Escalonadores

SJF, SRTF, Prioridades e RR

Terabytes

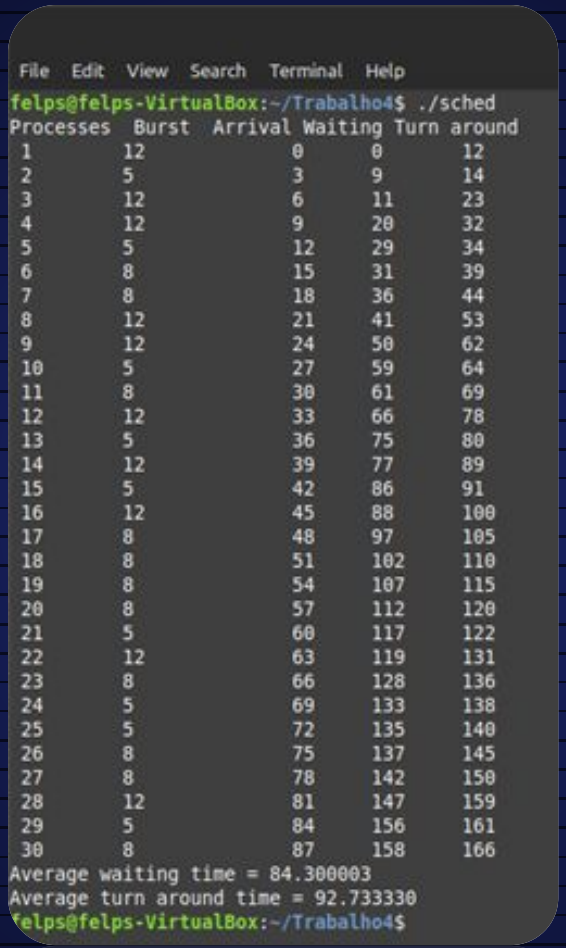
Escalonador

- Controla o acesso dos processos à CPU;
- O que causa essa necessidade:
  - Alta demanda da CPU;
  - Possibilidade de priorizar processos;
  - Necessário coordenar processos em sistemas multitarefas;
- Algoritmos explorados nesse trabalho:
  - Shortest Job First (SJF)
  - Shortest Remaining Time First (SRTF)
  - Round Robin (RR)
  - Prioridades(fixa)

Aumentando o numero de processo

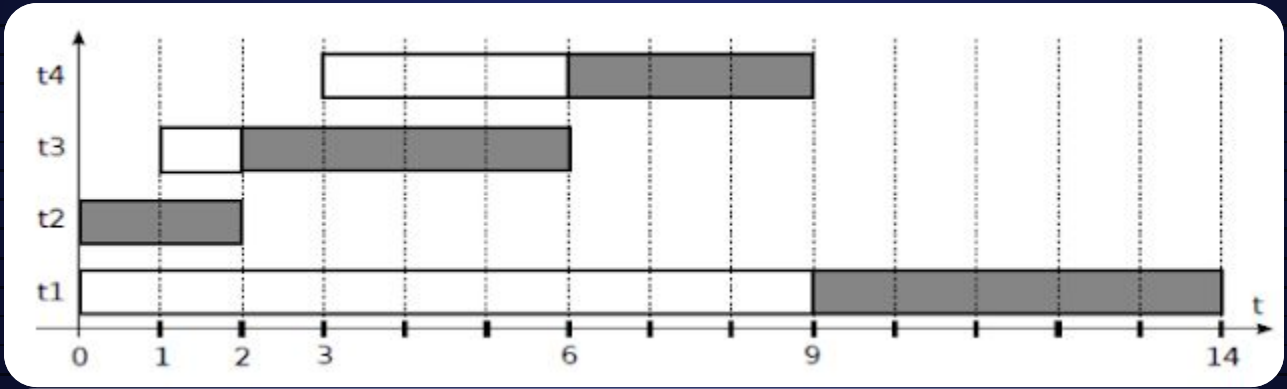
- Biblioteca para gerar números aleatórios: time.h
- Função rand() gera números de 0 a 32.767
- Burst time é selecionado aleatoriamente entre 5,8 e 12.

```c
void gera_processos(int proc[], int burst_time[],int arrival_time[], int n) {
    // Inicializa o gerador de numeros aleatorios com o tempo atual
    srand(time(NULL));
    int times[] = {5, 8, 12};
    for (int i = 0; i < n; i++) {
        proc[i] = i + 1;
        burst_time[i] = times[rand() % 3];  // Burst time 5,8 ou 12
        arrival_time[i] = i*3; // Gera arrival time com multiplos de 3
    }
}
```


