Université d'Oran 1 Ahmed Ben Bella
Faculté des Sciences Exactes & Appliquées
Département d'Informatique

Module Systèmes d'Exploitation I (L2/S4)

Fiche TP n°3 **Communication entre processus** **Année 2015/2016**

Exercice 1

Contrairement aux processus lancés simultanément qui s'exécutent sans relation entre eux, des processus concurrents sont synchronisés entre eux par la production d'information de l'un, et la consommation d'information de l'autre. Il est nécessaire pour cela, que le processus producteur soit apte à produire des caractères sur la sortie standard, et que le processus consommateur soit apte à lire des caractères sur l'entrée standard.

C'est ce que réalise la commande « pipe » dont la syntaxe est : **commande1 | commande2**

Exemple : **$ ps -alx | grep bash**

« ps » produit une liste de processus ; le pipe sert de raccordement de la sortie standard du processus « ps » sur l'entrée standard du processus « grep », qui va lui-même n'afficher que les lignes contenant la chaîne « bash ».

1. *Exécuter la ligne de commande de cet exemple.*
2. *Créer un fichier texte de nom « text1 » dont le contenu est :*

1 : la commande *pipe* sert à la communication entre processus
3 : la commande *tee* est utile pour capturer les informations qui circulent dans un pipe
2 : la commande *tee* recopie son entrée standard sur sa sortie standard et sur un fichier
4 : la commande *tee* peut être utilisée pour sauvegarder dans un fichier les traces des informations qui circulent sur sa sortie standard.
1 : qu'est-ce qu'un *pipe* et que fait la commande *tee* ?

i. *en utilisant la commande « | » écrire une commande qui compte le nombre de ligne contenant le mot « pipe » dans le fichier « text1 ».*
ii. *Regarder dans le manuel la commande « tee ». En utilisant cette commande modifier le résultat obtenu en i) afin de récupérer dans un fichier « text2 » les lignes contenant le mot « pipe ».*
iii. *Ecrire une ligne de commande qui permet de créer un fichier « text3 » qui contiendra les lignes du fichier « text1 » contenant le mot « pipe ». Ces lignes devront être triées sur le premier champ de chaque ligne. Enfin la commande affiche le nombre de ces lignes.*

Exercice 2

a. *Combien y a-t'il de processus actifs sur le système ?*
b. *Combien d'utilisateurs sont connectés sur le système ? En utilisant la commande « sort », afficher la liste de ces utilisateurs connectés, triée :*

i. *par ordre alphabétique.*
ii. *selon l'heure de connexion.*

a. *Combien le répertoire « /etc » a t-il de fichiers répertoire ? Utiliser les commandes « ls », « grep » et « wc ».*

b. *Combien le répertoire « /etc » a t-il de sous-répertoires ? Utiliser les commandes « ls », « grep » et « wc ».*
c. *Afficher la liste des fichiers du répertoire courant, triée par ordre de taille des fichiers.*

Exercice 3

1. *Utiliser la commande « cat » et l'opérateur « > » pour créer les fichiers « fich1 » et « fich2 ». Toujours en utilisant la commande « cat » mais cette fois en regardant le manuel créer le fichier « fich3 » constitué de la concaténation des fichiers « fich1 » et « fich2 ».*
2. *Lancer la commande « cat fich1 fich-inexistant » avec le fichier « fich-inexistant » inexistant et le fichier « fich1 » existant.*
3. *Nous pouvons rediriger la sortie standard en utilisant l'opérateur « > » ; par exemple « cat fich1 fich-inexistant > trace » ou*

   « cat fich1 fich-inexistant 1>trace ».

   Lancez les deux commandes ; que constatez-vous ?

