Programmation système

M. Davy MOUSSAVOU- Consultant
Linux, Certifié Linux International

Communication interprocessus

Objectifs du cours

Après cette session les apprenants doivent:

- Comprendre les caractéristiques d'un processus UNIX;
- Connaître le cycle de vie d'un processus
- Savoir utiliser les appels systèmes de gestion de processus

Agenda

- Séquence 1: Concepts théoriques
- Séquence 2: Tubes sans nom
- Séquence 2: Tubes de communication nommés

Séquence 1 : Concepts théoriques

Séquence 1 : Concepts théoriques

❑ Objectifs

- Comprendre le rôle d'un processus dans un système d'exploitation;
- Organisation des processus;

Séquence 1 : Concepts théoriques

❑ Introduction

Le système d'exploitation gère les processus. Malgré ce chaque processus:

- Est une enté autonome et indépendante;
- Vit isolé dans son propre espace mémoire;
- Les processus peuvent se trouver en conflit pour l'accès à certaines ressources communes

Solution

- Mécanisme de communication interprocessus
- Mise en œuvre de mécanisme de synchronisation

Séquence 1 : Concepts théoriques

❑ Introduction

Définition

Les communications interprocessus regroupent l'ensemble des mécanismes permettant la communication et la synchronisation entre processus concurrents (ou distants)

Séquence 1 : Concepts théoriques

❑ Processus d'exécution d'un processus

Les processus peuvent communiquer à travers les :

- **Variables et fichiers communs;**
- **Signaux;**
- **Messages;**
- **Tubes de communication.**

Séquence 1 : Concepts théoriques

❑ Concept de tube de communication

Les tubes de communication ou pipes permettent a deux ou plusieurs processus d'échanger des informations, sur un même ordinateur.
On distingue deux types de tubes :

