Assignment 1: C program to monitor different resources

Due Wednesday by 11:59pm **Points** 9 **Submitting** a file upload
File Types c, zip, tar, gz, and tar.gz

Write a C program that will report different metrics of the utilization of a given system as described below.

You should target this program to work in a Linux type OS, e.g. the workstations from the BV 473 lab. **I.e. if your code does not compile and/or run in the lab machines will receive a zero.**

The program should accept several command line arguments:

`--system`

to indicate that only the system usage should be generated

`--user`

to indicate that only the users usage should be generated

`--graphics`

to include graphical output in the cases where a graphical outcome is possible as indicated below.

`--samples=N`

if used the value *N* will indicate how many times the statistics are going to be collected and results will be average and reported based on the N number of repetitions.
If not value is indicated the default value will be 10.

`--tdelay=T`

to indicate how frequently to sample in seconds.
If not value is indicated the default value will be 1 sec.

The last two arguments can also be considered as *positional arguments* if not flag is indicated in the corresponding order: `samples tdelay`.

The reported "stats" should include:

- *user usage*
  - report how many users are connected in a given time
  - report how many sessions each user is connected to
- *system usage*
  - report how much utilization of the CPU is being done
  - report how much utilization of memory is being done (report used and free memory)
  - if the `--graphics` flag is used, generate a graphical representation showing the variation of memory used

Graphical representations

The following conventions were used while displaying the graphical outputs:

- for Memory utilization:

```
::::::@  total relative negative change
######*  total relative positive change

(OPTIONAL)
|o    zero+
|@    zero-
```

- for CPU utilization:

```
||||   positive percentage increase
```

General remarks:

- write proper **modular** code, i.e. with functions that have clearly specified goals and tasks using proper arguments and parameters
- do **not** use global variables
- include comments and documentation
- **Avoid** using any shell command to be run through your C program.
  Instead implement the required functionalities using C coding and the references mentioned below.

While working on the assignment, you may want to complement the material presented in class with more details on:

- command line arguments parsing
- files and strings manipulation
- depending on how end up doing the actual implementation, "**ESCape codes**" for managing different elements to output to screen, e.g. positioning, refreshing, cleaning, etc.
- auxiliary standard C-libraries to consider using/investigating (see the corresponding refs or the `man` pages):
  - `sys/resource.h` -- https://man7.org/linux/man-pages/man0/sys_resource.h.0p.html
  - `sys/utsname.h` -- https://man7.org/linux/man-pages/man2/uname.2.html
  - `sys/sysinfo.h` -- https://man7.org/linux/man-pages/man2/sysinfo.2.html
  - `sys/types.h` -- https://man7.org/linux/man-pages/man0/sys_types.h.0p.html
  - `utmp.h` -- https://man7.org/linux/man-pages/man5/utmp.5.html
  - `unistd.h` -- https://man7.org/linux/man-pages/man0/unistd.h.0p.html
- recall that in Linux EVERYTHING is a FILE, and some useful ones for this assignment are:
  - `/proc/cpuinfo`
  - `/proc/stat`

Examples of a possible implementation for different ways of running our monitoring tool:

`$ ./mySystemStats`

```
Nbr of samples: 10 -- every 1 secs
 Memory usage: 4092 kilobytes
---------------------------------------
### Memory ### (Phys.Used/Tot -- Virtual Used/Tot)
```

```
9.78 GB / 15.37 GB  -- 9.78 GB / 16.33 GB
9.77 GB / 15.37 GB  -- 9.77 GB / 16.33 GB
9.77 GB / 15.37 GB  -- 9.77 GB / 16.33 GB
9.77 GB / 15.37 GB  -- 9.77 GB / 16.33 GB
9.77 GB / 15.37 GB  -- 9.77 GB / 16.33 GB
9.77 GB / 15.37 GB  -- 9.77 GB / 16.33 GB
9.77 GB / 15.37 GB  -- 9.77 GB / 16.33 GB
9.77 GB / 15.37 GB  -- 9.77 GB / 16.33 GB
9.77 GB / 15.37 GB  -- 9.77 GB / 16.33 GB
9.77 GB / 15.37 GB  -- 9.77 GB / 16.33 GB
---------------------------------------
### Sessions/users ###
 marcelo       pts/0 (138.51.12.217)
 marcelo       pts/1 (tmux(3773782).%0)
 alberto       tty7 (:0)
 marcelo       pts/2 (tmux(3773782).%1)
 marcelo       pts/3 (tmux(3773782).%3)
 marcelo       pts/4 (tmux(3773782).%4)
---------------------------------------
Number of cores: 4
 total cpu use = 0.00%
---------------------------------------
### System Information ###
 System Name = Linux
 Machine Name = iits-b473-01
 Version = #99-Ubuntu SMP Thu Sep 23 17:29:00 UTC 2021
 Release = 5.4.0-88-generic
 Architecture = x86_64
---------------------------------------
```

