

Les flux

CSC 3102

Introduction aux systèmes d'exploitation

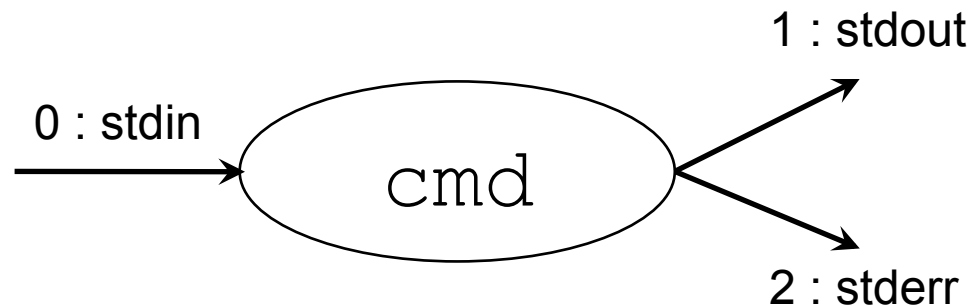
Gaël Thomas



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



- `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'echo)
- `stderr` (2) : *standard error*
 - canal de sortie pour les erreurs, par défaut écran du terminal
 - pour le moment, on n'utilise pas ce canal

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

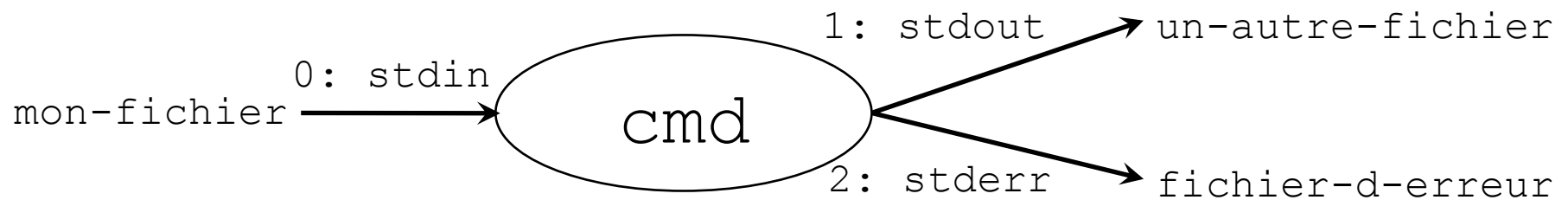
Tête de lecture dans le flux de sortie (écran associé au terminal)



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

Redirections simples

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



- À ce moment :
 - `echo` écrit dans `un-autre-fichier`
 - `read` lit à partir de `mon-fichier`

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	

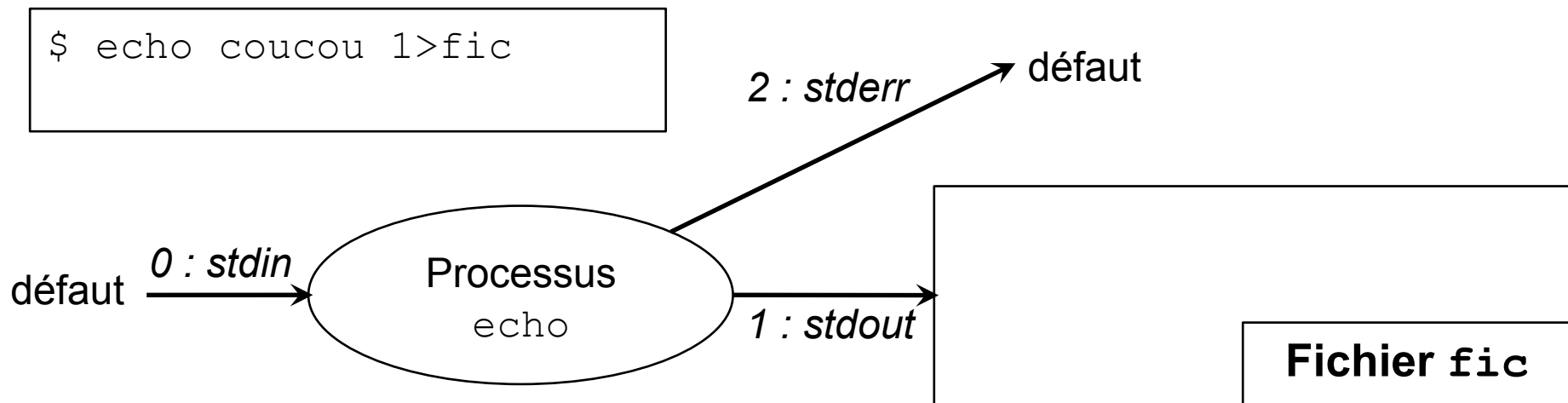
Remarque : lors d'une ouverture en écriture, le fichier est toujours créé si il n'existe pas