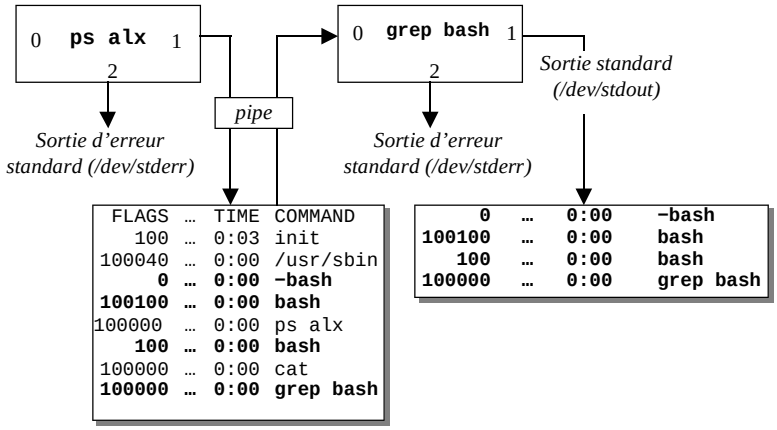
4. *Lancez la commande du a) en redirigeant la sortie d'erreur dans le fichier « err ». Comment peut-on rediriger la sortie standard sur la sortie d'erreur ?*

Corrigé Fiche de TP n°3

Exercice 1

1 Exécuter la ligne de commande : `$ ps –alx | grep bash`

```
$ ps alx | grep bash
    0  500  248  240  0   0  1228   804  117319  S  1   0:00  –bash
 100100 500  340  337  16  0  1220   788  117319  S  p0  0:00  bash
 100000 500  349  340  17  0  812    356  11cce9  D  p0  0:00  grep  bash
   100  500  345  344  18  0  1224   804  117319  S  p1  0:00  bash
```



2 Créer un fichier texte de nom "text1" dont le contenu est :

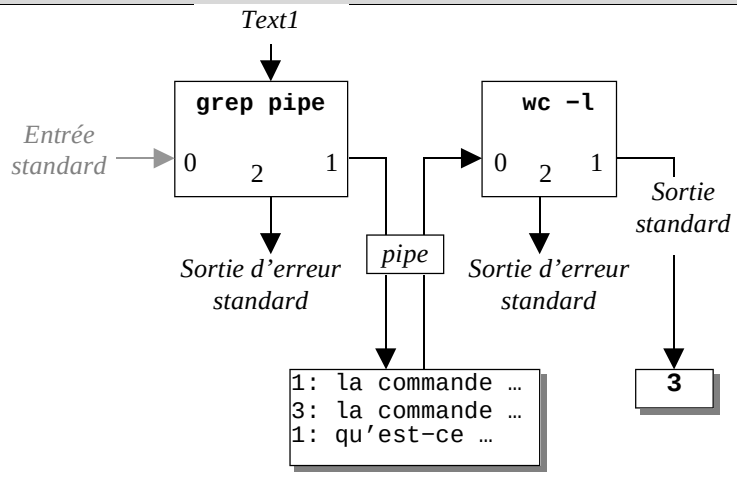
3

```
$ cat text1
1: la commande pipe sert a la communication entre processus
3: la commande tee est utile pour capturer les informations qui circulent
&
2: la commande tee recopie son entree standard sur sa sortie standard et &
4: la commande tee peut etre utilisee pour sauvegarder dans un fichier &
1: qu'est–ce qu'un pipe et que fait la commande tee ?
```

i En utilisant la commande "|" écrire une commande qui compte le nombre de ligne contenant le mot "pipe" dans le fichier "text1"

