Real Time Systems – 2023/2024

Practical Assignment #1

Scheduling, Priorities Assignment, and Measurement of Computation Times

Duarte de Sousa Cruz PL2 G5
2017264087

Gonçalo Nereu Figueiredo PL2 G5
2020226281

1 Introdução

Neste trabalho era nos pedido para realizar alguns exercícios de escalonamento de tarefas e medição do tempo de computação dessas tarefas. Usando apenas POSIX, com a extensão de Tempo Real, e configurado para correr em apenas 1 core, conseguimos configurar as tarefas para correrem de uma maneira escalonada e conseguimos ver se as tarefas cumprem a sua meta temporal.

Se a tarefa não cumprir a sua meta temporal dizemos que existe uma falha no sistema.

2 Exercícios propostos

2.1 Exercício 1

Neste exercício era nos pedido para medirmos o tempo de computação das tarefas f1, f2 e f3. Para medirmos o tempo de inicio da tarefa e o tempo final, usamos a system call `clock_gettime()`, e depois fazemos a subtracção dos tempos usando a função criada por nós, `struct timespec timeDiff()`.

Para que o programa corresse em apenas 1 core do CPU criamos uma mascara `cpu_set_t` e usamos a system call `CPU_SET()` para que apenas o core 0 do CPU fique associado à mascara criada. Para associar a mascara ao programa usamos a system call `sched_setaffinity()`.

Após a execução do programa, obtemos a seguinte tabela:

Tarefa	Tempo de Computação
f1	33.11 ms
f2	47.43 ms
f3	83.21 ms

Table 1: Tempos de computação.

2.2 Exercício 2

Neste exercício é nos pedidos para verificar se o sistema do execicio 1 é escalonável para dois tipos de ordenação RMPO e a sua inversa, utilizando o diagrama de Gantt e o método de Audsley.

2.2.1 Escalonamento RMPO

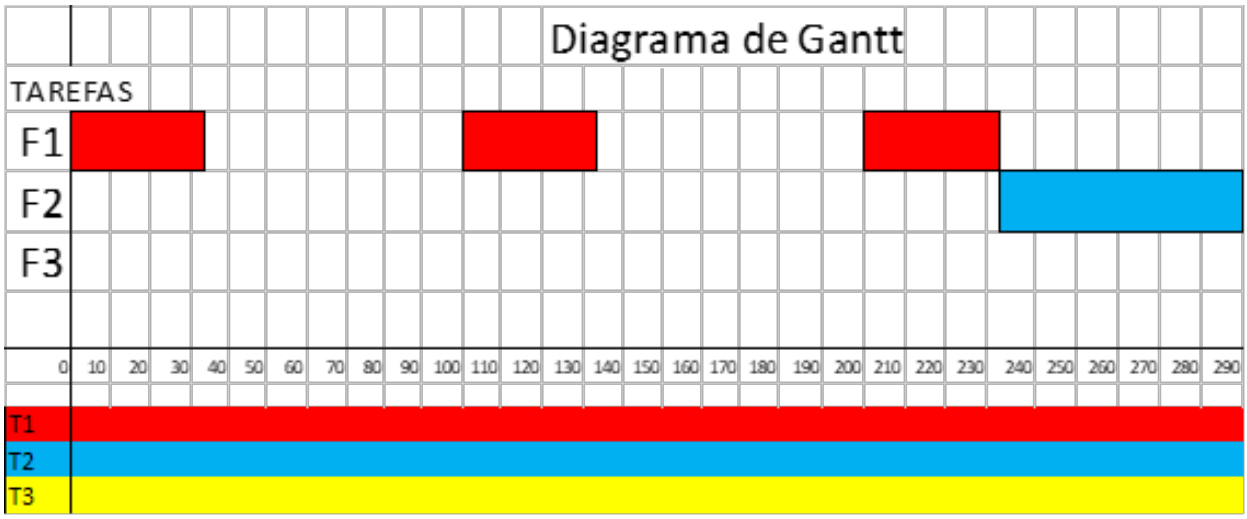

Figure 1: Diagrama de Gantt RMPO dos 0 aos 290 ms.

