Some useful slides for the introduction to Unix

SDAL course (2023) by **A. Pompili**

[alexis.pompili@ba.infn.it]

Note: **Linux is tightly connected to UNIX**
For the differences see here: https://www.html.it/25/05/2018/unix-e-linux-quali-sono-le-differenze/

Basic `UNIX` commands

Useful **`UNIX`** TUTORIALS :

- http://www.ee.surrey.ac.uk/Teaching/Unix/

- https://www.dia.uniroma3.it/~patrigna/recipes/unixrecipes.html#passwd

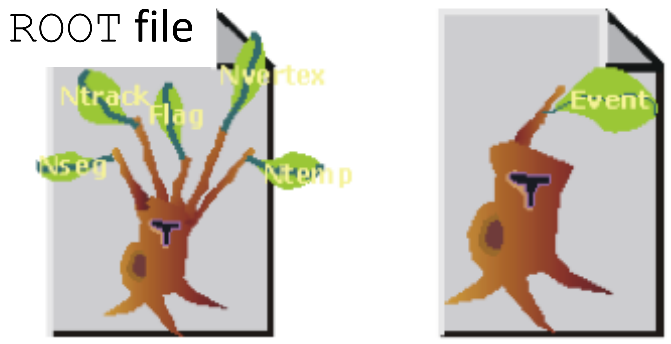
First of all you need to know the logic structure of the UNIX file repository :

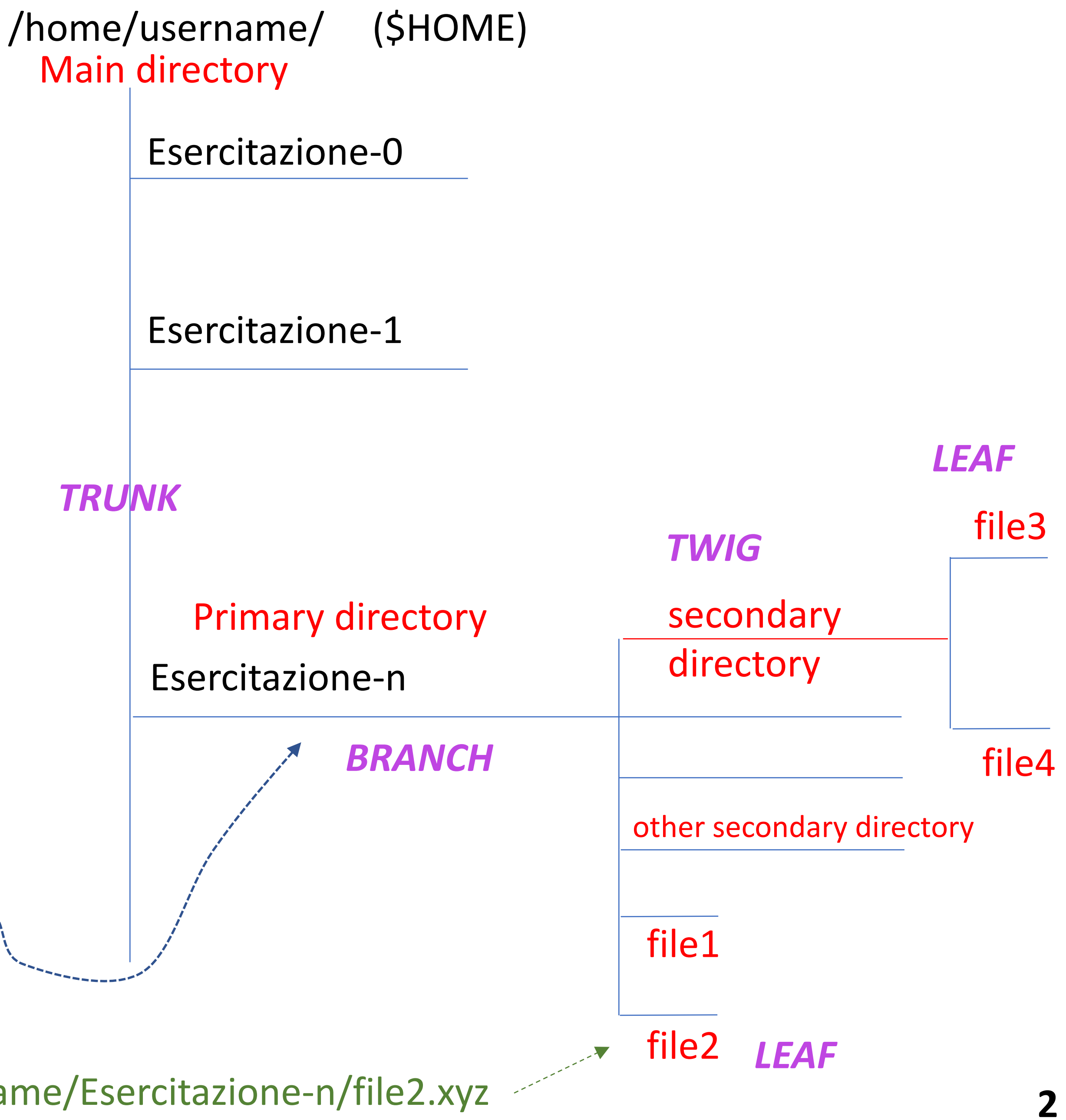
- **The concept of Tree as the logical skeleton**
- **Directories and files as bunches and leaves on a tree.**
- **Permissions**