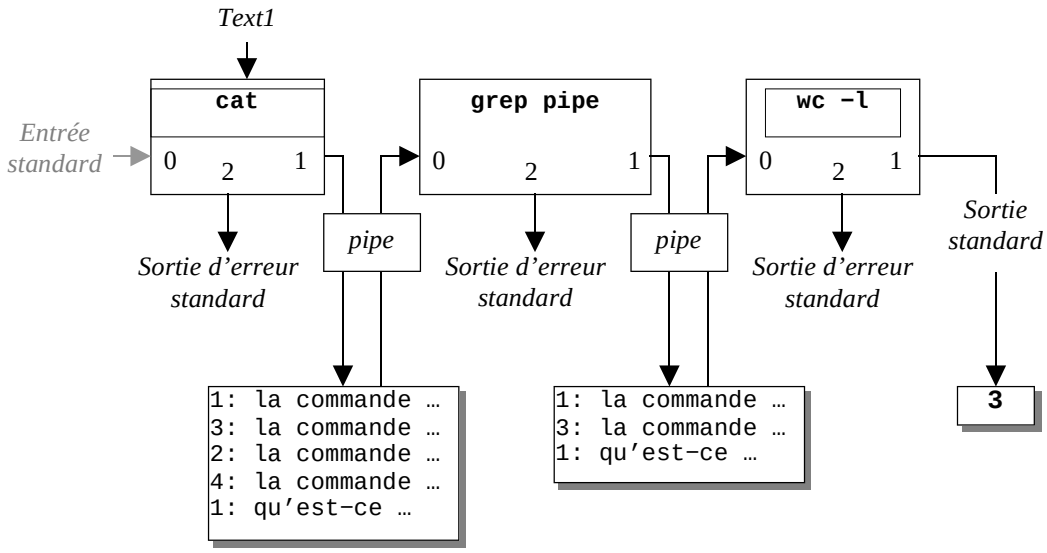
1ère solution :

```
$ grep pipe text1
1: la commande pipe sert a la communication entre processus
3: la commande tee est utile pour capturer les informations qui circulent
&
1: qu'est–ce qu'un pipe et que fait la commande tee ?
$ grep pipe text1 | wc –l
     3
```



2nde solution :

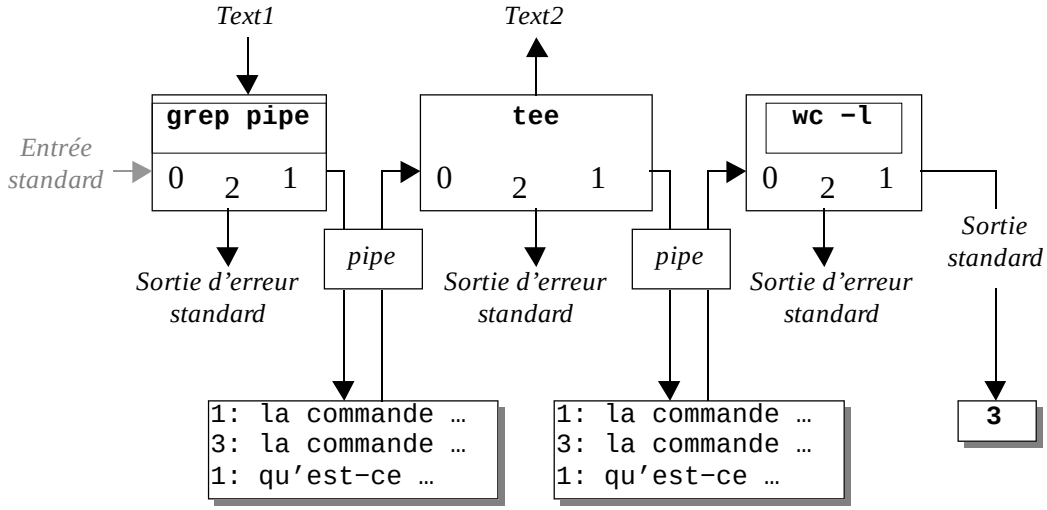
```
$ cat test1 | grep pipe
1: la commande pipe sert a la communication entre processus
3: la commande tee est utile pour capturer les informations qui circulent
&
1: qu'est-ce qu'un pipe et que fait la commande tee ?
$ cat test1 | grep pipe | wc -l
     3
```



ii *En utilisant la commande "tee" modifier le résultat obtenu en i) afin de récupérer dans un fichier text2 les lignes contenant le mot pipe*