Redirection de flux

- Lancement d'une commande en redirigeant un flux

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

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

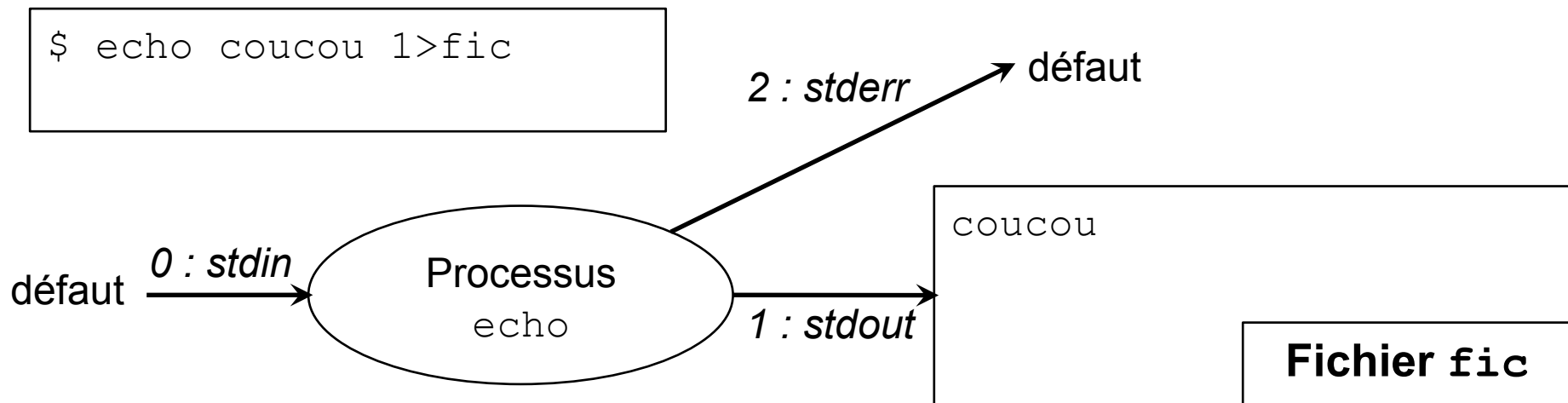


Redirection de flux

- Lancement d'une commande en redirigeant un flux

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

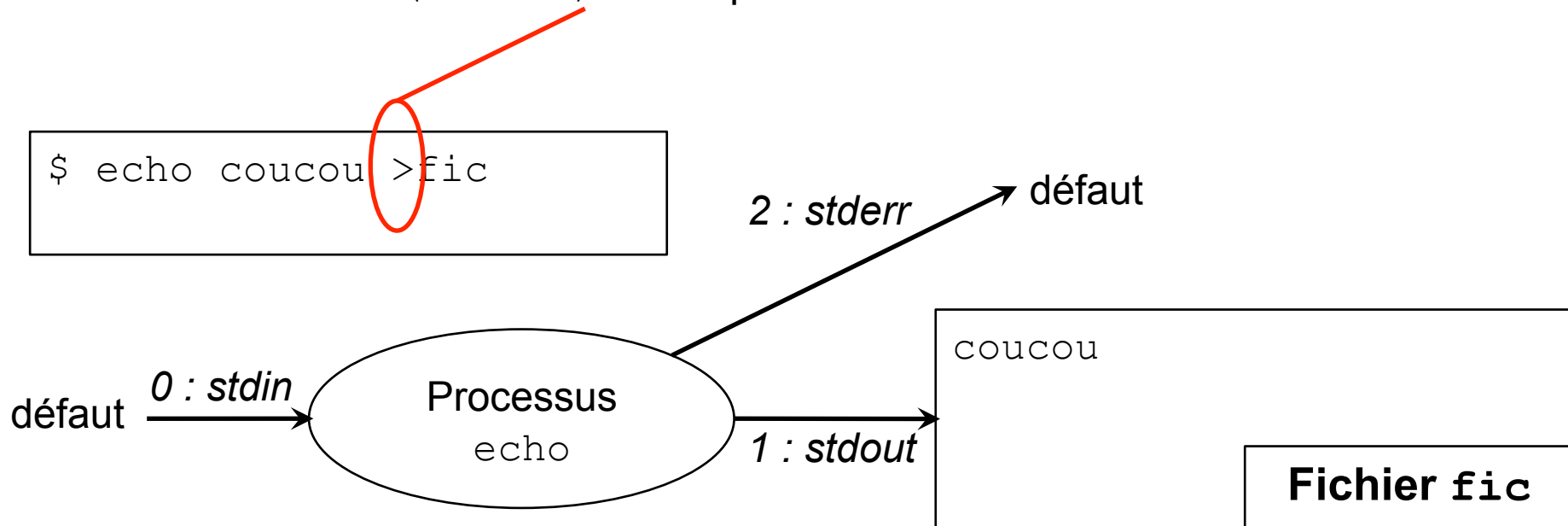
Lance la commande `cmd` dans un nouveau processus **après** avoir ouvert le flux numéro `n` associé au fichier `fic` avec le mode `idoine`



