



Systèmes
d'Exploitation

Didier Verna
EPITA

Généralités

Scheduling

Opérations

Communication

Multithreading

Systèmes d'Exploitation

Gestion des processus

Didier Verna

didier@lrde.epita.fr
<http://www.lrde.epita.fr/~didier>



Table des matières

Systèmes
d'Exploitation

Didier Verna
EPITA

Généralités

Scheduling

Opérations

Communication

Multithreading

- 1 Généralités
- 2 Ordonnancement des processus
- 3 Opérations sur les processus
- 4 Communication entre processus
- 5 Multithreading



Programme vs. Processus

Systèmes
d'Exploitation

Didier Verna
EPITA

Généralités

Scheduling

Opérations

Communication

Multithreading

- Un **programme** est une suite d'instructions (objet statique)
- Un **processus** est un programme en exécution (objet dynamique : programme + contexte)
- **Contexte** : Espace d'adresses (exécutable, zone de données, pile), registres (PC, SP), d'autres informations.
- **Mode d'exécution**
 - ▶ Exécution simultanée de copies d'un même programme.
 - ▶ Exécution simultanée de la *même* copie d'un *même* programme (« réentrance »).
 - ▶ Cas intermédiaires : partager seulement le code, les données en lecture seule.



Bloc de contrôle d'un processus (PCB)

Systèmes
d'Exploitation

Didier Verna
EPITA

Généralités

Scheduling

Opérations

Communication

Multithreading

Structure de sauvegarde du contexte d'exécution d'un processus (UNIX : « U-Structure »)

■ Gestion des processus

- ▶ **État** : prêt, en exécution *etc.*
- ▶ **Registres** : PC, PSW *etc.*
- ▶ **Identité** : PID, parent, groupe *etc.*
- ▶ **Ordonnement** : priorité, comptabilité *etc.*

■ Gestion de la mémoire

- ▶ **Zones d'accès** : pointeurs de pile, tas, texte *etc.*

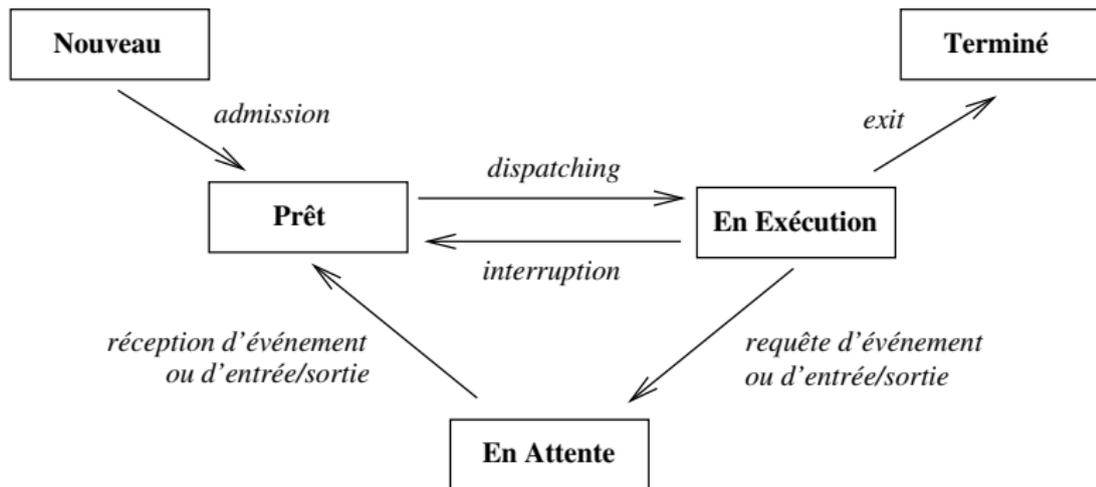
■ Gestion des fichiers

- ▶ **E/S** : fichiers ouverts *etc.*
- ▶ **Répertoires** : répertoire courant, racine *etc.*
- ▶ **Identité** : utilisateur, groupe *etc.*



