400 Communication inter-processus

INF3173

Principes des systèmes d'exploitation

Jean Privat

Université du Québec à Montréal

Hiver 2021

Introduction

Chaque processus

- Est autonome
- Vit isolé dans son propre espace mémoire

Trop restrictif

- Besoin de collaboration, communication et de coopération
- Processus différents opèrent ensemble vers un même objectif

Communication interprocessus

- IPC (*interprocess communication*)
- Mécanismes du système d'exploitation
- Parfois offerts par bibliothèques et démons (espace utilisateur)

Formes de coopération

Coopération interne

- Application conçue à la base multithreads et multiprocessus
- Exemples: navigateurs modernes

$\rightarrow$ Objectifs : performance, asynchronisme, isolation, etc.

Coopération protocolaire

- Communiquer avec des applications qui **respectent** un **protocole**
- Exemples: la plupart des applications réseaux

$\rightarrow$ Attention à être robuste

Coopération des données

- Application manipule des **données**, produite (ou non) par d'autres applications
- Exemples : ouvrir/enregistrer, tubes shell, etc.

Philosophie Unix

1978

- Make each program do one thing well. To do a new job, build afresh rather than complicate old programs by adding new "features".
- Expect the output of every program to become the input to another, as yet unknown, program. Don't clutter output with extraneous information. Avoid stringently columnar or binary input formats. Don't insist on interactive input.
- Design and build software, even operating systems, to be tried early, ideally within weeks. Don't hesitate to throw away the clumsy parts and rebuild them.
- Use tools in preference to unskilled help to lighten a programming task, even if you have to detour to build the tools and expect to throw some of them out after you've finished using them.

Philosophie Unix

1994

- Write programs that do one thing and do it well.
- Write programs to work together.
- Write programs to handle text streams, because that is a universal interface.

Source Raymond, E. (2003). The Art of Unix Programming.

Mise en garde

Vaste domaine

- La communication entre applications informatiques est un domaine bien large qui sort du cadre de ce cours

Voir

- Téléinformatique (INF3271)
- Programmation concurrente et parallèle (INF5171)
- Programmation Web avancée (INF5190)
- Réseaux sans fil et applications mobiles (TEL4165)
- Programmation parallèle haute performance (INF7235)
- Machines virtuelles (INF7741)
- Systèmes répartis (MGL7126)

$\rightarrow$ seulement une petite partie de l'iceberg

Deux modèles de communication

Données échangées entre processus (*message passing*)

- Chacun les traite à sa façon
- Un seul processus a les données à la fois

Données partagées entre processus (*shared memory*)

- Données communes
- Accès et modifications non exclusives

$\rightarrow$ On n'y reviendra plus tard

Questions: à quel modèle correspond

- Tube shell « `|` »
- 2 threads et une structure de données dans le tas
- Fichiers dans `~/Documents`
- Base de données

Exemples d'IPC chez POSIX

- Fichiers (déjà vu)
- Terminaison de processus: `exit(2)` et `wait(2)` (déjà vu)
- Signaux: `signal(7)` ; y compris `kill(1)` (410 Signaux)
- Tubes (*pipe*): `pipe(7)` ; y compris « | » et tubes nommés `fifo(7)`. (420 Tubes)
- Sockets UNIX: `unix(7)`. (430 Sockets Unix)
- Sockets classiques (TCP, UDP, etc.): `socket(7)`, `tcp(7)`, `udp(7)` (on y reviendra pas)
- Mémoire partagée: `shm_overview(7)` (on y reviendra)

→ Plus quelques autres

→ Plus tous ceux spécifiques à d'autres systèmes

→ Plus tous ceux "applicatifs" basés sur ces primitives systèmes RPC (*remote procedure call*), bus logiciels, etc.

Système de fichiers

Les fichiers ne sont pas exclusifs à un processus

- Les processus peuvent utiliser les fichiers pour communiquer entre eux
- L'accès et la protection sont connus

Exemples

- Fichiers de données et autres documents (explicites à l'utilisateur)
- Fichier temporaire pour passer des données (compilation, etc.)
- Fichiers spéciaux pour initier un autre type de communication (tubes nommés, etc.)
- *Spool* (impression, cron job, etc.)
- Fichier projeté en mémoire (`mmap(2)`), on y reviendra.

Socket réseau

Communication distante

- Objectif primaire : faire communiquer des applications sur des ordinateurs distincts
- Peut **aussi** être utilisé pour des applications sur une même machine
- Majoritairement pour du clients-serveur

Avantages

- Le SE peut optimiser l'efficacité du traitement
- Exemple : client/serveur X (`X(7)`)

Bus logiciels

Support de communication de haut niveau

- Majoritairement applicatif (en espace utilisateur)
- Permet de faire communiquer des applications via des objets partagés et des envois de messages
- Souvent pour le réseau (ex. CORBA)

Exemple : D-BUS

- Utilisé dans les bureaux graphiques Unix modernes
- Un démon + clients exposent&utilisent services&objets
- Bibliothèques utilisables par les programmes
- Démon et bibliothèque utilisent des primitives systèmes pour faire le travail de communication, de synchronisation et de protection
- https://www.freedesktop.org/wiki/Software/dbus/

Gestion de la communication

Il faut un **protocole** de communication

- Moyens de communication (données, structures de données) que les processus peuvent échanger et accéder
- Primitives d'accès et de protection
- Mécanismes de synchronisation
- Attention aux problèmes standards
  interblocage, famine, etc.

Qui s'en charge ? Le système d'exploitation

- Fournit des appels système pour IPC
- Plus ou moins riches et complexes
- Règles spécifiques au cas par cas

$\rightarrow$ Impose un protocole mais garantit certaines propriétés

Qui s'en charge ? Le programmeur

- Utilise les primitives systèmes pour implémenter ses propres protocoles et applications
- Prend en compte les limites et caractéristiques des IPC utilisées
- Est libre d'implémenter tout ce qu'il veut par-dessus

$\rightarrow$ les IPC systèmes sont des **outils** pour bâtir sa solution de coopération

Qui s'en charge ? Bibliothèques et langages

- Abstraient les IPC système
- Offrent des fonctionnalités et protocoles clé en main
- Exemple: requête https en JavaScript, JEE, D-Bus, etc.

$\rightarrow$ Il est normal de les utiliser quand c'est adapté

Sauf en INF3173 car pour apprendre, on fait tout à la main