EP1 MAC0422 2017

Guilherme Costa Vieira Nº USP: 9790930
Victor Chiaradia Gramuglia Araujo Nº USP: 9793756

Parte 1 - Shell

Estrutura do shell:

- Loop infinito
  - Pega o diretório atual com a syscall getcwd().
  - Pega o comando do usuário.
  - Verifica se o comando pedido é embutido.
  - Se embutido:
    - Chama as funções que implementam date ou chown.
  - Se não embutido:
    - Chama a syscall fork() para criar um processo filho do shell.
    - Chama a syscall execvp() para trocar o processo filho pelo processo do comando dado.
    - Chama a syscall waitpid() para o shell esperar o processo filho terminar.

Parte 1 - Shell

Comandos embutidos.

- Função embdDate():
  - Chama a syscall time() para pegar segundos desde 1 Jan de 1970.
  - Chama ctime() e localtime() definidas na time.h
    - Utiliza-se essas funções para formatar no padrão desejável de visualização.
- Função embdChown():
  - Recebe o comando dado pelo usuário.
  - Chama a syscall getpwnam() para pegar os dados do usuário que executou.
  - Chama a syscall getgrnam() para pegar os dados do grupo.
  - Chama a syscall chown() para mudar o grupo do arquivo desejado.

Parte 2 - Simulador de Processos

- Implementação monocore dos escalonadores de processos.

- Defini-se a struct Process para guardar as informações dos processos.

```
typedef struct {
        int line;
        double t0, dt, deadline, run_time;
        char *name;
        pthread_mutex_t mutex;
        pthread_t thread;
        } Process;
```

Parte 2 - Simulador de Processos

- Os processos são guardados em listas ligadas.
  - Lista to_arrive: processos que não chegaram no sistema.
  - Lista to_schedule: processos prontos para serem escalonados.

```
typedef struct node {
                    Process *info;
                    struct node *next;
                    } Cell;

typedef Cell *List;
```

Parte 2 - Simulador de Processos

- Principais funções compartilhadas pelos escalonadores:
  - List readFile(char *file_name);
  - List add(List to_schedule, List *to_arrive, double time, int optional);
  - void writeFile(char *output_file, Process *proc, double time, int optional);
  - void nap(double dt);

Parte 2 - Simulador de Processos

- Shortest Job First (SJF):
  - Loop que roda até as listas to_schedule e to_arrive estiverem vazias.
    - Verifica se processos novos chegaram com a função add()
    - Percorre a lista to_schedule para procurar o processo com menor tempo de execução.
    - Cria uma thread com pthread_create().
    - Utiliza-se pthread_join() para rodar o processo até o final de sua execução e voltar para o SJF.

Parte 2 - Simulador de Processos

- Round Robin e Escalonamento com Prioridade:
  - Utilizam a mesma função devido a sua semelhança.
  - A diferença está na função chamada para simular o processo.
  - Loop que roda até as listas to_schedule e to_arrive estiverem vazias.
    - Verifica se processos novos chegaram com a função add()
    - Percorre a lista to_schedule de forma circular para escolher um processo para ser executado com o apontador currProcess.
    - Se o processo vai ser executado pela primeira vez (run_time == 0), utiliza-se phthread_create().
      - Na função da simulação, o processo roda por um QUANTUM, no Round Robin por exemplo, e trava seu semáforo.
    - Se não, o semáforo do processo é liberado.

Parte 2 - Simulador de Processos

- Round Robin e Escalonamento com Prioridade (cont):
  - O semáforo do escalonador é travado.
  - O semáforo do escalonador é destravado pela função de simulação.

  Utiliza-se uma variável temporária para executar os dois próximos passos.

