Examples of Variables and Arrays in Bash

Create a variable, ${var} is the same as $var, but not ambiguous.

```
var="http://wiki.plecko.hr/"
echo $var
http://wiki.plecko.hr/
```

Return string length

```
echo ${#var}
31
```

Execute the contents of $var (same as 'eval $$var')

```
echo ${!var}
```

Returns variable names beginning with 'U'

```
echo ${!U*}
UID USER USERNAME
```

Returns the text from position 7

```
echo ${var:7}
wiki.plecko.hr/
```

</code bash Returns 6 characters from position 12> echo ${var:11:6} plecko </code>

Cut 'http://' from beginning of string

```
echo ${var#http://}
wiki.plecko.hr//
```

Cut '.hr/' from end of string

```
echo ${var%.hr/}
http://wiki.plecko
```

Replaces 'plecko' with 'eurekamoment' once

```
echo ${var/plecko/eurekamoment}
```

```
http://wiki.eurekamoment.hr/
```

Replaces 'o' with 'O', always"

```
echo ${var//o/O}
http://wiki.pleckO.hr/
```

If string starts with 'http', replace 'http' with 'Site'

```
echo ${var/#http/Site}
Site://wiki.plecko.hr/
```

If string ends with 'st', replace 'st' with 'STING'

```
var2="test"
echo ${var2/%st/STING}
teSTING
```

"" (double quotation marks) protect a string, but recognize $, \ and ` as specials

```
echo "$var"
http://wiki.plecko.hr/
```

'' (single quotes) protects a string, but recognizes $, \ and ` as specials

```
echo '$var'
$var
```

$'...' (dollar sign before single quotes) protects a string completely, but interprets \n, \t, \a, etc.

```
echo $'$var\n'
$var
```

... (between crases) Execute commands in a subshell, returning the result

```
echo `ls`
Desktop Documents Downloads Images Templates Music Public Videos
```

Execute commands on a subshell

```
(ls)
Desktop Documents Downloads Images Templates Music Public Videos
```

Execute commands on a subshell, returning the result

```
echo $(ls)
Desktop Documents Downloads Images Templates Music Public Videos
```

Tests an arithmetic operation, returning 0 or 1

```
((11>9))
```

Returns the result of an arithmetic operation.

```
echo $((11-9))
2
```

Tests an expression, returning 0 or 1 (alias of command 'test')

```
[[ $var ]] && echo 'Its bigger'
Its bigger
```

Tests an expression, returning 0 or 1 (can use && and ** **)

```
[[ $var ]] && echo 'Is there this variable'
Is there this variable
```

Special Variables

```
Variable    Positional Parameters
$0 Parameter Number 0 (Name of Command or Function)
$1 Parameter Number 1 (from command line or function)
... Parameter number N …
$9 Parameter Number 9 (from command line or function)
${10}  Parameter Number 10 (from command line or function)
... Parameter number NN …
$# Total number of command line or function parameters
$* All parameters as a single string
$@ All parameters, such as multiple protected strings
Variable    Miscellania
$$ PID number of current process (from script itself)
$! PID number of last background job
$_ Last argument of last command executed
$? Return code from last command executed
```

Special escapes to use at prompt (PS1)

```
Escape  Reminder    Expands to …
```

```
a   Alert    Alert (beep)
d   Date    Date in "Weekday Month Day" format (Sat Jan 15)
e   Escape    Esc Character
h   Hostname    Machine Name Without Domain (dhcp11)
H   Hostname    Full Machine Name (dhcp11.company)
j   Jobs    Number of Active Jobs
l   Tty    Current Terminal Name (ttyp1)
n   Newline    Newline
r   Return    Return by car
s   Shell    Name of the shell (basename $ 0)
t   Time    Time in 24-hour format HH: MM: SS
T   Time    12-hour format HH: MM: SS
@  At    Time in 12-hour format HH: MM am/pm
A   At    Time in 24-hour format HH:MM
u   User    Current user login
v   Version    Bash Version (2.00)
V   Version    Bash Version Subversion (2.00.0)
w  Working Dir    Current directory, full path ($PWD)
W   Working Dir    Current directory, only the last one (basename $PWD)
!  History ico    Current command number in history
# Number    Current command number
$   ID >    Show "#" if root, "$" if normal user
nnn Octal    Character whose octal is nnn
\   Backslash    Backslash \ literal
[  Escapes    Starts a sequence of escapes (color coded type)
]  Escapes    Ends an escape sequence
```

Formatters of the date Command

```
Format  Description
%a Abbreviated Weekday Name (Sun..Sab)
%A Name of the day of the week (Sunday..Saturday)
%b Abbreviated Month Name (Jan. Dec)
%B Name of the month (January.December)
%c Complete date (Sat Nov 04 12:02:33 EST 1989)
%y Year (two digits)
%Y Year (four digits)
%m Month (01..12)
%d Day (01..31)
%j Day of the year (001..366)
%H Hours (00..23)
%M Minutes (00..59)
%S Seconds (00..60)
%s Seconds since January 1, 1970
%% A % literal
%t One TAB
%n A line break
```

Printf Command Formatters

```
Format  Description
%d Decimal number
%o Octal Number
%x Hexadecimal Number (a-f)
%X Hexadecimal Number (A-F)
%f Floating-point number
%e Number in scientific notation (e + 1)
%E Number in scientific notation (E + 1)
%s String
```

To know all local variables, execute

```
set
#or, and open txt to see later
set > VariablesLocations.txt
```

Global Variables, to know all global variables, execute

```
env
#or
printenv
```

To assign a value to a local variable

```
LINUX=free
echo $LINUX
free
```

check if the variable created in the previous item appears in the list of local variables

```
set | grep LINUX
LINUX=free
```

Now let’s make this local variable a global variable.

```
export LINUX
env | grep LINUX
LINUX=free
```

Delete a ‘local environment variable’ from memory using unset command

```
unset LINUX
echo $LINUX
```

Create an alias (nickname for a command or program), you can still include it in your ~/.bashrc

```
alias list='ls -la color=auto'
```

Destroying an aliase

```
unalias list
```

Verify all commands entered

```
history
#run command by his number in history
!468
#execute last command typed
!!
#they stay in bash_history
cat ~/.bash_history
#clear history
history -c
```

Command Line Interpreters

```
    $ - Shell of an normal user;
    # - Superuser shell root (administrator)
#Check of Available Shells
cat /etc/shells
#Variable that shows the SHELL you use
echo $SHELL
```

Creating an Array (Array) "Variable Set"

```
DISTROS=("Debian" "Trisquel" "Ubuntu" "RedHat")
#If you print the DISTROS Array as a variable, it prints variable 0,
the array displays the variables contained in it starting at 0 (zero),
so it would be the same as printing at ${DISTROS[0]}
echo $DISTROS
```

Print element 1 of array DISTROSS

```
echo ${DISTROS[1]}
Trisquel
```

You can also create an array by inserting element one at a time.

```
DISTROS[0]="Debian"
```

```
DISTROS[1]="Trisquel"
DISTROS[2]="Ubuntu"
DISTROS[3]="RedHat"
```

Change element 2 of array DISTROSS

```
DISTROS[2]="Linux Mint"
echo ${DISTROS[2]}
Linux Mint
```

Display Element 2 Distro to End

```
echo ${DISTROS[@]:2}
Ubuntu RedHat
```

This displays the elements starting at position 1 followed by two more elements consecutive to position 1

```
echo ${DISTROS[@]:1:2}
Trisquel Ubuntu
```

Know how many elements the array has DISTROSS

```
echo ${#DISTROS[@]}
4
```