

# Programmation Système

---

## Cours 3

### Processus

Brice Goglin

## Copyright

---

- Copyright © 2004 Brice Goglin – all rights reserved
- Ce support de cours est soumis aux droits d'auteur et n'est donc pas dans le domaine public. Sa reproduction est cependant autorisée sous réserve de respecter les conditions suivantes :
  - Si ce document est reproduit pour les besoins personnels du reproducteur, toute forme de reproduction (totale ou partielle) est autorisée à condition de citer l'auteur.
  - Si le document est reproduit dans le but d'être distribué à des tierces personnes, il devra être reproduit dans son intégralité sans aucune modification. Cette notice de copyright devra donc être présente. De plus, il ne devra pas être vendu.
  - Cependant, dans le seul cas d'un enseignement gratuit, une participation aux frais de reproduction pourra être demandée, mais elle ne pourra pas être supérieure au prix du papier et de l'encre composant le document.
  - Toute reproduction sortant du cadre précisé ci-dessus est interdite sans l'accord écrit préalable de l'auteur.

## Plan

---

- Concepts
- Description
- Exécution
- Création et terminaison
- Changement de contexte

## Plan (2/2)

---

- Exécution du noyau
- Processus et threads
- Parallélisme
- Détails de Linux

---

## Concepts

## Buts

- Ressources disponibles pour de multiples applications
- Toutes les applications semblent progresser simultanément
  - Les processeurs physiques les exécutent en alternance
- Le processeur et les périphériques sont utilisés efficacement

## Définitions du *Processus*

- Programme en exécution
- Instance d'un programme s'exécutant sur un ordinateur
- Entité pouvant être assignée et exécutée sur un processeur
- Unité d'activité caractérisée par l'exécution d'une suite d'instruction, un état courant et un ensemble de ressources système

---

## Description

## Caractérisation du processus

- A l'initialisation
  - Code du programme
  - Ensemble de données
- Durant l'exécution
  - Bloc de contrôle

## Bloc de contrôle

- Identifiant
- État
- Priorité
- *Program Counter*
- Adresses mémoire
- Données de contexte
- État des entrées/sorties
- Statistiques

## Etats des processus

- *Running*
- *Ready*
- *Blocked*
- *New*
- *Exit*
- *Suspend*

## Ressources système

- Tables mémoire
- Tables de périphériques
- Tables des fichiers
- Tables des processus

## Ressources des processus

- Localisation mémoire
  - Données utilisateur
  - Programme
  - Pile système
  - Bloc de contrôle
- Attributs
  - Identifiant
  - État du processeur
  - Données de contrôle

## Identification d'un processus

- Identifiants généralement numériques
  - Processus cible
  - Parent
  - Utilisateur

## Données de contrôle d'un processus

- État et ordonnancement
- Gestion de la mémoire
- Ressources attribuées et/ou utilisées
- Communications inter-processus
- Privilèges
- Données de structure

## Exécution

## État du processeur

- Registres utilisateur
- Registres de statut et contrôle
- Pointeurs de pile

## Modes d'exécution

- Plusieurs niveaux de privilèges dans le processeur
  - Accès aux registres de contrôle
  - Instructions d'entrées/sorties de bas niveau
  - Gestion mémoire
    - Tables de pages
  - Accès à certaines zones mémoire
    - Limitées par les structures pointées par certains registres

## Modes d'exécution (2/2)

- Le noyau peut tout faire (mode *réel*)
- L'utilisateur est limité (mode *protégé*)
  - Pas d'accès à la mémoire noyau
    - Non visible dans les tables de pages
  - Pas d'accès bas niveau au matériel
- Autre mode généralement pas/peu utilisés
- Interruption/exception provoque passage en mode privilégié

## Création et terminaison

## Création d'un processus

- Obtenir un identifiant disponible
- Allouer de l'espace pour le processus
- Initialiser le bloc de contrôle
- Initialiser les structures de données

## Création d'un processus (2/2)

- Duplication (fork)
  - Partages des structures principales du processus courant
    - Espace d'adressage, fichiers, signaux, ...
  - Création d'une copie des structures uniques
- 2 processus identiques pour l'utilisateur
  - Valeur de retour différente

## Transformation d'un processus

- Exécution (exec)
  - Remplacement du programme courant
  - Lancement d'un nouveau programme
  - Espace d'adressage complètement réinitialisé

## *Copy-on-Write*

- Partage des données le plus longtemps possible
- Création d'une copie à la première modification par un des processus
- Duplication rapide (vfork)
  - Copie du strict minimum
  - Père bloqué tant que le fils ne s'est pas transformé

## Terminaison d'un processus

- Relâchement des ressources
- Passage d'un code de retour au père
- Père peut attendre terminaison d'un fils (`wait`)
- Que faire des orphelins ?