File Edit View Search Terminal Help
felps@felps-VirtualBox:~/Trabalho4$ ./sched
Processes Burst Arrival Waiting Turn around
1 12 0 0 12
2 5 3 9 14
3 12 6 11 23
4 12 9 20 32
5 5 12 29 34
6 8 15 31 39
7 8 18 36 44
8 12 21 41 53
9 12 24 50 62
10 5 27 59 64
11 8 30 61 69
12 12 33 66 78
13 5 36 75 80
14 12 39 77 89
15 5 42 86 91
16 12 45 88 100
17 8 48 97 105
18 8 51 102 110
19 8 54 107 115
20 8 57 112 120
21 5 60 117 122
22 12 63 119 131
23 8 66 128 136
24 5 69 133 138
25 5 72 135 140
26 8 75 137 145
27 8 78 142 150
28 12 81 147 159
29 5 84 156 161
30 8 87 158 166
Average waiting time = 84.300003
Average turn around time = 92.733330
felps@felps-VirtualBox:~/Trabalho4$

Shortest Job Frist

- Dá prioridade aos processos de menor tempo de execução.



SJF

- Implementado dentro da função avgtime.
- Primeiro laço percorre cada espaço da array.
- Segundo laço procura a partir desse índice o menor burst time.
- Escalona com base no burst e no tempo de chegada.

```
int current_time = 0;
for(i = 0; i < n; i++) {
    minId = -1;
    min_burst = __INT_MAX__;
    for(j = i; j < n; j++) {
        if(burst_time[j] < min_burst  && arrival_time[j] <= current_time){
            minId = j;
            min_burst = burst_time[j];
        }
    }
```

SJF

- Após cada iteração do primeiro laço:
  - Se nenhum processo estiver disponível, prossegue no tempo e continua procurando para o índice atual.
  - Se um processo foi selecionado, troca os processos na array.
- Após todas as iterações é feito o calculo do tempo médio de espera e execução

