CPU Usage Analyzer

Generated by Doxygen 1.9.4

Chapter 1

CPU Usage Analyzer

This is a CPU usage analyzer program that calculates and analyzes the CPU usage of the system. It consists of multiple components, including the reader, analyzer, printer, watchdog, and logger. The program reads the `/proc/stat` file to obtain CPU usage information, performs calculations, and presents the results.

1.1 How to Run

To run the CPU usage analyzer program, follow these steps:

1. Ensure that you have the required dependencies installed, including GCC (GNU Compiler Collection) and Python.
2. Clone or download this repository to your local machine.
3. Open a terminal and navigate to the project directory.
4. Build the program by running the following command:

   make cpuanalyzer
5. Once the compilation is successful, you can run the program by executing the following command:

   ./cpuanalyzer
6. The program will start analyzing the CPU usage and displaying the results.

1.2 Testing

To run the tests for the CPU usage analyzer, make sure you have Python installed on your system.

1. Open a terminal and navigate to the project directory.
2. Run the tests by executing the following command:

   make test

   This command will compile the program and run the test script (`test_cpuanalyzer.py`). The test cases will be executed, and the test results will be displayed.

1.3 Clean Up

To clean up the generated files, use the following command:

make clean

This command will remove the compiled program and object files from the project directory.
Feel free to explore the source code files (`cpuanalyzer.c`, `reader_cpuanalyzer.c`, `analyzer_cpuanalyzer.c`, `printer_cpuanalyzer.c`, `watchdog_cpuanalyzer.c`, `logger_cpuanalyzer.c`) for more details about the implementation of the CPU usage analyzer.
If you encounter any issues or have questions, please feel free to create an issue on the GitHub repository.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 5

Module Documentation

5.1 analyzer cpuanalyzer

This file implements analyzer cpuanalyzer.
This file implements analyzer cpuanalyzer.

Author

A.Voznesenskyi

Date

12.06.2023

5.2 cpuanalyzer

This file implements cpuanalyzer.
This file implements cpuanalyzer.

Author

A.Voznesenskyi

Date

06.12.2023

5.3 logger cpuanalyzer

This file implements logger cpuanalyzer.
This file implements logger cpuanalyzer.

Author

A.Voznesenskyi

Date

06.12.2023

5.4 printer cpuanalyzer

This file implements printer cpuanalyzer.
This file implements printer cpuanalyzer.

Author

A.Voznesenskyi

Date

06.12.2023

5.5 reader cpuanalyzer

This file implements reader cpuanalyzer.
This file implements reader cpuanalyzer.

Author

A.Voznesenskyi

Date

06.12.2023

5.6 watchdog cpuanalyzer

This file implements watchdog cpuanalyzer.
This file implements watchdog cpuanalyzer.

Author

A.Voznesenskyi

Date

06.12.2023

Chapter 6

Class Documentation

6.1 kernel_proc_stat Struct Reference

Public Attributes

- char **name** [BUFFER_SIZE]
- U_L **user**
- U_L **nice**
- U_L **system**
- U_L **idle**
- U_L **iowait**
- U_L **irq**
- U_L **softirq**
- U_L **steal**
- U_L **guest**
- U_L **guest_nice**

The documentation for this struct was generated from the following file:

- /home/kaliuser/Documents/portfolio/cpuusageanalizer/cpuanalyzer.h

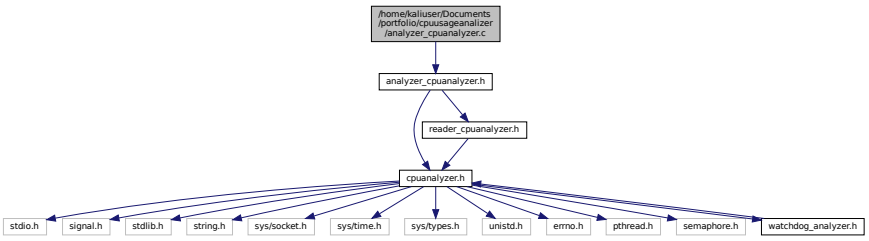
Chapter 7

File Documentation

7.1 /home/kaliuser/Documents/portfolio/cpuusageanalizer/analyzer_↩ cpuanalyzer.c File Reference

```
#include "analyzer_cpuanalyzer.h"
```

Include dependency graph for analyzer_cpuanalyzer.c:



Functions

- void * analyzer_proc_stat_thread (void *seq)

  analyzer_proc_stat_thread - Analyzer thread function
- U_L calculate_avarage_cpu_usage (struct kernel_proc_stat current, struct kernel_proc_stat previous)

  Calculates the avarage cpu usage.

