

IF 110 – Les flux



Joachim Bruneau-Queyreix

ENSEIRB-MATMECA

Bordeaux-INP

jbruneauqueyreix@enseirb-matmeca.fr

D'après le cours d'introduction aux systèmes d'exploitation de Télécom SudParis



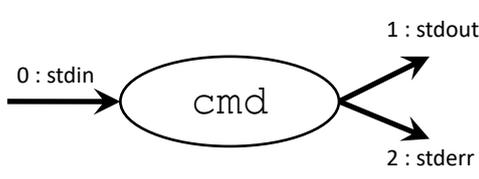
Notion de flux

- ❑ Pour accéder aux données d'un fichier (écran, clavier, fichier ordinaire...), le système d'exploitation définit une notion de flux

- ❑ Un flux est défini par :
 - Un fichier
 - Une fonction de lecture : permet d'extraire des données du flux
 - Une fonction d'écriture : permet d'ajouter des données au flux
 - Une tête de lecture : position dans le fichier pour les lectures/écritures

- ❑ Un flux est représenté par un numéro

Par défaut un processus possède 3 flux



```

graph LR
    stdin[0 : stdin] --> cmd((cmd))
    cmd --> stdout[1 : stdout]
    cmd --> stderr[2 : stderr]
  
```

- `stdin (0)` : *standard input*
 - » canal de lecture, par défaut clavier du terminal (celui de `read`)
- `stdout (1)` : *standard output*
 - » canal de sortie, par défaut écran du terminal (celui d'écho)
- `stderr (2)` : *standard error*
 - » canal de sortie pour les erreurs, par défaut écran du terminal
 - » pour le moment, on n'utilise pas ce canal

3

Les flux par défaut du terminal

```

$ read a b
Salut tout le monde!!!
$ echo $a
Salut
$ echo $b
tout le monde!!!
$
  
```

← Lecture à partir du clavier
← Le terminal affiche les caractères saisis au clavier
← Tête de lecture dans le flux de sortie (écran associé au terminal)

4



Plan

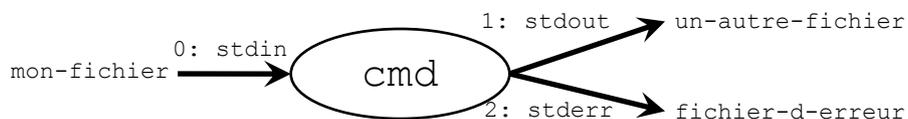
1. Redirections simples
2. Redirections avancées
3. Les tubes
4. Fichiers associés aux périphériques

5



Redirections simples

- ❑ Toute commande peut être lancée en redirigeant les flux du processus vers un fichier



- ❑ À ce moment :
 - `echo` écrit (en ajout) à la fin de `un-autre-fichier`
 - `read` lit ligne à ligne à partir de `mon-fichier`

6



3 paramètres pour rediriger un flux

- ❑ Un **fichier** associé au nouveau flux
- ❑ Le **numéro** du flux à rediriger
- ❑ Un **mode** d'ouverture

	Lecture	Écriture	Tête de lecture	Remarque
<	Oui	Non	Au début	
>	Non	Oui	Au début	Ancien contenu effacé
>>	Non	Oui	À la fin	
<>	Oui	Oui	Au début	

Remarque : lors d'une ouverture en écriture, le fichier est toujours créé si il n'existait pas

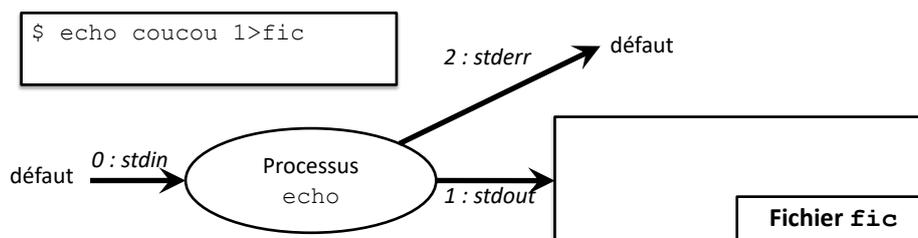
7



Redirection de flux

- ❑ Lance la commande `cmd` **après** avoir ouvert le flux numéro `n` associé au fichier `fic` avec le mode idoine

`cmd n[<,>,>>,<>]fic`



8

Redirection de flux

- Lance la commande `cmd` **après** avoir ouvert le flux numéro `n` associé au fichier `fic` avec le mode idoine

$$\text{cmd } n[\langle, \rangle, \gg, \langle \rangle] \text{fic}$$

9

Redirection de flux

- Si le numéro de flux n'est pas indiqué
 - Utilise 1 (stdout) si en écriture
 - Utilise 0 (stdin) si en lecture ou lecture/écriture
 1 (stdout) est implicite

10



Redirection de flux

```
$ echo Coucou vous >fic  
$
```

Exécute `echo` en redirigeant
la sortie standard
(redirection en écriture)

Coucou vous

Fichier `fic`

11



Redirection de flux

```
$ echo Coucou vous >fic  
$ read x y <fic  
$
```

Exécute `read` en redirigeant
l'entrée standard
(redirection en lecture)

