João Pedro Garcia - 1211768
Thiago Klein de Angelis - 1321929

Sistemas de computação - Trabalho 1

O trabalho 1 da matéria inf1019(Sistemas de computação) consistiu em fazer um escalonador de tarefas. Foi utilizado a linguagem C e o compilador utilizado foi o gcc.

O código fonte se encontra no arquivo main.c
Os programas a serem executados são : “programa1”, “programa2”, “programa3”, “programa4”, “programa5”, “programa6”, “programa7”. Seus código fontes por sua vez se encontram em seus arquivos com extensão .c

Durante o teste todos os programas serão executados. A main será o arquivo a ser chamado e executará cada programa como um processo. Tanto por ROUND-ROBIN quanto por PRIORIDADE ou LOTERIA.

Optamos por fazer três “mains” diferentes uma para cada tipo de escalonamento. Sendo assim existem três pastas diferentes, cada uma com um determinado tipo de escalonador. Para testar cada uma basta rodar a main.

Dentro de cada main há dois processos separados que realizam as operações. Um processo é o interpretador que, como diz o nome, lê do arquivo de entrada contendo as instruções e informações sobre os programas a serem executados e o outro é o próprio escalonador que realiza a preempção determinada. Os dois processos se comunicam através de memória compartilhada. Há um vetor que contem as informações dos processos a serem escalonados, uma variável com o numero desses processos e uma outra que determina se o interpretador já terminou ou não de ler o arquivo.

A ordem de entrada para o escalonador será sempre a ordem do arquivo de entrada “exec.txt”. Em todos os casos os interpretadores leem uma linha a cada três segundos e através de uma variável existente em memória compartilhada(numero de processos) o escalonador sabe se deve chamar a função criarProcesso que realiza um fork() e gera um novo processo a ser escalonado.

Em todos os casos o escalonador espera que haja ao menos um processo a ser executado para começar a rodar, quando isso acontece ele imediatamente cria o processo baseado nas informações coletadas pelo interpretador e executa este.

No caso de prioridade ele executa o processo por 3 segundos, ou seja, até haver um novo processo para ser criado, caso esse novo processo tenha uma prioridade maior(numero menor) ele tomará a frente e será executado, caso contrario o processo antigo continuará executando. Assim se repetindo até que todos os processos sejam encerrados. No momento da execução a primeira coisa que o escalonador faz é verificar se existe algum processo no estado pronto. Caso haja um ou mais ele

compara as prioridades e pega o de maior prioridade. Todos os processos tem um atributo status que determina se ele está no estado pronto para execução ou não.

No caso round-robin ele executará cada processo por 0.5 segundos(marcado com usleep) e pausará após esse tempo, verifica se o processo foi encerrado e marca ele como encerrado ou apenas pausa. Se houver outro(s) processo(s) nas fila ele executará o proximo. Deste modo executando aos poucos cada um de maneira circular. Assim como o caso de prioridade a cada 3 segundos o escalonador verifica a existência de um novo processo na lista dada pelo interpretador. Caso exista ele cria o processo e o coloca na fila.

No caso Lottery o escalonador sempre que recebe um processo novo(a cada três segundos, como nos outros dois casos) cria esse processo e sorteia e atribui a ele um numero de tickets. A cada 0.5 segundos ele sorteia um ticket e executa o processo “dono” desse ticket por meio segundo, pausando em seguida. Após o termino de determinado processo os tickets voltam para a lista de tickets validos onde podem ser novamente sorteados para outros processos. É interessante notar que há uma lista com os tickets disponíveis para novos processos que entrarem no escalonador e uma lista que contem os números do tickets que ja estão sendo utilizados. Sendo assim não há risco de dois processos que existem simultaneamente terem os mesmos tickets e como o sorteio de que processo deve ser executado ocorre a partir da lista de tickets não disponíveis(aqueles que já estão atrelados a algum processo) não existe o risco de starvation, a não ser que exista um programa infinito.

