Apresentação Ep1

MAC0422 - Sistemas Operacionais
Professor: Daniel Batista

12/set/2016

por:
Matheus Tavares Bernardino, nº USP: 9292987
Marcos Vinicius do Carmo Sousa, nº USP: 9298274

Shell

Principais Funções

- char *getPromptText ();
  - Retorna o uma string contendo o texto a ser usado no prompt.
- void read_command (char *promptText);
  - Imprime o texto promptText na saída padrão e lê um comando da entrada padrão, com seus possíveis parametros, salvando-os respectivamente, nas variaveis globais command e parameters.
- void chh ();
  - Executa o comando chmod
- void idu ();
  - Executa o comando id -u

main

char *promptText = getPromptText ();

- While (1) {

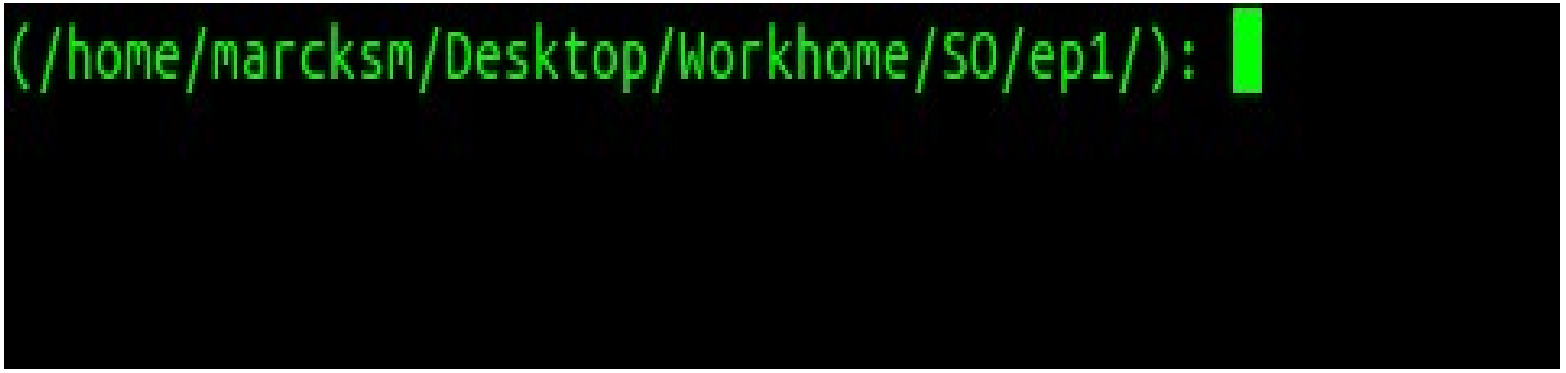
```
read_command(promptText);
If (command == exit)        break
If (command == chmod)    chh();
If (command == id -u)        Idu();
Else {
    pid = fork();
    if (pid == -1) error (“Fail”);
    else if (pid == 0) execve (command, parameters, 0);
    else waitpid (pid, &status, 0);
    }
```

}

clear_history();

•char *getPromptText ();

- Retorna string no formato "(dir/): "
  - char *dir = getcwd (); retorna o diretório
  - malloc ((strlen (dir) + 6) * sizeof (char));



void read_command (char *promptText);

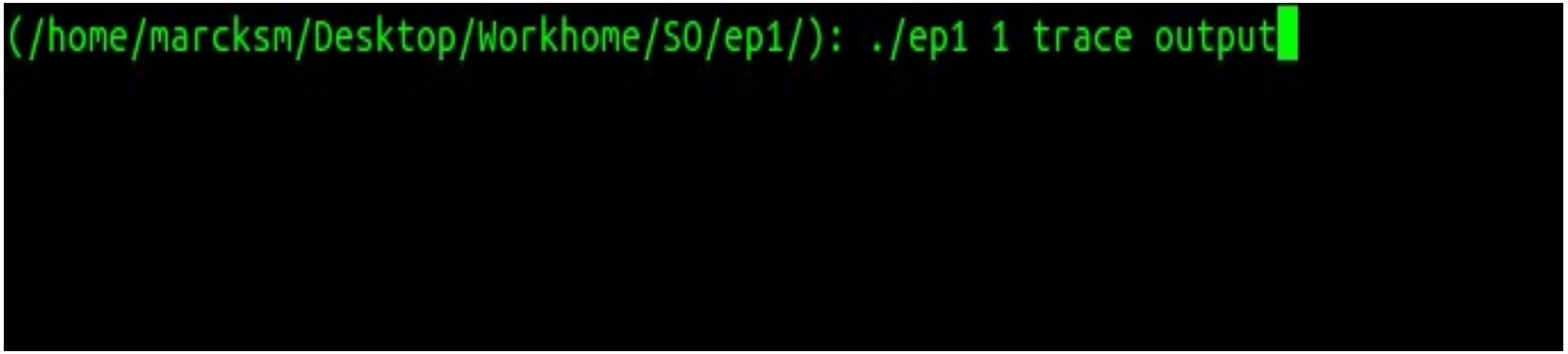
char *line_read = readline (promptText); // readline/readline.h