- ❑ Les tubes sans nom;
- ❑ Les tubes nommés,

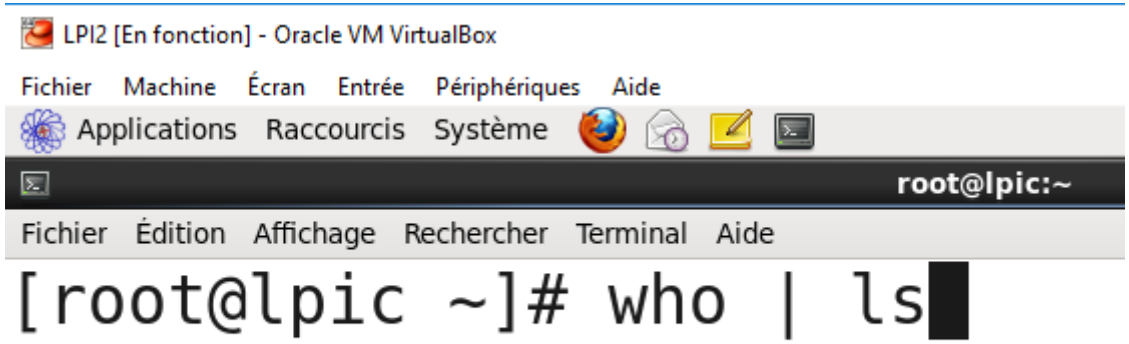
Séquence 2 : Tube sans nom

Séquence 2 : Tube sans nom

- **Les tubes sans nom sont des liaisons unidirectionnelles de communication.**

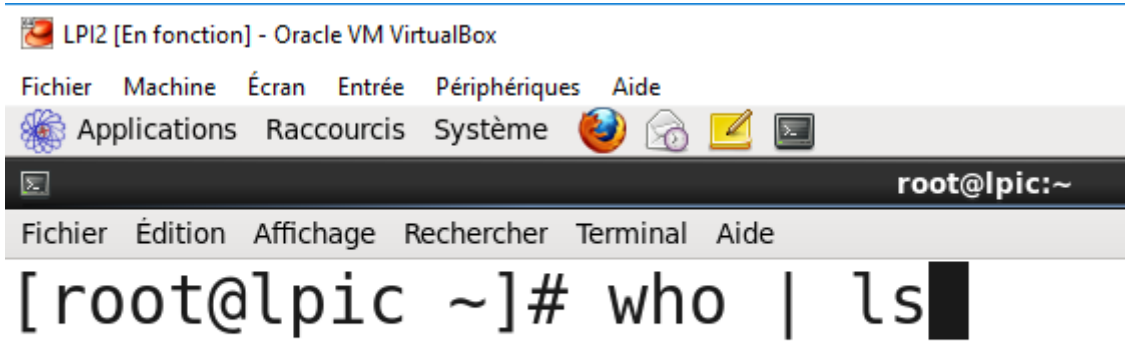
- **Les tubes sans nom du shell sont crées par l'operateur binaire | qui dirige la sortie standard d'un processus vers l'entrée standard d'un autre processus.**

Séquence 2 : Tube sans nom



Cree deux processus qui s'exécutent en parallèle et qui sont reliés par un tube de communication pipe.

Séquence 2 : Tube sans nom

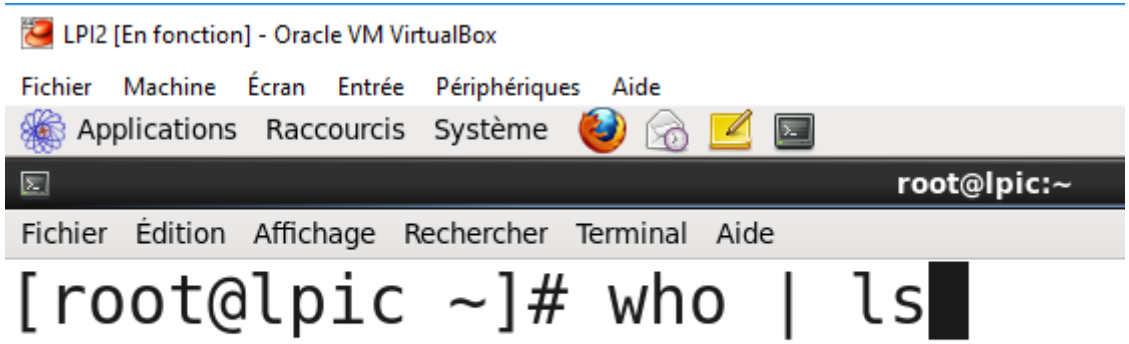


Étape 1: Le premier processus réalise la commande who

Étape 2: Le second processus exécute la commande wc -l

Étape 3: Les résultats récupérés sur la sortie standard du premier processus sont diriges vers l'entrée standard du deuxième processus via le tube de communication qui les relie.

Séquence 2 : Tube sans nom



Étape 1: Le premier processus réalise la commande who

Étape 2: Le second processus exécute la commande wc -l

Étape 3: Les résultats récupérés sur la sortie standard du premier processus sont diriges vers l'entrée standard du deuxième processus via le tube de communication qui les relie.

Étape 4: Les deux processus s'exécutent en parallèle, les sorties du premier processus sont stockées sur le tube de communication

Séquence 2 : Tube sans nom

Appel système de création de tube

❑ Création d'un tube

NOM

pipe, pipe2 - Créer un tube.

SYNOPSIS

```
#include <unistd.h>

int pipe(int pipefd[2]);

#define _GNU_SOURCE
#include <unistd.h>

int pipe2(int pipefd[2], int flags);
```

DESCRIPTION

pipe() crée un tube, un canal unidirectionnel de données qui peut être utilisé pour la communication entre processus. Le tableau pipefd est utilisé pour renvoyé deux descripteurs de fichier faisant référence aux extrémités du tube. pipefd[0] fait référence à l'extrémité de lecture du tube. pipefd[1] fait référence à l'extrémité d'écriture du tube. Les données écrites sur l'extrémité d'écriture du tube sont mises en mémoire tampon par le noyau jusqu'à ce qu'elles soient lues sur l'extrémité de lecture du tube. Pour plus de détails, voir **pipe**(7).

