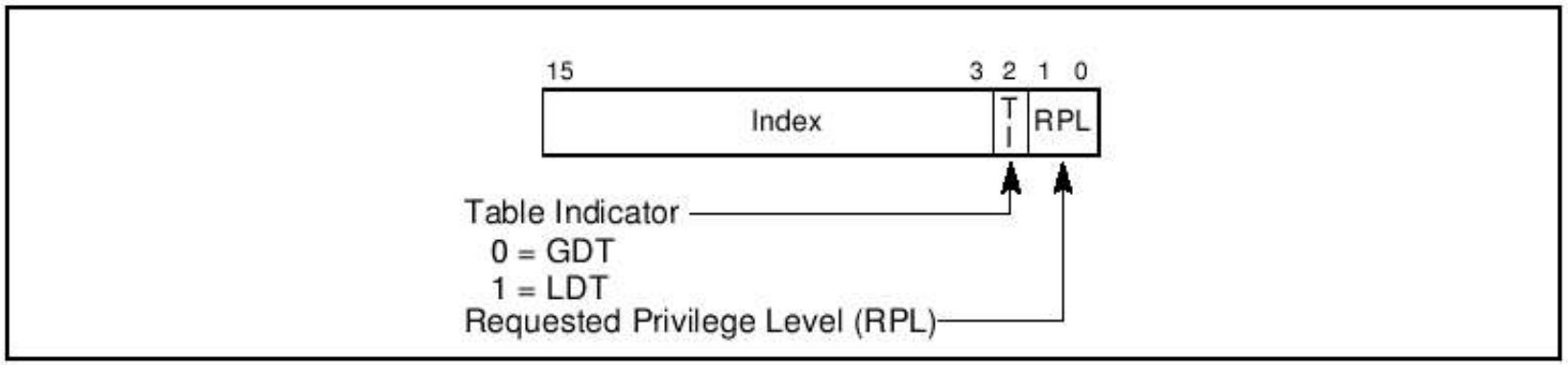


Figure 3-6. Segment Selector

x86 : segmentation (3)

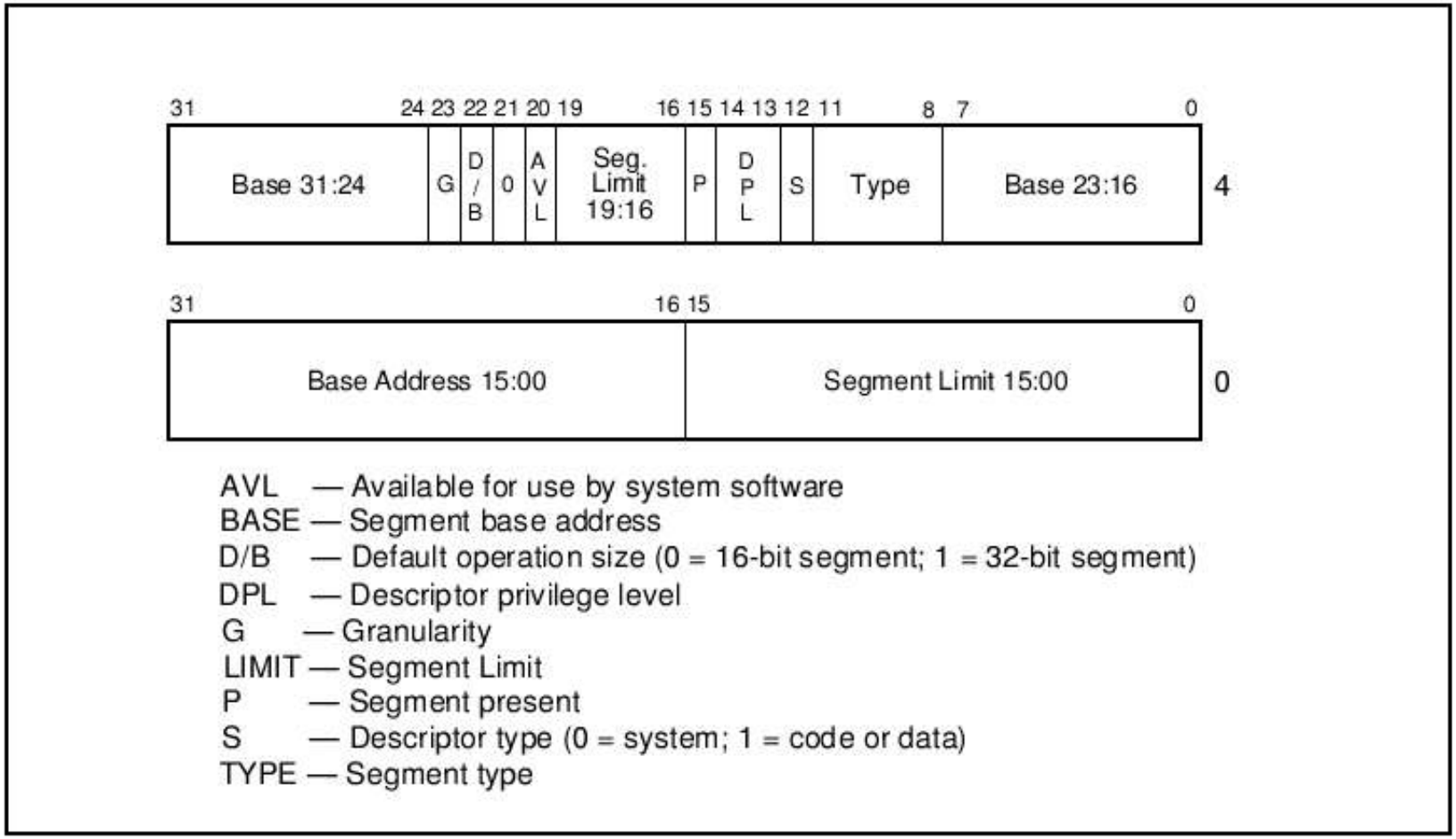


Figure 3-8. Segment Descriptor

+ Démo code KOS

x86 : pagination (1)

Sur x86, pagination à 2 niveaux

- PD : Page Directory
- PT : Page Table

Chaque espace d'adressage dispose d'un “Page Directory” dont l'adresse est dans le registre CR3.