- Faz a leitura da linha de comando

- parameters = wordsIn (line_read);
  - Vetor que contem os parametros da entrada
  - Parameters[0] = commando
  - parameters[i], i > 0 = parametros do comando

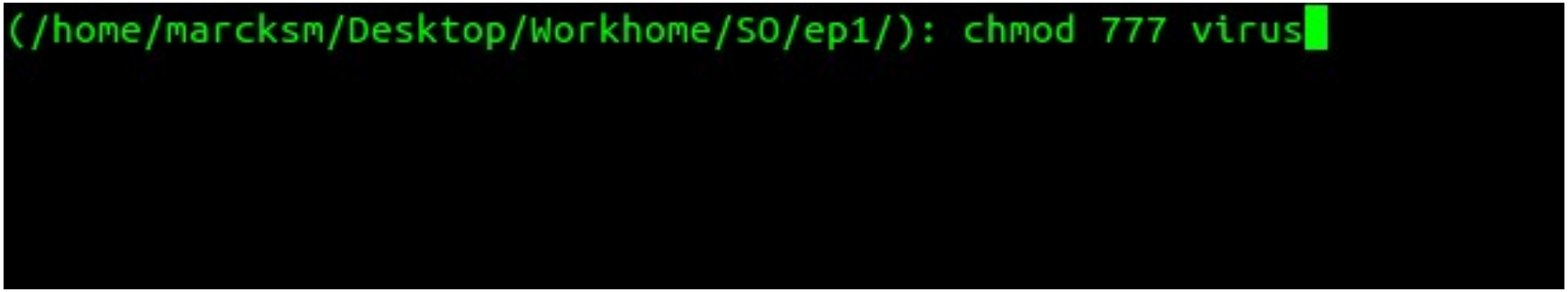
char **parameters; é global

- add_history (line_read); readline/history.h



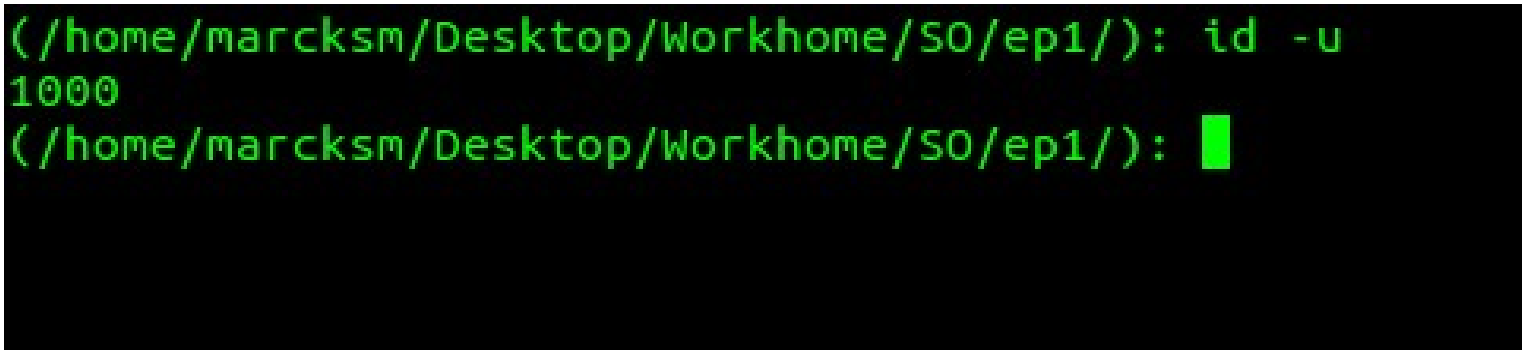
void chh (void)

- pl = strtol (parameters[1], 0, 8); //stdlib.h
- chmod (parameters[2], pl); //sys/stat.h



•void idu ();

- id = geteuid(); //sys/types.h



Simulador de Processos

Considerações gerais:

O simulador de processos foi implementado com uma pequena camada comum à todos os escalonadores:

- Leitura do arquivo trace
- Algumas variáveis globais guardam informações úteis à mais de um escalonador. (ex: variável que guarda o tempo inicial de simulação, variável que guarda informações do processo em execução, se houver um e etc.)
- Função que permite saber o tempo total de simulação até o momento que for chamada.

Demais funções e mecanismos, embora existam alguns mecanismos semelhantes, foram implementados separadamente para cada escalonador.

