MAC 422
Sistemas Operacionais

Gabriel Fernandes de Oliveira
Thiago Estrela Montenegro

shell ep1sh

- Biblioteca *readline* para manter o histórico dos comandos
- *getcwd* para o layout de terminal com o diretório atual
- Binários **ping, cal**, e **ep1** implementados como uma chamada de *fork* e *execve*
- Comando **chown** como o comando homônimo *chown* da biblioteca *<unistd.h>*
- **date** como chamada da função *localtime* e *asctime*, ambas de *<time.h>*

Simulador de Processos

Estruturas de Dados e bibliotecas auxiliares implementadas:

- Estruturas genéricas
  - Heap
  - Queue
- Estrutura de Processos com *thread* e *mutex* própria
- Bibliotecas de controle de tempo (*calctime.h)* e de saída (*print.h)*

```c
#include <stdlib.h>
#include <sys/time.h>
#include "calctime.h"

double sec(timev t){
        return t.tv_sec + t.tv_usec/1000000.;
}

double running_time(){
        timev act;
        gettimeofday(&act, NULL);
        return sec(act) - sec(start_time);
}
```

Programa *calctime.c*

Simulador de Processos

- Quantidade de núcleos do computador é parâmetro dos escalonadores
- Simulação abstrata do uso das CPU's
- Nos escalonadores com preempção, há uma constante renovação de threads e mutex para os processos colocados em espera

```
int ncores = sysconf(_SC_NPROCESSORS_ONLN);
switch (type) {
  case 1:
    SJF(trace, output, ncores);
    break;
  case 2:
    RR(trace, output, ncores);
    break;
  case 3:
    P(trace, output, ncores);
    break;
  default:
    fprintf(stderr, "Escalonador invalido\n");
    break;
}
```

Linhas 29 - 36 de *ep1.c*

Testes

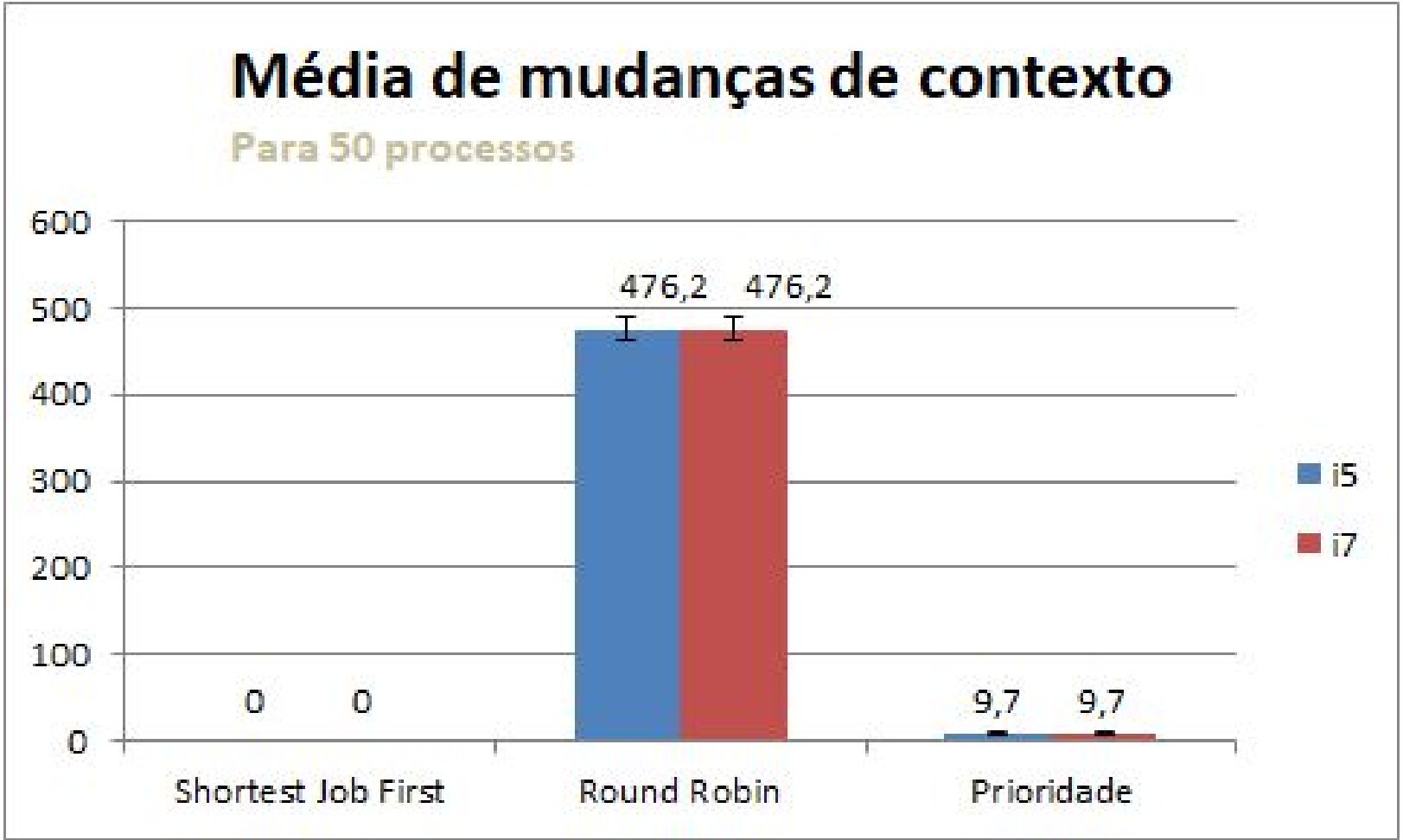
- Testes realizados em máquinas com processadores i5-3210M e i7-4770

