ADS Lab 03 - Pipelines

Authors: Céline Roger et Grégoire Guyot Date: 18 mars 2024

Task 1: Exercices on redirection

> Run the following commands and tell where `stdout` and `stderr` are redirected to.

1. `./out > file`

- `stdout` -> fichier `file`
- `stderr` -> terminal

2. `./out 2> file`

- `stdout` -> terminal
- `stderr` -> fichier `file`

3. `./out > file 2>&1` Redirige d'abord la sortie standard `stdout` vers le fichier `file` et ensuite la sortie d'erreur `stderr` vers le même emplacement

- `stdout` -> file
- `stderr` -> file

4. `./out 2>&1 > file` Redirige d'abord la sortie d'erreur `stderr` vers le même endroit que la sortie standard `stdout` et ensuite redirige la sortie standard `stdout` vers le fichier `file`.

- `stdout` -> fichier `file`
- `stderr` -> terminal

5. `./out &> file`

- `stdout` -> fichier `file`
- `stderr` -> fichier `file`

> What do the following commands do?

```
cat /usr/share/doc/cron/README | grep -i edit
```

1. Premièrement, `cat` affiche le contenu du fichier `/usr/share/doc/cron/README`. Ensuite, `|` est l'opérateur pipe pour rediriger la sortie de la commande précédente et la transmet en tant qu'entrée dans la commande suivante: `grep -i edit` recherche le mot "edit" dans le texte transmis par `cat`. L'option `-i` spécifie une recherche insensible à la casse. Le résultat est:

```
# * documentation (don't take credit for my work), mark your changes (don't
have to go edit a couple of files... So, here's the checklist:
```

```
    Edit config.h
    Edit Makefile
```

```
./out 2>&1 | grep -i eeeee
```

2. La commande `./out 2>&1` affiche `OEOEOEOEOE` dans le terminal. Cette sortie est redirigée grâce à l'opérateur pipe comme entrée de la commande `grep -i eeeee` qui est censé rechercher la chaîne "eeeee" sans être sensible à la casse. Cette commande ne retourne rien comme résultat car la chaîne recherchée ne correspond pas à l'output de `./out`.

```
./out 2>&1 >/dev/null | grep -i eeeee
```

3. La commande `./out 2>&1` affiche `OEOEOEOEOE` dans le terminal comme la commande précédente. `>/dev/null` redirige la sortie d'erreur `stderr` vers `/dev/null`, une destination fictive. Cela signifie que tout message d'erreur produit par `./out` sera jetée et ne sera pas affichée, la chaîne `EEEEE` est donc transmise en entrée à la commande `grep` pour rechercher la chaîne "eeeee" avec une correspondance insensible à la casse. Le résultat ici est `EEEEE`.

> Write commands to perform the following tasks:

1. Produce a recursive listing, using `ls`, of files and directories in your home directory, including hidden files, in the file `/tmp/homefileslist`.

```
ls -R /home/yourusername > /tmp/homefileslist
```

---

2. Produce a (non-recursive) listing of all files in your home directory whose names end in .txt , .md or .pdf , in the file /tmp/homedocumentslist . The command must not display an error message if there are no corresponding files.

```
ls /home/osboxes/*.txt /home/osboxes/*.md /home/osboxes/*.pdf 2>/dev/null >
/tmp/homefileslist
```

Task 2: Log analysis

Download the log file (http://ads.iict.ch/ads_website.log) using curl.

```
curl http://ads.iict.ch/ads_website.log > ads_website.log
```

---

1. How many log entries are in the file?

```
cat ads_website.log | wc -l
```

Le fichier contient 2781 entrées.

---

2. How many accesses were successful (server sends back a status of 200) and how many had an error of "Not Found"(status 404)?

```
cat ads_website.log | cut -f10 | grep -c '^200$'
```

1610 entrées ont renvoyées le code de statut 200.

```
cat ads_website.log | cut -f10 | grep -c '^404$'
```

21 entrées ont renvoyées le code de statut 404.

