Universidade de Évora

Escola de Ciências e Tecnologia

Departamento de Engenharia Informática

Sistemas Operativos I

Docente Luís Rato

2019/2020

1º Trabalho

```
  0 | READY  101           | RUN  100  | BLOCKED
  1 | READY  101 200 300   | RUN  100  | BLOCKED
  2 | READY  101 200 300   | RUN  100  | BLOCKED
  3 | READY  101 200 300   | RUN  100  | BLOCKED
  4 | READY  200 300       | RUN  101  | BLOCKED  100
  5 | READY  300           | RUN  200  | BLOCKED  100 101
  6 | READY                | RUN  300  | BLOCKED  100 101 200
  7 | READY                | RUN  300  | BLOCKED  100 101 200
  8 | READY                | RUN       | BLOCKED  100 101 200 300
  9 | READY                | RUN       | BLOCKED  100 101 200 300
 10 | READY  101           | RUN  100  | BLOCKED  200 300
 11 | READY                | RUN  101  | BLOCKED  200 300 100
 12 | READY                | RUN  101  | BLOCKED  200 300 100
 13 | READY                | RUN  200  | BLOCKED  300 100
 14 | READY  300           | RUN  200  | BLOCKED  100
 15 | READY                | RUN  300  | BLOCKED  100 200
 16 | READY                | RUN       | BLOCKED  100 200
 17 | READY                | RUN       | BLOCKED  100 200
 18 | READY                | RUN  100  | BLOCKED  200
 19 | READY                | RUN       | BLOCKED  200
 20 | READY                | RUN       | BLOCKED  200
 21 | READY                | RUN       | BLOCKED  200
 22 | READY                | RUN       | BLOCKED  200
 23 | READY                | RUN  200  | BLOCKED
 24 | READY                | RUN  200  | BLOCKED
 25 | READY                | RUN  200  | BLOCKED
 26 | READY                | RUN  200  | BLOCKED
Todos os Processos Terminaram.
```

José Santos – 43017

Introdução

Maior parte do trabalho está implementado na Main, com a exceção de uma função (print_instante), foram utilizadas queues, listas e uma estrutura criada por mim chamada process. Existe tambem um define QUANTUM.

Estruturas de Dados

As queues e as listas estão implementadas normalmente, com a adição de algumas funções.

A struct process, com dois argumentos o PID e o chegada, contem dois inteiros, um PID que identifica cada processo, e um chegada, para guardar o tempo de chegada, contem também um bool entrou que indica se o processo já entrou ou não, por fim tem duas queues, processRUN e processBlocked, estas servem para guardar o número de instantes que o processo tem de ficar no run e blocked.

Main

No inicio são criadas várias variáveis:

- char c - vai servir para ler o '\n' nos scans dos processos.
- int n - serve para guardar os valores correspondentes aos processos.
- int totalProcessos – irá guardar o número total de Processos, é inicializado a 0
- int ProcessosActivos– no final de adicionar todos os processos ao Array de Processos, este valor será igual ao totalProcessos.
- Int tempoCPU – Servirá para calcular quantos instantes um processo está no RUN, é inicializado a 0.

É criado o array de processos - processArr[50] – todos os processos criados serão aqui guardados.

São criadas duas queues, READY e RUN, e uma lista BLOCKED, estes serão os vários estados do programa.

Um ciclo for coloca todas as posições do processArr a NULL.

Outro ciclo for que vai fazer o scans necessários para guardar os processos e a sua informação, primeiro é criado um process dando o PID e o chegada lidos dentro do ciclo for, é utilizado um do..while até que seja encontrado o caracter nulo (não consegui usar para o '\r'), dentro deste do..while, é feito um scan guardando o valor no int n, e de seguida um scan para o char c.

É utilzada uma variável auxiliar x, este x começa a 0 e é incrementado por cada valor que é lido, se o x for pár, significa que o valor lido é tempo que o processo necessita de ficar no RUN, se for impár significa que tem de ficar no BLOCKED.

Com isto, é utilizado um if else, que verifica se o x é par ou impár e o valor de n é adicionado à queue respectiva do processo, verifica-se se o c é diferente de '\n', se não for, o ciclo continua, se for o processo é adicionado ao processArr, e o ciclo for continua até acabar de ler todos os processos.

Agora começa o ciclo, que irá acabar quando o processosActivos for igual a 0, dentro do ciclo for é inicializada a variável instante.