iii

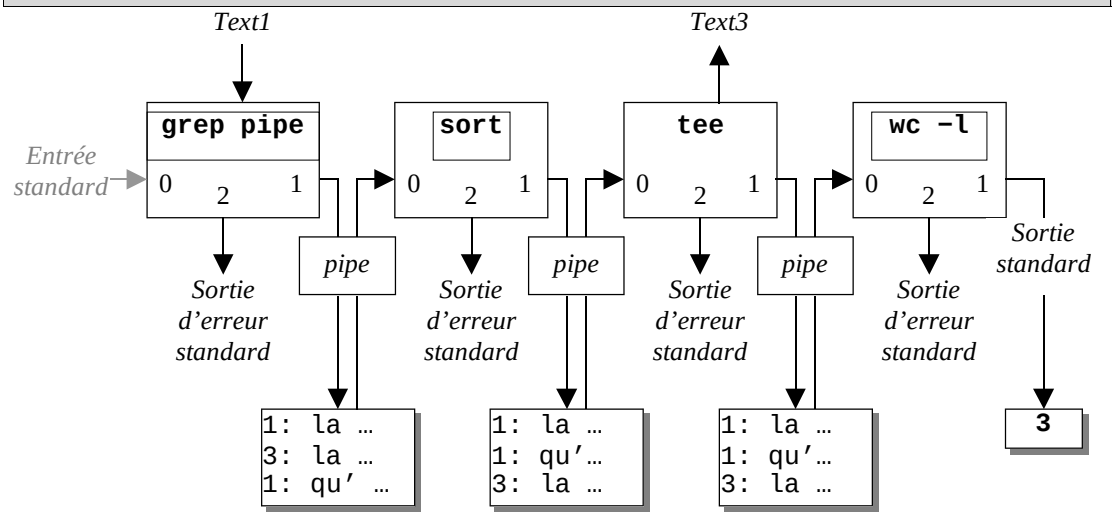
```
$ grep pipe test1 | tee test2 | wc -l
   3
$ cat test2
1: la commande pipe sert a la communication entre processus
3: la commande tee est utile pour capturer les informations qui circulent
&
1: qu'est-ce qu'un pipe et que fait la commande tee ?
```



iv *Ecrire une ligne de commande qui permette de créer un fichier "text3" qui contiendra les lignes du fichier "text1" contenant le mot "pipe". Ces lignes devront être triées sur le premier champ de chaque ligne. Enfin, la commande affiche le nombre de ces lignes.*

1ère solution :

```
$ sort test1
1: la commande pipe sert a la communication entre processus
1: qu'est-ce qu'un pipe et que fait la commande tee ?
2: la commande tee recopie son entree standard sur sa sortie standard et &
3: la commande tee est utile pour capturer les informations qui circulent
&
4: la commande tee peut etre utilisee pour sauvegarder dans un fichier &
$ grep pipe test1 | sort | tee text3 | wc -l
     3
$ cat text3
1: la commande pipe sert a la communication entre processus
1: qu'est-ce qu'un pipe et que fait la commande tee ?
3: la commande tee est utile pour capturer les informations qui circulent
&
```



2nde solution :

```
$ cat test1 | sort | grep pipe | tee text3 | wc -l
     3
$ cat text3
1: la commande pipe sert a la communication entre processus
1: qu'est-ce qu'un pipe et que fait la commande tee ?
3: la commande tee est utile pour capturer les informations qui circulent
&
```

Exercice 2

1 Combien y a-t'il de processus actifs sur le système ?

Les processus actifs de tous les utilisateurs connectés (option "a" de "ps") :

```
$ ps a | wc -l
    22
```