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



Redirection de flux

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

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

Coucou vous

Fichier `fic`

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`

Redirection de flux

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

Coucou vous

Fichier fic

Redirection de flux

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

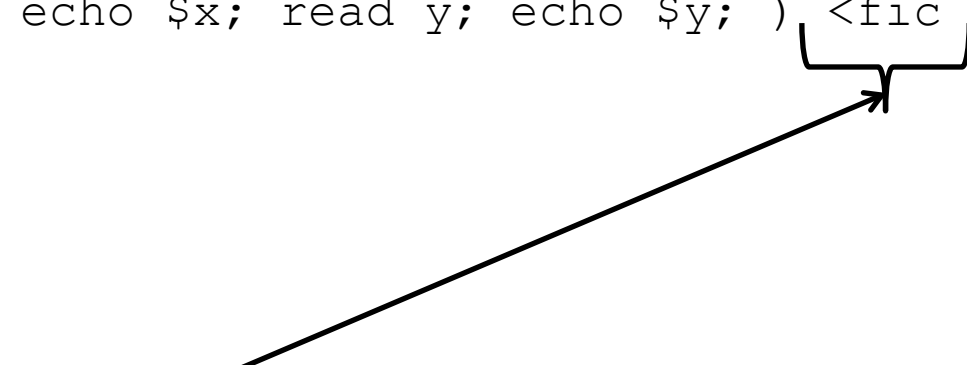
Coucou vous

Fichier fic