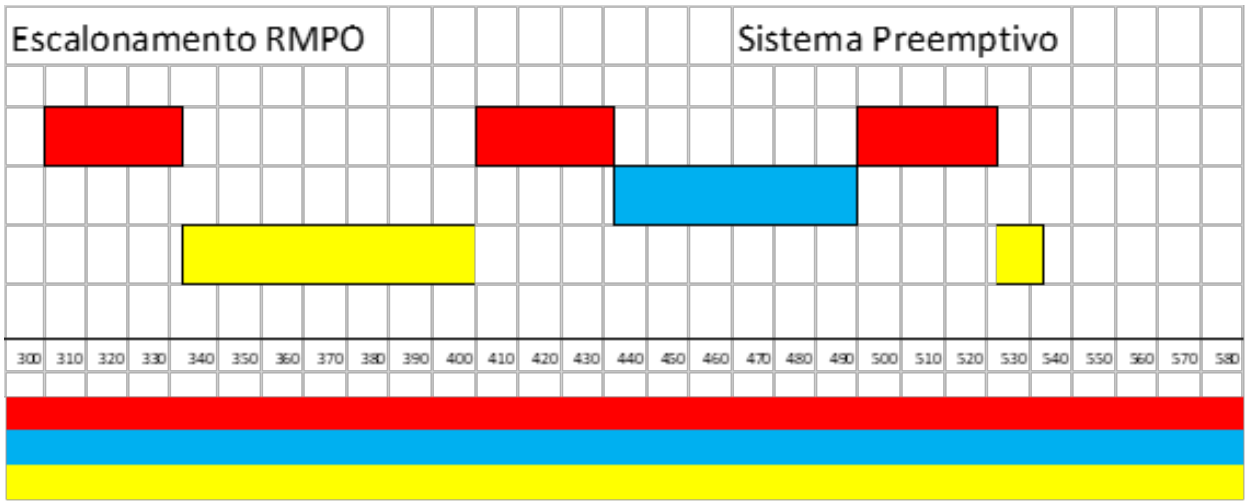

Figure 2: Diagrama de Gantt RMPO dos 290 aos 580 ms.

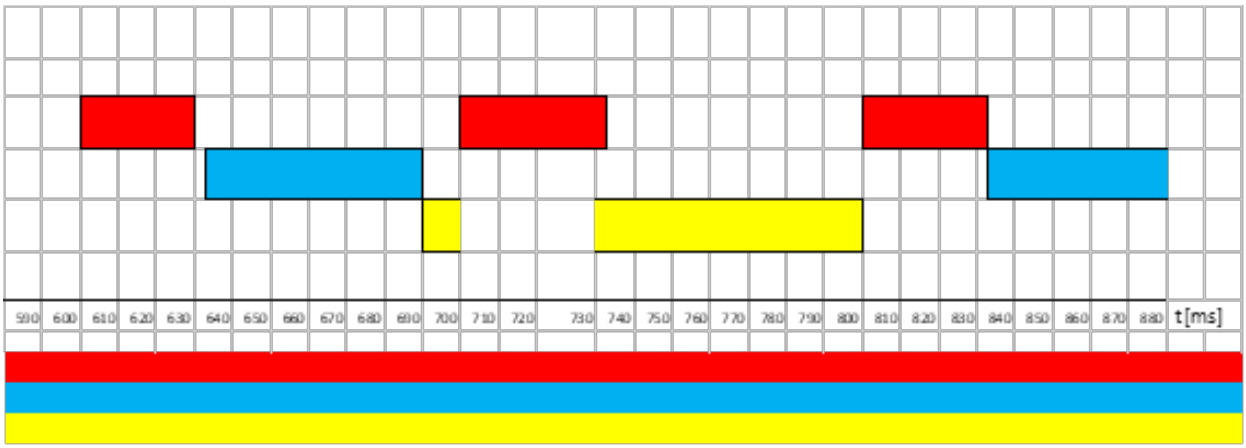

Figure 3: Diagrama de Gantt RMPO dos 580 aos 880 ms.

Tarefas	Período de Ativação	Tempo de Computação	Prioridade RMPO
f1	100 ms	33.11 ms	3 (higher)
f2	200 ms	47.43 ms	2
f3	300 ms	83.21 ms	1 (lower)

Table 2: Tabela de Prioridades RMPO.

