Basic tools : Introduction to Linux

KASHTANOVA Victoriya

Inria, Epione

Install / Uninstall applications to Linux

If you want install (or uninstall) some apps to your Linux, you need to make several steps:

1. **Google** (commands could be different for different Linux distribution)

Install

- Run command :
  sudo apt-get install "name_of_app"
  (for Ubuntu)

Uninstall

- Check list of application with **dpkg --list** command
- Run command :
  sudo apt-get remove "name_of_app" (for Ubuntu)
- Can also run **sudo apt-get autoremove** to delete any dependencies and not used apps

Downloading/searching in/displaying files on Linux

- **To download** any files from the internet you can use command :

  wget "file url"

- **To search** some information in file you can use command :

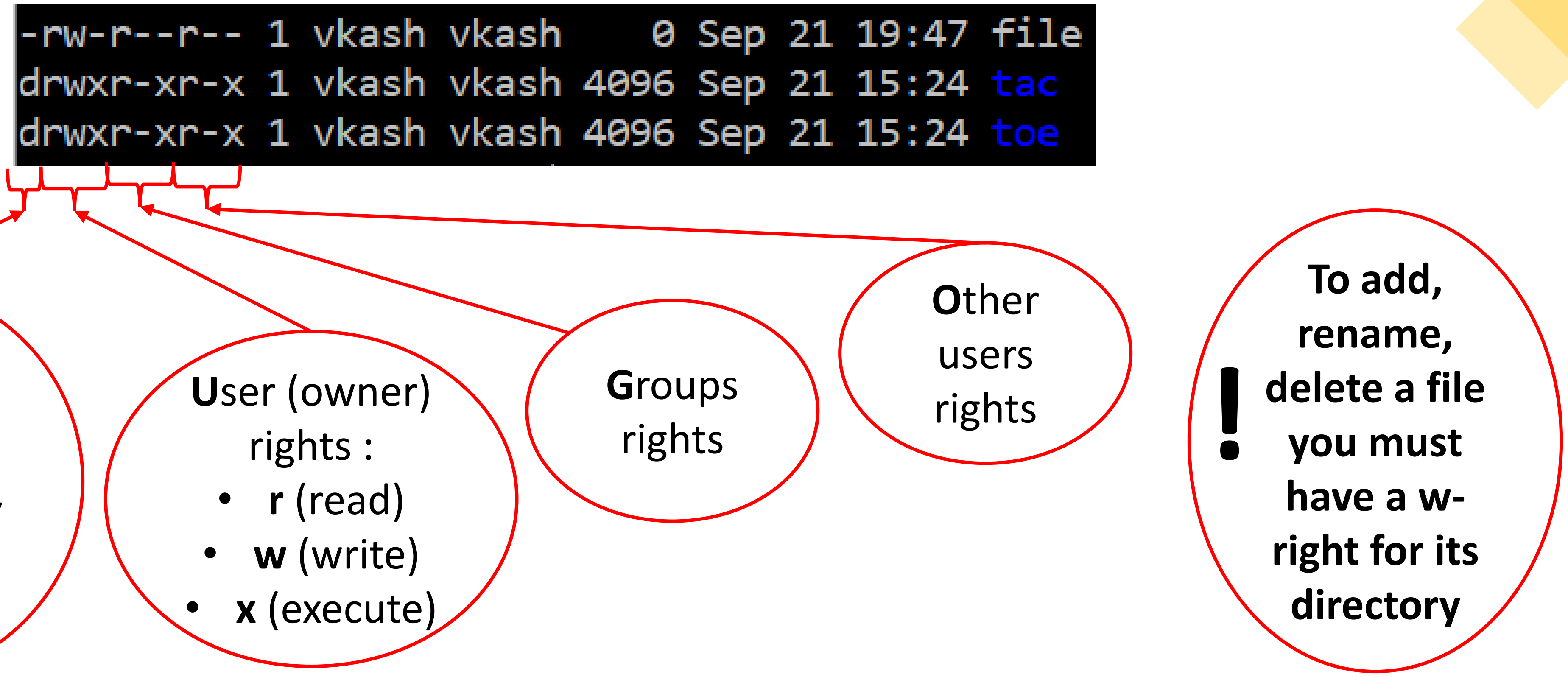
  grep "searching pattern" file

- **To display** your files you have different options :