La conversion réalisée par la MMU parcourt ces différentes tables pour déterminer l'adresse physique.

Erreur lors de la traduction signalée par le processeur à l'OS.

x86 : pagination (2)

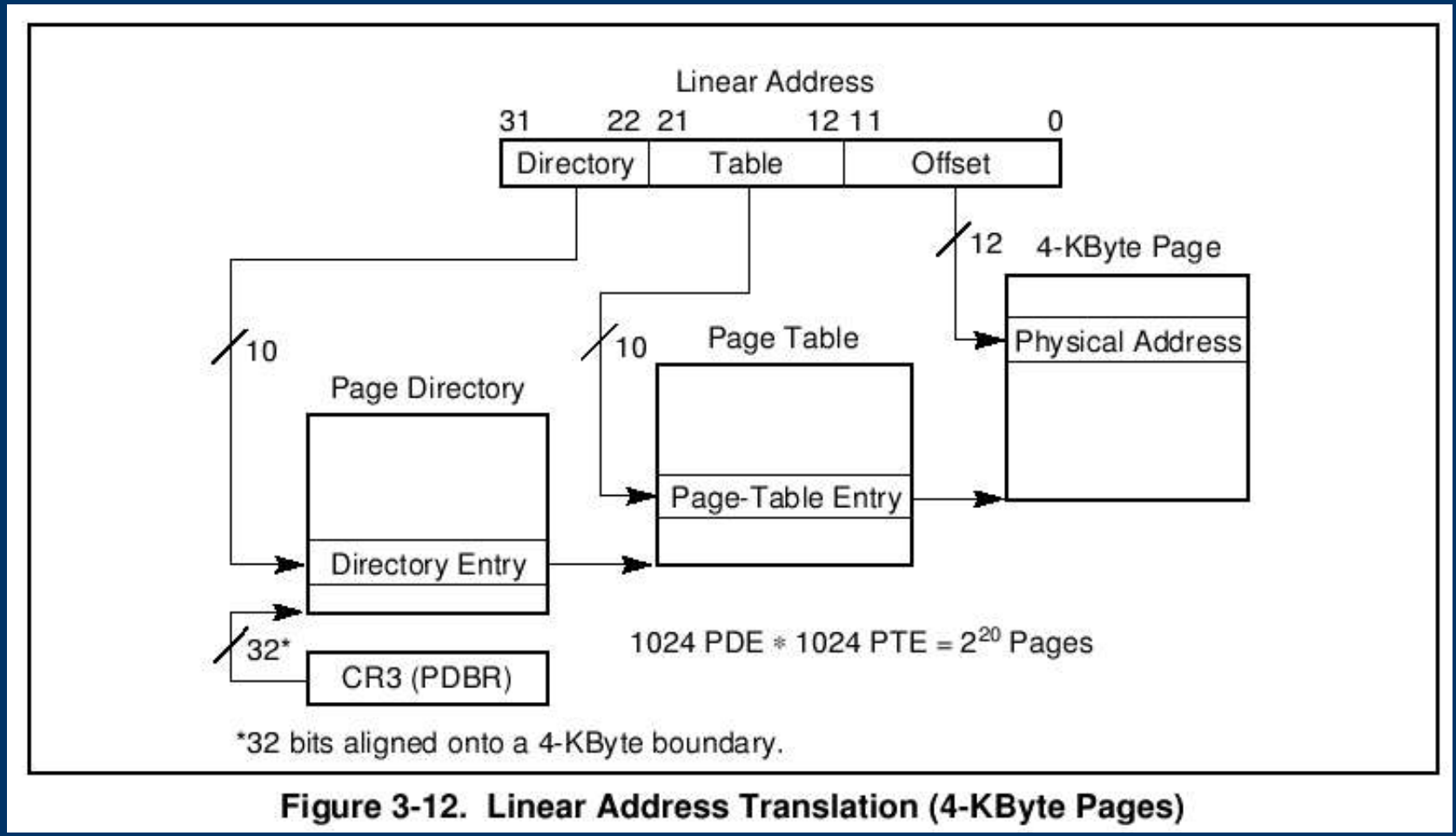


Figure 3-12. Linear Address Translation (4-KByte Pages)

x86 : pagination (3)