7.1.1 Function Documentation

7.1.1.1 analyzer_proc_stat_thread()

```
void * analyzer_proc_stat_thread (
            void * seq )
```

analyzer_proc_stat_thread - Analyzer thread function

Parameters

seq	The sequence [unused parameter]

Returns

[unused return value]

7.1.1.2 calculate_avarage_cpu_usage()

```
U_L calculate_avarage_cpu_usage (
            struct kernel_proc_stat current,
            struct kernel_proc_stat previous )
```

Calculates the avarage cpu usage.

Parameters

current	The current state of kernel_proc_stat
previous	The previous state of kernel_proc_stat

Returns

The avarage cpu usage.

7.2 /home/kaliuser/Documents/portfolio/cpuusageanalizer/analyzer_cpuanalyzer.h

```
1 #ifndef ANALYZER_CPUANALYZER_H
2 #define ANALYZER_CPUANALYZER_H
3
4 #include "cpuanalyzer.h"
5 #include "reader_cpuanalyzer.h"
6
7 struct kernel_proc_stat;
8 void *analyzer_proc_stat_thread(void *seq);
9 U_L calculate_avarage_cpu_usage(struct kernel_proc_stat current, struct kernel_proc_stat previouis);
10
11 #endif /* ANALYZER_CPUANALYZER_H */
```

7.3 /home/kaliuser/Documents/portfolio/cpuusageanalizer/cpuanalyzer.h

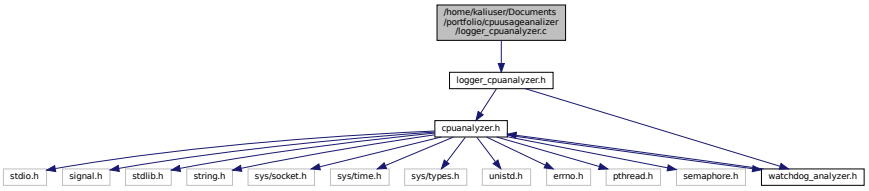
```
1 #ifndef CPUANALYZER_H
2 #define CPUANALYZER_H
3
4 #define _GNU_SOURCE
5 #include <stdio.h>
6 #include <signal.h>
7 #include <stdio.h>
8 #include <stdlib.h>
9 #include <string.h>
10 #include <sys/socket.h>
11 #include <sys/time.h>
12 #include <sys/types.h>
13 #include <unistd.h>
14 #include <signal.h>
15 #include <errno.h>
16 #include <pthread.h>
17 #include <semaphore.h>
18
19
20 #include "watchdog_analyzer.h"
21
22
23
24 #define U_L unsigned long
25
26 #define BUFFER_SIZE 10
27 #define ARRAY_BUFFER_SIZE 1024
28 #define S_TIME_SLEEP 2
29 #define THREADS_NUMBER 4
30
31 // Macro for handling errors with source indication:
32 #define ERR(source) { \
33 perror(source); \
34 fprintf(stderr, "%s:%d\n", __FILE__, __LINE__); \
35     exit(EXIT_FAILURE); \
36 }
37
38 #define HERR(source) { \
39 fprintf(stderr, "%s(%d) at %s:%d\n", source, h_errno, __FILE__, __LINE__); \
40     exit(EXIT_FAILURE); \
41 }
```

```
42
43 // struct THREAD_STATE
44 // {
45 //     int reader_thread;
46 //     int analizer_thread;
47 //     int printer_thread;
48 // };
49
50 //
51  enum THREAD_STATE {
52     reader_thread = 0,
53     analizer_thread = 1,
54     printer_thread = 2,
55     logger_thread = 3
56  };
57 // Set the initial state
58 // Miscellaneous kernel statistics in /proc/stat from the  https://docs.kernel.org/filesystems/proc.html:
59 struct kernel_proc_stat
60 {
61     char name[BUFFER_SIZE];
62     U_L user, nice, system, idle, iowait, irq, softirq, steal, guest, guest_nice;
63 };
64
65 extern volatile sig_atomic_t term_signal ;
66
67 extern int available_proc;
68 extern sem_t slots_filled_sem;
69 extern sem_t slots_empty_sem;
70
71 extern pthread_mutex_t bufferMutex;
72
73 extern struct kernel_proc_stat *array_stat[BUFFER_SIZE];
74
75
76 extern pthread_mutex_t data_mutex;
77
78 extern sem_t slots_filled_sem_printer;
79 extern sem_t slots_empty_sem_printer;
80 extern pthread_mutex_t print_bufferMutex;
81 extern U_L * print_buffer[BUFFER_SIZE];
82
83
84 extern pthread_mutex_t watchdog_bufferMutex;
85 extern int threads_to_watchdog[THREADS_NUMBER];
86
87 void usage(char *name);
88 int set_handler(void (*f)(int), int sigNo);
89 void sigterm_handler();
90 void readProcStat(FILE* file_to_read, struct kernel_proc_stat** stat, int* count_thread, int* read_cap);
91 void parseProcStatLine(char* line, struct kernel_proc_stat* stat);
92 void printProcStat(struct kernel_proc_stat* stat, int count_thread);
93 struct kernel_proc_stat *insert_to_array_stat();
94 U_L *insert_to_print_buffer();
95 void join_threads();
96 void cleanup_pthread_mutex_sem();
97 void cleanup();
98
99
100
101 #endif /* CPUANALYZER_H */
```

