

UNIVERSITATEA BUCUREȘTI  
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ  
SPECIALIZAREA INFORMATICĂ



Lucrare de licență  
Analiza statică a primitivelor de sincronizare

Coordonator științific  
Paul Irofti

Absolvent  
Darius Marian

București, iulie 2020

# Cuprins

<b>1</b>	<b>Introducere</b>	<b>3</b>
1.1	Comparație cu ThreadSanitizer . . . . .	3
1.2	Comparație cu Clang Thread Safety Analysis . . . . .	3
<b>2</b>	<b>Tehnologii folosite</b>	<b>4</b>
2.1	Limbajul C11 și biblioteca standard C . . . . .	4
2.2	Limbajul C++17 și biblioteca standard C++ . . . . .	4
2.3	Compilerul Clang . . . . .	4
2.4	Unelte de dezvoltare CMake, CTest și Make . . . . .	4
2.5	Biblioteca pthread . . . . .	4
2.6	Biblioteca DL . . . . .	4
2.7	Biblioteca unwind . . . . .	4
2.8	Executabilele atos și addr2line . . . . .	4
2.9	Biblioteca mcga::cli . . . . .	4
<b>3</b>	<b>Descrierea proiectului</b>	<b>5</b>
3.1	Compilarea și instalarea proiectului . . . . .	5
3.2	Biblioteca de înregistrare a evenimentelor . . . . .	5
3.2.1	Interfață . . . . .	5
3.2.2	Descrierea unui eveniment . . . . .	5
3.2.3	Capturarea stivei de execuție . . . . .	5
3.2.4	Serializare (fișierul DUMP) . . . . .	5
3.2.5	Implementare . . . . .	5
3.3	Unelte de integrare . . . . .	5
3.3.1	cxxsync (C++, începând cu C++98) . . . . .	5
3.3.2	pthread (C/C++) . . . . .	5
3.3.3	libc++ (pentru C++, începând cu C++11) . . . . .	5
3.4	Executabilul de analiză statică . . . . .	5
3.4.1	Parcurgerea și parsarea fișierului DUMP . . . . .	5
3.4.2	Stocarea și observarea obiectelor active . . . . .	5
3.4.3	Crearea de rapoarte . . . . .	5
3.4.4	Afișarea simbolurilor din stiva de execuție . . . . .	5

3.4.5	Înregistrarea analizorilor . . . . .	5
3.5	Analizori instalați . . . . .	5
3.5.1	mutex-lock-order . . . . .	5
3.5.2	lock-shadow . . . . .	5
3.5.3	redundant-recursive-mutex . . . . .	5
3.5.4	redundant-rwlock . . . . .	5
<b>4</b>	<b>Exemple de utilizare</b>	<b>6</b>
4.1	Testele analizorilor (folosind cxxsync) . . . . .	6
4.2	Folosind LD_PRELOAD și <pthread.h> . . . . .	6
4.3	Folosind libc++ . . . . .	6
<b>5</b>	<b>Concluzii</b>	<b>7</b>

# 1 Introducere

## 1.1 Comparație cu ThreadSanitizer

## 1.2 Comparație cu Clang Thread Safety Analysis

## 2 Tehnologii folosite

2.1 Limbajul C11 și biblioteca standard C

2.2 Limbajul C++17 și biblioteca standard C++

2.3 Compilatorul Clang

2.4 Uneltele de dezvoltare CMake, CTest și Make

2.5 Biblioteca pthread

2.6 Biblioteca DL

2.7 Biblioteca unwind

2.8 Executabilele atos și addr2line

2.9 Biblioteca mcga::cli

## 3 Descrierea proiectului

### 3.1 Compilarea și instalarea proiectului

### 3.2 Biblioteca de înregistrare a evenimentelor

#### 3.2.1 Interfață

#### 3.2.2 Descrierea unui eveniment

#### 3.2.3 Capturarea stivei de execuție

#### 3.2.4 Serializare (fișierul DUMP)

#### 3.2.5 Implementare

### 3.3 Unelte de integrare

#### 3.3.1 cxxsync (C++, începând cu C++98)

#### 3.3.2 pthread (C/C++)

#### 3.3.3 libc++ (pentru C++, începând cu C++11)

### 3.4 Executabilul de analiză statică

#### 3.4.1 Parcurgerea și parsarea fișierului DUMP

#### 3.4.2 Stocarea și observarea obiectelor active

#### 3.4.3 Crearea de rapoarte

#### 3.4.4 Afișarea simbolurilor din stiva de execuție

#### 3.4.5 Înregistrarea analizorilor

### 3.5 Analizori instalați

#### 3.5.1 mutex-lock-order

#### 3.5.2 lock-shadow

#### 3.5.3 redundant-recursive-mutex

#### 3.5.4 redundant-rwlock

## 4 Exemple de utilizare

4.1 Testele analizorilor (folosind `cxxsync`)

4.2 Folosind `LD_PRELOAD` și `<pthread.h>`

4.3 Folosind `libc++`

## 5 Concluzii