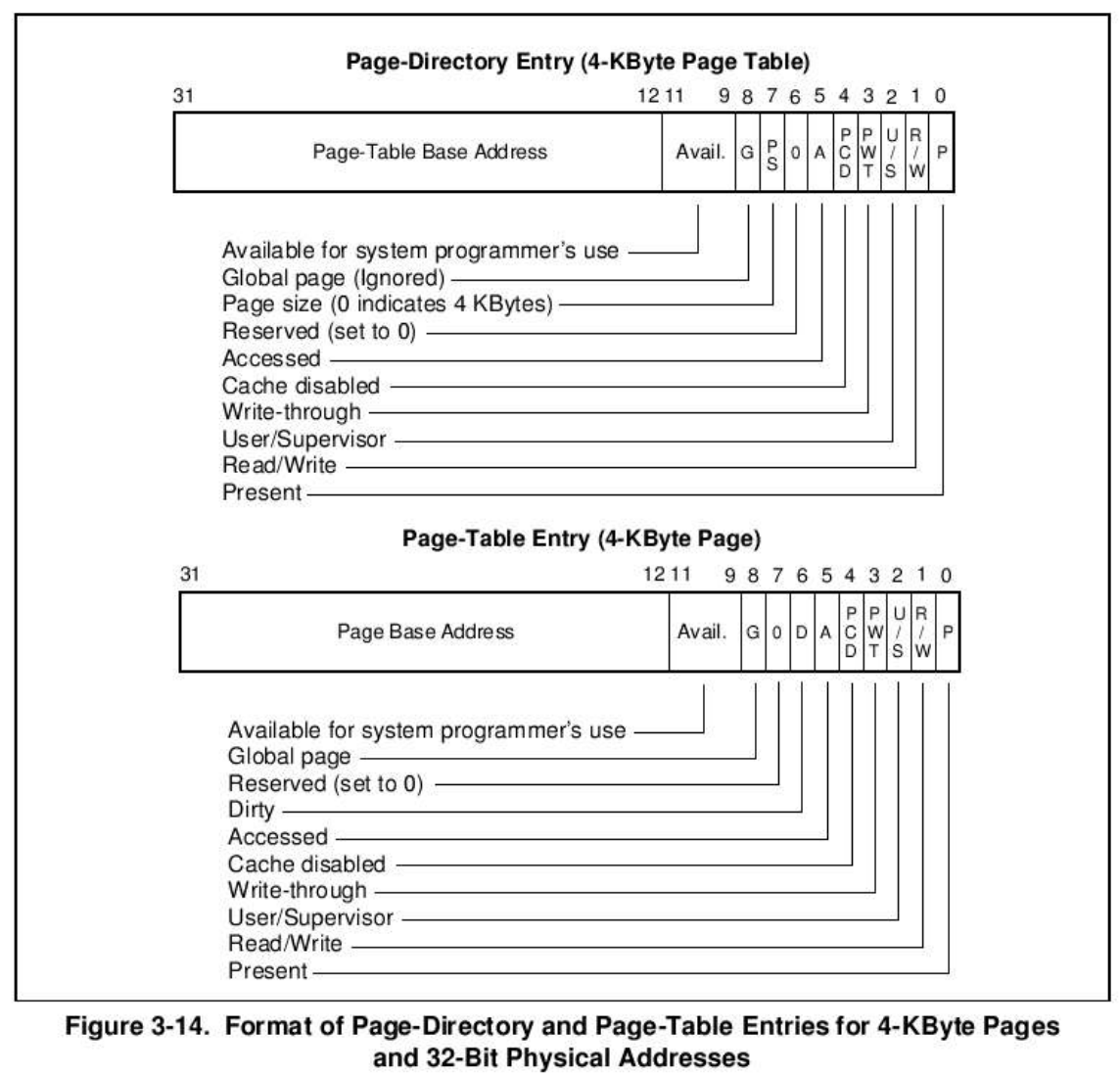


Figure 3-14. Format of Page-Directory and Page-Table Entries for 4-KByte Pages and 32-Bit Physical Addresses

