Bash and Linux - Processes (EN)

30 minutes Normal

DataScientest • com

Introduction to Linux

Process

What is a process ?

When you run a command in the terminal, you create a process. For example, the `ls` command starts a process that prints the content of a folder and then stops. Several processes can be run at the same time. There are two types of processes: **Foreground processes** and **Background processes**.

Foreground processes can be seen as your usual programs: they require user interaction. Putting data into an Office file is an example of a foreground process. On the other hand, a background process is a process that is running without user interaction. For example, an antivirus software or a data collection script.

With a console, a foreground process will deny you possibility to interact furthermore with the console.

Take a look at the following example:

```
import time
import datetime

i = 0

while i < (60 * 20):
    file = open("data.txt", "a")
    file.write(
        datetime.datetime.now().isoformat() + '
' + str(i) + "\n"
        )
    file.close()
    i += 1
    time.sleep(1)
```

This program is going to write data into a `data.txt` file for 20 minutes.

> Paste these lines in a file called `data_collection.py` and run it

```
python3 data_collection.py
```

While this is running, we may want to do something else, for example, check that the script is running smoothly and writing the correct data: let's say we want to run `tail -n 3 data.txt`. One first way would be to open a new console and execute the command but we have other solutions.

How to interact with a foreground process

Kill the process

`ctrl + c` is commonly used to stop any process that is running.

> Stop the process using `ctrl + c` and relaunch it

> If you want to retrieve the last command you ran, you can use the up arrow on your keyboard.

Pausing the process

To pause a foreground process, you can use `ctrl + z`. The process is not stopped, only paused.

> Pause the foreground process and take a look at the last lines of `data.txt`

Right after pausing the process, you should see something like this:

```
[1]+  Stopped                 python3
data_collection.py
```

To unpause the background process, you can use `fg`.

> Unpause the process, let it run for 10 seconds and pause it again

If you check the content of the `data.txt` file, unpausing did not start over the process but it took back where it had been stopped.

Running a process in background

To run a process in the background, we can use `&` at the end of the command:

> Stop the process and run the following command

```
python3 data_collection.py &
```

You should see something like:

```
[2] 3564382
```

This is the process id of our process. You can of course put it in foreground by using `fg`. The problem is that if you want to access the console again, you need to pause the process with `ctrl + z`. To unpause the process in background, we can use `bg` and you should see:

```
[2]+ python3 data_collection.py &
```

Other process tools

htop

To get information on the running processes, you can use `htop`. You can think of it as the task manager of Windows.

> Run `htop`

You can see that there is a lot of processes that are running in the background.

> Try to find your Python process using the `Command` column. Once this is done, you can exit this interface by pressing `q`

ps

`ps`, meaning **Process Status** is a command that will return similar information to `htop` but in a static way.

> Try running the `ps` command

`ux`. You can show its manual by using `man ps`.

Process ID

In the previous parts, we have seen different ways of listing processes. We have also talked about the process ID or **PID**. This ID is important to interact with a process. For example, we can kill a process using `kill` followed by the PID. To find the PID of a command, we can use `htop` or `ps` but also `pidof`: for example, `pidof python3 data_collection.py`.

Validate