Tree-structure of the Unix filesystem

Tree structure - based:

- Unix filesystem
- **ROOT/TTree**





/home/username/Esercitazione-n

/home/username/Esercitazione-n/file2.xyz

Few useful Unix commands - I

```
find ./directory -name "*.root" -print

find ./directory -name "*.C" -print -exec grep -i 'yield' {} \;
```

Other use of the **find** command :

$\underbrace{\texttt{find ./ -type f -ctime +3}}$ …

it finds the files older than 3 days in the current directory (./)

… `\-exec cp { } "./27oct/" \;` …

Few useful Unix commands - II

`df -h` (shows the structure of the file system included the muted devices)

`du -k --max-depth=2` (-k : kilobytes)

`du -hc --max-depth=2 /usr/` (-h : human readable, -c : display grand total; -hm in MegaBytes)

`tail -n20 history-5october2022.txt` (shows the last 20 rows of a file)

`sed "s/zanare/zanzare/g" file.txt > file-new.txt` (substitutes and writes in a new file)

Word count in a text file:

`wc history-5october2022.txt` (options: `-l` (only lines), `-c` (only fonts), `-w` (only words))

```
163  894  11091
```

(you can also do: `less file.txt | wc`)

163 → lines

894 → words

11091 → fonts (spaces are counted; starting a new paraghraph adds one space)

Command-line utilities: **SED, GREP, AWK**

If you want to study deeper: http://www.pluto.it/files/ildp/guide/abs/textproc.html

…from which is written: http://www.di.uniba.it/~reting/SEDGREP_LINUX.pdf

Tutorials: https://www.theunixschool.com/p/awk-sed.html

See also: https://www-users.york.ac.uk/~mijp1/teaching/2nd_year_Comp_Lab/guides/grep_awk_sed.pdf

SED FILTER

Shortcut of Stream Editor.

It works as a program to sequentially modify a stream of data expressed in textual form.

Specifically it filters an input file with a filter file (setting directives) provided as argument of a command line (after options); the sintax is:

sed [options] [setting directives] [file di input]

