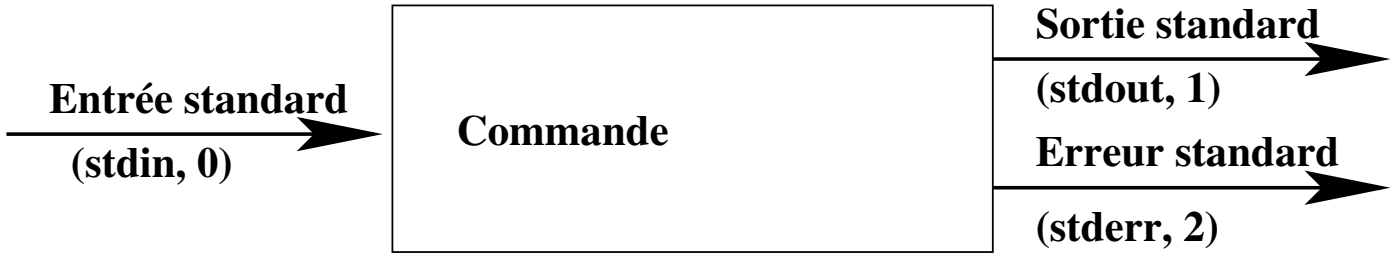
Programmation système - Shell et Commandes UNIX

Gestion des Flux et filtres

Tuyêt Trâm DANG NGOC
<dntt@u-cergy.fr>

Université de Cergy-Pontoise

Flux



- L'entrée standard (stdin : 0)) : le flux d'entrée du programme (par défaut, ce qui est tapé au clavier)
- La sortie standard (stdout : 1) : le flux de sortie du programme (par défaut, il sera affiché à l'écran)
- L'erreur standard : (stderr : 2) : le flux d'erreur du programme (par défaut, il sera affiché à l'écran)

Caractères de manipulation de flux

Syntaxe	Commande
cmd < fic	l'entrée de la commande provient du fichier
cmd << etq	l'entrée de la commande provient des lignes de commandes suivantes jusqu'à la ligne ne contenant que l'étiquette
cmd > fic	la sortie de la commande est placé dans le fichier
cmd >> fic	la sortie de la commande est mise à la suite du fichier
cmd 2 > fic	redirige la sortie d'erreur de la commande dans le fichier
cmd 2 >> fic	redirige la sortie d'erreur de la commande à la suite du fichier
cmd 2 > & desc	redirige la sortie d'erreur de la commande vers le descripteur donné
cmd > & desc	redirige la sortie de la commande vers le descripteur donné
cmd_1 | cmd_2	passe la sortie de la commande 1 comme entrée de la commande 2

Flux de sortie standard

$

- `cmd > fic` : redirige la sortie de la commande dans le fichier qui est créé s'il n'existait pas et dont le contenu est écrasé sinon.
- `cmd >> fic` : redirige la sortie de la commande à la suite du contenu du fichier (qui doit exister).

Flux de sortie standard

- `cmd > fic` : redirige la sortie de la commande dans le fichier qui est créé s'il n'existait pas et dont le contenu est écrasé sinon.
- `cmd >> fic` : redirige la sortie de la commande à la suite du contenu du fichier (qui doit exister).

```
$ echo "Le Corbeau et le Renard" > fic1
```

Flux de sortie standard

- `cmd > fic` : redirige la sortie de la commande dans le fichier qui est créé s'il n'existait pas et dont le contenu est écrasé sinon.
- `cmd >> fic` : redirige la sortie de la commande à la suite du contenu du fichier (qui doit exister).

```
$ echo "Le Corbeau et le Renard" > fic1
$
```

Flux de sortie standard

- `cmd > fic` : redirige la sortie de la commande dans le fichier qui est créé s'il n'existait pas et dont le contenu est écrasé sinon.
- `cmd >> fic` : redirige la sortie de la commande à la suite du contenu du fichier (qui doit exister).

```
$ echo "Le Corbeau et le Renard" > fic1
$ cat fic1
```

Flux de sortie standard

- `cmd > fic` : redirige la sortie de la commande dans le fichier qui est créé s'il n'existait pas et dont le contenu est écrasé sinon.
- `cmd >> fic` : redirige la sortie de la commande à la suite du contenu du fichier (qui doit exister).

```
$ echo "Le Corbeau et le Renard" >
fic1
$ cat fic1
Le Corbeau et le Renard
$
```

Flux de sortie standard

- `cmd > fic` : redirige la sortie de la commande dans le fichier qui est créé s'il n'existait pas et dont le contenu est écrasé sinon.
- `cmd >> fic` : redirige la sortie de la commande à la suite du contenu du fichier (qui doit exister).

```
$ echo "Le Corbeau et le Renard" > fic1
$ cat fic1
Le Corbeau et le Renard
$ echo "Maitre Corbeau, Sur un arbre perché," >> fic1
```

Flux de sortie standard

- `cmd > fic` : redirige la sortie de la commande dans le fichier qui est créé s'il n'existait pas et dont le contenu est écrasé sinon.
- `cmd >> fic` : redirige la sortie de la commande à la suite du contenu du fichier (qui doit exister).

```
$ echo "Le Corbeau et le Renard" > fic1
$ cat fic1
Le Corbeau et le Renard
$ echo "Maitre Corbeau, Sur un arbre perché," >> fic1
$
```

Flux de sortie standard

- `cmd > fic` : redirige la sortie de la commande dans le fichier qui est créé s'il n'existait pas et dont le contenu est écrasé sinon.
- `cmd >> fic` : redirige la sortie de la commande à la suite du contenu du fichier (qui doit exister).

```
$ echo "Le Corbeau et le Renard" > fic1
$ cat fic1
Le Corbeau et le Renard
$ echo "Maitre Corbeau, Sur un arbre perché," >> fic1
$ cat fic1
```

Flux de sortie standard

- `cmd > fic` : redirige la sortie de la commande dans le fichier qui est créé s'il n'existait pas et dont le contenu est écrasé sinon.
- `cmd >> fic` : redirige la sortie de la commande à la suite du contenu du fichier (qui doit exister).

```
$ echo "Le Corbeau et le Renard" > fic1
$ cat fic1
Le Corbeau et le Renard
$ echo "Maitre Corbeau, Sur un arbre perché," >> fic1
$ cat fic1
Le Corbeau et le Renard
Maitre Corbeau, Sur un arbre perché,
$
```

Flux de sortie standard

- `cmd > fic` : redirige la sortie de la commande dans le fichier qui est créé s'il n'existait pas et dont le contenu est écrasé sinon.
- `cmd >> fic` : redirige la sortie de la commande à la suite du contenu du fichier (qui doit exister).

```
$ echo "Le Corbeau et le Renard" > fic1
$ cat fic1
Le Corbeau et le Renard
$ echo "Maitre Corbeau, Sur un arbre perché," >> fic1
$ cat fic1
Le Corbeau et le Renard
Maitre Corbeau, Sur un arbre perché,
$ echo "La Cigale et la Fourmi" > fic1
```

