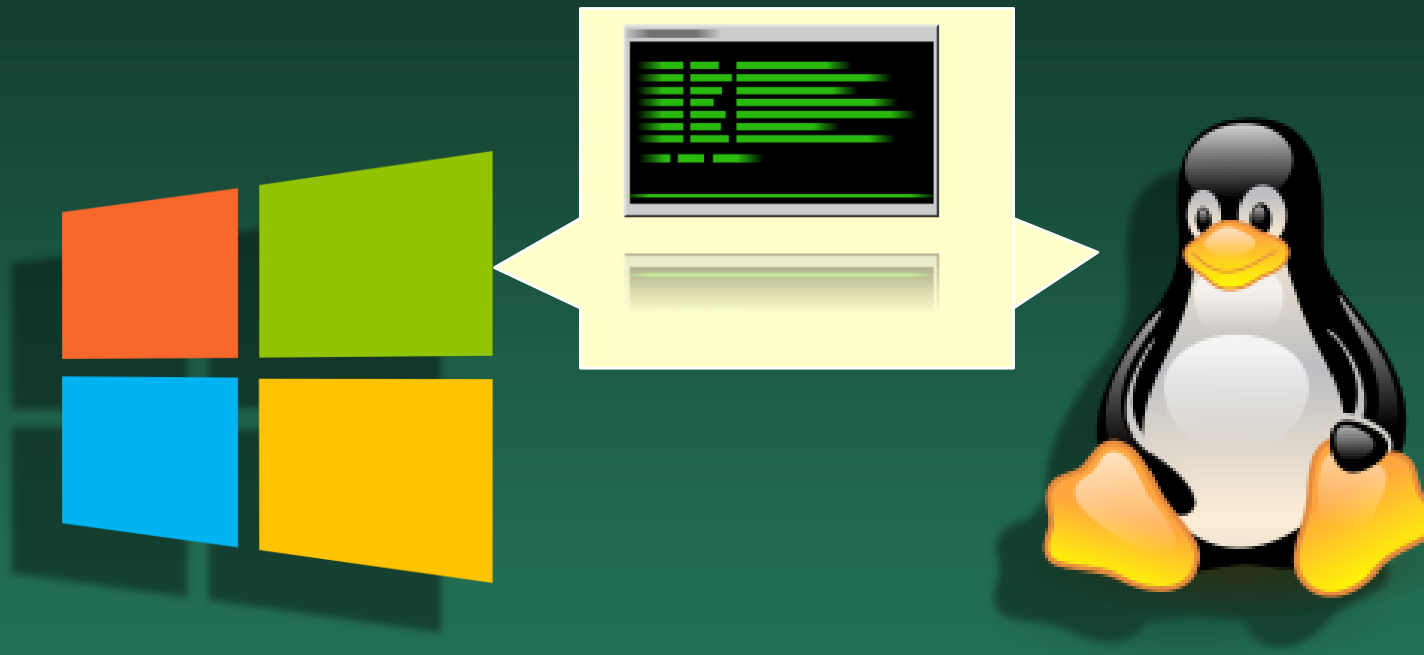
System shells



BASH syntax - 10

5 - Filters

Regex - sed, awk, grep, find, etc..

Pattern matching with REGEX

Basic Regular Expressions

.	One of any character	
*	Zero or any number of any character	
^	Start of line	
$	End of line	
[class]	One character in the group	`[Aez12]`
[^class]	One character NOT in the group	`[^ezEZ]`
[x-y]	One character in the interval	`[0-9]`
\	Escape for literal usage of a metacharacter	`\*`

Extended Regular Expressions

+	One or more of previous descriptor	`[0-9]+`
?	Zero or one of previous metacharacter	`[abc]?`
(X|Y)	Any of the strings "X" or "Y"	`(abc|D)`
{n,m}	from n to m occurrences of previous descriptor	`[0-9]{1,3}`
(...)	group descriptors together	`^([a-z][A-Z][0-9]+)$`

Special REGEX operator

- Use in `[[...]]` expression
- Operator `=~` evaluates with a REGEX

```
#!/bin/bash

read -p "Enter an email address: " email
if [[ $email =~ \
^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$ ]];
then
  echo "Valid email address."
else
  echo "Invalid email address."
fi
```

Modify a stream with sed

- sed stands for *Stream Editor*
  - Sed operates on one line at a time
    - From ^ to $
  - Select by line number
  - Select by pattern in the line
  - Substitute pattern(s)
  - Multiple commands
  - Multiple patterns
  - Combined

```
sed -n '1,4p'
sed -n '/[Hh]ello /p'
sed -n 's/Hello/Hi/gip'
sed -n -e '/^H/p' -e '/^I/p'
sed -n 's/Hello/Hi/gp;s/AA/BB/gp'
sed -n '/^A/ s/Hello/Hi/gip'
```

don't print the line

Print selected

substitute

All instances

In all lines starting with an 'A' subtitute all 'Hello' (not case sensitive) with 'Hi'