États d'un processus

Un processeur ne peut exécuter qu'un seul processus à la fois





Principe de l'ordonnancement

Systèmes
d'Exploitation

Didier Verna
EPITA

Généralités

Scheduling

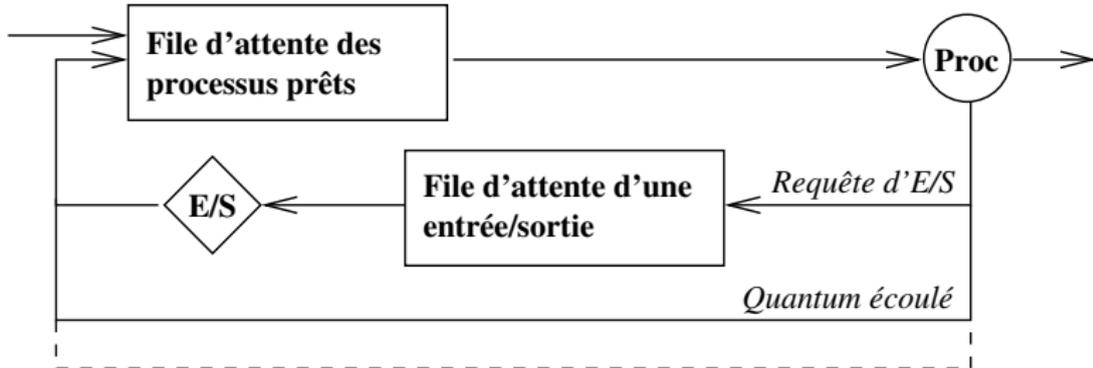
Opérations

Communication

Multithreading

Le système maintient :

- une **table de processus**,
- une **file d'attente** des processus prêts,
- des files d'attente de processus bloqués (ex. par périphérique).





Types d'ordonnanceurs

Systèmes
d'Exploitation

Didier Verna
EPITA

Généralités

Scheduling

Opérations

Communication

Multithreading

■ Long terme

Présélection des tâches spoolées sur disque et mise en mémoire (traitement par lots). S'exécute très peu souvent.

■ Court terme

Sélection des processus à exécuter parmi les processus prêts (en temps partagé). S'exécute très souvent.

■ Moyen terme

Gestion de la mémoire auxiliaire (en temps partagé).
« Swapping ».



Commutation de contexte

Systèmes
d'Exploitation

Didier Verna
EPITA

Généralités

Scheduling

Opérations

Communication

Multithreading

Échange des données concernant les processus (PCB) au moment du dispatching.

- Surcharge de travail pour le processeur.
- Vitesse liée à la complexité du système, la quantité de registres, l'existence d'instructions matérielles *etc.*
- Goulot d'étranglement pour les systèmes modernes (Cf. les threads).



Création de processus

Systemes
d'Exploitation

Didier Verna
EPITA

Généralités

Scheduling

Opérations

Communication

Multithreading

- **Nature** : processus interactifs, tâches de fond (background), démons *etc.* (UNIX : `ps`).
- **Causes** : Initialisation système, création par un autre processus, requête utilisateur *etc.* (UNIX : `fork`).
- **Types** : clonage du processus parent, nouvelle tâche (UNIX : `exec`).



Terminaison de processus

Systèmes
d'Exploitation

Didier Verna
EPITA

Généralités

Scheduling

Opérations

Communication

Multithreading

- **Terminaison (a)normale volontaire**

UNIX : `exit`.

- **Terminaison anormale involontaire**

Bugs : Division par 0, accès mémoire illégal *etc.*

Possibilité de paramétrer le comportement (signaux UNIX).

- **Terminaison normale involontaire**

UNIX : `kill`.



Processus Coopératif

Systèmes
d'Exploitation

Didier Verna
EPITA

Généralités

Scheduling

Opérations

Communication

Multithreading

■ Création

- ▶ **Efficacité** : répartition du travail en sous-tâches parallèles
- ▶ **Modularité** : répartition du travail en sous-tâches logiques

■ Terminaison

- ▶ **Synchronisation** : blocage et attente de la terminaison du fils (UNIX : `wait`).
- ▶ **Communication** : renvoi d'une valeur de terminaison.
- ▶ Exception : zombies.



