Shell 2
redirections, tubes

Juliusz Chroboczek

22 septembre 2015

L'endroit où un processus Unix obtient ses entrées s'appelle son *entrée standard* ; l'endroit où il écrit sa sortie s'appelle sa *sortie standard* ; et l'endroit où il affiche ses messages d'erreur s'appelle sa *sortie d'erreur*. Normalement, l'entrée standard est connectée au clavier, tandis que la sortie standard et la sortie d'erreur sont connectées à l'écran. Il est possible de connecter ces *descripteurs* à des fichiers, ou à d'autres processus.

1 Redirections

Il est possible de connecter l'entrée ou une des sorties d'un processus à un fichier. On dit alors que ce descripteur est *redirigé*.

On connecte la sortie standard d'un processus à un fichier en faisant suivre l'invocation d'un programme d'un signe supérieur « `>` » et du nom de fichier vers lequel on veut la rediriger. Par exemple, la commande suivante écrit une ligne de texte dans un fichier nommé `zazie.text` :

```
echo 'Tu causes, tu causes,' > zazie.text
```

Si le fichier `zazie.text` existait déjà, son contenu est écrasé.

L'opérateur de redirection « `>>` » est similaire à l'opérateur « `>` » en ce qu'il redirige la sortie standard. Cependant, si le fichier de destination existe déjà, il n'écrase pas celui-ci, mais écrit à sa fin :

```
echo "c'est tout ce que tu sais faire." >> zazie.text
```

Normalement, la sortie d'erreur n'est pas redirigée : même si la sortie standard l'est, les messages d'erreur apparaissent sur le terminal. Cependant, on peut rediriger la sortie d'erreur à l'aide de l'opérateur « `2>` » :

```
gcc -Wall bonjour.c 2> erreurs.log
```

Rediriger la sortie d'erreur est particulièrement utile lorsqu'un processus s'exécute en tâche de fond (voir poly suivant) et on ne veut pas qu'il nous dérange :

```
./a.out > resultats.log 2> erreurs.log &
```

Enfin, l'entrée standard peut être redirigée en faisant suivre l'invocation d'un programme d'un signe inférieur « < » et du nom d'un fichier existant :

```
sort < zazie.text
```

2 Tubes

On peut aussi connecter la sortie standard d'un processus à l'entrée standard d'un autre ; les deux processus s'exécutent alors en parallèle. Pour cela, on sépare les invocations de deux programmes par le symbole « | ». Par exemple, la ligne de commande suivante connecte la sortie du processus qui exécute `a.out` à l'entrée du processus qui exécute `grep` :

```
./a.out | grep -i erreur
```

Bien-sûr, les tubes peuvent être combinés aux redirections :

```
./a.out < input.text | grep -vi erreur > sortie.text
```