Relat´orio de Implementa¸c˜ao do Algoritmo Round Robin com Feedback

Nome – DRE

Nome - DRE

Nome - DRE

Grupo 5

Novembro de 2023

# Introdu¸c˜ao

O presente relatório descreve o trabalho elaborado pelo nosso grupo como projeto prático 2 da disciplina, cujo objetivo é construir um programa em linguagem C que simula a execução de um escalonador de processos, assim como feito em um sistema operacional.

Um escalonador de processos é a parte de um sistema operacional que gerencia a ordem de execução dos processos que estão competindo pelos recursos do sistema, especialmente o tempo de CPU. Para tal, ele utiliza algoritmos específicos para determinar a sequência em que os processos serão executados, com base em critérios como prioridade, tempo de execução e outras características. O programa simula a execução dos processos exibindo na tela o processo que ocupa a CPU a cada unidade de tempo (u.t.), uma medida de tempo arbitrária conhecida pelo sistema computacional.

Conforme novos processos requisitam o uso da CPU, eles são organizados de forma a fornecer uma competição justa, de modo que seja impedida uma conpetição injusta entre eles, resultando em processo esperando tempo indeterminado por um recurso (starvation), o que geraria um sistema operacional não responsivo e inconsistente.

# Comportamento do escalonador

A estratégia de escalonamento implementada é o Round Robin com feedback. O Round Robin é um algoritmo simples e amplamente utilizado, no qual cada processo recebe uma fatia de tempo de CPU fixa antes de passar para o proximo processo. Ao fim da execução, o processo corrente para o final da fila de espera. No entanto, o Round Robin com feedback introduz uma abordagem mais dinâmica. Esse algoritmo permite que processos que exige mais tempo de CPU possam receber fatias maiores do tempo de CPU. A ideia principal é monitorar o comportamento dos processos e ajustar dinamicamente as prioridades.

Cada processo é inicialmente atribuído a uma fila com uma prioridade padrão, no caso do simulador, uma **fila de alta prioridade**. Então os processos na fila são programados usando o algoritmo Round Robin, recebendo fatias de tempo fixas de CPU em uma ordem circular. Se um processo não conseguir ser concluído durante a sua fatia de tempo, ele é movido para uma **fila de baixa prioridade**.

Os processos podem ser interrompidos ao realizarem operações de E/S, pois ficam bloqueados até receberem uma resposta do processo que trata desta operação. No simulador, a depender da E/S realizada, o processo pode voltar quaisquer uma das duas filas iniciais existentes. Existem 3 tipos de E/S no simulador: Disco, fita magnética e impressora.

## Modelo de dados simulador

Segue abaixo a descrição das principais estruturas que são operadas neste simulador. Aliada à explicação do funcionamento do algoritmo, este modelo tornará mais intuitiva o comportamento interno do escalonador, pois ele apenas manipula estes dados de forma a implementar os procedimentos descritos neste relat´orio.

// Definição do tipo de E/S

typedef enum {

DISCK\_IO,

TAPE\_IO,

PRINTER\_IO,

} IOType;

// Definição dos atributos de uma operação de E/S

typedef struct {

IOType io\_type;

Int io\_duration;

Int start\_time;

Int remaining\_duration;

} IOOperation;

// Possiveis status que o processador assume neste S.O.

typedef enum {

READY,

RUNNING,

IO,

} ProcessStatus;

// Estrutura que define atributos de um processo.

Typedef struct {

Int pid;

// Dados de tempo

Int arrival\_time; //Instante de chegada

Int burst\_time; //Tempo de CPU

Int end\_time; //Instante de encerramento

Int turnaround\_time; //Tempo de execução

//Atributos temporarios

Int remaining\_quantum;

Int remainin\_burst\_time;

Int current\_burst\_time;

//Atributos relacionados as operações de E/S deste processo

IOOperation\* io\_operations;

Int num\_io\_operations;

Int current\_io\_operation;

IOType io\_type;

ProcessStatus status;

} Process;