## Changement de contexte

## Changement de contexte

- Nécessaire si
  - Blocage sur une I/O
  - Interruption
  - Erreur d'accès mémoire
- Recommandé pour
  - Équitabilité
  - Réactivité

## Changement de contexte (2/3)

- Vérifications régulières que le processus courant n'abuse pas du processeur
  - Interruptions d'horloge
  - Retour de traitement d'interruption
  - Retour d'appel système

## Changement de contexte (3/3)

- Sauvegarde du contexte du processeur (registres)
- Mise à jour du bloc de contrôle du processus 1
  - Changement de son état
- Choisir un nouveau processus 2
- Mise à jour du bloc de contrôle de 2
  - Changement de son état
- Mise à jour des structures d'accès à la mémoire
- Restauration du contexte du processeur

## Exécution du noyau

## Exécution du noyau

- Passage des processus en mode *réel* (noyau)
  - Traitement de leurs appels système
  - Traitement des interruptions
- Changement de contexte
  - Changement des privilèges
  - Utilisation d'une pile spéciale
  - Sauvegarde et restauration des registres

## Micronoyaux

- Appels système minimaux
  - Passage de messages
- Traitement réel du système d'exploitation en dehors du noyau
  - Dans les processus des serveurs dédiés



## Threads noyau

- Noyaux monolithiques ont besoin de traitement supplémentaire
  - Appels système imprévisibles et pas adaptés
  - *Handlers* d'interruption trop limités
- Utilisation des *threads* dédiés (`kernel_thread`)
  - Toujours en mode noyau
  - Traitent les travaux périodiques ou programmés
    - Traitement lourd post-interruption

## Processus et threads

## Processus et *threads*

- Plusieurs processus partageant des ressources
  - Utilisation mémoire partagée
  - Initialisation complexe
  - Pas forcément suffisant
- Plusieurs tâches dans un même processus
  - Recouvrement des blocages sur I/O

## Données partagées

- Piles et contextes d'exécution distincts
- Partage de
  - Espace d'adressage
  - Fichiers ouverts
  - Signaux
  - Identifiants de processus
- Clônage du thread courant (`clone`)

## Threads en espace utilisateur

- Ordonnancement des threads par l'application
  - Changement de contexte, création, ... rapides
- Noyau n'a aucune connaissance des threads
  - Un seul processus noyau, une seule file d'exécution
- Besoin d'assistance du système d'exploitation
  - Blocage de tous les threads si un thread bloque
    - *Scheduler Activations*
  - Temps processeur pas équitable

## Modèle 1-on-1

- Un processus système par thread
- Opérations quasi-normales
  - Création, terminaison, changement de contexte
  - Même principe que les processus
    - sauf l'espace d'adressage
  - Lent
- **pthread** sur Linux 2.4

## Modèle M-on-N

- N processus système pour M threads utilisateur
- Avantages des deux stratégies
  - Ordonnancement rapide en moyenne
  - Assistance du noyau pour les cas difficiles
- NPTL sur Linux 2.6
- Migration

## *LightWeight Processes*

- 1 ou plusieurs LWP par processus
- 1 thread noyau par LWP
- Threads utilisateurs liés à 1 ou plusieurs LWP
- Permet aux applications de contrôler leur degré de parallélisme
- **thr** sur Solaris

---

# Parallélisme

## Architectures parallèles

- SIMD (*Single Instruction Multiple Data stream*)
- MIMD (*Multiple Instruction Multiple Data stream*)
  - Mémoire partagée (fortement couplée)
    - Modèle maître/esclave
    - SMP (*Symmetric MultiProcessing*)
      - Homogène ou non
  - Mémoire distribuée (faiblement couplée)
    - Grappes (*Clusters*)

## SMP et NUMA, migration

- Exécution simultanée de plusieurs tâches
  - Plusieurs processus
  - Plusieurs threads d'un même processus
- Contraintes matérielles sur leur placement
  - Eviter le partage de structures
    - Défauts de lignes de cache
  - Accès NUMA aux données
- Migration tâches entre noeud
  - Avec leurs données

---

## Détails de Linux

## Hiérarchie

- **init**
  - Threads noyaux
  - Services
    - Daemons
  - Consoles texte
  - Gestionnaire de sessions graphiques
    - Applications graphiques
      - Terminaux

## Routines principales (2.4 et 2.6)

- **do\_fork(flags, stack, regs, ...)**
- **kernel\_thread(func, arg, flags)**
- **do\_exit(code)**
- **do\_execve(file, args, envs, ...)**
- **sys\_wait4(pid, &stat, options, ...)**
  - **wait\_task\_zombie(pid, &stat, ...)**

## Structures noyau

- File d'exécution (**struct task\_struct**)
- Espace d'adressage (**struct mm\_struct**)
  - **struct vm\_area\_struct** et leurs méthodes
  - **/proc/<pid>/maps**
- PID et table de hashage

## Descripteur

- **struct task\_struct**
  - 4 ou 8 ko
  - Ressources attribuées
  - PID et TGID
  - Liens vers père et fils
    - Vers **init** pour les orphelins
  - Pile noyau

## États des processus

- TASK\_RUNNING
- TASK\_INTERRUPTIBLE
- TASK\_UNINTERRUPTIBLE
- TASK\_STOPPED
- TASK\_ZOMBIE

## Ordonnancement et événements

- schedule
  - schedule\_timeout
- set\_current\_state
- wait\_queue\_t et wait\_queue\_head\_t
  - add/remove\_wait\_queue
- struct completion
  - wait\_for\_completion
  - complete/complete\_and\_exit