Implementação dos escalonadores

Implementação dos escalonadores:

- **Quanto à chegada de novos processos**
- **Quanto à parada de processos em execução**
- **Quanto à retomada de processos**

Implementação dos escalonadores:
Quanto à chegada de novos processos

- Para a simulação dos escalonadores FCFS e Múltiplas Filas foram criadas funções nomeadas de “gate” expecíficas para cada um. Estas funções são executadas por uma thread exclusiva e são responsáveis por acrescentar, no tempo certo, novos processos à fila de processos dos escalonadores.

- No caso da simulação do escalonador SRTN, como o evento da chegada de novos processos é gatilho para algumas ações/decisões do escalonador quem cuida do acréscimo de novos processos na fila de processos é a própria função do escalonador. Portanto não há thread a mais para esta função.

Implementação dos escalonadores:

- Quanto à chegada de novos processos
- **Quanto à parada de processos em execução**
- Quanto à retomada de processos

Implementação dos escalonadores: Quanto à parada de processos em execução

- Nos três escalonadores, a parada de um processo em execução (seja porque o seu tempo total de execução acabou ou porque o escalonador “decidiu” tirá-lo da CPU) é feita pela função sendo executada pela própria thread que simula um processo em execução. O modo como isso é feito, porém, é diferente para cada escalonador:

Implementação dos escalonadores:
Quanto à parada de processos em execução (FCFS)

- Para a simulação do escalonador FCFS, periodicamente a função que a thread está executando checa se o tempo rodado já alcançou o tempo total de execução do processo. Caso positivo, a thread finaliza. (não há preempção)

Implementação dos escalonadores:
Quanto à parada de processos em execução (SRTN)

- No caso da simulação do escalonador SRTN, além de checar se o processo já rodou o tempo total, a função que simula um processo também faz checagens periódicas a uma variável global através da qual o escalonador ordena para o processo sair da CPU. A thread é finalizada caso está variável indique ordem de finalização ou o tempo total de execução do processo já tiver sido alcançado. (este mecanismo é o que permite ao escalonador fazer preempção)

Implementação dos escalonadores:
Quanto à parada de processos em execução
(Múltiplas Fílas)

- Por fim, na simulação do Múltiplas filas, a função também checa se o processo já rodou o tempo total e checa, ainda, se já rodou o “quantum” que o processo tinha para rodar. Caso alguma destas checagens seja positiva, a thread finaliza. (a preempção é feita através da checagem do tempo rodado em comparação com o quantum)

Implementação dos escalonadores:

- Quanto à chegada de novos processos
- Quanto à parada de processos em execução
- **Quanto à retomada de processos**

Implementação dos escalonadores:
Quanto à retomada de processos

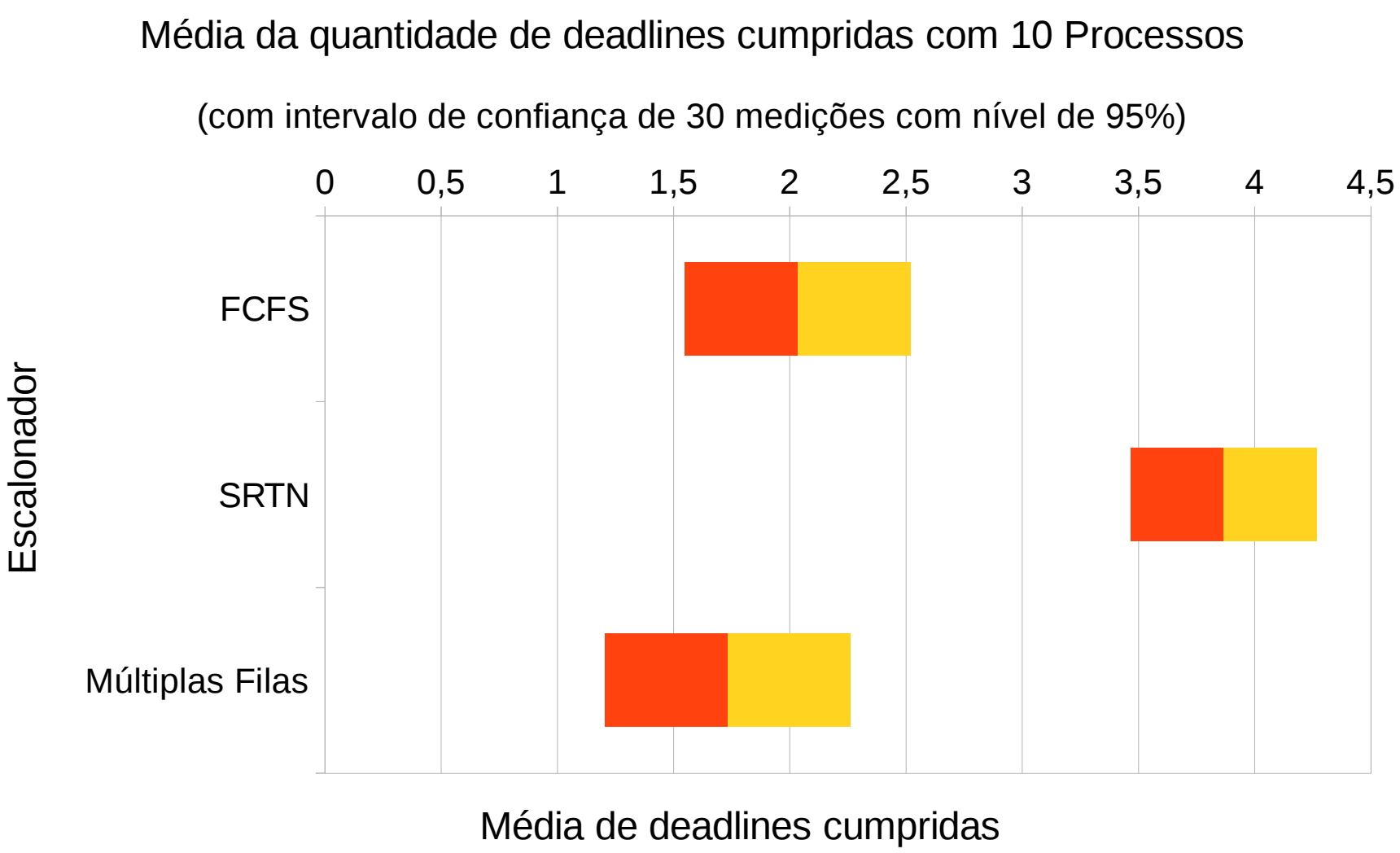
- Tirando o escalonador FCFS, onde um processo em execução, executa até o fim do seu tempo total de execução, todos os demais fazem preempção. Como é feita a retomada dos processos neste caso?

