

Compléments pour le Projet Tubes Nommés

Olivier Dalle
Université de Nice – Sophia
Antipolis
olivier.dalle@unice.fr

Plan de la présentation

- **Les tubes nommés**
 - ◆ Utilisation en shell
 - ◆ Utilisation en python
- **Problèmes d'interblocage**
 - ◆ Solution non symétrique
 - ◆ Solution symétrique
 - ◆ Programmation SPMD
- **Multiplexage**
- **Atomicité**
 - ◆ Principe
 - ◆ Règles de conception

Les Tubes Nommés

■ Tubes, mais accessibles comme des fichiers

Création depuis le shell

```
~$ mkfifo toto
~$ ls -al toto
prw-r--r-- 1 odalle staff 0 11 avr 11:38 toto
```

Création depuis programme python

```
import os
os.mkfifo("toto")
```

■ Intérêt?

- ◆ Connection possible entre processus quelconques

- ◆ Par exemple entre le shell de Pierre et le shell de Paul



Utilisation des Tubes Nommés

■ S'ouvrent comme des fichiers...

- ◆ Ex: `fd = os.open("toto",os.O_RDONLY)`

■ ...se lisent et s'écrivent comme des fichiers...

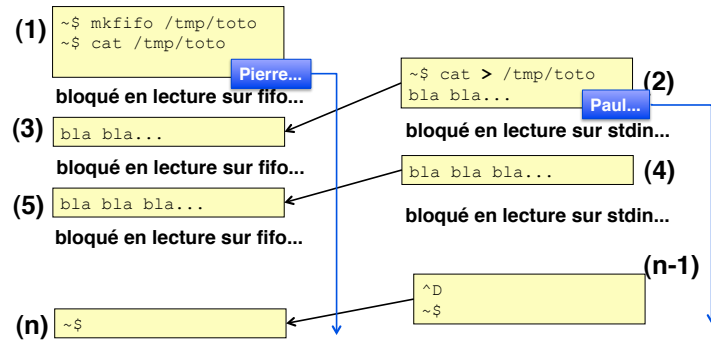
- ◆ Ex: `os.read(fd,"bla bla ...")`

■ ... mais fonctionnent comme des tubes (!!)

- ◆ suppression du contenu au fur et à mesure des lectures
- ◆ lectures/écritures bloquantes
 - ◆ si tube plein ou vide
- ◆ Fin de fichier atteinte lors de la fermeture

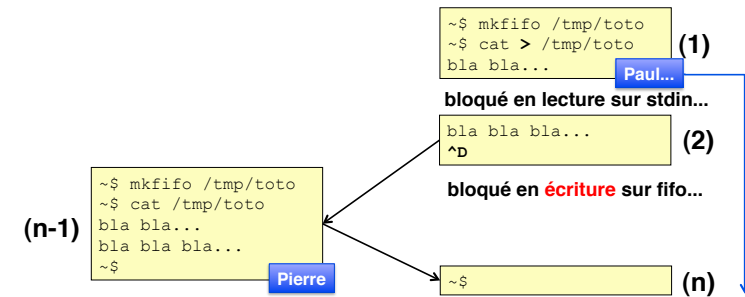
Exemples d'utilisation des tubes (nommés)

■ Bloquage en lecture (en shell)



Exemples d'utilisation des tubes (nommés)

■ Bloquage en écriture (en shell)



Exemple d'utilisation des tubes nommés

```
import sys,os

def creation_fifo(name):
    try: os.mkfifo(name)
    except OSError, (errno, strerror):
        print >>sys.stderr,\
            "Erreur creation fifo %s: %s(%d)" \
            %(name,strerror,errno)

if __name__ == "__main__":
    if len(sys.argv) < 2:
        print >>sys.stderr,\
            "Usage: %s nom_fifo" %(sys.argv[0])
        sys.exit(0)
    print "On essaie de creer (erreur non fatale)"
    creation_fifo(sys.argv[1])
    print "On essaie d'ouvrir:"
    fd = os.open(sys.argv[1],os.O_WRONLY)
    print "On essaie d'ecrire dans fifo:"
    os.write(fd,"blabla\n")
    print "On essaie d'ecrire encore dans fifo:"
    os.write(fd,"blabla2\n")
    print "On ferme la fifo"
    os.close(fd)
    sys.exit(0)
```

open_fifo.py

```
~$ python open_fifo.py toto
On essaie de creer (erreur non fatale)
Erreur creation fifo toto: File exists(17)
On essaie d'ouvrir:
```

shell(1)...

(1)

(2)

(3)

(4)

(6)

(5)

bloqué...

