Relatório - MAC0422 - EP1

Matheus Barbosa Silva e Gustavo Santos Morais

Instituto de Matemática e Estatística

05 de Oututbro de 2020

Sumário

Sumário

Arquitetura do Shell (bccsh)

A implementação aloca **dinamicamente** a memória necessária para exibir o prompt de usuário atualizado a cada instante.
A função `strtok` – da biblioteca `string.h` – separa cada argumento dos comandos digitado pelo usuário, possibilitando a execução dos seguintes funcionalidades:

- `ln`: executa a função `symlink` da biblioteca `unistd`, criando um link simbólico;
- `mkdir`: cria e atribui as permissões padrão a um novo diretório por meio de sua chamada de sistema de mesmo nome em `sys`;
- `kill`: executa diretamente sua chamada de sistema correspondente na biblioteca `signal`, enviando o sinal `SIGKILL` ao processo dado;
- Execução de Binários: separa os argumentos do comando por meio de `strtok` e demanda sua execução por meio de `execve`.

Sumário

Escalonadores

Alguns aspectos importantes a serem considerados acerca do escalonador são:

- **Organização do escalonador**: todos os escalonadores compartilham uma camada em comum dentro da função *atualizaExecucao* em *escalonador.h*.
  Essa camada é responsável pelo gerenciamento da dinâmica básica de um escalonador, manipulando a fila de prontos e atualizando o processo em execução, quando necessário. De forma complementar, cada escalonador realiza o tratamento do atual processo em execução de diferentes modos de acordo com o seu objetivo.

Escalonadores

- **Execução da Thread**: cada thread realiza operações aritméticas em um laço de repetição e, em seguida, é adormecida – ambas operações levam tempo proporcional ao tempo de execução da thread (1s ou um quantum $\approx$ 0.8s);
- **Manipulação de Threads**: utilizam-se *mutexes* para controlar o fluxo das threads e pausar suas execuções a cada segundo - permitindo trocas quando necessário.
  Também foi controlado, paralelamente, o fluxo de execução do programa principal; permitindo que este prossiga apenas após o término da execução da thread.

Escalonadores

Foram implementados os escalonadores a seguir com as decisões de projeto adequadas:

1. **FCFS**: implementado de acordo com a sua definição;
2. **SRTN**: para processos que chegam quando não há um processo em execução (a execução anterior terminou neste instante ou o processador está em espera), seus tempos de execução são comparados com o primeiro processo da fila. Ao realizar preempção, o processo anteriormente em execução é inserido no final da fila;
3. **Round Robin**: para um único processo em execução, sua manipulação segue a dinâmica padrão (com constantes mudanças de contexto). Define-se quantum ≈ 0.8s.

Geração dos arquivos de teste

Para testar o funcionamento do escalonador, foram gerados três arquivos de entrada de 10 (denominado pequeno), 100 (médio) e 1000 (grande) processos respectivamente, com o mesmo formato descrito no enunciado do EP.
Essas entradas foram geradas aleatoriamente, de modo a proporcionar conclusões menos enviesadas.
Os experimentos foram realizados em duas máquinas com quantidades de núcleos distintas – denominadas **Máquina 1** e **Máquina 2**, com 4 e 8 núcleos, respectivamente.
Usamos esses arquivos para averiguar a funcionalidade e eficiência dos três escalonadores implementados, para diferentes tipos de entrada, tendo como resultado, os gráficos apresentados a seguir.