Um for de modo a percorrer todos os processos dentro do processArr e atribuí-lo ao process p, um if que verifica se o process já entrou, se sim, vai procurar o processo no RUN, se este estiver no RUN, decrementa o valor no processRun e incrementa o tempoCPU, se o processo não estiver no RUN, mas sim no BLOCKED, decrementa o valor no processBlocked.

Agora dois ciclos for, um que percorre por ordem todos os processos no BLOCKED, e outro que percorre os processos no processArr, é realizada uma comparação entre o PID que está no BLOCKED com o PID dos processos que estão no processArr, se este for igual e o queue_front do processBlocked do processo for igual a 0, significa que este processo já não necessita de ficar no BLOCKED, entao é feito um dequeue do processBlocked e é removido o PID da lista BLOCKD, depois um if que verifica se o RUN está vazio, se sim, este processo vai para o RUN, se não, vai para o READY.

Mais um ciclo para percorrer todos os processos, desta vez de modo a encontrar qual o que está no RUN, é realizada uma comparação semelhante à enterior, comparando os PID's, se estes foram iguais, verifica-se se o queue_front do processRun é igual a 0, isto significa que o programa acabou o que tinha a fazer, são realizados os dequeues, e verifica-se se o processo ainda tem de ir para o BLOCKED, se sim é colocado, se não é ignorado por agora, é colocado o tempoCPU a 0, mas se o queue_front do processRun é diferente de 0, mas o tempoCPU é

igual ao QUANTUM, o processo é retirado do RUN e colocado no READY, e o tempoCPU é colocado a 0.

É feita uma condição, se o RUN estiver vazio, mas o READY não estiver, é colocado o primeiro processo do READY para o RUN.

Ainda mais um ciclo para percorrer todos os processos, desta vez verifica se o processo já entrou, se não, e o RUN estiver vazio, o processo é lá colocado, se não estiver vazio é colocado no READY, e é mudado o bool entrou para true.

Se o processo já entrou, verifica-se o tamanho do seu processBlocked e do seu processRUN se forem ambos 0, significa que o processo já acabou, então, a sua posição no processArr é colocada a NULL, é realizado um free do processo e é decrementado o numero de processosActivos.

Por fim uma condiçao que verifica se o número de processosActivos é maior que 0 se sim é realizada a chamada da função print_instante, se não é printado a dizer que todos os processos terminaram

Print_instantes

Esta função tem como objetivo realizar o output do programa, tem como argumentos o int instante, as queues READY e RUN, a lista BLOCKED e o int totalProcessos.

É realizado um calculo do número de algarismos do instante para que todas as barras verticais fiquem alinhadas, é realizado o print da queue READY, tendo certas coisas em conta, o número de processos que estão no READY, o numero total de processos e que não podem estar todos os processos no READY, então por cada processo que não está no ready (menos 1) são printados espaços, de seguida é printado o RUN, se este estiver vazio, são printados espaços, e por fim é printado o BLOCKED.

Conclusão

Fora alguns problemas, como o input não pode conter espaços a mais, a situação do \r, a quantidade de ciclos para percorrer o array de processos porque as queues estão implentadas para inteiros e não para estruturas, creio que o trabalho correu bem, e não houve grande dificuldade.

Output do ficheiro input1.txt

```
100 0 1 3 10 3 6
101 0 4 4 2
200 1 2 5 1 2 3
300 1 7 6 1
```