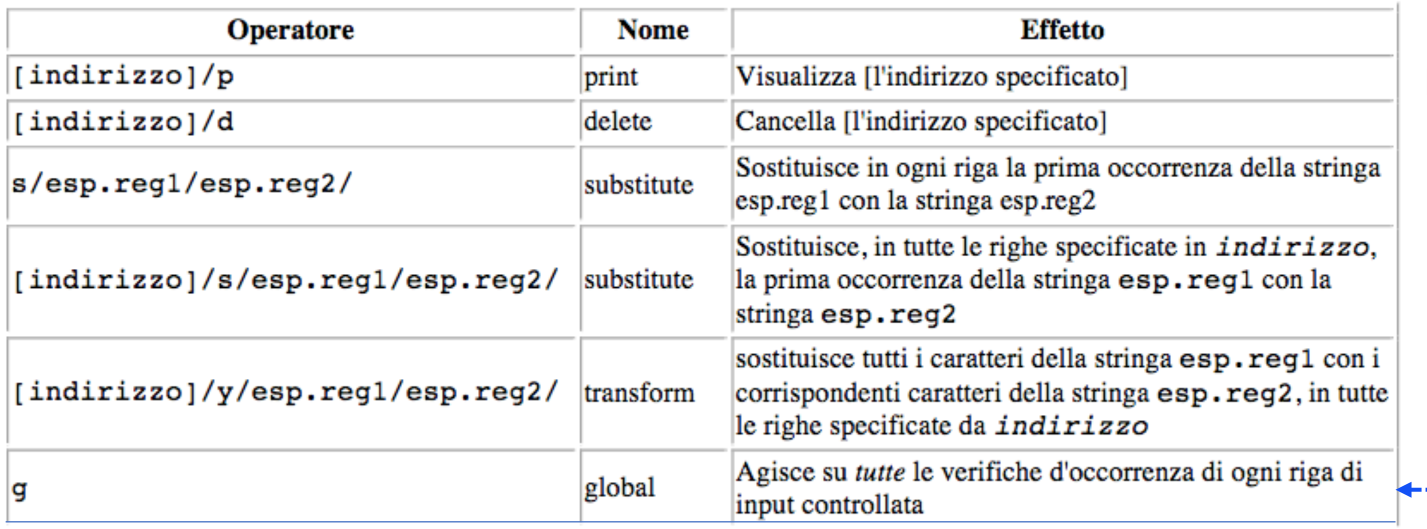
composed by a **regular expression** + command

they refers implicitely or explicitely to a group of lines (identified in some way) to which a command is applied:

[`selection of lines’]/command/

Basic **sed** operators:

(`indirizzo` = line number)