■ Windows

Pas de hiérarchie. Le « process handle » peut circuler.

■ UNIX

- ▶ Arborescence sous `init`.
- ▶ Notion de « groupe de processus » (Unix : `setpgid`). Réception des signaux par groupe.

■ Terminaison en cascade

Un processus ne peut pas survivre à son parent.

- ▶ Exception : orphelins (*orphan*). Cf. `nohup`.



Communication inter-processus

aka IPC

Systèmes
d'Exploitation

Didier Verna
EPITA

Généralités

Scheduling

Opérations

Communication

Multithreading

■ Contexte

- ▶ Les processus coopératifs ont besoin de communiquer
- ▶ Solution 1 : mémoire partagée
- ▶ Solution 2 : envoi de message

■ Problématique

- ▶ Comment établir une liaison ?
- ▶ Combien de processus par liaison ?
- ▶ Combien de liaison par processus ?
- ▶ *etc.*



Communication directe

Nommage explicite du destinataire ou de l'émetteur

Systèmes
d'Exploitation

Didier Verna
EPITA

Généralités

Scheduling

Opérations

Communication

Multithreading

■ Caractéristiques

- ▶ Liaison automatique créée par le système
- ▶ Une seule paire de processus par liaison
- ▶ Liaison uni ou bidirectionnelle

■ Modes

▶ Liaison symétrique

```
send (P, msg); recv (P, msg);
```

▶ Liaison asymétrique

```
send (P, msg); recv (id, msg);
```

■ Problèmes

- ▶ Nécessité de connaître les noms des processus
- ▶ Problème en cas de changement de nom



Communication indirecte

Envoi et réception sur des *ports*

Systèmes
d'Exploitation

Didier Verna
EPITA

Généralités

Scheduling

Opérations

Communication

Multithreading

■ Caractéristiques

- ▶ Liaison établie à condition que les processus partagent un même port
- ▶ Plus de deux processus par liaison
- ▶ Liaison uni ou bidirectionnelle

■ Mode

- ▶ `send (A, msg);` `recv (A, msg);`

■ Problèmes

- ▶ Réceptions multiples sur un même port (se restreindre à deux processus, limiter les types d'accès en lecture ou écriture *etc.*)



Bufferisation

Caractérisation de la capacité d'une liaison

Systèmes
d'Exploitation

Didier Verna
EPITA

Généralités

Scheduling

Opérations

Communication

Multithreading

■ Modes

- ▶ **Capacité nulle** : Aucun message en attente. Nécessité d'une synchronisation pour échanger des messages (« rendez-vous » / « hand shaking »).
- ▶ **Capacité limitée** : L'émetteur peut être retardé si le buffer est plein.
- ▶ **Capacité infinie** : L'émetteur n'est jamais retardé.

■ Synchronisation

- ▶ Pour des capacités non nulles, la communication est *asynchrone* (un émetteur ne sait pas si son message est reçu)
- ▶ Synchronisation possible par « acquittement » (acknowledgement)
- ▶ Synchronisation possible en rendant `send` bloquant jusqu'à l'arrivée d'un accusé de réception



Cas pathologiques

Systèmes
d'Exploitation

Didier Verna
EPITA

Généralités

Scheduling

Opérations

Communication

Multithreading

- **Terminaison** : P attend (ou envoie) un message de (ou vers) un processus terminé. Le système doit le terminer ou lui notifier le problème.
- **Messages perdus** :
 - ▶ Vérification sous la responsabilité de l'émetteur
 - ▶ Renvoi (ou notification à l'émetteur) sous la responsabilité du système
 - ▶ Détection par temporisation
 - ▶ Pas de détection
- **Messages bruités** : (bitflip) idem
Détection par contrôle de parité



Qu'est-ce que le multithreading ?

Systèmes
d'Exploitation

Didier Verna
EPITA

Généralités

Scheduling

Opérations

Communication

Multithreading

■ Problèmes

- ▶ Fournir un parallélisme intra-processus
- ▶ Amoindrir le coût de la commutation de contexte

■ Solution

- ▶ **Thread / LWP** (Light Weight Processus) : état, registres (dont PC) et pile. Partage des autres ressources.
- ▶ **Tâche / HWP** (High Weight Processus) : ensemble de threads. Processus traditionnel = tâche à un seul thread.

