Basic programming in Bash

Bash programming

- In the previous tutorial you got to know basic Bash commands
- Bash is also a programming (scripting) language
- More sophisticated execution of commands (upon a condition, several times in a row, etc.) is possible through Bash scripts

Motivation

- Basic programming is useful as it allows you to automate tasks
- MMseqs2 software suite allows creating tailored computational tools by combining its modules and workflows in Bash scripts

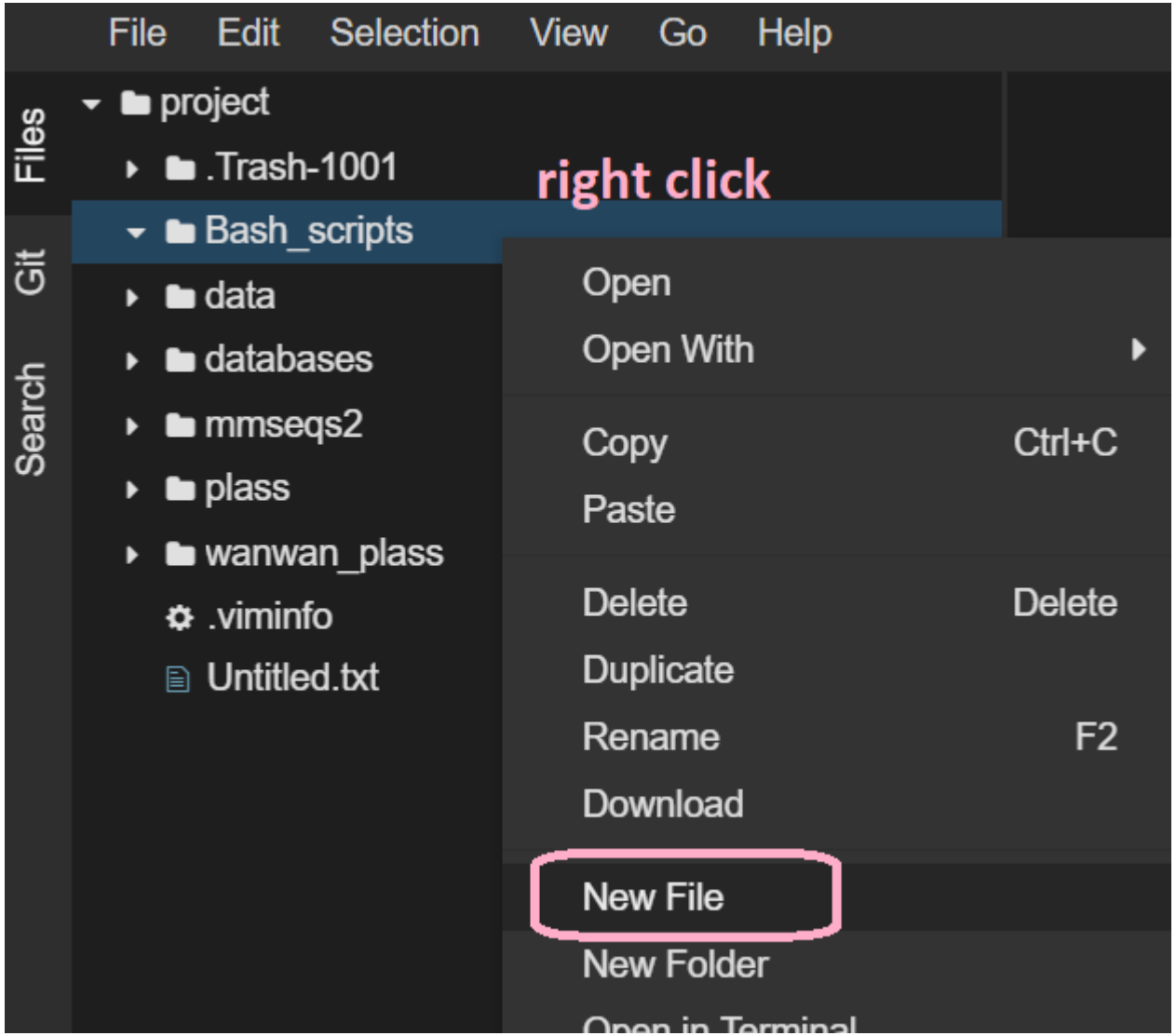
The script file

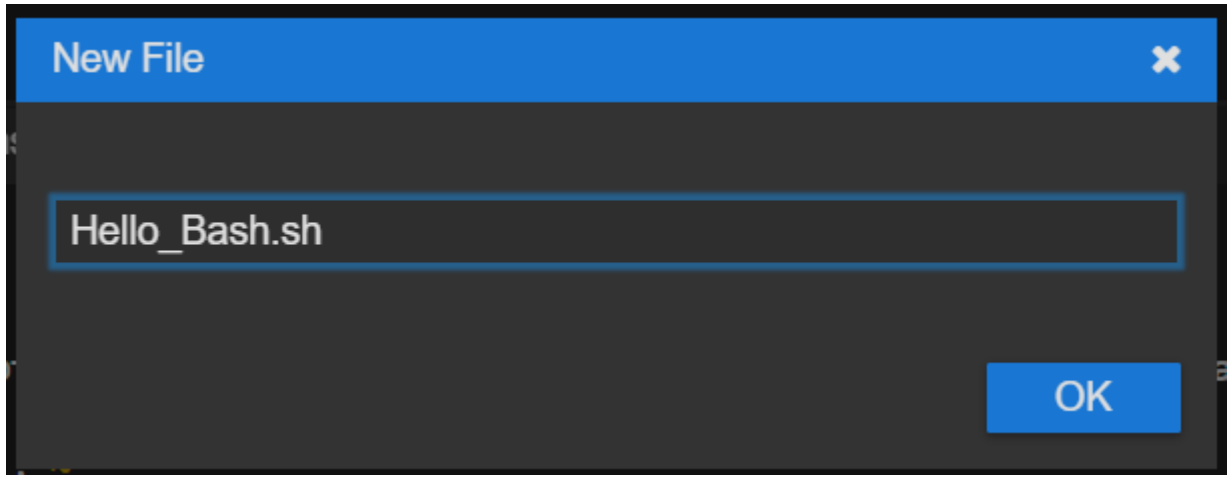
- The first line of a Bash script is usually:

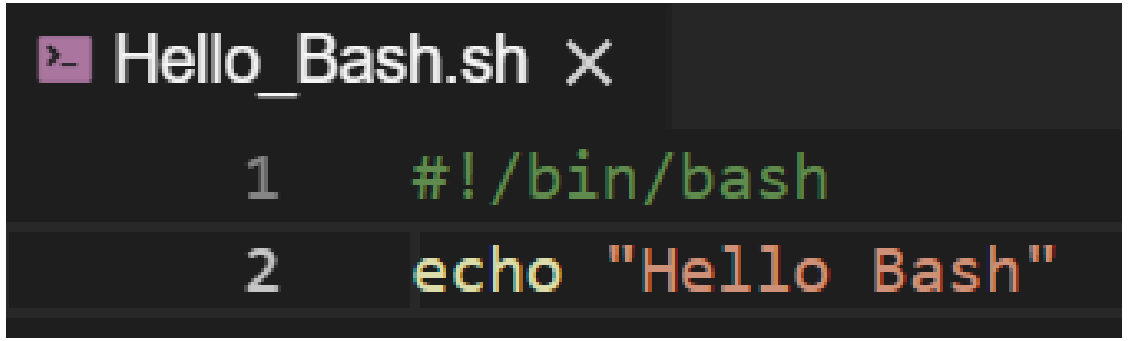
```
#!/bin/bash
```

- This indicates this file is a Bash script
- Lines that start with '#' are comments
- To print something we use '`echo`'
- A script is just a text file.
- **Under your home directory, create a directory called "Bash_scripts"**
- We will create Bash scripts there

Creating the Hello_Bash.sh script file







Running a Bash script

- You need to give your script execution permission:

```
chmod +x ~/Bash_scripts/Hello_Bash.sh
```

- Then you can run it from the terminal:

```
13:21:57 :: ~
$ chmod +x ~/Bash_scripts/Hello_Bash.sh

13:21:59 :: ~
$ ~/Bash_scripts/Hello_Bash.sh
```

Hello_Bash.sh

Create a Hello_Bash.sh script and run it

Bash variables

- A variable stores a value
- There are no variable types in Bash
- Assignment of a value is done with “=“:

```
#!/bin/bash
NAME="Eli"
NUMBER_OF_EYES=3