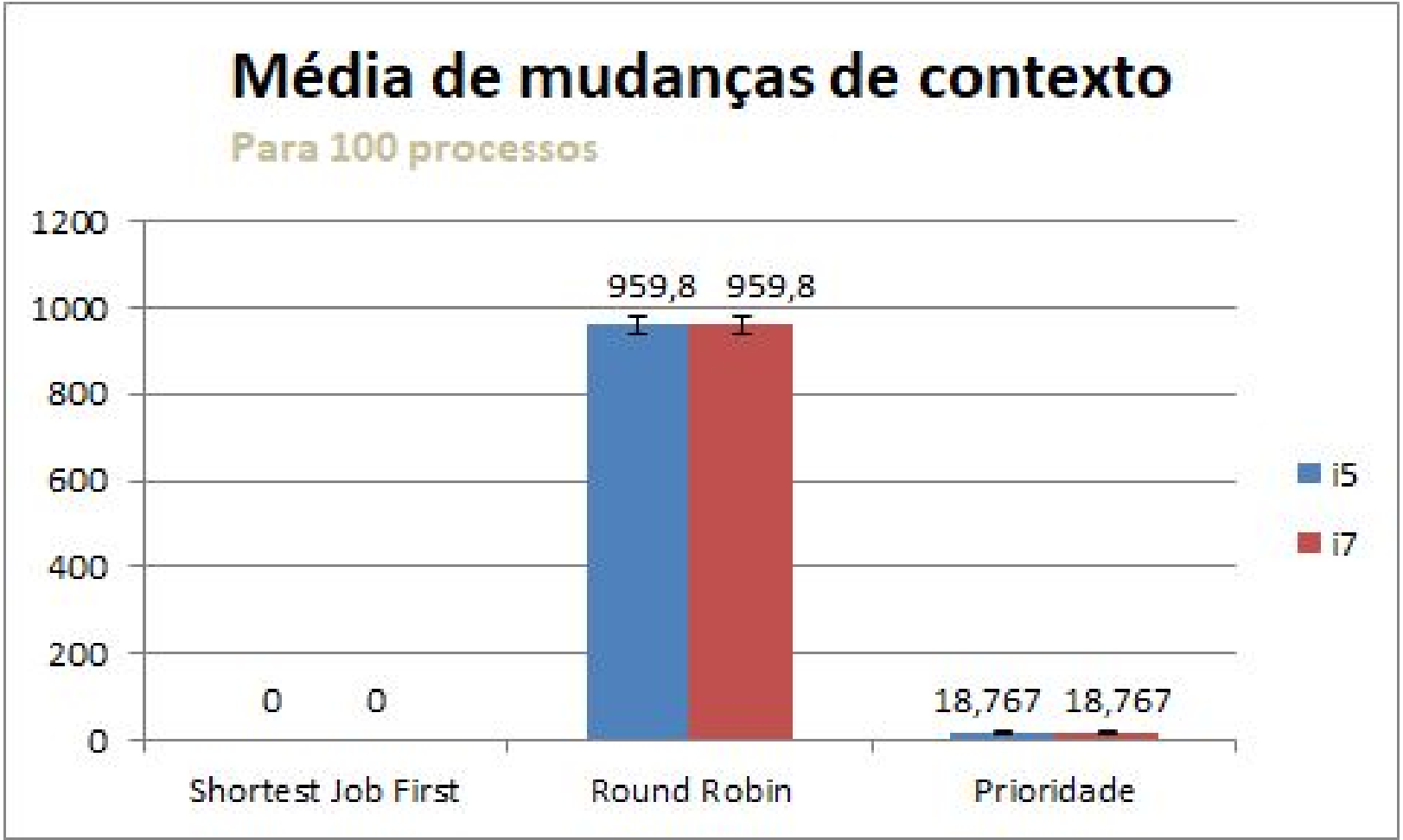
- Scripts para gerar testes aleatorizados com os seguintes parâmetros
  - *t0* entre 0,0 e 20,0 segundos
  - *dt* entre 0,1 e 2,0 segundos
  - *deadline* entre 1,0 e 6,0 *dt*'s após *t0*

