Communication entre processus Tube

Deux processus peuvent communiquer et se synchroniser à travers un tube (pipe |)

Les deux processus ont besoin d'une relation de parenté pour avoir les descripteurs

Efficace au niveau du système

Tube nommé

Type spécial de fichier

- Persistant dans le SGF (Système de Gestion de Fichier)
- Plus de parenté nécessaire
- Droits avec chmod
- Tube non bidirectionnels

Tube nommé

Type spécial de fichier

- Persistant dans le SGF (Système de Gestion de Fichier)
- Plus de parenté nécessaire
- Droits avec chmod
- Tube non bidirectionnels

Socket

Littéralement ”prises”

Introduit dans BSD 4.2 en 1981

- Entre processus locaux : sockets dans le ”domaine Unix”
- via un nom de socket dans le SGF (comme pour les tubes nommés).

  (netstat --unix)

Socket

Une socket est une ”adresse unique” : pour que 2 processus dialoguent,

- chacun crée sa propre socket,
- puis s’adresse à l’autre socket.

Communication bidirectionnelle

- Un dialogue = 2 sockets

Socket Domain

Les domaines définis dans <sys/socket.h> pour Linux :

Nom Description

- AF_UNIX Communications locales
- AF_INET Internet IPv4
- AF_INET6 Internet IPv6
- AF_IPX Novell IPX
- AF_NETLINK Communication avec le noyau

Socket Domain

- AF_X25 X25 (minitel, etc)
- AF_AX25 Radio amateur AX.25
- AF_ATMPVC Raw ATM Permanent Virtual Conn.
- AF_APPLETALK Appletalk
- AF_PACKET Interface bas niveau

Socket Type

- SOCK_RAW datagrammes, bas niveau
- SOCK_DGRAM datagrammes, non connecté, non fiable
- SOCK_STREAM flux, connecté, fiabilité

java.lang.System

Relation avec le Système d'exploitation (héritage des entrées/sorties)

- static InputStream in – entrée (descripteur 0)
- static PrintStream out – sortie (descripteur 1)
- static PrintStream err – sortie erreur (descripteur 2)

java.lang.System

Les entrées/sorties peuvent être modifiées à travers les méthodes :

- static void setIn(InputStream in)
- static void setOut(PrintStream out)
- static void setErr(PrintStream err)

java.lang.Throwable

Throwable permet la gestion des erreurs dans Java.

(java.lang.Error, java.lang.Exception)

dans un bloc try catch

exception.printStackTrace() affiche la trace des erreurs sur la sortie des erreurs

Exception

```
    // zone avec peut-être une erreur
    try
    {
       int x = 50/0;
    } catch (Exception e)// Gestion erreur
    {
       // affiche la pile des erreurs
       e.printStackTrace();
      // affiche l'exception
       System.out.println(e);
    }
 }
}
```

java.lang.System

public static void exit(int status)

La méthode normale pour terminer un programme java.

Dans le monde Unix un programme se termine par 0 si tout va bien.

Pour avoir la valeur de retour de la dernière commande en bash

> echo $?

java.lang.Runtime

Une application java possède une seule instance Runtime. On obtient cette instance par la méthode :

static Runtime getRuntime()

java.lang.Runtime

On peut obtenir des informations sur l'environnement à travers les méthodes

- int availableProcessors()
- long freeMemory()
- long maxMemory()

(mémoire max de la jvm paramètre -Xmx)

- long totalMemory()

(mémoire totale utilisée)

java.lang.Runtime

On peut lancer un processus avec la commande :

- Process exec(String command)
- Process exec(String[] cmdarray)
- Process exec(String[] cmdarray, String[] envp)
- Process exec(String[] cmdarray, String[] envp, File dir)
- ...

java.lang.System

Les méthodes suivantes permettent d'obtenir des informations du système :

- static String getenv(String name) – variable d’environnement
- static Map<String,String> getenv() – variables d’evironnements

java.lang.System

- static long nanoTime() timer en nanosecondes
- static long currentTimeMillis() le temps courant en milliseconde
- static Properties getProperties() donne les propriétés du système

java.util.Properties

La classe Properties permet de gérer un ensemble de propriétés du Système sous la forme de couple nom-valeur.
Constructeur :

- Properties() – création d'une liste vide de propriété
- Properties(Properties defaults) – création d'une liste avec les Propriétés defaults

Méthodes :