5 Combien d'utilisateurs sont connectés sur le système ?

```
$ who
meric    tty1     Nov 28 20:41
meric    ttyp0    Nov 28 20:41 (:0.0)
meric    ttyp1    Nov 28 21:49 (:0.0)
meric    ttyp2    Nov 28 22:12 (:0.0)
guest    ttyp3    Nov 28 22:12 (localhost)
$ who | wc -l
     5
```

Remarque : La commande "who −q" donne la liste (non ordonnée) et le nombre d'utilisateurs connectés sur le système.

```
$ who −q
meric meric meric meric guest
# users=5
```

En utilisant la commande « sort », afficher la liste de ces utilisateurs connectés, triée :

i *par ordre alphabétique.*

ii

```
$ who | sort
guest    ttyp3    Nov 28 22:12 (localhost)
meric    tty1     Nov 28 20:41
meric    ttyp0    Nov 28 20:41 (:0.0)
meric    ttyp1    Nov 28 21:49 (:0.0)
meric    ttyp2    Nov 28 22:12 (:0.0)
```

iii *selon l'heure de connexion (champ n° 2).*

iv

```
$ who | sort +2
meric    tty1     Nov 28 20:41
meric    ttyp0    Nov 28 20:41 (:0.0)
meric    ttyp1    Nov 28 21:49 (:0.0)
meric    ttyp2    Nov 28 22:12 (:0.0)
guest    ttyp3    Nov 28 22:12 (localhost)
```

6 Combien le répertoire "/etc" a t'il de fichiers répertoire ? Utiliser les commandes "ls", "grep" et "wc".

Utiliser la commande "ls −la" qui affiche tous les fichiers contenus dans le répertoire "/etc", y compris les fichiers cachés (i.e. les fichiers dont le nom commence par un "."). Le nombre total de fichiers répertoires contenu dans ce répertoire correspond au nombre de lignes commençant par le caractère "d" (qui distingue les fichiers répertoires des autres fichiers).

```
$ ls −la /etc | grep ^d
drwxr−xr−x  22 root     root         2048 Nov 28 20:36 .
drwxr−xr−x  17 root     root         1024 Jan  7  1999 ..
drwxr−xr−x  12 root     root         1024 Jan  7  1999 X11
drwxr−xr−x   2 root     root         1024 Jan  7  1999 cron.daily
drwxr−xr−x   2 root     root         1024 Aug 26  1997 cron.hourly
drwxr−xr−x   2 root     root         1024 Jul 25  1997 cron.monthly
drwxr−xr−x   2 root     root         1024 Jan  7  1999 cron.weekly
drwxr−xr−x   2 root     root         1024 Jan  7  1999 default
...
$ ls −la /etc | grep ^d | wc −l
    22
```

7 Combien le répertoire "/etc" a t'il de sous−répertoires ? Utiliser les commandes "ls", "grep" et "wc"

Les fichiers répertoires "." et ".." ne sont pas des sous−répertoires. Il faut donc calculer le nombre de fichiers répertoires non cachés (i.e. les fichiers répertoires dont le nom ne commence pas par un ".").

1ère solution : en utilisant les commandes "`ls −l /etc`" ou "`ls −ld /etc/*`" (l'option "d" permet de ne pas afficher le contenu des sous−répertoires). Dans les deux cas, le nombre de sous−répertoires correspond au nombre de lignes commençant par le caractère "d" qui distingue les fichiers répertoires des autres fichiers.

```
$ ls −l /etc | grep ^d
drwxr−xr−x  12 root     root         1024 Jan  7  1999 X11
drwxr−xr−x   2 root     root         1024 Jan  7  1999 cron.daily
drwxr−xr−x   2 root     root         1024 Aug 26  1997 cron.hourly
drwxr−xr−x   2 root     root         1024 Jul 25  1997 cron.monthly
drwxr−xr−x   2 root     root         1024 Jan  7  1999 cron.weekly
drwxr−xr−x   2 root     root         1024 Jan  7  1999 default
...
$ ls −l /etc | grep ^d | wc −l
    20
```

```
$ ls -ld /etc/* | grep ^d
drwxr-xr-x    12  root   root   1024   Jan 7 1999      /etc/X11
drwxr-xr-x    2   root   root   1024   Jan 7 1999      /etc/cron.daily
drwxr-xr-x    2   root   root   1024   Aug 26 1997     /etc/cron.hourly
drwxr-xr-x    2   root   root   1024   Jul 25 1997     /etc/cron.monthly
drwxr-xr-x    2   root   root   1024   Jan 7 1999      /etc/cron.weekly
drwxr-xr-x    2   root   root   1024   Jan 7 1999      /etc/default
...
$ ls -ld /etc/* | grep ^d | wc -l
    20
```

