Obz C Scheduler

Generated by Doxygen 1.12.0

1 The C runtime for the <tt>OBZ Scheduler</tt> : 1

2 Class Index 3

2.1 Class List . . . 3

3 File Index 5

3.1 File List . . . 5

4 Class Documentation 7

4.1 GreenThread Struct Reference . . . 7

4.1.1 Detailed Description . . . 7

4.2 Scheduler Struct Reference . . . 7

4.2.1 Detailed Description . . . 8

5 File Documentation 9

5.1 include/anonymous.h File Reference . . . 9

5.1.1 Detailed Description . . . 9

5.1.2 Macro Definition Documentation . . . 9

5.1.2.1 lambda . . . 9

5.2 anonymous.h . . . 10

5.3 include/obz_scheduler.h File Reference . . . 10

5.3.1 Detailed Description . . . 10

5.3.2 Function Documentation . . . 11

5.3.2.1 green_thread_create() . . . 11

5.3.2.2 green_thread_run() . . . 11

5.3.2.3 setup_fault_tolerance_signal_handler() . . . 11

5.3.2.4 thread_wrapper() . . . 11

5.4 obz_scheduler.h . . . 11

5.5 include/scheduler.h File Reference . . . 12

5.5.1 Detailed Description . . . 12

5.5.2 Typedef Documentation . . . 12

5.5.2.1 Scheduler . . . 12

5.6 scheduler.h . . . 12

5.7 include/thread.h File Reference . . . 13

5.7.1 Detailed Description . . . 13

5.7.2 Typedef Documentation . . . 13

5.7.2.1 GreenThread . . . 13

5.7.3 Enumeration Type Documentation . . . 13

5.7.3.1 ThreadState . . . 13

5.8 thread.h . . . 14

5.9 src/fallback.c File Reference . . . 14

5.9.1 Detailed Description . . . 14

5.9.2 Function Documentation . . . 14

5.9.2.1 fallback() . . . 14

5.9.2.2 setup_fault_tolerance_signal_handler() . . . 15
5.10 src/fault_tolerance.c File Reference . . . 15
5.10.1 Detailed Description . . . 15
5.10.2 Function Documentation . . . 15
5.10.2.1 setup_handle_signals() . . . 15
5.11 src/scheduler.c File Reference . . . 15
5.11.1 Detailed Description . . . 16
5.11.2 Function Documentation . . . 16
5.11.2.1 green_thread_run() . . . 16
5.11.2.2 thread_wrapper() . . . 16
5.11.3 Variable Documentation . . . 16
5.11.3.1 scheduler . . . 16
5.12 src/thread.c File Reference . . . 17
5.12.1 Detailed Description . . . 17
5.12.2 Function Documentation . . . 17
5.12.2.1 green_thread_create() . . . 17

Index **19**

Chapter 1

The C runtime for the <tt>OBZ Scheduler</tt> :

1. Compile the project :

```
 sh
make clean
make
```

That will create a new two folders , `lib/` and `build/` , we will be concerned by our `lib/libobzruntime.a` since it's our statically linked runtime , that gonna be consumed by the transpiler.

Chapter 2

Class Index

2.1 Class List

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

GreenThread 7
Scheduler 7

Chapter 3

File Index

3.1 File List

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

include/anonymous.h 9
include/obz_scheduler.h 10
include/scheduler.h 12
include/thread.h 13
src/fallback.c 14
src/fault_tolerance.c 15
src/scheduler.c 15
src/thread.c 17

Chapter 4

Class Documentation

4.1 GreenThread Struct Reference

`#include <thread.h>`

Public Attributes

- ucontext_t **context**
- void * **stack**
- int **id**
- ThreadState **state**
- void(* **function**)(void *)
- void * **arg**

4.1.1 Detailed Description

GreenThread is a struct that holds metadata about green thread : context: is the field that contains snapshot of all process , memory related data whilst running the green thread (register's values , process register counter ...) . stack: contains the custom stack that will hold the local variables for the wrapped anonymous functions (not working properly) . state: the current state of the green thread . function: the function that the green thread in quesiton will run . arg: the argument that are gonna be fed to the wrapped function .