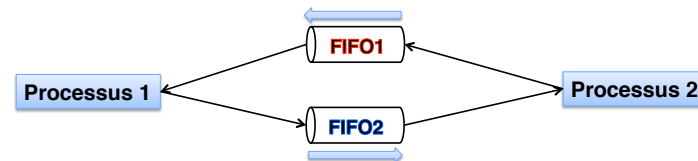
```
~$ cat toto
blabla
blabla2
~$
```

shell(2)

```
On essaie d'ecrire dans fifo:
On essaie d'ecrire encore dans fifo:
On ferme la fifo
~$
```

...shell(1)

Problème d'Interblocage à l'Ouverture

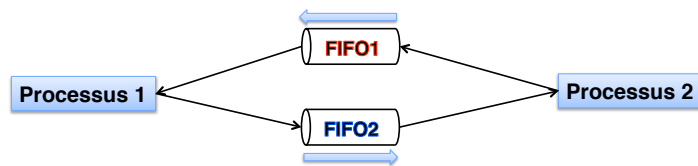


```
import sys,os
...
fdr = os.open("FIFO1",os.O_RDONLY)
fdw = os.open("FIFO2",os.O_WRONLY)
```

Verrou mortel
(interblocage)

```
import sys,os
...
fdr = os.open("FIFO2",os.O_RDONLY)
fdw = os.open("FIFO1",os.O_WRONLY)
~$
```

Interblocage : solution 1



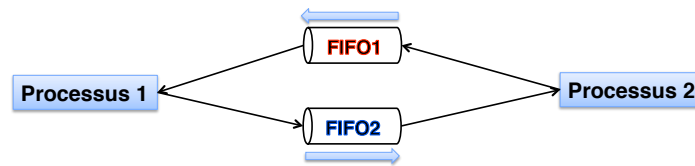
```
import sys,os
...
fdr = os.open("FIFO1",os.O_RDONLY)
fdw = os.open("FIFO2",os.O_WRONLY)
```

```
import sys,os
...
fdw = os.open("FIFO1",os.O_WRONLY)
fdr = os.open("FIFO2",os.O_RDONLY)
```

■ Inconvénient : programme différent (asymétrique)

- ◆ Ok jusqu'à 2 processus, ingérable au-delà...

Interblocage : solution 2



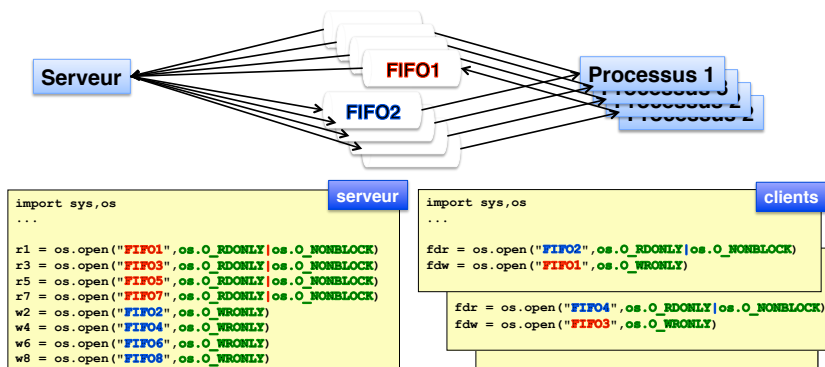
```
import sys,os
...
fdr = os.open("FIFO1",os.O_RDONLY|os.O_NONBLOCK)
fdw = os.open("FIFO2",os.O_WRONLY)
```

```
import sys,os
...
fdr = os.open("FIFO2",os.O_RDONLY|os.O_NONBLOCK)
fdw = os.open("FIFO1",os.O_WRONLY)
```

■ Programme quasi-identique (symétrique)

- ◆ Mais données différentes
- ◆ paradigme SPMD (Single Program Multiple DATA)

Exemple de solution client/serveur symétrique type SPMD

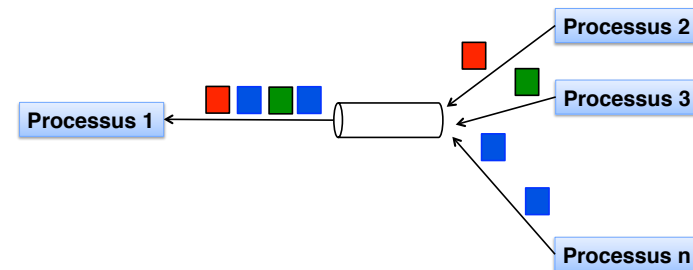


© 2010-2011, O. Dalle

10-11

Multiplexage

■ Partage d'un même medium par plusieurs entités



© 2010-2011, O. Dalle

10-12

Avantages et Inconvénients du Multiplexage

■ Avantage

- ◆ On économise des tuyaux!

■ Inconvénient

- ◆ On ne sait pas à qui on parle...



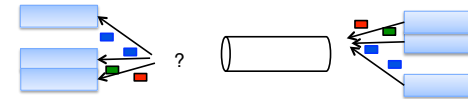
- ◆ ... mais parfois ce n'est pas nécessaire!

❖ Si nécessaire, définir un **protocole**

Semi-multiplex ou Full-multiplex?

