Unix & Linux Stack Exchange is a question and answer site for users of Linux, FreeBSD and other Un*x-like operating systems. It only takes a minute to sign up.

Sign up to join this community

Anybody can ask a question

Anybody can answer

The best answers are voted up and rise to the top

UNIX & LINUX

Difference between Login Shell and Non-Login Shell?

Asked 7 years, 10 months ago Active 9 months ago Viewed 187k times

341

I understand the basic difference between an interactive shell and a non-interactive shell. But what exactly differentiates a login shell from a non-login shell?

Can you give examples for uses of a ***non-login interactive*** shell?

shell login

215

asked May 8 '12 at 20:57

lgorio
4,739 6 16 11

52 I think the question is better phrased as "*Why* do/should we care to differentiate login and non-login shells?" Many places on the web already tell us *what* are the differences, in terms of what startup files each read; but none of them seems to answer the "why" in a satisfactory and convincing way. Example use cases where you definitely *do not want* one or the other behaviour would be great. – Kal Apr 15 '13 at 3:49

5 @Kal This would have to be a different question, since no answer here actually covers that. Edit : Actually, here it is : WHY a **login** shell over a **non-login** shell?. – Skippy le Grand Gourou Aug 14 '18 at 13:15

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

334

A login shell is the first process that executes under your user ID when you log in for an interactive session. The login process tells the shell to behave as a login shell with a convention: passing argument 0, which is normally the name of the shell executable, with a `-` character prepended (e.g. `-bash` whereas it would normally be `bash`. Login shells typically read a file that does things like setting environment variables: `/etc/profile` and `~/.profile` for the traditional Bourne shell, `~/.bash_profile` additionally for bash†, `/etc/zprofile` and `~/.zprofile` for zsh†, `/etc/csh.login` and `~/.login` for csh, etc.

When you log in on a text console, or through SSH, or with `su -`, you get an **interactive login** shell. When you log in in graphical mode (on an X display manager), you don't get a login shell, instead you get a session manager or a window manager.

It's rare to run a **non-interactive login** shell, but some X settings do that when you log in with a display manager, so as to arrange to read the profile files. Other settings (this depends on the distribution and on the display manager) read `/etc/profile` and `~/.profile` explicitly, or don't read them. Another way to get a non-interactive login shell is to log in remotely with a command passed through standard input which is not a terminal, e.g. `ssh example.com <my-script-which-is-stored-locally` (as opposed to `ssh example.com my-script-which-is-on-the-remote-machine`, which runs a non-interactive, non-login shell).

When you start a shell in a terminal in an existing session (screen, X terminal, Emacs terminal buffer, a shell inside another, etc.), you get an **interactive, non-login** shell. That shell might read a shell configuration file (`~/.bashrc` for bash invoked as `bash`, `/etc/zshrc` and `~/.zshrc` for zsh, `/etc/csh.cshrc` and `~/.cshrc` for csh, the file indicated by the `ENV` variable for POSIX/XSI-compliant shells such as dash, ksh, and bash when invoked as `sh`, `$ENV` if set and `~/.mkshrc` for mksh, etc.).

When a shell runs a script or a command passed on its command line, it's a **non-interactive, non-login** shell. Such shells run all the time: it's very common that when a program calls another program, it really runs a tiny script in a shell to invoke that other program. Some shells read a startup file in this case (bash runs the file indicated by the `BASH_ENV` variable, zsh runs `/etc/zshenv` and `~/.zshenv`), but this is risky: the shell can be invoked in all sorts of contexts, and there's hardly anything you can do that might not break something.

† I'm simplifying a little, see the manual for the gory details.

edited Feb 3 '17 at 12:17

answered Sep 1 '12 at 2:07

Gilles 'SO- stop being evil'

609k 147 1244 1758

2 Could you give example how to run `bash` as a non-interactive login shell? – Piotr Dobrogost Jun 16 '13 at 8:47

13 @PiotrDobrogost `echo $- | bash -lx` – Gilles 'SO- stop being evil' Jun 16 '13 at 12:11

1 I don't know if this is true in general, but I want to note that when I open a new terminal (on osx using default settings), I get a login shell even though I never type in my username or password. – Kevin Wheeler Aug 28 '15 at 22:55

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

Gilles 'SO- stop being evil' Aug 28 '15 at 23:01

2 @IAmJulianAcosta If `FOO` is an environment variable (i.e. `.profile` contains `export FOO=something`) then it's available to all subprocesses, including `foo.sh`. If you change `.profile` to `export FOO=something_else` then `./foo.sh` will still print `something` until the next time you log in. – Gilles 'SO- stop being evil' Oct 6 '16 at 15:13

53

To tell if you are in a login shell:

```
prompt> echo $0
-bash # "-" is the first character. Therefore, this is a login shell.

prompt> echo $0
bash # "-" is NOT the first character. This is NOT a login shell.
```

In Bash, you can also use `shopt login_shell` :

```
prompt> shopt login_shell
login_shell     off
```

(or `on` in a login shell).

Information can be found in `man bash` (search for Invocation). Here is an excerpt:

> A login shell is one whose first character of argument zero is a -, or one started with the --login option.