Flux de sortie standard

- `cmd > fic` : redirige la sortie de la commande dans le fichier qui est créé s'il n'existait pas et dont le contenu est écrasé sinon.
- `cmd >> fic` : redirige la sortie de la commande à la suite du contenu du fichier (qui doit exister).

```
$ echo "Le Corbeau et le Renard" > fic1
$ cat fic1
Le Corbeau et le Renard
$ echo "Maitre Corbeau, Sur un arbre perché," >> fic1
$ cat fic1
Le Corbeau et le Renard
Maitre Corbeau, Sur un arbre perché,
$ echo "La Cigale et la Fourmi" > fic1
$
```

Flux de sortie standard

- `cmd > fic` : redirige la sortie de la commande dans le fichier qui est créé s'il n'existait pas et dont le contenu est écrasé sinon.
- `cmd >> fic` : redirige la sortie de la commande à la suite du contenu du fichier (qui doit exister).

```
$ echo "Le Corbeau et le Renard" > fic1
$ cat fic1
Le Corbeau et le Renard
$ echo "Maitre Corbeau, Sur un arbre perché," >> fic1
$ cat fic1
Le Corbeau et le Renard
Maitre Corbeau, Sur un arbre perché,
$ echo "La Cigale et la Fourmi" > fic1
$ cat fic1
```

Flux de sortie standard

- `cmd > fic` : redirige la sortie de la commande dans le fichier qui est créé s'il n'existait pas et dont le contenu est écrasé sinon.
- `cmd >> fic` : redirige la sortie de la commande à la suite du contenu du fichier (qui doit exister).

```
$ echo "Le Corbeau et le Renard" >
fic1
$ cat fic1
Le Corbeau et le Renard
$ echo "Maitre Corbeau, Sur un
arbre perché," >> fic1
$ cat fic1
Le Corbeau et le Renard
Maitre Corbeau, Sur un arbre
perché,
$ echo "La Cigale et la Fourmi" >
fic1
$ cat fic1
La Cigale et la Fourmi
$
```

Flux d'erreur standard

$

- `cmd 2 > fic` : redirige la sortie d'erreur de la commande dans le fichier qui est créé s'il n'existait pas et dont le contenu est écrasé sinon.
- `cmd 2 >> fic` : redirige la sortie d'erreur de la commande à la suite du contenu du fichier (qui doit exister).

Flux d'erreur standard

```
$ ls abc fic1 fic2
```

- `cmd 2 > fic` : redirige la sortie d'erreur de la commande dans le fichier qui est créé s'il n'existait pas et dont le contenu est écrasé sinon.
- `cmd 2 >> fic` : redirige la sortie d'erreur de la commande à la suite du contenu du fichier (qui doit exister).

Flux d'erreur standard

- `cmd 2 > fic` : redirige la sortie d'erreur de la commande dans le fichier qui est créé s'il n'existait pas et dont le contenu est écrasé sinon.
- `cmd 2 >> fic` : redirige la sortie d'erreur de la commande à la suite du contenu du fichier (qui doit exister).

```
$ ls abc fic1 fic2
ls : abc : No such file or
directory
fic1 fic2
$
```

Flux d'erreur standard

- `cmd 2 > fic` : redirige la sortie d'erreur de la commande dans le fichier qui est créé s'il n'existait pas et dont le contenu est écrasé sinon.
- `cmd 2 >> fic` : redirige la sortie d'erreur de la commande à la suite du contenu du fichier (qui doit exister).

```
$ ls abc fic1 fic2
ls : abc : No such file or
directory
fic1 fic2
$ ls abc fic1 fic2 > sortiestd
```

Flux d'erreur standard

- `cmd 2 > fic` : redirige la sortie d'erreur de la commande dans le fichier qui est créé s'il n'existait pas et dont le contenu est écrasé sinon.
- `cmd 2 >> fic` : redirige la sortie d'erreur de la commande à la suite du contenu du fichier (qui doit exister).

```
$ ls abc fic1 fic2
ls : abc : No such file or
directory
fic1 fic2
$ ls abc fic1 fic2 > sortiestd
ls : abc : No such file or
directory
$
```

Flux d'erreur standard

- `cmd 2 > fic` : redirige la sortie d'erreur de la commande dans le fichier qui est créé s'il n'existait pas et dont le contenu est écrasé sinon.
- `cmd 2 >> fic` : redirige la sortie d'erreur de la commande à la suite du contenu du fichier (qui doit exister).

```
$ ls abc fic1 fic2
ls : abc : No such file or
directory
fic1 fic2
$ ls abc fic1 fic2 > sortiestd
ls : abc : No such file or
directory
$ cat sortiestd
```

Flux d'erreur standard

- `cmd 2 > fic` : redirige la sortie d'erreur de la commande dans le fichier qui est créé s'il n'existait pas et dont le contenu est écrasé sinon.
- `cmd 2 >> fic` : redirige la sortie d'erreur de la commande à la suite du contenu du fichier (qui doit exister).

```
$ ls abc fic1 fic2
ls : abc : No such file or
directory
fic1 fic2
$ ls abc fic1 fic2 > sortiestd
ls : abc : No such file or
directory
$ cat sortiestd
fic1 fic2
$
```

Flux d'erreur standard

- `cmd 2 > fic` : redirige la sortie d'erreur de la commande dans le fichier qui est créé s'il n'existait pas et dont le contenu est écrasé sinon.
- `cmd 2 >> fic` : redirige la sortie d'erreur de la commande à la suite du contenu du fichier (qui doit exister).

```
$ ls abc fic1 fic2
ls : abc : No such file or
directory
fic1 fic2
$ ls abc fic1 fic2 > sortiestd
ls : abc : No such file or
directory
$ cat sortiestd
fic1 fic2
$ ls abc fic1 fic2 2> erreurstd
```

Flux d'erreur standard

- `cmd 2 > fic` : redirige la sortie d'erreur de la commande dans le fichier qui est créé s'il n'existait pas et dont le contenu est écrasé sinon.
- `cmd 2 >> fic` : redirige la sortie d'erreur de la commande à la suite du contenu du fichier (qui doit exister).

```
$ ls abc fic1 fic2
ls : abc : No such file or
directory
fic1 fic2
$ ls abc fic1 fic2 > sortiestd
ls : abc : No such file or
directory
$ cat sortiestd
fic1 fic2
$ ls abc fic1 fic2 2> erreurstd
fic1 fic2
$
```