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

- include/thread.h

4.2 Scheduler Struct Reference

`#include <scheduler.h>`

Public Attributes

- GreenThread * **threads** [MAX_THREADS]
- int **thread_count**
- int **current_thread**
- ucontext_t **main_context**
- struct sigaction **old_action**
- struct itimerval **old_timer**
- bool **is_switching**

4.2.1 Detailed Description

scheduler is a struct that holds metadata about the scheduler global object , the one that will do the scheduling of the green thread : threads: is an allocated array that holds a pointer to greenthread objects current_thread: is the current thread with the `RUNNING` state , the concept of holding the number as an int is used mainly for indexing the threads[] array main_context : is the current ucontext_t of the current_thread (or threads[current_thread]) old↩ _timer: is used to set and hold the timer that's gonna be consumed by the kernel to send the `SIGALRM` signal old_action & is_switching : are not used !

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

- include/scheduler.h

Chapter 5

File Documentation

5.1 include/anonymous.h File Reference

```
#include "obz_scheduler.h"
```

Macros

- #define lambda(lambda_ret, lambda_args, lambda_body)

5.1.1 Detailed Description

Author

: Obz Team This file concerns

5.1.2 Macro Definition Documentation

5.1.2.1 lambda

```
#define lambda(
               lambda_ret,
               lambda_args,
               lambda_body)
```

Value:

```
({\
  lambda$_ret lambda$__anon$ lambda$_args\
    lambda$_body\
  &lambda$__anon$;\
})
```

This macro takes the return , the args (variable) and the body (as a block) and plugs it out and return the address of the function pointer to be consumed by the green_thread_create() function (takes it as a wrapper).

5.2 anonymous.h

Go to the documentation of this file.

```
00001
00009 #ifndef ANONYMOUS_H_
00010 #define ANONYMOUS_H_
00011 #include "obz_scheduler.h"
00012
00013
00019 #define lambda(lambda$_ret, lambda$_args, lambda$_body)\
00020   ({\
00021     lambda$_ret lambda$__anon$ lambda$_args\
00022       lambda$_body\
00023     &lambda$__anon$;\
00024   })
00025
00026
00027 #endif // ANONYMOUS_H_
```

5.3 include/obz_scheduler.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <ucontext.h>
#include <signal.h>
#include <string.h>
#include <stdbool.h>
#include <sys/time.h>
#include <unistd.h>
#include <fcntl.h>
```

Macros

- #define **STACK_SIZE** (1024 ∗ 1024)
- #define **MAX_THREADS** 64
- #define **TIME_SLICE_MS** 256

Functions

- void thread_wrapper (void)
- int green_thread_create (void(∗function)(void ∗), void ∗arg)
- void green_thread_run (void)
- void setup_fault_tolerance_signal_handler ()
- void **run** ()

5.3.1 Detailed Description

Author

: Obz Team This file contains the signature for the dynamically linked funtion that's gonna be used when compiling the sample program with our statically linked library. Each function is well documented in its appropriate file .

5.3.2 Function Documentation

5.3.2.1 green_thread_create()

```
int green_thread_create (
             void(* function )(void *),
             void * arg)
```

This function does create a wrapper around user-defined anonymous function and append it to the global thread array 'scheduler.threads[]'

5.3.2.2 green_thread_run()

```
void green_thread_run (
             void )
```

this function kickstarts the scheduler

5.3.2.3 setup_fault_tolerance_signal_handler()

```
void setup_fault_tolerance_signal_handler ()
```

This function does set the interput handler for bootstraping rudimentary fault tolerance for the failure of the Green threads

5.3.2.4 thread_wrapper()

```
void thread_wrapper (
             void )
```

this function does change the state of the green thread in order to execute the code in the anonymous function

5.4 obz_scheduler.h

Go to the documentation of this file.