- Toda vez que um processo vai ser executado, ele sai da fila de processos e um thread é criada para simula-lo rodando. Depois de certo tempo, a thread finaliza ou pelo motivo do tempo de execução total do processo ter sido alcançado ou por preempção do escalonador. No segundo caso, o processo volta para a fila, com suas informações atualizadas. E quando for sua vez de rodar de novo, uma nova thread é criada e roda até um dos dois motivos citados anteriormente a interromperem. Este ciclo é repetido até que o processo consiga rodar o seu tempo total de execução.

- Logo, a retomada de processos é feita tal como a primeira execução de um processo, criando uma nova thread para ele e finalizando-a quando necessário.

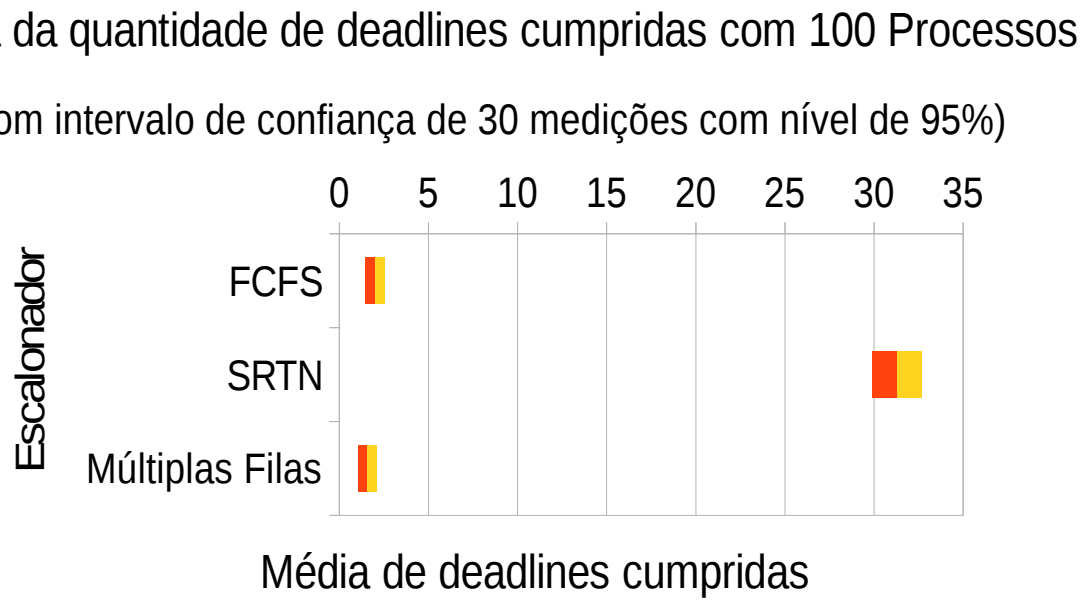
Resultados de testes com o simulador de Processos