echo "Hello $NAME, you have $NUMBER_OF_EYES eyes"
```

- **Modify the Hello_Bash.sh script to have a variable and run it**

Arithmetic evaluation

- In order for bash to treat the variable as numeric we need to use brackets:

```
CORRECT_NUMBER_OF_EYES=$((NUMBER_OF_EYES - 1))
echo "Humans usually don't have more than
$CORRECT_NUMBER_OF_EYES eyes"
```

- **Create a Bash script with a variable AGE and assign it your age. Print the age you will be in one year**

Conditionals

- If/else structures allow us to execute commands only in certain cases

```
AGE=20
if [ "$AGE" -eq 20 ]; then
   echo "Wow, you are exactly 20!"
fi
```

- Comparison operators:

Description	Numeric	String
less than	-lt	<
greater than	-gt	>
equal	-eq	=
not equal	-ne	!=
less or equal	-le	
greater or equal	-ge	

Exercise

- This simple Bash script asks the user for their name and says hi:

```
#!/bin/bash
echo "Enter your name and press [ENTER]: "
read NAME
echo "Hi $NAME"
```

- **Create a script that asks for the user's age and serves beer only if the user is at least 18**

What does this code do?

```
echo "Enter a directory name and press [ENTER]: "
read DIR
if [ -d "$DIR" ]; then
    ls "$DIR"
else
    mkdir "$DIR"
fi
```

Repetitive execution of commands

- Often we would like to perform the same thing more than once:
  - Say hello to all students in the class (there 22 of you!)
  - Make a copy of each file in a directory
  - Refine an MMseqs2 clustering…

- Bash loops allow us to do exactly that!

