Simulador de Sistema Operativo - Trabalho Prático

Sistemas Operativos I

Tomás Dias nº42784

Objetivo

Foi proposto na cadeira de Sistemas Operativos I a realização de um trabalho prático com o objetivo de implementar um simulador de Sistema Operativo com escalonamento ***Round Robin*** e gestão de memória **paginada** que consome programas constituídos por um conjunto instruções. O procedimento feito tendo como objetivo a implementação do simulador encontra-se descrito de seguida.

Estruturas de Dados

Para o bom funcionamento do escalonamento dos processos foi necesário implementar uma estrutura de filas. Para o efeito, foi implementado uma struct *Queue* que é constituída pelos seguintes atributos:

- **front**: Este atributo do tipo *int* corresponde ao indíce da parte frontal da fila.
- **rear**: Este atributo do tipo *int* corresponde ao indíce correspondente da parte traseira da fila.
- **size**: Este atributo do tipo *int* corresponde ao tamanho atual da fila.
- **capacity**: Este atributo do tipo *unsigned* corresponde ao tamanho máximo ou capacidade da fila.
- **array**: Este atributo do tipo *int** corresponde ao array onde são guardados os items da fila.

Em relação aos processos, foi implementado uma struct *processos* que é constituída pelos seguintes atributos:

- **pid**: Este atributo do tipo *int* corresponde ao pid do processo.
- **t_inicio**: Este atributo do tipo *int* corresponde ao instante de entrada do processo.
- **instrucoes**: Este atributo do tipo *int* corresponde ao array que guarda os códigos das instruções do processo.
- **parametros**: Este atributo do tipo *int* corresponde ao array que guarda os valores dos parâmetros das instruções do processo.
- **nist**: Este atributo do tipo *int* corresponde ao número de instruções (e respetivos parâmetros) do processo.
- **variaveis**: Este atributo do tipo *int* corresponde ao array que guarda os índices da memória onde as variáveis do processo se localizam.
- **instmem**: Este atributo do tipo *int* corresponde ao array que guarda os índices da memória onde as instruções do processo se localizam.
- **parammem**: Este atributo do tipo *int* corresponde ao array que guarda os índices da memória onde os parâmetros das instruções do processo se localizam.
- **nisntmem**: Este atributo do tipo *int* corresponde ao índice do valor de **instmem** (e de **parammem**) do processo que está a ser executado.
- **ninout**: Este atributo do tipo *int* corresponde ao número de instantes restantes que o processo deve ficar em BLOCKED.

A gestão de todos os processos é feita no array dinâmico **pr** que é definido na função **so_pag**.

Com vista a implementação do algoritmo de paginação de memória, foi criada a struct *paginas* que é constituída pelos seguintes atributos:

- **indice_inicial**: Este atributo do tipo *int* corresponde ao índice da memória onde se inicia a página.
- **indice_final**: Este atributo do tipo *int* corresponde ao índice da memória onde se finaliza a página.

A gestão das páginas da memória é feita no array **pag** que é definido na função **so_pag**.

O array correspondente à memória, **mem**, de dimensão 200, é definido na função **so_pag**. Para uma melhor interpretação dos espaços livres e ocupados da mesma, foram definidos com o valor -100 os espaços livres.

Funções

Tendo em vista a implementação das filas de processos, foram utilizadas as seguintes funções:

- **struct Queue* createQueue()**: Esta função cria uma fila recebendo como argumento a capacidade da mesma.
- **int isFull()**: Esta função indica se a fila está lotada, ou seja, se o tamanho da fila é igual à sua capacidade. Recebe como argumento a fila.
- **int isEmpty()**: Esta função indica se a fila está vazia, ou seja, se o tamanho da fila é igual a 0. Recebe como argumento a fila.
- **void enqueue()**: Esta função adiciona um processo à parte mais atrás da fila, recebendo como argumentos a fila à qual vai ser adicionado o processo e o próprio processo.
- **int dequeue()**: Esta função remove o processo que se encontra na frente da fila, recebendo como argumento a fila à qual o processo será removido e retorna o processo removido.
- **int front()**: Esta função indica qual o processo que está na frente da fila, recebendo como argumento a fila.
- **int rear()**: Esta função indica qual o processo que está na parte de trás da fila, recebendo como argumento a fila.