No case

- Works with fields \(...\)
  - Target pattern
  - Short way here
  - Swap in a csv

```
sed 's/\([^:]*\).*/\1/' /etc/passwd
sed 's/:.*//' /etc/passwd
sed 's/^\(.*\),\(.*\)$/\2,\1/'
```

1

2

Edit a Stream with sed

- sed does more than Pattern substitution
  - Change → `c`
  - Append → `a`
  - Insert → `i`
- Change the whole line

```
sed '/^Hello/c Bonjour tout le monde'
```

- Append text after line with pattern

```
sed '/^Hello/a ... We just said Hello!'
```

- Insert text before pattern

```
sed '/^Hello/i ... We are about to say Hello!'
```

Process a stream with awk

- `Date` format is clumsy

```
Sun Mar 19 08:46:24 CET 2023
```

- Use `awk` to re-format

```
date | awk '{print $3,$2,$6}'
```

→ `19 Mar 2023`

  - Output field separator `awk 'OFS="/" {print $2,$6}'`
  - Input field separator `awk -F: '{print $2,$6}'`

Input separator

  - Filter patterns

```
awk -F: '$3 >= 1000 {print $1}' /etc/passwd
```

  - Regex filter patterns

```
mount | awk '/on \/ type/ {print $0}'
```

  - awk BEGIN command

```
mount | awk 'BEGIN {print "ROOT"} /on \/ type/ {print $0}'
```

  - awk END command

```
mount | awk 'END {print "Done"} /on \/ type/ {print $0}'
```

Awk scripts

```
#!/bin/awk -f
BEGIN {
  FS=";"   # set the field separator to tab
  print "Name   Average Grade"  # output header
}
{ # process each line in the input file
  name = $1   # first field is the student's name
  grade = $2  # second field is the student's grade
  total_grades[name] += grade  # student's total grade
  num_grades[name]++   # student's nb of grades
}
END { # output the results for each student
  for (name in total_grades) {
    # calculate the average grade
    avg_grade = total_grades[name] / num_grades[name]
    if (name != "") {
      # output the student's name and average grade
       printf "%s\t  %.2f\n", name, avg_grade
    }
  }
}
```

```
Martin;8
Koroke;12
Arzack;15
Martin;12
Mahalla;13
Koroke;8
Martin;11
Arzack;18
```

```
Name    Average Grade
Koroke     10.00
Arzack     16.50
Mahalla    13.00
Martin     10.33
```

Search & Find

Seek content with grep

- The `grep` command is a line filter

```
@echo off
Echo foo foot Groot > text.txt
Echo Bar Bore Bier >> text.txt
Echo FooBar foo bar >> text.txt
grep "Bar" text.txt
Echo --------------
grep "[Ff]oo" text.txt
Del /Q text.txt
```

- `grep` has many arguments
  - `-i` will ignore case
  - `-v` will invert the match
  - Can be used as a filter with « pipe »

```
type text.txt | grep -i "foo"
```

Seek filesystem with find

- The find command is a file name filter

```
find /usr/lib -name "*linux*.so"
```

```
find /usr/lib -name "*vulkan*.so" -size +10000
```

Kilo bytes

```
find . -name "*.sh" -mtime +2
```

Past Days

```
find / -type d -name "*lib*"
```

- defaults to -type f

- find can be used to process the files

```
find . -name "*.sh" -mtime +3 -exec cp {} ./BKP \;
```

Compare

Compare files with diff

- Compare files line by line
  - Useful for developers !

```
diff -y mybash1.sh mybash2.sh
```

```
diff -e file1 file2
```

Output are modifications to apply on file1 to get file2

- Many options for output

Option –y shows columns

```
#!/bin/bash                                  #!/bin/bash
# ============= BASH ARGUMENTS =============| # ============= BASH CALLS =============

VAR=$1                                       | VAR="LEVEL 7"
echo $0                                      | echo "hello from $0 - $VAR - $1"
echo $1-$2-$3-$4-$5                          <
echo $6-$7-$8-$9-${10}                       <
echo ${11}-${12}-${13}-${14}-${15}           <

                                             > ./bsh06.sh AA BB CC
                                             > echo "back to $0 - $VAR  - $1"

                                             > source bsh06.sh AAA BB CC
                                             > echo "back to $0 - $VAR  - $1"
                                             >
                                             > bash bsh06.sh DDD EE FF
                                             > echo "back to $0 - $VAR  - $1"
```

Compare files with cmp

- Compare files byte by byte

```
cmp -b -n 58 img.bmp attach.bmp
```

- `cmp` will give differences between the two files
  - Give details if files have the same
  - May receive sets of files