Quanto ao cumprimento das deadlines

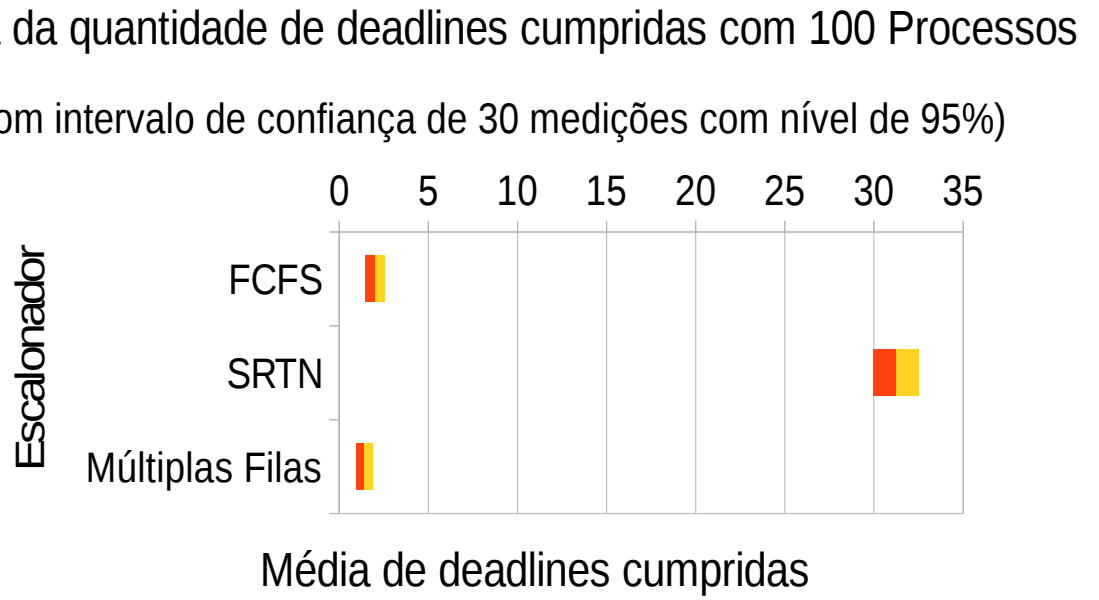


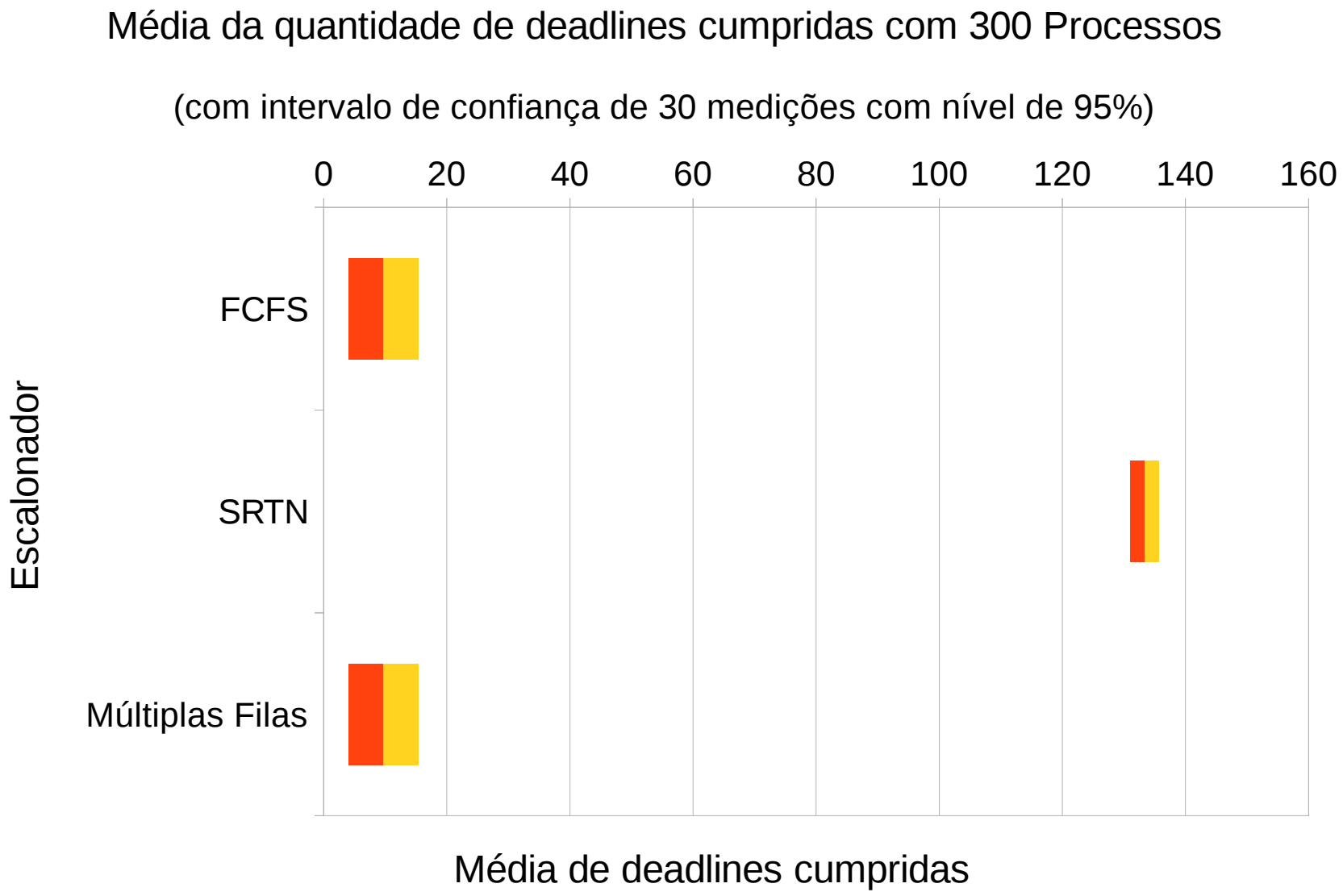
***obs: as médias e intervalos de confiança para este gráfico foram idênticos nas máquinas A (4 cores) e B (8 cores)**

