Pipelines & Logical Operators

- what is a pipeline |
- How to use logical operators in Bash (||, &&)

TRAINING
CENTER

Data Streams

Data streams are terms of transferring data between two points. Any command that you run, and even shell itself, use streams.

There are 3 types of stream connections in Linux:

Stream	Description
STDIN	standard input, receives text as input.
STDOUT	standard output, sends text as an output.
STDERR	standard error, sends only error text, so it doesn't mix up with stdout.

Every command can one way or another receive data from **STDIN**, output it's result to **STDOUT** and send error messages to **STDERR**. Everytime you see a result of your command in Bash, that means the command you've run sent its **STDOUT** data to the terminal that printed out the result.

Pipelines

Pipelines use data streams to transfer output data of one command directly to the input data of another one in one line. Pipelines let you create beautiful one-liners of code that do a lot of things instead of writing a huge multi-line script.

Simply speaking, **a pipeline let's you connect STDOUT of one command to STDIN of another one.**

```
bash-3.2$ echo 'Hello, World!' | rev | tr ' ' '\n' | rev | tr ',' '.' | tr '!' ','
World,
Hello.
```

pipelines example

There are 2 types of pipes:

|	Sends STDOUT only
|&	Sends STDOUT and STDERR

Pipelines

```
bash-3.2$ ls /var/log | grep ".log$" | sort
PanGPUninstall.log
alf.log
appfirewall.log
corecaptured.log
displaypolicyd.stdout.log
fsck_apfs.log
fsck_apfs_error.log
fsck_hfs.log
install.log
shutdown_monitor.log
system.log
wifi.log
```

```
bash-3.2$ wc -l /var/log/*.log | sort -n
       0 /var/log/alf.log
       0 /var/log/appfirewall.log
       0 /var/log/shutdown_monitor.log
       1 /var/log/displaypolicyd.stdout.log
       8 /var/log/fsck_apfs_error.log
      72 /var/log/PanGPUninstall.log
     511 /var/log/wifi.log
    2437 /var/log/fsck_hfs.log
    3419 /var/log/system.log
    8110 /var/log/fsck_apfs.log
  601401 /var/log/install.log
  628886 /var/log/corecaptured.log
 1244845 total
```

Logical Operators

Logical Operators are integral part of the computing itself and exist in a lot of different languages.

Sign	Name	Description
||	Logical OR	Executes command on the right if command on the left exited with code 0 (succeeded)
&&	Logical AND	Executes command on the right if command on the left exited with non-zero exit code (failed)

there are 2 useful ways of how we can use them:

1. Inside if between conditionals (**[ ]** or **[[ ]]**)
2. Between commands to implement some sort of if/else

Logical Operators

1. Inside if between conditionals (**[ ]** or **[[ ]]**)

```
bash-3.2$ cat example_1.sh
#!/bin/bash
if [ -z $2 ] || [ ! -z $3 ]; then
  echo "You need to set exactly 2 parameters"
  exit 1
fi

isnumber='^[-]?[0-9]+$'
if [[ $1 =~ $isnumber ]] && [[ $2 =~ $isnumber ]]; then
  echo "The sum of numbers are $(( $1 + $2))"
else
  echo 'Please, enter numbers only!'
  exit 1
fi
bash-3.2$
```

example code

```
bash-3.2$ ./example_1.sh 1
You need to set exactly 2 parameters
bash-3.2$ ./example_1.sh 1 2 3
You need to set exactly 2 parameters
bash-3.2$ ./example_1.sh 2 foo
Please, enter numbers only!
bash-3.2$ ./example_1.sh bar 123
Please, enter numbers only!
bash-3.2$ ./example_1.sh 12 24
The sum of numbers are 36
bash-3.2$
```

example results

Logical Operators

2. Between commands to implement some sort of if/else

```
#!/bin/bash
[ -z $2 ] || [ ! -z $3 ] && echo "You need to set exactly 2 parameters" && exit 1
isnumber='^[-]?[0-9]+$'
[[ $1 =~ $isnumber ]] && [[ $2 =~ $isnumber ]] && echo "The sum of numbers are $(( $1 + $2))" && exit 0 || echo 'Please, enter numbers only!' && exit 1
```

example code (cranked up to 11)

```
bash-3.2$ ./example_2.sh 1 || ./example_2.sh 1 2 3 || ./example_2.sh 4 5
You need to set exactly 2 parameters
You need to set exactly 2 parameters
The sum of numbers are 9
bash-3.2$ ./example_2.sh 1 aaa || ./example_2.sh bbb 3 || ./example_2.sh 23 34 && ./example_2.sh 34 45
Please, enter numbers only!
Please, enter numbers only!
The sum of numbers are 57
The sum of numbers are 79
bash-3.2$
```

example results

Thanks for watching!