Exercício Programa 1

Escalonador de processos

Lucas Paiolla Forastiere e Marcos Siolin Martins

IME-USP

05 de outubro de 2020

Arquitetura do Shell

O shell segue a arquitetura sugerida em aula, tendo sido implementado apenas no arquivo `bccsh.c`.

Conta com um loop principal em que lê um comando por iteração e executa esse comando internamente por meio de chamadas de sistema ou realiza a invocação externa do binário informado até que um sinal `EOF` seja emitido pelo usuário (pressionar as teclas `CTRL+D`).

Arquitetura do Shell

Além disso algumas decisões de projetos foram tomadas, entre elas:

- A função `read_command(command, parameters)` recebe o comando digitado pelo usuário, usando a função `readline()`, e devolve o comando na variável `command` e os parâmetros na variável `parameters`. Por definição, `parameters[0] = command`;
- A função `readline()` aloca a memória necessária. Guardamos seu retorno no histórico usando `add_history()` e tratamos seu retorno substituindo espaços em branco pelo caractere '\0' e mudando os ponteiros de `parameters` para o começo de cada parâmetro na string.

Arquitetura do Shell

- Todos os arrays que não são alocados por funções externas são alocados estaticamente com valor máximo definido por diretivas `#define`:
  - `CUR_DIR_SIZE`: tamanho máximo do nome do diretório;
  - `PROMPT_SIZE`: tamanho máximo da string exibida no prompt;
  - `MAX_PARAMETERS`: quantidade máxima de parâmetros.
- Por fim, implementamos a chamada de sistema `mkdir` passando como parâmetro a constante `S_IRWXU` que dá ao usuário todas as permissões sobre aquele diretório.