---

3. What are the URIs that generated a "Not Found" response? Be careful in specifying the correct search criteria: avoid selecting lines that happen to have the character sequence 404 in the URI.

```
cat ads_website.log | cut -f9-10 | grep 404 | cut -f1
```

Voici les URIs qui ont générés une réponse "Not Found":

```
"GET /heigvd-ads?website HTTP/1.1"
"GET /heigvd-ads?lifecycle HTTP/1.1"
"GET /heigvd-ads?cors HTTP/1.1"
"GET /heigvd-ads?policy HTTP/1.1"
"GET /heigvd-ads?website HTTP/1.1"
"GET /heigvd-ads?lifecycle HTTP/1.1"
"GET /heigvd-ads?policy HTTP/1.1"
"GET /heigvd-ads?cors HTTP/1.1"
"GET /heigvd-ads?cors HTTP/1.1"
"GET /heigvd-ads?policy HTTP/1.1"
"GET /heigvd-ads?website HTTP/1.1"
"GET /heigvd-ads?lifecycle HTTP/1.1"
"GET /heigvd-ads?lifecycle HTTP/1.1"
```

```
"GET /heigvd-ads?cors HTTP/1.1"
"GET /heigvd-ads?cors HTTP/1.1"
"GET /heigvd-ads?policy HTTP/1.1"
"GET /heigvd-ads?lifecycle HTTP/1.1"
"GET /heigvd-ads?policy HTTP/1.1"
"GET /heigvd-ads?policy HTTP/1.1"
"GET /heigvd-ads?policy HTTP/1.1"
"GET /heigvd-ads?cors HTTP/1.1"
```

---

4. How many different days are there in the log file on which requests were made?

```
cat ads_website.log | cut -f3 | cut -d'[' -f2 | cut -d':' -f1 | sort | uniq | wc -
l
```

Il y a 21 jours différents contenus dans les logs.

---

5. How many accesses were there on 4th March 2021?

```
cat ads_website.log | grep '04/Mar/2021' | wc -l
```

Il y a eu 423 accès le 4 mars 2021.

---

6. Which are the three days with the most accesses? Hint: Create first a pipeline that produces a list of dates preceded by the count of log entries on that date.

```
cat ads_website.log | cut -f3 | cut -d'[' -f2 | cut -d':' -f1 | sort | uniq -c |
sort -nr | head -n 3
```

Les trois jours avec le plus d'accès sont les suivants:

- 13/Mar/2021 avec 898 accès
- 06/Mar/2021 avec 580 accès
- 04/Mar/2021 avec 423 accès

---

7. Which is the user agent string with the most accesses?

```
cat ads_website.log | cut -f17 | uniq -c | sort -nr | head -n 1
```

L'agent qui à le plus d'accès est `"Mozilla/5.0 (Windows NT 6.3; WOW64; rv:27.0) Gecko/20100101 Firefox/27.0"` avec 184 accès.

8. If a web site is very popular and accessed by many people the user agent strings appearing in the server's log can be used to estimate the relative market share of the users' computers and operating systems. How many accesses were done from browsers that declare that they are running on Windows, Linux and Mac OS X (use three commands)?

```
cat ads_website.log | cut -f17 | grep -c -i "Windows"
```

1751 accès ont été effectués depuis l'OS Windows.

```
cat ads_website.log | cut -f17 | grep -c -i "Linux"
```

180 accès ont été effectués depuis l'OS Linux.

```
cat ads_website.log | cut -f17 | grep -c -i "Mac"
```

693 accès ont été effectués depuis l'OS Mac.

---

9. Read the documentation for the `tee` command. Repeat the analysis of the previous question for browsers running on Windows and insert `tee` into the pipeline such that the user agent strings (including repeats) are written to a file for further analysis (the filename should be useragents.txt).

```
cat ads_website.log | cut -f17 | tee useragents.txt | grep -i "Windows"
```

Task 3: Conversion to CSV

Produce a CSV file named `accesses.csv` that contains for each day (given by its date) the number of accesses on that day. Transfer that file to your workstation and use spreadsheet software to import the CSV file. Plot the data in a graph and produce a file named `accesses.pdf`.

```
cat ads_website.log | cut -f3 | cut -d'[' -f2 | cut -d':' -f1 | sort | uniq -c |
awk '{print $2 "," $1}' > accesses.csv
```