System Monitoring Tool [CSCB09]

Jeong Yuseon

February 7, 2025

Contents

1 Introduction

This document provides an overview of the implementation and functionality of the System Monitoring Tool. The tool reports real-time utilization metrics for:

- Memory usage
- CPU utilization
- Number of CPU cores

It runs under the following command:

```
gcc -Wall -std=c99 -Werror Assignment1.c Assignment1
```

2 Dependencies

Before running the code, ensure that the following dependencies are installed:

- `#include <stdio.h>` - Standard I/O functions.
- `#include <stdlib.h>` - Memory management and utilities.
- `#include <string.h>` - String manipulation.
- `#include <stdbool.h>` - Boolean types.
- `#include <unistd.h>` - POSIX system calls.
- `#include <ctype.h>` - Character classification.
- `#include <sys/resource.h>` - CPU utilization calculations.
- `#include <sys/sysinfo.h>` - System memory details.

3 Usage of `usleep()`

Important Notes:

- `usleep()` is included in `unistd.h` and works in most Linux environments.
- While `usleep()` is officially deprecated in POSIX.1-2001, it is assumed to be allowed for this assignment.
- The program does not require compiling with `gcc -std=c99`, allowing `usleep()` to function correctly.

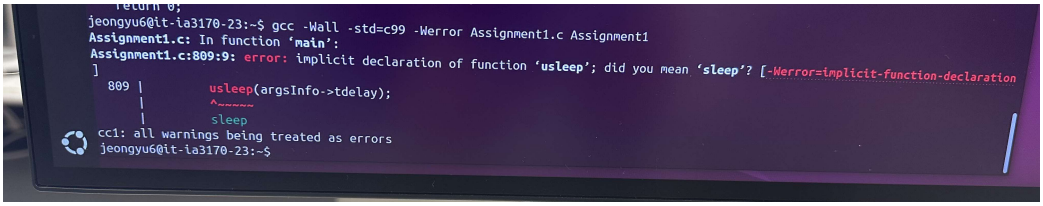

Figure 1: Usleep error in IA3170-23 computer

4 Struct Definitions

4.0.1 ArgsInfo

Stores information related to command-line arguments and program execution flags.

- `memory_flag` - Boolean flag for memory monitoring.
- `cpu_flag` - Boolean flag for CPU utilization monitoring.
- `cores_flag` - Boolean flag for CPU core monitoring.
- `samples` - Number of samples for monitoring.
- `tdelay` - Time delay (in microseconds) between samples.
- `updated_sample` - Flag to track sample count updates.
- `updated_tdelay` - Flag to track delay updates.
- `argc` - Number of command-line arguments.
- `argv` - Pointer to command-line arguments array.

4.0.2 CursorPosition

Tracks the cursor's current row and column position in the terminal for proper alignment and updates.

- `row` - The cursor row index.
- `col` - The cursor column index.

5 Main System Monitoring Implementation

5.0.1 Parameters:

- `argc` - The number of command-line arguments.
- `argv` - The array of command-line argument strings.

5.0.2 Key Variables Used:

- `ArgsInfo *argsInfo` - Stores parsed command-line arguments.
- `CursorPosition memory, cpu, cores` - Tracks graph positions.
- `long double preTotalCPU, preIdleCPU, finalTotalCPU, finalIdleCPU` - Stores CPU utilization data.
- `double max_frequency` - Stores the maximum CPU frequency.

5.0.3 Execution Flow:

1. Parse and validate command-line arguments.
2. Initialize the display and draw graphs based on user flags.
3. Continuously collect and update memory/CPU utilization data.
4. If enabled, display CPU core information at the end.
5. Restore the terminal cursor and free allocated memory.

6 Processing Command-Line Arguments

The program handles both positional and flagged arguments. The expected usage format is:

```
./myMonitoringTool [samples [tdelay]] [--memory] [--cpu]
[--cores] [--samples=N] [--tdelay=T]
```

6.0.1 Key rules:

- `samples` (optional) and `tdelay` (optional) must appear in 2nd and 3rd positionsof `argv[]`.
- Flagged arguments (‘–memory‘, ‘–cpu‘, ‘–cores‘) can appear in any order.
- ‘samples‘ and ‘tdelay‘ must be positive integers, excluding **0**
- Multiple modifications of ‘samples‘ and ‘tdelay‘ are **not allowed**.

6.0.2 processCommandLineArguments()

Parameters:

- `argc` - The number of command-line arguments.
- `argv` - An array of command-line argument strings.

Pseudocode:

1. If `argc == 1`, enable all graphs (memory, CPU, and cores).
2. If `argc > 1`:
   (a) If `argv[1]` is a value:
      i. Set it as the `samples` positional argument.
      ii. If `argv[2]` is also a positional argument, set it as `tdelay`.
      iii. Otherwise, proceed to checking if it's a flagged argument.
   (b) Else, check if it is a flagged argument:
      i. If it's a valid flag, process it.
      ii. Otherwise, return an error.
3. If no flags are enabled, enable all graphs by default.

Returns:

- `-1` - Failed to process the command line arguments
- `0` - Successfully processed the command line arguments

6.0.3 isPositional()

Parameters:

- `argsInfo` - The struct containing command line arguments.
- `current_index` - The index of the command-line arguments currently being processed.

Documentation:

- Checks if all individual characters in the string are digits

Returns:

- `TRUE` - If all characters are digits.
- `FALSE` - If at least one character is not a digit.

6.0.4 isFlag()

Parameters

- `argsInfo` - A struct containing flags to indicate which graph to show along with the total number of samples or tdelay values.
- `current_index` - Stores the current index of the command-line argument being processed.

