FYP Proposal

**Crash Analysis Tool for Profiling Multi-Tasking Applications**

by

Faizan Ullah

Supervisor

Prof. Anjum Naveed (ext.)

Dr. Adnan Iqbal

Department of Computer Science

Namal College, Mianwali (An Associate Institute of University of Bradford, UK)

2018-2019

Iteration 2.1 (added line from email)

 Date of Revision: October 19, 2018**Table of Contents**

Contents

[1 Introduction 3](#_Toc526668835)

[1.1 Overview 3](#_Toc526668836)

[1.2 Objectives 3](#_Toc526668837)

[2 Methodology 4](#_Toc526668838)

[2.1 Design and Building 4](#_Toc526668839)

[2.1.1 Individual detection mechanism of locks: 4](#_Toc526668840)

[2.1.2 Back-tracing method calls: 4](#_Toc526668841)

[2.1.3 Finding the locked resource: 4](#_Toc526668842)

[2.1.4 Find out the waiting processes: 4](#_Toc526668843)

[2.2 Implementation 4](#_Toc526668844)

[2.2.1 System integration 4](#_Toc526668845)

[2.3 Testing 5](#_Toc526668846)

[2.3.1 Test the integrated system. 5](#_Toc526668847)

[2.3.2 Testing some well-known software. 5](#_Toc526668848)

[2.4 Evaluation 5](#_Toc526668849)

[3 Project Planning 5](#_Toc526668850)

[3.1 GANTT Chart 5](#_Toc526668851)

[4 Required Hardware & Software 6](#_Toc526668852)

[4.1 Hardware 6](#_Toc526668853)

[4.2 Software 6](#_Toc526668854)

# Introduction

## Overview

In Linux, any process or thread is called a task. An application consists of multiple tasks (threads and(or) processes). When one task holds a resource, other tasks have to wait. In multitasking systems, different tasks can run simultaneously and even can share a common resource. And when sharing happens, the task which is going to use the resource will apply a lock (mutex, semaphore, spinlock etc.) to proclaim its ownership and prevent intruders from claiming that specific resource. In this way, several tasks can use the same resource, but exclusively.

But sometimes things don’t go as planned. Your application lock something but operating system crashes the application or user plug out the power. In this case, the application’s credibility depends upon its locking mechanism because it didn’t get the time to unlock the resource. If the locking mechanism was not time-dependent or object-dependent as file locks, so when the application gets restarted it needs to acquire the same resource which is not locked by another process, but itself. This is when the problem arises, and software will get stuck along with the resource. The user will manually locate this problem to make the system running, but everyone will never know how and why this error occurred in the first place.

The example of this issue is the MYSQL database. It uses a file to record socket lock, and once in a blue moon when system abruptly gets shut down it doesn’t get the time to remove that lock file which in result restrict acquiring that socket again or sometimes do not allow the database to restart.

## Objectives

The main purpose of my final year project is to make a terminal utility that will try to disrupt the application locks and identify any potential issues like software crashing, permanent locking or restarting only after a system reboot. If possible, we will try to identify the type of locks along with the critical resources that were locked. In addition to that, we will (if possible for that software) also identify the processes that were waiting for the resource as well as the back trace of specific calls that lead to this problem. This aiding information would be enough for the debugger/programmer to identify where the problem has occurred and where he/she needs to make changes to get the software live without errors.

# Methodology

## Design and Building

The Phase of the project started in early October, and we will continue working on the following aspects:

### Individual detection mechanism of locks:

This phase depends upon finding methods that will assist in identifying the application's system locks. Which include finding the types system calls the application make and then we will be trying to devise a method of disrupting these calls/locking mechanisms.

### Back-tracing method calls:

This step consists of generating back-trace method calls that lead to that problem that created an anomaly in our system and leads to the killing of our application.

### Finding the locked resource:

This stage includes finding of the resource (if possible) that was being locked by the application.

### Find out the waiting processes:

As the resource is locked, some processes are in waiting state or could go in waiting state after the restart. This step will consist of identifying these processes.

## Implementation

The implementation phase will include the following aspects:

### System integration

System integration will be done after we have built all the components. In this step, all individual component will be combined into a terminal utility.

## Testing

In the development phase, every individual component will be unit tested but integrated system testing will be done when all the pieces of the application get combined.

### Test the integrated system.

This step includes rigorous testing of the product for any anomalies and weird behavior. It includes some custom-made application

### Testing some well-known software.

This step will include the testing of some well-known applications like MongoDB, MySQL, Chrome etc.

## Evaluation

After we have finished all the testing, we will evaluate the system to check whether it fulfills our objectives or not.

# Project Planning

## GANTT Chart

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Task | Aug | Sep | Oct | Nov | Dec | Jan | Feb | Mar | Apr | May |
| Manual Testing Software with Kernel Module |  |  |  |  |  |  |  |  |  |  |
| Individual Detection Mechanism of system locks |  |  |  |  |  |  |  |  |  |  |
| Killing/disrupting Mechanism of lock calls |  |  |  |  |  |  |  |  |  |  |
| Back Tracing Method Calls |  |  |  |  |  |  |  |  |  |  |
| Unit testing of implemented components |  |  |  |  |  |  |  |  |  |  |
| Mid Semester Submission and Presentation |  |  |  |  |  |  |  |  |  |  |
| Finding the locked resource |  |  |  |  |  |  |  |  |  |  |
| Finding the waiting processes |  |  |  |  |  |  |  |  |  |  |
| Finding the Type of locks |  |  |  |  |  |  |  |  |  |  |
| Individual Component Testing |  |  |  |  |  |  |  |  |  |  |
| System Integration |  |  |  |  |  |  |  |  |  |  |
| Integrated system testing |  |  |  |  |  |  |  |  |  |  |
| Testing Well-known Application |  |  |  |  |  |  |  |  |  |  |
| Evaluation |  |  |  |  |  |  |  |  |  |  |
| Write the Proposal |  |  |  |  |  |  |  |  |  |  |
| Write the Monthly Reports |  |  |  |  |  |  |  |  |  |  |
| Open House |  |  |  |  |  |  |  |  |  |  |
| Write the Final Report |  |  |  |  |  |  |  |  |  |  |
| Prepare for the Presentation |  |  |  |  |  |  |  |  |  |  |
| Design the Project Poster |  |  |  |  |  |  |  |  |  |  |

# Required Hardware & Software

## Hardware

Development PC: PC with Linux 16.04 or later

Least Display Resolution: 1024 \* 768 with 16-bit color

Server PC: PC with 512GB hard drive and 8 GB RAM

## Software

Custom Written Kernel Modules, Linux Utilities etc.

JAVA, C, C++ etc.