bash

GNU Bash or simply Bash is a Unix shell and command language written by Brian Fox for the GNU Project as a free software replacement for the Bourne shell. First released in 1989, it has been used widely as the default login shell for most Linux distributions and Apple's macOS Mojave and earlier versions.

- bash pause — pause function can be easily implemented in bash by read function.
- reboot_if_ping_does_not_work — Sometimes servers need to update its status or reboot by the condition to return its status to the best working environment to remove junk process or something like that. You can check that status by shell script.
- bash case statements — Sometimes we may wish to take different paths based upon a variable matching a series of patterns. We could use a series of if and elif statements but that would soon grow to be unwieldily. Fortunately there is a case statement which can make things cleaner. It's a little hard to explain so here are some examples to illustrate:
- bash if statement — bash if statement is one of the key features you can cover various use cases
- Monitor incoming HTTPD traffics in bash — tail is useful utility can display logs in Linux bash. tail -f <filename> will help you to show the recent access continuously,
- Perfect solutions when you face "Argument list too long" — The reason this occurs is because bash actually expands the asterisk to every matching file, producing a very long command line. There are some workarounds for cp, mv, rm by find, xargs, and etc.
- tee - append to a file using the tee command — tee is a command-line utility in Linux that reads from the standard input and writes to both standard output and one or more files at the same time.
- sed - replacing a string in a text file on bash — "sed" provides a method for searching-and-replacing a text in a specified file in bash, so you can make some scripts which can replace necessary things easily.
- Thread implementation in shell script — When you execute a Bash script, it will at maximum use a single CPU thread, unless you start subshells /threads. If your machine has at least two CPU threads, you will be able to max-out CPU resources using multi-threaded scripting in Bash. The reason for this is simple; as soon as a secondary ‘thread’ (read: subshell) is started, then that subsequent thread can (and often will) use a different CPU thread.
- The default bash scripts useful on CentOS server
- Tail - useful when debug/monitor based on log file on Linux