  1. **cat** (to display full file into terminal)
  2. **head** (to display only the beginning of the file into terminal)
  3. **tail** (to display only the end of the file into terminal)
  4. **less** (to display full file page-by-page into terminal)

Files administration on Linux

Result of **ls -l** command gives all object's information, where the 1st column describes the special rights for different user categories :



```
-rw-r--r-- 1 vkash vkash    0 Sep 21 19:47 file
drwxr-xr-x 1 vkash vkash 4096 Sep 21 15:24 tac
drwxr-xr-x 1 vkash vkash 4096 Sep 21 15:24 toe
```

Indicate object's type:
- **-** file
- **d** directory
- **l** symbolic link

User (owner) rights :
- **r** (read)
- **w** (write)
- **x** (execute)

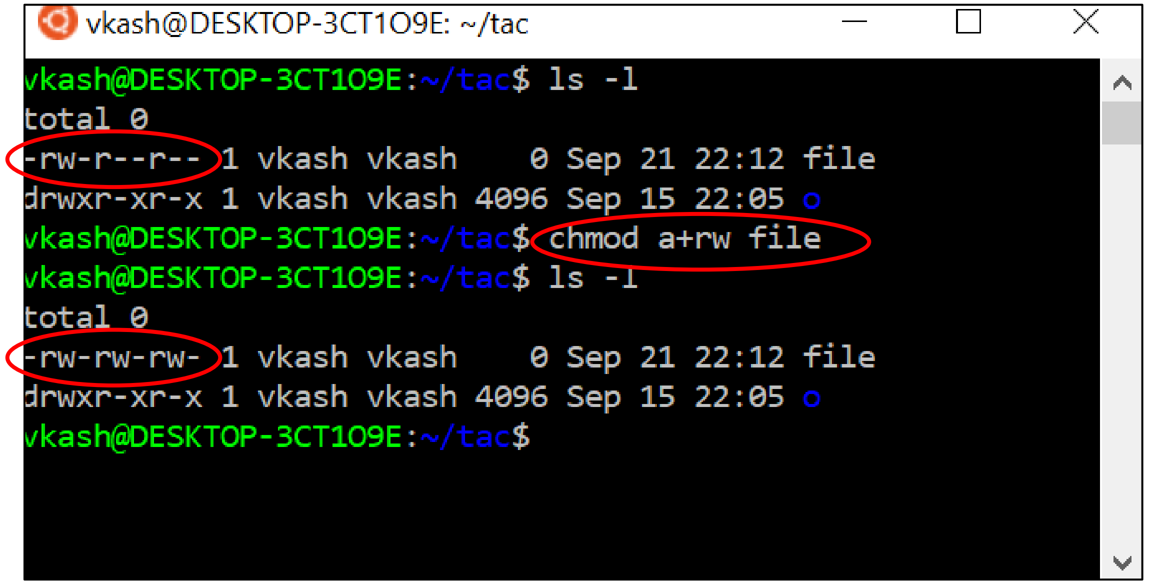
Groups rights

Other users rights

! To add, rename, delete a file you must have a w-right for its directory

Files administration on Linux

We can change these settings with **chmod** command :



vkash@DESKTOP-3CT1O9E: ~/tac

```
vkash@DESKTOP-3CT1O9E:~/tac$ ls -l
total 0
-rw-r--r-- 1 vkash vkash    0 Sep 21 22:12 file
drwxr-xr-x 1 vkash vkash 4096 Sep 15 22:05 o
vkash@DESKTOP-3CT1O9E:~/tac$ chmod a+rw file
vkash@DESKTOP-3CT1O9E:~/tac$ ls -l
total 0
-rw-rw-rw- 1 vkash vkash    0 Sep 21 22:12 file
drwxr-xr-x 1 vkash vkash 4096 Sep 15 22:05 o
vkash@DESKTOP-3CT1O9E:~/tac$
```