x86 : résumé

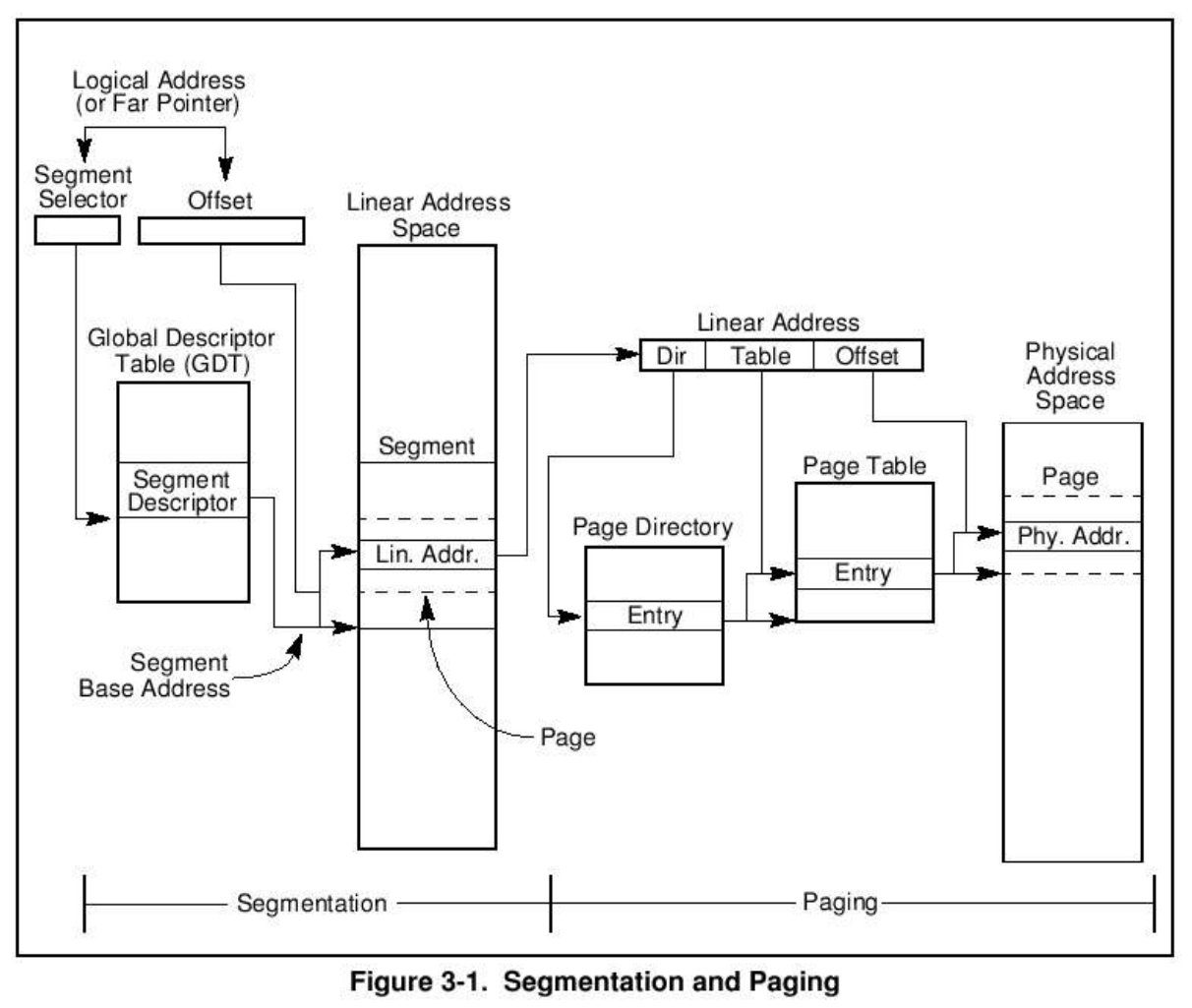


Figure 3-1. Segmentation and Paging

Mémoire virtuelle : pourquoi ?

- Espaces d'adressages différents
  - Cloisonnement
  - Adresses fixes
- Partage de mémoire possible
- Taille mémoire effective >> taille mémoire physique
=> Swap

Mémoire virtuelle : zones

Découpage des 4 Go en 2 zones :

- une zone pour le noyau, identique dans **tous** les espaces d'adressage
- une zone pour l'application concernée, spécifique à chaque espace d'adressage

Sous Linux :

- 1 Go pour le noyau (identity mapping)
- 3 Go pour les applications.

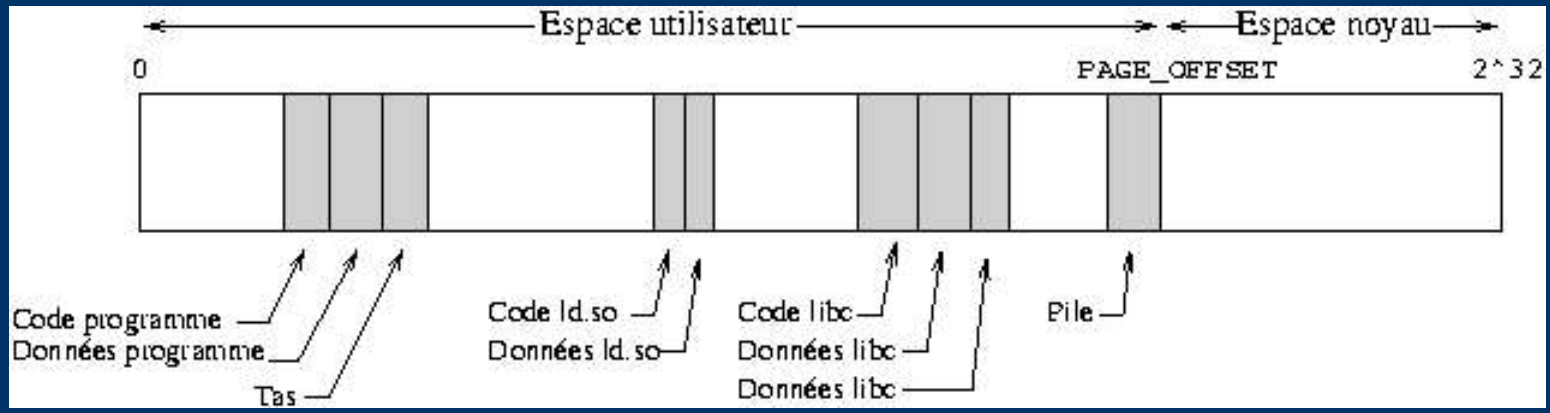
Mémoire virtuelle : VR

Découpage de la zone des applications en **régions virtuelles**, pour séparer les divers éléments chargés en mémoire :

- Code du programme
- Données du programme
- Code des bibliothèques partagées
- Données des bibliothèques partagées
- Tas
- Pile

Mémoire virtuelle : VR

```
thomas@crazy:~$ cat /proc/self/maps
08048000-0804c000 r-xp 00000000 03:02 12136      /bin/cat
0804c000-0804d000 rw-p 00003000 03:02 12136      /bin/cat
0804d000-0806e000 rwxp 00000000 00:00 0
40000000-40016000 r-xp 00000000 03:02 26269      /lib/ld-2.3.2.so
40016000-40017000 rw-p 00015000 03:02 26269      /lib/ld-2.3.2.so
40017000-40018000 rw-p 00000000 00:00 0
4001f000-4014d000 r-xp 00000000 03:02 24198      /lib/libc-2.3.2.so
4014d000-40156000 rw-p 0012d000 03:02 24198      /lib/libc-2.3.2.so
40156000-40159000 rw-p 00000000 00:00 0
bffff000-c0000000 rwxp 00000000 00:00 0
ffffe000-fffff000 ---p 00000000 00:00 0
```