O tempo de resposta é o tempo desde o instante em que a tarefa é activada até ser terminada. Este tempo é calculado através do método iterativo de Audsley usando a seguinte equação:

$$w_i^{n+1} = C_i + \sum_{j \, \epsilon \, hp(i)} \left\lceil \frac{w_n^i}{T_j} \right\rceil * C_j \tag{1}$$

Utilizando os valores da tabela 2 e aplicando a equação 1, vamos obter os seguintes resultados:

- Função 1, maior prioridade

$$w_1^0 = C_1 = 33.11 \; ms \tag{2}$$

  O tempo de resposta da função 1 vai ser 33.11 ms.

- Função 2

$$hp\,(2) = \{1\}\,, \; pode \; ser \; interrompida \; pela \; função \; 1 \tag{3}$$

$$w_2^0 = C_2 = 47.43 \; ms \tag{4}$$

$$w_2^1 = C^2 + \left\lceil \frac{w_2^0}{T_1} \right\rceil * C_1 = 47.43 + \left\lceil \frac{47.43}{100} \right\rceil * 33.11 = 80.54 \; ms \tag{5}$$

$$w_2^2 = C^2 + \left\lceil \frac{w_2^1}{T_1} \right\rceil * C_1 = 47.43 + \left\lceil \frac{80.54}{100} \right\rceil * 33.11 = 80.54 \; ms \tag{6}$$

  O tempo de resposta da função 2 vai ser 80.54 ms.

- Função 3

$$hp\,(3) = \{1, 2\}\,,\ pode\ ser\ interrompida\ pela\ função\ 1\ e\ 2 \tag{7}$$

$$w_3^0 = C_3 = 83.21\ ms \tag{8}$$

$$w_3^1 = C^3 + \left\lceil \frac{w_3^0}{T_1} \right\rceil * C_1 + \left\lceil \frac{w_3^0}{T_2} \right\rceil * C_2 = 83.21 + \left\lceil \frac{83.21}{100} \right\rceil * 33.11 + \left\lceil \frac{83.21}{200} \right\rceil * 47.43 = 163.75\ ms \tag{9}$$

$$w_3^2 = C^3 + \left\lceil \frac{w_3^1}{T_1} \right\rceil * C_1 + \left\lceil \frac{w_3^1}{T_2} \right\rceil * C_2 = 83.21 + \left\lceil \frac{163.75}{100} \right\rceil * 33.11 + \left\lceil \frac{163.75}{200} \right\rceil * 47.43 = 196.86\ ms \tag{10}$$

$$w_3^3 = C^3 + \left\lceil \frac{w_3^2}{T_1} \right\rceil * C_1 + \left\lceil \frac{w_3^2}{T_2} \right\rceil * C_2 = 83.21 + \left\lceil \frac{196.86}{100} \right\rceil * 33.11 + \left\lceil \frac{196.86}{200} \right\rceil * 47.43 = 196.86\ ms \tag{11}$$

(12)

O tempo de resposta da função 3 vai ser 196.86 ms.

Tarefas	Meta Temporal	Tempo de Computação	Tempo de Resposta	Cumpre Meta Temporal
f1	100 ms	33.11 ms	33.11 ms	Sim
f2	200 ms	47.43 ms	80.54 ms	Sim
f3	300 ms	83.21 ms	196.86 ms	Sim

Table 3: Tabela de Metas Temporais RMPO

Como podemos comprovar todas as tarefas cumprem as metas temporais, logo o sistema é escalonável.

