EMBL Bio-IT
The Portal for Computational Biology

Variables

The shell knows two types of variables: “Local” shell variables and “global” exported environment variables. By convention, enviroment variables are written in uppercase letters.

Shell variables are only available to the current not inherited when you start an other shell or script from the commandline. Consequently, these variables will not be available for your shellscripts.

Environment variables are inherited in shells and scripts started from your current.

Setting, Exporting and Removing Variables

Variables are set (created) by assigning them a value

```
# MYVAR=something
```

There must be no whitespace around the equal sign. To create an environment variable, `export` is used. You can either export while assigning a value or in a separate step. Both to the following procedures are equivalent:

```
# export MYGLOBALVAR=”something else”
```

```
# MYGLOBALVAR=”something else”
# export MYGLOBALVAR    # No “$” in front of the variable!
```

Variables are removed with `unset`:

```
# unset MYVAR
```

Assigning a variable an empty value (`MYVAR=`) will *not* remove it!

Listing Variables

You can list all your current environment variables with `env` and all shell variables with `set`. The list of shell variables will also contain all environment variables

Variable Inheritance

Only environment variables will be available in shells and scripts started from your current shell. However in shell commands run in subshells (i.e. commands run within round brackets) also local (shell) variables of your current shell are available.

Examples

Consider the following small shellscript `vartest.sh`:

```
#!/bin/sh
echo $MYLOCALVAR
echo $MYGLOBALVAR
echo -----
```

We will use it in the following examples to illustrate the various variable inheritances:

```
# export MYGLOBALVAR=”I am global”
# MYLOCALVAR=”I am local”
#
```

Set the variables

```
# ./vartest.sh

I am global
-----
#
```

Run the script normally, i.e. in a new shell

```
# . ./vartest.sh
I am local
I am global
-----
#
```

“source” the script, i.e. run it within your current shell

```
# (echo $MYGLOBALVAR; echo $MYLOCALVAR)
I am global
I am local
#
```

Access the variables in a subshell