7.4 /home/kaliuser/Documents/portfolio/cpuusageanalizer/logger_↩ cpuanalyzer.c File Reference

`#include "logger_cpuanalyzer.h"`

Include dependency graph for logger_cpuanalyzer.c:



Functions

- void * logger_proc_stat_thread (void *seq)

  Logger thread function to log CPU statistics to a file.

7.4.1 Function Documentation

7.4.1.1 logger_proc_stat_thread()

```
void * logger_proc_stat_thread (
            void * seq )
```

Logger thread function to log CPU statistics to a file.

Parameters

seq	additional paramteter for synchronization

Returns

void

This function runs in a separate thread and continuously logs CPU statistics to a log file. It opens the log file in write mode and writes the data received from the print buffer. The thread runs until a termination signal is received. The function uses the slots_filled_sem_printer semaphore and the print_bufferMutex mutex to synchronize access to the print buffer. The log file is flushed after writing each data set to ensure immediate persistence of the logged data.

Note

The log file is created as "log_cpuanalyzer.txt" in the current directory.

If the log file cannot be opened or closed successfully, an error message is printed.

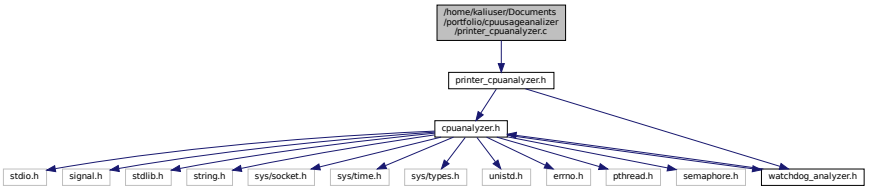
7.5 /home/kaliuser/Documents/portfolio/cpuusageanalizer/logger_↩ cpuanalyzer.h

```
1 #ifndef LOGGER_CPUANALIZER_H
2 #define LOGGER_CPUANALIZER_H
3
4 #include "cpuanalyzer.h"
5 #include "watchdog_analyzer.h"
6
7 void *logger_proc_stat_thread(void *seq);
8 #endif /* LOGGER_CPUANALIZER_H */
```

7.6 /home/kaliuser/Documents/portfolio/cpuusageanalizer/printer_↩ cpuanalyzer.c File Reference

```
#include "printer_cpuanalyzer.h"
```

Include dependency graph for printer_cpuanalyzer.c:



Functions

- void * printer_proc_stat_thread (void *seq)

  Printer thread function to print CPU statistics to the console.

7.6.1 Function Documentation

7.6.1.1 printer_proc_stat_thread()

```
void * printer_proc_stat_thread (
            void * seq )
```

Printer thread function to print CPU statistics to the console.

Parameters

seq	The sequence

Returns

void

This function runs in a separate thread and continuously prints CPU statistics to the console. It retrieves the data from the print buffer and prints it using the printf function. The thread runs until a termination signal is received. The function uses the slots_filled_sem_printer semaphore and the print_bufferMutex mutex to synchronize access to the print buffer. After printing the data, the thread sleeps for 1 second before processing the next set of data.

Note

The CPU statistics are printed in the format: "<processor_number>. <usage_percentage>"

The function does not perform error handling for the printf function.