2.2.2 Escalonamento RMPO Inverso

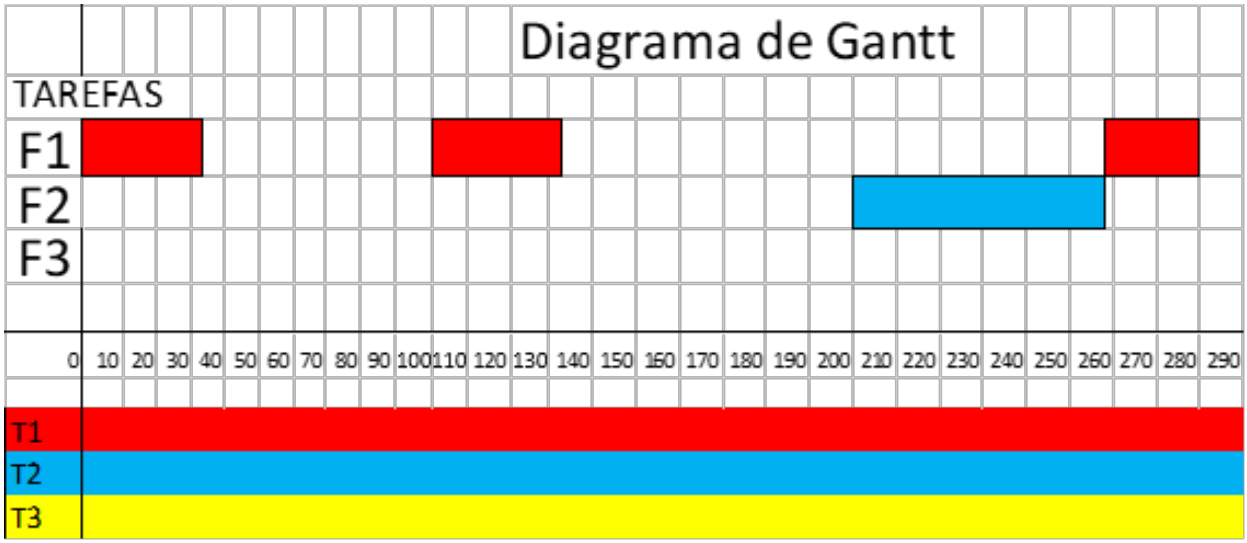

Figure 4: Diagrama de Gantt RMPO Inverso dos 0 aos 290 ms.

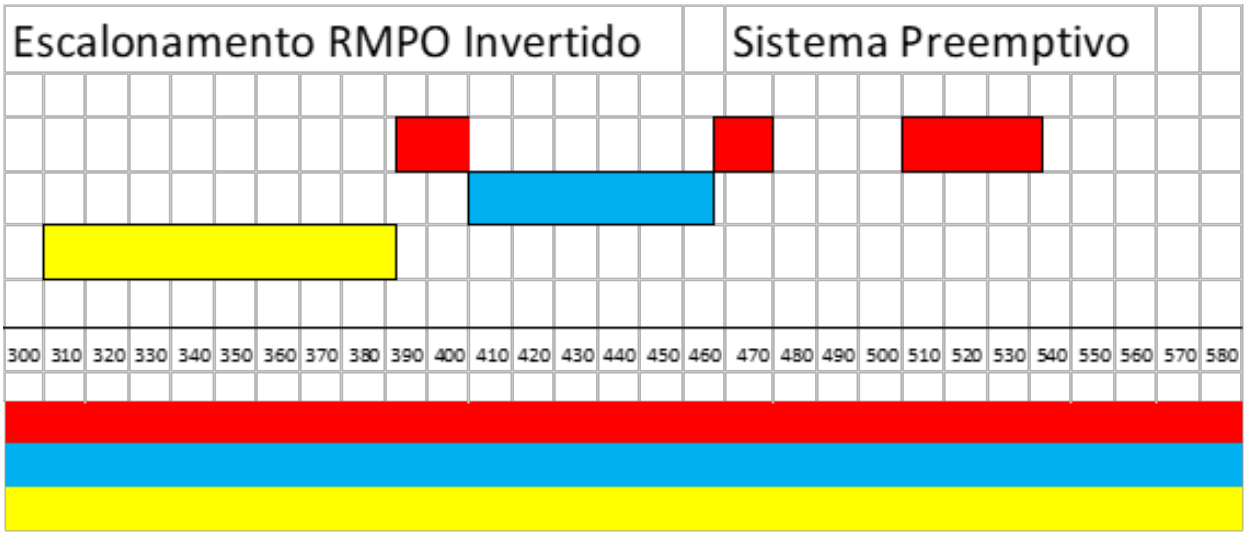

Figure 5: Diagrama de Gantt RMPO Inverso dos 290 aos 580 ms.

