



I106B

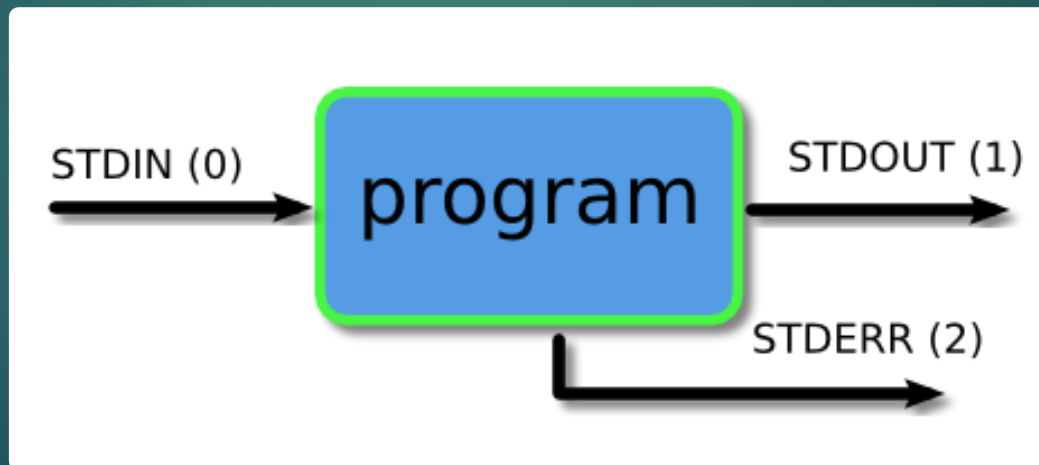
12. Redirection

[HTTPS://WEB.ARCHIVE.ORG/WEB/20161027104208/HTTP://RYANSTUTORIALS.NET/LINUXTUTORIAL/PIPING.PHP](https://web.archive.org/web/20161027104208/http://ryanstutorials.net/linuxtutorial/piping.php)

Entrée/sorties standard

2

- ▶ Chaque commande exécutée possède 3 flux standards toujours ouverts:
 - `stdin` : entrée standard
 - `stdout` : sortie standard
 - `stderr` : sortie d'erreur standard



Par défaut

3

- ▶ `stdout` = le terminal (écran)
- ▶ `stderr` = le terminal (écran)
- ▶ `stdin` = le terminal (clavier)
 - On peut rentrer les données au clavier
 - Pour terminer la lecture sur `stdin` :
Ctrl+D (\Leftrightarrow EOF)

Exemple

4

- ▶ `sort` sans aucun paramètre :

```
professeur@professeur-VirtualBox:~$ sort  
toto  
tata  
tutu
```

- ▶ `<Ctrl+D>` *flush* le contenu de `stdin`, ce qui signale à `sort` qu'il peut effectuer son tri :

```
tata  
toto  
tutu
```

En général...

- ▶ Les commandes qui traitent un fichier texte peuvent s'appeler sans donner de fichier en paramètre. Dans ce cas, elles travaillent sur `stdin`.

Redirection

6

On peut rediriger :

- ▶ `stdin` pour utiliser un fichier à la place du terminal :

```
tr -d '\n' < data.txt
```

- ▶ `stdout` pour utiliser un fichier à la place du terminal :

```
ls -l > results.txt
```

- ▶ `stderr` pour utiliser un fichier à la place du terminal :

```
rmkdir / 2> errors.txt
```

Raffinements

7

- ▶ Plusieurs redirections

```
sort <in.txt >out.txt 2>err.txt
```

- ▶ Ajout dans un fichier

```
ls /usr/bin > cmd.txt
```

```
ls /usr/local/bin >> cmd.txt
```

- ▶ Fichier oubliette (*null device*)

```
find / -name toto 2> /dev/null
```

- Imaginons qu'on veut compter le nombre de lignes du manuel de man contenant la suite de caractères « man » :

```
man man > etape1.txt
```

```
grep man < etape1.txt > etape2.txt
```

```
wc -l < etape2.txt
```

⇒ Beaucoup de fichiers de redirection intermédiaires et inutiles !

- Solution : les pipes !

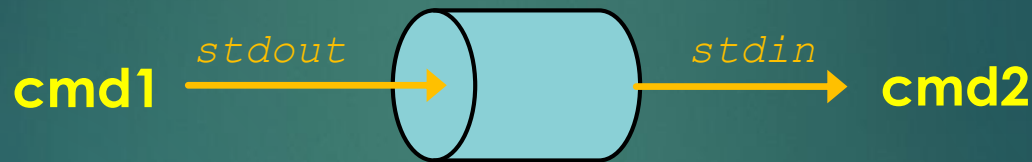
```
man man | grep man | wc -l
```


Pipe

9

`cmd1 | cmd2`

- ▶ Connecte `stdout` de `cmd1` à `stdin` de `cmd2`



- ▶ On peut les chaîner
- ➔ Philosophie Unix :
 - des commandes simples et très ciblées
 - qu'on peut combiner ensemble à l'infini

Substitution de commande


10

- Utilisation de `stdout` d'une commande comme argument d'une autre commande.

```
echo "la date est $(date)"
```

ou

```
echo "la date est `date`"
```



Apostrophe inversée,
difficile à trouver sur
certains claviers

Combinaison de commandes

11

▶ `cmd1 ; cmd2`

exécute `cmd1` **puis** `cmd2`

▶ `cmd1 && cmd2`

exécute `cmd1` puis `cmd2` **si** `cmd1` a réussi

▶ `cmd1 || cmd2`

exécute `cmd1` puis `cmd2` **si** `cmd1` a échoué

▶ `cmd1 & cmd2`

exécute `cmd1` et `cmd2` **en parallèle**
(`cmd1` est exécuté en arrière plan)