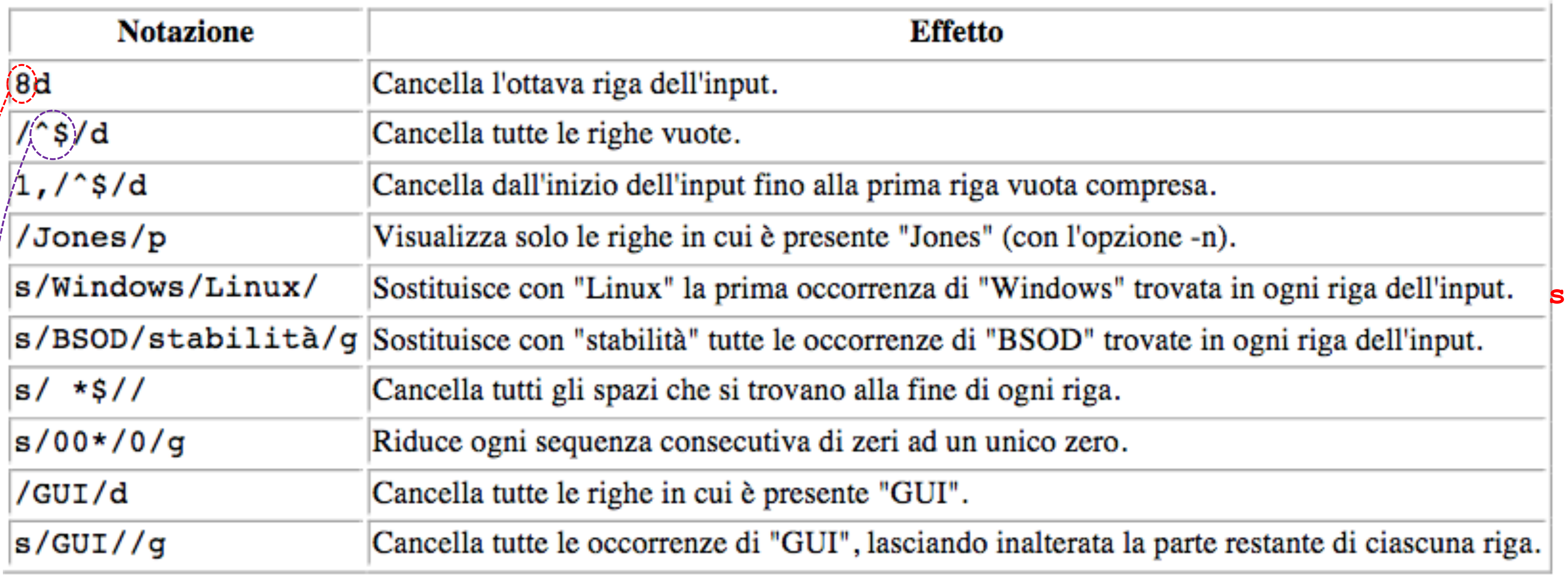


Operatore	Nome	Effetto
`[indirizzo]/p`	print	Visualizza [l'indirizzo specificato]
`[indirizzo]/d`	delete	Cancella [l'indirizzo specificato]
`s/esp.reg1/esp.reg2/`	substitute	Sostituisce in ogni riga la prima occorrenza della stringa esp.reg1 con la stringa esp.reg2
`[indirizzo]/s/esp.reg1/esp.reg2/`	substitute	Sostituisce, in tutte le righe specificate in ***indirizzo***, la prima occorrenza della stringa **esp.reg1** con la stringa **esp.reg2**
`[indirizzo]/y/esp.reg1/esp.reg2/`	transform	sostituisce tutti i caratteri della stringa **esp.reg1** con i corrispondenti caratteri della stringa **esp.reg2**, in tutte le righe specificate da ***indirizzo***
`g`	global	Agisce su *tutte* le verifiche d'occorrenza di ogni riga di input controllata

print to stdout

Se l'operatore g (*global*) non è accodato al comando *substitute*, la sostituzione agisce solo sulla prima verifica d'occorrenza di ogni riga.

SED FILTER - examples



Notazione	Effetto
8d	Cancella l'ottava riga dell'input.
/^$/d	Cancella tutte le righe vuote.
1,/^$/d	Cancella dall'inizio dell'input fino alla prima riga vuota compresa.
/Jones/p	Visualizza solo le righe in cui è presente "Jones" (con l'opzione -n).
s/Windows/Linux/	Sostituisce con "Linux" la prima occorrenza di "Windows" trovata in ogni riga dell'input.
s/BSOD/stabilità/g	Sostituisce con "stabilità" tutte le occorrenze di "BSOD" trovate in ogni riga dell'input.
s/ *$//	Cancella tutti gli spazi che si trovano alla fine di ogni riga.
s/00*/0/g	Riduce ogni sequenza consecutiva di zeri ad un unico zero.
/GUI/d	Cancella tutte le righe in cui è presente "GUI".
s/GUI//g	Cancella tutte le occorrenze di "GUI", lasciando inalterata la parte restante di ciascuna riga.

d = **delete**

p = **print**

s = **substitute**

`sed` enumerates lines from the first line of first file to the last line of the last file, without interrupting enumeration.

^= line beginning, **$**=line ending (regular expressions)

Some useful ways to use awk

```
$ echo "Hello World" | awk '{$2 = "Universe"; print $0}'
```

Substitute 2nd word "World" with "Universe"

Pre- & post-processing of a text file (I use the commands BEGIN & END respectively):

```
$ less myfile.txt
Linux e' sistema operativo portabile di tipo proprietario per computer inizialmente
sviluppato da un gruppo di ricerca dei laboratori AT&T e Bell Laboratories
```