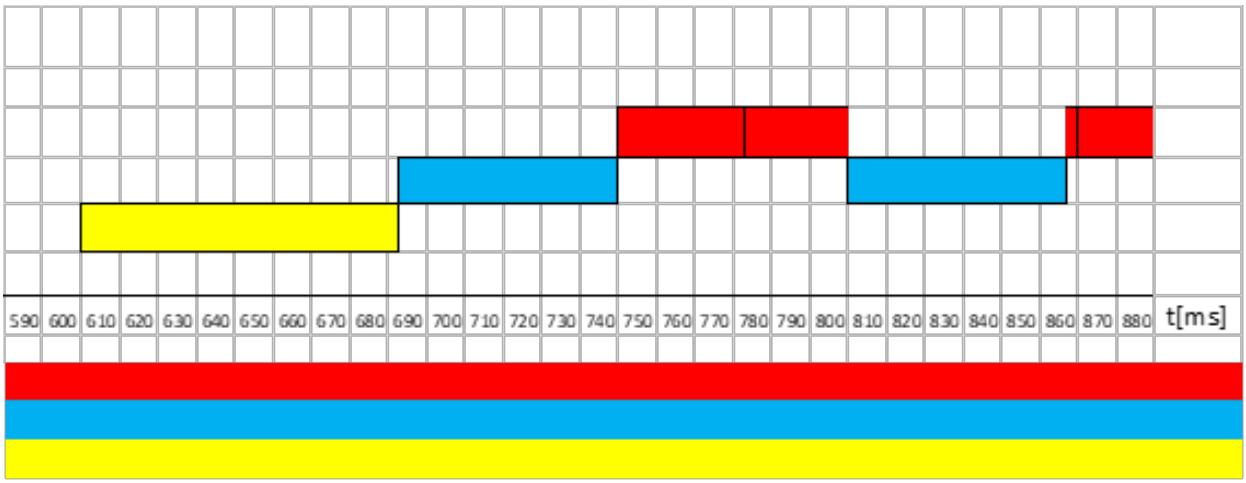

Figure 6: Diagrama de Gantt RMPO Inverso dos 580 aos 880 ms.

Tarefas	Período de Ativação	Tempo de Computação	Prioridade RMPO
f1	100 ms	33.11 ms	1 (lower)
f2	200 ms	47.43 ms	2
f3	300 ms	83.21 ms	3 (higher)

Table 4: Tabela de Prioridades RMPO Inverso.

- Função 3, maior prioridade

$$w_3^0 = C_3 = 83.21\ ms \tag{13}$$

  O tempo de resposta da função 3 vai ser 83.21 ms.

- Função 2

$$hp\,(2) = \{3\}\,,\ pode\ ser\ interrompida\ pela\ função\ 3 \tag{14}$$

$$w_2^0 = C_2 = 47.43\ ms \tag{15}$$

$$w_2^1 = C^2 + \left\lceil \frac{w_2^0}{T_3} \right\rceil * C_3 = 47.43 + \left\lceil \frac{47.43}{300} \right\rceil * 83.21 = 130.64\ ms \tag{16}$$

$$w_2^2 = C^2 + \left\lceil \frac{w_2^1}{T_3} \right\rceil * C_3 = 47.43 + \left\lceil \frac{130.64}{300} \right\rceil * 83.21 = 130.64\ ms \tag{17}$$

O tempo de resposta da função 2 vai ser 130.64 ms.

- Função 1

$$hp\,(1) = \{2,3\}\,,\ pode\ ser\ interrompida\ pela\ função\ 2\ e\ 3 \tag{18}$$

$$w_1^0 = C_1 = 33.11\ ms \tag{19}$$

$$w_1^1 = C^1 + \left\lceil \frac{w_1^0}{T_2} \right\rceil * C_2 + \left\lceil \frac{w_1^0}{T_3} \right\rceil * C_3 = 33.11 + \left\lceil \frac{33.11}{200} \right\rceil * 47.43 + \left\lceil \frac{33.11}{300} \right\rceil * 83.21 = 163.75\ ms \tag{20}$$

$$w_1^2 = C^1 + \left\lceil \frac{w_1^1}{T_2} \right\rceil * C_2 + \left\lceil \frac{w_1^1}{T_3} \right\rceil * C_3 = 33.11 + \left\lceil \frac{163.75}{200} \right\rceil * 47.43 + \left\lceil \frac{163.75}{300} \right\rceil * 83.21 = 163.75\ ms \tag{21}$$

(22)

O tempo de resposta da função 1 vai ser 163.75 ms.

