Entrée/Sortie

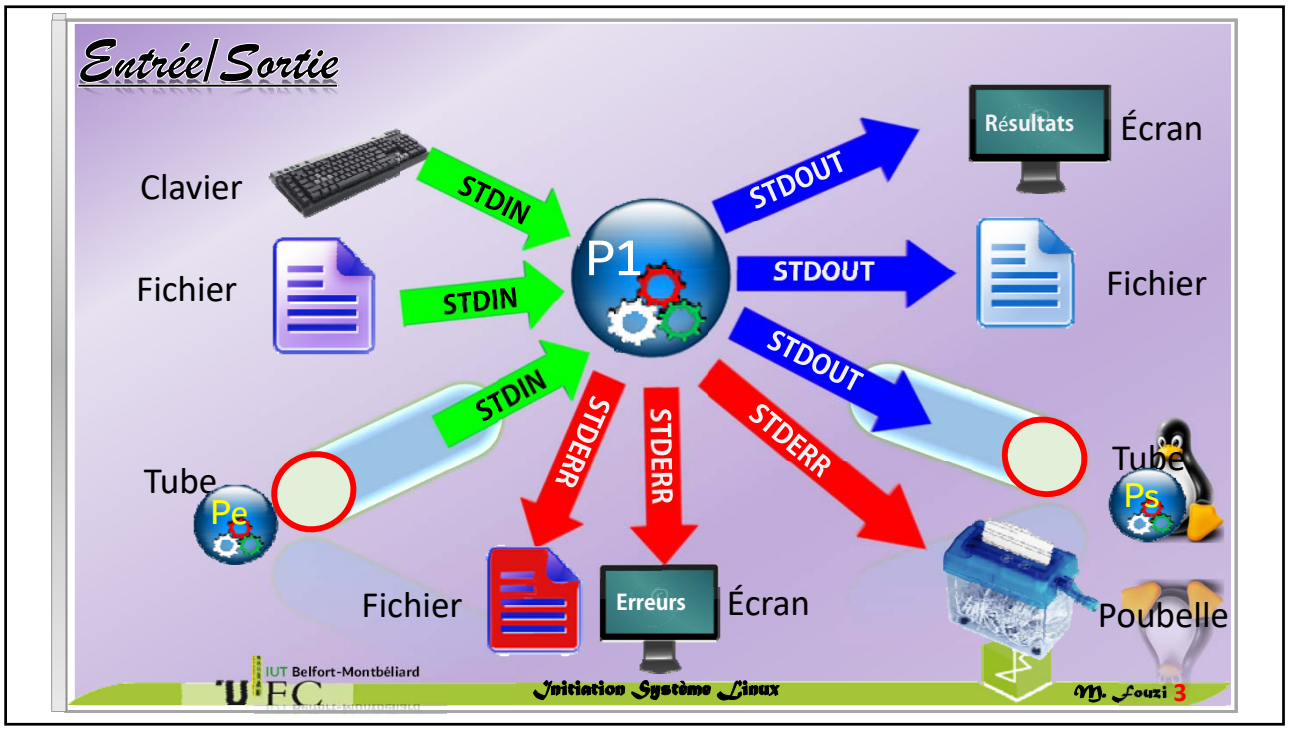
Exécutable :