Todos os processos auxiliares que tem alguma operação de entrada e saída tem seus próprios arquivos de entrada e saída. Para rodar os programas basta acessar a pasta e rodar a main.

Exemplo Prioridades:
Exemplo 1 -

Entrada - exec.txt

```
exec programa6 prioridade= 4
exec programa1 prioridade= 1
exec programa3 prioridade= 7
```

De acordo com os arquivos de entrada e saída podemos ver que um programa que tem prioridade maior e entra na fila toma a dianteira e é executado.

No caso o programa6 está na frente e quando o programa1 entra com uma prioridade maior o 6 pausa e o 1 passa a ser executado.

O programa 1 termina e o novo processo, programa3 entra na fila. O programa3 tem uma prioridade menor do que o programa6, sendo assim o programa6 passa a ser executado e o programa3 espera que ele termine para começar.

Vemos também que os numero de prioridade não precisam estar numa sequencia exata.

Saida - saida.txt

```
Escalonamento escolhido: LISTA DE
PRIORIDADES

 ----------
Entrada de novo processo na fila
Nome: programa6
Pid: 388
Prioridade: 4
---------------

Em execucao - programa programa6

 ----------
Entrada de novo processo na fila
Nome: programa1
Pid: 413
Prioridade: 1
---------------

Em execucao - programa programa1
Programa programa1 terminou sua execução...


 ----------
Entrada de novo processo na fila
Nome: programa3
Pid: 430
Prioridade: 7
---------------

Em execucao - programa programa6
Em execucao - programa programa6
Em execucao - programa programa6
Programa programa6 terminou sua execução...

Em execucao - programa programa3
Programa programa3 terminou sua execução...
```

Exemplo 2 -

Entrada - exec.txt

```
exec programa7 prioridade= 1
exec programa1 prioridade= 2
exec programa2 prioridade= 3
```

Vemos nesse exemplo que o programa7 demora para terminar, e durante sua execução dois processos chegam a entrar na fila, mas como tem uma prioridade menor eles devem esperar o programa7 terminar para serem executados.

É interessante notar nesse exemplo que um programa mais longo pode causar a espera de programas muito curtos.

Nota-se que em caso de o programa7 ser infinito os programas 1 e 2 sofreriam de starvation.

Saida - saida.txt

```
Escalonamento escolhido: LISTA DE
PRIORIDADES

 ----------
Entrada de novo processo na fila
Nome: programa7
Pid: 522
Prioridade: 1
---------------

Em execucao - programa programa7

 ----------
Entrada de novo processo na fila
Nome: programa1
Pid: 523
Prioridade: 2
---------------

Em execucao - programa programa7

 ----------
Entrada de novo processo na fila
Nome: programa2
Pid: 524
Prioridade: 3
---------------

Em execucao - programa programa7
Em execucao - programa programa7
Programa programa7 terminou sua execução...

Em execucao - programa programa1
Programa programa1 terminou sua execução...

Em execucao - programa programa2
Programa programa2 terminou sua execução...
```

Exemplos Round-Robin
Exemplo 1-

Entrada - exec.txt

```
exec programa5
exec programa3
```

Aqui temos um exemplo de como um escalonamento round robin funciona. A cada meio segundo pausando o processo, verificando se há algum outro processo a ser executado e se sim rodando, se não continuando com o processo antigo.

É interessante notar nesse exemplo que até o final da execução do programa5 o programa3 ainda não havia entrado na fila, assim os programas não chegaram a alternar sua execução.