Cumprimento de Deadlines - Entradas Pequenas

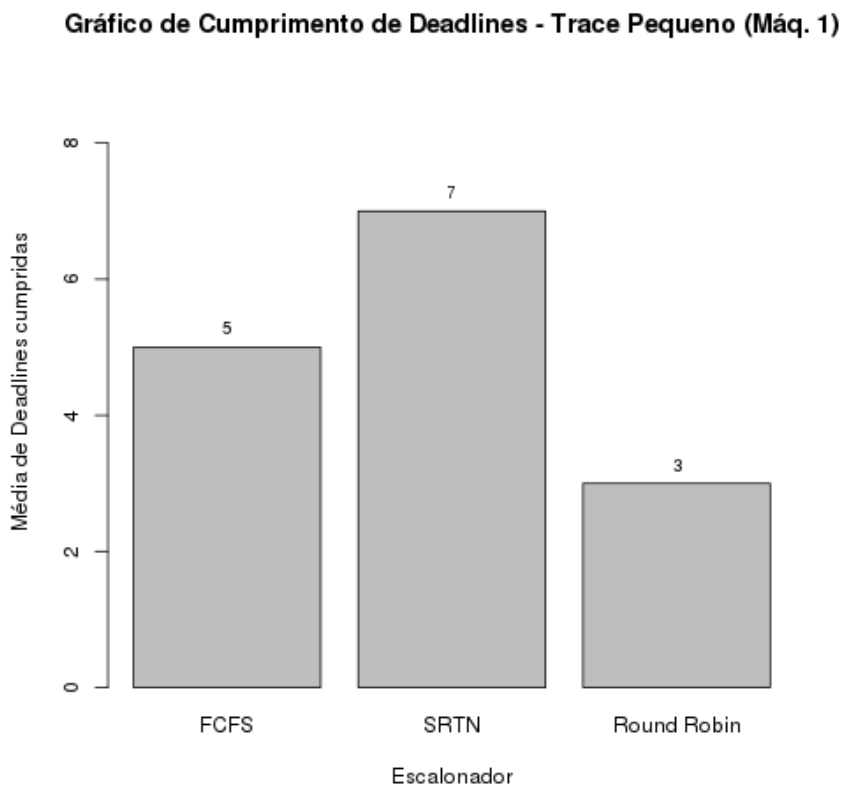

Figura 1: Gráfico da Média de Cumprimento de Deadlines - Trace Pequeno (Máquina 1).

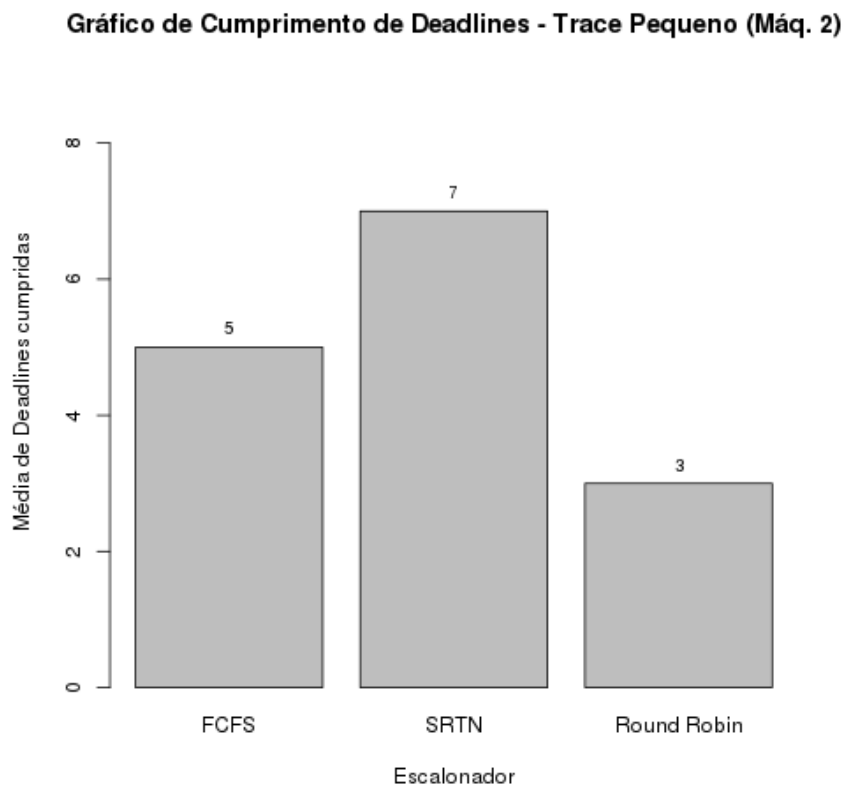

Figura 2: Gráfico da Média de Cumprimento de Deadlines - Trace Pequeno (Máquina 2).

