<epam>

Environment variables, Path

- **What are environment variables**
- **Why export variable**
- **What is path**

TRAINING

CENTER

<epam>

What are environment variable

In Linux and Unix based systems **environment variables** are a set of dynamic named values, stored within the system that are used by applications launched in shells or subshells. In simple words, an environment variable is a variable with a name and an associated value.

Environment variables allow you to customize how the system works and the behavior of the applications on the system. For example, the environment variable can store information about the default text editor or browser, the path to executable files, or the system locale and keyboard layout settings.

Variables have the following format:

- *KEY=value*
- *KEY="Some other value"*
- *KEY=value1:value2*

The names of the variables are case-sensitive. By convention, environment variables should have UPPER CASE names

Types of environment variables

Variables can be divided into two groups:

- **Environment variables** - set up by shell configuration file, the most common of which is **~/.bashrc** and are variables that are available system-wide and are inherited by all spawned child processes and shells. The syntax for setting a variable by configuration file is the same as setting a variable in your shell:

```
[root@EVUAKYIA001B vagrant]# cat ~/.bashrc
# .bashrc

# User specific aliases and functions

alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'
export MAESTRO_CLI_HOME=/home/maestro-cli
export PATH=$PATH:$MAESTRO_CLI_HOME/bin
export AWS_CLI=/usr/local/
export PATH=$PATH:$AWS_CLI/bin
```

- **Shell variables** – are variables that apply only to the current shell instance

```
[root@EVUAKYIA001B vagrant]# export VARIABLE="value"
[root@EVUAKYIA001B vagrant]# echo $VARIABLE
value
```

How to set and list environment variables

There are several commands available that allow you to list and set environment variables in Linux:

- **env** – The command allows you to run another program in a custom environment without modifying the current one. When used without an argument it will print a list of the current environment variables.
- **printenv** – The command prints all or the specified environment variables.
- **set** – The command sets or unsets shell variables. When used without an argument it will print a list of all variables including environment and shell variables, and shell functions.
- **unset** – The command deletes shell and environment variables.
- **export** – The command sets environment variables.

How to list environment variables

The most used commands to display the environment variables are **printenv** and **env**. If the name of the variable is passed as an argument to the command, only the value of that variable is displayed. If no argument is specified, **printenv** and **env** print a list of all environment variables, one variable per line.

```
[root@EVUAKYIA001B vagrant]# printenv VARIABLE
value
[root@EVUAKYIA001B vagrant]# printenv
VARIABLE=value
XDG_SESSION_ID=5
HOSTNAME=EVUAKYIA001B.kyiv.epam.com
SHELL=/bin/bash
TERM=xterm-256color
HISTSIZE=
USER=root
MAESTRO_CLI_HOME=/home/maestro-cli
```

The **printenv** and **env** commands print only the environment variables. If you want to get a list of all variables and shell functions you can use the **set** command:

```
[root@EVUAKYIA001B vagrant]# set | head -n 20
AWS_CLI=/usr/local/
BASH=/bin/bash
BASHOPTS=checkwinsize:cmdhist:expand_aliases:extglob:extquote:force_fignore:histappend:interactive_comments:progcomp:promptvars:sourcepath
BASH_ALIASES=()
BASH_ARGC=()
BASH_ARGV=()
BASH_CMDS=()
BASH_COMPLETION_COMPAT_DIR=/etc/bash_completion.d
BASH_LINENO=()
```

How to set environment variables

To better illustrate the difference between the Shell and Environment variables we'll start with setting Shell Variables and then move on to the Environment variables.

To create a new shell variable with the name FOO and value bar simply type:

```
[root@EVUAKYIA001B vagrant]# FOO="bar"
```

Use the printenv command to check whether this variable is an environment variable or not:

```
[root@EVUAKYIA001B vagrant]# printenv FOO
[root@EVUAKYIA001B vagrant]#
```

The output will be empty which tell us that the variable is not an environment variable.
You can also try to print the variable in a sub-shell and you will get an empty output.

```
[root@EVUAKYIA001B vagrant]# bash -c 'echo $FOO'

[root@EVUAKYIA001B vagrant]#
```

How to set environment variables

The **export** command is used to set Environment variables.

To create an environment variable simply export the shell variable as an environment variable:

```
[root@EVUAKYIA001B vagrant]# export FOO
[root@EVUAKYIA001B vagrant]# printenv FOO
bar
[root@EVUAKYIA001B vagrant]#
```

If you try to print the variable in a sub-shell this time you will get the variable name printed on your terminal:

```
[root@EVUAKYIA001B vagrant]# bash -c 'echo $FOO'
bar
[root@EVUAKYIA001B vagrant]#
```

You can also set environment variables in a single line created in this way are available only in the current session. If you open a new shell or if you log out all variables will be lost.

```
[root@EVUAKYIA001B vagrant]# export BAR="foo"
[root@EVUAKYIA001B vagrant]#
```

What is $PATH and how to modify it

PATH is an environmental variable systems that tells the shell which directories to search for executable files.

Sometimes, you may wish to install programs into other locations on your computer but be able to execute them easily without specifying their exact location. You can do this easily by adding a directory to your $PATH.

To see what's in your $PATH right now, type this into a terminal:

```
[root@EVUAKYIA001B vagrant]# echo $PATH
/sbin:/bin:/usr/sbin:/usr/bin:/home/maestro-cli/bin:/usr/local//bin
[root@EVUAKYIA001B vagrant]#
```

For example, we will write the path to the bin file for the maestro-cli program. At default, the shell doesn't know about the command or2help

```
[root@EVUAKYIA001B vagrant]# or2help
bash: or2help: command not found
[root@EVUAKYIA001B vagrant]#
```

How to use $PATH variable

To fix that error we should put the maestro-cli location to the **$PATH** variable into **~/.bashrc** file:

```
# .bashrc

# User specific aliases and functions

alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'
alias sshs='ssh -i /shared-folder/private-key-for-ssh -l devops'
alias dc='docker-compose pull && docker-compose run'

export MAESTRO_CLI_HOME=/home/maestro-cli
export PATH=$PATH:$MAESTRO_CLI_HOME/bin
export AWS_CLI=/usr/local/
export PATH=$PATH:$AWS_CLI/bin
```

After that we you should create new shell or use command **source ~/.bashrc** for applying changes and try to use **or2help**:

```
[root@EVUAKYIA001B vagrant]# or2help
Response:
================================================================================================================================================
 command                                | alias       | description
================================================================================================================================================
```