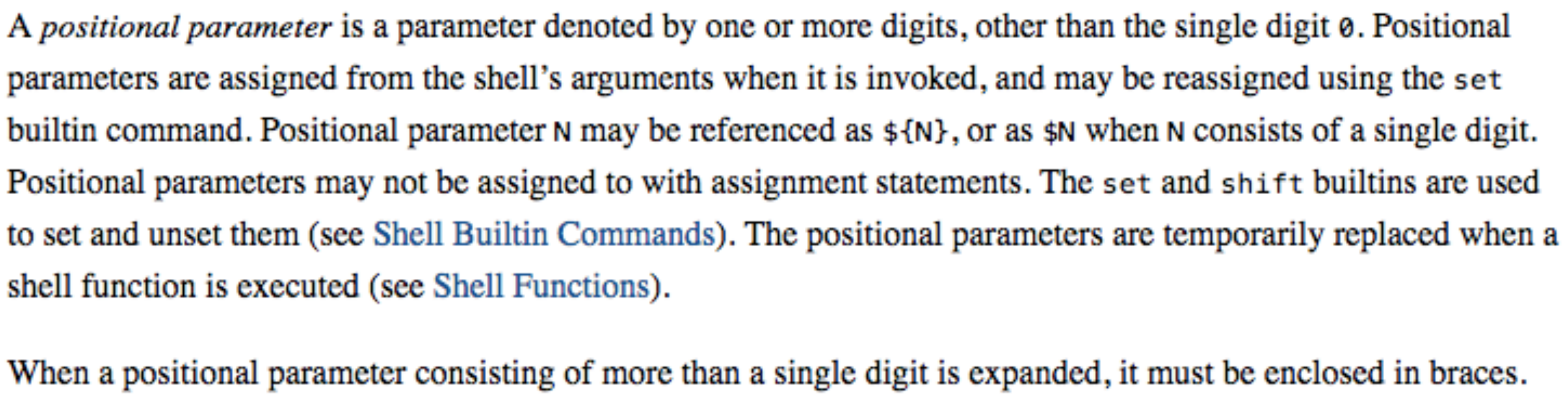
```
$ awk 'BEGIN {print "The file content starts now:"}; {print $0}; END {print "The file ends!"}' myfile.txt
```

```
The file content starts now:
Linux e' sistema operativo portabile di tipo proprietario per computer inizialmente
sviluppato da un gruppo di ricerca dei laboratori AT&T e Bell Laboratories
The file ends!
```

$0 is the executed script/file

```
$ awk '{if ($1 > 20) print $3}' myfile.txt
sistema
un
```

Nota: S1, $2, $3 are the so-called positional parameters (see next slide)