De maneira a concluir a implementação do escalonamento, foram utilizadas as seguintes funções auxiliares:

- **void coloca_posicao()**: Esta função coloca o processo numa determinada posição da fila. Recebe como argumentos o processo, a posição da fila na qual se pretende colocar o processo e a fila.

- **int elimina()**: Esta função elimina o processo do array de processos, returnando o tamanho atualizado do array. Recebe como argumentos o array de processos, o tamanho do array e o processo.

- **int verifica()**: Esta função verifica se o processo se encontra numa determinada fila returnando 1 se se verificar e 0 se não se verificar. Recebe como argumento o processo e a fila.

- **void ordena_menor()**: Esta função ordena os processos de forma crescente de tempos de chegada (t_inicio). Recebe como argumentos o array de processos e o tamanho do array.

- **void ordena_estado()**: Esta função ordena os processos pelo estado em que se encontram (RUN – BLOCKED – READY). Recebe como argumentos o array de processos, o tamanho do array e as três filas correspondentes aos estados (run, blocked e ready).

Com o intuito de implementar a de gestão de memória, foram utilizadas as seguintes funções auxiliares:

- **processos executa_instrucao()**: Esta função executa uma instrução de um processo returnando o processo com o valor do atributo **ninstmem** atualizado. Recebe como argumentos o processo, o array da memória, o código da instrução a ser executada, o índice da memória da variavel a ser atualizada, o valor do parâmetro da instrução a ser executada, a fila blocked e a fila que corresponde ao stdout.

- **int verifica_espaco_livre()**: Esta função verifica se existe espaço livre na memória para guardar o processo, returnando 1 se existe e 0 se não existe. Recebe como argumentos o array da memória, o array das páginas, o número de páginas e o número de instruções (e respetivos parâmetros) do processo.

- **processos guarda_memoria()**: Esta função guarda o processo na memória usando o algorimo de paginação, returnando o processo com os valores dos atributos **instmem**, **parammem** e **variaveis** atualizados. Recebe como argumentos o array da memória, o array das páginas, o número de páginas e o processo.

- **void liberta_espaco()**: Esta função retira o processo finalizado da memória. Recebe como argumentos o array da memória e o processo.

Durante a leitura do input, na função **main**, foi utilizada a seguinte função auxiliar:

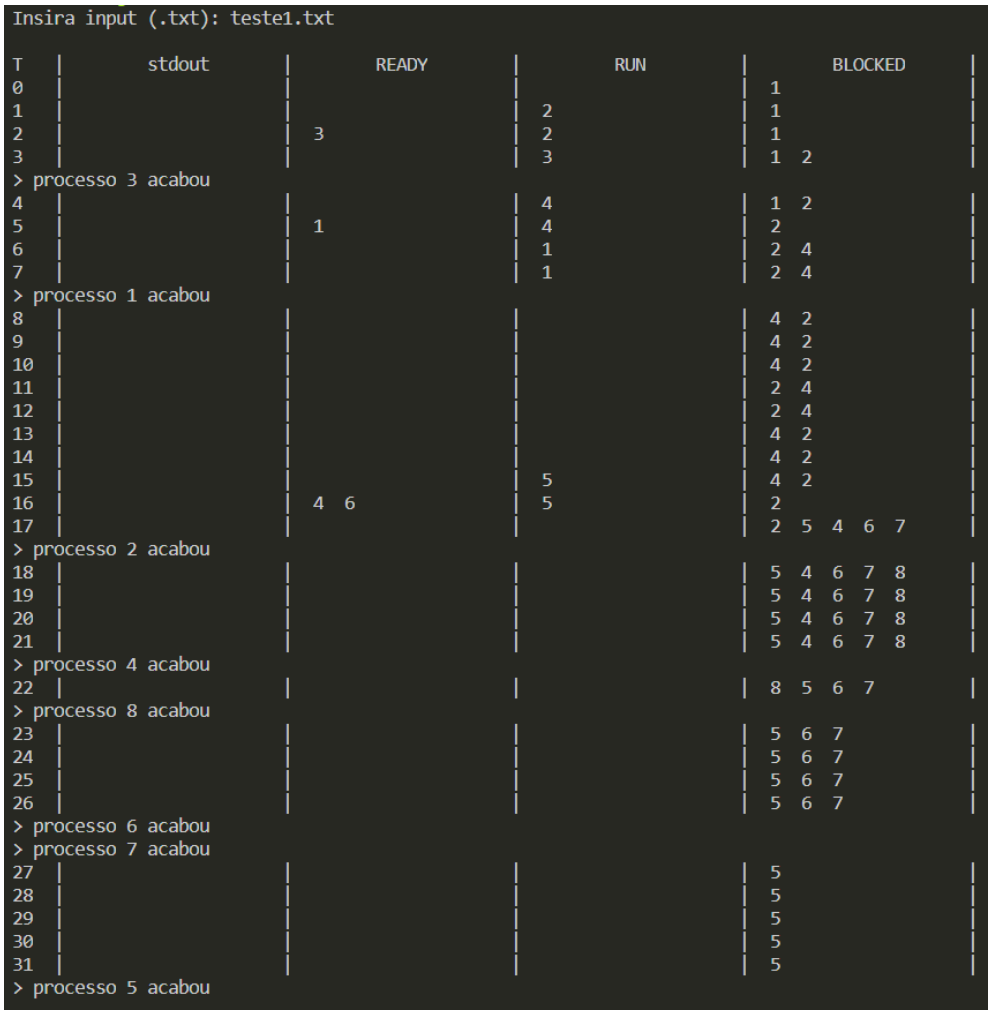
- **int codifica_instrucao()**: Esta função codifica uma instrução do processo. Recebe como argumento a string lida do ficheiro de input correspondente à instrução e retorna o número do código da instrução.

