INF 1019
Primeiro Trabalho de Sistemas de Computação

Aluna: Michelle Andrade Valente da Silva
Matricula: 1312828

Arquivos

round_robin.c
Interpretador de comandos que envia os programas para o escalonador round robin.

prioridades.c
Interpretador de comandos que envia os programas para o escalonador de prioridades.

escalonadores.c
Arquivo com o código fonte dos dois escalonadores.

escalonadores.h
Arquivo header dos escalonadores

entrada.txt
Entrada de programas a serem testados

saida.txt
Arquivo gerado quando se executa o round_robin.c ou prioridades.c com opção de saída do stdout no arquivo.

Programas a serem testados:
prog1.c, prog2.c, prog3.c, prog4.c

Compilação

```
gcc prog1.c -o p1
gcc prog2.c -o p2
gcc prog3.c -o p3
gcc prog4.c -o p4
gcc round_robin.c escalonadores.c -o round_robin
gcc prioridades.c escalonadores.c -o prioridades
```

Executar

./round_robin ou ./prioridades
Escolher opção de stdout no terminal ou no arquivo. Se escolher no arquivo demora alguns segundos, ja que o teste tem esperas para ficar mais fácil de ver o funcionamento do escalonador quando se executa com o stdout no terminal.
Não esquecer de configurar diferente o arquivo entrada.txt para diferentes escalonadores.

Formato arquivo entrada.txt

Round Robin:
exec nome_programa
Prioridades:
exec nome_programa prioridade valor_prioridade

Testes Escalonador Round Robin

Primeiro teste

Rodar 3 programas em loops de diferentes tamanhos.
Prog1.c (p1): loop de 0 ate 6
Prog2.c (p2): loop de 0 ate 7
Prog4.c (p4): loop de 0 ate 5

entrada.txt:
exec p1
exec p2
exec p4

Resultado no stdout:

```
Programa: p1
Programa: p2
Programa: p4
P1 rodando 0
P1 rodando 1
P1 rodando 2
P1 rodando 3
P1 rodando 4
P2 rodando 0
P2 rodando 1
P2 rodando 2
P2 rodando 3
P2 rodando 4
P4 rodando 0
P4 rodando 1
P4 rodando 2
P4 rodando 3
P1 rodando 5
P1 rodando 6
Terminou  P1
P2 rodando 5
P2 rodando 6
P2 rodando 7
Terminou  P2
P4 rodando 4
P4 rodando 5
Terminou  P4
```

Comentarios:

O teste mostra o funcionamento do round robin, onde cada programa executa em um time slice. Pode-se observar que algumas vezes alguns programas rodam um loop a mais que outros, isso ocorre por conta da sincronização, ja que esta se chamando sleep(1) tanto no escalonador para contar o time slice quanto no programa para ser mais fácil de observar como esta sendo rodado.

Segundo Teste

Rodar 4 programas em loops de diferentes tamanhos com um programa tendo espera por i/o.

Prog1.c (p1): loop de 0 ate 6
Prog2.c (p2): loop de 0 ate 7
Prog3.c (p3): loop de 0 ate 11, com espera de I/O na terceira iteração.
Prog4.c (p4): loop de 0 ate 5

entrada.txt:
exec p1
exec p2
exec p3
exec p4

Resultado no stdout:

Programa: p1
Programa: p2
Programa: p3
Programa: p4
P1 rodando 0
P1 rodando 1
P1 rodando 2
P1 rodando 3
P1 rodando 4
P2 rodando 0
P2 rodando 1
P2 rodando 2
P2 rodando 3
P3 rodando 0
P3 rodando 1
P3 rodando 2
I/O recebido!
P4 rodando 0
P4 rodando 1
I/O terminado!
P4 rodando 2
P4 rodando 3
P1 rodando 5
P1 rodando 6
Terminou P1
P2 rodando 4
P2 rodando 5
P2 rodando 6
P2 rodando 7
Terminou P2
P3 rodando 3
P3 rodando 4
P3 rodando 5
P3 rodando 6
P3 rodando 7
P4 rodando 4
P4 rodando 5
Terminou P4
P3 rodando 8

P3 rodando 9
P3 rodando 10
P3 rodando 11
Terminou P3

Comentarios:

O teste mostra que o round robin continua funcionando mesmo quando um programa precisa esperar pelo I/O. Quando isso acontece, ele apenas passa para o proximo programa da lista, como esperado. O mesmo problema de sincronização do teste anterior pode ser observado nesse teste.