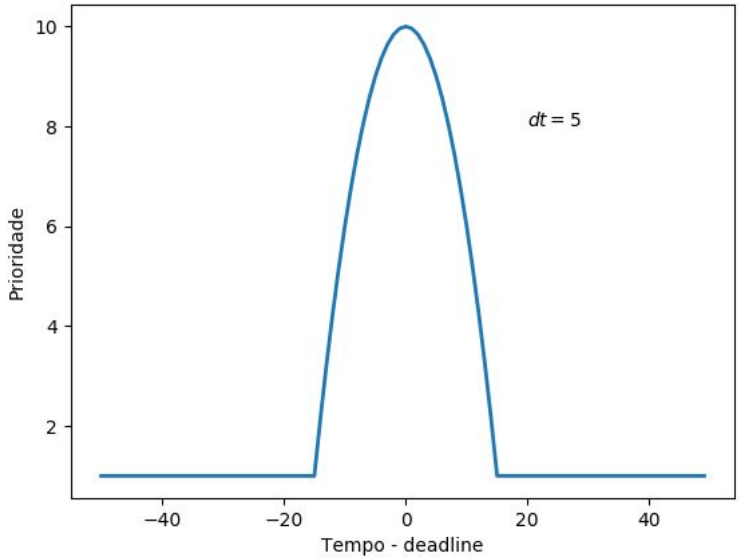
  - Se o processo que acabou de ser executado acabou, ele é removido da lista to_schedule.
  - O apontador currProcess é atualizado.

Parte 2 - Simulador de Processos

- Funções de simulação para Round Robin e Escalonamento com Prioridade :
  - Na função do Round Robin, o processo é simulado por um QUANTUM.
  - Já no Escalonamento com prioridade, o processo é simulado por prioridade*QUANTUM.
- A prioridade do processo é calculada logo antes dele rodar (dinamicamente).
- Quanto mais perto da deadline, mais prioridade um processo tem. Ao ultrapassar sua deadline, um processo perde prioridade até ela se tornar constante.
- A ideia é tentar cumprir a deadline conforme ela se aproxima e não atrapalhar os outros processos depois que a deadline já passou.

Parte 2 - Simulador de Processos

- Gráfico da função que utilizamos.

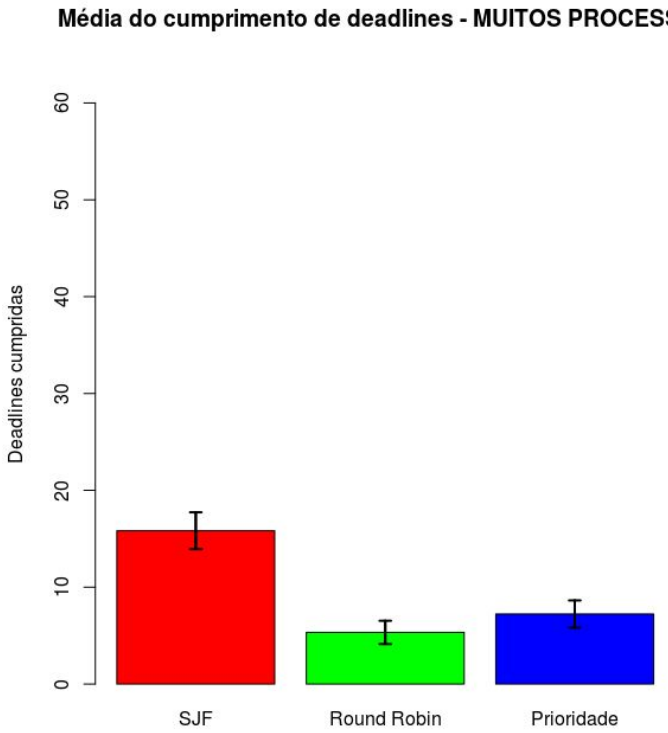


Experimentos

Como foram feitos:

- Distribuição uniforme de probabilidades para gerar inputs.
- Número de processos:
  - 8 para poucos processos.
  - 32 para médios processos.
  - 64 para muitos processos.
- Máximo t0 e máximo deadline utilizados na geração de inputs são proporcionais ao número de processos.
- Máquinas testadas:
  - A: Intel(R) Celeron(R) CPU N3160 @ 1.60GHz QUAD CORE.
  - B: Intel(R) Celeron(R) CPU N3010 @ 1.04GHz DUAL CORE.

