How to Add a Directory to PATH in Linux

When you type a command on the command line, you're basically telling the shell to run an executable file with the given name. In Linux, these executable programs, such as `ls` , `find` , `file` , and others, usually live inside several different directories on your system. All file with executable permissions stored in these directories can be run from any location. The most common directories that hold executable programs are `/bin`, `/sbin`, `/usr/sbin`, `/usr/local/bin` and `/usr/local/sbin`.

But how does the shell knows, what directories to search for executable programs? Does the shell search through the whole filesystem?

The answer is simple. When you type a command, the shell searches through all directories specified in the user `$PATH` variable for an executable file of that name.

This article explains how to add directories to the `$PATH` variable in Linux systems.

What is `$PATH` in Linux ?

The `$PATH` environmental variable is a colon-delimited list of directories that tells the shell which directories to search for executable files.

To check what directories are in your `$PATH`, you can use either the `printenv` or `echo` command:

```
echo $PATH
```

The output will look something like this:

```
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
```

If you have two executable files sharing the same name located in two different directories, the shell will run the file that is in the directory that comes first in the `$PATH`.

Adding a Directory to `$PATH`

There are situations where you may want to add other directories to the `$PATH` variable. For example, some programs may be installed in different locations, or you may want to have a dedicated directory for your personal scripts but be able to run them without specifying the absolute path to the executable files. To do this, you simply need to add the directory to your `$PATH`.

Let's say you have a directory called `bin` located in your Home directory in which you keep your shell scripts. To add the directory to your `$PATH`, type in:

```
export PATH="$HOME/bin:$PATH"
```

The `export` command will export the modified variable to the shell child process environments.

You can now run your scripts by typing the executable script name without specifying the full path to the file.

However, this change is only temporary and valid only in the current shell session.

To make the change permanent, you need to define the `$PATH` variable in the shell configuration files. In most Linux distributions, when you start a new session, environment variables are read from the following files:

- Global shell-specific configuration files such as `/etc/environment` and `/etc/profile`. Use this file if you want the new directory added to all system users `$PATH`.
- Per-user shell-specific configuration files. For example, if you use Bash, you can set the `$PATH` variable in the `~/.bashrc` file. If you are using Zsh the file name is `~/.zshrc`.

In this example, we'll set the variable in the `~/.bashrc` file. Open the file with your text editor and add the following line at the end of it:

```
nano ~/.bashrc
```

~/.bashrc

```
export PATH="$HOME/bin:$PATH"
```

Save the file and load the new `$PATH` into the current shell session using the `source` command:

```
source ~/.bashrc
```

To confirm that the directory was successfully added, print the value of your `$PATH` by typing:

```
echo $PATH
```

Removing a Directory from `$PATH`

To remove a directory from the `$PATH` variable, you need to open the corresponding configuration file and delete the directory in question from the `$PATH` variable. The change will be active in the new shell sessions.

Another rare situation is if you want to remove a directory from the `$PATH` only for the current session. You can do that by temporarily editing the variable. For example, if you want to remove the `/home/lina/bin` directory from the `$PATH` variable, you would do the following:

```
PATH=$(echo "$PATH" | sed -e 's/:\/home\/lina\/bin$//')
```

In the command above, we're passing the current `$PATH` variable to the `sed` command, which will remove the specified string (directory path).

If you temporarily added a new directory to the `$PATH`, you can remove it by exiting the current terminal and opening a new one. The temporary changes are valid only in the current shell session.

Conclusion

Adding new directories to your user or global `$PATH` variable is pretty simple. This allows you to execute commands and scripts stored on nonstandard locations without needing to type the full path to the executable.

The same instructions apply for any Linux distribution, including Ubuntu, CentOS, RHEL, Debian, and Linux Mint.

Revision #3
Created 29 January 2025 23:01:43 by Nicolas
Updated 13 February 2025 22:15:27 by Nicolas