Cumprimento de Deadlines - Entradas Médias

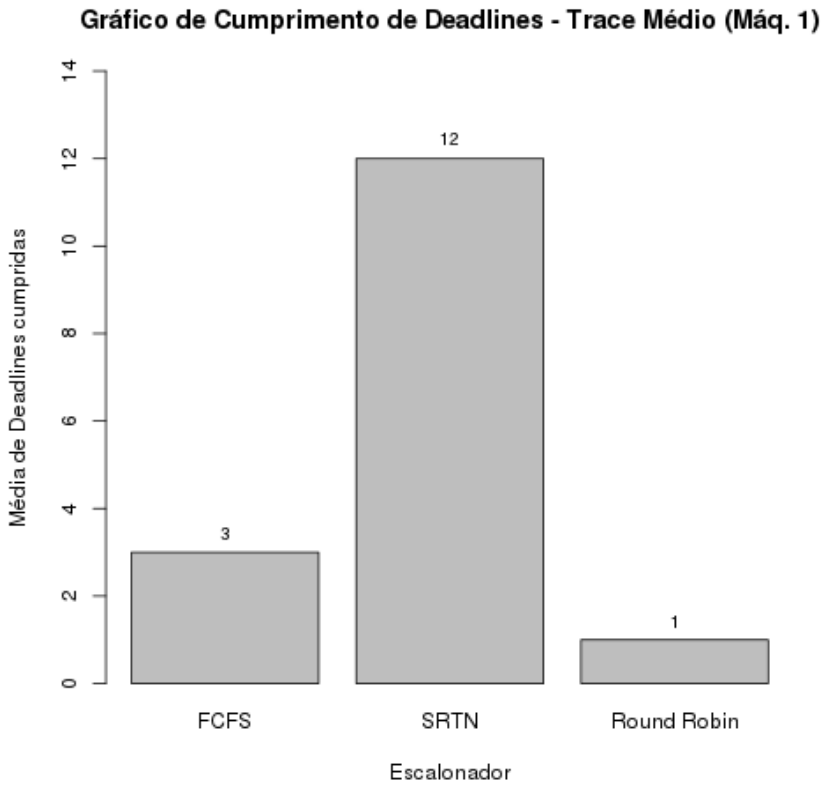

Figura 3: Gráfico da Média de Cumprimento de Deadlines - Trace Médio (Máquina 1).