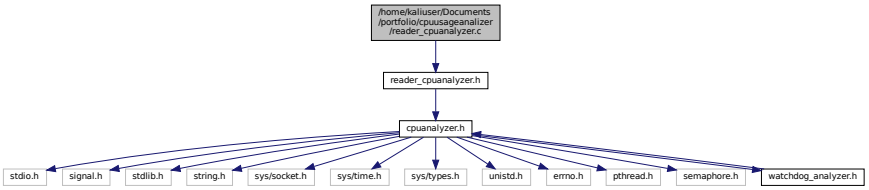
7.7 /home/kaliuser/Documents/portfolio/cpuusageanalizer/printer_↩ cpuanalyzer.h

```
1 #ifndef PRINTER_CPUANALIZER_H
2 #define PRINTER_CPUANALIZER_H
3
4 #include "cpuanalyzer.h"
5 #include "watchdog_analyzer.h"
6
7 void *printer_proc_stat_thread(void *seq);
8 #endif /* PRINTER_CPUANALIZER_H */
```

7.8 /home/kaliuser/Documents/portfolio/cpuusageanalizer/reader_↩ cpuanalyzer.c File Reference

```
#include "reader_cpuanalyzer.h"
```

Include dependency graph for reader_cpuanalyzer.c:



Functions

- FILE * open_proc_stat_file ()

  Function to open the /proc/stat file.
- char * check_token (char *token)

  unction to handle error checking for a token
- void set_kernel_proc_stat_values (struct kernel_proc_stat *stat, unsigned long values[ ], char *name)

  Function to set values in a kernel_proc_stat structure.
- int parse_proc_line (const char *line, struct kernel_proc_stat *stat)

  Function to parse a line from /proc/stat and store the values in a kernel_proc_stat structure.
- int get_proc_stat (struct kernel_proc_stat *stats)

  Function to read CPU statistics from /proc/stat file.
- void * read_proc_stat_thread (void *seq)

  Reader thread function to read CPU statistics from /proc/stat file.

7.8.1 Function Documentation

7.8.1.1 check_token()

```
char * check_token (
            char * token )
```

unction to handle error checking for a token

Parameters

token	The token to check

Returns

The input token if it is not NULL, otherwise NULL This function checks if the input token is NULL. If the token is NULL, a "Parsing error" message is printed, and NULL is returned. Otherwise, the input token is returned.

7.8.1.2 get_proc_stat()

```
int get_proc_stat (
            struct kernel_proc_stat * stats )
```

Function to read CPU statistics from /proc/stat file.

Parameters

stats	Pointer to the kernel_proc_stat array to store the read statistics

Returns

int 0 on success, -1 on failure

This function reads CPU statistics from the /proc/stat file. It opens the file using the open_proc_stat_file function and reads each line. The lines are parsed using the parse_proc_line function to extract the relevant statistics. The parsed statistics are stored in the kernel_proc_stat array. If any error occurs during the file reading or parsing, -1 is returned. On success, 0 is returned.

Note

The function reads one line for each CPU thread, excluding the overall CPU statistics.

The /proc/stat file is closed after reading.

7.8.1.3 open_proc_stat_file()

```
FILE * open_proc_stat_file ( )
```

Function to open the /proc/stat file.

Returns

FILE* Pointer to the opened file, or NULL if an error occurs This function opens the /proc/stat file in read mode and returns a pointer to the opened file. If the file cannot be opened, an error message is printed and NULL is returned. The caller is responsible for closing the file when no longer needed.

7.8.1.4 parse_proc_line()

```
int parse_proc_line (
            const char * line,
            struct kernel_proc_stat * stat )
```

Function to parse a line from /proc/stat and store the values in a kernel_proc_stat structure.

Parameters

line	The line from /proc/stat to parse
stat	Pointer to the kernel_proc_stat structure to store the parsed values

Returns

int 0 on success, -1 on failure

This function parses a line from the /proc/stat file. It uses sscanf to extract the values from the line and store them in the kernel_proc_stat structure. If the parsing fails or the number of parsed values is incorrect, an error message is printed, and -1 is returned. On success, 0 is returned.

7.8.1.5 read_proc_stat_thread()

```
void * read_proc_stat_thread (
            void * seq )
```

Reader thread function to read CPU statistics from /proc/stat file.