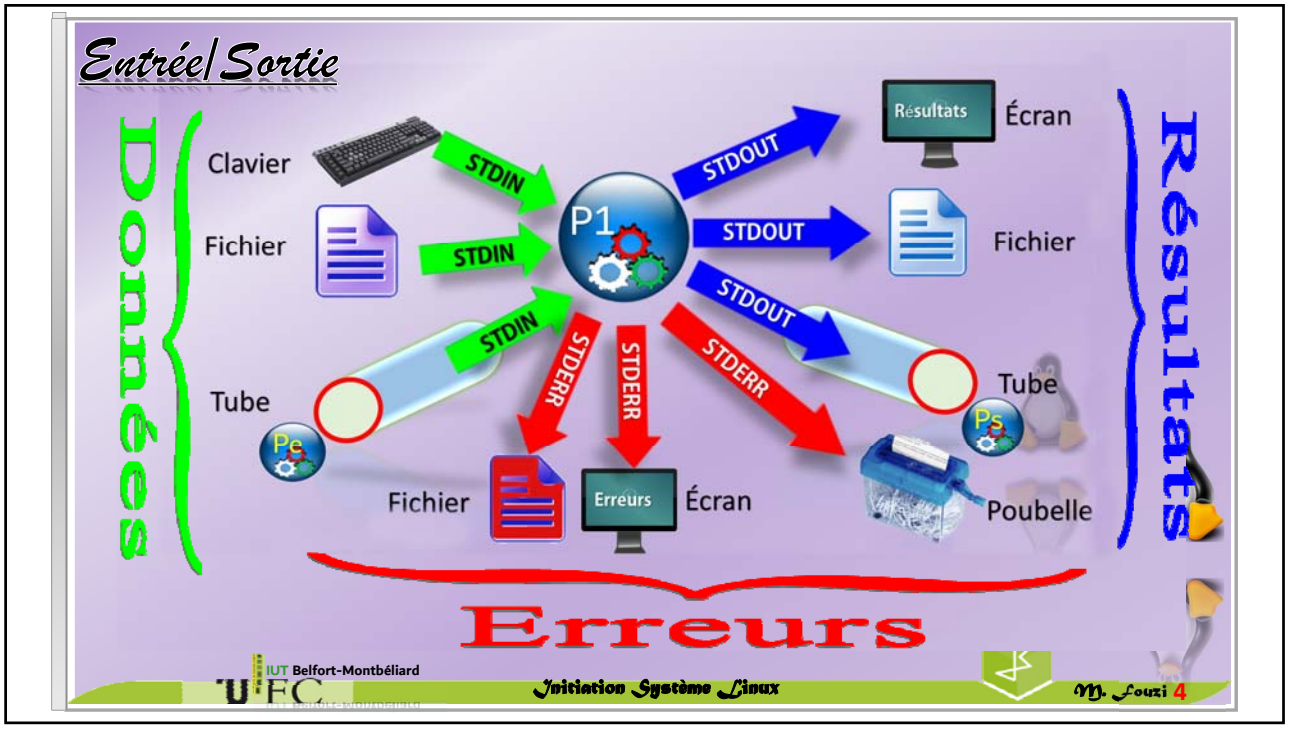
Un exécutable est un fichier qui contient un code.

Processus :

Un processus (process) est un programme en cours d'exécution dans une machine
plus un état courant de son environnement

Cet état est mémorisé dans PCB (*Process Control Block*)
Son contenu est : conteur ordinal, état des registres, ressources utilisées, variables de l'environnement ….


Entrée/Sortie
Clavier
Fichier
Tube
STDIN
STDIN
STDIN
P1
STDOUT
STDOUT
STDOUT
STDERR
STDERR
STDERR
Résultats
Écran
Fichier
Tube
Poubelle
Fichier
Erreurs
Écran
IUT Belfort-Montbéliard
Initiation Système Linux
M. Fouzi 3


Entrée/Sortie
Données
Clavier
Fichier
Tube
STDIN
STDIN
STDIN
P1
STDOUT
STDOUT
STDOUT
STDERR
STDERR
STDERR
Résultats
Écran
Fichier
Tube
Poubelle
Fichier
Erreurs
Écran
Résultats
Erreurs
IUT Belfort-Montbéliard
Initiation Système Linux
M. Fouzi 4

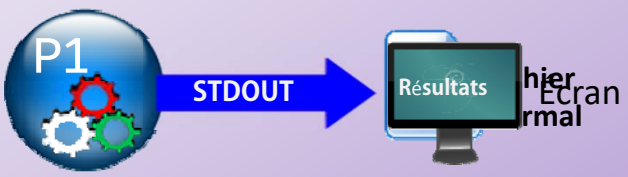
Exécution d'une commande : cmd

1. L'utilisateur utilise l'Entrée Standard de Données (ESD) pour saisir la commande : cmd
2. Le shell crée le processus P pour exécuter la commande cmd
3. Les résultats de cmd sont redirigés vers la Sortie Standard de Résultats (SSR)
4. Les erreurs de cmd sont redirigés vers la Sortie Standard d'Erreurs (SSE)

Linux considère un fichier comme étant une suite finie d'octets éventuellement vide.