Mémoire virtuelle : VR

Une région virtuelle correspond soit :

- à des données d'un fichier : **file mapping**
- à des données allouées dynamiquement : **mémoire anonyme**

File mapping : backing store = fichier
Mémoire anonyme : backing store = swap

Démo mmap-fichier + mmap_anonymous

Mémoire virtuelle : fork

Création de nouveaux processus : **fork()**
Duplication de l'espace d'adressage, donc des régions virtuelles.

Deux attributs pour les régions virtuelles :

- **MAP_SHARED** : les modifications sont partagées avec les fils
- **MAP_PRIVATE** : les modifications ne sont pas partagées avec les fils

Mécanisme de **COW**, Copy-On-Write pour la différenciation.

Mémoire virtuelle : défaut de page

Défaut de page :

- Pas de page physique pour l'adresse virtuelle accédée
- Accès non autorisé

=> exécution d'une routine de résolution dans le noyau

Traitement :

- Page hors région => SIGSEGV
- Page en lecture seule accédée en écriture
  - Si région en lecture/écriture => COW
  - Sinon => SIGSEGV
- Sinon
  - *demand paging*

Mémoire ... au niveau utilisateur

Allocation et libération de mémoire en utilisant **malloc()** et **free()**.

Allocateur de mémoire implémenté dans la bibliothèque standard C.

Utilise les appels systèmes **mmap()** et **brk()** pour allouer de la mémoire, puis la découpe en petits éléments.

Thread

Application = un ou plusieurs flots d'instruction
Processeur = éxécute un flot d'instruction
Multitâche = illusion d'éxécution en parallèle

Thread = un flot d'instruction

Pour passer d'un flot à l'autre : **changement de contexte**

Thread (2)

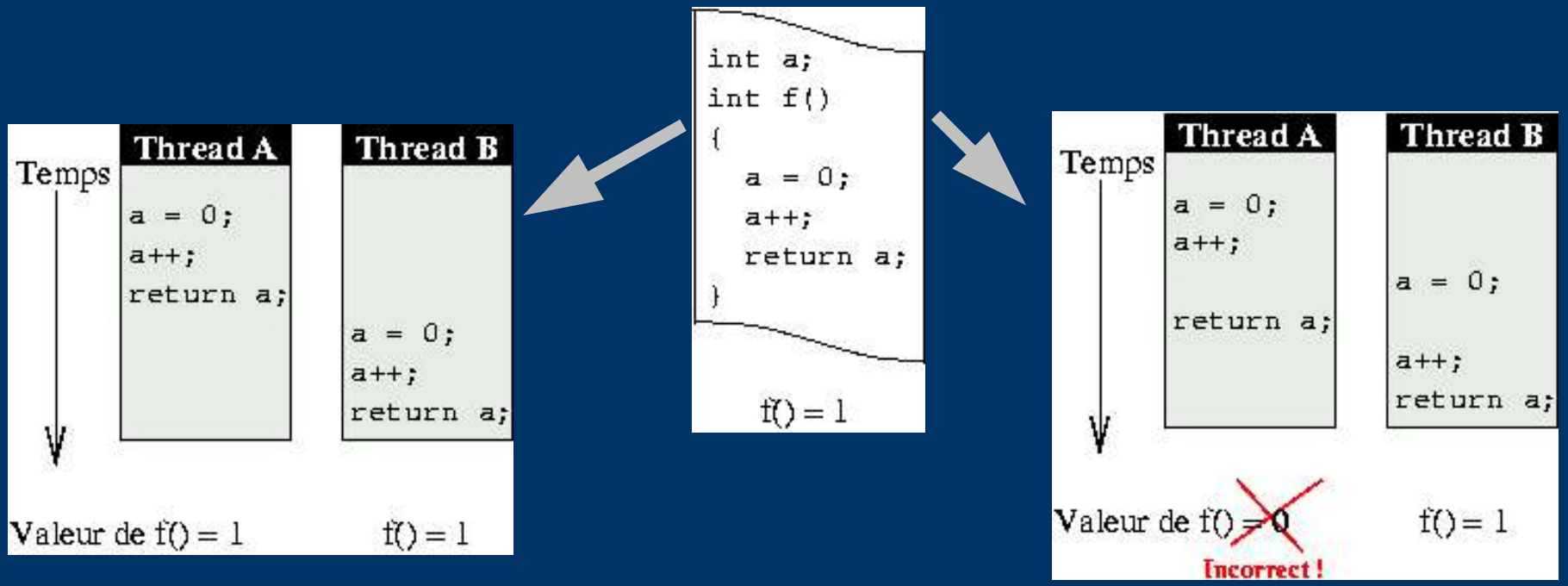
Gestion des threads : **ordonnanceur** ou **scheduler**

État du thread : prêt, bloqué ou en éxécution, en fonction des ressources demandées et disponibles.

Sources de blocage :
- matérielles
- logiques
- virtuelles

Mécanisme de synchronisation logicielle pour protéger les ressources matérielles et logiques.