Com as funções auxiliares implementadas, partiu-se para a implementação do sistema operativo. As principais funções utilizadas foram:

- **void so_pag()**: Esta função é responsável por implementar o simulador de sistema operativo com o escalonamento *Round Robin* e com gestão de memória paginada. Recebe como argumentos o array de processos lidos do ficheiro de input, o número de processos lidos do ficheiro de input, o valor do quantum, o tamanho do array da memória e as filas run, blocked, ready e stdout.
- **int main()**: Esta função é responsável por ler o ficheiro de input, calcular o número de processos existentes no ficheiro de input, guardar os processos lidos num array de processos, definir as filas utilizadas no sistema operativo e executar o simulador de sistema operativo.

Testes

Encontram-se em baixo os outputs para os testes 1, 2, 4 e 5, respetivamente.



```
Insira input (.txt): teste1.txt

T   |        stdout       |        READY        |        RUN          |        BLOCKED      |
0   |                     |                     |                     | 1                   |
1   |                     |                     | 2                   | 1                   |
2   |                     | 3                   | 2                   | 1                   |
3   |                     |                     | 3                   | 1  2                |
> processo 3 acabou
4   |                     |                     | 4                   | 1  2                |
5   |                     | 1                   | 4                   | 2                   |
6   |                     |                     | 1                   | 2  4                |
7   |                     |                     | 1                   | 2  4                |
> processo 1 acabou
8   |                     |                     |                     | 4  2                |
9   |                     |                     |                     | 4  2                |
10  |                     |                     |                     | 4  2                |
11  |                     |                     |                     | 2  4                |
12  |                     |                     |                     | 2  4                |
13  |                     |                     |                     | 4  2                |
14  |                     |                     |                     | 4  2                |
15  |                     |                     | 5                   | 4  2                |
16  |                     | 4  6                | 5                   | 2                   |
17  |                     |                     |                     | 2  5  4  6  7       |
> processo 2 acabou
18  |                     |                     |                     | 5  4  6  7  8       |
19  |                     |                     |                     | 5  4  6  7  8       |
20  |                     |                     |                     | 5  4  6  7  8       |
21  |                     |                     |                     | 5  4  6  7  8       |
> processo 4 acabou
22  |                     |                     |                     | 8  5  6  7          |
> processo 8 acabou
23  |                     |                     |                     | 5  6  7             |
24  |                     |                     |                     | 5  6  7             |
25  |                     |                     |                     | 5  6  7             |
26  |                     |                     |                     | 5  6  7             |
> processo 6 acabou
> processo 7 acabou
27  |                     |                     |                     | 5                   |
28  |                     |                     |                     | 5                   |
29  |                     |                     |                     | 5                   |
30  |                     |                     |                     | 5                   |
31  |                     |                     |                     | 5                   |
> processo 5 acabou
```

```
Insira input (.txt): teste2.txt

T   |        stdout          |          READY          |          RUN          |          BLOCKED          |
0   |                        |  2                      |  1                    |                           |
1   |  2                     |  2                      |  1                    |                           |
2   |                        |  2  3                   |  1                    |                           |
3   |                        |  3  1  4                |  2                    |                           |
4   |  4                     |  3  1  4                |  2                    |                           |
5   |                        |  3  1  4  5             |  2                    |                           |
6   |                        |  1  4  5  2  6          |  3                    |                           |
7   |  6                     |  1  4  5  2  6          |  3                    |                           |
> fork sem sucesso
8   |                        |  1  4  5  2  6          |  3                    |                           |
