EEE8126 – EMBEDDED SYSTEMS AND PROGRAMMING
PROJECT 1: REAL-TIME PARSING USING C++ POINTERS AND STRUCTURES/CLASSES

TOTAL MARKS: 100

Objectives

1. To understand how real-time parsing can be carried out using C++.
2. To develop a strong working experience of using C++ pointers and structures.

The aim of this assignment is to write your own Linux system information parser reading the file "/proc/stat". Various pieces of information about operating system's activity are available in the /proc/stat file, including CPU activity, number of cores available in the system, and so on. All of the numbers reported in this file are aggregates since the system first booted. For a quick look, simply cat the file in *Cygwin bash shell* (on the Computing Lab systems only; otherwise you are going to see different versions of this):

```
> cat /proc/stat
```

```
cpu 194138531 0 66809735 10852009959
cpu0 53110727 0 20893572 2704235573
cpu1 38159108 0 12035040 2728045428
cpu2 57820709 0 17624914 2702793827
cpu3 45047987 0 16256209 2716935131
page 19197419 47547443
swap 19197419 47476678
intr 3609666743
ctxt 2647568339
btime 1446127737
```

The very first "cpu" line aggregates the numbers in all of the other "cpuN" lines. These numbers identify the amount of time the CPU has spent performing different kinds of work. Time units are in hundredths of a second (i.e. 1/100 seconds).

The meanings of the columns are as follows, from left to right:

cpu user nice system idle

user: normal processes executing in user mode
nice: niced (i.e. changed priority) processes executing in user mode
system: operating system processes executing in kernel mode
idle: idle times in all modes

The cpu lines continue for as many cores as there are, as

cpuN user nice system idle

The “page” line reports the number of pages written in and out to the disk.

The “swap” line reports the number of swap pages written in and out to the disk.

The "intr" line gives counts of interrupts serviced since boot time, for each of the possible system interrupts. The first column is the total of all interrupts serviced; each subsequent column (if you can see them) is the total for that particular interrupt.

The "ctxt" line gives the total number of context switches across all CPUs. Context switching happens when a task running on a CPU switches to another CPU due to task pre-emption carried out by the operating system from time to time.

The "btime" line gives the time at which the system booted, in seconds since the Unix epoch (00:00:00 UTC Thursday 1 January, 1970).

Your program should have the following features:

1. Read each line as strings from “/proc/stat” and use pointer arithmetic to parse the lines (i.e. access and convert from string of characters to their respective types)
   [**Max marks** 20 marks]
2. Use classes to organise the information of each component, such as CPU, page, swap, intr, and ctxt
   [**Max marks** 10 marks]
3. Process the information to present as follows (including –s and =s)

```
BEGIN
-------------------------------------------------------------------------------------------
CPU Cores: 4
-------------------------------------------------------------------------------------------
CPU         busy        nice        system              idle
=========================================================================
CPU0        1.9%        0.0%        0.8%        97.3%
CPU1        1.4%        0.0%        0.4%        98.2%
CPU2        2.1%        0.0%        0.6%        97.3%
CPU3        1.6%        0.0%        0.6%        97.8%
-------------------------------------------------------------------------------------------
Page in/out ratio: 0.403753
Swap in/out ratio: 0.404355
Interrupts serviced: 3.61 billions since booting
Context switch counts: 2.65 billions since booting
-------------------------------------------------------------------------------------------
END
```

Note that the presentation will need to be similar (for example 1-digit precision after decimal point for the time stats and in %, 2-digit precision for the numbers of interrupts and context switches with billions to round up to an approximate number). The above numbers are generated using the example stats of "/proc/stats" shown earlier – a demo calculation of these stats are shown in the stats-arithmetic.xlsx file attached.

Your program must update the stats real-time. For this you will need an outer loop to read the "/proc/stat" each time and carry out the necessary conversions, and parsing within the loop. After displaying the output in each instance, you should ask the system to delay the next reading and stats update by 500 milliseconds. An example of sleep based delay process is given in: http://linux.die.net/man/3/usleep

Additional hint: You can read file in exactly the same way as you did in the first assignment, except that now you will need to open the file using read mode (i.e. "r", in place of "w") and the file must exist. The file you are reading must be explicitly specified with directories to be able to read the file. A worked out example with these features can be found in file_pointer_example.c (see attached)

[**Max marks** Correct codes: 50 + Correct presentation: 20= 70 marks]

Deliverables

You should submit a single C++ file with the correct program. No report is necessary.

Marking

This part of the module is assessed by 50% programming assessment. This assignment will constitute 25% of the total mark for the module. The marks' distribution is already shown in the problem descriptions, totalling 100. The actual mark will be scaled to the module's marks afterwards.

MARKING INSTRUCTIONS

Correct, and properly commented code with correct presentation will merit up to 80% marks. Memory efficient code will merit another 20% marks.

Plagiarism is strictly prohibited as it may result in serious penalties. There would be informal tutorial discussions on this assignment after the regular lecture hours. **The submission deadline is strictly 7 Dec 2018**, after which the submissions would be considered late and usual late submission rules would apply.

Feedback

After your assignment has been marked, feedback is provided as follows:

1) *Blackboard*: the document submitted has been annotated with little 'blue balloons' in the usual manner. I will notify you when this information becomes available.

2) *Email*: Shortly after the Blackboard feedback becomes available, a feedback sheet (a single page pdf file) with a detailed breakdown of your marks will be communicated to everyone.