```
00001
00009 #ifndef OBZ_SCHEDULER_H_
00010 #define OBZ_SCHEDULER_H_
00011
00012
00013 #include <stdio.h>
00014 #include <stdlib.h>
00015 #include <ucontext.h>
00016 #include <signal.h>
00017 #include <string.h>
00018 #include <stdbool.h>
00019 #include <sys/time.h>
00020 #include <unistd.h>
00021 #include <fcntl.h>
00022
00023
00024 static void schedule_next_thread(void);
00025 void thread_wrapper(void);
00026 static void setup_timer(void);
00027 static void timer_handler(int signum);
00028 int green_thread_create(void (*function)(void*), void* arg);
00029 void green_thread_run(void);
00030 void setup_fault_tolerance_signal_handler();
00031 void run();
00032
00033 #define STACK_SIZE (1024 * 1024)  // 1MB stack size ( arbitraire )
00034 #define MAX_THREADS 64
00035 #define TIME_SLICE_MS 256 // this is what sets the context switichign ( a hack )
00036
00037
00038
00039
00040 #endif // OBZ_SCHEDULER_H_
```

5.5 include/scheduler.h File Reference

```
#include "thread.h"
#include "obz_scheduler.h"
```

Classes

- struct Scheduler

Typedefs

- typedef struct Scheduler Scheduler

5.5.1 Detailed Description

Author

: Obz Team This file contains foundational data structure Scheduler , that comprises all the data about the global scheduler object

5.5.2 Typedef Documentation

5.5.2.1 Scheduler

```
typedef struct Scheduler Scheduler
```

scheduler is a struct that holds metadata about the scheduler global object , the one that will do the scheduling of the green thread : threads: is an allocated array that holds a pointer to greenthread objects current_thread: is the current thread with the `RUNNING` state , the concept of holding the number as an int is used mainly for indexing the threads[] array main_context : is the current ucontext_t of the current_thread (or threads[current_thread]) old↩ _timer: is used to set and hold the timer that's gonna be consumed by the kernel to send the `SIGALRM` signal old_action & is_switching : are not used !

5.6 scheduler.h

Go to the documentation of this file.

```
00001
00007 #ifndef SCHEDULER_H_
00008 #define SCHEDULER_H_
00009
00010 #include "thread.h"
00011 #include "obz_scheduler.h"
00012
00021 typedef struct Scheduler {
00022     GreenThread* threads[MAX_THREADS];
00023     int thread_count;
00024     int current_thread;
00025     ucontext_t main_context;
00026     struct sigaction old_action; // place holder for context switching
00027     struct itimerval old_timer;
00028     bool is_switching;
00029 } Scheduler;
00030
00031
00032 #endif // SCHEDULER_H_
```

5.7 include/thread.h File Reference

```
#include <ucontext.h>
```

Classes

- struct GreenThread

Typedefs

- typedef struct GreenThread GreenThread

Enumerations

- enum ThreadState { **READY** , **RUNNING** , **FINISHED** }

5.7.1 Detailed Description

Author

: Obz Team This file contains foundational data structures and Enums , GreenThread is struct comprising metadata about the green threads , the ThreadState enum contains possible state of a green thread.

5.7.2 Typedef Documentation

5.7.2.1 GreenThread

```
typedef struct GreenThread GreenThread
```

GreenThread is a struct that holds metadata about green thread : context: is the field that contains snapshot of all process , memory related data whilst running the green thread (register's values , process register counter ...) . stack: contains the custom stack that will hold the local variables for the wrapped anonymous functions (not working properly) . state: the current state of the green thread . function: the function that the green thread in quesiton will run . arg: the argument that are gonna be fed to the wrapped function .

5.7.3 Enumeration Type Documentation

5.7.3.1 ThreadState

```
enum ThreadState
```

This Enum is used to differentiate between the possible green threads states .

5.8 thread.h

Go to the documentation of this file.

```
00001
00008 #ifndef THREAD_H_
00009 #define THREAD_H_
00010 #include <ucontext.h>
00011
00012
00016 typedef enum {
00017     READY,
00018     RUNNING,
00019     FINISHED
00020 } ThreadState;
00021
00031 typedef struct GreenThread {
00032     ucontext_t context;
00033     void* stack;
00034     int id;
00035     ThreadState state;
00036     void (*function)(void*);
00037     void* arg;
00038 } GreenThread;
00039
00040 #endif // THREAD_H_
```

5.9 src/fallback.c File Reference

```
#include "obz_scheduler.h"
#include "scheduler.h"
```

Functions

- void fallback (int signum)
- void setup_fault_tolerance_signal_handler ()

Variables

- Scheduler **scheduler**

5.9.1 Detailed Description

Author

: Obz team

This file contains functions concerning handling for interupts and other signals besides the `SIGALRM` that's used in the scheduling of the green threads

5.9.2 Function Documentation

5.9.2.1 fallback()

