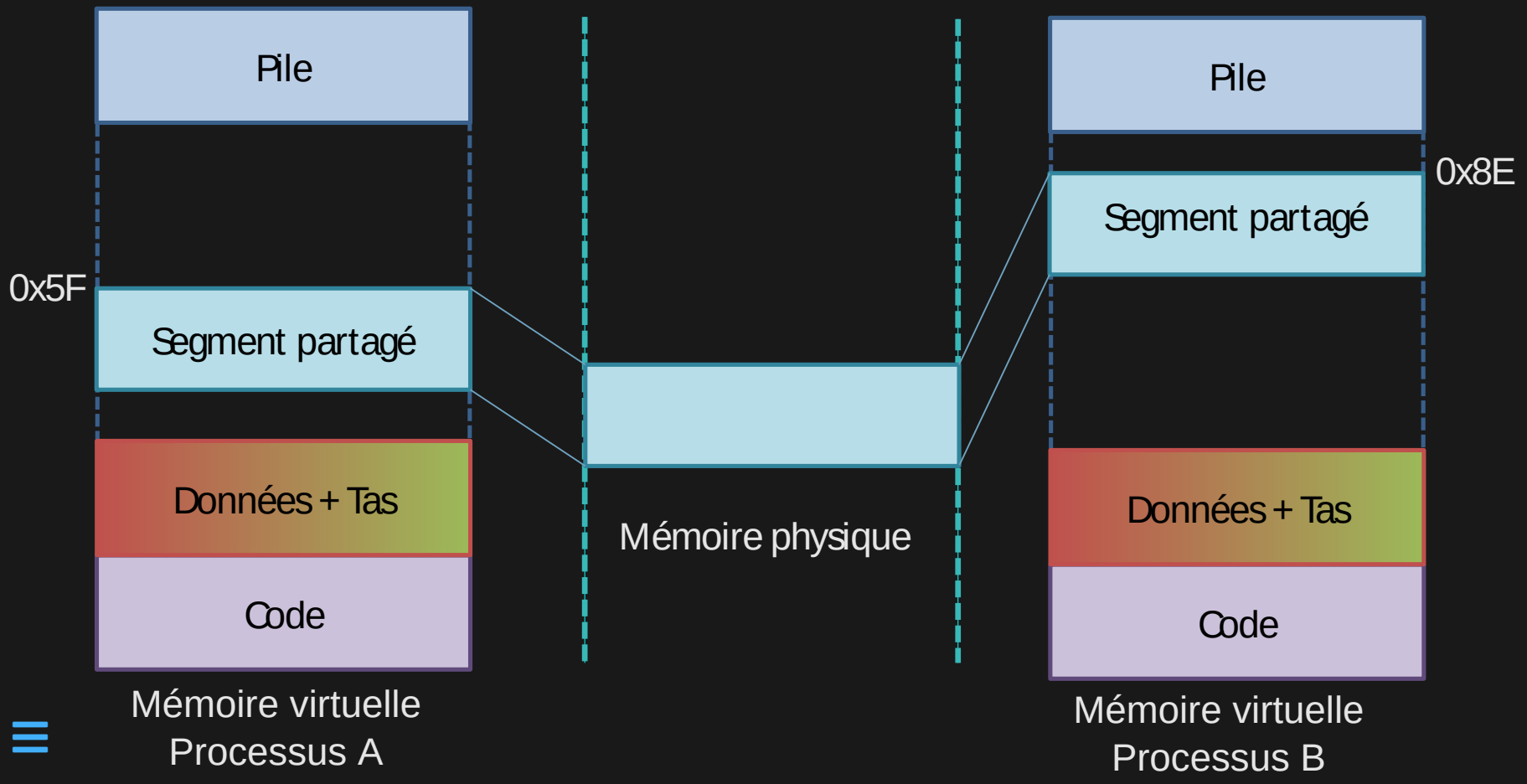
MÉMOIRE PARTAGÉE

Guillaume Chanel

VUE GÉNÉRALLE

Deux processus peuvent partager un segment en y associant des espaces d'adressage virtuel différents:



SEGMENT DE MÉMOIRE PARTAGÉE

Un segment partagé:

- est alloué par le noyau sur requête d'un processus;
- peut être intégré dans les espaces d'adressage d'autres processus qui se l'attachent à des adresses potentiellement différentes;
- contient forcément un nombre entier de pages (ce qui est vrai pour tous les segments).