Flux d'erreur standard

- `cmd 2 > fic` : redirige la sortie d'erreur de la commande dans le fichier qui est créé s'il n'existait pas et dont le contenu est écrasé sinon.
- `cmd 2 >> fic` : redirige la sortie d'erreur de la commande à la suite du contenu du fichier (qui doit exister).

```
$ ls abc fic1 fic2
ls : abc : No such file or
directory
fic1 fic2
$ ls abc fic1 fic2 > sortiestd
ls : abc : No such file or
directory
$ cat sortiestd
fic1 fic2
$ ls abc fic1 fic2 2> erreurstd
fic1 fic2
$ cat erreurstd
```

Flux d'erreur standard

- `cmd 2 > fic` : redirige la sortie d'erreur de la commande dans le fichier qui est créé s'il n'existait pas et dont le contenu est écrasé sinon.
- `cmd 2 >> fic` : redirige la sortie d'erreur de la commande à la suite du contenu du fichier (qui doit exister).

```
$ ls abc fic1 fic2
ls : abc : No such file or
directory
fic1 fic2
$ ls abc fic1 fic2 > sortiestd
ls : abc : No such file or
directory
$ cat sortiestd
fic1 fic2
$ ls abc fic1 fic2 2> erreurstd
fic1 fic2
$ cat erreurstd
ls : abc : No such file or
directory
$
```

Flux d'erreur standard

- `cmd 2 > fic` : redirige la sortie d'erreur de la commande dans le fichier qui est créé s'il n'existait pas et dont le contenu est écrasé sinon.
- `cmd 2 >> fic` : redirige la sortie d'erreur de la commande à la suite du contenu du fichier (qui doit exister).

```
$ ls abc fic1 fic2
ls : abc : No such file or
directory
fic1 fic2
$ ls abc fic1 fic2 > sortiestd
ls : abc : No such file or
directory
$ cat sortiestd
fic1 fic2
$ ls abc fic1 fic2 2> erreurstd
fic1 fic2
$ cat erreurstd
ls : abc : No such file or
directory
$ ls abc fic1 fic2 2> erreurstd >
sortiestd
```

Flux d'erreur standard

- `cmd 2 > fic` : redirige la sortie d'erreur de la commande dans le fichier qui est créé s'il n'existait pas et dont le contenu est écrasé sinon.
- `cmd 2 >> fic` : redirige la sortie d'erreur de la commande à la suite du contenu du fichier (qui doit exister).

```
$ ls abc fic1 fic2
ls : abc : No such file or
directory
fic1 fic2
$ ls abc fic1 fic2 > sortiestd
ls : abc : No such file or
directory
$ cat sortiestd
fic1 fic2
$ ls abc fic1 fic2 2> erreurstd
fic1 fic2
$ cat erreurstd
ls : abc : No such file or
directory
$ ls abc fic1 fic2 2> erreurstd >
sortiestd
$
```

Flux d'erreur standard

- `cmd 2 > fic` : redirige la sortie d'erreur de la commande dans le fichier qui est créé s'il n'existait pas et dont le contenu est écrasé sinon.
- `cmd 2 >> fic` : redirige la sortie d'erreur de la commande à la suite du contenu du fichier (qui doit exister).

```
$ ls abc fic1 fic2
ls : abc : No such file or
directory
fic1 fic2
$ ls abc fic1 fic2 > sortiestd
ls : abc : No such file or
directory
$ cat sortiestd
fic1 fic2
$ ls abc fic1 fic2 2> erreurstd
fic1 fic2
$ cat erreurstd
ls : abc : No such file or
directory
$ ls abc fic1 fic2 2> erreurstd >
sortiestd
$ cat erreurstd
```

Flux d'erreur standard

- `cmd 2 > fic` : redirige la sortie d'erreur de la commande dans le fichier qui est créé s'il n'existait pas et dont le contenu est écrasé sinon.
- `cmd 2 >> fic` : redirige la sortie d'erreur de la commande à la suite du contenu du fichier (qui doit exister).

```
$ ls abc fic1 fic2
ls : abc : No such file or
directory
fic1 fic2
$ ls abc fic1 fic2 > sortiestd
ls : abc : No such file or
directory
$ cat sortiestd
fic1 fic2
$ ls abc fic1 fic2 2> erreurstd
fic1 fic2
$ cat erreurstd
ls : abc : No such file or
directory
$ ls abc fic1 fic2 2> erreurstd >
sortiestd
$ cat erreurstd
ls : abc : No such file or
directory
```

Flux d'erreur standard

- `cmd 2 > fic` : redirige la sortie d'erreur de la commande dans le fichier qui est créé s'il n'existait pas et dont le contenu est écrasé sinon.
- `cmd 2 >> fic` : redirige la sortie d'erreur de la commande à la suite du contenu du fichier (qui doit exister).

```
$ ls abc fic1 fic2
ls : abc : No such file or
directory
fic1 fic2
$ ls abc fic1 fic2 > sortiestd
ls : abc : No such file or
directory
$ cat sortiestd
fic1 fic2
$ ls abc fic1 fic2 2> erreurstd
fic1 fic2
$ cat erreurstd
ls : abc : No such file or
directory
$ ls abc fic1 fic2 2> erreurstd >
sortiestd
$ cat erreurstd
ls : abc : No such file or
directory
```

Flux d'erreur standard

- `cmd 2 > fic` : redirige la sortie d'erreur de la commande dans le fichier qui est créé s'il n'existait pas et dont le contenu est écrasé sinon.
- `cmd 2 >> fic` : redirige la sortie d'erreur de la commande à la suite du contenu du fichier (qui doit exister).

```
$ ls abc fic1 fic2
ls : abc : No such file or
directory
fic1 fic2
$ ls abc fic1 fic2 > sortiestd
ls : abc : No such file or
directory
$ cat sortiestd
fic1 fic2
$ ls abc fic1 fic2 2> erreurstd
fic1 fic2
$ cat erreurstd
ls : abc : No such file or
directory
$ ls abc fic1 fic2 2> erreurstd >
sortiestd
$ cat erreurstd
ls : abc : No such file or
directory
```

Flux d'entrée standard

$

- `cmd < fic` l'entrée de la commande provient du fichier
- `cmd << etq` l'entrée de la commande provient des lignes de commandes suivantes jusqu'à la ligne ne contenant que l'étiquette

Flux d'entrée standard

- `cmd < fic` l'entrée de la commande provient du fichier
- `cmd << etq` l'entrée de la commande provient des lignes de commandes suivantes jusqu'à la ligne ne contenant que l'étiquette

```
$ wc < fic1
```

Flux d'entrée standard

- `cmd < fic` l'entrée de la commande provient du fichier
- `cmd << etq` l'entrée de la commande provient des lignes de commandes suivantes jusqu'à la ligne ne contenant que l'étiquette

```
$ wc < fic1
 1 5 23
$
```