Where the arguments mean :

1. For whom :
   - **a** (all)
   - **u** (user)
   - **g** (group)
   - **o** (other)
2. Add (**+**) or remove (**-**)
3. Rights of :
   - **r** (read)
   - **w** (write)
   - **x** (execute)

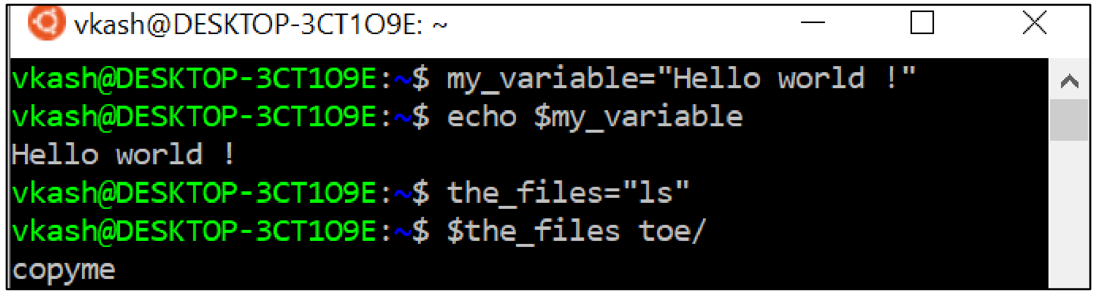
Introduction to SHELL coding

- A **shell script** is a computer program designed to be run by the Unix shell, a command-line interpreter.
- All **shell script** program files have extension **.sh** and begin **with shebang : #!** followed by name of bash interpreter (ex**. #!/bin/bash**). To know yours run: **echo $SHELL**
- **Shell script** can help you organize your work on Linux, save your time if you need to use Unix commands or command sequence repeatedly.
- Also, like in an another coding language, you can create a simple applications/games for Linux users (example of game creation will be one of your exercises today)

- For writing your **shell script** you can use command :
  - **echo "line"** with **>** (write line in file, deletes all previous file's content) or with **>>** (write line in the end of file, keeps all previous file's content)
  - Linux test redactors **vim/emacs**

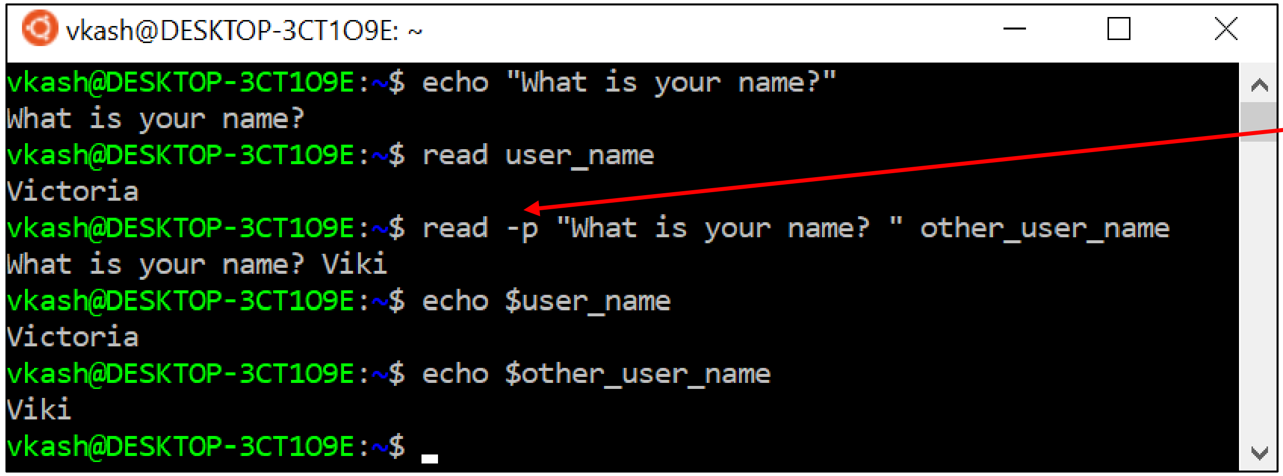
Introduction to SHELL coding : Variables