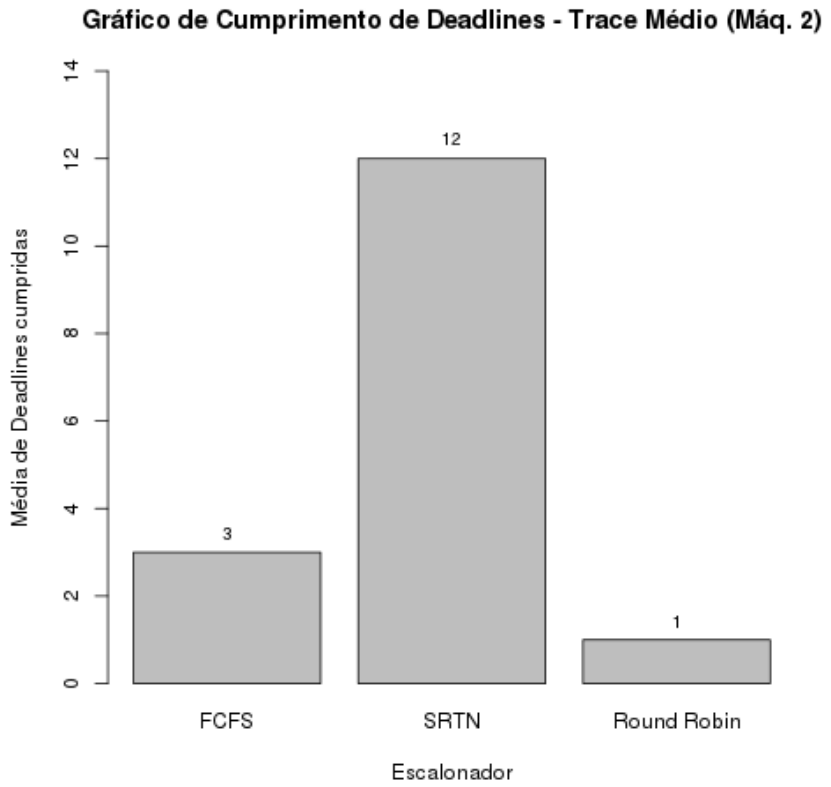

Figura 4: Gráfico da Média de Cumprimento de Deadlines - Trace Médio (Máquina 2).

Cumprimento de Deadlines - Entradas Grandes

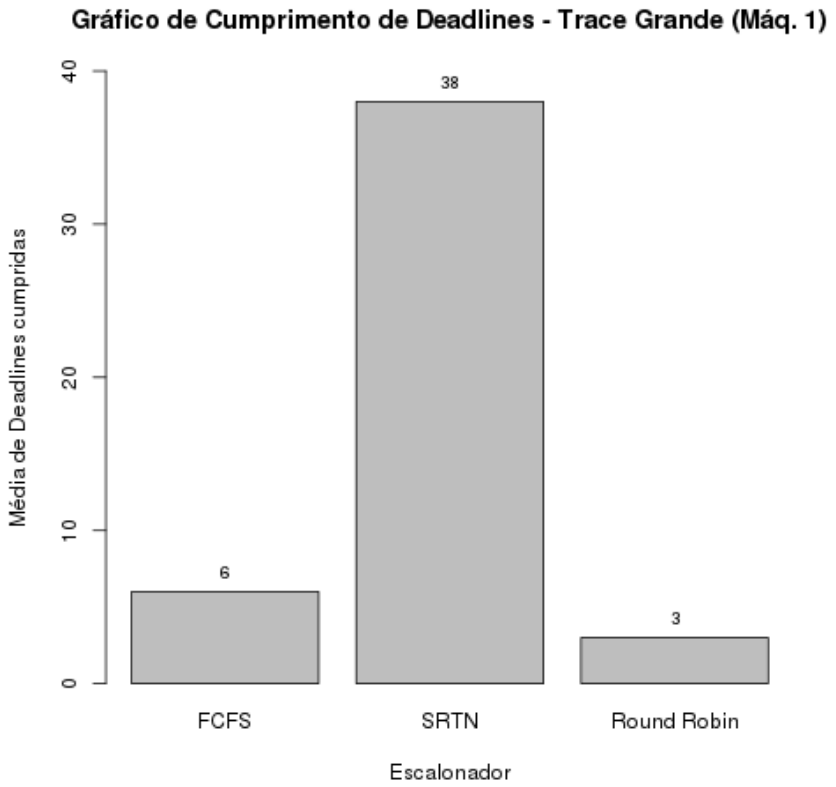

Figura 5: Gráfico da Média de Cumprimento de Deadlines - Trace Grande (Máquina 1).