Coucou vous

Fichier `fic`

12



Redirection de flux

```
$ echo Coucou vous >fic  
$ read x y <fic  
$ echo $x  
Coucou  
$
```

Coucou vous

Fichier **fic**

13



Redirection de flux

```
$ echo Coucou vous >fic  
$ read x y <fic  
$ echo $x  
Coucou  
$ echo $y  
vous  
$
```

Coucou vous

Fichier **fic**

14



Redirection de flux et regroupement

- ❑ Toute expression bash peut être redirigée

```
$ ( read x; echo $x; read y; echo $y; ) <fic
```

Étape 1 : associe le flux 0 (stdin) à fic en lecture



15

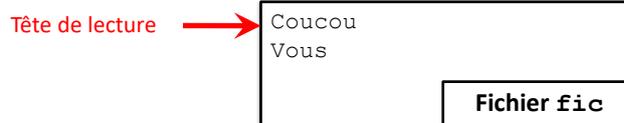


Redirection de flux et regroupement

- ❑ Toute expression bash peut être redirigée

```
$ ( read x; echo $x; read y; echo $y; ) <fic
```

Étape 2 : lance l'exécution du regroupement en redigeant son entrée



16

Redirection de flux et regroupement

☐ Toute expression bash peut être redirigée

```
$ ( read x; echo $x; read y; echo $y; ) <fic
```

Étape 3 : exécute read ⇒ déplace la tête de lecture

Tête de lecture

Valeur lue dans x

Coucou
Vous

17

Redirection de flux et regroupement

☐ Toute expression bash peut être redirigée

```
$ ( read x; echo $x; read y; echo $y; ) <fic  
Coucou
```

Étape 4 : affiche la variable x

Tête de lecture

Coucou
Vous

Fichier fic

18

Redirection de flux et regroupement

☐ Toute expression bash peut être redirigée

```
$ ( read x; echo $x; read y; echo $y; ) <fic  
Coucou
```

Étape 5 : exécute read ⇒ déplace la tête de lecture

Valeur lue dans y

```
Coucou  
Vous
```

19

Redirection de flux et regroupement

☐ Toute expression bash peut être redirigée

```
$ ( read x; echo $x; read y; echo $y; ) <fic  
Coucou  
Vous
```

Étape 6 : affiche la variable y

Tête de lecture →

```
Coucou  
Vous
```

Fichier fic

20



Redirection de flux et regroupement

- ❑ Toute expression `bash` peut être redirigée

```
$ for x in 1 2 3; do  
>   echo $x  
> done >fic
```

```
1  
2  
3
```

Fichier `fic`

21



Lecture d'un flux ligne à ligne

- ❑ `read` lit une ligne d'un flux et avance la tête de lecture
⇒ `read` dans une boucle permet de lire un fichier ligne à ligne
- ❑ Il faut aussi détecter la fin d'un flux pour terminer la boucle
 - Fin de flux indiquée par un code EOF (end-of-file)
 - » Généré sur le terminal lorsque l'utilisateur saisie `Control+d`
 - » Généré automatiquement lorsque la tête de lecture atteint la fin d'un fichier
 - Lecture de EOF indiquée dans le code de retour de `read`
 - » `read` retourne faux si lecture renvoie EOF
 - » `read` retourne vrai sinon

22



Lecture d'un flux ligne à ligne

\$

```
#!/bin/bash  
while read line; do  
  echo ":: $line"  
done <fic
```

Fichier **script.sh**

```
Avec ses quatre dromadaires  
Don Pedro d'Alfaroubeira
```

Fichier **fic**

23



Lecture d'un flux ligne à ligne

\$./script.sh

```
#!/bin/bash  
while read line; do  
  echo ":: $line"  
done <fic
```

Fichier **script.sh**

Ouverture du flux associé
à **fic** avant d'exécuter la boucle

Tête de lecture →

```
Avec ses quatre dromadaires  
Don Pedro d'Alfaroubeira
```

Fichier **fic**

24



Lecture d'un flux ligne à ligne

```
$ ./script.sh
```

read renvoie vrai car pas fin de fichier

Tête de lecture →

```
#!/bin/bash  
while read line; do  
  echo "$line"  
done <fic
```

Fichier script.sh

```
Avec ses quatre dromadaires  
Don Pedro d'Alfaroubeira
```

Fichier fic

25



Lecture d'un flux ligne à ligne

```
$ ./script.sh  
:: Avec ses quatre dromadaires
```

Tête de lecture →

```
#!/bin/bash  
while read line; do  
  echo ":: $line"  
done <fic
```

Fichier script.sh

```
Avec ses quatre dromadaires  
Don Pedro d'Alfaroubeira
```

Fichier fic

26

Lecture d'un flux ligne à ligne

```
$ ./script.sh
:: Avec ses quatre dromadaires
```

read renvoie vrai car pas fin de fichier

```
#!/bin/bash
while read line; do
  echo "$line"
done <fic
```

