Documentation Technique
Ordonnanceur Préemptif Ring 3

Julie BOURGEAIS Célia LAVERGNE

31 décembre 2025

Table des matières

1 Introduction

Ce document décrit l'implémentation d'un micro-noyau capable de gérer deux tâches utilisateur (`T_1` et `T_2`) en Ring 3.
Le système met en œuvre :

- un ordonnanceur préemptif,
- une isolation mémoire via la pagination,
- une zone mémoire partagée,
- un mécanisme d'appels système.

2 Cartographie Mémoire

L'organisation de la mémoire repose sur une séparation entre l'espace noyau (Ring 0, privilégié) et l'espace utilisateur (Ring 3, privilèges limités).
Le noyau et les tâches utilisateur utilisent un modèle de **segmentation flat**, tout en s'appuyant sur la pagination pour l'isolation mémoire et la protection des tâches. Chaque tâche dispose d'une pile noyau distincte utilisée lors des interruptions, et d'une pile utilisateur dédiée à l'exécution normale, conformément au modèle x86. Les tâches sont alignées sur des frontières de 4 Mo afin de simplifier la configuration des tables de pages et d'éviter les chevauchements.

Afin de visualiser clairement l'adressage mémoire choisi et la configuration des segments GDT associés, le tableau suivant présente l'ensemble des éléments du système, leurs adresses physiques et virtuelles, ainsi que les privilèges et segments utilisés.

Élément	**Adresse Physique**	**Adresse Virtuelle**	**Taille**	**Privilège**	**Segment GDT**
Noyau (code et données)	0x000000	Identity mapped	4 Mo	Ring 0	CS=1, DS=9
Tâche 1 - PGD	0x00610000	Identity mapped	4 Ko	Ring 0	
Tâche 1 - Noyau	0x00611000	Identity mapped	4 Ko	Ring 0	
Tâche 1 - Pile noyau[1]	0x00614000	Identity mapped	4 Ko	Ring 0	
Tâche 2 - PGD	0x00620000	Identity mapped	4 Ko	Ring 0	
Tâche 2 - Noyau	0x00621000	Identity mapped	4 Ko	Ring 0	
Tâche 2 - Pile noyau	0x00624000	Identity mapped	4 Ko	Ring 0	
Code user1	0x400000 - 0x7FFFFF	Identity mapped	4 Mo	Ring 3	CS=4
Code user2	0x400000 - 0x7FFFFF	Identity mapped	4 Mo	Ring 3	CS=6
Tâche 1 - Pile utilisateur[2]	0x00800000	Identity mapped	4 Ko	Ring 3	SS=5
Tâche 2 - Pile utilisateur	0x00801000	Identity mapped	4 Ko	Ring 3	SS=7
Page partagée	0x00802000	Identity mapped	4 Ko	Ring 3	DS=8
Tâche 1 - Compteur partagé	0x00802000	0xB0001000	4 Ko	Ring 3	DS=8
Tâche 2 - Compteur partagé	0x00802000	0x03A0C000	4 Ko	Ring 3	DS=8
Noyau (pile)	0x00C00000	Identity mapped	4 Ko	Ring 0	SS=9

TABLE 1 – Cartographie de l'adressage mémoire et configuration des segments GDT

Nous avons choisi d'identity mappé pour assurer une continuité de l'exécution. En effet, l'identity mapping permet d'accéder directement aux adresses physiques connues sans translation compliqué. De plus, lors de l'activation de la pagination, si l'instruction suivante n'est pas mappée à la même adresse virtuelle que son

adresse physique, le processeur génère un Triple Fault.

2.1 Pagination

Le système utilise une pagination à deux niveaux (PGD → PTB → Pages). Chaque tâche possède son propre répertoire de pages :

— `T_1` : PGD à `0x610000`
— `T_2` : PGD à `0x620000`

Les deux adresses virtuelles pointent vers la même page physique (`0x802000`) avec les droits `PG_USR | PG_RW`, mais avec des adresses virtuelles distinctes pour chaque tâche :

— `T_1` : 0xB0001000
— `T_2` : 0x03A0C000

3 Choix d'Implémentation Logicielle

3.1 Gestion des Tâches

Chaque tâche est représentée par une structure `process_t` (définie dans intr.c) contenant :

— `kstack_ptr` : le pointeur vers la valeur ESP dans la pile noyau
— `cr3` : le registre cr3

Les piles utilisateur et noyau sont distinctes pour chaque tâche, garantissant l'isolation et la sécurité des accès mémoire.

3.2 Changement de Contexte

Le changement de tâche est déclenché par l'interruption du timer (IRQ0)(définie dans intr.c) qui effectue :

1. La détection du niveau de privilège courant (`CS%3`) pour savoir si l'interruption vient du noyau (Ring 0) ou d'une tâche utilisateur (Ring 3). Cette information permet de choisir correctement où sauvegarder le contexte.
2. La sauvegarde du contexte courant sur la pile noyau de la tâche interrompue.
3. La mise à jour de la variable `current_task` pour pointer vers la tâche suivante.
4. Le chargement de `CR3` avec le PGD de la nouvelle tâche.
5. La mise à jour du champ `esp0` dans le TSS pour la gestion des interruptions futures.

Le gestionnaire d'IRQ0 distingue une interruption noyau ou utilisateur via le niveau de privilège courant, afin de sauvegarder et restaurer le contexte approprié.

3.3 Appels Système

L'appel système est déclenché par `int 0x80` :

— L'adresse du compteur est passée via le registre `EAX`
— Le noyau incrémente la valeur et l'affiche
— L'IDT autorise l'appel avec un DPL de 3

3.4 Initialisation du Ring 3

Le passage initial en Ring 3 est réalisé en forgeant manuellement une trame d'interruption (`int_ctx_t`) au sommet de la pile noyau :

— Sélecteurs CS et SS configurés pour Ring 3
— `EFLAGS.IF = 1`
— Retour en mode utilisateur via `IRET`

1. **Pile noyau :** Utilisée uniquement lors d'une interruption ou d'un appel système. Le processeur y sauvegarde automatiquement le contexte.
2. **Pile utilisateur :** Utilisée pour l'exécution normale du code utilisateur (`while(1)`).

4 Résumé du Flux d'Exécution

1. Initialisation des PGD, du TSS et de l'IDT
2. Lancement de `T_1` en Ring 3 via une trame `IRET` construite manuellement
3. `T_1` incrémente le compteur partagé
4. IRQ0 détecte si l'interruption provient d'une tâche utilisateur ou du noyau, sauvegarde le contexte et bascule vers `T_2`
5. `T_2` déclenche une interruption `int 0x80` avec l'adresse du compteur dans `EAX`, et le noyau l'affiche
6. Le basculement périodique continue entre les tâches à chaque IRQ0