

List of Bash online-tutorials

List of Bash online-tutorials on bash hacker wiki !

<http://wiki.bash-hackers.org/scripting/tutoriallist>

<http://archive.is/YzvuQ>

Here's a list of some Bash tutorials.

The primary purpose of that list is to lead beginners to **good tutorials** and not to the wrong ones. However, the secondary purpose is to provide information to increase quality of the linked tutorials.

My experience shows that nobody is interested when you "just send a mail to the author", even if he links a big "contact me" in his article(s). This is another try of influencing the Bash world.

List

This is a test for the data plugin. For now, please use the next section.




Short name	↓ Recommendation Index
Bash guide on Greg's wiki	90



Recommendations





Note that these recommendations are my personal opinion. Please **contact** me

- if you have reviews or new sites
- if you're not okay with a recommendation
- if you're the author of a mentioned site (remove link, copyright, discussion, ...)
- etc...

The recommendation-indicator "REC" used below is a number between 1 and 10 visualized as a bar:

-  Not recommended to read, at best, don't click the link.
- ...
-  Use it with care!
- ...
-  **The perfect godlike tutorial** (I doubt I'll ever find it)

Name (Links to review below)	Weblink	REC indicator	Comments
Bash guide on Greg's wiki	click		This guide teaches modern stuff and good practises. I recommend learning from it. It was written by the guys in #bashIRC channel on Freenode (mainly 1hunath), because there are so many bad tutorials out there.
Steve Parker's shell scripting guide	click		Very good (not only Bash) shell scripting guide. Teaches good practices, gives background information.

Name (Links to review below)	Weblink	REC indicator	Comments
Bash Guide for Beginners (review)	click		Good introduction that really requires no previous knowledge, also covers the most important unix utilities
Advanced Bash Scripting Guide (ABS) (review)	click		Has a lot of information that is hard to find, is outdated and often unsafe. To be avoided until you can filter out the good stuff.
IBM developerWorks "Bash by example"	click(1) click(2) click(3)		Doesn't teach outdated stuff, doesn't tell you wrong things. A good start, though not that detailed.
Deadman's	click		Focus isn't scripting per se. Focus is interactive use and increasing the productivity on the prompt. Teaches some nice features.
Bash Shell Programming in Linux (P. Lutus)	click		Good start. Though there are small bugs.
BASH Help at hypexr.org (review)	click		Shows you some nice stuff and links to other resources. Not a tutorial to learn Bash, though.
Bash Programming Introduction HowTo (TLDP) (review)	click		Absolute crap. Many syntax errors alone.

Name (Links to review below)	Weblink	REC indicator	Comments
Quick guide (review)	click		Usable as a start. Doesn't teach wrong stuff, shows you good practices.
LinuxCommand.org: Writing shell scripts. (review) incomplete, thus ranking isn't complete	click		Practise oriented, some mistakes/flaws, but sadly it stops in the middle
Linux Shell Scripting Tutorial v2.0 (review)	click		currently reviewing (the tutorial is also under development)
linuxconfig.org Bash Scripting Tutorial (review)	click		Teaches many outdated, unstable, undetailed stuff. You won't learn scripting from there.
Beginner Linux Tutorial	click		A comprehensive introduction to the Linux Command Line including ample examples to make learning easy.
Beginner Bash Scripting Tutorial	click		A beginners guide to Bash scripting under Linux.
Linuxcommand.org: The Linux Command Line	click		A beginners guide to using Bash shell, basic unix utilities, and shell scripting. Shell scripting part is not so good. But good introduction on how to use various utilities in Bash.

Detailed reviews

linuxconfig.org Bash Scripting Tutorial

Article link: http://www.linuxconfig.org/Bash_scripting_Tutorial Discussion link: http://www.linuxconfig.org/Talk:Bash_scripting_Tutorial UPDATE: Discussion page is gone.

Though the basic idea is nice, using flash terminal sessions and screenshots, there are many bugs or bad constructs.

