

Lab 7 - Linux User Environment

User Shell Environment

In Linux, each user has their own shell environment and their configurations. The configurations about each user's shell is stored in the following file(s):

- `/etc/profile` is a system-wide configuration file in Linux that sets environment variables and executes commands that apply to all users on the system. It is executed during the system startup or when a user logs in, and its primary purpose is to provide a common environment for all users.

For individual users, their shell environment files are typically stored inside their home folders. These files are:

1. `~/.bashrc` : This file is specific to the Bash shell and is usually used for defining user-specific settings. It can contain custom aliases, environment variables, and other configurations that apply only to the user when they are using the Bash shell.
2. `~/.bash_profile` : This file is also specific to the Bash shell. It is executed when a user logs in and can be used to set environment variables and perform other login-specific configurations.
3. `~/.profile` : Like `~/.bash_profile`, this file is executed during login for Bash and other compatible shells. It can contain user-specific environment variables and configurations.

NOTE: In case you're using `ZSH`, you will have `.zshrc` instead of `.bashrc`.

Aliases

In Bash, an alias is a way to create shortcuts or custom command names for commonly used commands or command sequences. Aliases can make your command-line usage more efficient and convenient by allowing you to use shorter or more memorable names for commands. You can define aliases in your `~/.bashrc` or `~/.bash_profile` file or directly in the terminal.

Creating Alias:

In order to create alias, we prefix the shortcut we want to create with the keyword `alias`. Suppose that we want to create an alias `lr` that will simply do `ls -laR`.

```
alias lr='ls -laR'
```

Now, in order to make this alias usable, we will write this to `~/.bashrc`. At the end of this file, paste the above command. However, in most Debian-distros:

```
# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi
```

This code is present inside the `.bashrc` file. This simply means that whatever alias we write inside the `.bash_aliases`, they will automatically be loaded. So, instead of writing it to `~/.bashrc`, let's write it to `~/.bash_aliases`.

```
echo "alias lr='ls -laR'" >> ~/.bash_aliases
```

Now, the above command will simply add the alias to the `bash_aliases`.

If you type `lr` in your terminal now:

```
pwn@ubuntu:~$ lr

Command 'lr' not found, but can be installed with:
sudo apt install lr
```

It didn't work. That is because we didn't load the changes. There are two ways to load the changes.

1. Opening up a new terminal
2. Using `source` command in your current terminal.

1 is pretty self-explanatory, but 2. Let's see what `source` command does.

```
pwn@ubuntu:~$ source --help
source: source filename [arguments]
    Execute commands from a file in the current shell.

    Read and execute commands from FILENAME in the current shell. The
    entries in $PATH are used to find the directory containing FILENAME.
    If any ARGUMENTS are supplied, they become the positional parameters
    when FILENAME is executed.

    Exit Status:
    Returns the status of the last command executed in FILENAME; fails if
    FILENAME cannot be read.
pwn@ubuntu:~$
```

Now, in simpler terms, `source` loads a file's variables and configurations into the current shell session. So, in order to load our changes, we'll use the following command:

```
source ~/.bashrc
```

After running this:

```
pwn@ubuntu:~/Desktop$ lr
Command 'lr' not found, but can be installed with:
sudo apt install lr

pwn@ubuntu:~/Desktop$ source ~/.bashrc
pwn@ubuntu:~/Desktop$ lr
.:
total 8
drwxr-xr-x  2 pwn pwn 4096 Oct 25 04:27 .
drwxr-xr-x 15 pwn pwn 4096 Oct 25 06:37 ..
pwn@ubuntu:~/Desktop$
```

This means that we have successfully setup an alias.

Viewing all Aliases:

In order to view all the set aliases, we can use `alias` command.

```
pwn@ubuntu:~/Desktop$ alias
alias alert='notify-send --urgency=low -i "[ $? = 0 ] && echo terminal || echo error"'
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -aF'
alias lr='ls -laR'
alias ls='ls --color=auto'
pwn@ubuntu:~/Desktop$
```

By default, many aliases are already set by the OS.