2nde solution : en utilisant la commande "`ls -p /etc`". Dans ce cas, le nombre total de sous-répertoires correspond au nombre de lignes se terminant par le caractère "/" qui distingue les fichiers répertoires des autres fichiers.

```
$ ls -p /etc
DIR_COLORS   default/         initrunlvl@      minicom.users    rpc
HOSTNAME     dosemu.conf      inittab          motd             securetty
X11/         dosemu.users     ioctl.save       mtab             security/
adjtime      drums.o3         issue            mtools.conf      sendmail.cf
aliases      drums.sb         issue.net        nmh/             sendmail.cw
...
$ ls -p /etc | grep /$
X11/
cron.daily/
cron.hourly/
cron.monthly/
cron.weekly/
default/
...
$ ls -p /etc | grep /$ | wc -l
    20
```

8 Afficher la liste des fichiers du répertoire courant, triée par ordre de taille des fichiers.

La taille des fichiers est renseignée par le 5ième champ (i.e. le champ d'indice 4) dans la liste affichée par la commande "`ls -l`".

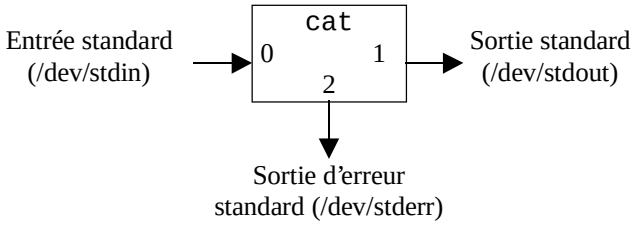
```
$ ls -l
total 546
drwxr-xr-x   3 meric    meric        1024 Jan 11  1999 GNUstep
-rw-rw-r--   1 meric    meric         410 Nov 28 20:41 Xrootenv.0
drwxrwxr-x   2 meric    meric        1024 Apr 10  1999 archives
-rw-rw-r--   1 meric    meric      495185 Mar  4  1999 article.tar.gz
drwxrwxr-x   4 meric    meric        1024 Apr 12  1999 matisse
-rw-rw-r--   1 meric    meric       20480 Apr 13  1999 matisse.tar
drwx------   2 meric    meric        1024 Sep 28 23:38 nsmail
drwxr-xr-x   6 meric    meric        1024 Mar 26  1999 oRis
-rwxrwxr-x   1 meric    meric       27357 Apr 12  1999 simumanu.txt
$ ls -l | sort +4
total 546
-rw-rw-r--   1 meric    meric         410 Nov 28 20:41 Xrootenv.0
drwxrwxr-x   2 meric    meric        1024 Apr 10  1999 archives
drwxrwxr-x   4 meric    meric        1024 Apr 12  1999 matisse
drwxr-xr-x   3 meric    meric        1024 Jan 11  1999 GNUstep
drwxr-xr-x   6 meric    meric        1024 Mar 26  1999 oRis
drwx------   2 meric    meric        1024 Sep 28 23:38 nsmail
-rw-rw-r--   1 meric    meric       20480 Apr 13  1999 matisse.tar
-rwxrwxr-x   1 meric    meric       27357 Apr 12  1999 simumanu.txt
-rw-rw-r--   1 meric    meric      495185 Mar  4  1999 article.tar.gz
```