```
void fallback (
            int signum)
```

This function is a rudimentary fallback signal handler , the sleep is to make the printf micro task run the last , since other green processes will be printing theirs

5.9.2.2 setup_fault_tolerance_signal_handler()

```
void setup_fault_tolerance_signal_handler ()
```

This function does set the interput handler for bootstraping rudimentary fault tolerance for the failure of the Green threads

5.10 src/fault_tolerance.c File Reference

```
#include "obz_scheduler.h"
#include <signal.h>
```

Functions

- void **signal_handler** (int sig)
- void setup_handle_signals ()

Variables

- struct sigaction **sa**

5.10.1 Detailed Description

Author

: Obz team

This file contains functions concerning handling for interupts and other signals besides the `SIGALRM` that's used in the scheduling of the green threads

5.10.2 Function Documentation

5.10.2.1 setup_handle_signals()

```
void setup_handle_signals ()
```

This function does set the interput handler for bootstraping rudimentary fault tolerance for the failure of the Green threads

5.11 src/scheduler.c File Reference

```
#include "obz_scheduler.h"
#include "scheduler.h"
```

Functions

- void thread_wrapper (void)
- void green_thread_run (void)

Variables

- Scheduler scheduler

5.11.1 Detailed Description

Author

: Obz team

This file contains functions concerning the initialization of the global scheduler object , and function concerning the internals of the scheduler (runtime)

5.11.2 Function Documentation

5.11.2.1 green_thread_run()

```
void green_thread_run (
            void )
```

this function kickstarts the scheduler

5.11.2.2 thread_wrapper()

```
void thread_wrapper (
            void )
```

this function does change the state of the green thread in order to execute the code in the anonymous function

5.11.3 Variable Documentation

5.11.3.1 scheduler

```
Scheduler scheduler
```

Initial value:

```
= {
    .thread_count = 0,
    .current_thread = -1,
    .is_switching = false
}
```

5.12 src/thread.c File Reference

```
#include "obz_scheduler.h"
#include "thread.h"
#include "scheduler.h"
```

Functions

- int green_thread_create (void(∗function)(void ∗), void ∗arg)

Variables

- Scheduler **scheduler**

5.12.1 Detailed Description

Author

: Obz team

This file contains functions concerning the scheduler , and the spawn function (green_thread_create) , the distincting between this name and 'spawn' is made because green_thread_create does create a wrapper around our user-defined anonymous functions and append it into an already global allocated array for green_threads !

5.12.2 Function Documentation

5.12.2.1 green_thread_create()

```
int green_thread_create (
            void(* function )(void *),
            void * arg)
```

This function does create a wrapper around user-defined anonymous function and append it to the global thread array 'scheduler.threads[]'

Index

anonymous.h
    lambda, 9

fallback
    fallback.c, 14
fallback.c
    fallback, 14
    setup_fault_tolerance_signal_handler, 14
fault_tolerance.c
    setup_handle_signals, 15

green_thread_create
    obz_scheduler.h, 11
    thread.c, 17
green_thread_run
    obz_scheduler.h, 11
    scheduler.c, 16
GreenThread, 7
    thread.h, 13

include/anonymous.h, 9, 10
include/obz_scheduler.h, 10, 11
include/scheduler.h, 12
include/thread.h, 13, 14

lambda
    anonymous.h, 9

obz_scheduler.h
    green_thread_create, 11
    green_thread_run, 11
    setup_fault_tolerance_signal_handler, 11
    thread_wrapper, 11

Scheduler, 7
    scheduler.h, 12
scheduler
    scheduler.c, 16
scheduler.c
    green_thread_run, 16
    scheduler, 16
    thread_wrapper, 16
scheduler.h
    Scheduler, 12
setup_fault_tolerance_signal_handler
    fallback.c, 14
    obz_scheduler.h, 11
setup_handle_signals
    fault_tolerance.c, 15
src/fallback.c, 14
src/fault_tolerance.c, 15
src/scheduler.c, 15
src/thread.c, 17

The C runtime for the <tt>OBZ Scheduler</tt> :, 1
thread.c
    green_thread_create, 17
thread.h
    GreenThread, 13
    ThreadState, 13
thread_wrapper
    obz_scheduler.h, 11
    scheduler.c, 16
ThreadState
    thread.h, 13