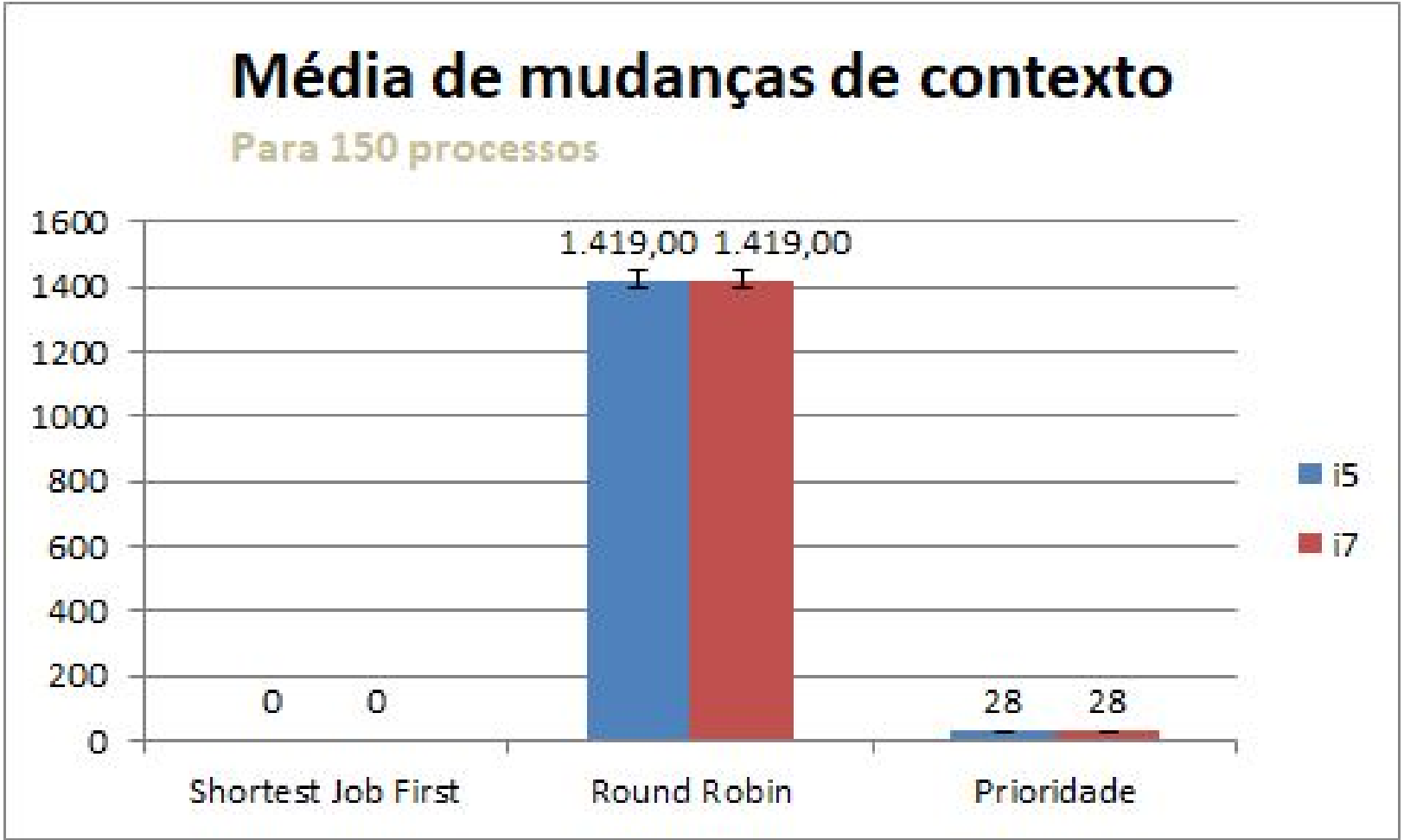
- Testes em baterias de 30 variando-se o número de processos por arquivo

```
12.3 0.6 15.2 process0
18.2 1.6 21.9 process1
9.0 0.3 10.1 process2
19.1 1.0 20.6 process3
10.0 0.9 13.6 process4
7.5 0.2 8.5 process5
15.8 1.5 22.7 process6
9.7 0.9 13.8 process7
16.1 0.6 16.9 process8
7.2 0.7 11.1 process9
```