You can create variables like :



```
vkash@DESKTOP-3CT1O9E:~$ my_variable="Hello world !"
vkash@DESKTOP-3CT1O9E:~$ echo $my_variable
Hello world !
vkash@DESKTOP-3CT1O9E:~$ the_files="ls"
vkash@DESKTOP-3CT1O9E:~$ $the_files toe/
copyme
```

Or via interaction with user :



```
vkash@DESKTOP-3CT1O9E:~$ echo "What is your name?"
What is your name?
vkash@DESKTOP-3CT1O9E:~$ read user_name
Victoria
vkash@DESKTOP-3CT1O9E:~$ read -p "What is your name? " other_user_name
What is your name? Viki
vkash@DESKTOP-3CT1O9E:~$ echo $user_name
Victoria
vkash@DESKTOP-3CT1O9E:~$ echo $other_user_name
Viki
vkash@DESKTOP-3CT1O9E:~$ _
```

echo+read in one command

Introduction to SHELL coding : If

```
if [ condition ]
then
        # if-code
else
        # else-code
fi
```

```
if [ ... ]; then
        # do something
fi
```

```
if [ something ]; then
        echo "Something"
elif [ something_else ]; then
        echo "Something else"
else
        echo "None of the above"
fi
```

Introduction to SHELL coding : Case

```
case $variable in
            1st_option)
                        echo “Smth 1"
                        ;;
            2nd_option)
                        echo “Smth 2“
                        ;;
            3nd_option)
                        echo " Smth next"
                        ;;
esac
```

Introduction to SHELL coding : Loops For/While

For loops iterate through a set of values until the list is exhausted:

```
for i in 1 2 3 4 5
do
        echo "Looping ... number $i"
done
```

```
for i in $(seq 1 5)
do
        echo "Looping ... number $i"
done
```

Introduction to SHELL coding : Loops For/While

While loops iterate until condition is true :

```
while [ "$INPUT_STRING" != "bye" ]
do
		echo "Please type something in (bye to quit)"
		read INPUT_STRING
		echo "You typed: $INPUT_STRING"
done
```

Introduction to SHELL coding : Operators

Math operators :

- **-eq** (equal)
- **-ne** (not equal)
- **-gt** (greater than)
- **-lt** (less than)
- **-ge** (greater or equal)
- **-le** (less or equal)

File's operators :

- **-e** (exists)
- **-f** (file)
- **-d** (directory)
- **-L** (link)
- **-s** (size)
- **-r** (readable)
- **-w** (writable)
- **-x** (executable
- **-nt** (newer than)
- **-ot** (older than)

Introduction to SHELL coding : Operators

- **[…]** (condition)
- **[! …]** (inverse condition)
- **-a** (and)
- **-o** (or)
- **&&** (do if True)

  ```
  ls file.txt && echo “File existe”
  ```
- **||** (do if False)

  ```
  ls file.txt || echo “File doesn’t existe”
  ```

Examples :

```
x=1;y=0
[ ! $x -eq 1 ] || echo "Number is not null"
[ $x -eq 1 -o $y -eq 1 ] && echo "At least one condition is True"
if [ -e ~/script.sh ] ; then echo " Your script file existe "
```

Introduction to SHELL coding : Functions

```
function my_function ()
{
        # list of commands
         return "Smth"
}
```

```
myfunc()
{
        echo "myfunc was called as : $@"
        x=2
}
### Main script starts here
x=1
echo "x is $x"
myfunc 1 2 3
echo "x is $x"
```