FCFS

```
  0 | READY  101           | RUN  100  | BLOCKED
  1 | READY  200 300       | RUN  101  | BLOCKED  100
  2 | READY  200 300       | RUN  101  | BLOCKED  100
  3 | READY  200 300       | RUN  101  | BLOCKED  100
  4 | READY  200 300 100   | RUN  101  | BLOCKED
  5 | READY  300 100       | RUN  200  | BLOCKED  101
  6 | READY  300 100       | RUN  200  | BLOCKED  101
  7 | READY  100           | RUN  300  | BLOCKED  101 200
  8 | READY  100           | RUN  300  | BLOCKED  101 200
  9 | READY  100 101       | RUN  300  | BLOCKED  200
 10 | READY  100 101       | RUN  300  | BLOCKED  200
 11 | READY  100 101       | RUN  300  | BLOCKED  200
 12 | READY  100 101 200   | RUN  300  | BLOCKED
 13 | READY  100 101 200   | RUN  300  | BLOCKED
 14 | READY  101 200       | RUN  100  | BLOCKED  300
 15 | READY  101 200       | RUN  100  | BLOCKED  300
 16 | READY  101 200       | RUN  100  | BLOCKED  300
 17 | READY  101 200       | RUN  100  | BLOCKED  300
 18 | READY  101 200       | RUN  100  | BLOCKED  300
 19 | READY  101 200       | RUN  100  | BLOCKED  300
 20 | READY  101 200 300   | RUN  100  | BLOCKED
 21 | READY  101 200 300   | RUN  100  | BLOCKED
 22 | READY  101 200 300   | RUN  100  | BLOCKED
 23 | READY  101 200 300   | RUN  100  | BLOCKED
 24 | READY  200 300       | RUN  101  | BLOCKED  100
 25 | READY  200 300       | RUN  101  | BLOCKED  100
 26 | READY  300           | RUN  200  | BLOCKED  100
 27 | READY  100           | RUN  300  | BLOCKED  200
 28 | READY                | RUN  100  | BLOCKED  200
 29 | READY  200           | RUN  100  | BLOCKED
 30 | READY  200           | RUN  100  | BLOCKED
 31 | READY  200           | RUN  100  | BLOCKED
 32 | READY  200           | RUN  100  | BLOCKED
 33 | READY  200           | RUN  100  | BLOCKED
 34 | READY                | RUN  200  | BLOCKED
 35 | READY                | RUN  200  | BLOCKED
 36 | READY                | RUN  200  | BLOCKED
Todos os Processos Terminaram.
```

RoundRobin (Quantum = 3)

```
  0 | READY  101           | RUN  100  | BLOCKED
  1 | READY  200 300       | RUN  101  | BLOCKED  100
  2 | READY  200 300       | RUN  101  | BLOCKED  100
  3 | READY  200 300       | RUN  101  | BLOCKED  100
  4 | READY  300 100 101   | RUN  200  | BLOCKED
  5 | READY  300 100 101   | RUN  200  | BLOCKED
  6 | READY  100 101       | RUN  300  | BLOCKED  200
  7 | READY  100 101       | RUN  300  | BLOCKED  200
  8 | READY  100 101       | RUN  300  | BLOCKED  200
  9 | READY  101 300       | RUN  100  | BLOCKED  200
 10 | READY  101 300       | RUN  100  | BLOCKED  200
 11 | READY  101 300 200   | RUN  100  | BLOCKED
 12 | READY  300 200 100   | RUN  101  | BLOCKED
 13 | READY  200 100       | RUN  300  | BLOCKED  101
 14 | READY  200 100       | RUN  300  | BLOCKED  101
 15 | READY  200 100       | RUN  300  | BLOCKED  101
 16 | READY  100 300       | RUN  200  | BLOCKED  101
 17 | READY  300 101       | RUN  100  | BLOCKED  200
 18 | READY  300 101       | RUN  100  | BLOCKED  200
 19 | READY  300 101 200   | RUN  100  | BLOCKED
 20 | READY  101 200 100   | RUN  300  | BLOCKED
 21 | READY  200 100       | RUN  101  | BLOCKED  300
 22 | READY  200 100       | RUN  101  | BLOCKED  300
 23 | READY  100           | RUN  200  | BLOCKED  300
 24 | READY  100           | RUN  200  | BLOCKED  300
 25 | READY  100           | RUN  200  | BLOCKED  300
 26 | READY                | RUN  100  | BLOCKED  300
 27 | READY  300           | RUN  100  | BLOCKED
 28 | READY  300           | RUN  100  | BLOCKED
 29 | READY  100           | RUN  300  | BLOCKED
 30 | READY                | RUN  100  | BLOCKED
 31 | READY                | RUN       | BLOCKED  100
 32 | READY                | RUN       | BLOCKED  100
 33 | READY                | RUN       | BLOCKED  100
 34 | READY                | RUN  100  | BLOCKED
 35 | READY                | RUN  100  | BLOCKED
 36 | READY                | RUN  100  | BLOCKED
 37 | READY                | RUN  100  | BLOCKED
 38 | READY                | RUN  100  | BLOCKED
 39 | READY                | RUN  100  | BLOCKED
Todos os Processos Terminaram.
```