Running programs in the background from terminal

128

85

How do I run a program in the background of a shell, with the ability to close the shell while leaving the program running? Lets say my UI is having problems or for some reason, I need to boot up a program from the terminal window, say, `nm-applet` :

```
nm-applet
```

When it's started, it occupies the foreground of the terminal window.

Is there any simple way to run the program in the background without needing to leave the terminal open or have it occupy the whole terminal?

On that note, I did find a way to run programs from the terminal and have it allow for other inputs, by appending an ampersand (`&`) to the command as such:

```
nm-applet &
```

But this isn't much use as any processes started in the terminal are killed once the terminal is closed.

command-line bash process console

edited Jun 5 '14 at 23:18
Nicolas Barbulesco
103 3

asked Feb 22 '12 at 0:07
OVERTONE
1,022 3 9 15

7 nohup 'command' & This seems to work. Any problems with this? – OVERTONE Feb 22 '12 at 0:11

9 Answers

167

I've recently come to like `setsid` . It starts off looking like you're just running something from the terminal but you can disconnect (close the terminal) and it just keeps going.

This is because the command actually forks out and while the input comes through to the current terminal, it's owned by a completely different parent (that remains alive after you close the terminal).

An example:

```
setsid gnome-calculator
```

I'm also quite partial to `disown` which can be used to separate a process from the current tree. You use it in conjunction with the backgrounding ampersand:

```
gnome-calculator & disown
```

I also just learnt about spawning subshells with parenthesis. This simple method works:

```
(gnome-calculator &)
```

And of course there's `nohup` as you mentioned. I'm not wild about `nohup` because it has a tendency to write to `~/nohup.out` without me asking it to. If you rely on that, it might be for you.

```
nohup gnome-calculator
```

And for the longer-term processes, there are things like `screen` and other virtual terminal-muxers that keep sessions alive between connections. These probably don't really apply to you because you just want temporary access to the terminal output, but if you wanted to *go back* some time later and view the latest terminal activity, screen would probably be your best choice.

The internet is full of `screen` tutorials but here's a simple quick-start:

- http://thingsilearned.com/2009/05/26/gnu-screen-super-basic-tutorial/

edited Feb 22 '12 at 1:04

answered Feb 22 '12 at 0:41
Oli ♦
225k 91 568 770

Oli, in the case of DISPLAY=:0 unity --replace which of the above would you use and why? – nutty about natty Feb 7 '13 at 12:40

4 @nuttyaboutnatty wrap the whole lot in its own shell session: `sh -c "sleep 10s && cvlc '/home/omm.ogg'" & disown` . That's pretty much my solution for everything to make sure it forks out properly. – Oli ♦ Mar 12 '13 at 14:56

1 @nuttyaboutnatty That's what I mean but you wouldn't end up with two instances running because `--replace` does what it sounds like and actually ends the old instance. Running it the second time from within the xsession would end the copy bound to TTYn and would let you use that in the future if you needed it. There is probably a better way of doing it but just what comes to my mind each time I need it. – Oli ♦ Mar 12 '13 at 15:25

1 Perfect, the one that worked for me was the subshell '(gnome-calculator &)'. I use it to launch a mono console app in the background. With the other techniques, the mono app would crash instantly. – Nikolaos Georgiou Oct 28 '13 at 20:55

2 **Note to future users**: creation of `nohup.out` can be suppressed with redirecting both stdout and sterr to `/dev/null` like so: `nohup firefox &> /dev/null &` – Sergiy Kolodyazhnyy Aug 7 '16 at 18:13

50

From a terminal, run

```
nm-applet &
```

But do NOT close the terminal yourself. That is, do not hit the X-button to close, and do not use File -> Exit from its menubar. If you close the terminal that way, it will send a HUP (Hang UP) signal to the bash running within, which in turn will send the HUP signal to all its children (which is why nohup works in this case).

Instead, exit the shell by running `exit` or hitting Ctrl+D. bash will then disown its children, then exit, leaving the background processes still running. And when bash exits, the terminal has lost its child process, so it will close too.

