

How to execute Linux commands in the background

Linux commands are a great way to interact with the system using Terminal. However, it can sometimes take a while to complete the task at hand. This forces the user to wait a significant amount of time or create a new shell entirely.

Fortunately, you can run Linux commands in the background by following some simple methods. The rest of this article illustrates some of these methods.

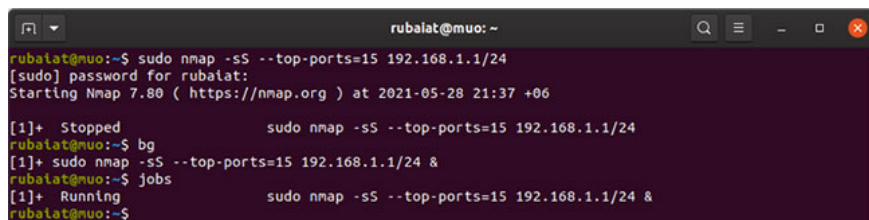
1. Add the ampersand after the command

The easiest way to run a Linux command in the background is to add the **&** symbol after the command. For example, if you start the gedit editor from your Terminal, you cannot use the shell until the editor is closed. However, when you add a symbol **&** in your command, you can use the shell immediately.

```
gedit &
```

2. Use bg to send running commands to the background

Sometimes you run a command and realize that it takes even longer to complete. You can easily send these commands to the background by pressing **Ctrl + Z** and then using the **bg** command. **Ctrl + Z** stops the running process and **bg** brings it to the background.



```
rubalat@muo:~  
rubalat@muo:~$ sudo nmap -sS --top-ports=15 192.168.1.1/24  
[sudo] password for rubalat:  
Starting Nmap 7.80 ( https://nmap.org ) at 2021-05-28 21:37 +06  
[1]+ Stopped sudo nmap -sS --top-ports=15 192.168.1.1/24  
rubalat@muo:~$ bg  
[1]+ sudo nmap -sS --top-ports=15 192.168.1.1/24 &  
rubalat@muo:~$ jobs  
[1]+ Running sudo nmap -sS --top-ports=15 192.168.1.1/24 &  
rubalat@muo:~$
```

You can see a list of all the jobs running in the background by typing the **jobs** into Terminal. Use the **fg** command to return to the running task.

3. Send commands to the background with nohup

The **nohup** command in Linux allows admins to run Terminal commands that are immune to HUP or Hang Up signals. You can run Linux commands in the background using **nohup**.

The example below runs an Nmap port scan in the background.

```
nohup sudo nmap -sS --top-ports=15 192.168.1.1/24
```

A major benefit of `nohup` is that your commands will run even if you exit the shell. Furthermore, it generates log files that record the execution process. Look for **`nohup.out`** in the current directory or inside **`$HOME`**.



```
rubalat@muo:~$ nohup sudo nmap -sS --top-ports=15 192.168.1.1/24
nohup: ignoring input and appending output to 'nohup.out'
rubalat@muo:~$ tail nohup.out
135/tcp closed msrpc
139/tcp closed netbios-ssn
143/tcp closed lnmap
443/tcp closed https
445/tcp closed microsoft-ds
3306/tcp closed mysql
3389/tcp closed ms-wbt-server
8080/tcp closed http-proxy

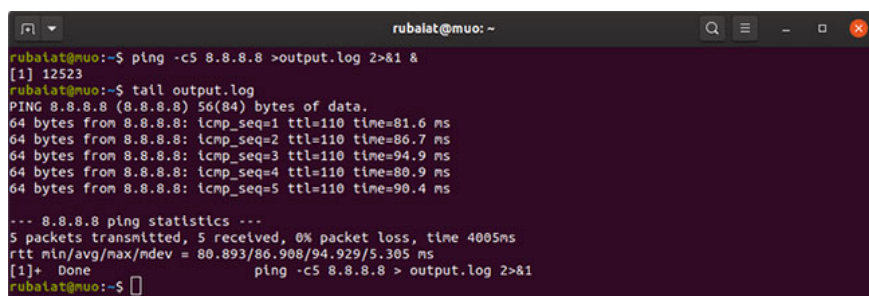
Nmap done: 256 IP addresses (3 hosts up) scanned in 5.27 seconds
rubalat@muo:~$
```