Thread (3) : Synchronisation



Solution : synchroniser

- Protection de ressources partagées
- Points de rendez-vous temporels

Thread (4)

Deux types de threads :

- **Threads noyau :** mode privilégié, dans n'importe quel espace d'adressage. Ex: keventd, kswap, ksoftirqd_CPU0
- **Threads utilisateur :** mode utilisateur, dans un espace d'adressage donné.

Thread (5)

Création par fork() ou clone() *kernel/fork.c:do_fork()*

Fork : nouveau thread dans un nouvel espace d'adressage par duplication.

Clone : nouveau thread partageant certains éléments avec le contexte d'exécution appelant.

Thread (6)

La bibliothèque Linux Threads (pthread_*) utilise clone().

Suite à clone() ou fork(), les espaces d'adressage sont identiques : utilisation de exec() pour réinitialiser l'espace d'adressage avec l'image d'un nouveau programme fs/exec.c:do_execve().

Terminaison d'un thread : exit(), *kernel/exit.c:exit()*

Thread (7)

Changement de contexte T1 => T2

- sauvegarde contexte T1
- restauration contexte T2

En général, le contexte est sauvegardé sur la pile. (Code KOS).

Ordonnanceur à temps partagé : répartition équitable du temps, selon des priorités.

L'ordonnanceur est dans *kernel/sched.c:schedule()*.

Appellé périodiquement, ou sur blocage.

Appel système

Applications utilisateur => noyau : **appel système**.

Sous Linux : interruption logicielle 0x80.

Numéro de fonction demandée dans *eax*.

Table des appels systèmes : *arch/i386/entry.S:syscall_table*

Le traitement des appels se fait sur une autre pile. Chaque thread a donc 2 piles :

- une **pile utilisateur**, pour l'exécution de l'application
- une **pile noyau**, pour l'exécution des appels systèmes et la sauvegarde du contexte

=> Démo code appel système manuel.

Synchronisation dans le noyau

Opérations atomiques

- Évitement des accès concurrents : impossible d'être interrompu.
- Deux types :
  - Sur les bits (test/set/change/clear)
  - Sur les variables (set/sub/inc_and_test)

Spinlocks

- Désactivation des interruptions
- Opération exécutée en un seul tenant
- En SMP, attente active
- Impossible de bloquer

Sémaphores

- Compteur
- Pas d'attente active
- Possibilité de bloquer

Synchronisation pour l'utilisateur

Sémaphores

- Même principe que les sémaphores noyau
- semget(), semctl(), semop() (ipc/sem.c)

Files de messages

- Mécanisme de synchro et de communication
- msgget(), msgctl(), msgsnd() (ipc/msg.c)

Fichiers

- Moyen de communication (*pipe* et *fifo*)
- Moyen de synchro (*fcntl)*

Signaux

- Communication basique : déclenchement d'une routine dans d'autres processus.

Mémoire partagée

- Région virtuelle identique dans plusieurs espaces d'adressage.

Pilotes de périphériques

Pilote de périphérique : représenter une ressource matérielle sous la forme d'une ressource logique.

Certains pilotes accessibles directement par l'utilisateur, d'autres non.

Types de périphériques :

- bloc
- caractère

Mais également, des périphériques internes, comme les cartes réseaux, utilisées via des piles de protocoles.

Interaction avec le matériel

- Ports d'entrées sorties
  - Petites zones pour dialoguer avec le périphérique
  - *inb / outb*
- Périphériques ou zones mappés
  - Une partie de la mémoire physique est “déroutée”
  - Pour dialoguer, écrire et lire dans la mémoire physique
- Interruptions (IRQ)
- DMA : Direct Memory Access
  - Transfert de données direct entre périphérique et mémoire, sans utiliser le processeur.