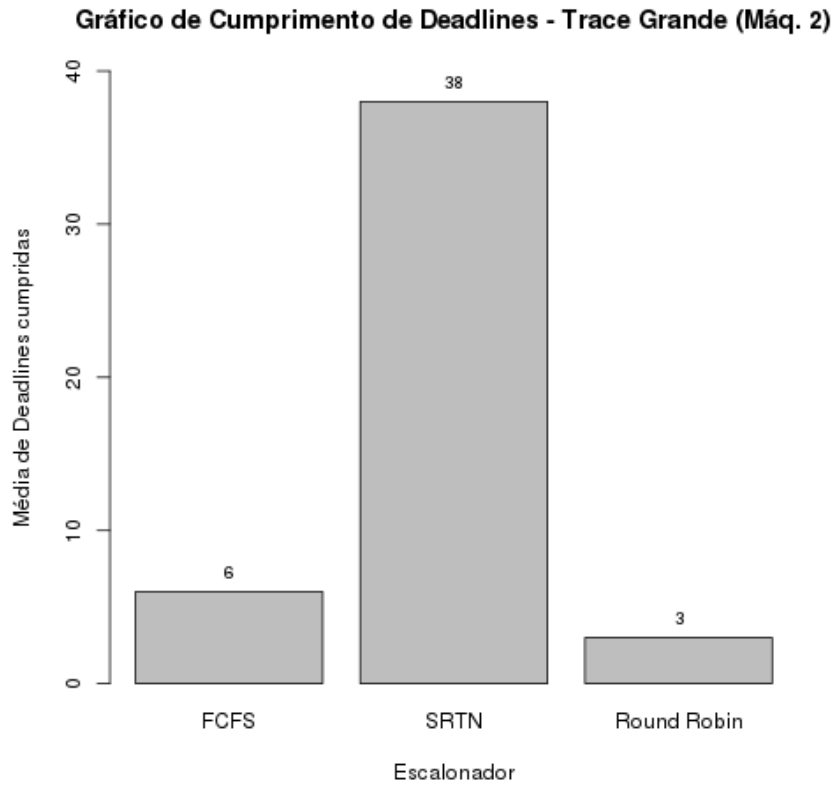

Figura 6: Gráfico da Média de Cumprimento de Deadlines - Trace Grande (Máquina 2).

Cumprimento de Deadlines - Conclusões

- Conforme o esperado, o escalonador SRTN apresentou os melhores resultados para todos os tipos de entrada;
- O escalonador Round Robin apresentou os piores resultados - dado o seu menor tempo de execução e maior interatividade;
- O FCFS apresentou, em todos os casos, resultados razoavelmente melhores que o Round Robin;
- O algoritmo construído demonstra ser determinístico quanto ao cumprimento de deadlines – todas as 30 execuções de cada arquivo em cada escalonador obtiveram os mesmos resultados.

Mudanças de Contexto - Entradas Pequenas

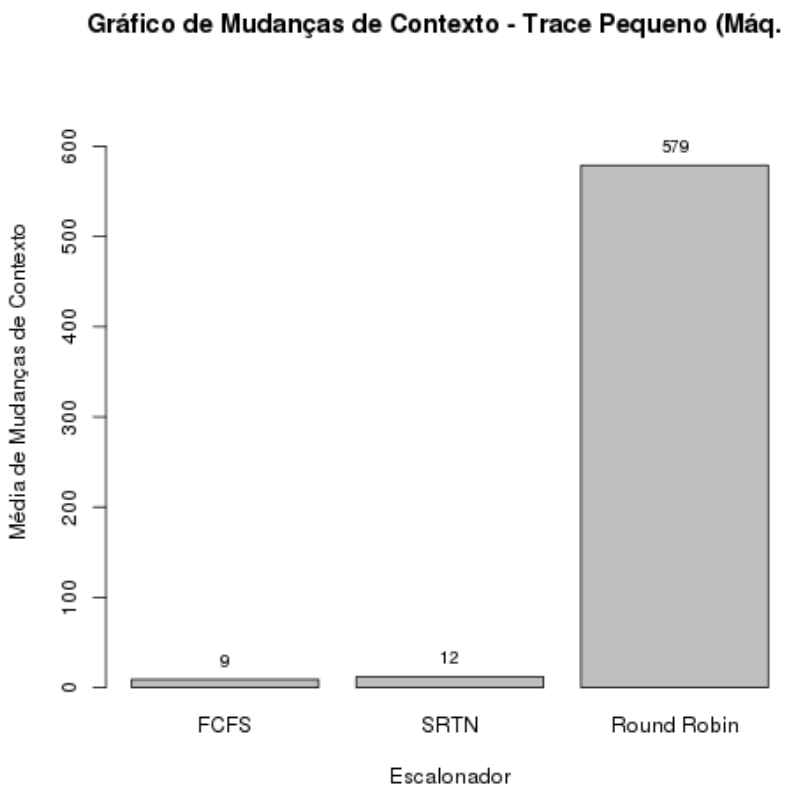

Figura 7: Gráfico da Média de Mudanças de Contexto - Trace Pequeno (Máquina 1).