Tarefas	Meta Temporal	Tempo de Computação	Tempo de Resposta	Cumpre Meta Temporal
f1	100 ms	33.11 ms	163.75 ms	Não
f2	200 ms	47.43 ms	130.64 ms	Sim
f3	300 ms	83.21 ms	83.21 ms	Sim

Table 5: Tabela de Metas Temporais RMPO Inverso.

É possível confirmar que a função 1 não cumpre as metas temporais, logo o sistema não é escalonável.

2.3 Exercício 3

Neste exercício era nos proposto para fazermos um programa que corresse durante 6 segundos e que tivesse as 3 tarefas a correr conjuntamente usando RMPO. As tarefas 1,2 e 3 têm respectivamente um período de activação de $0.1, 0.2$ e 0.3 segundos.

Era nos pedido também que no decorrer do programa apresentássemos os instantes temporais de cada tarefa a ser executada, nomeadamente o seu tempo de activação, tempo de começo, tempo de finalização e se cumpriu ou não a sua meta (deadline).

A primeira operação que fazemos no programa é associar o programa a apenas um core do CPU usando a system call `sched_setaffinity()`. Depois começamos com a atribuição dos atributos das threads, usamos o escalonamento FIFO e atribuímos as prioridades das threads de acordo com RMPO. As prioridades que atribuímos para a tarefas 1, 2 e 3 foram foram respectivamente 99, 98 e 97.

Criamos a estrutura `struct threadInput` para auxiliar nos paramentos de entradas da função `performWork()`.

Na função `performWork()`, que é a função em que as threads trabalham e onde medimos todos os instantes temporais das tarefas executadas pelas threads.

Depois de correrm os programa podemos verificar que todas as tarefas cumprem a sua meta temporal, como confirmado na tabela seguinte:

Tarefa	Periodo	Meta temporal
f1	100 ms	Fullfilled
f2	200 ms	Fullfilled
f3	300 ms	Fullfilled

Table 6: Cumprimento das metas temporais.

Relativamente ao tempo de resposta máxima e ao tempo de jitter obtemos os seguintes valores:

Tarefa	Resposta Max	Jitter
f1	33.87 ms	0.81 ms
f2	80.46 ms	0.29 ms
f3	196.61 ms	0.42 ms

Table 7: Tempos de resposta máxima e Jitter.

2.4 Exercício 4

Este exercício está dividido em duas alternativas.

2.4.1 Alternativa A

Neste exercício foi nos proposto para realizarmos um programa parecido com o do exercício 3 mas que invertesse as prioridades das tarefas no instante $t = 1.95s$ e voltar as prioridades originais no instante $t = 3.95s$.

Como a thread 1 é a que vai ter a prioridade mais alta antes do primeiro instante vamos fazer com que seja esta thread a alterar as prioridades de todas as threads. Para que consiga alterar as prioridades vai ter de existir uma varialvel global que guarda os id's das threads todas, `pthread_t thread[3]`. Verificamos se o tempo de mudar já chegou comparando sempre o tempo decorrido com o instante de inversão usando as funções `timeMenor()`, `timeDiff()` e `clock_gettime()`.

Para voltar às prioridades originais terá de ser a thread 2 a mudar as prioridades de todas as threads, visto que neste instante esta terá a maior prioridade.

Depois de correr o programa `sudo ./ex4` podemos observar que depois de inverter as prioridades a tarefa 1 não cumpre a sua meta temporal e a tarefa 2 e 3 continuam a cumprir as suas metas temporais.

Tarefa	Período	Meta temporal
f1	100 ms	Not fullfilled
f2	200 ms	Fullfilled
f3	300 ms	Fullfilled

Table 8: Cumprimento das metas temporais.

Relativamente aos tempos de resposta máxima e ao tempo de jitter podemos ver que para as tarefas 1 e 2 os tempos de resposta máxima aumentaram relativamente ao exercício 3 em que não são invertidas as prioridades a meio da execução do programa. Observando o tempo de jitter das tarefas vemos que o jitter aumentou em todas as tarefas, o que era o esperado visto que o tempo de resposta máximo aumentou.