9   |  3                     |  4  5  2  6  3          |  1                    |                           |
> fork sem sucesso
10  |                        |  4  5  2  6  3          |  1                    |                           |
11  |  0                     |  4  5  2  6  3          |  1                    |                           |
> fork sem sucesso
12  |                        |  5  2  6  3  1          |  4                    |                           |
13  |  0                     |  5  2  6  3  1          |  4                    |                           |
> fork sem sucesso
14  |                        |  5  2  6  3  1          |  4                    |                           |
> fork sem sucesso
15  |                        |  2  6  3  1  4          |  5                    |                           |
16  |  0                     |  2  6  3  1  4          |  5                    |                           |
> fork sem sucesso
17  |                        |  2  6  3  1  4          |  5                    |                           |
18  |  5                     |  6  3  1  4  5          |  2                    |                           |
> fork sem sucesso
19  |                        |  6  3  1  4  5          |  2                    |                           |
20  |  0                     |  6  3  1  4  5          |  2                    |                           |
> fork sem sucesso
21  |                        |  3  1  4  5  2          |  6                    |                           |
22  |  0                     |  3  1  4  5  2          |  6                    |                           |
> fork sem sucesso
```

```
23  |                        |  3  1  4  5  2          |  6                    |                           |
24  |  0                     |  1  4  5  2  6          |  3                    |                           |
> fork sem sucesso
25  |                        |  1  4  5  2  6          |  3                    |                           |
26  |  0                     |  1  4  5  2  6          |  3                    |                           |
27  |                        |  4  5  2  6  3          |                       |  1                        |
28  |  0                     |  5  2  6  3             |  4                    |  1                        |
> fork sem sucesso
29  |                        |  5  2  6  3             |  4                    |  1                        |
30  |  0                     |  5  2  6  3             |  4                    |  1                        |
31  |  0                     |  2  6  3  4             |  5                    |  1                        |
> fork sem sucesso
32  |                        |  2  6  3  4  1          |  5                    |                           |
33  |  0                     |  2  6  3  4  1          |  5                    |                           |
34  |                        |  6  3  4  1  5  2       |                       |                           |
35  |  0                     |  3  4  1  5  2          |  6                    |                           |
> fork sem sucesso
36  |                        |  3  4  1  5  2          |  6                    |                           |
37  |  0                     |  3  4  1  5  2          |  6                    |                           |
38  |                        |  4  1  5  2  6  3       |                       |                           |
39  |                        |  1  5  2  6  3  4       |                       |                           |
> processo 1 acabou
40  |                        |  5  2  6  3  4          |                       |                           |
> processo 2 acabou
41  |                        |  6  3  4  5             |                       |                           |
> processo 3 acabou
42  |                        |  4  5  6                |                       |                           |
> processo 4 acabou
43  |                        |  5  6                   |                       |                           |
> processo 5 acabou
44  |                        |  6                      |                       |                           |
> processo 6 acabou
```

```
Insira input (.txt): teste4.txt

T   |        stdout       |        READY        |        RUN          |        BLOCKED      |
0   |                     |                     |  1                  |                     |
1   |  0                  |  2                  |  1                  |                     |
> processo 1 acabou
2   |                     |  3                  |  2                  |                     |
3   |                     |  3  4               |  2                  |                     |
4   |  1                  |  3  4               |  2                  |                     |
5   |                     |  4  2               |  3                  |                     |
6   |                     |  4  2               |  3                  |                     |
7   |                     |  4  2               |  3                  |                     |
> erro de segmentacao do processo 4
8   |                     |  2  3               |  4                  |                     |