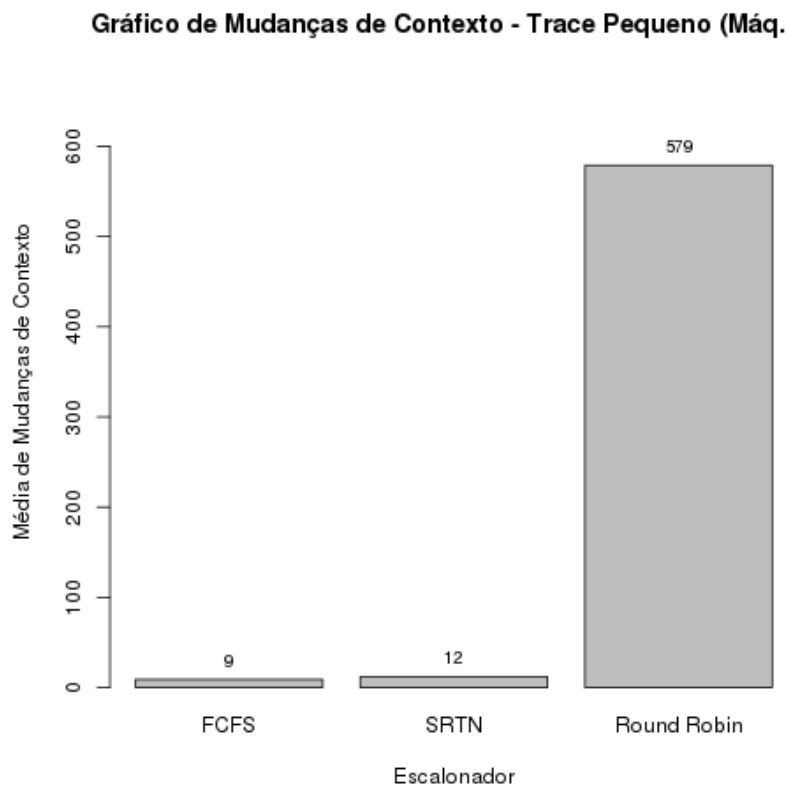

Figura 8: Gráfico da Média de Mudanças de Contexto - Trace Pequeno (Máquina 2).

