Before we start:

You need the data-temp directory and download from Canvas the script t1.bash

stream redirection: stdout

Standard Output: stdout refers to the streams of data (plain text) that are produced by command line. By default, stdout is sent to the screen

> Redirect stdout to a file

```
command (options) argument > file
echo Hello Unix > file1
#create a new file if it doesn’t exist
#overwrite an existing file
```

>> Redirect and append stdout to an existing file

```
command (options) argument >> file
echo 2 >> file1
```

command substitution: standard output is redirected and stored into a variable

```
output_var=`line of code` #must use back quotes
output_var=$(line of code)

x=`grep -c CA temp.dat` # or  x=$(grep -c CA temp.dat)
echo $x
oneline=`grep CA temp.dat| tail -1`
echo $oneline
```

Pipeline of std out

By using the pipe operator | the output text (*stdout)* of one command can be piped into the input (*stdin)* of another command

```
command1 (options) [arguments]| command2 (options)
```

Commands after the first pipe do not have an argument.

Example

```
grep AZ temp.dat | head -3
grep AZ temp.dat | head -3 | tail -1

#the sequence of commands in a pipeline usually matters
tail -2 temp-clean.dat | grep CA
grep CA temp-clean.dat | tail -2
```

More on command grep

```
grep pattern1 filename | grep pattern2 #in any order

grep AZ temp.dat  | grep 203 #extract lines containing both patterns
```

```
grep –e pattern1 –e pattern2 filename

grep -e AZ -e CA temp.dat #extract lines containing either pattern1
or pattern2
```

stream redirection: stderr

Standard Errors: stderr refers to the streams of data (plain text) reporting error messages from command line. By default, stderr is sent to the screen

2> Redirect stderr

```
data 2> my_error #redirect errors to a file
find / -name pwd 2> /dev/null #redirect errors to the null device
```

Redirecting standard input <

Standard input: stdin: refers to the text you type as input to commands. By default, it is the keyboard.

```
sort
#type one letter in each line
#Control D
```

```
command (options) filename  #in this case the stdin is the filename
sort temp-clean.dat
```

tr

In contrast to many **commands**, **tr** does not accept **file** names as **arguments**.

```
command (options) charset1 charset2 < filename
tr A-Z a-z < temp-clean.dat
```

```
sort < temp-clean.dat
tr a-z A-Z  temp-clean.dat #wrong because < is missing
```

tr command

tr accepts two sets of characters, usually with the same length, and replaces the characters of the first set with the corresponding characters of the second set.

```
tr (options) charset1 charset2  < filename
tr A-Z a-z < temp-clean.dat  #tr takes input from temp-clean.dat
```

```
command argument | tr (options) charset1 charset2
```

```
echo linuxize | tr 'lin' 'red' #tr takes input via pipe
```

each occurrence of l is replaced with r, each occurrence of i with e, and each occurrence of n with d:

The character sets can also be defined using character ranges.

```
echo linuxize | tr l-n w-z
```

```
echo I really like tr! | tr ' ' '\n'
echo I reallllly like tr! | tr -s 'l' #-s squeeze repetition
```

Difference between tr and sed

tr performs characters transformation, sed translates words

echo good | tr 'good' 'best' #tr does character-based transformation and it replaces g with b, o with e, o with s, d with t

echo good | sed 's/good/best/g' #sed replaces the word good with the word best

```
sed  s/word1/word2/g filename

tr (options) charset1 charset2 < filename
```

Activity

Make a script called **A5-stream.bash** and answer the following questions.

For **questions 1-7**, you should run the script t1.bash as reported in the text, and in a comment line, write what happens to both the stderr (standard error) and stdout (standard output).

Look at the first two questions to understand what you should do.

```
#class header
#Q1
. t1.bash
#both stdout and stderr are sent to screen

#Q2
. t1.bash 2> t1.out
#stderr is redirected to t1.out, and stdout sent to screen

#Q3
. t1.bash > t1.out

#Q4
. t1.bash >> t1.out

#Q5
. t1.bash &> t1.out
```

Activity

In **A5-stream.bash** answer the following questions.