Flux d'entrée standard

- `cmd < fic` l'entrée de la commande provient du fichier
- `cmd << etq` l'entrée de la commande provient des lignes de commandes suivantes jusqu'à la ligne ne contenant que l'étiquette

```
$ wc < fic1
 1 5 23
$ wc << EOF
> Maitre Corbeau, Sur un arbre
perché,
> Tenait dans son bec un fromage,
> Maitre Renard, par l'odeur
alléché,
> Lui tint à peu près ce langage :
> EOF
```

Flux d'entrée standard

- `cmd < fic` l'entrée de la commande provient du fichier
- `cmd << etq` l'entrée de la commande provient des lignes de commandes suivantes jusqu'à la ligne ne contenant que l'étiquette

```
$ wc < fic1
 1 5 23
$ wc << EOF
> Maitre Corbeau, Sur un arbre
perché,
> Tenait dans son bec un fromage,
> Maitre Renard, par l'odeur
alléché,
> Lui tint à peu près ce langage :
> EOF
4 25 138
$
```

Tube (*Pipe*) entre flux d'entrée et flux de sortie

$

- `cmd_1 | cmd_2` passe la sortie de la commande 1 comme entrée de la commande 2

cmd1 | cmd2	$\Leftrightarrow$	cmd1 $>$ fictmp cmd2 $<$ fictmp rm fictmp

Tube (*Pipe*) entre flux d'entrée et flux de sortie

```
$ ls fic1 fic2 | wc
```

- `cmd_1 | cmd_2` passe la sortie de la commande 1 comme entrée de la commande 2

cmd1 | cmd2	$\Leftrightarrow$	cmd1 $>$ fictmp cmd2 $<$ fictmp rm fictmp

Tube (*Pipe*) entre flux d'entrée et flux de sortie

- `cmd_1 | cmd_2` passe la sortie de la commande 1 comme entrée de la commande 2

```
$ ls fic1 fic2 | wc
 2 2 10
$
```

cmd1 | cmd2 $\Leftrightarrow$

cmd1 > fictmp
cmd2 < fictmp
rm fictmp

Flux vers descripteurs

- `cmd >& desc` redirige la sortie de la commande vers le descripteur donné (0 : entrée std, 1 : sortie std, 2 : erreur std)

```
$
```

Flux vers descripteurs

- `cmd >& desc` redirige la sortie de la commande vers le descripteur donné (0 : entrée std, 1 : sortie std, 2 : erreur std)

```
$ ls fic1 fic2 abc > resultat
2> resultat
```

Flux vers descripteurs

- `cmd >& desc` redirige la sortie de la commande vers le descripteur donné (0 : entrée std, 1 : sortie std, 2 : erreur std)

```
$ ls fic1 fic2 abc > resultat
2> resultat
cat resultat
fic1
fic2
o such file or directory
$
```

Flux vers descripteurs

- `cmd >& desc` redirige la sortie de la commande vers le descripteur donné (0 : entrée std, 1 : sortie std, 2 : erreur std)

```
$ ls fic1 fic2 abc > resultat
2> resultat
cat resultat
fic1
fic2
o such file or directory
$ ls fic1 fic2 abc > resultat
2>& 1
```

Flux vers descripteurs

- `cmd > & desc` redirige la sortie de la commande vers le descripteur donné (0 : entrée std, 1 : sortie std, 2 : erreur std)

```
$ ls fic1 fic2 abc > resultat
2> resultat
cat resultat
fic1
fic2
o such file or directory
$ ls fic1 fic2 abc > resultat
2>& 1
```

$

Flux vers descripteurs

- `cmd >& desc` redirige la sortie de la commande vers le descripteur donné (0 : entrée std, 1 : sortie std, 2 : erreur std)

```
$ ls fic1 fic2 abc > resultat
2> resultat
cat resultat
fic1
fic2
o such file or directory
$ ls fic1 fic2 abc > resultat
2>& 1
```

```
$ cat resultat
```

Flux vers descripteurs

- cmd > & desc redirige la sortie de la commande vers le descripteur donné (0 : entrée std, 1 : sortie std, 2 : erreur std)

```
$ ls fic1 fic2 abc > resultat
2> resultat
cat resultat
fic1
fic2
o such file or directory
$ ls fic1 fic2 abc > resultat
2>& 1
```

```
$ cat resultat
ls : abc : No such file or
directory
fic1
fic2
$
```

Flux vers descripteurs

- cmd > & desc redirige la sortie de la commande vers le descripteur donné (0 : entrée std, 1 : sortie std, 2 : erreur std)

```
$ ls fic1 fic2 abc > resultat
2> resultat
cat resultat
fic1
fic2
o such file or directory
$ ls fic1 fic2 abc > resultat
2>& 1
```

```
$ cat resultat
ls : abc : No such file or
directory
fic1
fic2
$ ls fic1 fic2 abc 2>
resultat >& 2
```

Flux vers descripteurs

- cmd > & desc redirige la sortie de la commande vers le descripteur donné (0 : entrée std, 1 : sortie std, 2 : erreur std)

```
$ ls fic1 fic2 abc > resultat
2> resultat
cat resultat
fic1
fic2
o such file or directory
$ ls fic1 fic2 abc > resultat
2>& 1
```

```
$ cat resultat
ls : abc : No such file or
directory
fic1
fic2
$ ls fic1 fic2 abc 2>
resultat >& 2
$
```

Flux vers descripteurs

- cmd > & desc redirige la sortie de la commande vers le descripteur donné (0 : entrée std, 1 : sortie std, 2 : erreur std)

```
$ ls fic1 fic2 abc > resultat
2> resultat
cat resultat
fic1
fic2
o such file or directory
$ ls fic1 fic2 abc > resultat
2>& 1
```

```
$ cat resultat
ls : abc : No such file or
directory
fic1
fic2
$ ls fic1 fic2 abc 2>
resultat >& 2
$ cat resultat
```

Flux vers descripteurs

- cmd > & desc redirige la sortie de la commande vers le descripteur donné (0 : entrée std, 1 : sortie std, 2 : erreur std)

```
$ ls fic1 fic2 abc > resultat
2> resultat
cat resultat
fic1
fic2
o such file or directory
$ ls fic1 fic2 abc > resultat
2>& 1
$ cat resultat
ls : abc : No such file or
directory
fic1
fic2
$ ls fic1 fic2 abc 2>
resultat >& 2
$ cat resultat
ls : abc : No such file or
directory
fic1
fic2
$
```

Ordre d'évaluation de la ligne de commande

Ordre de gauche à droite

1. Redirection des entrées/sorties
2. Substitution des variables
3. Substitution des noms de fichiers

Comparaison de fichiers : **diff, cmp, comm**

diff *fic1 fic2*

Trouve les différences entre deux fichiers.

