Défi BigData
Information Systems for Big Data 1
– Written Exam –

Luis Gustavo Nardin
<gnardin@emse.fr>

18 April 2025

Duration : 2 hours (9h00–11h00)

Instructions

- You are **allowed** to use an *A4 sheet of paper*, *handwritten by you*.
- You are **forbidden** to use calculators, telephones, computers or any other means of communication.
- Write your **full name** in the first page of this document.
- Write your **answers directly on this document** and hand it in in its entirety at the end of the allotted time.
- The **multiple choice questions** have **only one correct answer**.

Name	

Grade :

Remarks :

Question 1 : (1 point) What is `who` in the command `LANG=C man -k who` :

☐ an argument to the option `-k`

☐ a long option

X an argument to the program

☐ a relative path

Question 2 : (1 point) When executing a command, the command is searched in the directories listed in the PATH environment variable :

X from left to right

☐ in no particular order

☐ from right to left

☐ including only the first 256 characters

Question 3 : (1 point) Which is NOT a valid command to declare the environment variable `ENV_VAR` ?

☐ `export ENV_VAR=1`

X `newvar -x ENV_VAR=1`

☐ `ENV_VAR=1`

☐ `declare -x ENV_VAR=1`

Question 4 : (1 point) Which command allows us to copy the file `notes.txt` located in the directory `tmp` in the current directory to the directory `/home/user` with the name `new_notes.txt` ?

☐ `cp /tmp/notes.txt /home/user/new_notes.txt`

☐ `cp /tmp/notes.txt home/user/new_notes.txt`

X `cp tmp/notes.txt /home/user/new_notes.txt`

☐ `cp tmp/notes.txt  /new_notes.txt`

Question 5 : (1 point) The `#!/bin/bash` in the first line of a Shell script

☐ specifies the path of the file in which the result of the script will be written

☐ specifies the path where the script is stored

☐ is a comment

X specifies the executable shell to use for running the script

Question 6 : (1.5) Write the regular expression that matches all the patterns below.

Hello World
hello world
world hello
WORLD HELLO
hello hello world
hello world world

^Hello World$|^hello world$|^world hello$|^WORLD HELLO$|^hello hello world$|^hello world world$

Question 7 : (1.5 points) The file `games.txt` in your current directory has the following content

2024-08-16	Nimes vs Orléans	Loss	1-2
2024-08-23	Nimes vs Châteauroux	Win	1-0
2024-08-30	Valenciennes vs Nimes	Win	2-0
2024-09-06	Nimes vs Rouen	Match	2-2
2024-09-13	Concameau vs Nimes	Win	1-0
2004-09-20	Nimes vs Versailles	Win	1-0

Write a command or chain of commands (NOT a script) to find all games in which *Nimes* won when playing at home. The team playing at home is the team name before "vs".

grep "Nimes vs" games.txt | grep Win

Question 8 : (2 points) The command `who` returns the list of users logged on in the computer in the format

gnardin	tty1	2025-04-15 18:03	
gnardin	pts/1	2025-04-15 18:03	(tmux(1613).%0
asmith	tty2	2025-03-10 13:53	
wblack	tty3	2025-04-16 11:09	

Write a command or chain of commands (NOT a script) to create the file `logged_on.txt` which contains a listing of the unique user names currently logged on in inverted alphabetical order.

who | cut -d' ' -f1 | uniq | sort -r > logged_on.txt

Question 9 : (3.0 points) Suppose you have the `users.csv` file with the content below.

```
id,first-name,last-name,email,login,logout,cost
1,John,Smith,john.smith@example.com,2023-01-15,2023-03-20,125.99
2,Jane,Doe,jane.doe@example.com,2023-01-16,2023-03-21,210.50
3,Bob,Johnson,bob@example.com,2023-01-17,2023-03-22,0
4,Alice,Williams,alice.williams@example.com,2023-01-18,,75.25
5,,Brown,mike.brown@example.com,2023-01-19,2023-03-24,150.75
6,Sarah,Miller,sarah.miller@example.com,invalid_date,2023-03-25,95.00
7,David,Jones,david.jones@example.com,2023-01-21,2023-03-26,300.00
8,Lisa,Garcia,lisa.garcia@example.com,2023-01-22,2023-03-27,-50.00
9,James,Martinez,mymail@example.com,2023-01-23,2023-03-28,125.00
```

