Report

EECS 3221 – Assignment 2 – Jia Xu

*POSIX Threads – Implementation of New\_Alarm\_Mutex.C*

Group 6 - Team Members:

Aggrim Arora

Arnesh Dawar

Inderpal Dhillon

Rui Huang

Xiahan Chen

Index

1. POSIX
2. pthreads in Linux
3. Dependencies for New\_Alarm\_Mutex.C
4. Requirements for implementation of New\_Alarm\_Mutex.C
5. High-level view of the design of New\_Alarm\_Mutex.C
6. Sample Run of New\_Alarm\_Mutex.C
7. Issues and Difficulties Encountered While Implementing the Design
8. Testing and outputs
9. Acknowledgements

## **POSIX**

Portable Operating System Interface (POSIX) is an application programming interface for UNIX systems. A program written using POSIX standards can easily be ported to other UNIX derivates (like Linux). It extends the ANSI C programming language and defines the following features:

* POSIX.1 (core services)
* Process creation and control
* Signals
* Pipe operations
* C Library
* POSIX.1b (Real-Time extensions)
* Priority Scheduling
* Real-Time signals
* Message passing
* Shared Memory
* Asynchronous and Synchronous I/O
* POSIX1.c (thread extensions)
* Control, creation and cleanup
* Scheduling
* Synchronization
* POSIX.2 (Shell and Utilities)
* Command interpreter

## **Pthreads**

Pthreads allow the execution of multiple tasks that overlap in time. At any instance, a thread is in one of the four basic states:

* **Ready**: able to run but is waiting for a processor or has just been started/unblocked/pre-empted by another process
* **Running**:in execution
* **Blocked**: is not able to run because it is waiting for a condition variable, to lock a mutex, or waiting for an /O operation to complete
* **Terminated**: has terminated by returning from its start function or by calling pthread\_exit.

Once terminated, a thread is recycled. Detailed analysis:

1. **Creation**

The initial thread of a process is created when the process is created. Additional threads can be created on a Pthreads system by calling thread\_create. As soon as a thread is created, its state is ready.

1. **Startup**

Once a thread has been created, it starts executing machine instructions. It starts with the execution of the thread start function specified in the pthread\_create.

1. **Running and Blocking**

A thread goes to sleep when the required resource is not available (blocked) or because it has been pre-empted by another thread. Most of its active life is spent in three stages: ready, running and blocked. When a thread attempts to lock a mutex that is currently locked, it becomes blocked. It is made ready for execution as soon as it is unblocked (some other thread unlocks the mutex and the current thread locks it again).

1. **Termination**

A thread usually terminates by returning from its start function. Another way to terminate a thread is by calling the pthread\_exit or pthread\_cancel function. If a thread is already detached, it moves to the next step immediately, recycling. On the other hand, if a thread returns a specific value, the value can then be extracted by using the pthread\_join function that also specifies the next thread to be executed and to which the return value is passed.

## **Dependencies for New\_Alarm\_Mutex.c**

* pthread.h

The pthread.h file consists of a number of constants and prototypes for the Pthreads functions

* errors.h

The errors.h file consists of headers and some error checking functions.

## **Design Requirements for New\_Alarm\_Mutex.C**

**Main Thread:**

Allocating 1 thread for input parsing to make sure the commands received are valid and then follow them up with the appropriate request.

The two requests handled by the main thread are:

