Linux et Bash - Les flux de redirection

30 minutes Normal

DataScientest • com

Introduction à Linux

Les flux de redirection

Un mot sur les redirections

En utilisant `>` et `>>`, nous avons pu rediriger la **sortie standard** dans d'autres fichiers. La sortie standard est un outil qui contient ce que les commandes entrées dans l'entrée standard (**standard input**) impriment. Cependant il existe un troisième élément, l'erreur standard ou **standard error** qui affiche les messages d'erreur des commandes entrées dans l'entrée standard. Par défaut, la sortie standard et l'erreur sont imprimées dans la console.

Lorsque nous utilisons une commande dans le terminal, la réponse s'affiche dans le flux standard de sortie. Il est cependant possible de rediriger ces flux vers d'autres endroits, comme pour rediriger le flux d'erreur vers un fichier texte, par exemple.

Exécutez cette commande qui génère une erreur :

```
cat no_file

```

Si nous utilisons le mot-clef `2>` ou `2>>`, nous pouvons rediriger l'erreur standard dans un fichier :

Exécutez cette commande pour stocker les données dans le fichier `errors_file` :

```
cat no_file 2> errors_file

```

Le message d'erreur ne s'affiche pas dans la console mais se trouve dans le fichier `errors_file`.

Imprimez le contenu de ce fichier :

```
head errors_file

```

Machine status

Ubuntu Server 18.04 LTS SSD Volume Type 64-bit x86

Online

34.245.135.23

Connect

Reset Stop

correspondent respectivement aux entiers 0, 1, 2. C'est pourquoi nous retrouvons `2>` pour la sortie d'erreur, `>` est un diminutif de `1>`, c'est-à-dire de la sortie standard. Nous pouvons aussi rediriger du texte vers l'entrée standard en utilisant `<` ou `0<`.

Exécutez la commande suivante :

```
head < root_content
```

Cette commande est équivalente à `head root_content`. Il est difficile de percevoir l'utilité de l'entrée standard `<`, mais sa principale fonctionnalité est de pouvoir donner en argument le contenu d'un fichier.

Par exemple, créez le fichier `nom.py` avec les lignes suivantes :

```
name = input("Entrez votre nom\n")
print("Bonjour {} !".format(name))
```

Ensuite, écrivez Daniel dans un fichier `prenom` via la sortie standard.

Show / Hide solution

Lancez la commande ci-dessous :

```
python3 nom.py < prenom
```

Sans l'entrée standard, le programme Python vous aurait invité à saisir votre prénom, puis aurait affiché *Bonjour votre-prénom* !

Il est aussi possible d'utiliser plusieurs flux en une seule commande, par exemple avec la commande ci-dessous, nous décidons de rediriger l'erreur standard dans un fichier `log` et la sortie standard dans un fichier `fic` :

```
ca 1>fic 2>log
cat fic
cat log
```

Par ailleurs, si nous voulons que la sortie standard ainsi que l'erreur standard soient imprimées dans le même fichier, nous pouvons utiliser le mot-clef `2>&1`. Pour cela, il faut le placer à la fin de votre commande, comme ci-dessous :

```
ls -l > fic 2>&1
```

Opérateur pipe

Cet opérateur est très intéressant car il permet d'enchaîner les instructions. L'opérateur `|` prend la sortie d'une commande et l'utilise comme entrée pour une autre commande.

Exécutez la commande suivante :

```
ls / | grep bin
```

pouvons tout à fait les enchaîner :

Exécutez cette commande et essayez de comprendre ce qu'elle fait :

```
ls / | grep bin | head -n 1
```

Il existe un autre opérateur `&&` qui lui ne permet que d'enchaîner les commandes, sans lien entre elles.

```
ls -l && python3 --version && mkdir Test
```

Cette commande utilise **l'évaluation paresseuse**, c'est à dire si une des premières commandes ne fonctionne pas alors la suite ne sera pas exécutée.

Validated