UNIVERSIDADE FEDERAL
DO RIO DE JANEIRO

Universidade Federal do Rio de Janeiro
Instituto de Computação
ICP246 – Arquitetura de Computadores e Sistemas Operacionais

Vitória Serafim
Riquelme Gomes
Daniel Levacov

Relatório Trabalho 2: Simulação de escalonamento de processos

2023.2

1 Objetivos

- Desenvolver um simulador que implementa o algoritmo de escalonamento de processos, usando a estratégia de seleção Round Robin (ou Circular) com Feedback.

- Trabalhar em equipe para organizar, projetar e desenvolver soluções para problemas formulados que envolvem o estudo e o conhecimento sobre gerenciamentos do sistema operacional.

- Servir como uma ferramenta de aprendizagem, fixação e demonstração de alguns conceitos e princípios de escalonamento de processos para a estratégia de seleção adotada.

- Permitir a análise interativa das decisões de escalonamento e do estado do sistema em diferentes momentos e avaliar o impacto de diferentes configurações de parâmetros no desempenho geral do sistema.

2 Premissas consideradas no desenvolvimento

- O limite máximo de processos criados é de 50, no entanto estamos trabalhando com a simulação de um caso que conta com 5 processos.
- Valor da fatia de tempo dada aos processos em execução: 3 unidades de tempo.
- Os tempos de serviço e o tipo de I/O que cada processo faz foram definidos de forma aleatória para cada processo criado. Eles se encontram na tabela abaixo.

Processo	Tempo de serviço	Tipo de I/O
0	4	Disco
1	5	Fita magnética
2	2	Impressora
3	3	Fita magnética
4	1	Não faz I/O

- Tipos de I/O e seus tempos de duração:

1) **Disco** — 2 u.t. de duração e retorna para a fila de **baixa** prioridade.
2) **Fita magnética** — 3 u.t. de duração e retorna para a fila de **alta** prioridade.
3) **Impressora** — 4 u.t. de duração e retorna para a fila de **alta** prioridade.

Tipo de I/O	Tempo de duração (em u.t.)
Disco	2
Fita magnética	3
Impressora	4

• Estamos considerando que todos os eventos de I/O começam depois de 3 u.t. (1 quantum)

- Gerência de Processos

1) Cada processo possui: PID, tempo de ativação, tempo de execução, tempo restante, tempo de conclusão, turnaround, tempo de espera, prioridade e tipo de I/O (0: Nenhum, 1: Disco, 2: Fita magnética, 3: Impressora).

2) O PID de cada processo consiste em um número natural definido a partir da ordem de chegada dos processos.

3) O escalonador funciona com 5 filas, sendo uma fila de alta e uma de baixa prioridade (0: alta e 1: baixa) para execução na CPU e outras 3 filas de I/O, uma para cada tipo de dispositivo.

- Ordem de entrada na fila de prontos

1) Processos novos — entram na fila de **alta** prioridade.

2) Processos que retornam de I/O – dependente do tipo de I/O solicitado.

3) Processos que sofreram preempção – retornam na fila de **baixa** prioridade.

3 Saída da execução do simulador

Para executar o código do simulador, utilizamos o seguinte comando:

```
gcc main.c -Wpedantic -Wall -o main
```

Bibliotecas utilizadas:

Figura 1 – Bibliotecas

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
```

Abaixo estão as premissas consideradas no código:

Figura 2 – Premissas

```c
// Algumas premissas usadas na construção do código
#define MAX_PROCESS 50
#define HIGH_PRIORITY 0
#define LOW_PRIORITY 1

// Valores para identificar o tipo de prioridade de cada I/O
#define DISK_IO_PRIORITY 2
#define TAPE_IO_PRIORITY 3
#define PRINTER_IO_PRIORITY 4

// Tempo de cada I/O
#define DISK_TIME 2
#define TAPE_TIME 3
#define PRINTER_TIME 4

// Tempo de quantum fixo
#define TIME_QUANTUM 3
```

Em nosso simulador, os processos e as filas são estruturados da seguinte forma:

Figura 3 – Estrutura do processo

```
// Definição da estrutura dos processos
struct Process
{
  int id;             // PID
  int arrivalTime;    // Tempo de ativação
  int burstTime;      // Tempo de execução
  int remainingTime;  // Tempo de execução restante do processo
  int completionTime; // Tempo de conclusão
  int turnaroundTime; // Turnaround do processo
  int waitingTime;    // Tempo de espera
  int priority;       // Prioridade de execução
  int ioType;         // Tipo de I/O --> 0: Nenhum, 1: Disco, 2: Fita, 3: Impressora
};
```

Figura 4 – Estrutura da fila

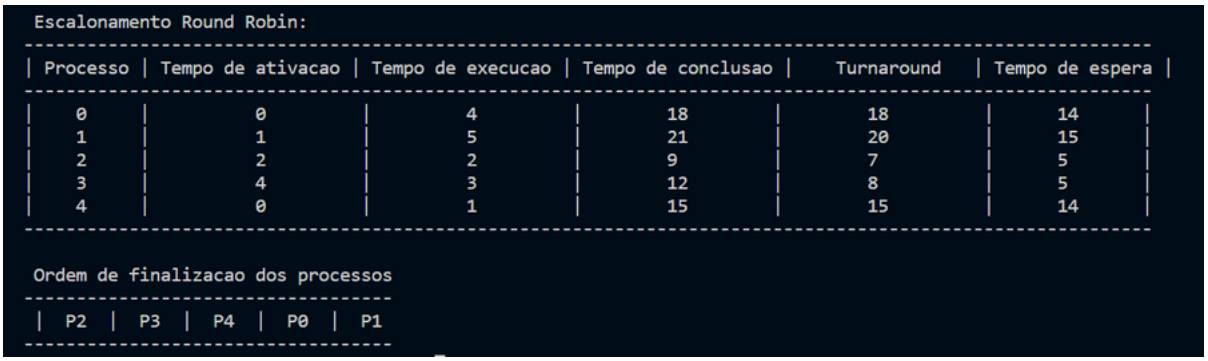
```
// Definição das filas
struct Queue
{
  struct Process *array[MAX_PROCESS];
  int front, rear;
};
```

A saída da execução do simulador consiste em uma tabela seguida por uma ordem de finalização dos processos. Na tabela há as seguintes informações para cada processo:

- PID
- Tempo de ativação
- Tempo de execução
- Tempo de conclusao
- Turnaround
- Tempo de espera

Na imagem abaixo mostramos a saída obtida pelo código para o nosso caso teste com 5 processos:

Figura 5 – Saída caso teste



Recriei a tabela e a ordem de finalização obtidas como saída do código para facilitar a visualização dos dados:

Tabela 1 – Tabela de saída do caso teste

Processo	Tempo de ativação	Tempo de execução	Tempo de conclusão	Turnaround	Tempo de espera
0	0	4	18	18	14
1	1	5	21	20	15
2	2	2	9	7	5
3	4	3	12	8	5
4	0	1	15	15	14

Tabela 2 – Ordem de finalização dos processos

P2	P3	P4	P0	P1

(Patterson; Hennessy, 2014; Maziero, 2019; Stallings, 2018)

Referências

MAZIERO, C. A. **Sistemas Operacionais: Conceitos e Mecanismos**. 2019.

PATTERSON, D. A.; HENNESSY, J. L. **Organização e Projeto de Computadores**: A Interface Hardware/Software. 4. ed. Rio de Janeiro: Elsevier Editora Ltda, 2014.

STALLINGS, W. **Operating Systems**: Internals and Design Principles. 9. ed. [S.l.]: Pearson, 2018.