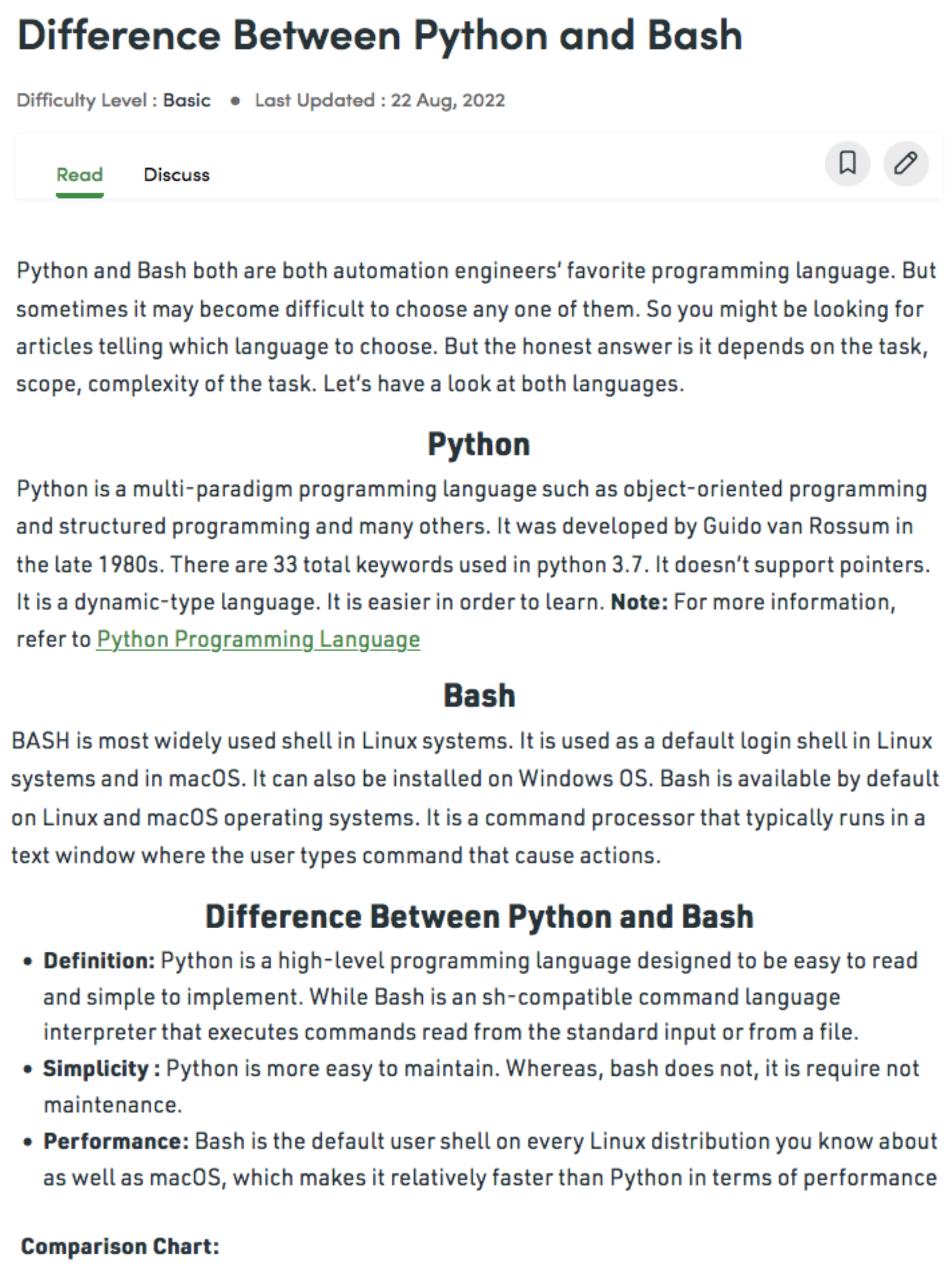


A *positional parameter* is a parameter denoted by one or more digits, other than the single digit `0`. Positional parameters are assigned from the shell's arguments when it is invoked, and may be reassigned using the `set` builtin command. Positional parameter `N` may be referenced as `${N}`, or as `$N` when `N` consists of a single digit. Positional parameters may not be assigned to with assignment statements. The `set` and `shift` builtins are used to set and unset them (see Shell Builtin Commands). The positional parameters are temporarily replaced when a shell function is executed (see Shell Functions).

When a positional parameter consisting of more than a single digit is expanded, it must be enclosed in braces.

In general the Unix shell scripting and python are "competitors"

(see next slide; from https://www.geeksforgeeks.org/difference-between-python-and-bash/)



Difference Between Python and Bash

Difficulty Level : Basic • Last Updated : 22 Aug, 2022

Read Discuss

Python and Bash both are both automation engineers' favorite programming language. But sometimes it may become difficult to choose any one of them. So you might be looking for articles telling which language to choose. But the honest answer is it depends on the task, scope, complexity of the task. Let's have a look at both languages.

Python

Python is a multi-paradigm programming language such as object-oriented programming and structured programming and many others. It was developed by Guido van Rossum in the late 1980s. There are 33 total keywords used in python 3.7. It doesn't support pointers. It is a dynamic-type language. It is easier in order to learn. **Note:** For more information, refer to Python Programming Language