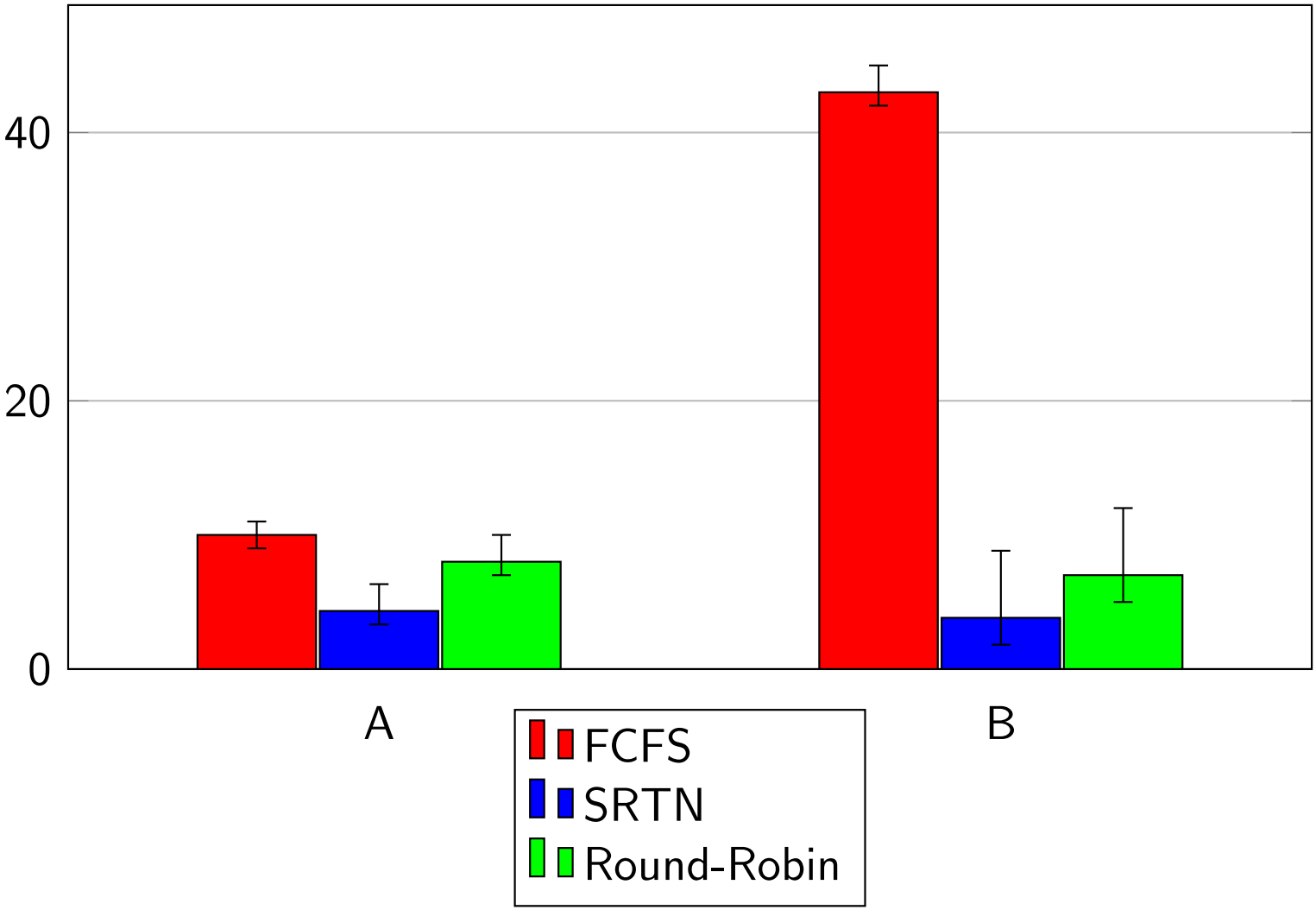
Implementação dos Escalonadores

Inicialmente, todas as threads que eventualmente chegarão no sistema são carregadas do arquivo de entrada, criadas e ficam bloqueadas até que o escalonador lhes dê a permissão de rodar.

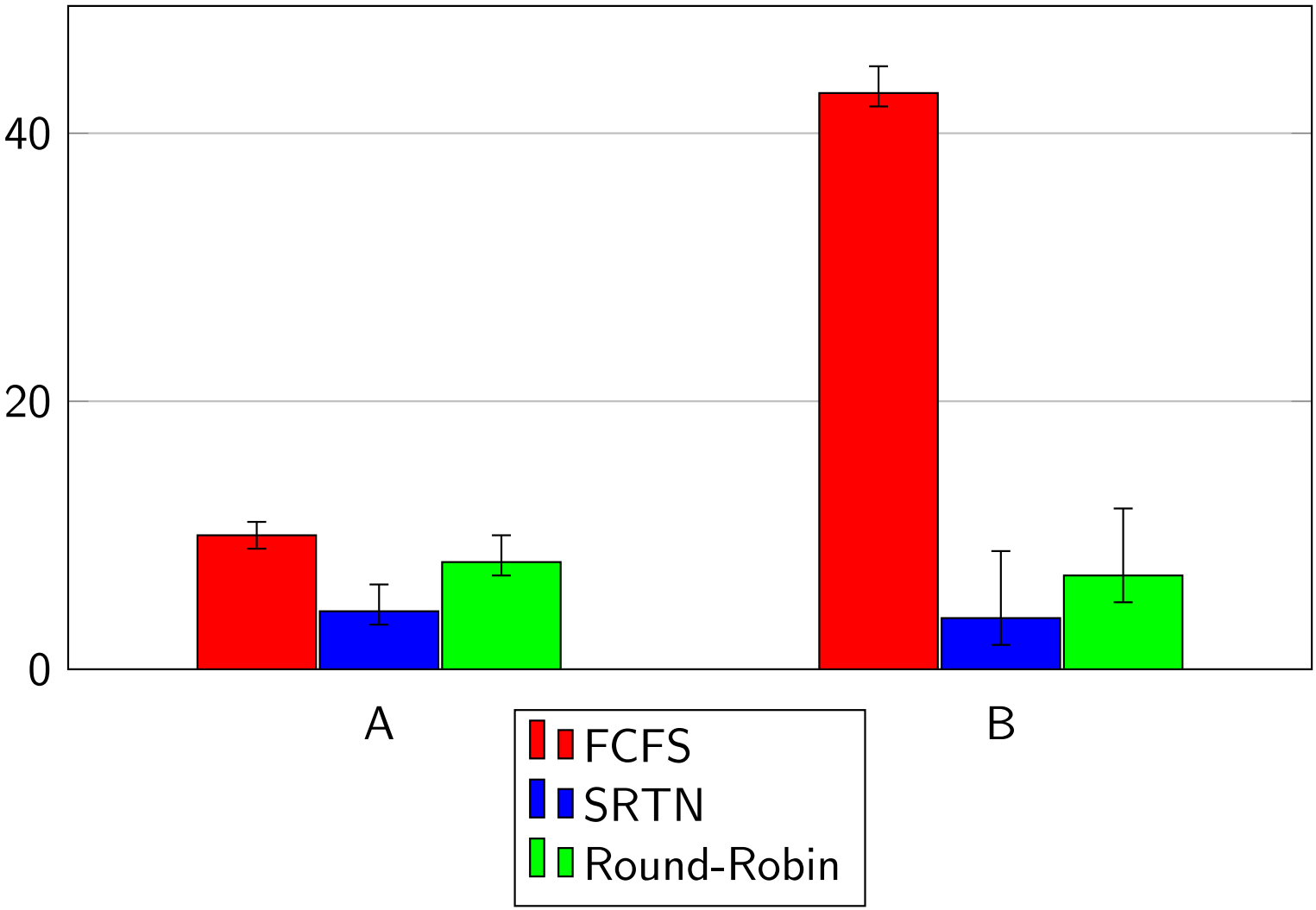
Para fazer esse gerenciamento das threads, existe um array de mutex (chamado `mutex`) em que cada mutex está associado a uma thread. Se o mutex está liberado, então a thread pode rodar. Caso contrário, a thread fica bloqueada. Usamos os mutex da biblioteca pthread para fazer esse gerenciamento.