La commande suivante permet de n'afficher que les fichiers ordinaires (même principe que dans le 4).

```
$ ls -l | grep ^- | sort +4
-rw-rw-r--   1 meric    meric         410 Nov 28 20:41 Xrootenv.0
-rw-rw-r--   1 meric    meric       20480 Apr 13  1999 matisse.tar
-rwxrwxr-x   1 meric    meric       27357 Apr 12  1999 simumanu.txt
-rw-rw-r--   1 meric    meric      495185 Mar  4  1999 article.tar.gz
```

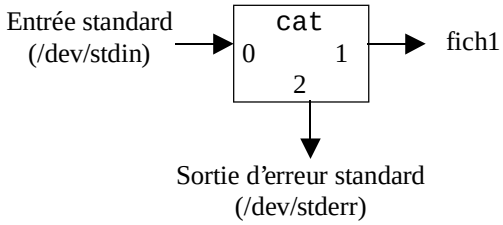
Exercice 3

2 Utiliser la commande "cat" et l'opérateur ">" pour créer les fichiers "fich1" et "fich2".
La commande "cat" sans paramètre lit l'entrée standard et affiche sur la sortie standard.




```
$ cat
Lecture de l'entree standard                    (entré au clavier)
Lecture de l'entree standard                    (affiché à l'écran)
Ecriture sur la sortie standard                 (entré au clavier)
Ecriture sur la sortie standard                 (affiché à l'écran)
```

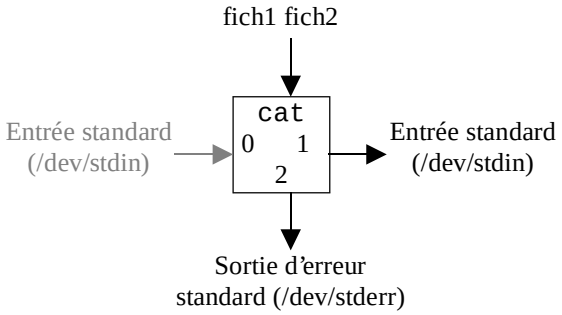
Le caractère ">" permet de rediriger la sortie vers un fichier. **Le fichier est créé s'il n'existait pas et écrasé s'il existait.**




```
$ cat > fich1
Lecture de l'entree standard                    (entré au clavier)
Redirection dans le fichier fich1               (entré au clavier)
$ cat fich1
Lecture de l'entree standard
Redirection dans le fichier fich1
$ cat > fich2
Lecture de l'entree standard  (entré au clavier)
Redirection dans le fichier fich2               (entré au clavier)
$ cat fich2
Lecture de l'entree standard
Redirection dans le fichier fich2
```

Toujours en utilisant la commande "cat" mais cette fois en regardant le manuel créer le fichier "fich3" constitué de la concaténation des fichiers "fich1" et "fich2".

1ère solution : La commande "cat" peut prendre 1 ou plusieurs noms de fichiers en paramètres. Dans ce cas, elle recopie le contenu de ces fichiers sur la sortie standard.




```
$ cat fich1 fich2
Lecture de l'entree standard
Redirection dans le fichier fich1
Lecture de l'entree standard
Redirection dans le fichier fich2
$ cat fich1 fich2 > fich3
$ cat fich3
```

```
Lecture de l'entree standard
Redirection dans le fichier fich1
Lecture de l'entree standard
Redirection dans le fichier fich2
```

2nde solution : la double redirection ">>" permet de rediriger la sortie vers un fichier. Si le fichier n'existe pas, il est créé, sinon, les données sont ajoutées à la fin du fichier.

```
$ cat fich1 >> fich3
$ cat fich3
Lecture de l'entree standard
Redirection dans le fichier fich1
$ cat fich2 >> fich3
$ cat fich3
Lecture de l'entree standard
Redirection dans le fichier fich1
Lecture de l'entree standard
Redirection dans le fichier fich2
```