$

fichier a	fichier b	fichier c
un	un	zero
deux	deux	un
trois	trois	trois
quatre	quatre	four

Comparaison de fichiers : **diff, cmp, comm**

diff *fic1 fic2*

Trouve les différences entre deux fichiers.

```
$ diff a b
```

fichier a	fichier b	fichier c
un	un	zero
deux	deux	un
trois	trois	trois
quatre	quatre	four

Comparaison de fichiers : **diff, cmp, comm**

diff *fic1 fic2*

Trouve les différences entre deux fichiers.

```
$ diff a b
$
```

fichier a	fichier b	fichier c
un	un	zero
deux	deux	un
trois	trois	trois
quatre	quatre	four

Comparaison de fichiers : **diff, cmp, comm**

diff *fic1 fic2*

Trouve les différences entre deux fichiers.

```
$ diff a b
$ diff a c
```

fichier a	fichier b	fichier c
un	un	zero
deux	deux	un
trois	trois	trois
quatre	quatre	four

Comparaison de fichiers : **diff, cmp, comm**

diff *fic1 fic2*

Trouve les différences entre deux fichiers.

fichier a	fichier b	fichier c
un	un	zero
deux	deux	un
trois	trois	trois
quatre	quatre	four

```
$ diff a b
$ diff a c
0a1
> zero
2d2
< deux
4c4
< quatre
---
> four
$
```

Comparaison de fichiers : **diff, cmp, comm**

cmp *fic1 fic2*

Compare deux fichiers binaires

comm *fic1 fic2*

Comparer ligne à ligne deux fichiers triés

Lignes de fichiers : **head, tail**

head -n *nb_ligne fic1 fichier*

Afficher les premières lignes du fichier

tail -n *nb_ligne fic1 fichier*

Afficher les dernières lignes du fichier

-f Boucler indéfiniment, en essayant de lire de plus en plus de caractères à la fin du fichier. Très utile pour surveiller des fichiers de log.

$

fichier essai
un
deux
trois
quatre

Lignes de fichiers : **head, tail**

head -n *nb_ligne fic1 fichier*

Afficher les premières lignes du fichier

tail -n *nb_ligne fic1 fichier*

Afficher les dernières lignes du fichier

-f Boucler indéfiniment, en essayant de lire de plus en plus de caractères à la fin du fichier. Très utile pour surveiller des fichiers de log.

```
$ head -n 2 essai
```

fichier essai
un
deux
trois
quatre

Lignes de fichiers : **head, tail**

head -n *nb_ligne fic1 fichier*

Afficher les premières lignes du fichier

tail -n *nb_ligne fic1 fichier*

Afficher les dernières lignes du fichier

-f Boucler indéfiniment, en essayant de lire de plus en plus de caractères à la fin du fichier. Très utile pour surveiller des fichiers de log.

fichier essai
un
deux
trois
quatre

```
$ head -n 2 essai
un
deux
$
```

Lignes de fichiers : **head, tail**

head -n *nb_ligne fic1 fichier*

Afficher les premières lignes du fichier

tail -n *nb_ligne fic1 fichier*

Afficher les dernières lignes du fichier

-f Boucler indéfiniment, en essayant de lire de plus en plus de caractères à la fin du fichier. Très utile pour surveiller des fichiers de log.

fichier essai
un
deux
trois
quatre

```
$ head -n 2 essai
un
deux
$ tail -n 2 essai
```

Lignes de fichiers : **head, tail**

head -n *nb_ligne fic1 fichier*

Afficher les premières lignes du fichier

tail -n *nb_ligne fic1 fichier*

Afficher les dernières lignes du fichier

-f Boucler indéfiniment, en essayant de lire de plus en plus de caractères à la fin du fichier. Très utile pour surveiller des fichiers de log.

fichier essai
un
deux
trois
quatre

```
$ head -n 2 essai
un
deux
$ tail -n 2 essai
trois
quatre
$
```

Tri de lignes de fichiers : **sort**

sort OPTIONS *fichier*

Trie les lignes d'un fichier texte

-n –numeric-sort	dans l'ordre numérique
-d –dictionary-order	dans l'ordre alphabétique
-t *car*	le caractère séparateur de champs
-k	le champs considéré pour le tri

$

Tri de lignes de fichiers : **sort**

sort OPTIONS *fichier*

Trie les lignes d'un fichier texte

-n –numeric-sort	dans l'ordre numérique
-d –dictionary-order	dans l'ordre alphabétique
-t *car*	le caractère séparateur de champs
-k	le champs considéré pour le tri

```
$ cat fichier
```

Tri de lignes de fichiers : **sort**

sort OPTIONS *fichier*

Trie les lignes d'un fichier texte

-n –numeric-sort	dans l'ordre numérique
-d –dictionary-order	dans l'ordre alphabétique
-t *car*	le caractère séparateur de champs
-k	le champs considéré pour le tri

```
$ cat fichier
0 :Harry :Cover :36
1 :John :Doeuf :18
2 :Jean :Breille :1
3 :Rose :Well :2
$
```

Tri de lignes de fichiers : **sort**

sort OPTIONS *fichier*

Trie les lignes d'un fichier texte

-n –numeric-sort	dans l'ordre numérique
-d –dictionary-order	dans l'ordre alphabétique
-t *car*	le caractère séparateur de champs
-k	le champs considéré pour le tri

```
$ cat fichier
0 :Harry :Cover :36
1 :John :Doeuf :18
2 :Jean :Breille :1
3 :Rose :Well :2
$
```

```
$
```

Tri de lignes de fichiers : **sort**

sort OPTIONS *fichier*

Trie les lignes d'un fichier texte

-n –numeric-sort	dans l'ordre numérique
-d –dictionary-order	dans l'ordre alphabétique
-t *car*	le caractère séparateur de champs
-k	le champs considéré pour le tri

```
$ cat fichier
0 :Harry :Cover :36
1 :John :Doeuf :18
2 :Jean :Breille :1
3 :Rose :Well :2
$
```

```
$ cat fichier | sort -t : -k4 -n
```

Tri de lignes de fichiers : **sort**

sort OPTIONS *fichier*

Trie les lignes d'un fichier texte

-n –numeric-sort	dans l'ordre numérique
-d –dictionary-order	dans l'ordre alphabétique
-t *car*	le caractère séparateur de champs
-k	le champs considéré pour le tri

```
$ cat fichier
0 :Harry :Cover :36
1 :John :Doeuf :18
2 :Jean :Breille :1
3 :Rose :Well :2
$
```

```
$ cat fichier | sort -t : -k4 -n
2 :Jean :Breille :1
3 :Rose :Well :2
1 :John :Doeuf :18
0 :Harry :Cover :36
$
```

