Portfolio Document

Faizan Ullah

2019

Chapter One

# Introduction

This part of the document will discuss about the project planning and how its been executed during the whole year. We’ll discuss the changes that has been made inspite of the year plan and the rationales that persuaded the change in the development cycle. This document will also include the summarized meetings details with the supervisor and how they effectively helped in achieving the end product. Further we’ll shed light on the aspects of personal growth and how this project has helped in shaping oneself as a perfect professional candidate for the corporate world. It’s a document that keeps track of all the growth this project has provided to my profession and my career. It documents my learnings, and my achievements.

Chapter Two

# Project Plan

## Pre-project Apprenticeship and Project Research (August 20, 2018 to Sep 27, 2018)

Apprenticeship In Software Testing and Analysis (2018) Under the Supervision of Dr. Anjum Naveed. I have spent the past summer testing futex locks and analyzing crash patterns of some widely familiar softwares and finding abnormalities in their functionalities. I have been communicating with Dr. Anjum Naveed (Ext. Supervisor). He explained me this problem and asked me to do extensive testing on different software applications and write report. Manual Crash analysis have been documented and problem’s traces was detected in these software. These reports are provided with the portfolio document. These manual testing demands a programmatical solution that could enhance the testing process and provides the developer with a tool that can do everything programmatically.

## Project Initial Proposal (Oct 9, 2018)

As after the problem identification; a potential project proposal should be formulated and observed. Following box will show the document submitted to supervisors. This was the tentative proposal like an initial understanding of the project.

A screenshot of a social media post

Description automatically generated

# 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.

First Tentative Proposed Plan

## Tools and Concepts learning phase (Oct 9, 2018 to Nov 17, 2018)

As this project demands an extensive knowledge of operating system, so well be learning tools and concepts in order to develop a protype model for mid semester evaluation.

Following are the tools and concept that needs to be learned before starting this project.

### C language

For the understanding of core kernel code and using system calls to develop a Syscalls tracer the C language must be learned. An online course has been taken named as C programming in Linux by DartmouthX IMTx available on edX.org. It has helped me improve my grip on the language so I could easily devise soilution.

A screenshot of a computer

Description automatically generated

### PTRACE Library

The core library upon which my project is based should be learned. As there is no online documentation is provided and no source is available to learn it except the poorly documented man page. I have done dozens of examples and experiments in order to learn its basic functionality.

### 2.4.2.1 Inter Process Communication

As ptrace extensively uses inter process communication and sometimes give result as form of signal its learning was must.

#### 2.4.2.2 Wait system call

Ptrace also extensively uses wait system call to work with. To fully understand the functionality of ptrace wait system call must be leaned.

### Futex System call

As project demands the killing of futex call so its understanding was necessary.

### Revive Basic Understanding

As this project would be a command line tool to debug other application so Basic understand of Linux debugging tools was must. For this purposes a book “ Linux Debugging and Performance Tuning by steave best” was read.

## Prototype Building (Nov 18, 2018 to Dec 20, 2018)

Basic working prototype of the intended software was created. It was proven as a skeleton code for the whole project. It had very minimalistic functionalities like tracing system calls, and identifying the different types of system calls etc.

## Mid-Semester Presentation (January 23, 2019)

Mid semester presentation proven very good as I was able to communicate the problem statement and proposed solution. I was able to show them the working prototype of the software.

## Post Mid-Semester Change of plan (Feb 3, 2019)

As multiple questions was raised and different objection was made. Inspite of the project plan was followed my supervisor asked for a new plan that is to be followed in this semester.

# Project phases

1. Literature Understanding.
2. Code Skeleton -tracing grandchild(threads) and keeping their records in specific data structure.
3. Specific calls that uses for Lock detection.
4. Adding Libunwind functionality to pinpoint problematic methods using stack trace.
5. Detection of locked resources.
6. Detection of waiting process.
7. Notifying the type of lock.
8. TESTING.

# Phases Explanation

## Literature Understanding

* Ptrace (API)
* Inter Process Communication Understanding –Signal Handling.
* Linux: Debugging and Performance Tuning (Book)
* “C Programming with Linux" (Seven Courses on edx)
* Self Service Linux (Book)

## Code Skeleton

1. Working on practice exercises to understand and learn ptrace
2. Understanding strace code for structural understanding
3. Writing Mini strace
4. Evolving code into skeleton code for project

## Detection Mechanism For Locking System Calls

*A kernel module will be written*

## Printing stacktrace

1. Learning libunwind library
2. Adding functionality in skeleton
3. Testing

## Testing

* Self-written Small Test Cases that have the issue built in.
* Testing of some of the well-known software.

# Plan of Action

## Semester One

### Phase-1: Literature Understanding and tools learing.

Status: Complete

### Phase 2: Building Prototype

Status: Complete

#### Functionality

* Tracing system call of a process.
* Can start a tracee via prototype
* Can connect to the tracee via prototype

## Semester Two

|  |  |  |
| --- | --- | --- |
| Week **1** | **1** – 8 Feb | Testing and Organizing Prototype. |
| Week **2** | **8** – 15 Feb | Tracing system call of the forked process. |
| Week **3** | **15** – 22 Feb | // |
|  | | |
| Week **4** | **22** – 1 March | // |
| Week **5** | **1** – 8 March | Adding Libunwind functionality. |
| Week **6** | **8** – 15 March | // |
| Week **7** | **15** – 22 March | Testing stacktrace fucntionality |
| Week **8** | **22** – 29 March | // |
|  | | |
| Week **9** | **29** – 5 April | Code merging |
| Week **10** | **5** – 12 April | Pre-Demo Testing |
| Week **11** | **12** – 19 April | Finishing and Demo Planning. |

### *Post Demo Plan.*

|  |  |
| --- | --- |
| **April 24 – May 24** | Post-Demo Testing and Debugging. |
| Project Documentation. |
| Releasing Beta Version. |

## Development

As to stick with the new module the development cycle will follow the new plan of action. This has helped a lot and I was able to complete the project on time.

## Open House (April 23, 2019)

Project demonstration day was celebrated and our projects were displayed. As I have finished the development of the software. The response of the audience was very impressive as most of the companies liked the idea and wanted to work on it.

A screenshot of a cell phone

Description automatically generated

## Documentation and Testing (May 7, 2019)

A product is incomplete when it is not well tested. We did three types of testing with the software developed and we’ll explain how each has its individual benefits. In the end, well explain how this product can’t be programmatically unit tested as the results of any given command could provide different results each time as the product is dependent upon other software (it test other software) which provides unexpected but understandable results. This makes it difficult to test the product but different tricks have been applied to get the desired results.

Chapter Three

# Meetings Record

Whole development process can be divided in number of meeting whose record is also available on GitHub. Every week we had a scheduled time slot to give supervisor an overview of the progress and make them aware of any problems that is being faced.

As I was working with external supervisor the time of meeting wasn’t fixed and every technical communication was done via series of Whatsapp messages both audio and text.

A screenshot of a cell phone

Description automatically generated

A whatsapp group was also formed in order to get a group conversation so both supervisors must be on a single page.

A screenshot of a cell phone

Description automatically generated

Every meeting has a record here and can be provided. As the meetings were informal and project plan was followed so series of repetitive plan is unnecessary.

Chapter Four

# Personal Development

During the course of this project I have learned a lot. How to communicate professionally and how to meet deadlines. Above all I have learned how to document each and everything. The learning curve during this academic year was very high. As Along with the project we were working on other semester course work that make us multitasker and responsible.

Curriculum Vitae

A close up of text on a white background

Description automatically generated