Saida - saida.txt

```
Escalonamento escolhido: ROUND ROBIN

 ----------
Entrada de novo processo na fila
Nome: programa5
Pid: 546
---------------

Em execucao - Programa programa5 - Pid 546
Pausando - Programa programa5

Em execucao - Programa programa5 - Pid 546
Pausando - Programa programa5

Em execucao - Programa programa5 - Pid 546
Pausando - Programa programa5

Em execucao - Programa programa5 - Pid 546
Programa programa5 terminou sua execução...


 ----------
Entrada de novo processo na fila
Nome: programa3
Pid: 547
---------------

Em execucao - Programa programa3 - Pid 547
Programa programa3 terminou sua execução...
```

Exemplo 2-

Entrada - exec.txt

```
exec programa7
exec programa5
exec programa6
```

Obs: para esse exemplo vi a necessidade de cortar alguns trechos irrelevantes meramente repetitivos da saída.

Nota-se que no inicio o programa7 esta executando sozinho, após a entrada do programa5 eles começam a alternar sua execução os dois e com a entrada do programa6 eles começam a revezar os 3.

Após o termino de um dos processos os dois outros continuam sofrendo preempção até que todos tenham terminado.

É visível que independente do tempo de execução do código com o round robin nunca haverá starvation de algum processo, pois assim pouco depois que entra na fila já será executado. Mesmo que haja algum processo extremamente longo aos poucos os processos que sofreriam starvation caso fosse uma lista de prioridades são executados.

Saida - saida.txt

```
Escalonamento escolhido: ROUND ROBIN

 ----------
Entrada de novo processo na fila
Nome: programa7
Pid: 555
---------------

Em execucao - Programa programa7 - Pid 555
Pausando - Programa programa7

[...]

Em execucao - Programa programa7 - Pid 555
Pausando - Programa programa7


 ----------
Entrada de novo processo na fila
Nome: programa5
Pid: 556
---------------

Em execucao - Programa programa5 - Pid 556
Pausando - Programa programa5

Em execucao - Programa programa7 - Pid 555
Pausando - Programa programa7

Em execucao - Programa programa5 - Pid 556
Pausando - Programa programa5

Em execucao - Programa programa7 - Pid 555
Pausando - Programa programa7

Em execucao - Programa programa5 - Pid 556
Pausando - Programa programa5

Em execucao - Programa programa7 - Pid 555
Pausando - Programa programa7


 ----------
Entrada de novo processo na fila
Nome: programa6
Pid: 560
---------------

Em execucao - Programa programa5 - Pid 556
Programa programa5 terminou sua execução...

Em execucao - Programa programa6 - Pid 560
Pausando - Programa programa6

Em execucao - Programa programa7 - Pid 555
Pausando - Programa programa7

Em execucao - Programa programa6 - Pid 560
Pausando - Programa programa6

Em execucao - Programa programa7 - Pid 555
Pausando - Programa programa7

[...]


Em execucao - Programa programa6 - Pid 560
Pausando - Programa programa6

Em execucao - Programa programa7 - Pid 555
Programa programa7 terminou sua execução...

Em execucao - Programa programa6 - Pid 560
Pausando - Programa programa6

Em execucao - Programa programa6 - Pid 560
Programa programa6 terminou sua execução...
```

Exemplos Lottery.
Exemplo 1-

Entrada - exec.txt

exec programa7 numtickets= 3
exec programa2 numtickets= 1
exec programa6 numtickets= 2

Podemos ver nesse exemplo que existe uma alternância entre os processos que se da de forma pseudo aleatória.

Podemos também notar que nunca é sorteado para execução um bilhete que não esta atrelado a algum dos processos isso de da por que o sorteio é apenas entre os bilhetes atrelados de modo a não causar starvation.

Vemos também após a entrada do 6 que o numero de tickets influencia na quantidade de vezes que um processo deve rodar. O processo com 3 tickets (programa7)tende a ser sorteado mais frequentemente do que um processo com 2(programa6)

Saida - saida.txt

Escalonamento escolhido: LOTERIA

Entrada de novo processo na fila
Nome: programa7
Pid: 533
Bilhetes: 14 15 2

Bilhete sorteado: 2
Em execucao - Programa programa7 - Pid 533
Pausando - Programa programa7

Bilhete sorteado: 14
Em execucao - Programa programa7 - Pid 533
Pausando - Programa programa7

Bilhete sorteado: 15
Em execucao - Programa programa7 - Pid 533
Pausando - Programa programa7

Entrada de novo processo na fila
Nome: programa2
Pid: 534
Bilhetes: 7

Bilhete sorteado: 7
Em execucao - Programa programa2 - Pid 534
Programa programa2 terminou sua execução...