Teste com 10 processos (*small/in0.txt*)


Média de mudanças de contexto
Para 50 processos
600
500
400
300
200
100
0
476,2
476,2
0
0
9,7
9,7
Shortest Job First
Round Robin
Prioridade
i5
i7


Média de mudanças de contexto
Para 100 processos
1200
1000
800
600
400
200
0
959,8
959,8
0
0
18,767
18,767
Shortest Job First
Round Robin
Prioridade
i5
i7

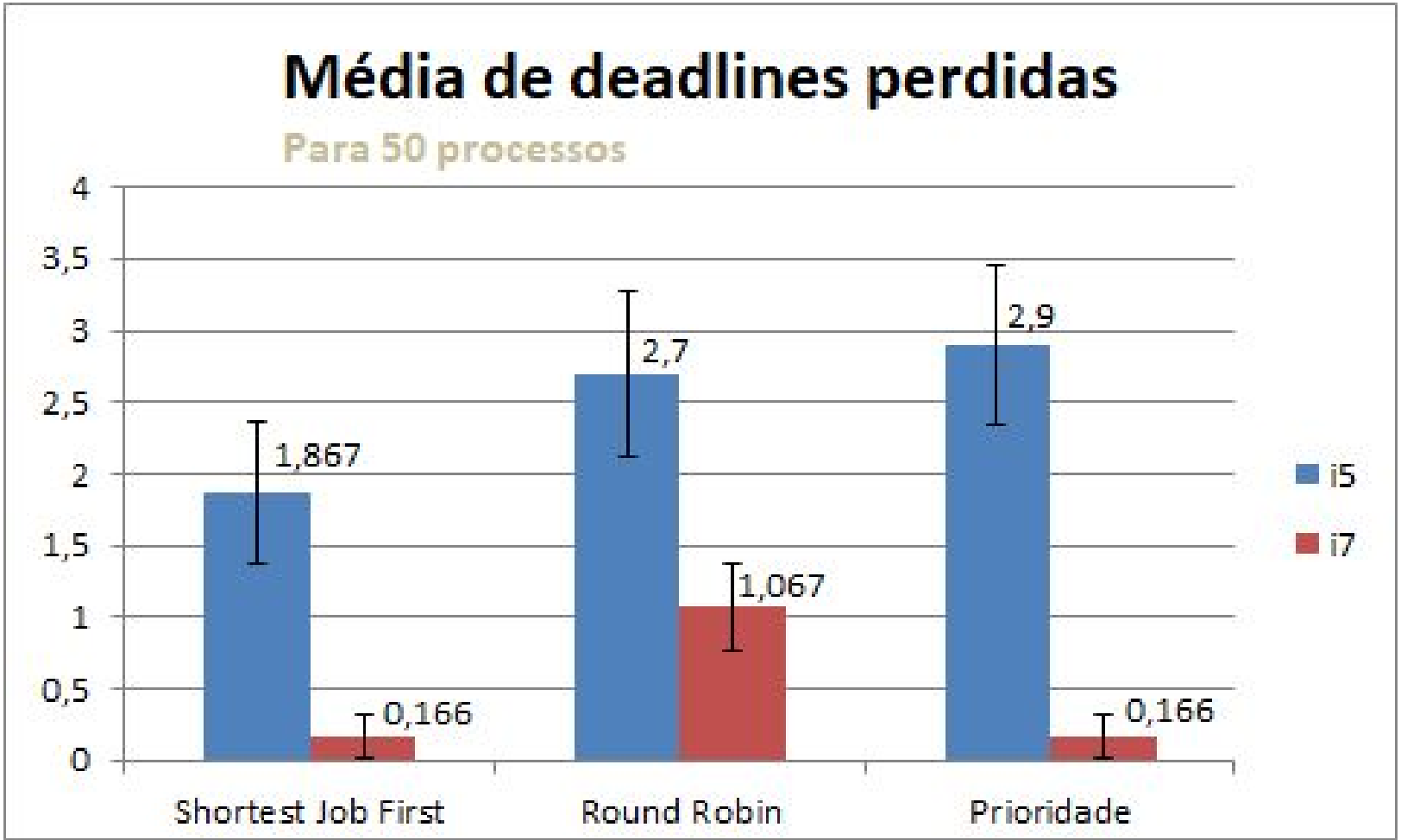

Média de mudanças de contexto
Para 150 processos
1600
1400
1200
1000
800
600
400
200
0
1.419,00
1.419,00
0
0
28
28
Shortest Job First
Round Robin
Prioridade
i5
i7