You can test this yourself. Anytime you SSH, you are using a login shell. For Example:

```
prompt> ssh user@localhost
user@localhost's password:
prompt> echo $0
-bash
```

The importance of using a login shell is that any settings in `/home/user/.bash_profile` will get executed. Here is a little more information if you are interested (from `man bash`)

> "When bash is invoked as an interactive login shell, or as a non-interactive shell with the --login option, it first reads and executes commands from the file /etc/profile, if that file exists. After reading that file, it looks for `~/.bash_profile`, `~/.bash_login`, and `~/.profile`, in that order, and reads and executes commands from the first one that exists and is readable. The --noprofile option may be used when the shell is started to inhibit this behavior."

edited May 14 '19 at 16:37

Stephen Kitt
230k 30 552 628

answered Oct 21 '15 at 14:46

Timothy Pulliam
2,003 2 14 29

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

23

In a login shell, `argv[0][0] == '-'` . This is how it knows it's a login shell.

And then in some situations it behaves differently depending on its "login shell" status. E.g. a shell, that is not a login shell, would not execute a "logout" command.

edited May 9 '12 at 4:52

Mat
42.9k 9 133 134

answered May 8 '12 at 22:57

BOPOHOK
231 1 2

5 According to `man bash` , with emphasis added, *"A login shell is one whose first character of argument zero is a -, **or one started with the --login option.**"* – Wildcard Jan 23 '17 at 11:49

18

A shell started in a new terminal in a GUI would be an interactive non-login shell. It would source your .bashrc, but not your .profile, for example.

answered May 8 '12 at 22:20

Julian
479 2 5

5

I'll elaborate on the great answer by Gilles, combined with Timothy's method for checking login shell type.

If you like to see things for yourself, try the snippets and scenarios bellow.

Checking whether shell is (non-)interactive

```
if tty -s; then echo 'This is interactive shell.'; else echo 'This is non-interactive
shell.'; fi
```

Checking whether shell is (non-)login

If output of `echo $0` starts with `-` , it's login shell (`echo $0` output example: `-bash`). Otherwise it's non-login shell (`echo $0` output example: `bash`).

```
if echo $0 | grep -e ^\- 2>&1>/dev/null; then echo "This is login shell."; else echo
"This is non-login shell."; fi;
```

Let's combine the two above together to get both pieces of information at once:

```
THIS_SHELL_INTERACTIVE_TYPE='non-interactive';
THIS_SHELL_LOGIN_TYPE='non-login';
if tty -s; then THIS_SHELL_INTERACTIVE_TYPE='interactive'; fi;
if echo $0 | grep -e ^\- 2>&1>/dev/null; then THIS_SHELL_LOGIN_TYPE='login'; fi;
echo "$THIS_SHELL_INTERACTIVE_TYPE/$THIS_SHELL_LOGIN_TYPE"
```

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

Typical SSH session without special options

```
ssh ubuntu@34.247.105.87
Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.4.0-1083-aws x86_64)

ubuntu@ip-172-31-0-70:~$ THIS_SHELL_INTERACTIVE_TYPE='non-interactive';
ubuntu@ip-172-31-0-70:~$ THIS_SHELL_LOGIN_TYPE='non-login';
ubuntu@ip-172-31-0-70:~$ if tty -s; then THIS_SHELL_INTERACTIVE_TYPE='interactive'; fi;
ubuntu@ip-172-31-0-70:~$ if echo $0 | grep -e ^\- 2>&1>/dev/null; then
THIS_SHELL_LOGIN_TYPE='login'; fi;
ubuntu@ip-172-31-0-70:~$ echo "$THIS_SHELL_INTERACTIVE_TYPE/$THIS_SHELL_LOGIN_TYPE"

interactive/login
```

Running script or executing explicitly via new shell

```
ubuntu@ip-172-31-0-70:~$  bash -c 'THIS_SHELL_INTERACTIVE_TYPE='non-interactive';
THIS_SHELL_LOGIN_TYPE='non-login'; if tty -s; then
THIS_SHELL_INTERACTIVE_TYPE='interactive'; fi; if echo $0 | grep -e ^\- 2>&1>/dev/null;
then THIS_SHELL_LOGIN_TYPE='login'; fi;
echo "$THIS_SHELL_INTERACTIVE_TYPE/$THIS_SHELL_LOGIN_TYPE"'

interactive/non-login
```

Running local script remotely

```
ssh ubuntu@34.247.105.87 < checkmy.sh
Pseudo-terminal will not be allocated because stdin is not a terminal.
Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.4.0-1083-aws x86_64)

non-interactive/login
```

Running a command over ssh remotely

```
ssh ubuntu@34.247.105.87 'THIS_SHELL_INTERACTIVE_TYPE='non-interactive';
THIS_SHELL_LOGIN_TYPE='non-login'; if tty -s; then
THIS_SHELL_INTERACTIVE_TYPE='interactive'; fi; if echo $0 | grep -e ^\- 2>&1>/dev/null;
then THIS_SHELL_LOGIN_TYPE='login'; fi; echo
"$THIS_SHELL_INTERACTIVE_TYPE/$THIS_SHELL_LOGIN_TYPE"'

non-interactive/non-login
```

Running a command over ssh remotely with `-t` switch

You can explicitly request interactive shell when you want to run command remotely via ssh by using `-t` switch.

```
ssh ubuntu@34.247.105.87 -t 'THIS_SHELL_INTERACTIVE_TYPE='non-interactive';
THIS_SHELL_LOGIN_TYPE='non-login'; if tty -s; then
THIS_SHELL_INTERACTIVE_TYPE='interactive'; fi; if echo $0 | grep -e ^\- 2>&1>/dev/null;
then THIS_SHELL_LOGIN_TYPE='login'; fi; echo
```

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

Note: On topic why running command remotely is not `login shell` more info here.

answered Jun 3 '19 at 21:09

Patrik Stas
151 1 3

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.