Bilhete sorteado: 14
Em execucao - Programa programa7 - Pid 533
Pausando - Programa programa7

Bilhete sorteado: 15
Em execucao - Programa programa7 - Pid 533
Pausando - Programa programa7

Entrada de novo processo na fila
Nome: programa6
Pid: 535
Bilhetes: 6 17

Bilhete sorteado: 15
Em execucao - Programa programa7 - Pid 533
Pausando - Programa programa7

Bilhete sorteado: 6
Em execucao - Programa programa6 - Pid 535
Pausando - Programa programa6

Bilhete sorteado: 14
Em execucao - Programa programa7 - Pid 533
Pausando - Programa programa7

Bilhete sorteado: 2
Em execucao - Programa programa7 - Pid 533
Pausando - Programa programa7

Bilhete sorteado: 17
Em execucao - Programa programa6 - Pid 535
Pausando - Programa programa6

Bilhete sorteado: 15
Em execucao - Programa programa7 - Pid 533
Pausando - Programa programa7

Bilhete sorteado: 6
Em execucao - Programa programa6 - Pid 535
Pausando - Programa programa6

Bilhete sorteado: 14
Em execucao - Programa programa7 - Pid 533
Pausando - Programa programa7

Bilhete sorteado: 2
Em execucao - Programa programa7 - Pid 533
Programa programa7 terminou sua execução...

Bilhete sorteado: 17
Em execucao - Programa programa6 - Pid 535
Pausando - Programa programa6

Bilhete sorteado: 6
Em execucao - Programa programa6 - Pid 535
Programa programa6 terminou sua execução...

Exemplo 2-

Entrada - exec.txt

exec programa1 numtickets= 4
exec programa2 numtickets= 4
exec programa3 numtickets= 4
exec programa4 numtickets= 4
exec programa5 numtickets= 4

Nesse exemplo é interessante notar que um bilhete ja sorteado foi sorteado novamente após o primeiro processo que o recebeu ter sido encerrado, que é o caso dos programas 1, 3 e 4. Todos eles receberam o bilhete 17.

Saida - saida.txt

Escalonamento escolhido: LOTERIA

Entrada de novo processo na fila
Nome: programa1
Pid: 560
Bilhetes: 12 1 14 17

Bilhete sorteado: 12
Em execucao - Programa programa1 - Pid 560
Programa programa1 terminou sua execução...

Entrada de novo processo na fila
Nome: programa2
Pid: 561
Bilhetes: 18 11 0 19

Bilhete sorteado: 0
Em execucao - Programa programa2 - Pid 561
Programa programa2 terminou sua execução...

Entrada de novo processo na fila
Nome: programa3
Pid: 562
Bilhetes: 17 7 5 1

Bilhete sorteado: 1
Em execucao - Programa programa3 - Pid 562
Programa programa3 terminou sua execução...

Entrada de novo processo na fila
Nome: programa4
Pid: 563
Bilhetes: 17 2 13 18

Bilhete sorteado: 17
Em execucao - Programa programa4 - Pid 563
Programa programa4 terminou sua execução...

Entrada de novo processo na fila
Nome: programa5
Pid: 564
Bilhetes: 2 5 14 11

Bilhete sorteado: 5
Em execucao - Programa programa5 - Pid 564
Pausando - Programa programa5

Bilhete sorteado: 2
Em execucao - Programa programa5 - Pid 564
Programa programa5 terminou sua execução...