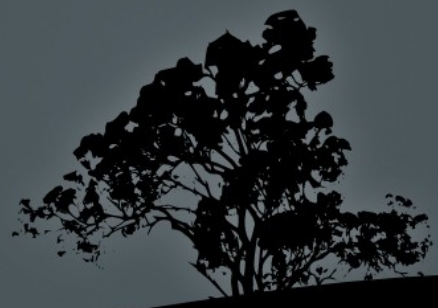
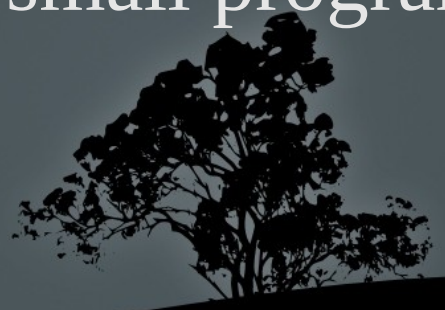
LPIC-1 102-400 – Lesson 1

105.1 Customize and use the shell environment



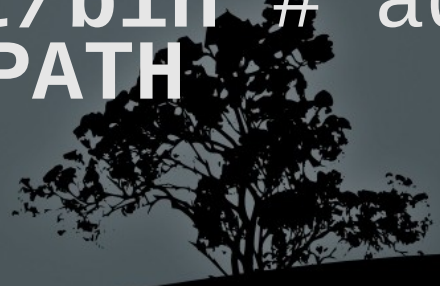
The *bash* shell

- By far, the most popular Linux shell
- Part of the GNU project
- Provides a Command Line Interface to Linux
- Has its own scripting language
- Hosts shell and environment variables
- Supports alternative command naming (**aliases**)
- Supports **functions**
- Provides the facility to run **scripts**, small programs for automating tasks



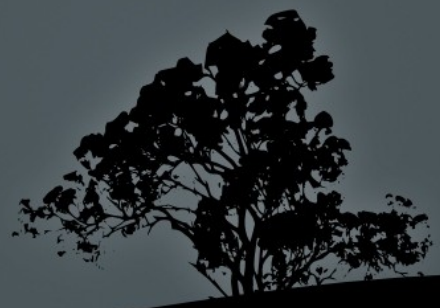
Shell variables – Environment variables

- **$ NAME=Nick** # set shell variable **NAME** with value **Nick**
- **$ export NAME** # export the **NAME** variable as an environment variable, inherited by child shells and processes
- **$ export NAME=Nick** # combine the two commands above in one
- **$ echo $NAME** # show variable **NAME** if set
- **$ export PATH=$PATH:/opt/bin** # add directory **/opt/bin** in **PATH**



Predefined Environment Variables

- **$ echo $PATH** # defines the paths for executable programs and commands
- **$ echo $HOME** # defines the home directory of the current user
- **$ echo $USER** # defines the username of the current user
- **$ echo $TERM** # sets the terminal type. it can assume values like **xterm**, **linux** or **vt100**
- **$ echo $PS1** # sets the bash prompt



Command *aliases*

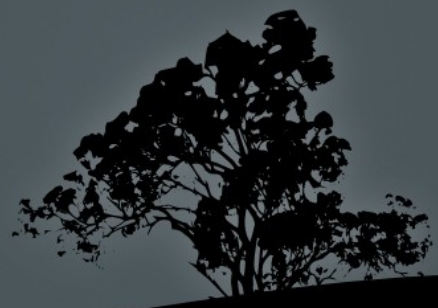
- Command **aliases** are used to create alternative commands which combine or alter the behavior of current commands
- **$ alias grep='grep --color=auto'** # the **grep** alias will run the command: **grep –color=auto**
- **$ \grep UUID /etc/fstab** # run the bare **grep** command not the alias!
- **$ alias many='cd; ls -la; pwd'** # combine many commands into one
- **$ many** # run the previously set alias as a command
- **$ alias** # running **alias** without arguments will print the current aliases