Fichier script.sh

Tête de lecture →

```
Avec ses quatre dromadaires
Don Pedro d'Alfaroubeira
```

Fichier fic

27

Lecture d'un flux ligne à ligne

```
$ ./script.sh
:: Avec ses quatre dromadaires
:: Don Pedro d'Alfaroubeira
```

```
#!/bin/bash
while read line; do
  echo "$line"
done <fic
```

Fichier script.sh

Tête de lecture →

```
Avec ses quatre dromadaires
Don Pedro d'Alfaroubeira
```

Fichier fic

28



Lecture d'un flux ligne à ligne

```
$ ./script.sh  
:: Avec ses quatre dromadaires  
:: Don Pedro d'Alfaroubeira
```

read renvoie faux car fin de fichier
(valeur de line indéfinie)

Tête de lecture →

```
#!/bin/bash  
while read line; do  
  echo ":: $line"  
done <fic
```

Fichier script.sh

```
Avec ses quatre dromadaires  
Don Pedro d'Alfaroubeira
```

Fichier fic

29



Lecture d'un flux ligne à ligne

```
$ ./script.sh  
:: Avec ses quatre dromadaires  
:: Don Pedro d'Alfaroubeira  
$
```

Termine la boucle, ferme le flux puis
termine le processus

```
#!/bin/bash  
while read line; do  
  echo ":: $line"  
done <fic
```

Fichier script.sh

```
Avec ses quatre dromadaires  
Don Pedro d'Alfaroubeira
```

Fichier fic

30



Plan

1. Redirections simples
2. Redirections avancées
3. Les tubes
4. Fichiers associés aux périphériques

31



Redirections avancées

- ❑ La commande `exec` redirige les flux du processus courant (au lieu de lancer un nouveau processus)

```
exec n[<,>,>>,<>]fic
```

⇒ ouvre le flux `n` associé à `fic` avec le mode idoine

- ❑ Et une redirection peut se faire vers n'importe quel flux ouvert

```
cmd n[<,>,<>]&k
```

⇒ lance `cmd` en redirigeant le flux `n` vers le flux `k`

32



Redirections avancées

```
#!/bin/bash  
echo coucou >fic
```



```
#!/bin/bash  
exec 3>fic  
echo coucou >&3
```

33



Intérêt des redirections avancées

- ❑ Permet de lire et écrire dans plusieurs fichiers

```
#!/bin/bash  
exec 3> redoublants # flux 3 pour les redoublants  
  
while read etudiant note; do  
  if [ $note -lt 10 ]; then  
    echo $etudiant >&3  
  fi  
  echo "Etudiant $etudiant a eu $note/20"  
done <fichier-de-notes
```

Les redoublants sont ajoutés à la fin du fichier `redoublants`

34



Plan

1. Redirections simples
2. Redirections avancées
3. Les tubes
4. Fichiers associés aux périphériques

35



Les tubes

- ❑ On peut rediriger la sortie d'une commande dans l'entrée d'une autre

```
cmd1 | cmd2
```

 - Exécute `cmd1` et `cmd2` en parallèle
 - La sortie de `cmd1` est redirigée dans l'entrée de `cmd2`
- ❑ gros grain, comportement proche de
 - `cmd1 >temp-file`
 - `cmd2 <temp-file`
 - `rm temp-file`

(la mise en œuvre est différente et repose sur des concepts vus aux prochains cours)

36



1. Redirections simples
2. Redirections avancées
3. Les tubes
4. Fichiers associés aux périphériques

37



Redirection et fichiers de périphérique

- ❑ Le système définit un fichier par périphérique
 - Périphérique matériel connecté à l'unité centrale
 - » /dev/sda : premier disque dur
 - » /dev/bus/usb : périphériques usb
 - » /dev/hpet : horloge haute fréquence de la machine
 - » /dev/input/mice : souris
 - Périphérique logiciel appelé pseudo-périphérique
 - » /dev/sda1 : première partition d'un disque dur
 - » /dev/null : lecture renvoie EOF et écriture *jette* les données
 - » /dev/tty : terminal
 - » /dev/urandom : générateur de nombres aléatoire

38



Redirection et fichiers de périphérique

- ❑ On peut utiliser les redirections avec les périphériques
 - `read a </dev/urandom`
⇒ lit une ligne à partir du générateur de nombres aléatoires

 - `echo "J'ai cassé mon disque" > /dev/sda`
⇒ écrit "J'ai cassé mon disque" au début du disque dur
(à ne pas faire, vous allez vraiment casser votre disque dur!)

39



Concepts clés

- ❑ Un flux est la réunion de
 - Un numéro représentant le flux
 - Un mode d'ouverture (lecture/écriture, ajout/écrasement)
 - Un fichier associé au flux

- ❑ Tout flux peut être redirigé avec
 - `cmd n[<, >, >>, <>] fic` où `fic` est un fichier
(la commande `exec` ouvre le flux dans le processus courant)
 - `cmd n[<, >, <>] &k` où `k` est un numéro de flux ouvert

- ❑ Le tube (`|`) permet de chaîner une sortie et une entrée
`cmd1 | cmd2`

40