A similar output using the `--graphics` or `-g` flag would be:

```
$ ./mySystemStats --graphics
```

```
Nbr of samples: 10 -- every 1 secs
 Memory usage: 4052 kilobytes
---------------------------------------
### Memory ### (Phys.Used/Tot -- Virtual Used/Tot)
9.75 GB / 15.37 GB  -- 9.75 GB / 16.33 GB   |o 0.00 (9.75)
9.75 GB / 15.37 GB  -- 9.75 GB / 16.33 GB   |* 0.00 (9.75)
9.75 GB / 15.37 GB  -- 9.75 GB / 16.33 GB   |* 0.00 (9.75)
9.76 GB / 15.37 GB  -- 9.76 GB / 16.33 GB   |* 0.00 (9.76)
9.85 GB / 15.37 GB  -- 9.85 GB / 16.33 GB   |#########* 0.09 (9.85)
10.06 GB / 15.37 GB  -- 10.06 GB / 16.33 GB   |####################* 0.20 (10.06)
10.13 GB / 15.37 GB  -- 10.13 GB / 16.33 GB   |#######* 0.07 (10.13)
10.16 GB / 15.37 GB  -- 10.16 GB / 16.33 GB   |##* 0.03 (10.16)
10.28 GB / 15.37 GB  -- 10.28 GB / 16.33 GB   |############* 0.12 (10.28)
10.38 GB / 15.37 GB  -- 10.38 GB / 16.33 GB   |###########* 0.11 (10.38)
---------------------------------------
### Sessions/users ###
 marcelo       pts/0 (138.51.12.217)
 marcelo       pts/1 (tmux(277015).%0)
 alberto        tty7 (:0)
 marcelo       pts/2 (tmux(277015).%1)
 marcelo       pts/5 (138.51.12.217)
---------------------------------------
Number of cores: 4
 total cpu use = 15.57%
         ||| 0.25
```

```
         ||||||||| 6.93
         |||||||||||||| 12.08
         |||||||||||||| 13.83
         ||||||||| 6.41
         |||||||||||||| 13.97
         |||||||||||||||| 15.37
         |||||||||||||||| 14.91
         |||||||||||||||||| 16.34
         |||||||||||||||| 15.57
---------------------------------------
### System Information ###
 System Name = Linux
 Machine Name = iits-b473-01
 Version = #99-Ubuntu SMP Thu Sep 23 17:29:00 UTC 2021
 Release = 5.4.0-88-generic
 Architecture = x86_64
---------------------------------------
```

Submission

- Submit your code, and
- Include a report, e.g. it could be a **Readme** file explaining:
  - how did you solve the problem
  - an overview of the functions (including documentation)
  - how to run (use) your program

Late submissions

- Late submissions will be accepted up to 5 days after the deadline with the **penalty of 5% off per day late**.

File Upload | O365 OneDrive

Upload a file, or choose a file you've already uploaded.

Choose File no file selected

+ Add Another File

Click here to find a file you've already uploaded

Comments...

Cancel Submit Assignment

19

To get CPU usage, periodically sample the *total* process time, and find the difference.

For example, if these are the CPU times for process 1:

```
kernel: 1:00:00.0000
user:   9:00:00.0000
```

And then you obtain them again two seconds later, and they are:

```
kernel: 1:00:00.0300
user:   9:00:00.6100
```

You subtract the kernel times (for a difference of `0.03`) and the user times (`0.61`), add them together (`0.64`), and divide by the sample time of 2 seconds (`0.32`).

So over the past two seconds, the process used an average of 32% CPU time.

The specific system calls needed to get this info are (obviously) different on every platform. On Windows, you can use GetProcessTimes, or GetSystemTimes if you want a shortcut to *total* used or idle CPU time.

A fully working example is worth a thousand words. ;-)

20

```c
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <sys/utsname.h>

int main(void) {

   struct utsname buffer;

   errno = 0;
   if (uname(&buffer) < 0) {
      perror("uname");
      exit(EXIT_FAILURE);
   }

   printf("system name = %s\n", buffer.sysname);
   printf("node name   = %s\n", buffer.nodename);
   printf("release     = %s\n", buffer.release);
   printf("version     = %s\n", buffer.version);
   printf("machine     = %s\n", buffer.machine);

   #ifdef _GNU_SOURCE
      printf("domain name = %s\n", buffer.domainname);
   #endif

   return EXIT_SUCCESS;
}
```