Máquina A (4 cores)



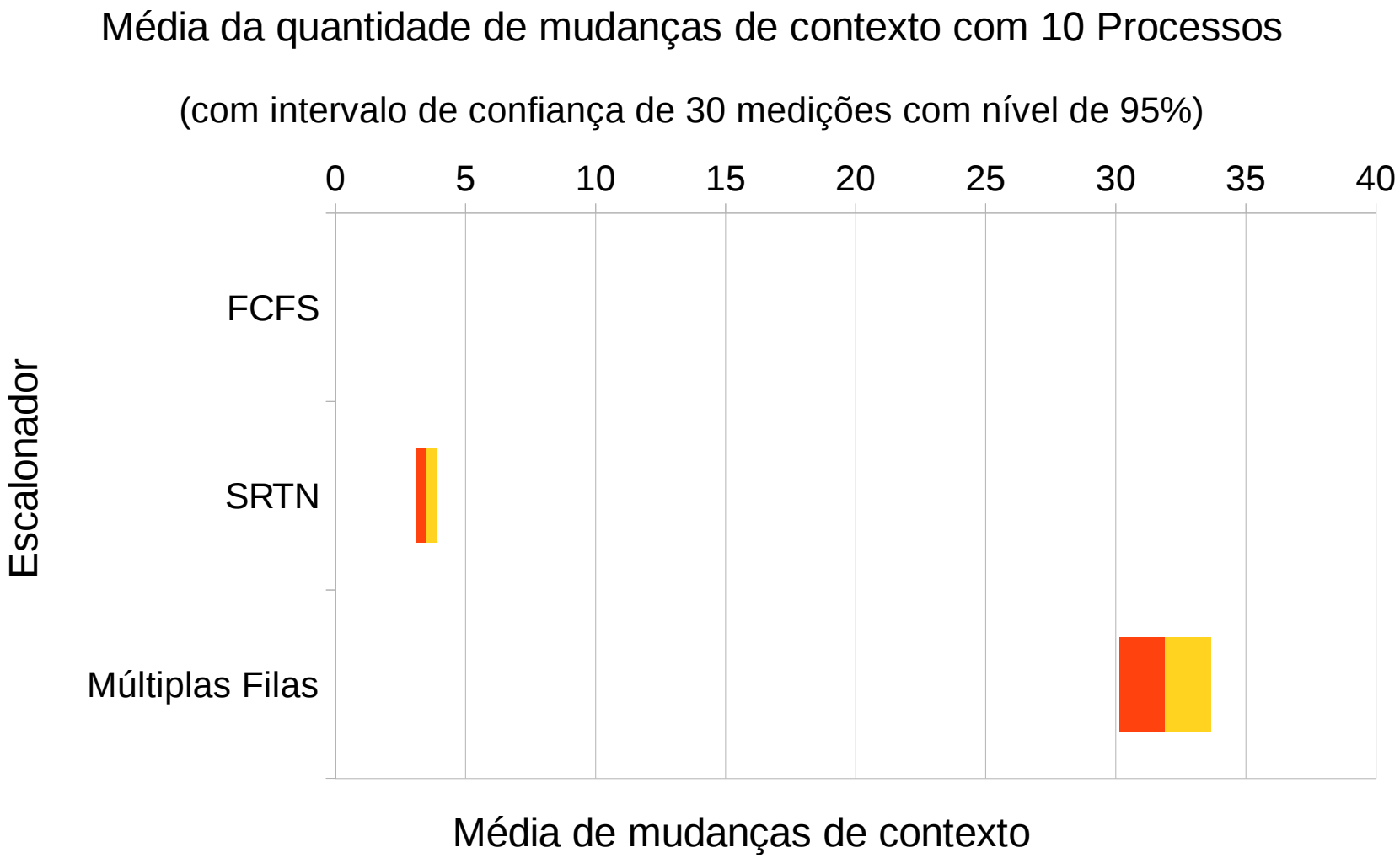
Máquina B (8 cores)