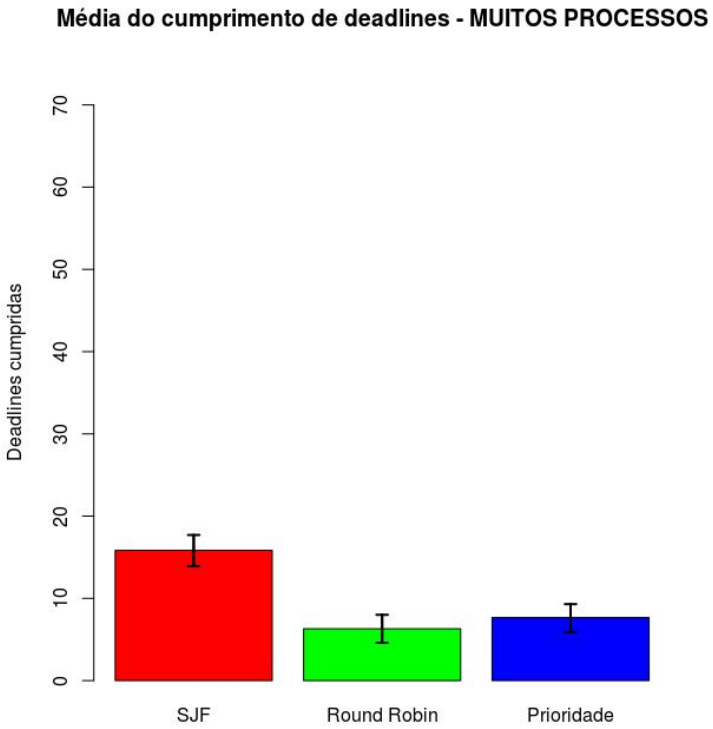
Máquina A - Celeron 4 núcleos



Confiança de 95%

- Média SJF = 15.8 (24.7%)
  - IC = (13.9, 17.7)
- Média Round Robin = 5.3 (8.3%)
  - IC = (4.1, 6.5)
- Média Prioridade = 7.2 (11,25%)
  - IC = (5.8, 8.6)

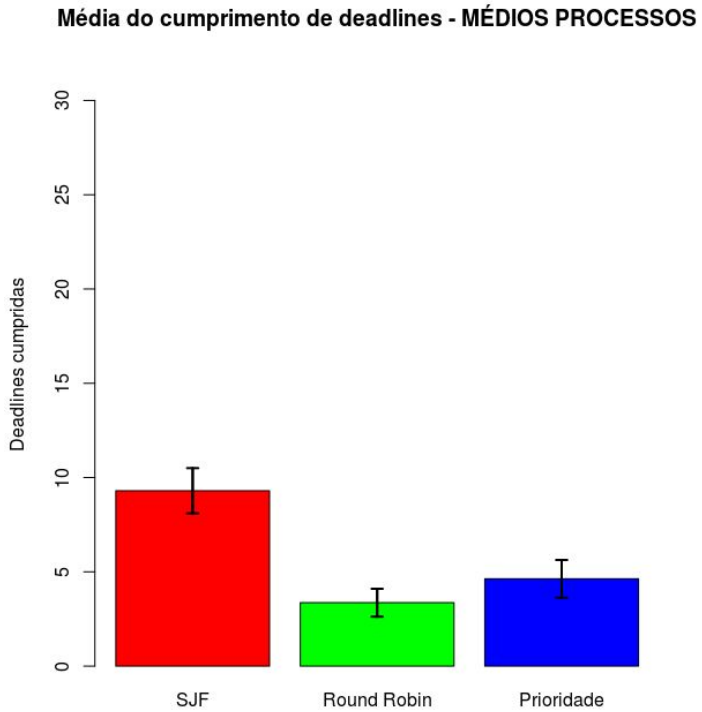
Máquina B - Celeron 2 núcleos



Confiança de 95%

- Média SJF = 15.8 (24.6%)
  - IC = (13.9, 17.7)
- Média Round Robin = 6.3 (9.8%)
  - IC = (4.6, 8)
- Média Prioridade = 7.6 (11.9%)
  - IC = (5.9, 9.3)