Parameters

seq	The sequence

Returns

void

This function runs in a separate thread and continuously reads CPU statistics from the /proc/stat file. It opens the /proc/stat file using the open_proc_stat_file function and reads each line. The lines are parsed using the parse↩ _proc_line function to extract the relevant statistics. The parsed statistics are then stored in the kernel_proc_stat structure. The thread runs until a termination signal is received. The function uses the slots_empty_sem semaphore and the bufferMutex mutex to synchronize access to the array_stat buffer. After reading the statistics, the thread suspends execution for a specified amount of time using the usleep function.

Note

The /proc/stat file is read line by line, and each line represents the statistics for a specific CPU thread.

The function handles error checking for file opening, parsing, and file closing.

7.8.1.6 set_kernel_proc_stat_values()

```
void set_kernel_proc_stat_values (
            struct kernel_proc_stat * stat,
            unsigned long values[],
            char * name )
```

Function to set values in a kernel_proc_stat structure.

Parameters

stat	Pointer to the kernel_proc_stat structure to set values in
values	Array of unsigned long values to set in the structure
name	string to set in the structure

Returns

void

This function sets the values in a kernel_proc_stat structure. The name is copied into the structure using strncpy. The values are assigned to the corresponding members of the structure.

7.9 /home/kaliuser/Documents/portfolio/cpuusageanalizer/reader_↩ cpuanalyzer.h

```
1 #ifndef READER_CPUANALIZER_H
2 #define READER_CPUANALIZER_H
3 #include "cpuanalyzer.h"
4
5
6 #define TIME_SUSPEND 50000
7
8 struct kernel_proc_stat;
9 FILE* open_proc_stat_file();
10 char* check_token(char *token);
11 void set_kernel_proc_stat_values(struct kernel_proc_stat* stat, unsigned long values[], char* name);
12 int parse_proc_line(const char* line, struct kernel_proc_stat* stat);
13 int get_proc_stat(struct kernel_proc_stat *stats);
14 void *read_proc_stat_thread(void *seq);
15
16 #endif /* READER_CPUANALIZER_H */
```

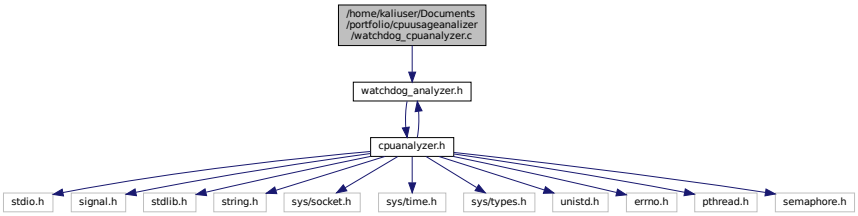
7.10 /home/kaliuser/Documents/portfolio/cpuusageanalizer/watchdog_↩ analyzer.h

```
1 #ifndef WATCHDOG_CPUANALYZER_H
2 #define WATCHDOG_CPUANALYZER_H
3
4 #include "cpuanalyzer.h"
5
6 void watchdog_notifier(int thread_id);
7 void watchdog_proc_stat_thread();
8
9 #endif /* WATCHDOG_CPUANALYZER_H */
```

7.11 /home/kaliuser/Documents/portfolio/cpuusageanalizer/watchdog_↩ cpuanalyzer.c File Reference

```
#include "watchdog_analyzer.h"
```

Include dependency graph for watchdog_cpuanalyzer.c:



Functions

- void watchdog_notifier (int thread_id)

  Function to notify the watchdog about a thread's activity.
- void watchdog_proc_stat_thread ()

  Watchdog thread function to monitor thread activity.

7.11.1 Function Documentation

7.11.1.1 watchdog_notifier()

```
void watchdog_notifier (
            int thread_id )
```

Function to notify the watchdog about a thread's activity.

Parameters

thread↩ _id	The ID of the thread to notify

Returns

void

This function notifies the watchdog about a thread's activity by setting the corresponding flag in the threads_to↩ _watchdog array. It acquires the watchdog_bufferMutex mutex before updating the flag to ensure thread safety.

7.11.1.2 watchdog_proc_stat_thread()

```
void watchdog_proc_stat_thread ( )
```

Watchdog thread function to monitor thread activity.

Returns

void

This function runs in a separate thread and continuously monitors the activity of the CPU analyzer threads. It sleeps for a specific time using the sleep function to allow the threads to perform their work. After the sleep period, it checks the threads_to_watchdog array to ensure that all threads have notified their activity. If any thread fails to notify its

activity, an error message is printed, the sigterm_handler function is called to initiate program termination, and the thread exits. The watchdog_bufferMutex mutex is used for thread safety during the activity check.