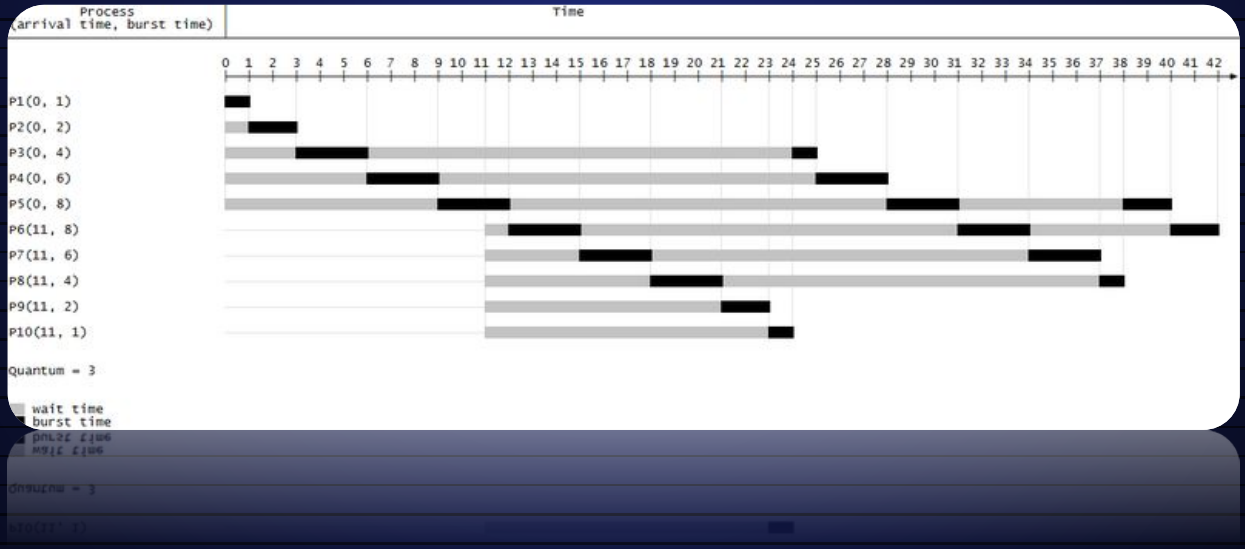
```
//troca os burstTime
if(minId == -1){
current_time++;
i--;
}else{if(i != minId){
    int temp = burst_time[i];
    burst_time[i] = burst_time[minId];
    burst_time[minId] = temp;
    // troca os proc
    temp = proc[i];
    proc[i] = proc[minId];
    proc[minId] = temp;
    // arrivaltime
    temp = arrival_time[i];
    arrival_time[i] = arrival_time[minId];
    arrival_time[minId] = temp;
     }
   current_time += burst_time[i];
}
```

SJF
Teste

```
File  Edit  View  Search  Terminal  Help
felps@felps-VirtualBox:~/Trabalho4$ ./SJF
Processes  Burst  Arrival   Waiting Turn around
 1         8              0       0       8
 2         12             3       5       17
 5         5              12      8       13
 6         5              15      10      15
 10        5              27      3       8
 11        5              30      5       10
 7         8              18      22      30
 17        5              48      0       5
 12        8              33      20      28
 19        5              54      7       12
 20        5              57      9       14
 24        5              69      2       7
 25        5              72      4       9
 27        5              78      3       8
 29        5              84      2       7
 16        8              45      46      54
 8         8              21      78      86
 18        8              51      56      64
 21        8              60      55      63
 22        8              63      60      68
 23        8              66      65      73
 13        8              36      103     111
 26        8              75      72      80
 14        8              39      116     124
 4         12             9       154     166
 3         12             6       169     181
 9         12             24      163     175
 28        12             81      118     130
 15        12             42      169     181
 30        12             87      136     148
Average waiting time = 55.333332
Average turn around time = 63.166668
felps@felps-VirtualBox:~/Trabalho4$
```

Shortest Remaining Time Frist

- Prioriza processos com menor tempo restante.
- Pode parar a execução de um processo para dar prioridade a outro.(Troca de contexto)



SRTF

- Utiliza duas funções, findMenorProc e SRTF.
- A primeira passa por todos os processos e encontra qual tem o menor tempo restante