Removing an alias:

In order to remove an alias, we use `unalias` command.

```
unalias <alias_name>
## Example:
unalias lr
```

Let's unset our defined alias:

```
pwn@ubuntu:~/Desktop$ lr
.:
total 8
drwxr-xr-x  2 pwn pwn 4096 Oct 25 04:27 .
drwxr-xr-x 15 pwn pwn 4096 Oct 25 06:37 ..
pwn@ubuntu:~/Desktop$ unalias lr
pwn@ubuntu:~/Desktop$ lr

Command 'lr' not found, but can be installed with:

sudo apt install lr

pwn@ubuntu:~/Desktop$
```

PATH Variable

The `PATH` variable is an environment variable in Unix-like operating systems, including Linux, that specifies a list of directories where the system should look for executable files when a command is entered in the command line. When you enter a command, the system searches for the corresponding executable file in the directories listed in the `PATH` variable in the order they are specified. If the executable

file is found in one of those directories, the command is executed. If not, you get a "command not found" error.

Here's how the `PATH` variable works:

1. You enter a command in the terminal (e.g., `ls`).
2. The system checks each directory in the `PATH` variable, in order, to find an executable file with a matching name (e.g., `/bin/ls`, `/usr/bin/ls`, etc.).
3. If the system finds an executable with the same name, it executes that command.

To view the current `PATH` variable, you can use the `echo` command:

```
echo $PATH
```

In order to add your own commands to the `PATH`, we can modify the following few files:

1. `~/.bashrc` → This will only change the `PATH` variable for the current user.
2. `/etc/environment` → This will modify the `PATH` variable for all existing users.

Now, to modify the `PATH` variable for the current session, we will use `export` command.

```
pwn@ubuntu:~$ export --help
export: export [-fn] [name[=value] ...] or export -p
Set export attribute for shell variables.

Marks each NAME for automatic export to the environment of subsequently
executed commands. If VALUE is supplied, assign VALUE before exporting.

Options:
  -f      refer to shell functions
  -n      remove the export property from each NAME
  -p      display a list of all exported variables and functions

An argument of '--' disables further option processing.

Exit Status:
Returns success unless an invalid option is given or NAME is invalid.
pwn@ubuntu:~$
```

In simpler terms, `export` command is used to a `SYSTEM` / `SHELL` variable.

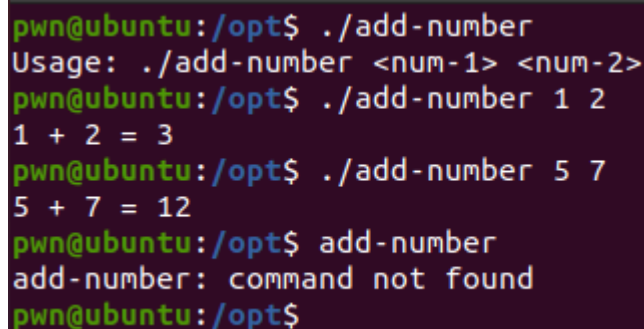
In order to add a path to the existing `PATH` variable, we use the following command:

```
export PATH=$PATH:/path/to/your/directory
```

Now, in order to understand this, let's create a simple bash script called `add-number` inside the `/opt` directory.

```
#!/bin/bash
if [[ -z $1 || -z $2 ]]; then echo "Usage: $0 <num-1> <num-2>"; fi
sum=$(( $num1 + $num2 ))
echo "$num1 + $num2 = $sum"
```

```
cd /opt/
nano /opt/add-number # Use this to add the above script
./add-number # This will run the script.
```



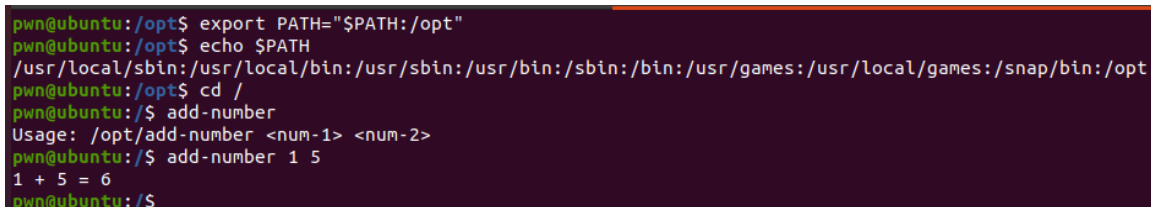
```
pwn@ubuntu:/opt$ ./add-number
Usage: ./add-number <num-1> <num-2>
pwn@ubuntu:/opt$ ./add-number 1 2
1 + 2 = 3
pwn@ubuntu:/opt$ ./add-number 5 7
5 + 7 = 12
pwn@ubuntu:/opt$ add-number
add-number: command not found
pwn@ubuntu:/opt$
```

Now, you can see that `add-number` is not working. For this to work, we need to add the path `/opt` to our `PATH` variable.

```
export PATH="$PATH:/opt"
```

This command adds the current `PATH` variable and appends `/opt` to it.