Séquence 2 : Tube sans nom

Appel système de création de tube

```
NOM
       pipe, pipe2 - Créer un tube.

SYNOPSIS
       #include <unistd.h>

       int pipe(int pipefd[2]);
```

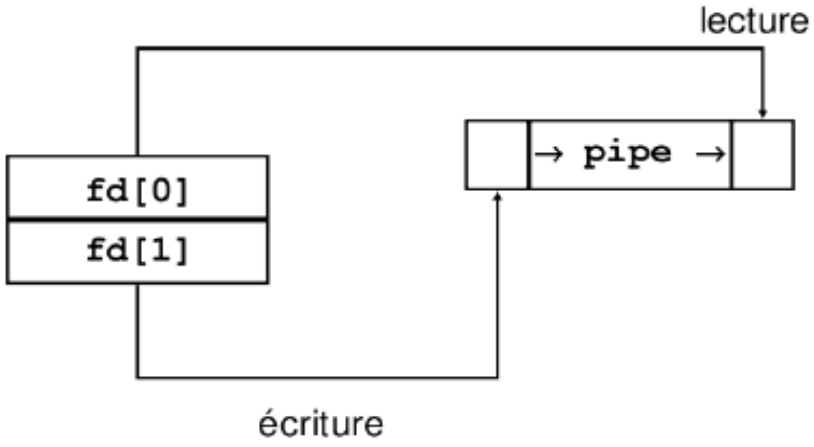
REMARQUE

Seul le processus créateur du tube et ses descendants (ses fils) peuvent accéder au tube. Si le système ne peut pas créer de tube par manque d'espace, l'appel système pipe() retourne la valeur -1, sinon il retourne la valeur 0.

Séquence 2 : Tube sans nom

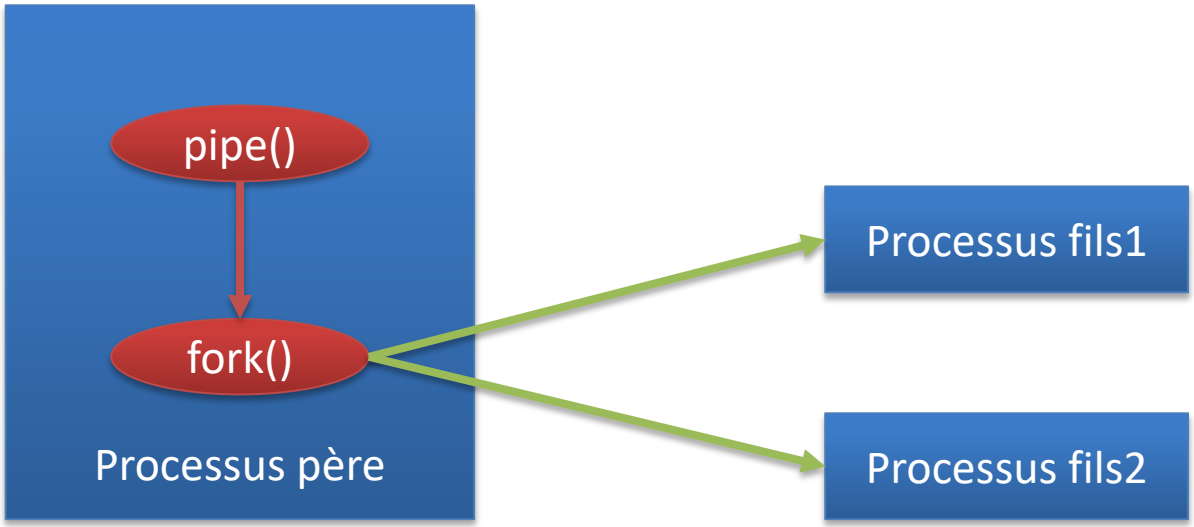
La pertinence des tubes

Les tubes sans nom sont, en général, utilisés pour la communication entre un processus père et ses processus fils.



