Bash Scripting Cheat Sheet

© Erik E. Lorenz, May 23, 2014

Internal Files and Directories

~/.bashrc	user-specific constants. Sourced when bash starts
~/.bash_profile	similar to ~/.bashrc
~/.bash_history	list of previous bash commands
~/.bash_logout	sourced on bash logout
/bin/bash	location of the bash executable

Terms

term	description	examples
user	a user account of the system	root e.lorenz
file	regular file	~/file.txt code/asd/src/main.cpp
dir	regular directory	~/directory /etc
cmd	any command	echo date
host	url or ip of a remote machine	enssim.etit.tu-chemnitz.de 134.109.52.89
port	a network port for interaction	22
	with a specific program	31159
url	fully qualified url	http://*host*:port/*dir*/*file*
pid	process id	18738
alias	command alias	alias ssk='ssh enssim'
export	define an environment variable	export PATH=~/bin:$PATH
source	read a script in the current bash	. ~/.bashrc
	instance, e.g. for exporting	source ~/.bashrc

Useful Environment Variables

$HOME	home directory. Usually /home/*user*
~	same as $HOME
$USER	name of the current *user*
$EUID, $UID	(effective) user id
$PATH	colon-separated list of search directories for binaries
$LIBRARY_PATH	search paths for .so and .a files at compile time
$LD_LIBRARY_PATH	search paths for .so and .a files at run time
$PWD	current working directory
$EDITOR	preferred command line text editor, e.g. vim
$IFS	internal field separator, e.g. for for...in constructs
$LINENO	current line number in a script, e.g. for debugging
$COLUMNS	width of the terminal
$LINES	height of the terminal
$LANG	preferred language of the user
$SHELL	path of the shell-executable. Should be /bin/bash
$SHLVL	shell nesting level on the current machine
$$	*pid* of the current script or bash instance
$PPID	*pid* of the parent process
$!	*pid* of the last child process (see Forking)
!!	previous command
!$	last argument of the previous command
!^	first arguments of the previous command
!:1, !:2, ...	arguments of the last command
!:1-	all arguments of the last command
$@	array of arguments to a script or function
$0	command used to run this script or bash instance
$1, $2, ... $9	first, second, ... ninth argument

Cheat Sheet Color Coding

cmd	Most frequent commands
cmd	Usually not harmful
cmd	deletes data or annoys others. May require root permissions

Debugging

set -x	print every command which is executed
trap read debug	confirm every command with [Enter]

Hotkeys

Tab	autocomplete the current command or path
Ctrl+I	same as Tab
Alt+*	insert all possible completions
Ctrl+C	kill the current command
Ctrl+D	exit the current shell
Ctrl+X Ctrl+E	write the next command in your $EDITOR
Ctrl+R	reverse-search your history for a command
Ctrl+Z	suspend the process. Resume with %
Ctrl+L	clear the terminal. Similar to clear
Ctrl+U	clear the line before the cursor
Ctrl+K	clear the line after the cursor
Alt+F	move forward one word
Alt+B	move backward one word
Alt+D	delete next word
Alt+Backspace	delete previous word

Redirecting Standard I/O

cmd > *file*	write output to a new *file*
cmd | tee *file*	both print and write to a file
cmd >> *file*	append output to *file*
cmd 2> *file*	write errors to *file*
cmd 2>&1	redirect errors to standard output
cmd &>/dev/null	discard all output
cmd < *file*	read input from *file*
cmd << EOF	read input from command line until word "EOF"
cmd <<< *cmd*	read input from the rest of the line
cmd | *cmd*	redirect output from the first to the second *cmd*

Child Processes and Forking

cmd &	run *cmd* in the background as a child process (fork)
(*cmd* &); exit	run *cmd* in the background and exit
set -m	run func when a child exits
trap func CHLD	
wait	wait for all child processes to exit

Automatic String Expansion

.bash*	.bash_history .bash_logout .bash_profile .bashrc
.bash_???????	.bash_history .bash_profile
{7..11}	7 8 9 10 11
{07..11}	07 08 09 10 11
{a..g}	a b c d e f g
sim{08..10}	sim08 sim09 sim10

Conditions

```
if expression; then
 do something
else
 do something else
fi
```

typical if statement. Expressions can be commands and functions (return 0 → true) or built-in conditional expressions, see below

expression && *cmd*	run *cmd* if *expression* is true
expression || *cmd*	run *cmd* if *expression* is false

Unary Expressions

[-z *str*]	*str* is empty	[-n *str*]	*str* is not empty
[-e *file*]	*file* exists	[-s *file*]	*file* is not empty
[-f *file*]	*file* is a regular file	[-d *dir*]	dir is a directory
[-L *file*]	*file* is a symlink	[-x *file*]	*file* is executable
[-r *file*]	*file* is readable	[-w *file*]	*file* is writable
[-v *str*]	*str* is a variable	[-O *file*]	$USER owns *file*

Comparison Expressions

[*arg1* < *arg2*]	strings	[[*arg1* < *arg2*]]	raw strings
[*arg1* > *arg2*]		[[*arg1* > *arg2*]]	
[*arg1* == *arg2*]		[[*arg1* == *arg2*]]	
[*arg1* != *arg2*]		[[*arg1* != *arg2*]]	
[*arg1* -lt *arg2*]	integers	((*arg1* < *arg2*))	integers
[*arg1* -gt *arg2*]		((*arg1* > *arg2*))	
[*arg1* -eq *arg2*]		((*arg1* == *arg2*))	
[*arg1* -ne *arg2*]		((*arg1* != *arg2*))	

Loops

```
for word in $words; do
 echo $word
done
```

print every word in $words

```
while expression; do
 do something
done
```

traditional while loop

Two ways of iterating from 0 to 9:

```
for i in {0..9}; do echo $i; done
i=0; while (( num < 10 )); do echo $i; let i++; done
```

iterate over every line in $var:

```
IFS=$'\r\n'; for line in $var; do echo $line; done
```

Parallel Workers

```
num=100
next(){
 (( num > 0 )) && let num-- || return
 cmd&
}
set -o monitor
trap next CHLD
for i in `grep proc /proc/cpuinfo`;do
 next
done wait
trap - CHLD
```

Run *cmd* 100 times in parallel, one process per cpu core

I/O Processing

```
echo $@
```

process all arguments at once

```
while true; do
 echo "$1"
 shift || break
done
```

print every single argument on its own. To be used in a script or a function.

ls | xargs	merge all output lines
echo "foo bar" | xargs -n1	split to one word per line
echo "foo bar" | xargs -n1 *cmd*	run *cmd* on every single word
read myvar	read a line from stdin into $myvar

Bash invocation

bash -c "*cmd*"	run *cmd* in a fresh bash instance

Further Reading

BASH Programming - Introduction HOW-TO
http://www.faqs.org/docs/Linux-HOWTO/Bash-Prog-Intro-HOWTO.html

Bash Manpage (online for faster searching)
http://linux.die.net/man/1/bash