Write an *awk* script that

1. Drop the heading line (i.e., first line)
2. Fill in empty name fields (i.e., `first-name` and `last-name`) with the word `Unknown`
3. Check if the dates are properly formatted, replacing those not properly formatted by: `login` field with `2023-01-20` and `logout` with `2023-01-23`
4. Replace negative costs by `0.0`

```
awk -F ',' 'BEGIN{OFS=","}
      NR>1{
        if ($2=="") {$2="Unknown"}
        if ($3=="") {$3="Unknown"}
        if ($5 !~ /[0-9][0-9][0-9][0-9]-[01][0-9]-[0-3][0-9]/) {$5="2023-01-20"}
        if ($6 !~ /[0-9][0-9][0-9][0-9]-[01][0-9]-[0-3][0-9]/) {$6="2023-01-23"}
        if ($7<0) {$7=0}
        print $1,$2,$3,$4,$5,$6,$7
      }' users.csv
```

Question 10 : (3 points) Explain each line of code in the script below (arguments, processing, results, etc.).

```
#!/bin/bash

if [[ $# -ne 1 ]]
then
  echo "Usage: ${0} nameList" >&2
  exit 1
fi

if [[ ! -f "$1" ]]
then
  echo "${1} error." >&2
  exit 1
fi

usernames=‘cat "$1"‘
for i in $usernames
do
  if [[ -d $i ]]
  then
    echo "${i} already exists." >&2
    exit 1
  fi

  mkdir $i
  chmod 775 $i
  chown $i $i
done
```

Line 1 : Define the shell interpreter
Line 3–7 : If the number of parameter is not equal 1, display a message and terminate the script with an error
Line 9–13 : If the parameter is not a file, display an error message and terminate the script with an error
Line 14 : Load the variable usernames with the content of the file identified in the parameter
Line 15–25 : For each entry in the usernames variable, checks if a directory with that username exists; if so display a message and terminates the script with an error, otherwise create a directory and change the permission and ownership.

Question 11 : (4 points) You have been hired to provide a solution to summarize and consolidate the data of the daily sales generated by the stores of a large retailer. The stores are geographically distributed all over the world and the headquarter is located in Saint-Étienne. The stores generate one file per day containing their sales in the format below.

Date_Time;Store;Product;Quantity;Unit_Price;Unit_Cost;Unit_Profit;Total_Price
02/10/2022 08:01;store10;p3;7;269.33;105.20;164.13;1885.28
02/10/2022 08:49;store10;p32;9;38.32;15.21;23.10;344.85
02/10/2022 08:54;store10;p2;4;414.34;211.85;202.49;1657.35
...

You are required to write a Bash script (no *awk* script only) that receives as input the directory where the daily files are stored. The script should check if the directory exists and return an error if not. Moreover, the script must contain a function that receives a full path to a file containing the sale records, summarizes the daily sales by product and writes the result to the same directory with name `summary.txt`.

```
#!/bin/bash

dir=$1
if [ ! -d dir ]; then
    echo "$1 does not exist"
    exit 1
fi

summarize() {
    product=$(cut -d ";" -f3 $1 | sort | uniq)
    for i in $product;
    do
        awk -F ";" -v prod="$i" 'BEGIN{total=0};
             NR>1{
               if ($3==prod) {total+=$8}
             };
             END{print prod" "total}' $1
    done
}

for f in `ls $dir`;
do
    summarize $dir/$f >> $dir/summary.txt
done
```

Command References

```
CAT(1)                          User Commands                          CAT(1)

NAME
       cat - concatenate files and print on the standard output

SYNOPSIS
       cat [OPTION]... [FILE]...
```

```
GREP(1)                         User Commands                         GREP(1)

NAME
       grep, egrep, fgrep, rgrep - print lines that match patterns

SYNOPSIS
       grep [OPTION...] PATTERNS [FILE...]
       grep [OPTION...] -e PATTERNS ... [FILE...]

DESCRIPTION
       grep  searches  for  PATTERNS in each FILE.  PATTERNS is one or
       more patterns separated by newline characters, and grep  prints
       each line that matches a pattern.  Typically PATTERNS should be
       quoted when grep is used in a shell command.

       A FILE of ‘‘-’’ stands for standard input.  If no FILE is  given,
       recursive   searches   examine   the   working   directory,   and
       nonrecursive searches read standard input.

       Debian also includes the  variant  programs  egrep,  fgrep  and
       rgrep.   These  programs  are the same as grep -E, grep -F, and
       grep -r, respectively.  These variants are deprecated upstream,
       but Debian provides for backward compatibility. For portability
       reasons, it is recommended to avoid the variant  programs,  and
       use grep with the related option instead.

OPTIONS
   Pattern Syntax
       -E, --extended-regexp
              Interpret   PATTERNS  as  extended  regular  expressions
              (EREs, see below).

   Matching Control
       -v, --invert-match
              Invert  the  sense  of  matching, to select non-matching
              lines.

   General Output Control
       -c, --count
              Suppress  normal  output;  instead  print  a  count   of
              matching  lines  for  each  input  file.   With  the -v,
              --invert-match option (see  above),  count  non-matching
              lines.

       -o, --only-matching
              Print  only  the matched (non-empty) parts of a matching
              line, with each such part on a separate output line.
```

```
SORT(1)                         User Commands                         SORT(1)

NAME
       sort - sort lines of text files

SYNOPSIS
       sort [OPTION]... [FILE]...
       sort [OPTION]... --files0-from=F

DESCRIPTION
       Write sorted concatenation of all FILE(s) to standard output.

       With no FILE, or when FILE is -, read standard input.
```