Interruptions

Signalement d'événements : interruptions

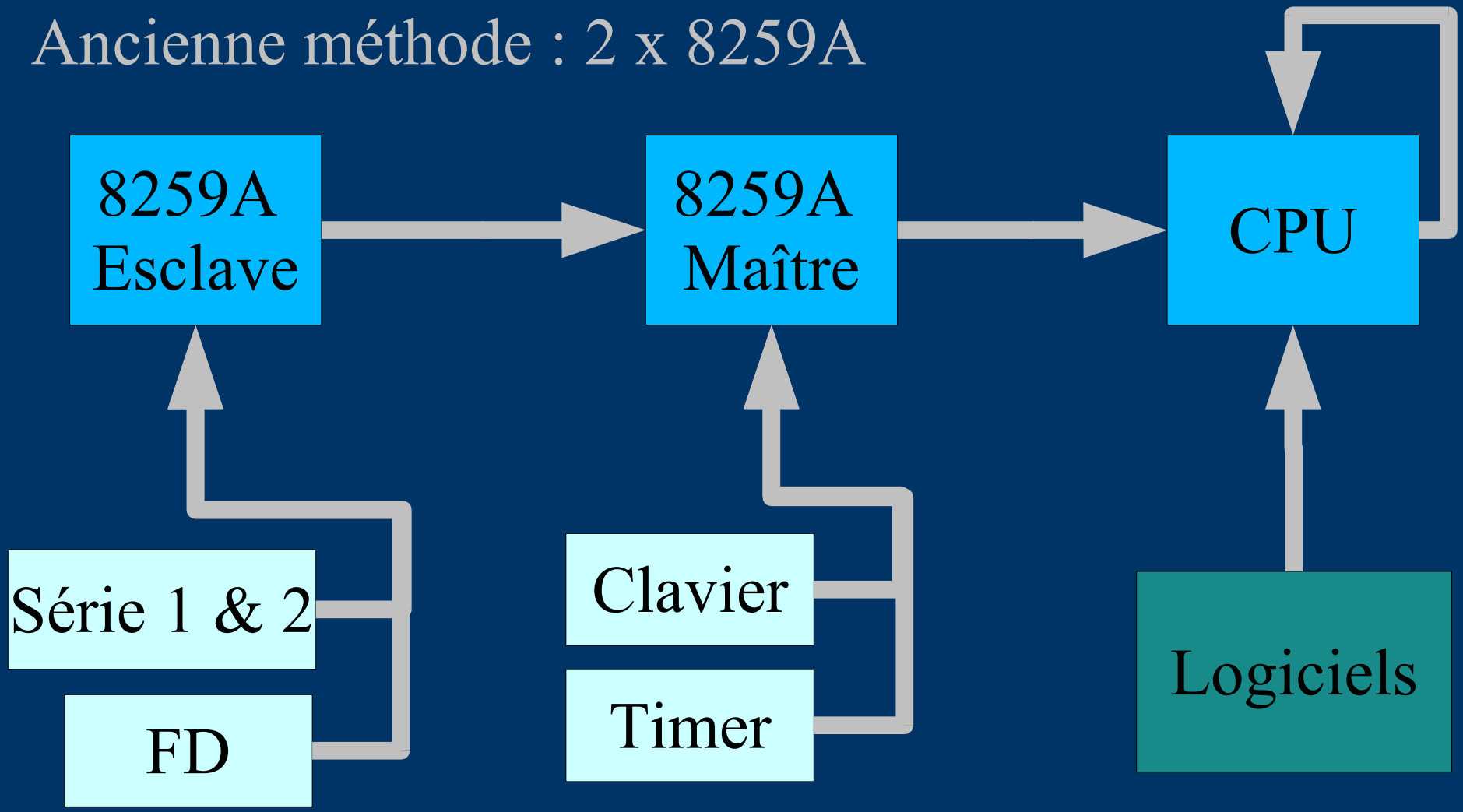
- **Exceptions** : interne au processeur
  - Division par zéro
  - Défaut de page
  - Erreur de protection
- **IRQ** : du matériel
  - Timer
  - Clavier
  - Disque
  - Carte réseau
  - ..
- **Appel système** : du logiciel

Report du traitement :

- tasklets et softirqs

Interruptions : sur PC

Ancienne méthode : 2 x 8259A



Nouvelle méthode : APIC

Interruptions sur x86

Jusqu'à 256 interruptions
Exceptions de 0 à 31
IRQ 0 à 16 routées où on le souhaite

Configuration des interruptions dans l'**IDT** (Interrupt Descriptor Table).

Trois types de descripteur :