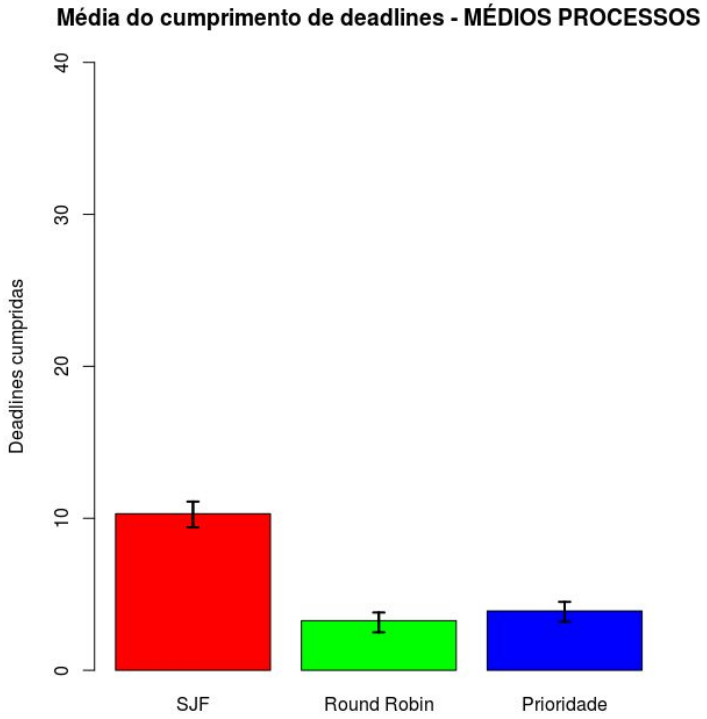
Máquina A - Celeron 4 núcleos



Confiança de 95%

- Média SJF = 9.3 (29%)
  - IC = (8.1, 10.5)
- Média Round Robin = 3.3 (10.3%)
  - IC = (2.6, 4)
- Média Prioridade = 4.6 (14.4%)
  - IC = (3.6, 5.6)

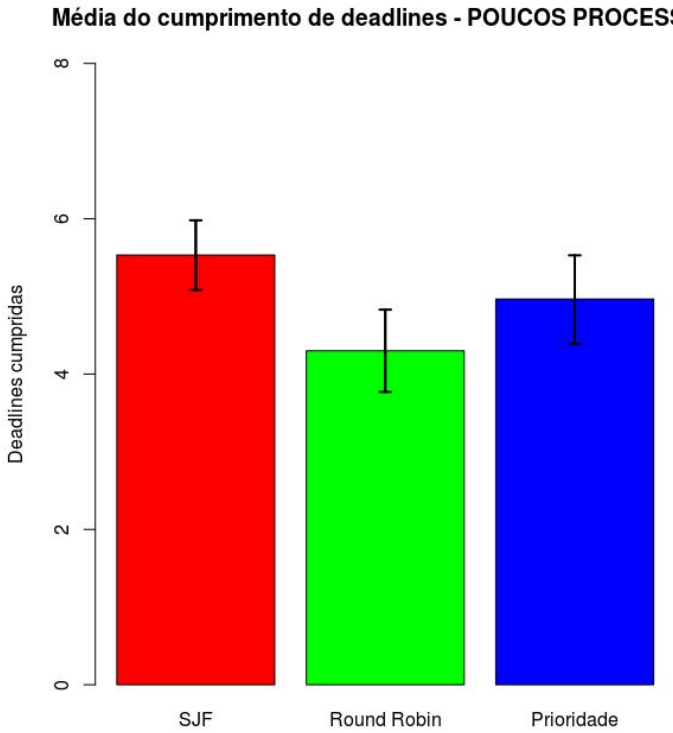
Máquina B - Celeron 2 núcleos



Confiança de 95%

- Média SJF = 10.3 (32.2%)
  - IC = (9.5, 11.1)
- Média Round Robin = 3.2 (10%)
  - IC = (2.6, 3.8)
- Média Prioridade = 3.9 (12.2%)
  - IC = (3.3, 4.5)