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

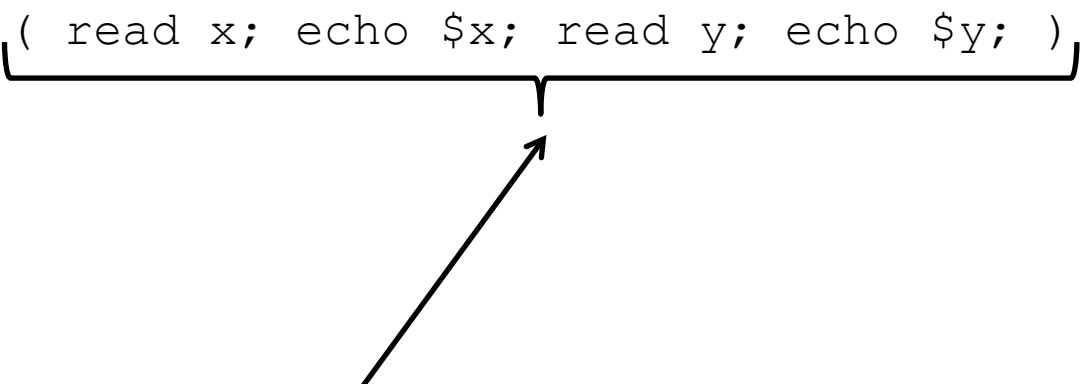
Tête de lecture



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 (nouveau processus)
en redirigeant son entrée

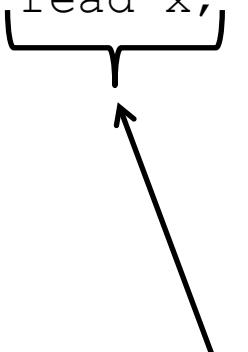
Tête de lecture



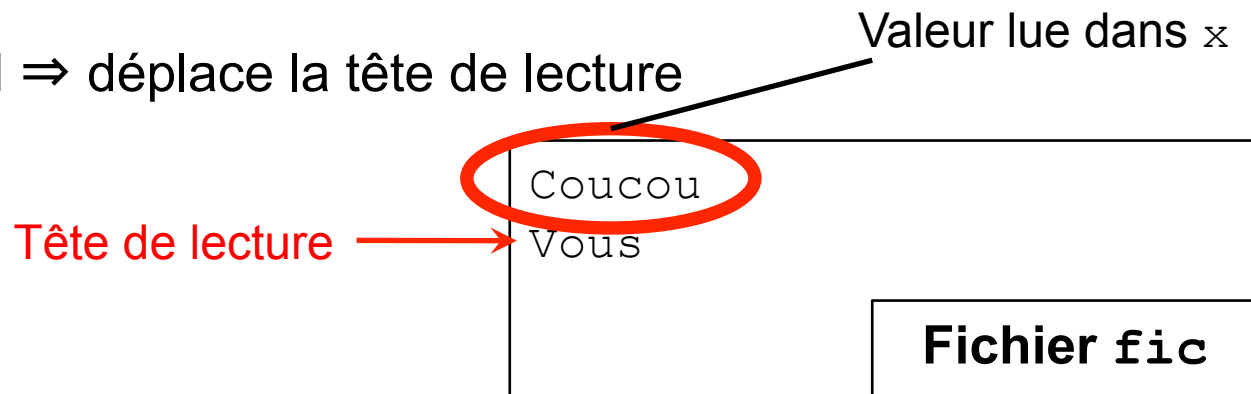
Redirection de flux et regroupement

- Toute expression bash peut être redirigée

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



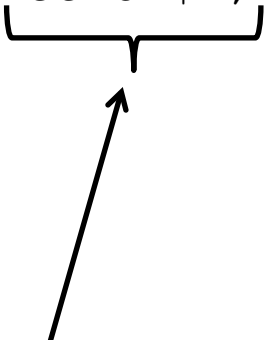
Étape 3 : exécute read ⇒ déplace la tête de lecture