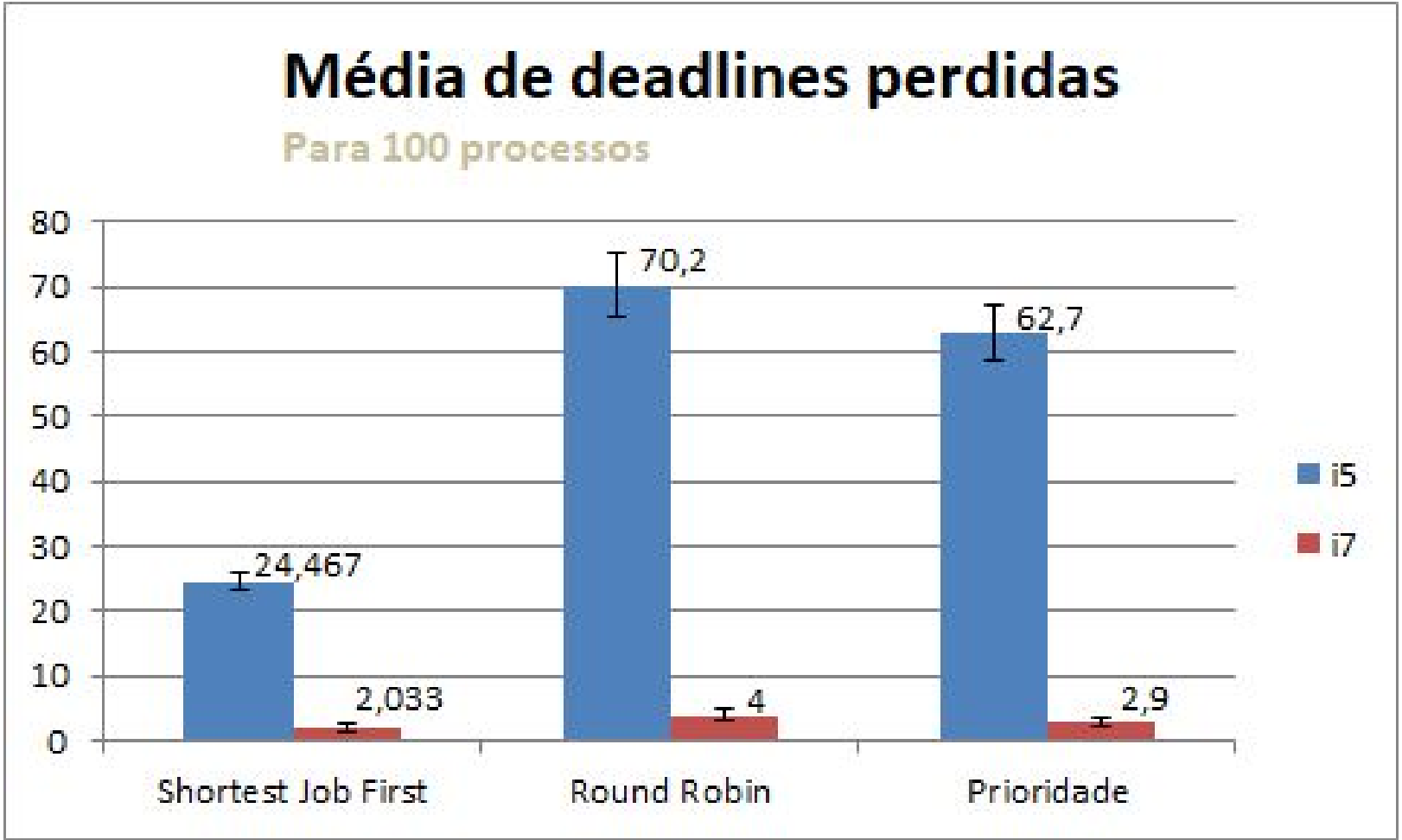
Resultados

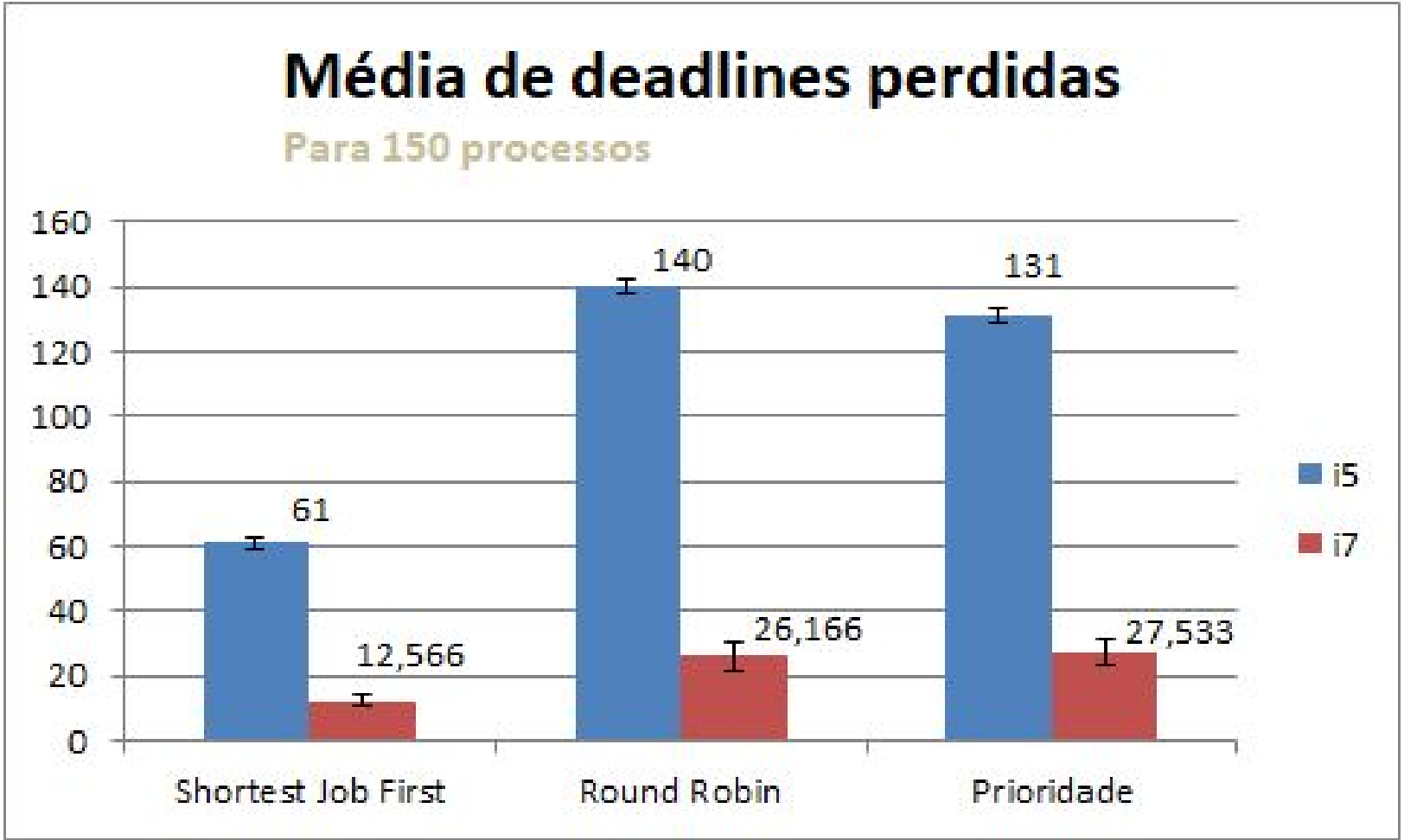
- Sem mudanças de contexto no **Shortest Job First**, como era de se esperar

- **Round Robin** com aproximadamente **9,6** mudanças de contexto por processo
  - Quantum de 10ms estabelecido
  - Valor esperado de dt para cada processo de 100ms

- Escalonamento por **Prioridade** apresentando uma média de apenas **0,19** mudanças de contexto por processo
  - Grande contraste com o Round Robin, economizando tempo de escalonamento


Média de deadlines perdidas
Para 50 processos
4
3,5
3
2,5
2
1,5
1
0,5
0
1,867
0,166
2,7
1,067
2,9
0,166
Shortest Job First
Round Robin
Prioridade
i5
i7


Média de deadlines perdidas
Para 100 processos
80
70
60
50
40
30
20
10
0
24,467
2,033
70,2
4
62,7
2,9
Shortest Job First
Round Robin
Prioridade
i5
i7


Média de deadlines perdidas
Para 150 processos
160
140
120
100
80
60
40
20
0
61
12,566
140
26,166
131
27,533
Shortest Job First
Round Robin
Prioridade
i5
i7

Resultados

- **Shortest Job First** garante o menor número de deadlines perdidas
- **Round Robin** lidera na quantidade de deadlines perdidas
- Escalonamento por **Prioridade** se assemelha muito ao Round Robin nos menores casos, porém tem uma performance estatisticamente melhor no caso de 150 processos

Conclusão

Shortest Job First

- Menor perda de deadlines
- Não há perda de tempo com mudanças de contexto
- Pode causar Starvation
- Menor Interatividade

Conclusão

Round Robin

- Maior interatividade entre os escalonadores implementados
- Maiores quantidades de mudança de contexto e perda de deadlines

Conclusão

Escalonamento por Prioridade

- Taxas de mudança de contexto muito inferiores às do *Round Robin*
- Minimiza o tempo que os processos passam com seu deadline estourado
- Perda de deadline alta, ainda que ligeiramente menor ao do *Round Robin*