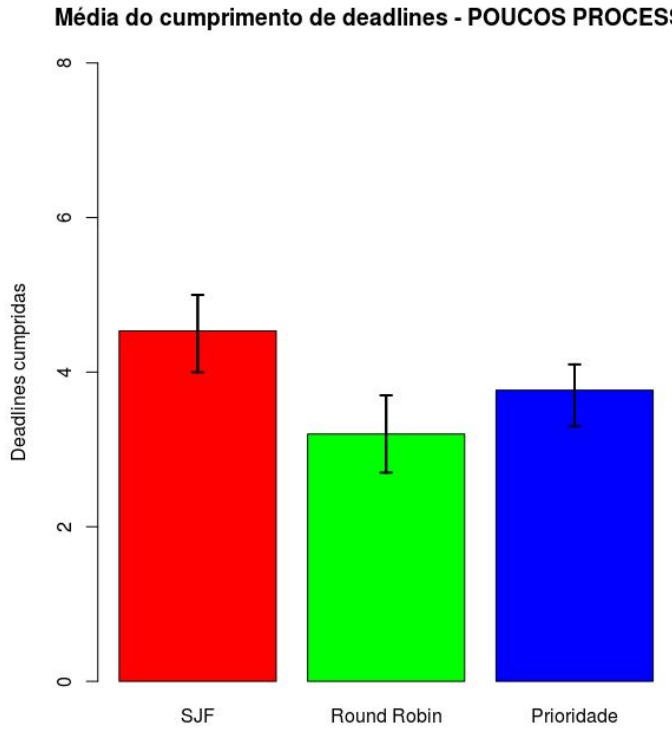
Máquina A - Celeron 4 núcleos



Confiança de 95%

- Média SJF = 5.5 (69%)
  - IC = (5, 6)
- Média Round Robin = 4.3 (54%)
  - IC = (3.8, 4.8)
- Média Prioridade = 4.9 (61.2%)
  - IC = (4.3, 5.5)

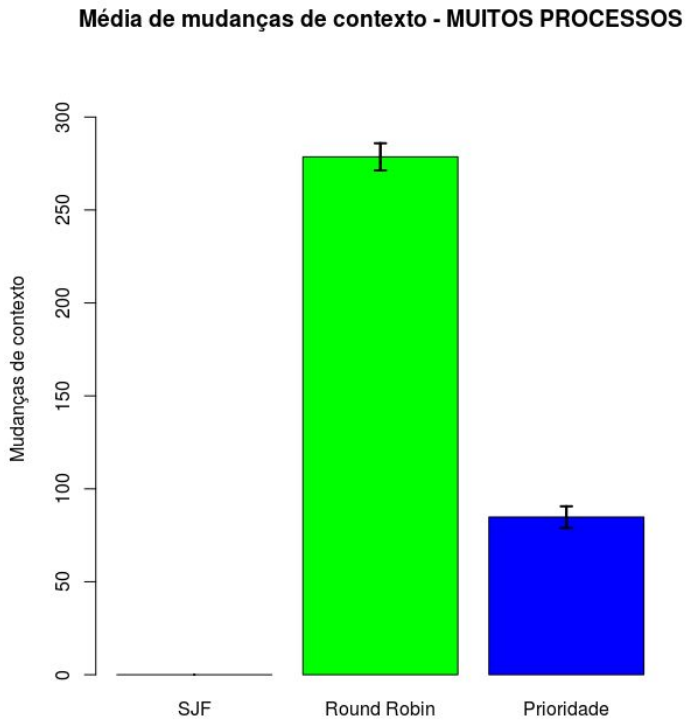
Máquina B - Celeron 2 núcleos



Confiança de 95%

- Média SJF = 4.5 (56.2%%)
  - IC = (4, 5)
- Média Round Robin = 3.2 (40%)
  - IC = (2.7, 3.7)
- Média Prioridade = 3.7 (46.2%)
  - IC = (3.3, 4.1)