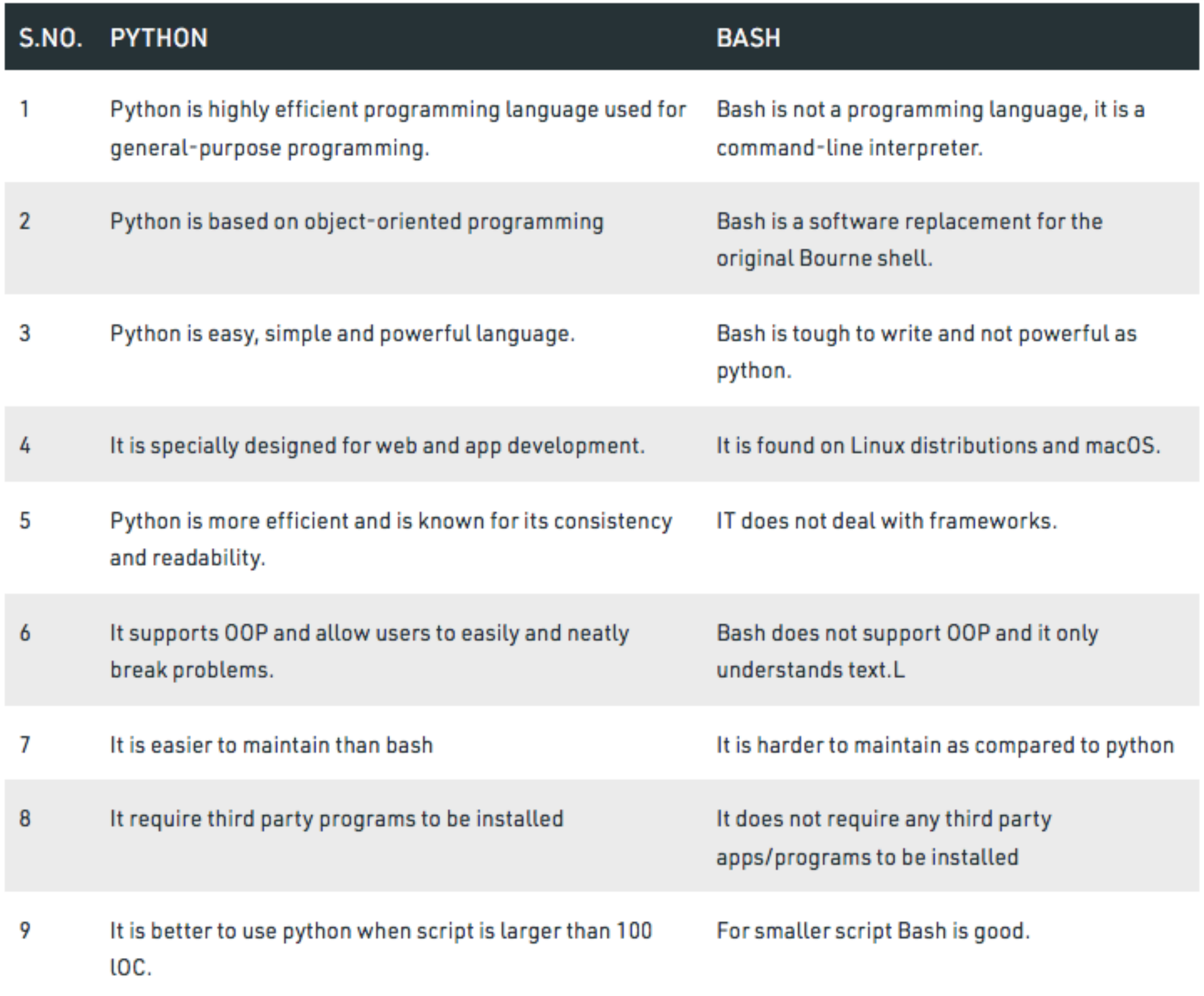
Bash

BASH is most widely used shell in Linux systems. It is used as a default login shell in Linux systems and in macOS. It can also be installed on Windows OS. Bash is available by default on Linux and macOS operating systems. It is a command processor that typically runs in a text window where the user types command that cause actions.

Difference Between Python and Bash

- **Definition:** Python is a high-level programming language designed to be easy to read and simple to implement. While Bash is an sh-compatible command language interpreter that executes commands read from the standard input or from a file.
- **Simplicity :** Python is more easy to maintain. Whereas, bash does not, it is require not maintenance.
- **Performance:** Bash is the default user shell on every Linux distribution you know about as well as macOS, which makes it relatively faster than Python in terms of performance

Comparison Chart:



S.NO.	PYTHON	BASH
1	Python is highly efficient programming language used for general-purpose programming.	Bash is not a programming language, it is a command-line interpreter.
2	Python is based on object-oriented programming	Bash is a software replacement for the original Bourne shell.
3	Python is easy, simple and powerful language.	Bash is tough to write and not powerful as python.
4	It is specially designed for web and app development.	It is found on Linux distributions and macOS.
5	Python is more efficient and is known for its consistency and readability.	IT does not deal with frameworks.
6	It supports OOP and allow users to easily and neatly break problems.	Bash does not support OOP and it only understands text.L
7	It is easier to maintain than bash	It is harder to maintain as compared to python
8	It require third party programs to be installed	It does not require any third party apps/programs to be installed
9	It is better to use python when script is larger than 100 lOC.	For smaller script Bash is good.

(*) per usare bash utilmente consultare: http://fmgroup.polito.it/quer/teaching/so/lucidi/u06-shell/u06s02-script.pdf