Doing it all at once:

```
nm-applet & exit
```

edited Feb 23 '12 at 20:01

answered Feb 23 '12 at 11:45
geirha
31.8k 9 58 60

That's amazing! Never knew that and you seem to be the only person to mention this. Definitly that's my favorite approach when in a Desktop Environment. Too bad I can't upvote more ;p – 7hi4g0 Jan 24 '15 at 6:01

I tried this with a different application and it failed: `matlab & exit` Running as two separate commands worked well: `matlab & exit` – MattKelly Nov 10 '16 at 20:00

I was wondering why OP said closing the terminal caused his background programs to crash. I've never tried closing a terminal by the "x" button on the window. – user1717828 Mar 29 '17 at 12:51

the x button is kill it doesn't cleanly exit, the exit command does. you could remap the x button to a exit command instead... – Tim Baker Feb 2 '18 at 7:35

13

As you pointed out, you can run

```
nohup nm-applet &
```

to ignore the end signal when closing the terminal. No problem with that.

answered Feb 22 '12 at 0:17
desgua
28k 8 83 113

Any other alternatives? Just for knowings sake, not for anything else – OVERTONE Feb 22 '12 at 0:38

At wikipedia (en.wikipedia.org/wiki/Nohup) there is a suggestion to use `echo command | at now` which I couldn't get it to work. – desgua Feb 22 '12 at 0:45

2 I wonder how it works from a the UI when you double click an icon or program. – OVERTONE Feb 23 '12 at 10:22

11

One thing that many other answers are missing is how to detach a running process that currently blocks the shell. In most terminals and shells, Ctrl+Z will halt the running process and bring you back to an input prompt. Then, you can issue

```
bg
```

to send the running process into the background. Issue

```
fg
```

instead to put the running process back into the foreground.

EDIT: More detail in this answer I discovered later.

edited Apr 13 '17 at 12:23
Community ♦
1

answered Mar 31 '15 at 14:37
krlmlr
2,297 3 25 50

7

Use `(exec PROGRAM &> /dev/null &)` to allow PID of subshell to be taken over by `PROGRAM`. I've tested this approach multiple times with several different programs. Closing the original terminal has no affect on the newly-spawned program

Small demo:

```
$ # this is before running
$ (exec firefox &> /dev/null &)
$ # and look, we still in side the terminal and can continue working
```

edited Aug 7 '16 at 18:37

answered Nov 30 '15 at 11:35
Sergiy Kolodyazhnyy
75.8k 9 158 333

Hmmm `(exec firefox)` hangs the terminal until I exit Firefox...and closing the terminal SIGHUPs the shell and the subshell – kos Nov 30 '15 at 11:48

I've edited my answer. Tested it with several other programs. I've not observed firefox or other program hanging the controlling terminal in this version – Sergiy Kolodyazhnyy Aug 7 '16 at 18:39

works well for me – Zanna Aug 21 '16 at 19:12

2

I can recommend the byobu terminal. You can easily detach your process by pressing the F6 key.

answered Mar 5 '13 at 0:07
speter
416 3 17

1

Although there are good answers above, I would like to give my 2 cents on how I use MATLAB in background.

```
sudo -b matlab
```

Probably, unrelated but there is wonderful website that you can use to explain shell commands. http://explainshell.com/explain/8/sudo

answered Jun 28 '16 at 14:39
user3342981
11 1

0

I dont know its the right way but i just start another session while leaving the previous one allone. For example i ran a simple web server on my raspberry. the web.py one. then i start a new session while leaving it alone. thats it. it also is wuite useful since you are still updated even though you are workin on the other session.

answered Jul 5 '16 at 22:15
leo
1

2 Welcome to Ask Ubuntu! I recommend editing this answer to expand it with specific details about how to do this. (See also How do I write a good answer? for general advice about what sorts of answers are considered most valuable on Ask Ubuntu.) – David Foerster Jul 5 '16 at 22:51

0

In case KDE is used, you can also use `kstart`, which will start your program detached from the terminal. It also makes sure that the KDE environment is correctly setup for the command. (See `kstart.cpp` source code for reference. As you see from there, it uses `KProcess::startDetached`, and `KProcess` is derived from `QProcess`, and `startDetached` starts a new process, and detaches from it.)

Similar is also `kde-open` or `xdg-open` or `gnome-open`.

edited Mar 5 at 15:42

answered Mar 5 at 10:00
Albert
544 4 12 25

As this stands, it looks almost more like a comment than an answer. Please edit to expand. Don't just give a one-liner; explain why you right, ideally with citations. Answers without explanation are subject to removal. – anonymous2 Mar 5 at 13:49

---

protected by Anwar Aug 7 '16 at 18:01

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site (the association bonus does not count).

Would you like to answer one of these unanswered questions instead?