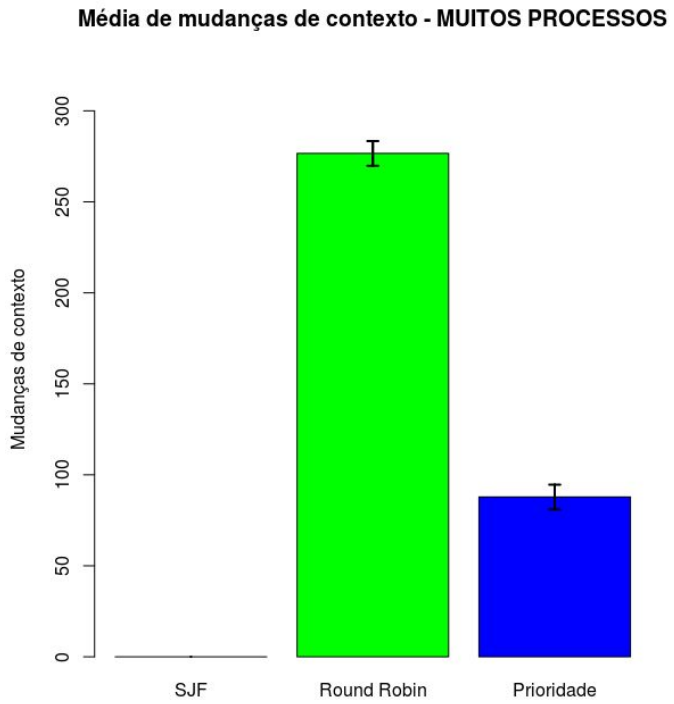
Máquina A - Celeron 4 núcleos



Confiança de 95%

- Média SJF = 0
- Média Round Robin = 278.6
  - IC = (271.3, 285.9)
- Média Prioridade = 84.8
  - IC = (79, 90.6)

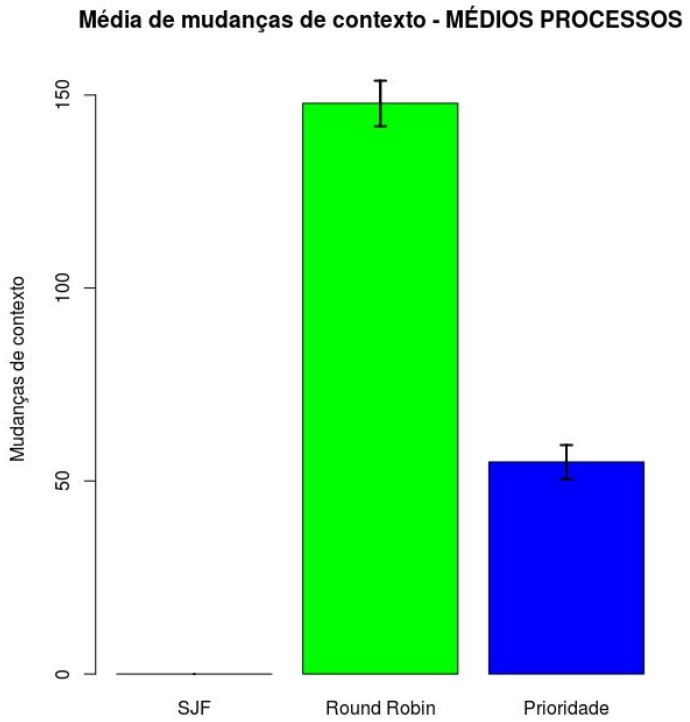
Máquina B - Celeron 2 núcleos



Confiança de 95%

- Média SJF = 0
- Média Round Robin = 276.6
  - IC = (269.8, 283.4)
- Média Prioridade = 87.8
  - IC = (81, 94.6)

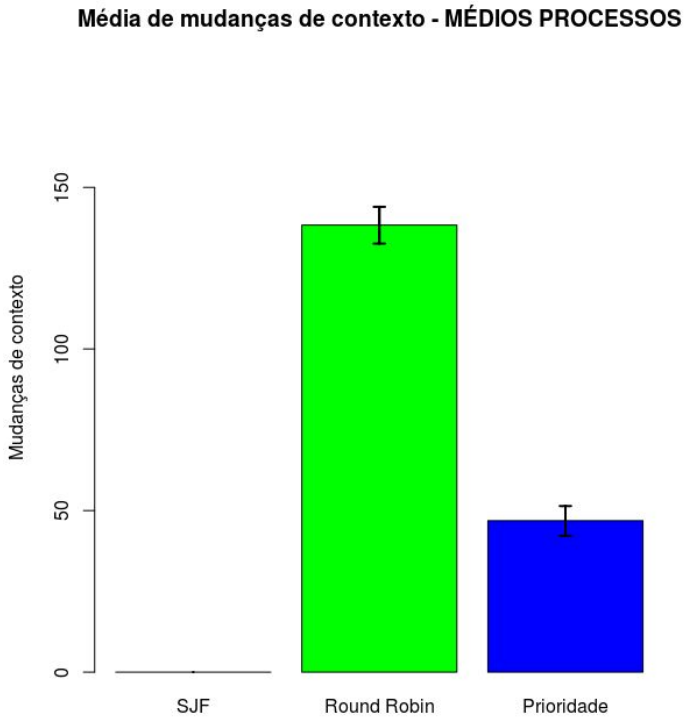
Máquina A - Celeron 4 núcleos



Confiança de 95%

- Média SJF = 0
- Média Round Robin = 147.8
  - IC = (141.9, 153.7)
- Média Prioridade = 54.9
  - IC = (50.5, 59.3)