## Premissas do simulador

* + A macro MAX PROCESSES define o número de processos que serão criados e executados no simulador. No casos da nossa simulação, este valor estava configurado como 5.
  + O PID dos processos varia incrementalmente de 0 a MAX\_PROCESSES -1.
  + A macro QUANTUM define a quantidade de unidades de tempo que compóem um quantum. Na simulção, este valor foi 2, pois valores maiores resultam em outputs impraticaveis de escrever no relatório.
  + Novos processos podem ter tempo de chegada entre 0 e MAX\_ARRIVAL\_TIME, definida, nesta seção, para 4.
  + O tempo de serviço de um processo varia entre MIN\_BURST\_TIME e MAX\_BURST\_TIME. Na simulação estes valores são, respectivamente, 7 e 10.
  + Processos novos entram na fila de alta prioridade.
  + Processos que sofreram preempção entram na fila de baixa prioridade.
  + Processos que retornam de E/S retornam para a fila ditada pela regra definida para o dispositivo em quest˜ao.
  + **Descri¸c˜ao das opera¸c˜oes de E/S**
    1. **Disco** - Duração definida pela macro DISK\_TIME, configurada para 2 u.t. na simulação. Ao encerrar, o processo que a chamou entra na fila de **baixa** prioridade.
    2. **Fita magnética** - Duração definida pela macro TAPE\_TIME, configurada para 3 u.t. na simulação. Ao encerrar, o processo que a chamou entra na fila de **alta prioridade.**
    3. **Impressora** - Duração definida pela macro PRINTER\_TIME, configurada para 4 u.t. na simulação. Ao encerrar, o processo que a chamou entra na fila de **alta prioridade**.
  + Ao início da simulação, um processo pode necessitar realizar de 0 a 2 operações de E/S, sendo a quantidade de operações e tempo de início aleatórios, esta definido dentro do intervalo do tempo de execução do programa. O tempo de chegada do processo de E/S é a quantidade de u.t. desde o início da execução de processo, contando apenas o tempo em que o mesmo passou na CPU.

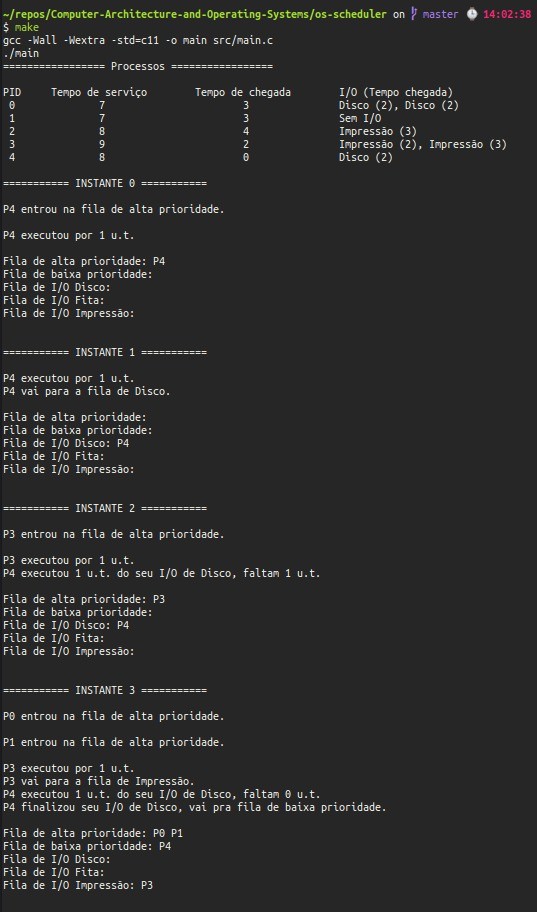
# Output do simulador

Como o simulador possui um componente aleatório, seu output não é determinístico, portanto, esta saída ilustra apenas uma execução arbitrário do programa.