1/ Redirection de sortie

1.1/ Création d'un fichier



Syntaxe : cmd > [chemin/] nom_fich

Le caractère " > " indique au shell de rediriger le résultat de cmd sur un fichier ordinaire dont le chemin et le nom sont spécifiés et non pas sur SSR.

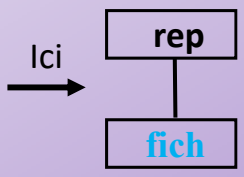
Exécution de la commande : cmd > fich

Si "fich" n'existe pas *Alors*
- Création du fichier ordinaire "fich"
- Les résultats de "cmd" sont rangés dans "fich"

Sinon
  Si "fich" est ordinaire *Alors*
  - L'ancien contenu de "fich" est perdu.
  - Les résultats de "cmd" seront rangés dans "fich"

  Sinon - Erreur est affichée
  Fsi
Fsi

3° cas: fich existe et non ordi.



echo "Bonjour " > fich

Étudier le cas où fich est :
- Un lien physique
- De type lien symbolique

Exemple:

Compter le nombre de fichiers visibles

```
ls >fich
wc -l fich
rm fich
```

wc [- cwl] fich
c : compte les caractères
w : compte les mots
l : compte les lignes

Options possibles :
cw ; cl ; wl ; clw

Création d'un fichier vide : >fich

Si "fich" n'existe pas alors
Création de "fich" ordinaire et vide
Sinon le contenue de "fich" est perdu
Fsi

Création d'un fichier ordinaire Nommé "fich" Non vide

```
cat >fich
# Ici votre texte
# ....
```
CTRL + d

1/ Redirection de sortie
1.2/ Ajout dans un fichier

Syntaxe : cmd >> [chemin/] nom_fich

Les caractères " >> " indiquent au shell **d'ajouter** le résultat de cmd sur un fichier ordinaire dont le chemin et le nom sont spécifiés et non pas sur SSR.

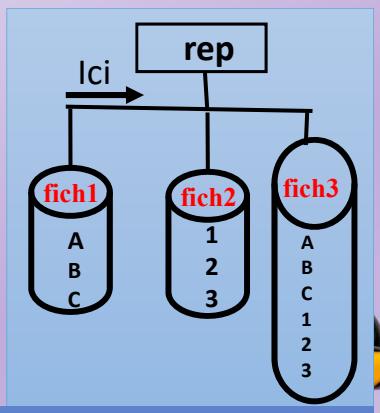
Exécution de la commande : cmd >> fich

Si "fich" n'existe pas *Alors*
- Création du fichier ordinaire "fich"
- Les résultats de "cmd" sont rangés dans "fich"

Sinon
Si "fich" est ordinaire *Alors*
- L'ancien contenu de "fich" est conservé.
- Les résultats de "cmd" seront ajoutés en fin de fichier "fich"

Sinon - **Erreur** est affichée

Fsi
Fsi



Exemple : Concaténation de fich1 et fich2 avec fich3
cat fich1 fich2 >> fich3

2/ Redirection en entrée

Syntaxe : cmd < [chemin/] nom_fich

Le caractère " < " indique au shell de rediriger l'entrée de cmd sur un fichier ordinaire dont le chemin et le nom sont spécifiés et non pas sur ESD.