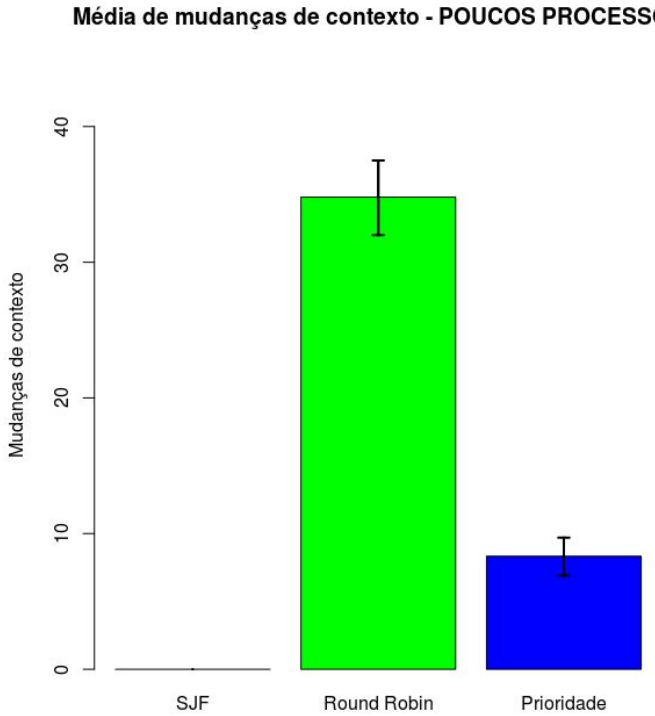
Máquina B- Celeron 2 núcleos



Confiança de 95%

- Média SJF = 0
- Média Round Robin = 138.3
  - IC = (132.6, 144)
- Média Prioridade = 46.8
  - IC = (42.2, 51.4)

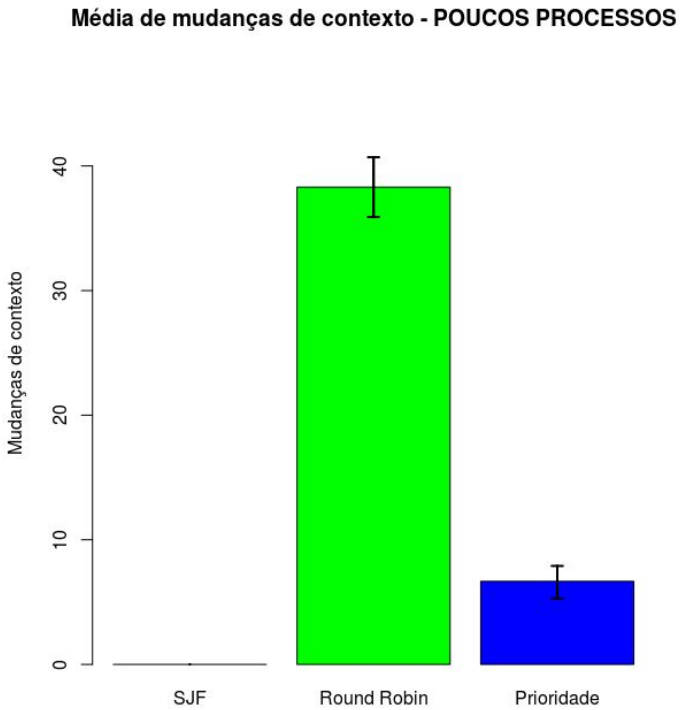
Máquina A - Celeron 4 núcleos



Confiança de 95%

- Média SJF = 0
- Média Round Robin = 34.8
  - IC = (32.1, 37.5)
- Média Prioridade = 8.3
  - IC = (6.9, 9.7)

Máquina B - Celeron 2 núcleos



Confiança de 95%

- Média SJF = 0
- Média Round Robin = 38.3
  - IC = (35.9, 40.7)
- Média Prioridade = 6.6
  - IC = (5.3, 7.9)