* Start\_Alarm
* Change\_Alarm

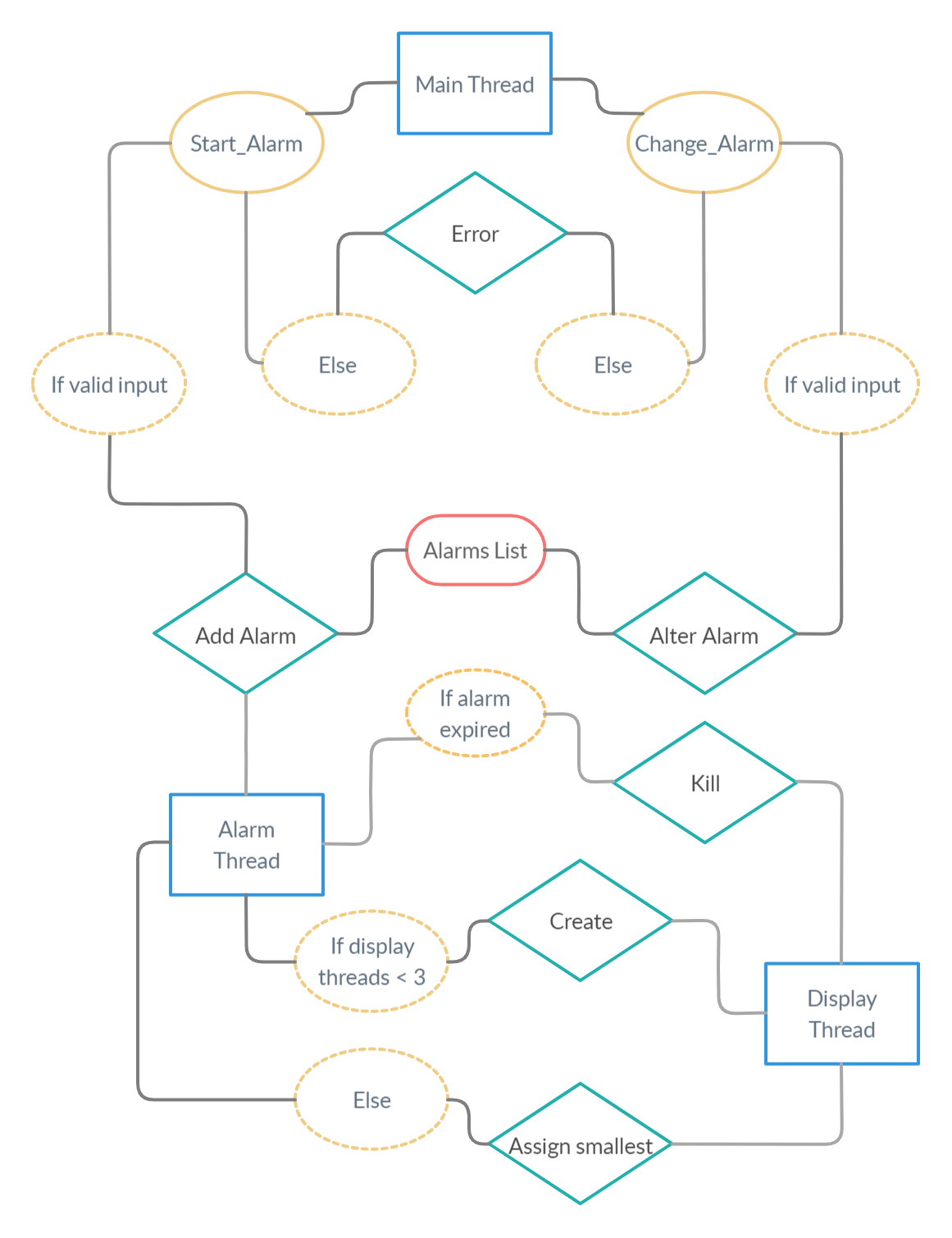
**Alarm Thread:**

Responsible for maintaining as many as 3 display threads for every alarm that is added to the list by the requests from main thread.

**Display Threads:**

Periodically prints the details of the alarm it has been assigned by the alarm thread until its reaches its expiry time when it is killed by the alarm thread.

## **High-Level View of New\_Alarm\_Mutex.C**



## **Sample Run of New\_Alarm\_Mutex.C**

## **Issues and Difficulties Encountered While Implementing the Design**

**Testing and Outputs**

**Acknowledgements**

This program was written as a part of the course EECS 3221 with Professor Jia Xu of the EECS department of York University, according to the specs entailed in *3221M\_W20\_Assignment\_2.pdf.*

*POSIX threads* and the library *pthreads.h* is a part of the operating system standards *POSIX* itemized by IEEE Computer Society.

*ANSI C* is a standard for the language C published by *American National Standards Institute* and the *International Organization for Standardization*, and this program is written in compliance to this standard.