Mudanças de Contexto - Entradas Médias

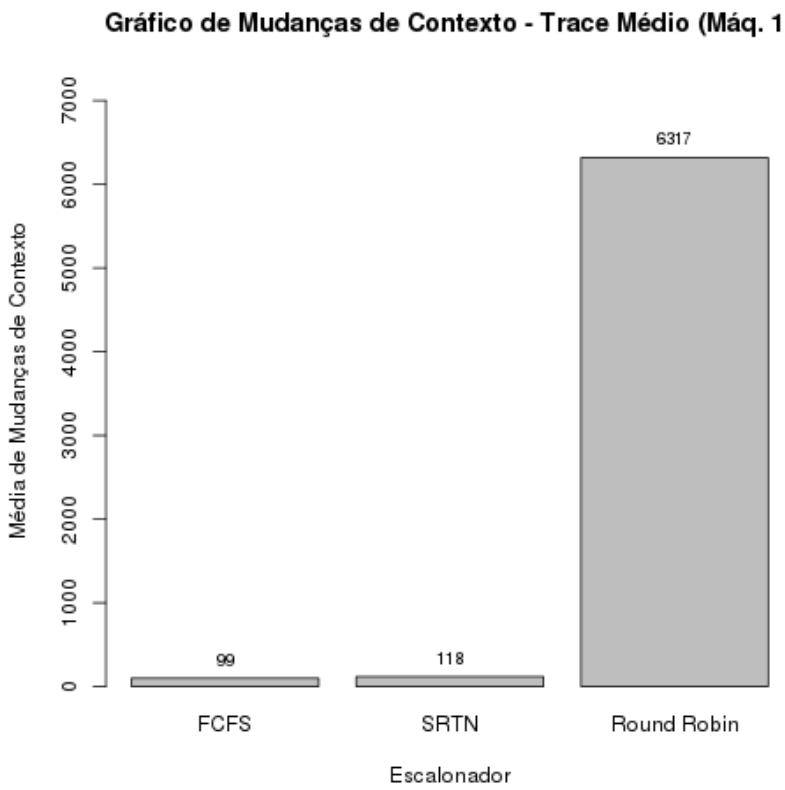

Figura 9: Gráfico da Média de Mudanças de Contexto - Trace Médio (Máquina 1).

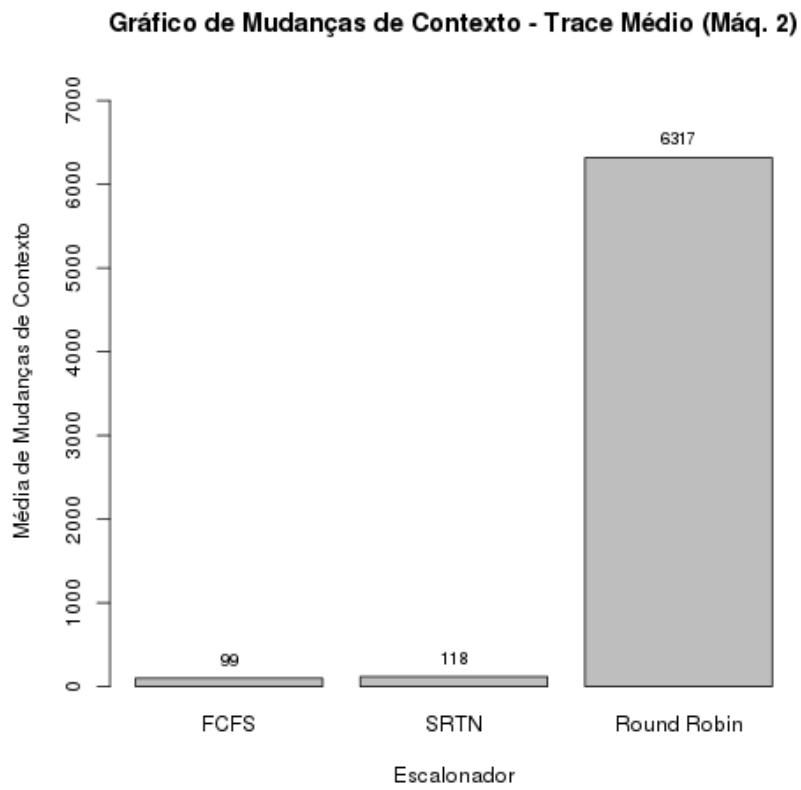

Figura 10: Gráfico da Média de Mudanças de Contexto - Trace Médio (Máquina 2).