Mandatory arguments to long options are mandatory for short options too. Ordering options:

-d, --dictionary-order
consider only blanks and alphanumeric characters

-h, --human-numeric-sort
compare human readable numbers (e.g., 2K 1G)

-n, --numeric-sort
compare according to string numerical value

-r, --reverse
reverse the result of comparisons

-t, --field-separator=SEP
use SEP instead of non-blank to blank transition

-u, --unique
with -c, check for strict ordering; without -c, output only the first of an equal run

TEE(1) User Commands TEE(1)

NAME
tee - read from standard input and write to standard output and files

SYNOPSIS
tee [OPTION]... [FILE]...

TR(1) User Commands TR(1)

NAME
tr - translate or delete characters

SYNOPSIS
tr [OPTION]... STRING1 [STRING2]

DESCRIPTION
Translate, squeeze, and/or delete characters from standard input, writing to standard output. STRING1 and STRING2 specify arrays of characters ARRAY1 and ARRAY2 that control the action.

-c, -C, --complement
use the complement of ARRAY1

-d, --delete
delete characters in ARRAY1, do not translate

-s, --squeeze-repeats
replace each sequence of a repeated character that is listed in the last specified ARRAY, with a single occurrence of that character

-t, --truncate-set1
first truncate ARRAY1 to length of ARRAY2

--help display this help and exit

--version
output version information and exit

ARRAYs are specified as strings of characters. Most represent themselves. Interpreted sequences are:

\NNN character with octal value NNN (1 to 3 octal digits)
\\ backslash
\a audible BEL
\b backspace
\f form feed

\n new line
\r return
\t horizontal tab
\v vertical tab

CHAR1-CHAR2
all characters from CHAR1 to CHAR2 in ascending order
[CHAR*]
in ARRAY2, copies of CHAR until length of ARRAY1
[CHAR*REPEAT]
REPEAT copies of CHAR, REPEAT octal if starting with 0
[=CHAR=]
all characters which are equivalent to CHAR

Translation occurs if -d is not given and both STRING1 and STRING2 appear. -t may be used only when translating. ARRAY2 is extended to length of ARRAY1 by repeating its last character as necessary. Excess characters of ARRAY2 are ignored. Character classes expand in unspecified order; while translating, [:lower:] and [:upper:] may be used in pairs to specify case conversion. Squeezing occurs after translation or deletion.

NAME
uniq - report or omit repeated lines

SYNOPSIS
uniq [OPTION]... [INPUT [OUTPUT]]

DESCRIPTION
Filter adjacent matching lines from INPUT (or standard input), writing to OUTPUT (or standard output).

With no options, matching lines are merged to the first occurrence.

-c, --count
prefix lines by the number of occurrences

Note: 'uniq' does not detect repeated lines unless they are adjacent. You may want to sort the input first, or use 'sort -u' without 'uniq'.

NAME
wc - print newline, word, and byte counts for each file

SYNOPSIS
wc [OPTION]... [FILE]...
wc [OPTION]... --files0-from=F

DESCRIPTION
Print newline, word, and byte counts for each FILE, and a total line if more than one FILE is specified. A word is a non-zero-length sequence of printable characters delimited by white space.

With no FILE, or when FILE is -, read standard input.

The options below may be used to select which counts are printed, always in the following order: newline, word, character, byte, maximum line length.

-c, --bytes
print the byte counts

-m, --chars
print the character counts

-l, --lines
print the newline counts