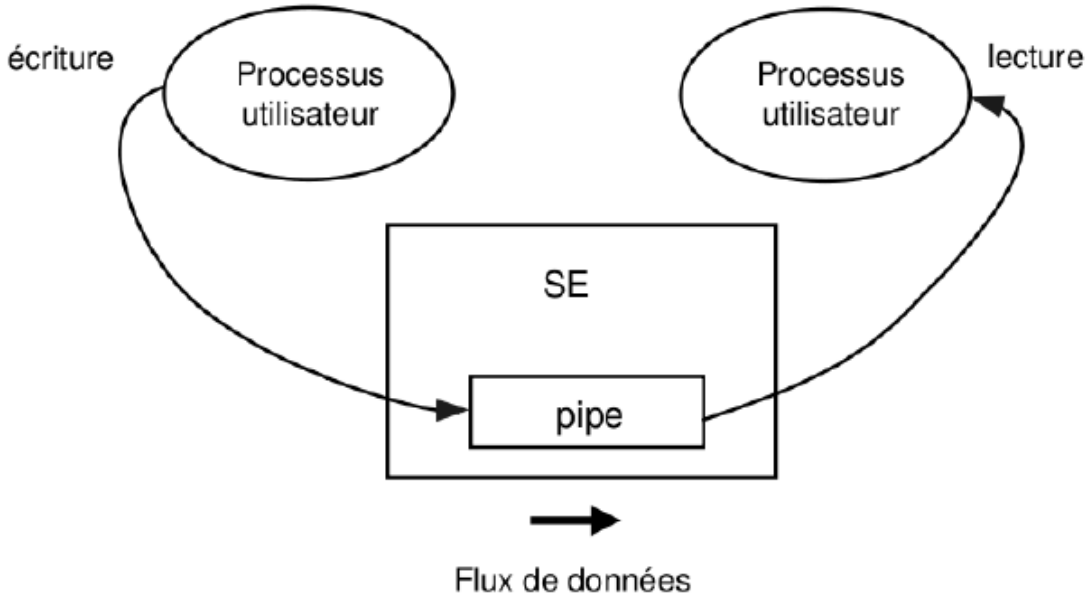
Séquence 2 : Tube sans nom

Mécanisme du pipe



Séquence 2 : Tube sans nom

Mécanisme du pipe



Séquence 2 : Tube sans nom

Exemple

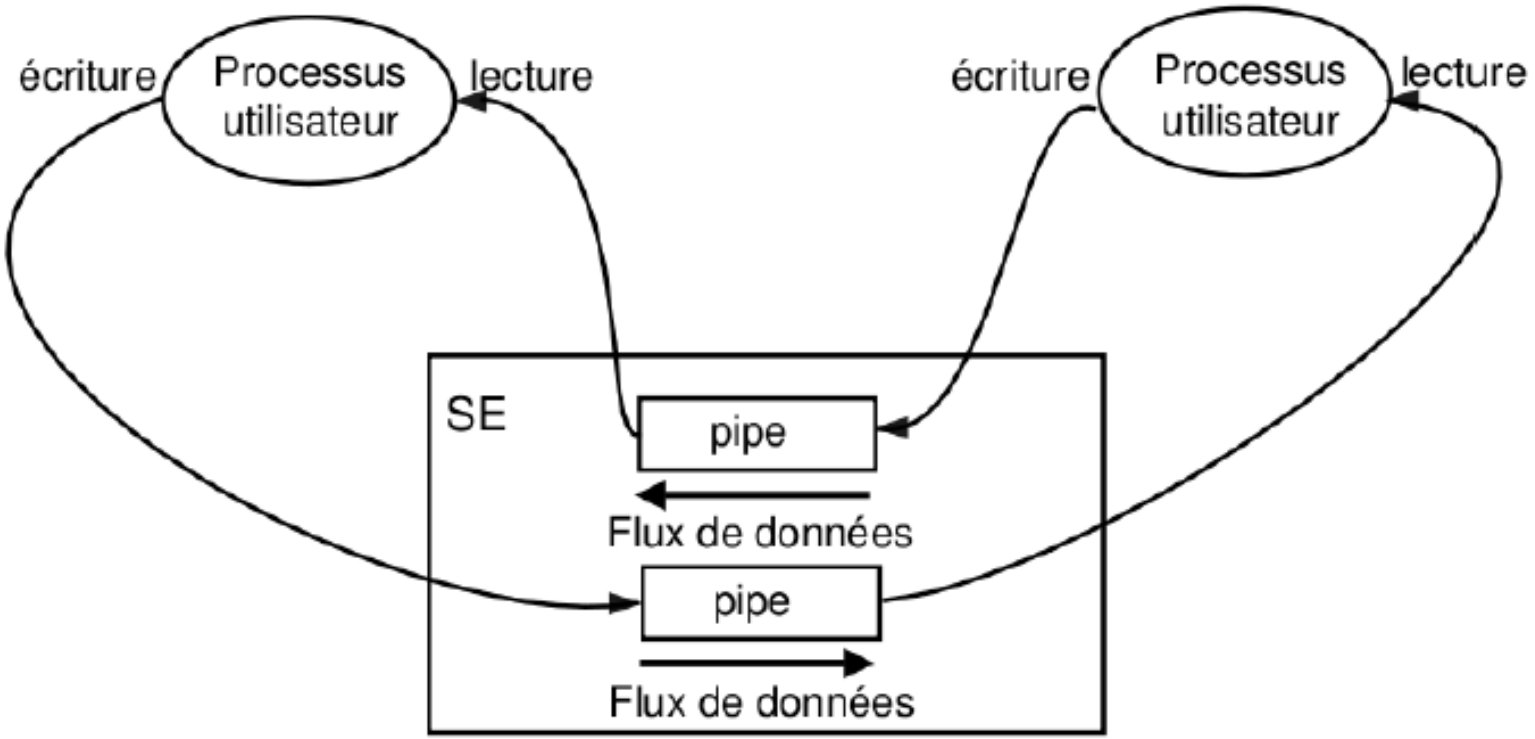
```c
#include <sys/types.h> // types
#include <unistd.h>    // fork, pipe, read, write, close
#include <stdio.h>
#define R 0
#define W 1

int main()
{
int fd[2];
char message[100];   // pour recuperer un message
int nboctets;
char *phrase = "message envoye au pere par le fils";

pipe(fd);              // creation d'un tube sans nom
if (fork() ==0)  // creation d'un processus fils
{
    // Le fils ferme le descripteur non utilise de lecture
    close(fd[R]);
    // depot dans le tube du message
    write(fd[W],phrase, strlen(phrase)+1) ;
    // fermeture du descripteur d'ecriture
    close (fd[W]) ;
}
else
{
    // Le pere ferme le descripteur non utilise d'ecriture
    close(fd[W]) ;
    // extraction du message du tube
    nboctets = read (fd[R], message,100) ;
    printf ("Lecture %d octets : %s\n", nboctets, message) ;
    // fermeture du descripteur de lecture
    close (fd[R]) ;
}
return 0;
}
```

Séquence 2 : Tube sans nom

Communication bidirectionnelle



Séquence 3 : Tubes de communication nommés

Séquence 2 : Tubes de communication nommés

Principe

Définition

Les tubes de communication nommés fonctionnent comme des files de type FIFO (First In First Out).

Séquence 2 : Tubes de communication nommés

Avantages

- Chacun dispose d’un nom dans la table des fichiers
- Ils sont considérés comme des fichiers spéciaux
- Ils peuvent être utilises par des processus indépendants, a condition qu'ils s'exécutent sur une même machine.

Atelier