Some stuff I didn't like there:

- uses external, unreliable command `which` instead of builtin `type -p` to determinate the location of a program
- lacks [sane quoting](#)
- uses `function SOMENAME` keyword instead of the common POSIX-compatible variant `SOMENAME ()` to [define a function](#)
- uses backticks instead of `$ (...)` for [command substitution](#)
- incorrectly uses an additional array to store [positional parameters](#), disregarding that `$@` already is array-like
- uses `echo -e` and the historical control character `\c` instead of modern [The printf command](#)
- uses `for a in seq 1 10`` instead of a [C-like counter loop] (https://archive.is/o/YzvuQ/wiki.bash-hackers.org/syntax/ccmd/c_for) `for ((a=1; a <= 10; a++))``
- the `if/else` stuff looks as if the `test` (or `[...]`) command is the only thing Bash can execute and check
- a `for` loop example that **will** explode on [word-splitting](#)
- arithmetic tests (the `while/until` loop examples) using the historical "old way" with `test`, not [modern arithmetic components](#)
- useless [quoting](#) of one-word strings (not that it hurts per se, but it shows that the author has no clue when to use quotes)
- a weird construct I don't understand (example for stdout/err redirection): `grep -r hda6 * . 1>&2 stderr.txt`

twkm commented some things on their http://www.linuxconfig.org/Talk:Bash_scripting_Tutorial, I linked **this article** there. UPDATE: Discussion page is gone.

Overall, if the author doesn't change the article, it's unusable from my point of view. At least unusable to teach sane Bash scripting.

UPDATE: Discussion is available directly below the article. I linked this page, but waiting for moderator approval.

Quick Guide

Article link: <http://www.panix.com/~elflord/unix/bash-tute.html> Discussion Link: **not available**

This article is usable as basic introduction into the Bash world. It doesn't teach you wrong things, and it uses correct syntax and explanations nearly everywhere. However, it's not usable as complete learning tutorial - but that is not the goal.

One point (I **have** to criticize **something** 😊):

- the article says that there are no [C-styled for-loops](#) in Bash, this is wrong (maybe the article is for a very old (pre 2.05b) Bash version)

Bash Programming Introduction HowTo

Article link: <http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO.html> Discussion link: **not available**; EMail: mikkey (AT) dynamo.com.ar

A few points:

- uses `function SOMENAME` instead of more common and most portable `SOMENAME ()` to [define functions](#)
- wrong description of the [if clause](#) (says "the base [...] is `if [expression];`", which is wrong)
- a [for-loop example](#) will explode due to [word splitting](#) (`for i in $(ls) - evil!`)
- the mentioned C-like for-loop is not the [real C-like for-loop](#), it just calls `seq(1) +`

- Many syntax errors: The examples won't even parse correctly

I like the style this article is written in. If the points are fixed, it could be a really usable starting point for newbies. But at the moment it's unusable

BASH Help at hypexr.org

Article link: http://www.hypexr.org/bash_tutorial.php Discussion link: **not available**; EMail: scott (AT) hypexr.org

The article is usable to step into the shell world. It's not a tutorial per se, it will tell you some nice CLI-specific things like readline or completion.

Only one point:

- confusing description of the dotfiles

Advanced Bash Scripting guide (ABS)

Article link: <http://tldp.org/LDP/abs/html/> Discussion link: **not available**; EMail: thegrendel (AT) theriver.com

The only big problem I see with the ABS is, that the name doesn't match the reality. It doesn't teach **advanced** techniques. That **doesn't mean** the guide isn't good! It's one of the biggest, most complete and interesting Bash guides I've seen.

I don't want to write every point here that disturbs me. In general it's not that important since the mistakes or wrong assumptions it makes are minimal. Also I noticed that mistakes in example scripts vanish with time, the author polishes his work. Thanks Mr. Cooper.

The ABS is definitely worth reading to step deeper into the Bash commandline world of Linux (since many Linux-specific examples are there, they're unusable for Unices).

Bash Guide for Beginners

Article link: <http://tldp.org/LDP/Bash-Beginners-Guide/html/> Discussion link: **not available**

Good introduction to bash and shell scripting, the guide is fairly complete and requires almost no previous knowledge other than be able to type some commands in a shell.

Some advice is a bit strange or outdated "Real Programmers - Most programmers will prefer to use the test built-in command" "Wherever possible, Bash users should try to use the syntax with angular brackets:(\$[])" but all in all a nice tutorial to get a good overview of shell programming starting from 0.