Testes Escalonador de Prioridades

Primeiro teste

Rodar 3 programas em loops de diferentes tamanhos com diferentes prioridades.
Prog1.c (p1): loop de 0 ate 6 com prioridade 2
Prog2.c (p2): loop de 0 ate 7 com prioridade 1
Prog4.c (p4): loop de 0 ate 5 com prioridade 3

entrada.txt:
exec p1 prioridade 2
exec p2 prioridade 1
exec p4 prioridade 3

Resultado no stdout:

Programa: p1, prioridade: 2
Programa: p2, prioridade: 1
Programa: p4, prioridade: 3
Entrou na lista de pronto: p1 com prioridade: 2
P1 rodando 0
P1 rodando 1
P1 rodando 2
Entrou na lista de pronto: p2 com prioridade: 1
P2 rodando 0
P2 rodando 1
P2 rodando 2
Entrou na lista de pronto: p4 com prioridade: 3
P2 rodando 3
P2 rodando 4
P2 rodando 5
P2 rodando 6
P2 rodando 7
P1 rodando 3
P1 rodando 4
P1 rodando 5
P1 rodando 6
P4 rodando 0
P4 rodando 1
P4 rodando 2
P4 rodando 3
P4 rodando 4
P4 rodando 5

Comentarios:

Podemos observar que os programas com maior prioridade tem preferencia na ordem de serem executados. Colocando programas com menor prioridade na lista de prontos ate os que com maior prioridade terminem de executar.

Segundo teste

Rodar 4 programas em loops de diferentes tamanhos com diferentes prioridades, e um programa com espera de i/o.

Prog1.c (p1): loop de 0 ate 6 com prioridade 2
Prog2.c (p2): loop de 0 ate 7 com prioridade 1
Prog3.c (p3): loop de 0 ate 11, com espera de I/O na terceira iteração, com prioridade 3
Prog4.c (p4): loop de 0 ate 5 com prioridade 4

entrada.txt:
exec p1 prioridade 2
exec p2 prioridade 1
exec p3 prioridade 3
exec p4 prioridade 4

Resultado no stdout:

```
Programa: p1, prioridade: 2
Programa: p2, prioridade: 1
Programa: p3, prioridade: 3
Programa: p4, prioridade: 4
Entrou na lista de pronto: p1 com prioridade: 2
P1 rodando 0
P1 rodando 1
P1 rodando 2
Entrou na lista de pronto: p2 com prioridade: 1
P2 rodando 0
P2 rodando 1
P2 rodando 2
Entrou na lista de pronto: p3 com prioridade: 3
P2 rodando 3
P2 rodando 4
P2 rodando 5
P2 rodando 6
Entrou na lista de pronto: p4 com prioridade: 4
P2 rodando 7
P1 rodando 3
P1 rodando 4
P1 rodando 5
P1 rodando 6
P3 rodando 0
P3 rodando 1
P3 rodando 2
I/O recebido!
P4 rodando 0
P4 rodando 1
P4 rodando 2
I/O terminado!
P3 rodando 3
P3 rodando 4
P3 rodando 5
P3 rodando 6
P3 rodando 7
P3 rodando 8
P3 rodando 9
```

P3 rodando 10
P3 rodando 11
P4 rodando 3
P4 rodando 4
P4 rodando 5

Comentarios:

Podemos observar que os programas com maior prioridade tem preferencia na ordem de serem executados. E quando um programa precisa esperar por um I/O, ele sai da lista de prontos, e quando volta se tiver prioridade maior que o atual, ele voltará a ser executado.