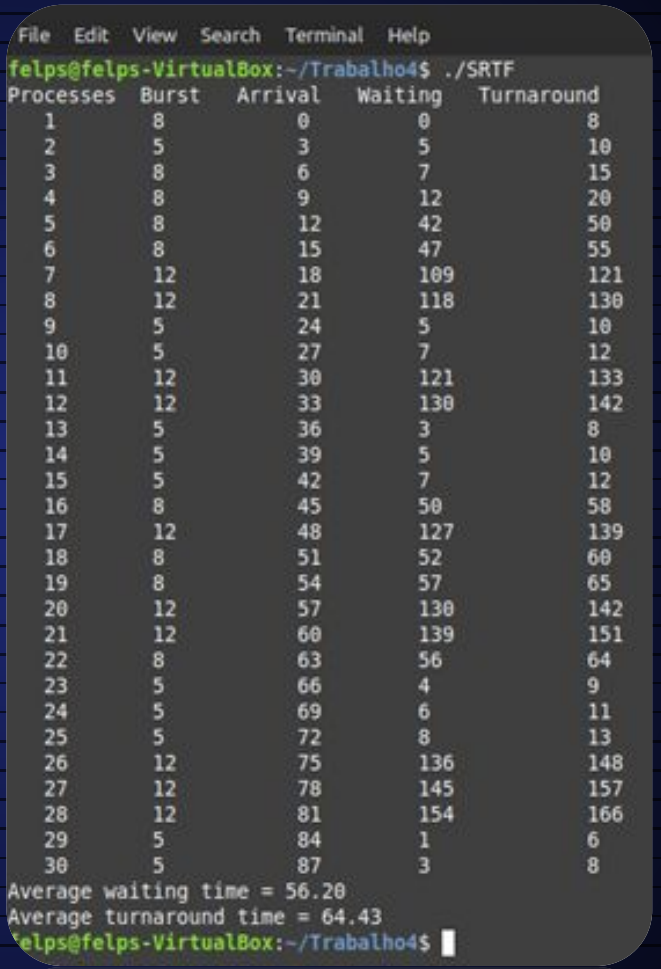
```c
int findMenorProc(int arrival_time[], int n, int remaining_time[], int current_time) {
    int min_time = __INT_MAX__;
    int menorProc = -1;
    for (int i = 0; i < n; i++) {
        if (remaining_time[i] > 0 && remaining_time[i] < min_time && arrival_time[i] <= current_time) {
            min_time = remaining_time[i];
            menorProc = i;
        }
    }
    return menorProc;
}
```

SRTF

- Na função SRTF
- Usamos uma cópia do vetor de burst time.
- Criamos um laço que ocorre enquanto houver processos não concluídos.
- Em cada iteração reduzimos o tempo do processo selecionado pela função anterior.

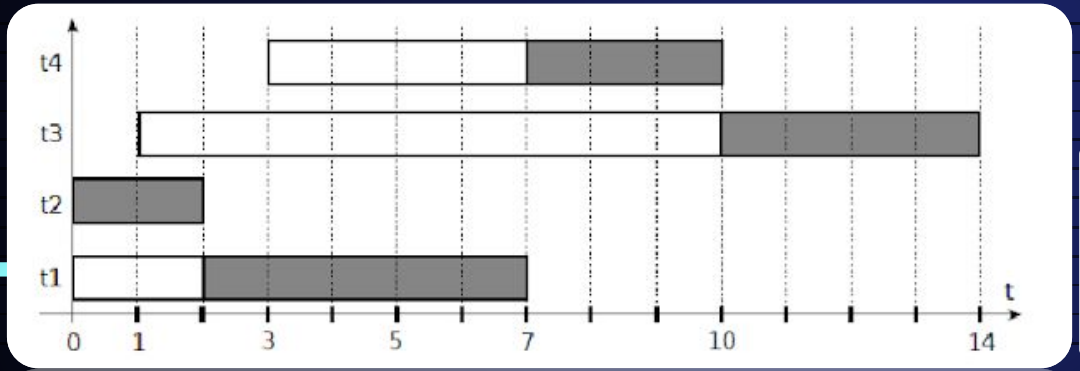
```
while (completed < n) {
    menorProc = findMenorProc(arrival_time, n, remaining_time, current_time);
    if (menorProc == -1) {
        current_time++;
    } else {
        remaining_time[menorProc]--;
        if (remaining_time[menorProc] == 0) {
            completed++;
            int finish_time = current_time + 1;
            turnaround_time[menorProc] = finish_time - arrival_time[menorProc];
            wait_time[menorProc] = turnaround_time[menorProc] - burst_time[menorProc];
            total_wait_time += wait_time[menorProc];
            total_turnaround_time += turnaround_time[menorProc];
        }

        current_time++;
    }
}
```