Linux Shell Scripting Tutorial v2.0

Article link: http://bash.cyberciti.biz/guide/Main_Page Discussion link: **use the individual MediaWiki discussion pages**

Additional problem: The author rates his shell skills as "9 of 10" in his CV. After reading this tutorial personally I'd rate him 3/10

This guide has some big problems. It seems to cover a lot of material but has some pretty nasty issues, too. Examples:

- ~~When showing how to echo variables, it shows `echo $var` and `echo ${var}` mostly without quoting.~~
- Lots of 'test' and not much '['. Not really a "problem" - more a style thing. But...
- test and [and great, but what happened to [[? And test == is not used to check if two strings are equal. You use =.
- ~~How to deal with case insensitive matching mentions converting to lowercase with tr and doing a pattern like `[tT][aA][rR]`. I propose a third solution: `shopt -o nomasematch`~~
- The for loop examples reads like a how-not-to. for i in <don't do this>: <
(*ls/tmp/**) >, < **12345** >, <files>, <
** >. Don't parse ls. Use 1..5. Use an array for a list of files. Use quotes (and @).*
- The infinite-while-loop example has him reimplementing the bash select builtin. Not bad for an exercise, but it ought to acknowledge that bash does it better.
- the description for compound commands is wrong: It shows 3 compound commands (grouping with and without subshell) where two of the examples are the same and the reason given for using () applies equally to { }
- ~~the function definition focuses on the "function" keyword, which is the worst way~~

- The initial example page for pipes seems like a good list of examples of parsing things-you-do-NOT-want-to-parse
- ~~the page explaining special parameters misses \$@, which is very important~~
- ~~the page(s) about shell variables and environment doesn't even mention "environment" so far some improvement.~~
- ~~backticks are **not** a quoting mechanism, they are a substitution mechanism. but that doesn't matter since he doesn't mention all quoting mechanisms anyways~~ some improvement, but the quoting page is still misleading
- ~~another hint that he doesn't know what he's talking about: It's said that command substitution is allowed inside backquotes (where backquotes are a quoting mechanism!) - backquotes are **command substitution!**~~
- etc. etc. etc. (nearly endless list, sadly)

Conclusion

Beside all the bashing (sorry!) above: I think the problem is the following, the author did a lot using the shell, and he knows many things. But he doesn't know and/or understand the underlying concepts of most of the material covered. This - in my personal opinion - disqualifies him as the author of a guide/tutorial for shell scripting.

But

The tutorial is under development. It improves here and there. But the code style and robustness problems still remain.

UPDATE: Over time, the author fixed a lot of things and created new chapters. From time to time, I'll visit again and re-check it.

LinuxCommand.org: Writing shell scripts.

Article link: http://linuxcommand.org/writing_shell_scripts.php Discussion link: **not available**

Bad:

- Difference between startup files `.bash_profile` and `.bashrc` is wrong ("Though placing your aliases and shell functions in your `.bash_profile` will work, it is not

considered good form.")

- Reserved words (for the parser) are **not** disallowed as shell variable names (if the variable name is a reserved word, the reserved word takes precedence)
- It suggests the `which` command, which might be popular here and there, but should not be used for various reasons
- It says, variables with `UPPERCASE` names are constants, this might be a good style, but it's not a programming feature. Real "constants" are made from read-only variables
- Common mistake it shares with many other tutorials: The ANSI C escapes are not recognized by (normal) quoting mechanisms, it's the `echo` command that interprets them! On the other hand, it misses the quoting style that makes the shell interpret ANSI C escape sequences.
- It suggests the `function` keyword to define a function, which is correct, but bad style and unportable
- It could use `getopts` for positional parameters, at least as an "advanced example"
- It uses special parameters that represent positional parameters (`$@`) unquoted, and thus unsafe or buggy
- Slightly wrong: `SIGKILL` and the process stop signal can't be trapped. But that shouldn't really matter for daily shell coding
- "Creating safe temporary files", which is about creating a unique name, should mention tools like `mktemp` or the like
- It's far from complete, it "suddenly stops". Maybe it's still under development?

Good:

- Practise oriented
- It explains the behaviour of unquoted variable arguments to the `test` command in a nice, understandable way
- It does not use `expr` or `let` for arithmetic, good