**Scheduling tasks**

**Revision: 0.1**

**Date: 19-Jul-22**

**Authors**:

Lea Greenboum

Rachel Adler

Yehudit Erlbaum

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date** | **Summary of Changes** | **Author(s)** |
| 1.0 | 19/07/2022 | basic design |  |

Table of Contents

[1 Introduction 4](#_Toc108424799)

[1.1 Definitions, Acronyms & Abbreviations 4](#_Toc108424800)

[1.2 Reference Materials 4](#_Toc108424801)

[1.3 Overview \ Background 5](#_Toc108424802)

[1.4 Requirements 6](#_Toc108424803)

[2 High Level Design 7](#_Toc108424804)

[2.1 Block Diagram 7](#_Toc108424805)

[2.2 High Level Flows 7](#_Toc108424806)

[2.3 Assumptions 7](#_Toc108424807)

[2.4 Limitations of the suggested design. 7](#_Toc108424808)

[3 Detailed Design 8](#_Toc108424809)

[3.1 Detailed Description of flows. 8](#_Toc108424810)

[3.2 Flowcharts (when needed). 8](#_Toc108424811)

[3.3 Data structures. 8](#_Toc108424812)

[3.4 Public API’s signatures. 8](#_Toc108424813)

[4 Testing 9](#_Toc108424814)

[5 Open Issues 10](#_Toc108424815)

# Introduction

## Definitions, Acronyms & Abbreviations

|  |  |
| --- | --- |
| Abbreviation | Interpretation |
| RR | Round Robin - Algorithm for scheduling |

## Reference Materials

|  |  |
| --- | --- |
| Item | Name (Linked) |
|  | RR - <https://www.geeksforgeeks.org/cpu-scheduling-in-operating-systems/#rr> |

## Overview \ Background

Task scheduling project allows you to perform multiple tasks in a system with one processing unit in the most efficient way.

A large load of tasks on one processor can cause starvation and death of tasks.

The goal of the project is to prevent starvation and death, enable the processor to handle multiple tasks, and execute them in the shortest amount of time and in a way that we can achieve maximum CPU utilization.

This way we will gain maximum output, minimizing waiting time and response time

Data structure:

* The tasks are stored in three queues, one for each priority.
* can support eight tasks simultaneously.
* To enable a situation where all tasks have the same priority - each queue is size 8.

Algorithm:

The queues will be scheduled according to priority using the scheduling algorithm round robin. quantum for queue 1 is 80 milliseconds, queue 2 is 30 millisecond, and queue 3 is 20 millisecond.

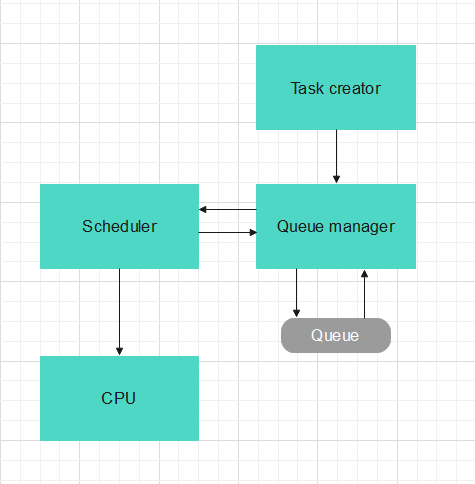
In each queue, the tasks will be scheduled by scheduling algorithm round robin. Quantum of all task is 10 millisecond.

## Requirements

* Schedule tasks for three of priorities.
* Schedule must prevent starvation of any of the tasks.
* Prevent task death due to timeout.