- String getProperty(String key) – permet d'obtenir la valeur de la clef
- void list (PrintStream out) – affiche sur out les propriétés

java.lang.Process

La classe Process permet de créer un sous-processus et de le gérer

Méthodes :

- void destroy() – Tue le process
- int exitValue() – donne la valeur de retour
- InputStream getErrorStream() - flux des erreurs
- InputStream getInputStream() - flux de sortie du process
- OutputStream getOutputStream() - flux d'entrée
- int waitFor() - permet d'attendre la fin du sous -processus

java.io.BufferedReader

La classe BufferedReader permet lire des données textes en utilisant un buffer

Méthodes :

- String readLine() - permet de lire une ligne de caractères
- int read() - permet de lire un caractère
- int read(char[] cbuf, int off, int len) – permet de lire un tableau de caractères
- ...

java.io.InputStreamReader

La classe InputStreamReader permet de faire le lien entre des octets et des caractères.

Méthodes :

- String getEncoding() - permet d'avoir le fomat d'encodage
- ...

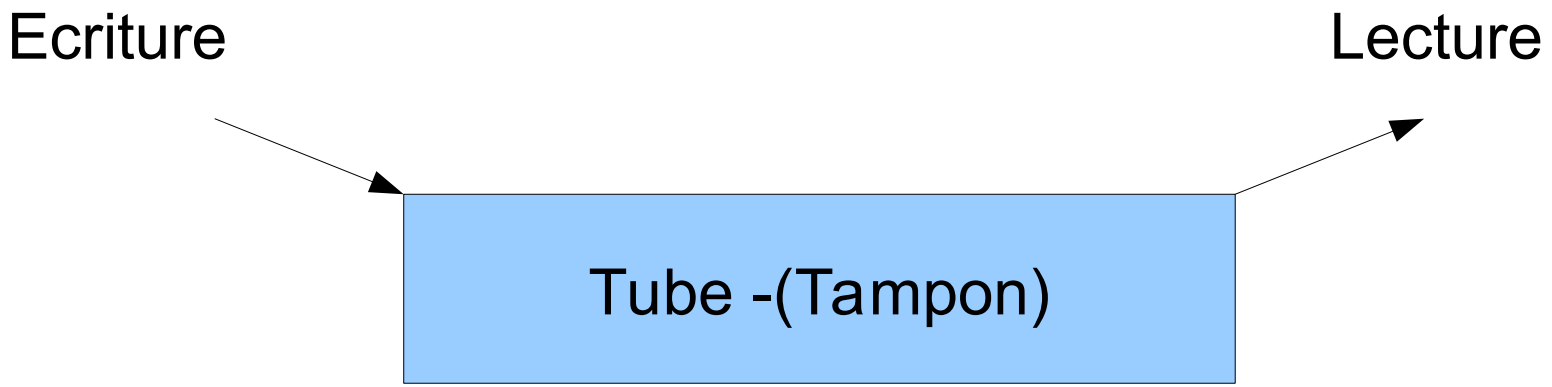
Exemple

```
BufferedReader in = new
BufferedReader(new
InputStreamReader(System.in));
```

Tube

Le tube permet de réaliser une synchronisation entre une tâche qui écrit des données et une tâche qui lit

(le système permet une utilisation efficace des ressources)



java.io.PipedWriter

Constructeurs

PipedWriter() : Création d'un pipe non connecté

PipedWriter(PipedReader snk) : Connexion d'un pipe writer au PipedReader snk

Méthodes

close() : fermeture

connect(PipedReader snk) : connexion

flush() : vidage du cache

write(char[] cbuf, int off, int len) : écriture

write(int c) : écriture d'un caractère

java.io.PipedReader

Constructeurs

PipedReader() : Création d'un pipe non connecté

PipedReader(PipedWriter snk) : Connexion d'un pipe reader au PipedWriter snk

Méthodes

close() : fermeture

connect(PipedWriter snk) : connexion

flush() : vidage du cache

int read(): lecture d'un caractère

int read(char[] cbuf, int off, int len) :lecture de caractères

boolean ready() indique si on peut lire

java.lang.Runnable

Interface pour les Threads

Méthode :

public void run()

définit une méthode qui sera exécutée par exemple par un Thread

java.lang.Thread

Une classe qui reprend Runnable

Méthode :
void start()
lance l'exécution de la méthode run du Thread
static void sleep(long millis)
fait dormir le Thread courant pendant millis