# CPU-Scheduling

The Illusion of Multitasking

**Mark Krutzler**
*Kantonschule Im Lee — 2024/25*

# Contents

# Part I

# Forewords?

# Introduction

We all use computers. The modern person couldn't do half of the things that is expected from him without an instance of Windows, MacOs or any other Operating System.We surf the web, write emails, documents and have a video call at the same time. The question is: "How does CPU achieve all of this?" The things listed are only the parts of the OS that we come in contact with on a daily basis. What about all the other things hidden under the hood of the graphics environment? If it comes down to doing one thing than it's fine. The problem arises then we have multiple tasks: The CPU can only do one thing at a time and yet we all multitask on our machines. This Black Box effect of taking multiple jobs and working on them in a proper order is what I will disect in this "Maturarbeit".

The art of scheduling tasks could seem easy for us humans. We ?instinctively? estimate the many aspects of the task:

- Job: Convince our boss to give us a pay rise
  Priority: High
  Time Required: Medium (With a lot of Waiting Time)
  Effort: High (Said "No Way" last week)

- Job: Walk the Dog
  Priority: High
  Time Required: High
  Effort: Low

- Job: Go to the Hairdresser
  Priority: Low (Went last week as well)
  Time Required: High
  Effort: Low

These prediction are often highly complex and are based on previous experiences. We also include factors like: When was the last time I did it? In addition to all that every human plans differently based on what's important to him/her.

A computer does nothing like that. How could it? What he sees is:

- Job: Process 1
  Priority: Needs to be Set
  Time Required: ??? (Waiting Time: ???)
  Effort: ???

What he knows is when the task arrived and how much time he worked on it. With this in mind the let's get to a quick introduction of the metrics, so that we know how to rate the policies that we'll look at.

# Metrics

We can measure different aspects of a Scheduler Policy. There is however a differentiation between fairness and performance. It is often a tradeoff between the two. For example the concept of everybody getting a same sized piece of cake is fair. However, it would be faster to give more to the fast eaters than to those who like to talk during eating. In this scenario we can't have both fairness and performance. In reality it is really difficult to predict, who'll eat faster.

**Definition 0.1: Turnaround Time**

The turnaround time is a performance metric. It measures how long a task took to be completed from the time it arrived. It is calculated as follows:

$$T_{Turnaround} = T_{Completion} - T_{Arrival}$$

We'll often look at the Average Turnaround time for a predetermined set of jobs. This helps us to quantify how "efficient" a single task is handeled. In the best case scenario a single jobs turnaround time is equivalent to the runtime of that job. Another really important thing for us is Response Time. We want to move our mouse around. If we type we want the letter to appear in an instant and the Microsoft suite should start without much delay. Of course the startup time of a program depends on many other factors, however if we never even start to work on it, it'll take a good while.

**Definition 0.2: Response Time**

The response time is a performance metric. It measures the responsiveness of our computer. How much time does it take until the job is run? It is calculated as follows:

$$T_{Response} = T_{First\_Run} - T_{Arrival}$$

# Part II

# The ABCs of CPU Scheduling

# Chapter 1

# Basic Algorithms

## 1.1 Optimized for Performance

### 1.1 First In, First Out (FIFO)

### 1.1 Shortest Job First (SJF)

### 1.1 Shortest Time-to-Completion (STCF)

## 1.2 Optimized for Fairness

### 1.2 Round Robin (RR)

# Part III

# The Industry Standard of CPU-Scheduling

# Chapter 2

# Multi-Level Feedback Queue

# Chapter 3

# Lottery and Stride Scheduling

# Part IV

# Examples

# Chapter 4

# My Implementation

# Chapter 5

# Solaris Scheduling

# Chapter 6

# Linux 2.6 Fair Scheduler

# Part V

# Conclusion

# Sources

- https://texblog.org/2015/09/30/fancy-boxes-for-theorem-lemma-and-proof-with-mdframed/