Tri de lignes de fichiers : **sort**

sort OPTIONS *fichier*

Trie les lignes d'un fichier texte

-n –numeric-sort	dans l'ordre numérique
-d –dictionary-order	dans l'ordre alphabétique
-t *car*	le caractère séparateur de champs
-k	le champs considéré pour le tri

```
$ cat fichier
0 :Harry :Cover :36
1 :John :Doeuf :18
2 :Jean :Breille :1
3 :Rose :Well :2
$
```

```
$ cat fichier | sort -t : -k4 -n
2 :Jean :Breille :1
3 :Rose :Well :2
1 :John :Doeuf :18
0 :Harry :Cover :36
$ ls | sort -d
```

Identification des doublons : **uniq**

uniq OPTIONS *fichier*

Élimine les lignes dupliquées dans un fichier trié par défaut :

-d seulement les lignes répétées

-u seulement les lignes non répétées

$

Identification des doublons : **uniq**

uniq OPTIONS *fichier*

Élimine les lignes dupliquées dans un fichier trié par défaut :

-d seulement les lignes répétées

-u seulement les lignes non répétées

```
$ cat fichier
```

Identification des doublons : **uniq**

uniq OPTIONS *fichier*

Élimine les lignes dupliquées dans un fichier trié par défaut :

-d seulement les lignes répétées

-u seulement les lignes non répétées

```
$ cat fichier
 un
deux
deux
trois
quatre
quatre
cinq
$
```

Identification des doublons : **uniq**

uniq OPTIONS *fichier*

Élimine les lignes dupliquées dans un fichier trié par défaut :

-d seulement les lignes répétées

-u seulement les lignes non répétées

```
$ cat fichier
 un                                    $
deux
deux
trois
quatre
quatre
cinq
$
```

Identification des doublons : **uniq**

uniq OPTIONS *fichier*

Élimine les lignes dupliquées dans un fichier trié par défaut :

-d seulement les lignes répétées

-u seulement les lignes non répétées

```
$ cat fichier
 un
deux
deux
trois
quatre
quatre
cinq
$
```

```
$ cat fichier | uniq
```

Identification des doublons : **uniq**

uniq OPTIONS *fichier*

Élimine les lignes dupliquées dans un fichier trié par défaut :

-d seulement les lignes répétées

-u seulement les lignes non répétées

```
$ cat fichier
 un
deux
deux
trois
quatre
quatre
cinq
$
```

```
$ cat fichier | uniq
 un
deux
trois
quatre
cinq
$
```

Sélection de colonnes : **cut**

cut OPTIONS *fichier*

Supprime une partie de chaque ligne d'un fichier par défaut :

-d separateur	le séparateur
-f listes de champs	les champs séparés
-c listes de caractères	la caractères aux positions indiquées

$

Sélection de colonnes : **cut**

cut OPTIONS *fichier*

Supprime une partie de chaque ligne d'un fichier par défaut :

-d separateur	le séparateur
-f listes de champs	les champs séparés
-c listes de caractères	la caractères aux positions indiquées

```
$ cat fichier
```

Sélection de colonnes : **cut**

cut OPTIONS *fichier*

Supprime une partie de chaque ligne d'un fichier par défaut :

-d separateur	le séparateur
-f listes de champs	les champs séparés
-c listes de caractères	la caractères aux positions indiquées

```
$ cat fichier
0 :Harry :Cover :36
1 :John :Doeuf :16
2 :Jean :Breille :29
3 :Rose :Well :36
$
```

Sélection de colonnes : **cut**

cut OPTIONS *fichier*

Supprime une partie de chaque ligne d'un fichier par défaut :

-d separateur	le séparateur
-f listes de champs	les champs séparés
-c listes de caractères	la caractères aux positions indiquées

```
$ cat fichier
0 :Harry :Cover :36
1 :John :Doeuf :16
2 :Jean :Breille :29
3 :Rose :Well :36
$
```

```
$
```

Sélection de colonnes : **cut**

cut OPTIONS *fichier*

Supprime une partie de chaque ligne d'un fichier par défaut :

-d separateur	le séparateur
-f listes de champs	les champs séparés
-c listes de caractères	la caractères aux positions indiquées

```
$ cat fichier
0 :Harry :Cover :36
1 :John :Doeuf :16
2 :Jean :Breille :29
3 :Rose :Well :36
$
```

```
$ cat fichier | cut -d : -f1,3
```

Sélection de colonnes : **cut**

cut OPTIONS *fichier*

Supprime une partie de chaque ligne d'un fichier par défaut :

-d separateur	le séparateur
-f listes de champs	les champs séparés
-c listes de caractères	la caractères aux positions indiquées

```
$ cat fichier
0 :Harry :Cover :36
1 :John :Doeuf :16
2 :Jean :Breille :29
3 :Rose :Well :36
$
```

```
$ cat fichier | cut -d : -f1,3
 0 :Cover
1 :Doeuf
2 :Breille
3 :Well
$
```

Regroupement de colonnes : **paste**

paste OPTIONS *fichier...*

Regrouper les lignes de différents fichiers

$

Regroupement de colonnes : **paste**

paste OPTIONS *fichier...*

Regrouper les lignes de différents fichiers

```
$ cat fichier
```

Regroupement de colonnes : **paste**

paste OPTIONS *fichier...*

Regrouper les lignes de différents fichiers

```
$ cat fichier
0 :Harry :Cover :36
1 :John :Doeuf :16
2 :Jean :Breille :29
3 :Rose :Well :36
$
```

Regroupement de colonnes : **paste**

paste OPTIONS *fichier...*

Regrouper les lignes de différents fichiers

```
$ cat fichier
0 :Harry :Cover :36
1 :John :Doeuf :16
2 :Jean :Breille :29
3 :Rose :Well :36
$ cat fichier2
```

Regroupement de colonnes : **paste**

paste OPTIONS *fichier...*

Regrouper les lignes de différents fichiers

```
$ cat fichier
0 :Harry :Cover :36
1 :John :Doeuf :16
2 :Jean :Breille :29
3 :Rose :Well :36
$ cat fichier2
2 :New-York
1 :Paris
4 :Tokyo
3 :Berlin
$
```

Regroupement de colonnes : **paste**

paste OPTIONS *fichier...*

Regrouper les lignes de différents fichiers

```
$ cat fichier
0 :Harry :Cover :36
1 :John :Doeuf :16                    $
2 :Jean :Breille :29
3 :Rose :Well :36
$ cat fichier2
2 :New-York
1 :Paris
4 :Tokyo
3 :Berlin
$
```

Regroupement de colonnes : **paste**

paste OPTIONS *fichier...*

Regrouper les lignes de différents fichiers

```
$ cat fichier
0 :Harry :Cover :36
1 :John :Doeuf :16
2 :Jean :Breille :29
3 :Rose :Well :36
$ cat fichier2
2 :New-York
1 :Paris
4 :Tokyo
3 :Berlin
$
```

```
$ paste fichier fichier2
```