4. Run commands in the background using system redirection

You can also run commands in the background on Linux using the system redirection feature. For example, if you run the ping command below, the shell will run it in the background and immediately return a Terminal prompt:

```
ping -c5 8.8.8.8 >output.log 2>&1 &
```

Here, the output of the ping command is redirected to the **`output.log`** file. You can replace it with **`/dev/null`** if you want to get rid of the results. **`2>&1`** tells bash to redirect any errors to the same file. Finally **`&`** signals bash to run this command in the background.



```
rubalat@muo:~$ ping -c5 8.8.8.8 >output.log 2>&1 &
[1] 12523
rubalat@muo:~$ tail output.log
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data:
64 bytes from 8.8.8.8: icmp_seq=1 ttl=110 time=81.6 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=110 time=86.7 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=110 time=94.9 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=110 time=80.9 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=110 time=90.4 ms

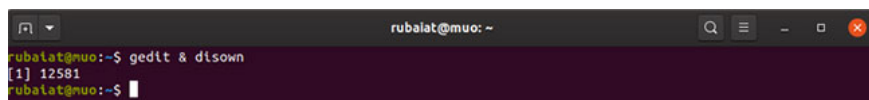
--- 8.8.8.8 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 80.893/86.908/94.929/5.305 ms
[1]+  Done                  ping -c5 8.8.8.8 > output.log 2>&1
rubalat@muo:~$
```

5. Set a Linux command to run in the background using the `disown` command

The `disown` command in Linux makes it easy to run commands in the background. First, you need to dispatch the task in the background using the **`&`** operator. Then type **`disown`** to detach it from the shell:

```
gedit & disown
```

A major advantage of `disown` is that, like `nohup`, the system will not stop the task when you close the shell or log out.

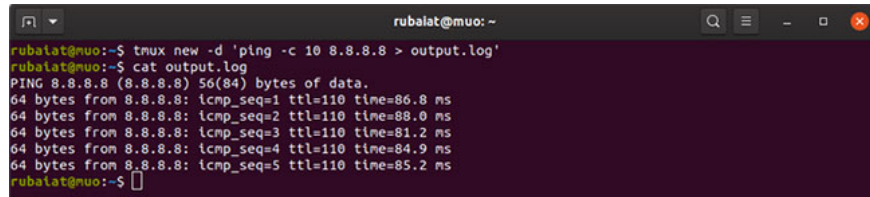


```
rubalat@muo:~$ gedit & disown
[1] 12581
rubalat@muo:~$
```

6. Run Linux commands in the background using Tmux

Tmux is a powerful multiplexer (multiplexer or MUX) that allows to run multiple Terminal sessions in a single window. Learning Tmux is a great option for those who are new to the tool. Tmux makes it easy to run commands in the background on Linux.

```
tmux new -d 'ping -c 10 8.8.8.8 > output.log'
```

A terminal window titled 'rubalat@muo: ~' showing the execution of a tmux command. The user enters 'tmux new -d 'ping -c 10 8.8.8.8 > output.log''. The prompt returns to '\$'. The user then enters 'cat output.log'. The terminal displays the output of the ping command: 'PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.', followed by five lines of ping results, each showing '64 bytes from 8.8.8.8: icmp_seq=X ttl=110 time=X.X ms' for X from 1 to 5. The prompt returns to '\$'.

When you run the tmux command above, it will do the ping in a separate shell and keep it in the background. You can execute any Linux command in the background using this method.

You finished reading the article "**How to execute Linux commands in the background**" edited by the [TipsMake](#) team. We hope this article has provided you with many useful tech tips and tricks. You can search for similar articles on tips and guides. Thank you for reading and for following us regularly.