SRTF
Teste


File Edit View Search Terminal Help
felps@felps-VirtualBox:~/Trabalho4$ ./SRTF
Processes Burst Arrival Waiting Turnaround
1 8 0 0 8
2 5 3 5 10
3 8 6 7 15
4 8 9 12 20
5 8 12 42 50
6 8 15 47 55
7 12 18 109 121
8 12 21 118 130
9 5 24 5 10
10 5 27 7 12
11 12 30 121 133
12 12 33 130 142
13 5 36 3 8
14 5 39 5 10
15 5 42 7 12
16 8 45 50 58
17 12 48 127 139
18 8 51 52 60
19 8 54 57 65
20 12 57 130 142
21 12 60 139 151
22 8 63 56 64
23 5 66 4 9
24 5 69 6 11
25 5 72 8 13
26 12 75 136 148
27 12 78 145 157
28 12 81 154 166
29 5 84 1 6
30 5 87 3 8
Average waiting time = 56.20
Average turnaround time = 64.43
felps@felps-VirtualBox:~/Trabalho4$

Prioridades (fixa)

- Atribui valores de prioridade para os processos;
- Executa em ordem de prioridade;



Tarefa	t_1	t_2	t_3	t_4
Ingresso	0	0	1	3
Duração	5	2	4	3
Prioridade	2	3	1	4

Prioridades (fixa)

- Adicionado a geração aleatória de prioridades de 1 a 5;
- Implementado dentro de avgtime;
- Primeiro laço percorre cada espaço da array;
- Segundo laço procura a partir desse índice o processo de maior prioridade disponível;

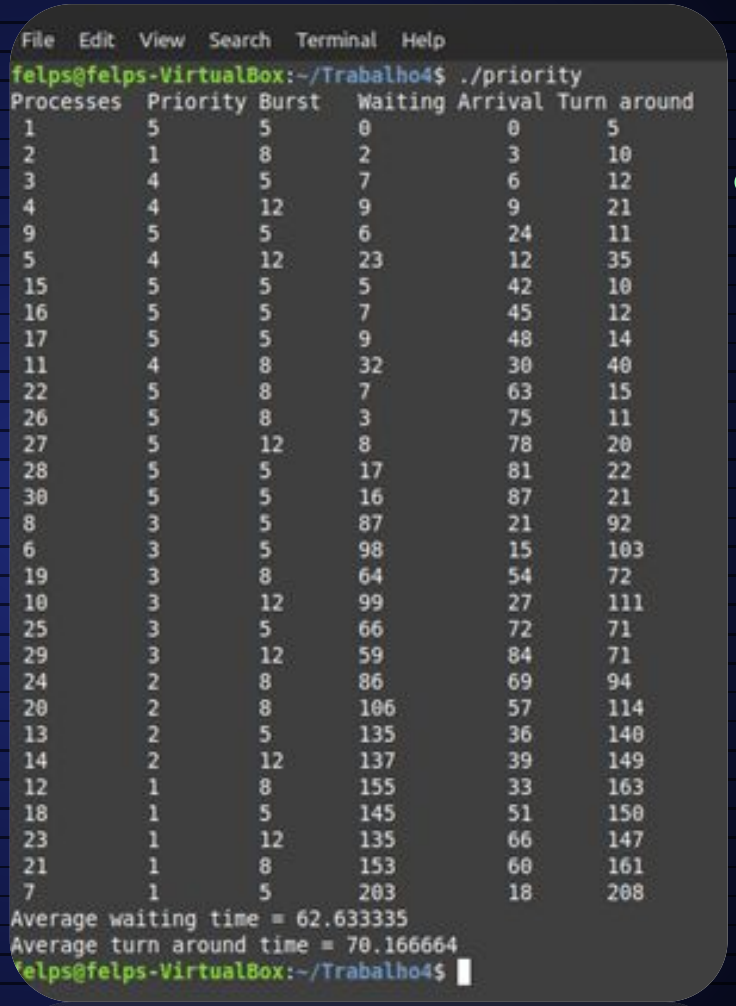
```
for (i = 0; i < n; i++) {
 idMax = -1;
 maior_p = 0;
 for (j = i; j < n; j++) {
     if (arrival_time[j] <= current_time && priority[j] > maior_p  ){
         idMax = j;
         maior_p = priority[j];
     }
 }
```