Tarefa	Resposta Max	Jitter
f1	163.41 ms	130.35 ms
f2	130.33 ms	83.24 ms
f3	197.24 ms	114.08 ms

Table 9: Tempos de resposta máxima e Jitter.

2.4.2 Alternativa B

Neste exercício era nos pedido para correr a alternativa A mas neste caso com vários cores do CPU. É de esperar que todos os tempos melhorem pois o trabalho das tarefas vai ser dividido pelos vários cores. Na maquina em que testamos o código, o programa correu em 6 cores e estes foram os resultados:

Tarefa	Período	Meta temporal
f1	100 ms	Fullfilled
f2	200 ms	Fullfilled
f3	300 ms	Fullfilled

Table 10: Cumprimento das metas temporais.

Tarefa	Resposta Max	Jitter
f1	33.69 ms	0.61 ms
f2	47.71 ms	0.60 ms
f3	83.63 ms	0.42 ms

Table 11: Tempos de resposta máxima e Jitter.

2.5 Exercício 5

Neste exercício era nos proposto para implementarmos um ficheiro c `func2.c` que realize o mesmo trabalho que `func.o`. Para isso, no ficheiro func2.c implementamos 3 funções idênticas `f1()`, `f2()`

e `f3` que tiram partido de uma função que criamos `wait(int time)` em que time é o tempo de computação das tarefas que observamos no exercício 1. Na função `wait()`, usamos `clock_gettime()` para sabermos o instante temporal em que a thread entra na função e comparamos esse tempo com o tempo de input da função.

Depois de executar, `sudo ./ex5_1` obtemos os seguintes tempos de computação:

Tarefa	Tempo de Computação
f1	30.00 ms
f2	50.00 ms
f3	80.00 ms

Table 12: Tempos de computação.

Podemos concluir então que a implementação da nova função foi bem sucedida pois os tempos de computação das tarefas são semelhantes.

Depois de executar o programa `sudo ./ex5_3`, que é uma versão do exercício 3 mas com o ficheiro `func2.o`, obtemos os seguintes resultados:

Tarefa	Resposta Max	Jitter
f1	30.44 ms	0.43 ms
f2	80.33 ms	0.29 ms
f3	180.34 ms	20.25 ms

Table 13: Tempos de resposta máxima e Jitter.

Podemos observar que os tempos de resposta máxima são parecidos com aqueles observados no exercício 3. Já o jitter, também é igual nas tarefas 1 e 2 mas na tarefa 3 o jitter é significativamente maior do que aquele observado no exercício 3.

2.6 Exercício 6

Neste exercicio era nos proposto para testar o programa desenvolvido no exercicio 3, mas desta vez em vez de usar SCHED_FIFO teríamos de usar SCHED_RR. Para isso temos de atribuir a mesma prioridade a todas as thread e com a systems call `pthread_attr_setschedpolicy()` para associar o escalonamento Round Robin.

Se correrrmos o exercício 6, `sudo ./ex6` obtemos os seguintes resultados:

Tarefa	Período	Meta temporal
f1	100 ms	Not fullfilled
f2	200 ms	Fullfilled
f3	300 ms	Fullfilled

Podemos ver que a primeira task, como tem uma deadline mais apertada, não consegue cumprir a sua meta temporal. Todas as outras conseguem. Relativamente aos tempos de resposta máxima e ao jitter, apresentamos a seguinte tabela:

Table 14: Cumprimento das metas temporais.

Tarefa	Resposta Max	Jitter
f1	163.40 ms	130.33 ms
f2	162.86 ms	115.77 ms
f3	163.37 ms	80.20 ms

Table 15: Tempos de resposta máxima e Jitter com rr_timeslice = 100 ms.

Tarefa	Resposta Max	Jitter
f1	137.10 ms	104.02 ms
f2	164.71 ms	94.30 ms
f3	194.42 ms	73.37 ms

Table 16: Tempos de resposta máxima e Jitter com rr_timeslice = 20 ms.

References

[1] The Linux Man-Pages Project. The Linux Man-Pages Project. [Online]. Available: http://www.kernel.org/doc pages/