Bash *functions*

- Bash **functions** provide additional functionality with respect to aliases
- **$ function manyf () { cd; ls -la; pwd; }** # similar to the command: **$ alias many='cd ; ls -la ; pwd'**
- **$ manyf () { cd; ls -la; pwd; }** # the **function** command is optional and can be omitted
- **$ manyf** # run the **manyf** function
- **$ manyf2 () { cd $1; ls -la; pwd; }** # modify the **manyf** function so as to accept arguments: **$1** = first cli argument
- **$ manyf2 /etc** # run function **manyf2** with argument: **$1 = /etc**

Show variables, aliases and functions

- **$ env** # show environment variables
- **$ alias** # show aliases
- **$ set** # show shell variables and functions
- **$ man env** # more info about **env**
- **$ man builtins** # look for more information about **alias**, **set** and **unset**

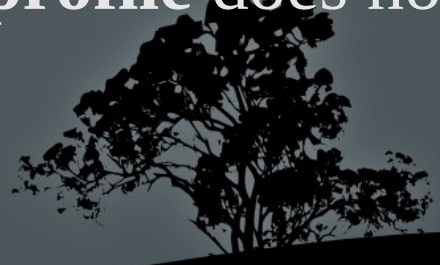


The `set` and `unset` commands

- **$ set -o** # show **bash** configuration parameters
- **$ set -o <param>** # activate parameter
- **$ set +o <param>** # deactivate parameter
- **$ unset <var>** # unset a shell or environment variable

Bash Configuration files

- **/etc/profile**: global initialization files executed on login for all users. It usually contains global variables like **$PATH** and startup applications. There is also the **/etc/profile.d/** directory where different script files serve the same purpose as **/etc/profile**
- **/etc/bashrc (or /etc/bash.bashrc)**: global initialization file, executed on **bash** startup for all users. Usually contains functions or aliases
- **~/.bash_profile**: personal initialization file, different for each user. Executed on login
- **~/.bash_login**: personal initialization file, different for each user. Executed on login only if **bash_profile** does not exist.



Bash Configuration files

- **~/.profile**: personal initialization file, different for each user. Executed on login if **bash_profile** or **bash_login** does not exist
- **~/.bashrc**:personal initialization file, different for each user. Executed on **bash** startup
- **~/.bash_logout**: executes on logout from bash
- **~/.inputrc**: optional personal configuration file, that may contain **bash** configuration option that vary from the default

The */etc/skel* directory

- The **/etc/profile** and **/etc/bashrc** are common for all users and executed before the respective personal configuration files (**.bash_profile**, **.bash_login**, **.profile**, **.bashrc**)
- The personal **.bash_profile**, **.bash_login**, **.profile**, **.bashrc** and **.bash_logout** are created on new user creation and copied from **/etc/skel**
- After the creation of the personal configuration files the users have the right to tweak these files as they please
- The **/etc/skel** directory provides the “skeleton” for the structure and content of the personal home directory of new users

The *$?* special variable

- The **$?** special variable holds the exit status of the previous command. If there are no errors, the result is ”**0**” while on errors it can assume different values like ”**1**”, ”**2**”, ”**127**” etc
- **$ grep ext /etc/fstab** # search for the existing string '**ext**' in **fstab**
- **$ echo $?** # in this case the exit value is "**0**" (string exists)
- **$ grep bogus_string /etc/fstab** # search for a non existing string in **fstab**
- **$ echo $?** # in this case the exit status is "**1**"
- **$ grep --nopar ext /etc/fstab** # use an invalid grep option
- **$ echo $?** # in this case the exit status is "**2**"
- **$ man <command> | grep -A 4 "exit status"**

Bash Lists

- Bash **lists** are sequences of commands or expressions which are separated by one of the operates “**;**”, “**&**”, “**&&**”, or “**||**” and optionally terminated by “**;**”, “**&**” or a newline “**\n**” (**man bash** and look for **Lists**)
- **$ cd /etc ; ls -la ; pwd** # the commands in this list will be executed sequentially. When the command in the left terminates, the next command starts and so on
- **$ cd /etc && ls -la && pwd** # logical AND list. Every next command will be executed only if the previous command terminated successfully, i.e. the exit status was “**0**”
- **$ cd /etc || ls -la || pwd** # logical OR list. Every next command will be executed only if the previous command terminated erroneously, i.e. the exit status was different than “**0**”

License