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

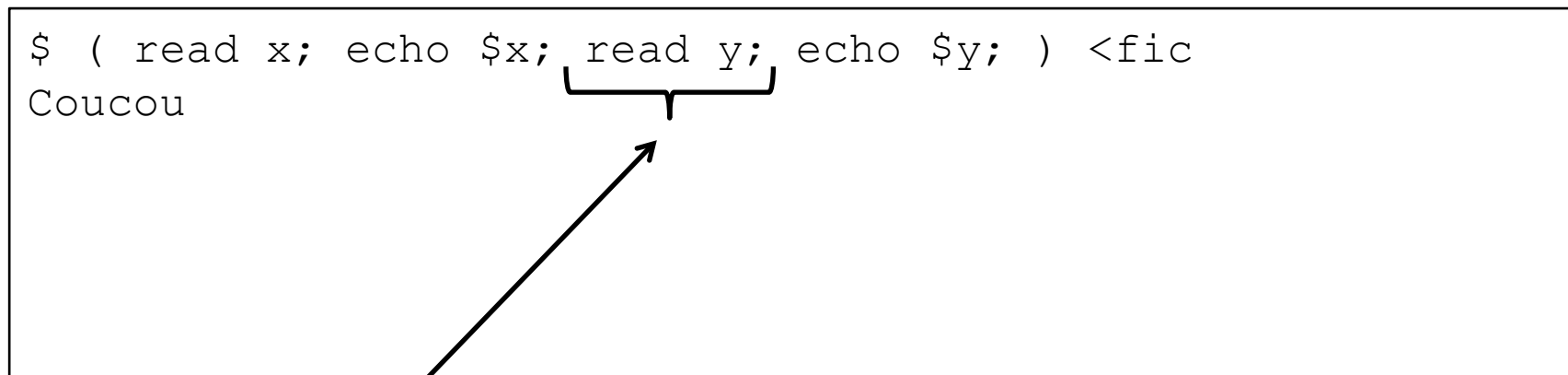
Tête de lecture →



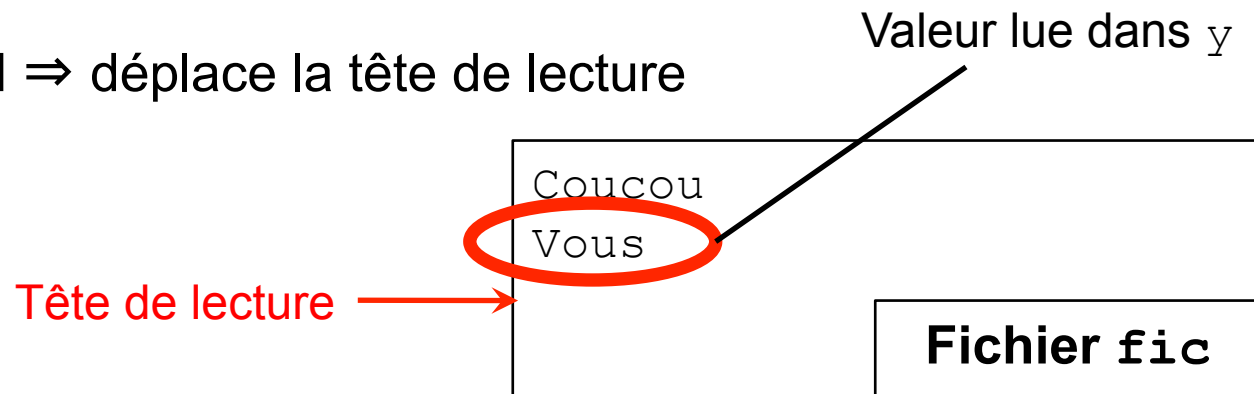
Redirection de flux et regroupement

- Toute expression bash peut être redirigée

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



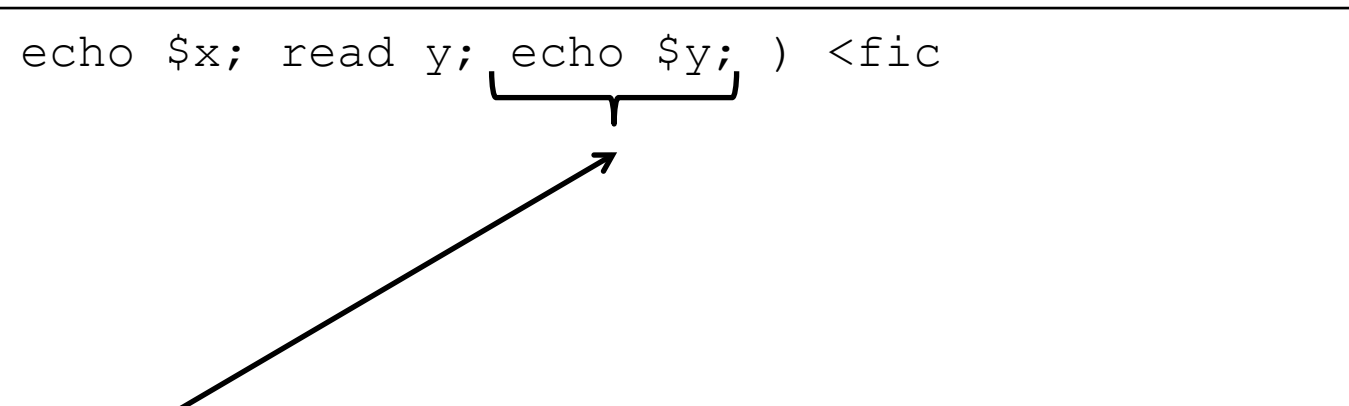
Étape 5 : exécute read ⇒ déplace la tête de lecture



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 →



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

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

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

Lecture d'un flux ligne à ligne

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

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

```
#!/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

Lecture d'un flux ligne à ligne

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

read renvoie vrai car pas fin de fichier

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

Fichier script.sh

Avec ses quatre dromadaires
Don Pedro d'Alfaroubeira

Tête de lecture →

Fichier fic

Lecture d'un flux ligne à ligne

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

```
#!/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

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

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

Tête de lecture →

Fichier fic

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

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

Tête de lecture →

Fichier fic

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)

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

Fichier script.sh

Avec ses quatre dromadaires
Don Pedro d'Alfaroubeira

Fichier fic

Tête de lecture →

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



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