TYPES DE MÉMOIRES PARTAGÉES

Historiquement il existe deux moyens UNIX / Linux pour créer des mémoires partagées entre deux processus sans relation parent / enfant.

- XSI Inter Process Communication (IPC)
  - est héritée de System 5;
  - est basée sur un système de clef et d'identificateurs;
  - n'est pas limité au mémoires partagées mais est aussi utilisé pour créer des files de messages et des sémaphores.
- **POSIX shared memory objects**:
  - **est basé sur la fonction mmap**;
  - **utilise des descripteurs de fichier**;
  - **est utilisé par Linux pour gérer les mémoires partagées**.

CRÉATION DE MÉMOIRE PARTAGÉE

Pour créer un nouvel objet "mémoire partagé" ou pour ouvrir un objet existant on utilise:

```c
#include <sys/mman.h>
#include <sys/stat.h> // For mode constants
#include <fcntl.h>    // For O_* constants
int shm_open(const char *name, int oflag, mode_t mode);
```

- retourne un descripteur de fichier représentant la mémoire partagée;
- `name` n'est pas un nom de fichier standard mais doit commencer par "/", on retrouvera la mémoire partagée dans /dev/shm/name;
- `oflag` et `mode` fonctionnent comme pour open (notamment O_RDONLY, O_RDWR, O_CREAT, O_EXCL). **Attention à ne pas écraser une mémoire partagée crée par un autre utilisateur !**

TAILLE DE LA MÉMOIRE

Le descripteur obtenu par `shm_open` crée un objet de mémoire partagée POSIX mais avant sont utilisation il faut lui donner une taille. Cela est effectué par l'appel à:

```
#include <unistd.h>
int ftruncate(int fd, off_t length); // length est la taille en octets
```

`fd`: un descripteur de fichier (ici une mémoire partagée mais fonctionne aussi sur les fichiers); `length`: la nouvelle taille de la mémoire.

MAPPING DE LA MÉMOIRE

On utilise les fonctions habituelles pour effectuer un mapping de la mémoire dans l'espace virtuel du processus:

```
#include <sys/mman>
void *mmap(void *addr, size_t length, int prot, int flags, int fd, off_t offset);
int munmap(void *addr, size_t len);
```

DESTRUCTION DE LA MÉMOIRE

Pour supprimer la référence à la mémoire partagée on utilise:

```
int shm_unlink(const char *name);
```

Comme pour l'appel à unlink, uniquement la référence est supprimée (i.e. `/dev/shm/name`). La mémoire ne sera effectivement détruite que si elle est désassociée par `munmap`.

EXEMPLE

Le programme ci-dessous montre un exemple de construction de mémoire partagée:

- shm.h
- creator.c
- helper.c
- makefile

Toutefois il souffre d'un défaut. Lequel ?

CONDITIONS DE COURSE

Lorsque deux processus coopèrent (e.g. partagent des données) il faut faire extrêmement attention aux **conditions de course**:

- les deux processus peuvent être vu comme concurrent sur l'accès aux données;
- si il n'y a pas de contrôle d'accès sur ces données il peut y avoir conflit dans leur utilisation.

Cela est partiellement dû au fait qu'un processus peut être **suspendu par l'ordonnanceur au milieu d'une opération, laissant les données partagées dans un état intermédiaire**.

Seule les **opérations dites atomiques** garantissent d'être exécutée « en une fois »: le processeur n'est jamais alloué à un autre processus pendant leur exécution.

RÉSOLUTION DES CONDITIONS DE COURSE

Afin de régler les problèmes de concurrence il faut utiliser des mécanismes de coordination tel que:

- des variables communes;
- des signaux;
- des mécanismes dédiés:
  - la mémoire partagée étant représentée par un descripteur de fichier il est possible d'utiliser les `lock` comme pour les fichiers;
  - il existe d'autre mécanismes comme les sémaphores ou les mutex:

```
#include <fcntl.h>
#include <sys/stat.h>
#include <semaphore.h>
sem_t *sem_open(const char *name, int oflag);
sem_t *sem_open(const char *name, int oflag, mode_t mode, unsigned int value);
```