Assumptions made:

- `--flagname=`
  - All flags will be written without any space between `--flagname` and the `=` sign.
  - The value must immediately follow the flag `--flagname= N` is invalid
- `--samples=N --tdelay=T`
  - `N` and `T` must be positive integers greater than zero.
  - Any non-numeric values will be considered invalid input.
  - Any spaces between the –samples = N is also considered invalid

Documentation:

1. If `argsInfo` is `NULL` or there is no text in the current index, return `FALSE`.
2. If `argv` is `"--memory"`:
   - Set `memoryflag` in `argsInfo` to `true`.
   - Return `TRUE`.
3. If `argv` is `"--cpu"`:
   - Set `cpuflag` in `argsInfo` to `true`.
   - Return `TRUE`.
4. If `argv` is `"--cores"`:
   - Set `coresflag` in `argsInfo` to `true`.
   - Return `TRUE`.
5. If `argv` starts with `"--samples="`:
   (a) Extract `value_str` from the string right after `"--samples="`. If spaces are present between = and the value, then it is considered invalid
   (b) Ensures that this is the first change in sample value, is greater than 0 and is a digit.
6. If `argv` starts with `"--tdelay="`:
   (a) Extract `value_str` from the string after `"--tdelay="`. If spaces are present between = and the value, then it is considered invalid
   (b) Ensures that this is the first change in tdelay value, is greater than 0 and is a digit.

Returns:

- `TRUE` if the argument is successfully processed.
- `FALSE` if the argument is invalid.

7 Fetching System Statistics

The program retrieves system statistics using system files and function calls:

7.1 CPU utilization

- Opens `/proc/stat` to retrieve CPU time statistics.
- Tracks the total CPU time and idle time at each sample interval.
- Stores previous total CPU time (`preTotalCPU`) and idle time (`preIdleCPU`) for accurate calculations.
- For the first sample, initializes `preTotalCPU` and `preIdleCPU` without computing utilization.
- For subsequent samples, calculates CPU utilization as:
- CPU Utilization (%) =

$$\left(\frac{U_2 - U_1}{T_2 - T_1}\right) \times 100$$

  where:

  - T_1, I_1, U_1 are total, idle, and usage times at the first sampling.
  - T_2, I_2, U_2 are total, idle, and usage times at the second sampling.
  - $U = T - I$ represents the CPU usage time.
- Updates `preTotalCPU` and `preIdleCPU` for the next iteration.

7.2 CPU core count

- Using /proc/cpuinfo to retrieve CPU core count.
- Checks through the file to look for the word "processor" and adds up all its occurences to get the total number of cores

7.3 Memory used

- Using sysinfo() to obtain memory usage
- Convert the obtained values in bytes to gigabytes by dividing it using (1024* 1024 * 1024)
- Subtract the total memory from the free memory to get memory used

7.4 Maximum frequency

- Access /sys/devices/system/cpu/cpu0/cpufreq/cpuinfo_max_freq
- Convert the value obtained from the file from KHz to GHz by dividing it by 1,000,000

8 Drawing Real-Time Graphs

The program displays:

- The initial graph layout using `draw_graph()`.
- Live-updated Memory used and CPU utilization graphs in `main()`.
- A box representation of CPU cores using `coresGraph()`.

8.0.1 draw_graph()

Parameters:

- `label` - string of the heading label for the graph
- `unit` - the y-axis label for the graph
- height - has the total height needed for the graph
- baseline - contains the x-axis label
- current_row - has the current row location in the terminal
- current_column - has the current column location in the terminal
- samples - has the total number of samples to decide the total width of the graph

Assumptions

- if the samples are smaller than 20, the graph will draw the x-axis of 20 regardless
- if the samples is bigger than 20, it will draw a bigger x-axis to fit the required data needed

Documentation

- `Draws the initial structure of the graph, including the label, unit, height and baseline, based on the available memory usage, CPU utilization or sample count`

8.0.2 draw_cpu_graph()

Documentation:

- The graph height is fixed at 11 units.
- Each unit represents 10% increments where the last unit represents 100% of CPU utilization.

Return:

- A `CursorPosition` struct representing the final cursor position.

8.0.3 draw_memory_graph()

Documentation

- The graph height is fixed at 10 units.
- Each unit represents `total_memory / 10`.

Return:

- A `CursorPosition` struct representing the final cursor position.

8.0.4 draw_cpu_graph()

Documentation

- Each core is represented as a boxed unit.
- The maximum number of cores displayed per row is 4

Return:

- A `CursorPosition` struct representing the final cursor position.

9 How to Use and Examples of testing

- `./Assignment1` : Expected output is CPU, memory, and cores graph.
- `./Assignment1 --cpu --cores` : Expected output is CPU and cores graph.
- `./Assignment1 --cpu` : Expected output is CPU graph.

- `./Assignment1 20 4000` : Expected output is 20 samples and 4000 tdelay with all graphs.
- `./Assignment1 --cpu=20` : Error: Unknown argument.
- `./Assignment1 --samples=20 --samples=30` : Error: Cannot have multiple sample values. Error: Unknown argument.
- `./Assignment1 --samples=apple` : Error: Invalid value for `--samples`. Error: Unknown argument.
- `./Assignment1 --samples=-1` : Error: Invalid value for `--samples`. Error: Unknown argument.
- `./Assignment1 --tdelay=-1` : Error: Invalid value for `--tdelay`. Error: Unknown argument.
- `./Assignment1 --samples= 29` : Error: Missing value. Error: Unknown argument