Implementação dos Escalonadores

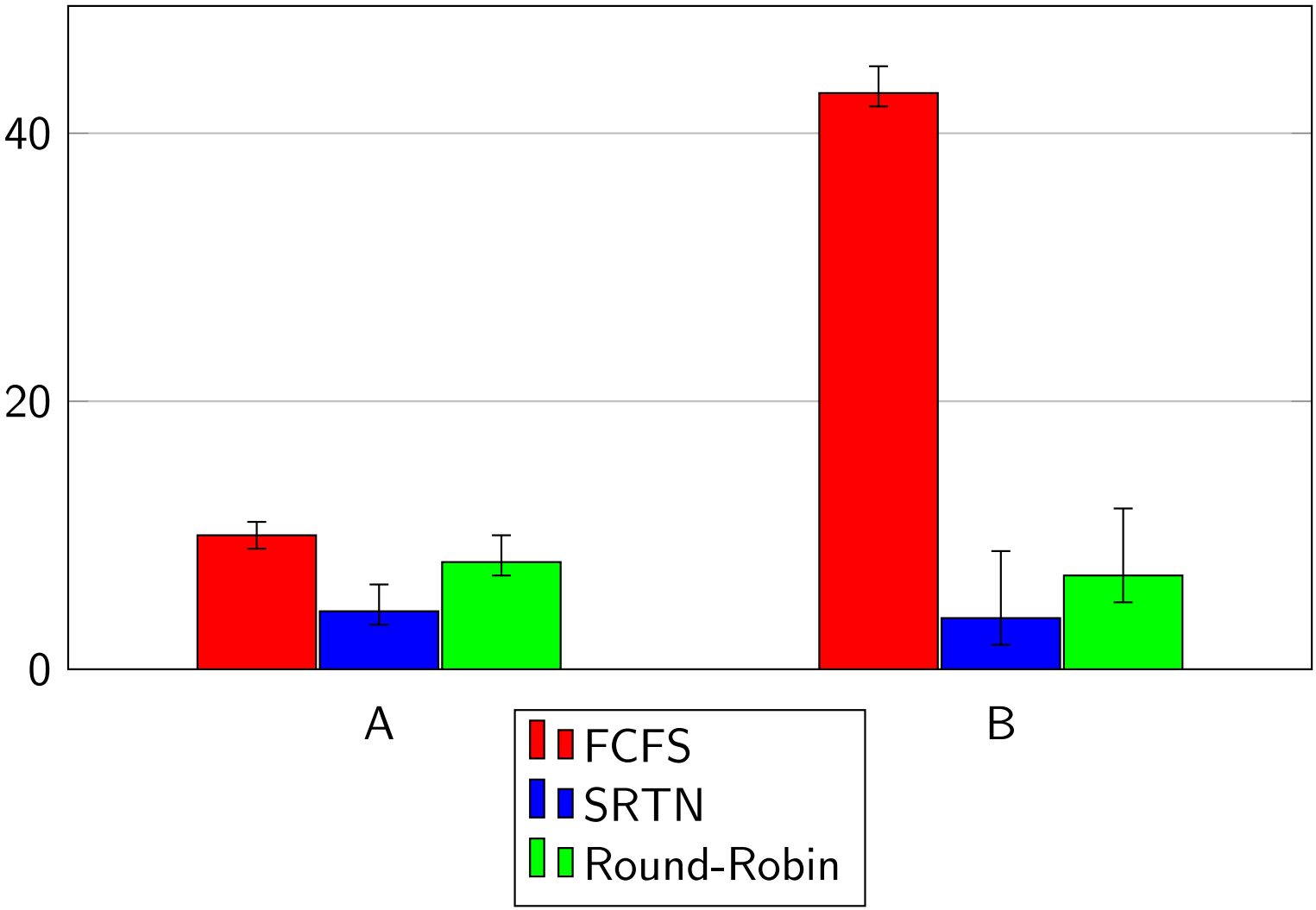
Arquivo de Trace: 10 processos



Arquivo de Trace: 100 processos



Arquivo de Trace: 1000 processos



Conclusões dos experimentos