<http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO.html>

<http://www.tldp.org/LDP/Bash-Beginners-Guide/html/>

<http://linuxconfig.org/bash-scripting-tutorial>

List of some online bash tutorials

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

Bash shell scripting resources

<http://forum.codecall.net/topic/73228-bash-shell-scripting-resources/>

Guide for bash

<http://guide.bash.academy/>

Notes from this link

**So what *is* bash?**

Short answer: Bash is a *program* on your computer like any other, but designed to be easy for you to talk to.

Every program on your computer has the ability to do a vast amount of different things. Read files, start other programs, do math, control devices. The main difference between bash and most other programs is that unlike them, bash was not programmed to perform a certain task. Bash was programmed to take *commands* from you, the user. To do so efficiently, a "language" was created which allows users to "speak" to the bash program and tell it what to do. This language is the bash shell language

In essence, a *shell* program is one that provides users with an interface to interact with other programs. There is a large variety of shell programs, each with their own language. Some popular ones are the *C shell (csh)*, *Z shell (zsh)*, *Korn shell (ksh)*, *Bourne shell*,*Debian's Almquist shell (dash)*, etc. Bash (also called the *Bourne Again shell*) is currently the most popular and ubiquitously available shell. Even though all of these shells use seemingly similar syntax, it is important to be fully aware of what shell you are working in.

VNXe runs Bash shell you can verify this by running

Echo $0

***bash is a tool***, a single tool in a huge toolbox of programs. Bash alone will only let you do basic things with files and other programs. You will need to understand all the other tools in the toolbox of your system

take note of the two distinct modes of operation that the bash shell supports:

*interactive mode*

In interactive mode, the bash shell waits for your commands before performing them. Each command you pass it is executed. While a command is being executed, you cannot interact with the bash shell. As soon as the command is finished, you can interact with bash again while bash awaits your next command.

*non-interactive mode*

The bash shell can also execute *scripts*. A script is a pre-written series of commands which bash can execute without needing to ask you what to do next. Scripts are generally saved in files and subsequently used to automate a wide range of tasks.

Apart from the source of the commands bash executes, these two modes of operation are very similar. For now, suffice it to say that if bash is asking you for a command to run, you're in interactive mode. If it's running commands stored in a file, it's running a script in non-interactive mode