10 Lancer la commande "cat fich1 fich−inexistant" avec le fichier "fich−inexistant" inexistant et le fichier "fich1" existant.

```
$ cat fich1 fich−inexistant
Lecture de l'entree standard
Redirection dans le fichier fich1
cat: fich−inexistant: Aucun fichier ou répertoire de ce type.
```

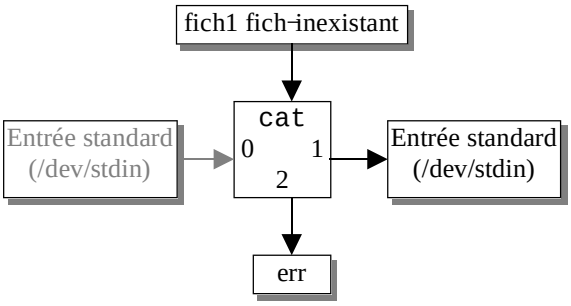
11 Nous pouvons rediriger la sortie standard en utilisant lopérateur ">" ; par exemple "cat fich1 fich−inexistant > trace" ou "cat fich1 fich−inexistant 1>trace". Lancez les deux commandes ; que constatez−vous ?

```
$ cat fich1 fich−inexistant > trace
cat: fich−inexistant: Aucun fichier ou répertoire de ce type.
$ cat trace
Lecture de l'entree standard
Redirection dans le fichier fich1
$ rm trace
$ cat fich1 fich−inexistant 1> trace
cat: fich−inexistant: Aucun fichier ou répertoire de ce type.
$ cat trace
Lecture de l'entree standard
Redirection dans le fichier fich1
```

Deux constats :

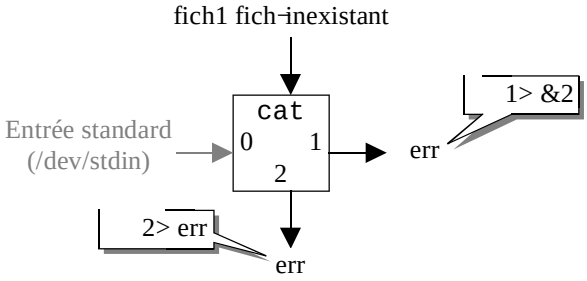
1. La sortie standard et la sortie d'erreur sont bien distinct.
2. "1>" et ">" sont équivalents. L'opérateur ">" permet de rediriger par défaut la sortie standard.

12 Lancez la commande du 2) en redirigeant la sortie d'erreur dans le fichier "err".



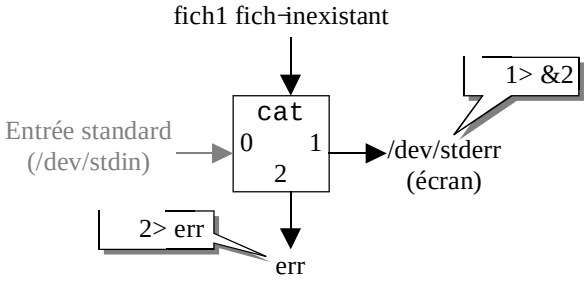

```
$ cat fich1 fich-inexistant 2> err
Lecture de l'entree standard
Redirection dans le fichier fich1
$ cat err
cat: fich-inexistant: Aucun fichier ou répertoire de ce type.
```

Comment peut-on rediriger la sortie standard sur la sortie d'erreur ?




```
$ cat fich1 fich-inexistant 2>err >&2
$ cat err
Lecture de l'entree standard
Redirection dans le fichier fich1
cat: fich-inexistant: Aucun fichier ou répertoire de ce type.
```

L'ordre des redirections est très important &




```
$ cat fich1 fich-inexistant >&2 2>err
Lecture de l'entree standard
Redirection dans le fichier fich1
$ cat err
cat: fich-inexistant: Aucun fichier ou répertoire de ce type.
```