Regroupement de colonnes : **paste**

paste OPTIONS *fichier...*

Regrouper les lignes de différents fichiers

```
$ cat fichier
0 :Harry :Cover :36
1 :John :Doeuf :16
2 :Jean :Breille :29
3 :Rose :Well :36
$ cat fichier2
2 :New-York
1 :Paris
4 :Tokyo
3 :Berlin
$
```

```
$ paste fichier fichier2
 0 :Harry :Cover :36
2 :New-York
1 :John :Doeuf :16 1 :Paris
2 :Jean :Breille :29 4 :Tokyo
3 :Rose :Well :36 3 :Berlin
$
```

join OPTIONS *fichier1 fichier2*

Fusionner les lignes de deux fichiers ayant un champ commun

-j1 *n*	jointure sur nième champ du premier fichier
-j2 *n*	jointure sur nième champ du deuxième fichier
-t *car*	le caractère séparateur de champs

$

join OPTIONS *fichier1 fichier2*

Fusionner les lignes de deux fichiers ayant un champ commun

-j1 *n*	jointure sur nième champ du premier fichier
-j2 *n*	jointure sur nième champ du deuxième fichier
-t *car*	le caractère séparateur de champs

```
$ cat fichier
```

join OPTIONS *fichier1 fichier2*

Fusionner les lignes de deux fichiers ayant un champ commun

-j1 *n*	jointure sur nième champ du premier fichier
-j2 *n*	jointure sur nième champ du deuxième fichier
-t *car*	le caractère séparateur de champs

```
$ cat fichier
0 :Harry :Cover :36
1 :John :Doeuf :16
2 :Jean :Breille :29
3 :Rose :Well :36
$
```

join OPTIONS *fichier1 fichier2*

Fusionner les lignes de deux fichiers ayant un champ commun

-j1 *n*	jointure sur nième champ du premier fichier
-j2 *n*	jointure sur nième champ du deuxième fichier
-t *car*	le caractère séparateur de champs

```
$ cat fichier
0 :Harry :Cover :36
1 :John :Doeuf :16
2 :Jean :Breille :29
3 :Rose :Well :36
$ cat fichier2
```

join OPTIONS *fichier1 fichier2*

Fusionner les lignes de deux fichiers ayant un champ commun

-j1 *n* jointure sur nième champ du premier fichier

-j2 *n* jointure sur nième champ du deuxième fichier

-t *car* le caractère séparateur de champs

```
$ cat fichier
0 :Harry :Cover :36
1 :John :Doeuf :16
2 :Jean :Breille :29
3 :Rose :Well :36
$ cat fichier2
1 :Paris
2 :New-York
3 :Berlin
4 :Tokyo
$
```

join OPTIONS *fichier1 fichier2*

Fusionner les lignes de deux fichiers ayant un champ commun

-j1 *n* jointure sur nième champ du premier fichier

-j2 *n* jointure sur nième champ du deuxième fichier

-t *car* le caractère séparateur de champs

```
$ cat fichier
0 :Harry :Cover :36
1 :John :Doeuf :16
2 :Jean :Breille :29
3 :Rose :Well :36
$ cat fichier2
1 :Paris
2 :New-York
3 :Berlin
4 :Tokyo
$
```

$

join OPTIONS *fichier1 fichier2*

Fusionner les lignes de deux fichiers ayant un champ commun

- -j1 *n* jointure sur nième champ du premier fichier
- -j2 *n* jointure sur nième champ du deuxième fichier
- -t *car* le caractère séparateur de champs

```
$ cat fichier
0 :Harry :Cover :36
1 :John :Doeuf :16
2 :Jean :Breille :29
3 :Rose :Well :36
$ cat fichier2
1 :Paris
2 :New-York
3 :Berlin
4 :Tokyo
$
```

```
$ join -t : -j1 1 -j2 1
fichier fichier2
```

join OPTIONS *fichier1 fichier2*

Fusionner les lignes de deux fichiers ayant un champ commun

-j1 *n* jointure sur nième champ du premier fichier

-j2 *n* jointure sur nième champ du deuxième fichier

-t *car* le caractère séparateur de champs

```
$ cat fichier
0 :Harry :Cover :36
1 :John :Doeuf :16
2 :Jean :Breille :29
3 :Rose :Well :36
$ cat fichier2
1 :Paris
2 :New-York
3 :Berlin
4 :Tokyo
$
```

```
$ join -t : -j1 1 -j2 1
fichier fichier2
1 :John :Doeuf :16 :Paris
2 :Jean :Breille :29 :New-York
3 :Rose :Well :36 :Berlin
$
```

grep OPTIONS *fichier...*

Afficher les lignes correspondant à un motif donné

- -v les lignes complémentaires
- -i sans tenir compte de la casse (majuscule=minuscule)
- -n en précédant les lignes sélectionnées par leur numéro de ligne
- -r récursivement dans les répertoires

$

grep OPTIONS *fichier...*

Afficher les lignes correspondant à un motif donné

- -v les lignes complémentaires
- -i sans tenir compte de la casse (majuscule=minuscule)
- -n en précédant les lignes sélectionnées par leur numéro de ligne
- -r récursivement dans les répertoires

```
$ cat fichier
```

grep OPTIONS *fichier...*

Afficher les lignes correspondant à un motif donné

- -v les lignes complémentaires
- -i sans tenir compte de la casse (majuscule=minuscule)
- -n en précédant les lignes sélectionnées par leur numéro de ligne
- -r récursivement dans les répertoires

```
$ cat fichier
0 :Harry :Cover :36
1 :John :Doeuf :16
2 :Jean :Breille :29
3 :Rose :Well :36
$
```

grep OPTIONS *fichier...*

Afficher les lignes correspondant à un motif donné

- -v les lignes complémentaires
- -i sans tenir compte de la casse (majuscule=minuscule)
- -n en précédant les lignes sélectionnées par leur numéro de ligne
- -r récursivement dans les répertoires