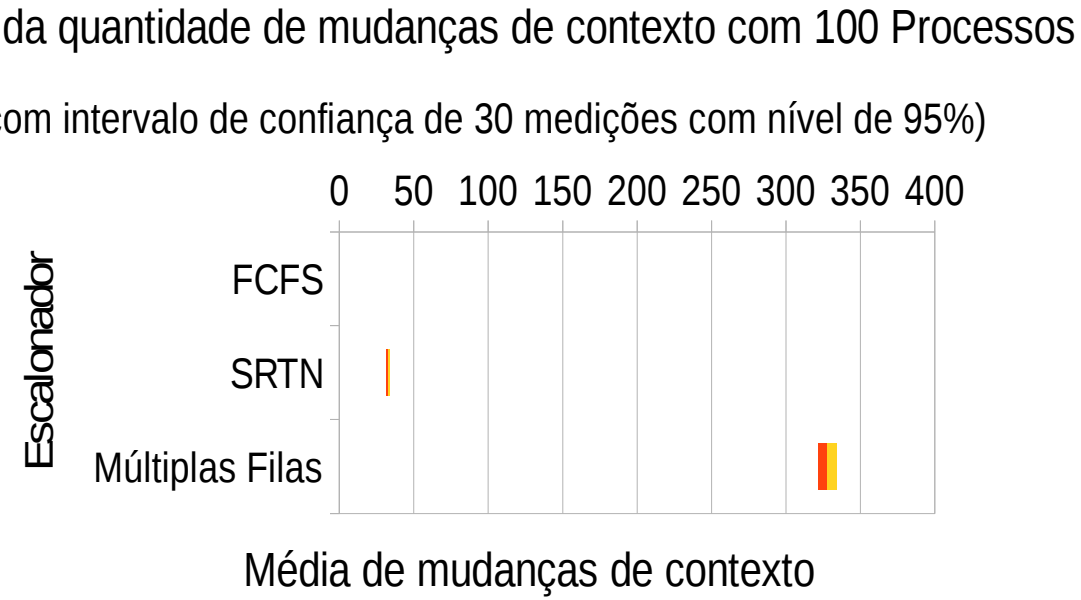
***obs: as médias e intervalos de confiança para este gráfico foram idênticos nas máquinas A (4 cores) e B (8 cores)**