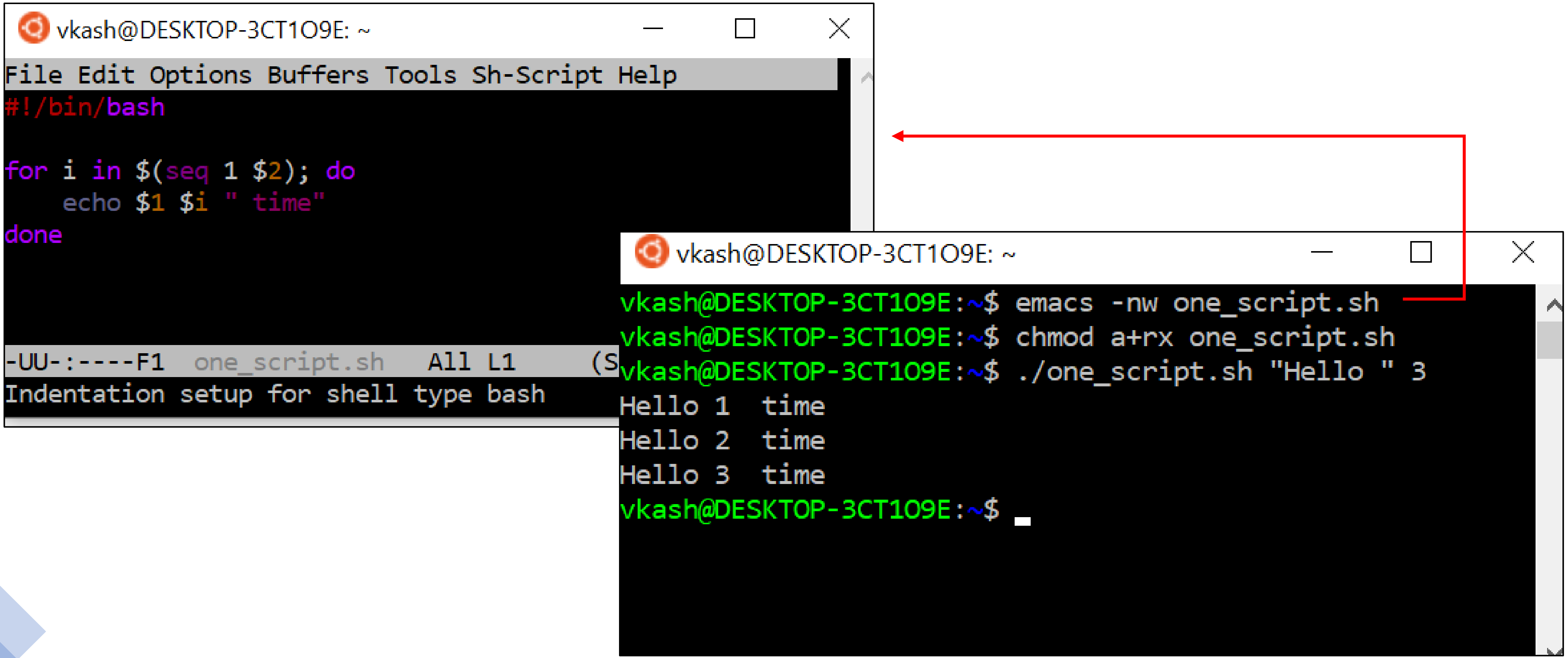
Introduction to SHELL coding : Script parameters

To get the parameters of a Shell script (the arguments that you can add on the command line when you will call the script) you can use special variables that contain these values :

- **$0** (contains the full name of the Shell script which is running)
- **$1, $2, $3, ...** (contain, respectively, the 1st, 2nd and 3rd arguments passed to the command line)
- **$*** (contains the set of arguments that were passed to the command line)
- **$#** (contains the number of arguments)

Note : To execute you Shell script use **./your_script_name.sh** (but before make it executable)

Introduction to SHELL coding : Script example



```
#!/bin/bash

for i in $(seq 1 $2); do
    echo $1 $i " time"
done
```

```
-UU-:----F1  one_script.sh    All L1     (S
Indentation setup for shell type bash
```

```
vkash@DESKTOP-3CT1O9E:~$ emacs -nw one_script.sh
vkash@DESKTOP-3CT1O9E:~$ chmod a+rx one_script.sh
vkash@DESKTOP-3CT1O9E:~$ ./one_script.sh "Hello " 3
Hello 1  time
Hello 2  time
Hello 3  time
vkash@DESKTOP-3CT1O9E:~$ _
```

Check www.shellscript.sh/ for more examples