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`

Redirections avancées

```
$ echo coucou >fic
```



```
$ exec 3>fic  
$ echo coucou >&3
```

Intérêt des redirections avancées

- Permet de lire et écrire dans plusieurs fichiers simultanément

```
#!/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
```

script.sh

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



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

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`

- À 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 dans les prochains cours)

Les tubes par l'exemple

- Chaînage de deux commandes utiles (vues au cours 4)
 - `cat fic` : affiche le contenu de `fic` sur la sortie standard
 - `grep motif` : lit ligne à ligne l'entrée et n'affiche que celles qui contiennent `motif`

```
$ cat dromadaire.txt
```

```
Avec ses quatre dromadaires  
Don Pedro d'Alfaroubeira  
Courut le monde et l'admira.  
Il fit ce que je voudrais faire  
Si j'avais quatre dromadaires.
```

```
$ cat dromadaire.txt | grep Pedro
```

```
Don Pedro d'Alfaroubeira
```



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

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/input/mice : souris
 - ...
 - Périphérique logiciel appelé pseudo-périphérique
 - /dev/null : lecture donne une chaîne vide et écriture ne fait rien
 - /dev/tty : terminal
 - /dev/urandom : générateur de nombres aléatoire
 - ...

Redirection et fichiers de périphérique

- On peut utiliser les redirections avec les périphériques
 - Lecture d'une ligne à partir du générateur de nombres aléatoires

```
$ read a </dev/urandom
$ echo $a
?..?ó??oJu檄xÆ4??,??F?s{?6
```

← Ligne de caractères aléatoires

- Suppression d'un affichage

```
$ echo "Je ne veux pas voir ça" >/dev/null
$
```


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`