Prioridades (fixa)

- Para cada iteração do primeiro laço:
  - Se nenhum processo estiver disponível, prossegue no tempo e continua procurando para o índice atual.
  - Se um processo foi selecionado, troca os processos nos vetores.

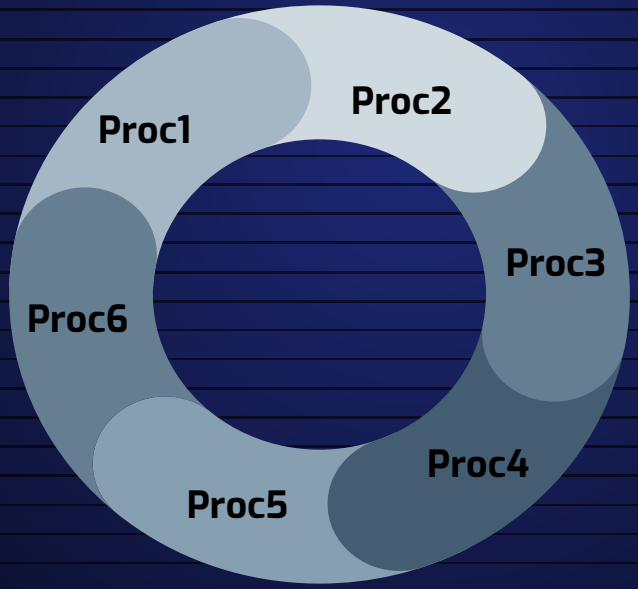
```
    if (idMax == -1) // se nenhum processo estiver pronto
       {current_time++;
       i--;}
    else{if (idMax != i) {//faz as trocas
          int temp;

          // Trocar os IDs dos processos
          temp = proc[i];
          proc[i] = proc[idMax];
          proc[idMax] = temp;

          // Trocar as prioridades
          temp = priority[i];
          priority[i] = priority[idMax];
          priority[idMax] = temp;

          // Trocar os burst times
          temp = burst_time[i];
          burst_time[i] = burst_time[idMax];
          burst_time[idMax] = temp;

          // Trocar os tempos de chegada
          temp = arrival_time[i];
          arrival_time[i] = arrival_time[idMax];
          arrival_time[idMax] = temp;
       }
// Atualizar o current_time apos escalonar um processo
current_time += burst_time[i];
}
```

Prioridades Teste



Round Robin

- Percorre a fila atribuindo um quantum a cada processo.
- Após executar seu quantum o processo retorna ao fim da fila.



RR

- Implementado dentro da função waitingtime
- Quantum de 2 u.t.
- Laço do-while ocorre enquanto ainda houver processos não concluídos.
- Laço for percorre todos os processos dando o quantum de tempo a cada um.

```
do{
    done = 1; // Suponha que todos os processos sejam concluidos
    for (int i = 0; i < n; i++) { // Atravesse todos os processos
        if (remaining_time[i] > 0) { // Se o processo ainda nao estiver concluido
            done = 0; // Marcar como nao concluido
            if(arrival_time[i] <= current_time){ // testa se o processo ja esta pronto para a execucao
                if (remaining_time[i] <= QUANTUM){ // Se o tempo restante for menor ou igual ao quantum
                    current_time += remaining_time[i]; // Adicionar o tempo restante ao horario atual
                    wait_time[i] = current_time - burst_time[i]-arrival_time[i] ; // Calcular tempo de espera
                    remaining_time[i] = 0; // Marcar processo como concluido
                } else {
                    current_time += QUANTUM; // Adicionar quantum ao tempo atual
                    remaining_time[i] -= QUANTUM; // Subtraia o quantum do tempo restante
                }
            }
        }
    }while(done == 0);
```