Quanto à quantidade de mudanças de contexto

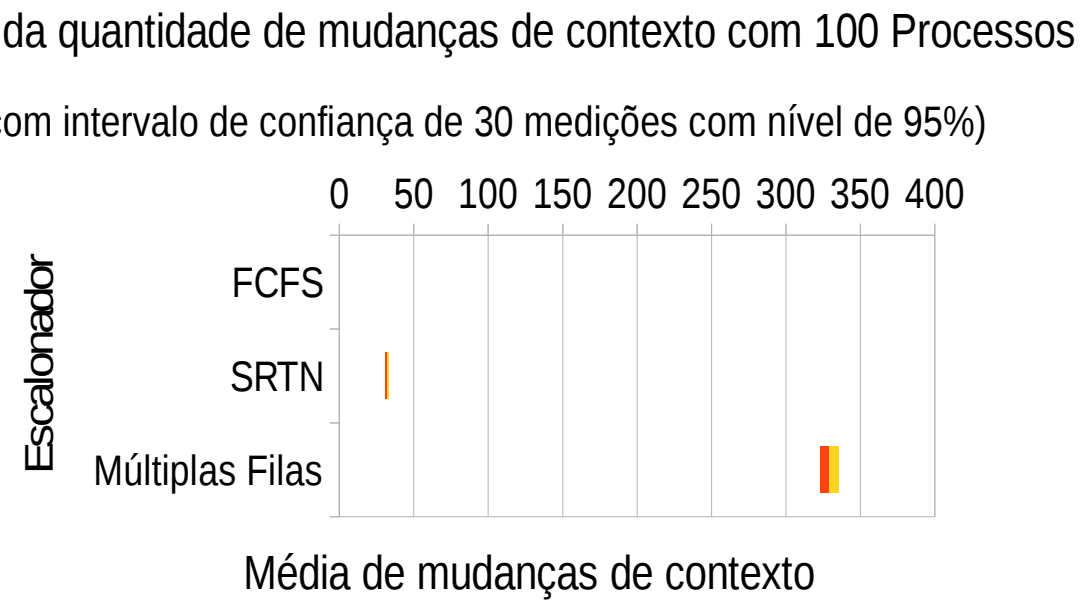


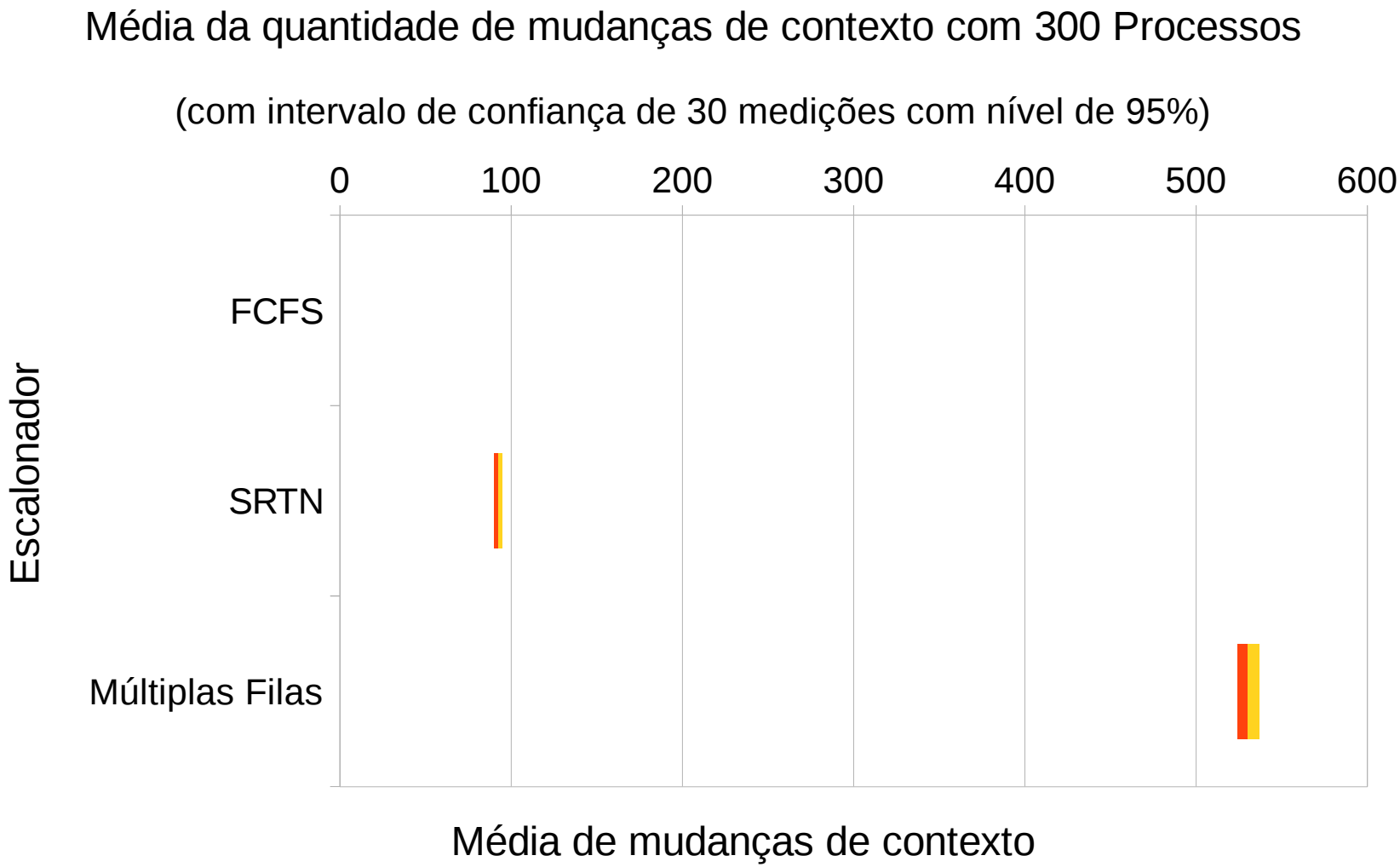
***obs: as médias e intervalos de confiança para este gráfico foram idênticos nas máquinas A (4 cores) e B (8 cores)**

Máquina A (4 cores)



Máquina B (8 cores)





***obs: as médias e intervalos de confiança para este gráfico foram idênticos nas máquinas A (4 cores) e B (8 cores)**

***obs: Tivemos um problema durante a execução dos testes com 100 processos no simulador de processos na máquina B (8 cores). O computador “travou” durante o processo e os testes tiveram de ser reiniciados. Provavelmente devido à essa falha ou a uma falha nossa na retomada do processo de execução dos testes levou à diferença vista nos resultados dos testes com 100 processos da máquina A (4 cores) e os da B (8 cores).**