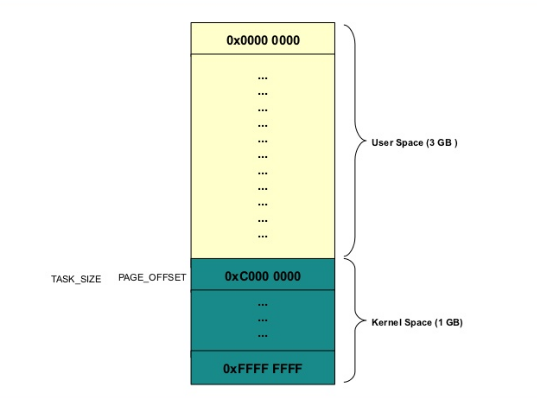
Contexte d’exécution :

* Process context :
  + Kernel mode s’exécutant au nom du processus (system calls ou exceptions handlers) dans le kernel space. Current macro est valide ici.
  + User mode s’exécutant en user space.
* Interrupt context (associé à aucun processus) en kernel mode



Piles :

Pile user space grande et dynamiquement agrandissable.

Pile kernel fixe et petite (peut supporter le max de fonctions imbriques que le kernel peut faire) : 4kb or 8kb.

Autres :

Le linux kernel est préemptif comparé aux autres Unix classiques.

Pas de distinction entre threads et process dans linux.

Les kernel threads n’ont pas d’espace d’adresses : mm pointe sur NULL. Ils sont schedulables et préemptables.

Mécanismes d’exclusion (protection des ressources partagées) :

* spinlocks
* sémaphores.
* Désactivation des IRQs (déconseillé)
* Noyau non préemptif : ne marche pas avec multiple CPU

Attention aux deadlocks !

C programming :

Printk() au lieu de printf() dans le kernel : pareil à une exception près :

printk(KERN\_ERR "this is an error!\n");

KERN\_ERR est un DEFINE qui permet d’ajouter un string à l’erreur.

* Static Inline : Ces fonctions sont intégrées directement dans le code des fonctions appelantes : Permet d’éviter des copies dans la pile. Utilisés dans les petites fonctions temps-critique.
* Asm volatile () : intégration de code ASM.
* Likely () – unlikely () : macros qui optimisent les branches conditionnelles.