- Interrupt Gate
- Trap Gate
- Task Gate

Interruptions sur x86

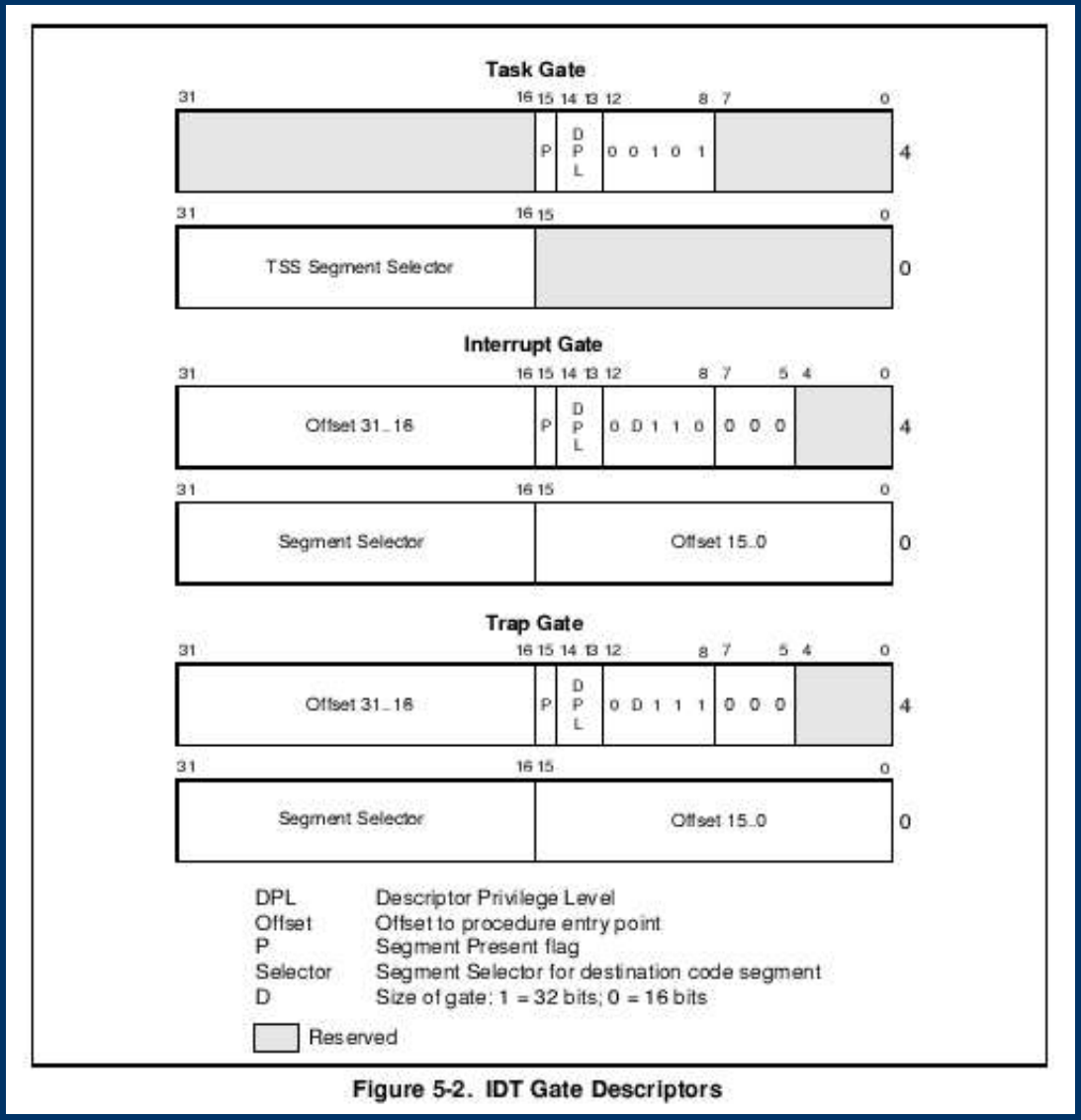


Figure 5-2. IDT Gate Descriptors

Interruptions sur x86

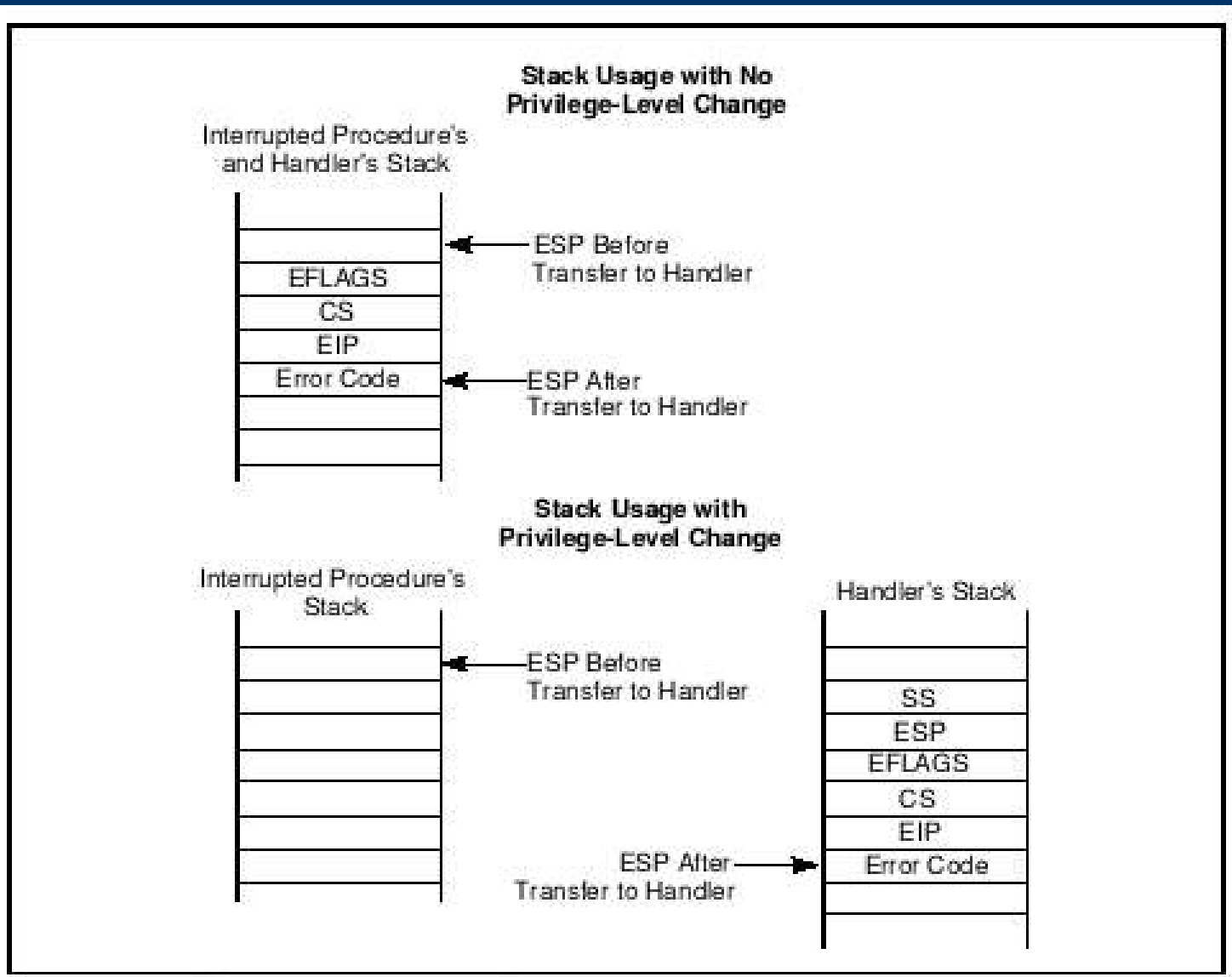


Figure 5-4. Stack Usage on Transfers to Interrupt and Exception-Handling Routines

Système de fichiers

VFS : Virtual File System

- cohérence de l'espace de nommage global
- interface de programmation unique
- intégration avec les autres sous-systèmes, en particulier mémoire virtuelle

Un schéma vaut mieux qu'un long discours...

VFS : Virtual File System