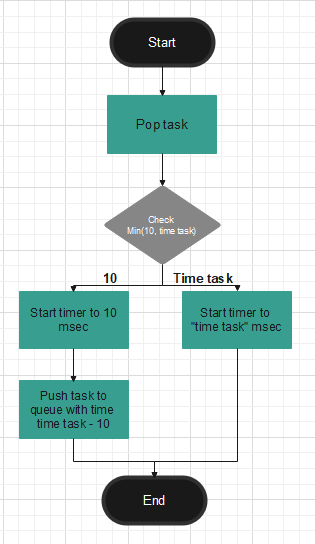
# High Level Design

## Block Diagram

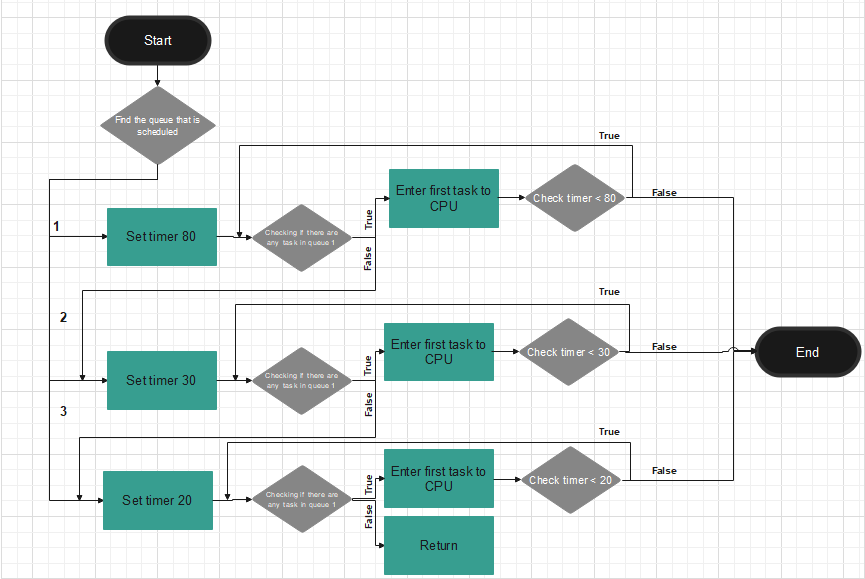


## High Level Flows

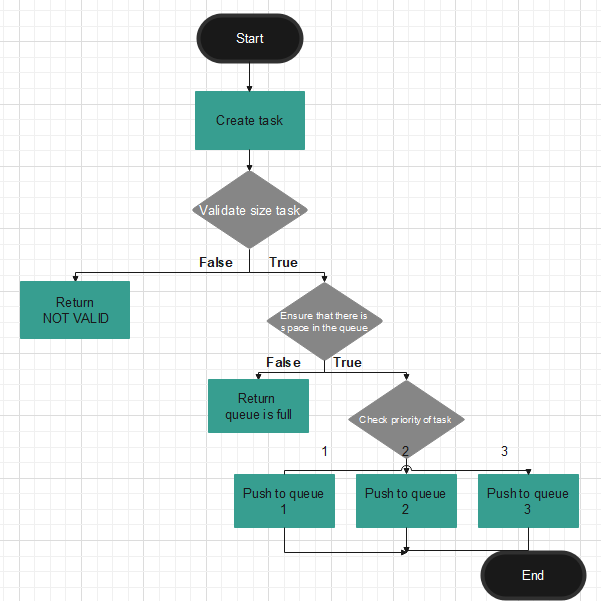
**Round Robin -** Description of the scheduling algorithm



**Next task -** Scheduling the next task to run on the CPU



**Push task -** Add the task to the queue based on its priority



## Assumptions

* The program supports 8 tasks at the same time.
* maximum running time for task is 20 milliseconds.
* The program supports only on priorities 1,2,3.

## Limitations of the suggested design.

Waiting times increase as the data structure fills up.

The maximum waiting time for tasks with difference priorities:

* Priority 1: 170 milliseconds.//320
* Priority 2: 440 milliseconds.//500
* Priority 3: 740 milliseconds.

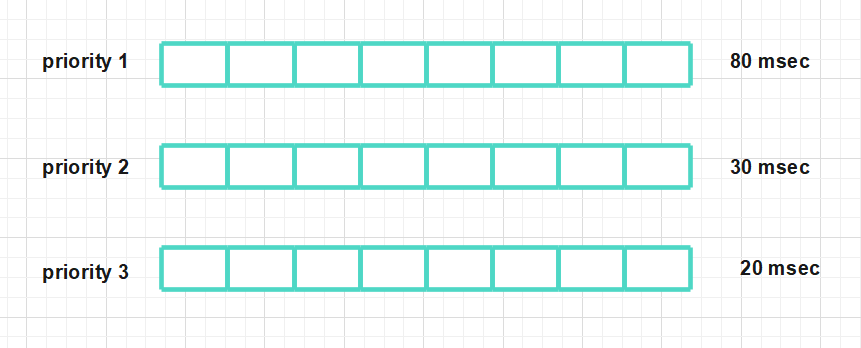
# Detailed Design

## Detailed Description of flows.

## Flowcharts (when needed).

## Data structures.

The data structure implement by 3 queues that implement by arrays with size 8.



## Public API’s signatures.

* TypeTask\* getTypeTaskByPriority (int priority);

Create TypeTask by priority.

# Testing

We will create edge cases in terms of size, frequency, quantity - and in the end we will check that they meet the conditions we have allocated.

# Open Issues

* What happen if task want to enter to queue and queue full? Queue Job
* What do when size of task more 20? (divide task…)
* Throw message when timeout of task almost finish?