> erro de segmentacao do processo 4
9   |                     |  2  3               |  4                  |                     |
> processo 4 acabou
> erro de segmentacao do processo 2
10  |                     |  3                  |  2                  |                     |
11  |  1                  |  3                  |  2                  |                     |
> processo 2 acabou
12  |  2                  |                     |  3                  |                     |
> erro de segmentacao do processo 3
13  |                     |                     |  3                  |                     |
> processo 3 acabou
```

```
Insira input (.txt): teste5.txt

T   |        stdout        |         READY         |         RUN          |         BLOCKED         |
0   |                      |                       |                      | 1 2 3 4 5 6 7 8 |
1   |                      |                       |                      | 1 2 3 4 5 6 7 8 |
2   |                      |                       |                      | 1 2 3 4 5 6 7 8 |
3   |                      |                       |                      | 1 2 3 4 5 6 7 8 |
4   |                      |                       |                      | 1 2 3 4 5 6 7 8 |
5   |                      | 2 3 4 5 6 7 8         | 1                    |                 |
6   |                      | 3 4 5 6 7 8           | 2                    | 1               |
7   |                      | 4 5 6 7 8             | 3                    | 1 2             |
8   |                      | 5 6 7 8               | 4                    | 1 2 3           |
9   |                      | 6 7 8                 | 5                    | 1 2 3 4         |
10  |                      | 7 8                   | 6                    | 1 2 3 4 5       |
11  |                      | 7 8 1                 | 6                    | 2 3 4 5         |
12  |                      | 7 8 1 2               | 6                    | 3 4 5           |
13  |                      | 8 1 2 3 6             | 7                    | 4 5             |
14  |                      | 8 1 2 3 6 4           | 7                    | 5               |
15  |                      | 8 1 2 3 6 4 5         | 7                    |                 |
16  |                      | 1 2 3 6 4 5 7         | 8                    |                 |
17  |                      | 1 2 3 6 4 5 7         | 8                    |                 |
18  |                      | 1 2 3 6 4 5 7         | 8                    |                 |
19  |                      | 2 3 6 4 5 7 8         | 1                    |                 |
> processo 1 acabou
20  |                      | 3 6 4 5 7 8           | 2                    |                 |
> processo 2 acabou
21  |                      | 6 4 5 7 8             | 3                    |                 |
> processo 3 acabou
22  |                      | 4 5 7 8               | 6                    |                 |
23  |                      | 4 5 7 8               | 6                    |                 |
24  |                      | 4 5 7 8               | 6                    |                 |
25  |                      | 5 7 8 6               | 4                    |                 |
> processo 4 acabou
26  |                      | 7 8 6                 | 5                    |                 |
> processo 5 acabou
27  |                      | 8 6                   | 7                    |                 |
28  |                      | 8 6                   | 7                    |                 |
29  |                      | 8 6                   | 7                    |                 |
30  |                      | 6 7                   | 8                    |                 |
31  |                      | 6 7                   | 8                    |                 |
32  |                      | 6 7                   | 8                    |                 |
33  |                      | 7 8                   |                      | 6               |
34  |                      | 8                     |                      | 6 7             |
35  |                      |                       |                      | 6 7 8           |
36  |                      |                       |                      | 6 7 8           |
37  |                      |                       |                      | 6 7 8           |
38  |                      |                       | 6                    | 7 8             |
> processo 6 acabou
39  |                      |                       | 7                    | 8               |
> processo 7 acabou
40  |                      |                       | 8                    |                 |
41  |                      |                       | 8                    |                 |
42  |                      |                       | 8                    |                 |
43  |                      |                       | 8                    |                 |
44  |                      |                       | 8                    |                 |
45  |                      |                       | 8                    |                 |
> processo 8 acabou
```