■ Fonctionnement

- ▶ **Identique à celui des processus** : création, terminaison, notion d'état *etc.*
- ▶ **Problèmes nouveaux** : concurrence d'accès aux ressources partagées (synchronisation) *etc.*



■ Principe

- ▶ Implémentés par une bibliothèque en espace utilisateur
- ▶ Une table de threads par processus

■ Avantages

- ▶ Utilisable au dessus d'un système non multithreadé
- ▶ Commutation de contexte très rapide (pas de « kernel trapping »)
- ▶ Algorithmes d'ordonnancement personnalisables

■ Inconvénients

- ▶ **Besoin d'appels systèmes non bloquants**
Wrappers utilisant `select`
- ▶ **Threads monopolisant le CPU**
`yield` + requête d'alarme
- ▶ L'intérêt des threads réside justement là où il y a souvent blocage. Mais le noyau est là pour ça !



Threads noyau

Systèmes
d'Exploitation

Didier Verna
EPITA

Généralités

Scheduling

Opérations

Communication

Multithreading

■ Principe

- ▶ Une table de threads en plus de la table de processus
- ▶ Tout appel bloquant est implémenté par appel système

■ Avantages

- ▶ Facilité de conception des applications
- ▶ Pas de nécessité de procédures supplémentaires non bloquantes

■ Inconvénients

- ▶ Coût de gestion des threads (recyclage)
- ▶ Coût des appels bloquant (interruptions)



Approches hybrides (ex. Solaris 2)

Systèmes
d'Exploitation

Didier Verna
EPITA

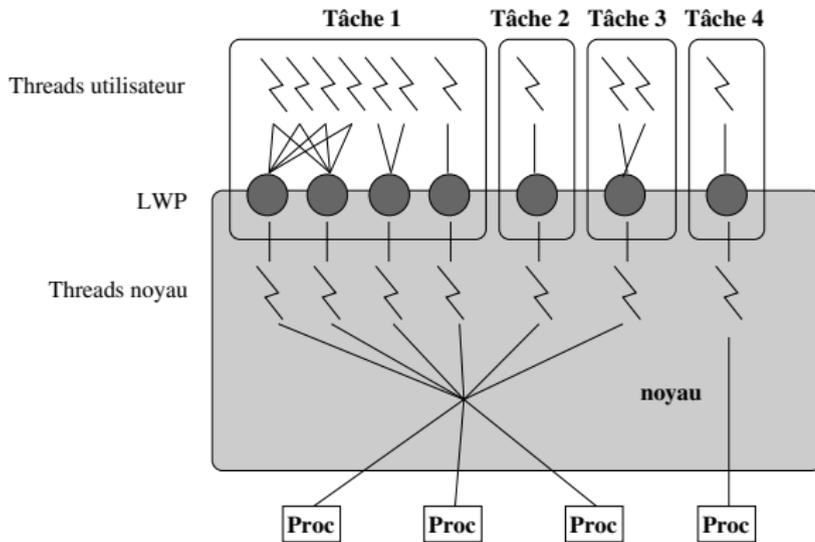
Généralités

Scheduling

Opérations

Communication

Multithreading



■ Scheduler activation

- ▶ « upcall » vers l'ordonnanceur logiciel (userland) quand un thread se (dé)bloque
- ▶ Rupture avec le modèle en couches



Transition vers le multithreading

Des problèmes nouveaux

Systèmes
d'Exploitation

Didier Verna
EPITA

Généralités

Scheduling

Opérations

Communication

Multithreading

- **Variables globales** (programmes ou système)
 - ▶ Les interdire
 - ▶ Fournir des valeurs locales par thread (ex. `errno`)
- **Signaux** : qui les récupère ?
- **Réentrance des fonctions** (bibliothèques systèmes)
 - ▶ mettre des drapeaux d'exclusion mutuelle (attention au parallélisme),
 - ▶ réécrire les bibliothèques.