Exemple : On suppose que sous le répertoire courant, il y a au moins un programme java. Compter le nombre de programmes java visibles.

```
echo  *.java  > fichiers_java
wc  -w  < fichiers_java
```


Les caractères "<<" indiquent au shell que l'entrée standard suit jusqu'à l'apparition d'un caractère spéciale (noté ici @) dans le flot de données.

Syntaxe : cmd << @

Exemple :

- Les caractères en entrée seront acceptés jusqu'à l'apparition de@.
- @ doit être sur une ligne vierge.
- ranger le résultat dans "f"

```
cat  << @  >f
# Ici votre texte
# ...
@
```

```
>f  <<@ cat ou <<@  >@ cat  ou  >f  cat  <<@ ou  cat  >f  <<@  ou ...
```

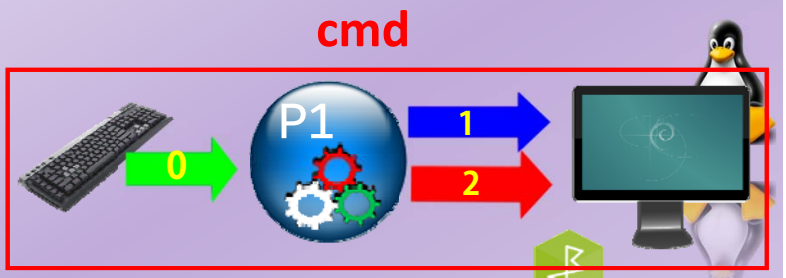
3/ Descripteur de fichier

A chaque fichier est associé un entier appelé descripteur de fichier (file descriptor) : 0, 1, 2, ...

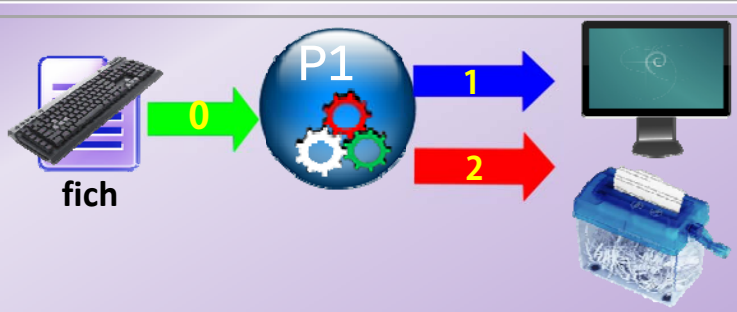
Ces entiers référencent des flux de données associés :

- à un fichier
- à un périphérique (terminal)
- aux descripteurs d'un autre processus

Flots	Descripteur
ESD	0
SSR	1
SSE	2



Exemple 1 :

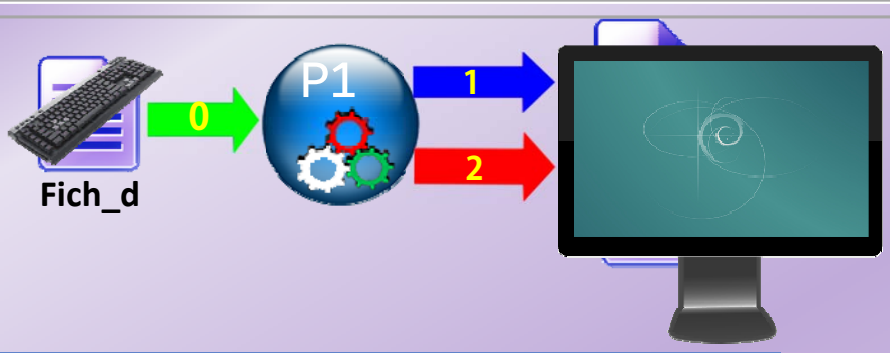



```
cmd    0< fich    2 >/dev/null
```

Remarque :

cmd 0<fich est idem à cmd <fich

Exemple 1 :




```
cmd  < fich_d  1 >fich_r  2 >fich_e  &
```

Remarque :

& est utile pour lancer un processus en arrière plan

>fich_r et 1>fich_r sont idem

Fin de la première partie du cours sur la redirection