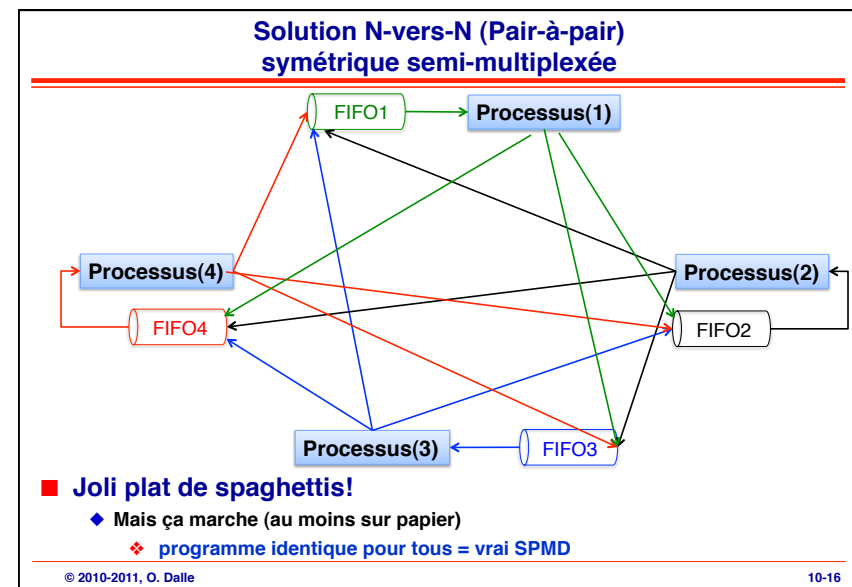
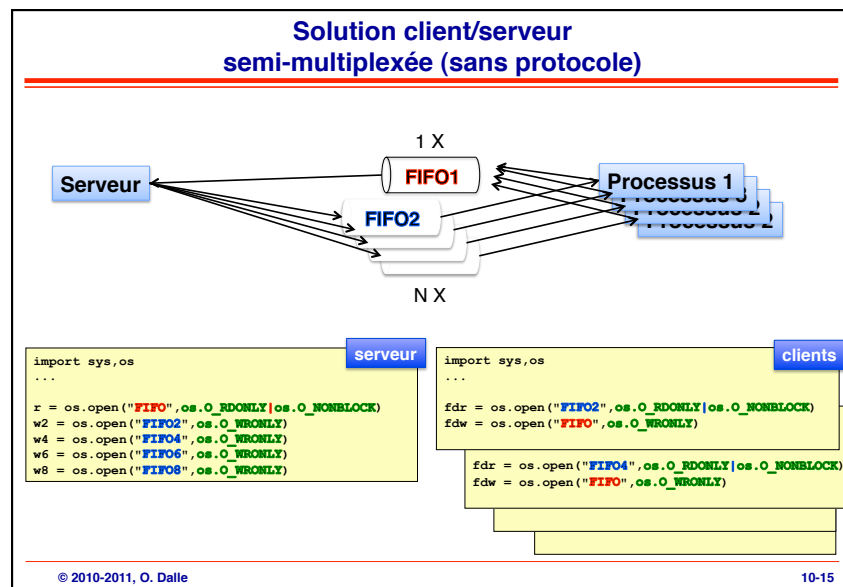
■ Full-multiplex

- ◆ Chaque tube peut être lu par et écrit par plusieurs entités
- ◆ Problème : la lecture consomme (détruit) le message
 - ❖ 1 seul lecteur reçoit... mais qui?



■ Semi-multiplex (Ex: 1-lecteur, n-écrivains)

- ◆ Un tube peut être **écrit par plusieurs**, mais **lu par un seul**
- ◆ Le (unique) lecteur peut consommer le message sans se soucier des autres
- ◆ OK si on n'a pas besoin de savoir d'où ça vient.



Identification de la source?

■ Concevoir un **protocole**

◆ Phase de présentation

- ❖ Je suis toto, on peut me joindre sur fifo "/tmp/toto"
- ❖ Je suis titi, on peut me joindre sur ...

◆ Phase de dialogue

- ❖ toto: bla bla ...
- ❖ titi: bla bla ...

◆ Autres ... ?

■ Définir des types et structures de message

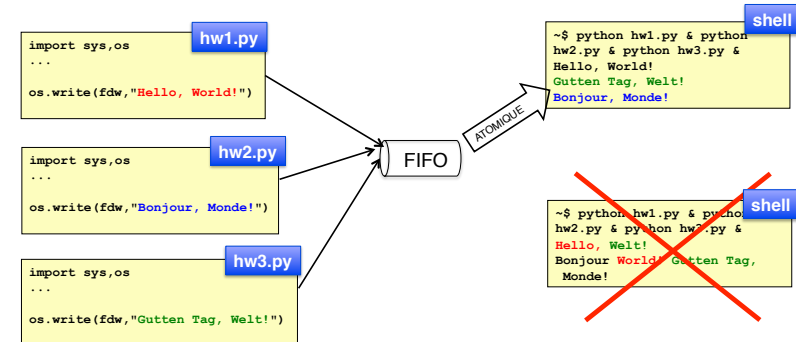
◆ Ex:

- ❖ Presentation = 'P' + nom + '\n' + chemin + '\n' ?
- ❖ Dialogue = 'D' + ...

Ecritures Atomiques

■ Ecriture Atomique = mélange impossible

◆ Contenu d'une opération indivisible



Granularité de l'Atomicité

■ Granularité = 1 écriture

◆ mélange possible entre plusieurs écritures!