$

```
$ cat fichier
0 :Harry :Cover :36
1 :John :Doeuf :16
2 :Jean :Breille :29
3 :Rose :Well :36
$
```

grep OPTIONS *fichier...*

Afficher les lignes correspondant à un motif donné

- -v les lignes complémentaires
- -i sans tenir compte de la casse (majuscule=minuscule)
- -n en précédant les lignes sélectionnées par leur numéro de ligne
- -r récursivement dans les répertoires

```
$ cat fichier | grep 36
```

```
$ cat fichier
0 :Harry :Cover :36
1 :John :Doeuf :16
2 :Jean :Breille :29
3 :Rose :Well :36
$
```

grep OPTIONS *fichier...*

Afficher les lignes correspondant à un motif donné

- -v les lignes complémentaires
- -i sans tenir compte de la casse (majuscule=minuscule)
- -n en précédant les lignes sélectionnées par leur numéro de ligne
- -r récursivement dans les répertoires

```
$ cat fichier
0 :Harry :Cover :36
1 :John :Doeuf :16
2 :Jean :Breille :29
3 :Rose :Well :36
$
```

```
$ cat fichier | grep 36
0 :Harry :Cover :36
3 :Rose :Well :36
$
```

grep OPTIONS *fichier...*

Afficher les lignes correspondant à un motif donné

- -v les lignes complémentaires
- -i sans tenir compte de la casse (majuscule=minuscule)
- -n en précédant les lignes sélectionnées par leur numéro de ligne
- -r récursivement dans les répertoires

```
$ cat fichier
0 :Harry :Cover :36
1 :John :Doeuf :16
2 :Jean :Breille :29
3 :Rose :Well :36
$
```

```
$ cat fichier | grep 36
0 :Harry :Cover :36
3 :Rose :Well :36
$ cat fichier | grep -v 36
```

grep OPTIONS *fichier...*

Afficher les lignes correspondant à un motif donné

- -v les lignes complémentaires
- -i sans tenir compte de la casse (majuscule=minuscule)
- -n en précédant les lignes sélectionnées par leur numéro de ligne
- -r récursivement dans les répertoires

```
$ cat fichier
0 :Harry :Cover :36
1 :John :Doeuf :16
2 :Jean :Breille :29
3 :Rose :Well :36
$
```

```
$ cat fichier | grep 36
0 :Harry :Cover :36
3 :Rose :Well :36
$ cat fichier | grep -v 36
1 :John :Doeuf :16
2 :Jean :Breille :29
$
```

Concatenation et affichage de fichiers : **cat**

cat [*fichier...*]

Affiche sur la sortie standard ce qu'il reçoit en entrée standard. Si **cat** est utilisé avec des fichiers en arguments, il affiche en sortie standard le contenu de ces fichiers.

Affichage page par page : **more, less**

more [*fichier*]

Filtre lecteur de fichier

less [*fichier*]

Filtre lecteur de fichier (plus évolué que **more**

Statistiques sur un fichier : **wc**

wc OPTIONS [*fichier*]

Afficher le nombre d'octets, de mots et de lignes d'un fichier.

-c	affiche le nombre d'octets
-l	affiche le nombre de ligne
-m	affiche le nombre de caractères
-w	affiche le nombre de mots

Duplication de flux : **tee**

tee OPTIONS [*fichiers...*]

Renvoi sur qui est lu sur l'entrée standard à la fois sur la sortie standard, mais aussi dans un (des) fichier(s).

Éditeur de flux : **sed**

sed

Éditeur de flux

-e commande	commande d'édition semblable à celles de l'éditeur **vi**

(voir cours suivant)

exemple :

```
sed -e "s/:/ /g"  fichier
```

Remplace tous les ' :' du fichier par un blanc.

Modification de caractères : **tr**

tr

Langage de recherche de motif et manipulation de texte
(voir cours suivant)

Éditeur de flux évolué : **awk**

awk

Langage évolué de recherche de motif et manipulation de texte (voir cours suivant)

Affichage des personnes logguées : **w**

w

Affiche les personnes logguées et ce qu'elles font

$

Affichage des personnes logguées : w

w

Affiche les personnes logguées et ce qu'elles font

```
$ w
```

Affichage des personnes logguées : w

w

Affiche les personnes logguées et ce qu'elles font

```
$ w
 23 :39 :20 up 4 :17, 6 users, load average : 0,00, 0,00,
0,00
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT
dntt :0 - 19 :23?xdm? 1 :02 0.10s - :0
card pts/0 :0.0 20 :05 46 :05 0.02s 0.02s ssh bigdaddy
dntt pts/1 :0.0 20 :12 4 :05 0.06s 0.01s man ps
dntt pts/2 :0.0 19 :28 2.00s 0.38s 0.38s vi -u NONE
cours.tex
dntt pts/3 :0.0 21 :36 2 :00m 0.03s 0.03s -pdksh
dntt pts/4 :0.0 23 :33 0.00s 0.00s 0.00s w
$
```

Usage des ressources : **time**

time *commande*

Lance des programmes et résume les usages de ressources

$

- real : temps réel de l'exécution de la commande
- user : temps CPU utilisé par le programme utilisateur
- sys : temps utilisé par le système pour gérer l'exécution du job

Usage des ressources : **time**

time *commande*

Lance des programmes et résume les usages de ressources

```
$ time ./long_programme
```

- real : temps réel de l'exécution de la commande
- user : temps CPU utilisé par le programme utilisateur
- sys : temps utilisé par le système pour gérer l'exécution du job

Usage des ressources : **time**

time *commande*

Lance des programmes et résume les usages de ressources

```
$ time ./long_programme
0.16s real 0.15s user 0.00s system
$
```

- real : temps réel de l'exécution de la commande
- user : temps CPU utilisé par le programme utilisateur
- sys : temps utilisé par le système pour gérer l'exécution du job

Horloge système : **date** *format*

date *format*

Afficher ou configurer la date et l'heure du système

YYMMddhhmm	configure la date avec la date donnée
+ format	affiche la date suivant le format donné (%d %M %Y %H %m %s)

$

Horloge système : **date** *format*

date *format*

Afficher ou configurer la date et l'heure du système

YYMMddhhmm	configure la date avec la date donnée
+ format	affiche la date suivant le format donné (%d %M %Y %H %m %s)

```
$ date "+%d-%M-%Y %H :%m"
```

Horloge système : **date** *format*

date *format*

Afficher ou configurer la date et l'heure du système

YYMMddhhmm	configure la date avec la date donnée
+ format	affiche la date suivant le format donné (%d %M %Y %H %m %s)

```
$ date "+%d-%M-%Y %H :%m"
24-12-2006 23 :01
$
```

Horloge système : **date** *format*

date *format*

Afficher ou configurer la date et l'heure du système

YYMMddhhmm	configure la date avec la date donnée
+ format	affiche la date suivant le format donné (%d %M %Y %H %m %s)

```
$ date "+%d-%M-%Y %H :%m"
24-12-2006 23 :01
$ date "+%H heures %m minutes"
```

Horloge système : **date** *format*

date *format*

Afficher ou configurer la date et l'heure du système

YYMMddhhmm	configure la date avec la date donnée
+ format	affiche la date suivant le format donné (%d %M %Y %H %m %s)

```
$ date "+%d-%M-%Y %H :%m"
24-12-2006 23 :01
$ date "+%H heures %m minutes"
23 heures 01 minutes
$
```

Enregistrement des entrées/sorties du shell

script *fichier*

Enregistre toutes les entrées/sorties du shell dans un fichier.
Terminer avec 'exit' ou un envoi de fin de fichier.