Let's run the following command and check whether we'll be able to run `add-number` or not.



```
pwn@ubuntu:/opt$ export PATH="$PATH:/opt"
pwn@ubuntu:/opt$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/opt
pwn@ubuntu:/opt$ cd /
pwn@ubuntu:/$ add-number
Usage: /opt/add-number <num-1> <num-2>
pwn@ubuntu:/$ add-number 1 5
1 + 5 = 6
pwn@ubuntu:/$
```

Now, as you can see, we can access our own command `add-number` from anywhere now. We can also give the absolute path `/opt/add-number`, and that will work as well.

Locating files and folders in Linux

There are two most common ways of finding files inside Linux

- Using `find` command
- Using `locate` command.

1. Using `locate` command

By default, `locate` command doesn't exist on most linux distros. To install it, we use the following command:

```
sudo apt install mlocate
```

Locate command is very fast in finding a specific file. This is because, it uses a database to store information about all the file names currently on your system.

Locate will be unable to find newer files on your system because those won't exist in its database. In order to update the database, use `updatedb` command.

```
locate <file-name>
## Example:
locate md5
### This will locate all files that contain the keyword `md5` in their names
```

```
pwn@ubuntu:/$ locate md5
/boot/grub/i386-pc/gcry_md5.mod
/snap/core18/2128/usr/bin/md5sum
/snap/core18/2128/usr/bin/md5sum.textutils
/snap/core18/2128/usr/share/base-files/dot.profile.md5sums
/snap/core18/2128/usr/share/base-files/profile.md5sums
/snap/core18/2128/usr/share/bash-completion/completions/rpmbuild-md5
/snap/core18/2128/usr/share/openssh/sshd_config.md5sum
/snap/core18/2128/usr/share/pam/common-account.md5sums
/snap/core18/2128/usr/share/pam/common-auth.md5sums
/snap/core18/2128/usr/share/pam/common-password.md5sums
/snap/core18/2128/usr/share/pam/common-session-noninteractive.md5sums
/snap/core18/2128/usr/share/pam/common-session.md5sums
/snap/core22/864/usr/bin/md5sum
/snap/core22/864/usr/bin/md5sum.textutils
/snap/core22/864/usr/share/base-files/dot.profile.md5sums
/snap/core22/864/usr/share/base-files/profile.md5sums
/snap/core22/864/usr/share/bash-completion/completions/rpmbuild-md5
/snap/core22/864/usr/share/openssh/sshd_config.md5sum
```

2. Using `find` command:

Find command is a very powerful command. It is used to find files and folders within a specific folder, find a file according to its name and many many more features.

Let's say, I want to find a `file`, that have `.txt` as their extensions. We will use `wildcards`. Wildcards are also known as `globs`.

Wildcards are simply `*` characters that automatically find and replace according to the search. Let's say, I want to find all files that end with `.txt`, my wildcard will be: `*.txt`. Where `*` will automatically be replaced by the file names.

Now, usage is as follows:

```
find <directory-to-search-in> [arguments]
```

By default, `find` command searches in the current directory (`.`).

Most commonly used `find` arguments are:

```
-type <type>
-name <name>
```

According to the man page, following are the types we can search for using `find`:

```
-type c
File is of type c:
b      block (buffered) special
c      character (unbuffered) special
d      directory
p      named pipe (FIFO)
f      regular file
l      symbolic link; this is never true if the -L option or the -follow option is in effect, unless the symbolic link is broken. If you want to
search for symbolic links when -L is in effect, use -xtype.
s      socket
D      door (Solaris)

To search for more than one type at once, you can supply the combined list of type letters separated by a comma ',' (GNU extension).

-uid n File's numeric user ID is n.
```

In our case, we will only deal with `f` and `d`, which are file and directory respectively.

Now, for `name`, we can use the wildcard we decided earlier, therefore, our final search command will be:


```
find -type f -name "*.txt"
```

This will find all the files that end with `.txt` in the current directory. However, if I want to find all the files that end in `*.txt` on my system, I'll use `/` as the directory:

```
find / -type f -name "*.txt"
```