For loop

```
#!/bin/bash
START=1
END=22
for (( i=$START; i<=$END; i++ ))
do
    echo "$i. Hi, student!"
done
```

While loop

```
# continue from last slide
i=1
while [[ $i -le $END ]]
do
    echo "$i. Oh hi there, student!"
    ((i = i + 1))
done
```

Exercises

1. **Compute the sum of the first 40 natural numbers:**

   1+2+…

2. **Sum the numbers the user provides you until they provide a negative number**

   Can you tell how many numbers you summed?

Taxonomy Bash workflow

- In the previous tutorial you saw MMseqs2 workflow to assign taxonomic units.
- This workflow is written as a Bash script which calls Bash commands as well as MMseqs2 native CPP modules
- Let’s have a look…

Taxonomy Bash workflow

```
INPUT="$1"
TARGET="$2"
RESULTS="$3"
TMP_PATH="$4"

if [ ! -e "${TMP_PATH}/first" ]; then
    "$MMSEQS" search "${INPUT}" "${TARGET}" "${TMP_PATH}/first"
       "${TMP_PATH}/tmp_hsp1" ${SEARCH1_PAR} \
        || fail "First search died"
fi

if [ ! -e "${TMP_PATH}/top1" ]; then
    "$MMSEQS" filterdb "${TMP_PATH}/first" "${TMP_PATH}/top1" --extract-lines 1 \
        || fail "Filterdb died"
fi
```

Taxonomy Bash workflow

```
if [ ! -e "${TMP_PATH}/aligned" ]; then
    "$MMSEQS" extractalignedregion "${INPUT}" "${TARGET}" "${TMP_PATH}/top1"
       "${TMP_PATH}/aligned" --extract-mode 2 \
        || fail "Extractalignedregion failed"
fi


if [ ! -e "${TMP_PATH}/round2" ]; then
    "$MMSEQS" search "${TMP_PATH}/aligned" "${TARGET}" "${TMP_PATH}/round2"
       "${TMP_PATH}/tmp_hsp2" ${SEARCH2_PAR} \
        || fail "Second search died"
fi
```

Taxonomy Bash workflow

```
# Concat top hit from 1st search with all results from 2nd search
if [ ! -e "${TMP_PATH}/merged" ]; then
    "$MMSEQS" mergedbs "${TMP_PATH}/top1" "${TMP_PATH}/merged" "${TMP_PATH}/top1"
       "${TMP_PATH}/round2" \
        || fail "Mergedbs died"
fi

# Filter out 2nd search entries that do not reach the evalue of the top 1 hit
if [ ! -e "${TMP_PATH}/2b_ali" ]; then
    "$MMSEQS" filterdb "${TMP_PATH}/merged" "${TMP_PATH}/2b_ali" --beats-first --
       filter-column 4 --comparison-operator le \
        || fail "First filterdb died"
fi

"$MMSEQS" lca "${TARGET}" "${TMP_PATH}/2b_ali" "${RESULTS}" ${LCA_PAR} \
    || fail "Lca died"
```

Exercise solutions

```
#!/bin/bash
echo "Hello Bash"
```

Exercise solutions

```
#!/bin/bash
AGE=99
AGE_NEXT_YEAR=$((AGE + 1))
echo "Next year you will be $AGE_NEXT_YEAR"
```

Exercise solutions

```
#!/bin/bash
echo "Enter your age and press [ENTER]: "
read USER_AGE
if [ $USER_AGE -ge 18 ]; then
    echo "Here is your beer"
fi
```

Exercise solutions

```
#!/bin/bash
START=1
END=40
SUM=0
for ((i=$START; i<=$END; i++)) do
    SUM=$((SUM+i))
done
echo "The result is $SUM"
```

Exercise solutions

```
#!/bin/bash
USER_NUMBER=0
NUM_NUMBERS=-1
SUM=0
while [[ $USER_NUMBER -ge 0 ]]
do
      SUM=$((SUM+USER_NUMBER))
      NUM_NUMBERS=$((NUM_NUMBERS+1))
      echo "Insert a new number [negative number to exit]:"
      read USER_NUMBER
done
echo "Final sum is $SUM and $NUM_NUMBERS numbers were summed"
```