Mudanças de Contexto - Entradas Grandes

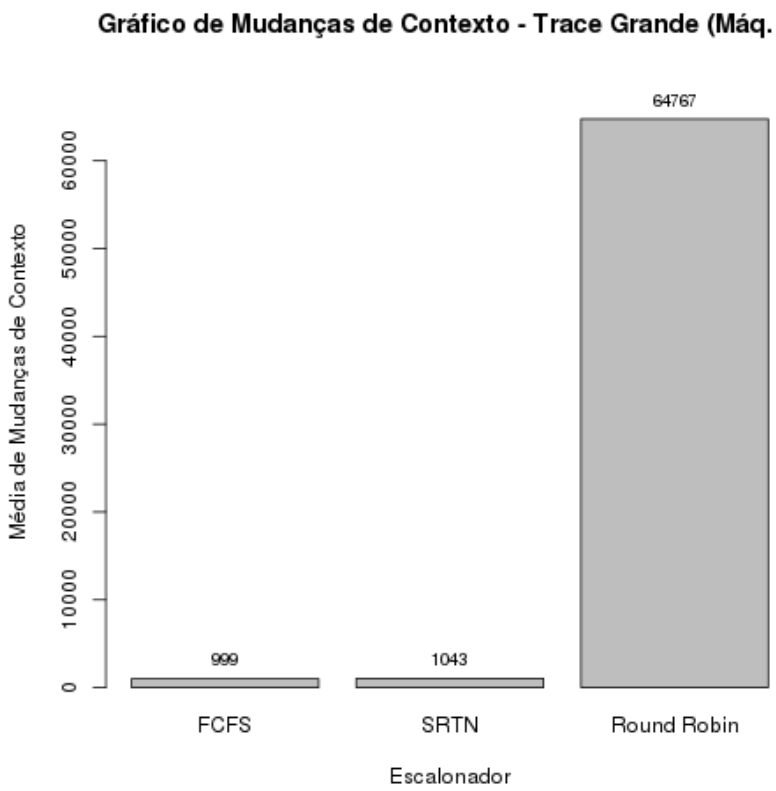

Figura 11: Gráfico da Média de Mudanças de Contexto - Trace Grande (Máquina 1).

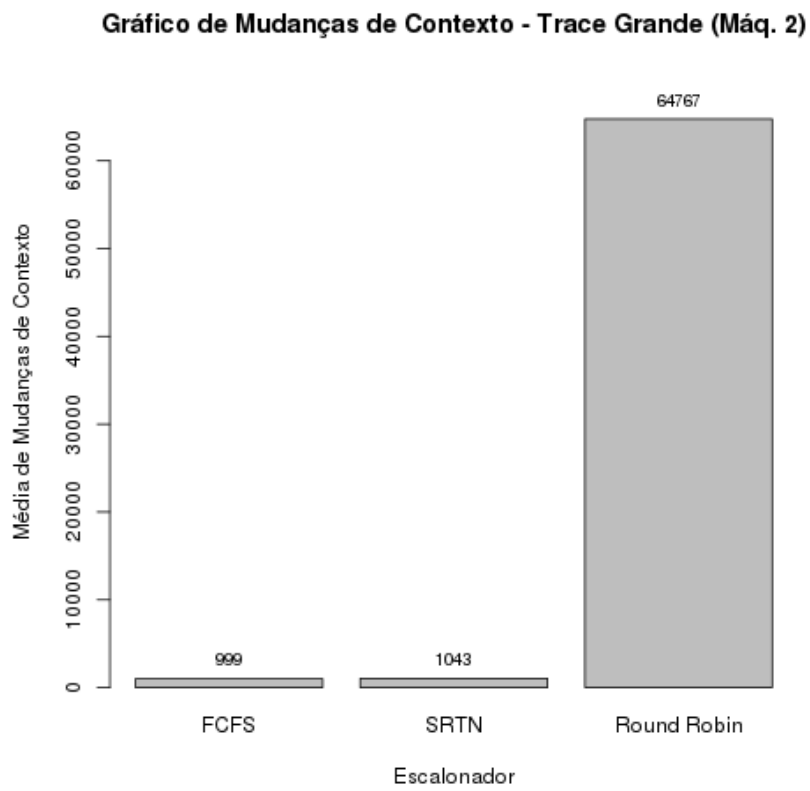

Figura 12: Gráfico da Média de Mudanças de Contexto - Trace Grande (Máquina 2).

Mudanças de Contexto - Conclusões

- Conforme o esperado, o escalonador Round Robin apresentou uma quantidade expressivamente maior de mudanças de contexto;
- O escalonador FCFS ocasionou uma quantidade de mudanças de contexto igual a quantidade de processos subtraída de um;
- O escalonador SRTN apresentou um número de mudanças de contexto maior que o FCFS (mas significativamente menor que o Round Robin) devido a sua dinâmica de mudança de contexto de acordo com processos recém-chegados;
- O algoritmo construído demonstra ser determinístico quanto às mudanças de contexto – todas as 30 execuções de cada arquivo em cada escalonador obtiveram os mesmos resultados.