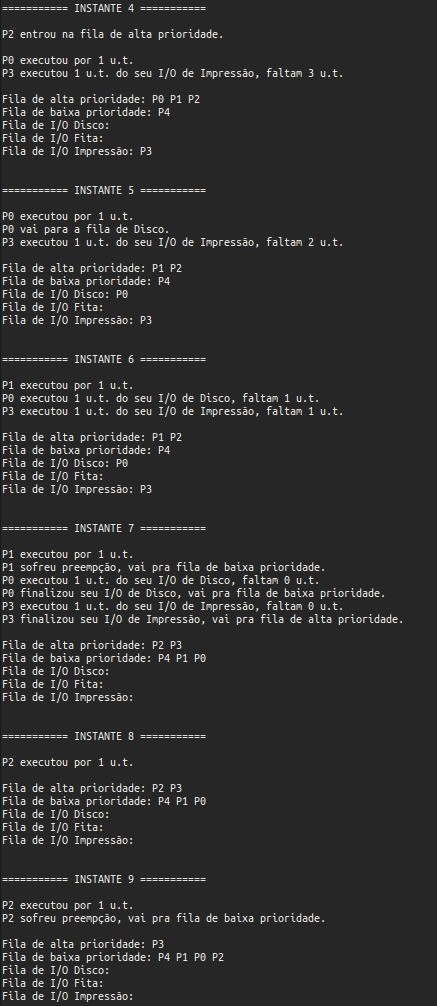
Para compilar e executar o simulador, anexamos um Makefile no diretório fonte onde também se encontram uma pasta src e uma pasta include, com os códigos em .c e os headers em .h, respectivamente. Basta utilizar o comando make quando dentro do diretório principal onde se encontram os arquivos e pastas mencionados.

# Referˆencias

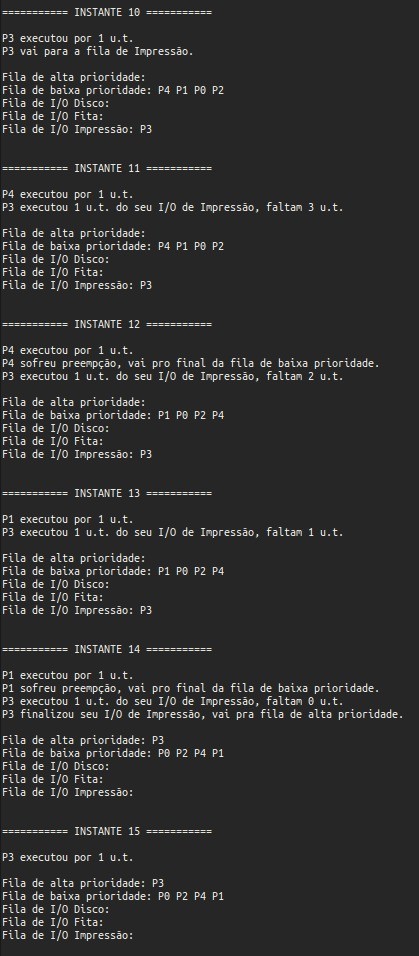
1. Brian W Kernighan and Dennis M Ritchie. The c programming language. 2002.
2. Carlos A Maziero. Sistemas operacionais: conceitos e mecanismos. *Livro aberto*, 2014.
3. William Stallings. *Operating systems internals and design principles*. Prentice-Hall, Inc., 1998.