#Q6

```
var=`. t1.bash`
```

#Q7

```
. t1.bash 2> /dev/null | tr a-z A-Z
```

#Q8

Start with

```
echo Hello, I can get it
```

and use **ONE tr** in pipeline to translate:

e with 3

a with @

t with 9

, with *

to obtain

```
H3llo* I c@n g39 i9
```

Activity

In **A5-stream.bash** do the following – this is optional

#Q9- optional
Start with echo {z..a} and use tr and another command in pipeline to produce the following output:

```
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
```

Activity - Analyzing a file from an EEG database

cd into the data-temp directory. There is a file called eeg-control.dat.

The file eeg-control.dat contains data of an EEG, or electroencephalogram, which records the electrical signals of the brain. View the content of the file

1st field is a number – 0 means control
2nd field is the channel name
3rd field is the time step in ms
4th field is the Voltage in mV

Make a script called **A5-eeg.bash** in the data-temp directory and in it answer the following questions by using one line of code. Use Q&A format

1. Extract all the lines of **eeg-control.dat** that do not contain the # sign, and redirect the stdout into a file called eeg.dat

2. Use the echo command and stdout redirection to save the line reported below into a file called eeg-sorted.dat

```
#data are sorted according to Voltage values
```

Activity - Analyzing a file from an EEG database

1st field is a number – 0 means control
2nd field is the channel name – there are several channels
3rd field is the time step in ms
4th field is the Voltage in mV

In **A5-eeg.bash**

3. Sort the file eeg.dat according to the 4th filed, numerically, and append the stdout into the file eeg-sorted.dat

4. Count the number of lines of eeg.dat that contain the keyword FP1 and store the stdout into a variable called v1 by using command substitution.

5. Use echo to display the value of variable v1 with some text, like

```
There are .. recorded values for channel FP1
```

6. Display to screen ONLY the maximum voltage value (only one number) of channel Y. Use grep, and sort in pipeline with other commands. There are several channels. You should extract the lines containing channel Y. Use eeg.dat.

Activity - Analyzing a file from an EEG database

1st field is a number – 0 means control
2nd field is the channel name
3rd field is the time step in ms
4th field is the Voltage in mV

In **A5-eeg.bash**

7. Use tr and input redirection to change the field separator of eeg.dat from one blank character to colon : and display only the last 5 lines of the modified file to screen. Do not make a new file. Use one line of code, with input redirection and pipeline.
The output should be

```
0:Y:251:14.476
0:Y:252:15.452
0:Y:253:14.964
0:Y:254:13.987
0:Y:255:12.034
```

8. Use tr to change the field separator of eeg.dat from one blank character to comma and save the new file in eeg1.dat. Use both stdin and stdout redirections

Activity - Analyzing a file from an EEG database

1st field is a number – 0 means control
2nd field is the channel name – there are several channels
3rd field is the time step in ms
4th field is the Voltage in mV

In **A5-eeg.bash**

9. Extract all the lines of **eeg1.dat** that contain either channel FP1 or channel FP2 and save the output into a file called FP12.dat. Use options of grep

10. Extract the line of **eeg.dat** containing the channel FP1 at time step 1. You can grep the number 1 by using **" 1 "**. Use two greps in pipeline.

11. Optional- Use command substitution to store the highest observed voltage in a variable called vmax. Use eeg1.dat

12 – Optional . Use variable vmax to print to screen this formatted output by using printf.

The highest recorded voltage is 34.3 mV

13 – Optional . Find a way to make a file called channels.dat containing a list of unique channel names. You should achieve this with one line of code, and by using cut and the -u option of the sort command in pipeline. Keep in mind that –u will remove lines (can have multiple fields) that are identical. Think about if you should use cut before sort in pipeline. Use eeg1.dat

Submit to A5

- **A5-stream.bash 1-8 mandatory 1.5 points**
- **A5-eeg.bash 1-10 mandatory 1.5 points**