RR
Teste

```
File  Edit  View  Search  Terminal  Help
felps@felps-VirtualBox:~/Trabalho4$ ./RR
Processes  Burst Arrival  Waiting Turn around
 1         12             0       124     136
 2         5              3       21      26
 3         12             6       169     181
 4         8              9       123     131
 5         12             12      201     213
 6         8              15      121     129
 7         8              18      120     128
 8         5              21      64      69
 9         12             24      191     203
 10        8              27      158     166
 11        8              30      157     165
 12        8              33      156     164
 13        8              36      155     163
 14        5              39      113     118
 15        8              42      151     159
 16        8              45      150     158
 17        8              48      149     157
 18        5              51      108     113
 19        8              54      145     153
 20        8              57      144     152
 21        12             60      171     183
 22        5              63      103     108
 23        12             66      167     179
 24        5              69      100     105
 25        5              72      98      103
 26        12             75      160     172
 27        8              78      149     157
 28        8              81      148     156
 29        8              84      147     155
 30        12             87      152     164
Average waiting time = 137.166672
Average turn around time = 145.533340
felps@felps-VirtualBox:~/Trabalho4$
```

COMPARAÇÃO DE ALGORITMOS

	Tempo med. Espera	Tempo med. Resposta
SJF	55,3 u.t.	63,1 u.t.
SRTF	56,2 u.t.	64,4 u.t.
Prioridades	62,6 u.t.	70,1 u.t.
RR	137,1 u.t.	145,5 u.t.

SJF

- Vantagens:
  - Tempo médio de espera é menor;
  - Processos menores são favorecidos;
- Desvantagens:
  - É necessário saber o tempo de execução de cada processo;
  - Pode causar inanição em processos longos;

COMPARAÇÃO DE ALGORITMOS

	Tempo med. Espera	Tempo med. Resposta
SJF	55,3 u.t.	63,1 u.t.
SRTF	56,2 u.t.	64,4 u.t.
Prioridades	62,6 u.t.	70,1 u.t.
RR	137,1 u.t.	145,5 u.t.

SRTF

- Modelo preemptivo do SJF;
- Vantagens:
  - Tempo médio de espera é curto.
  - Tempo de resposta mais rápido para processos menores.
- Desvantagens:
  - Pode causar sobrecarga nas trocas de contexto;

COMPARAÇÃO DE ALGORITMOS

	Tempo med. Espera	Tempo med. Resposta
SJF	55,3 u.t.	63,1 u.t.
SRTF	56,2 u.t.	64,4 u.t.
Prioridades	62,6 u.t.	70,1 u.t.
RR	137,1 u.t.	145,5 u.t.

Prioridades

- Vantagens:
  - Garante que processos críticos sejam executados imediatamente.
- Desvantagens:
  - Pode gerar inanição nos processos de baixa prioridade.
  - Requer a atribuição de prioridades de forma equilibrada.

COMPARAÇÃO DE ALGORITMOS

	Tempo med. Espera	Tempo med. Resposta
SJF	55,3 u.t.	63,1 u.t.
SRTF	56,2 u.t.	64,4 u.t.
Prioridades	62,6 u.t.	70,1 u.t.
RR	137,1 u.t.	145,5 u.t.

RR

- Vantagens:
  - Todos os processos têm acesso igual a CPU.
  - Não há inanição.
- Desvantagens:
  - Tempo médio de espera e resposta geralmente é alto.
  - Pode gerar sobrecarga nas trocas de contexto.

Conclusão

- Não há algoritmo perfeito;
- Cada algoritmo trás resultados diferentes;
- A escolha do algoritmo de escalonamento depende das características desejadas para o sistema;