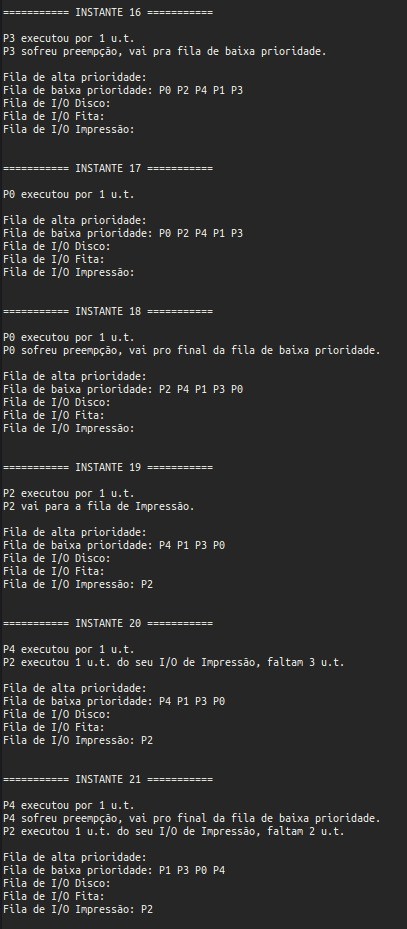
5



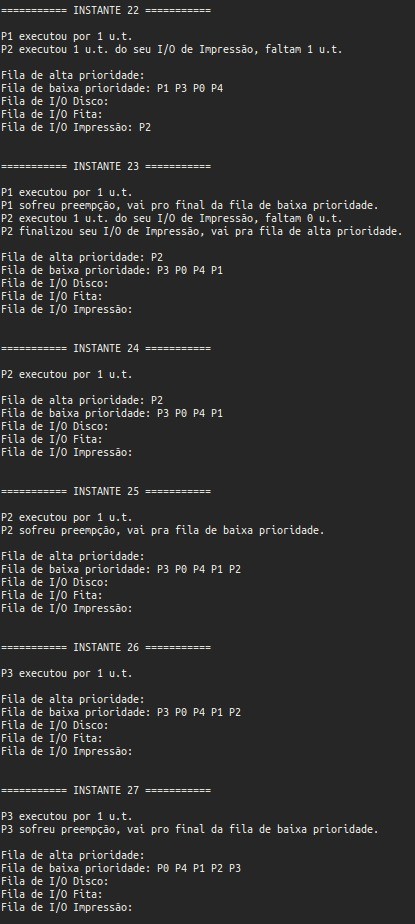
6



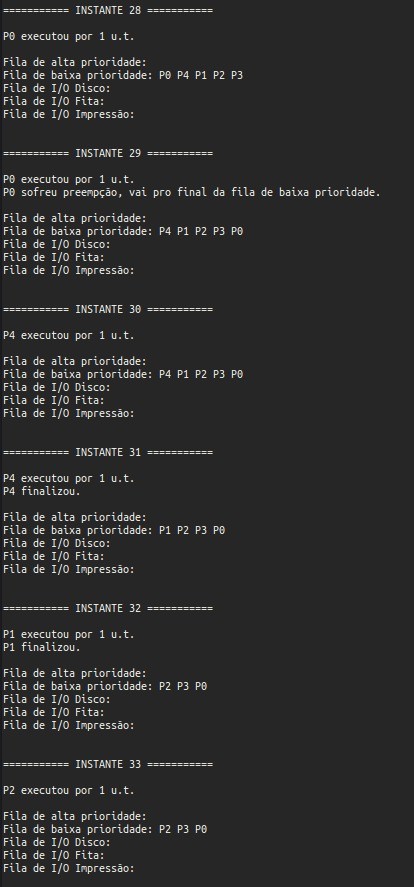
7



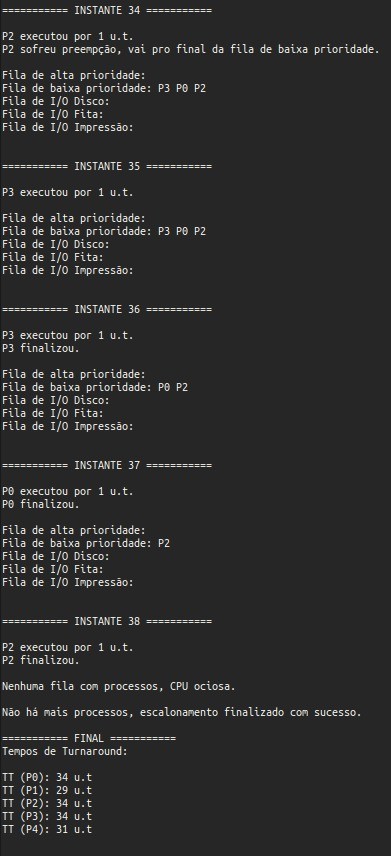
8



9



10



11