Relatório Sistemas Operativos
Trabalho 1
2021/2022

Trabalho realizado por:
António Nanita nº 48407
Diogo Carreiro nº 48729
Rodrigo Alves nº 48681

Introdução

Neste trabalho pretende se implementar, usando a linguagem C, um simulador de Sistema Operativo, considerando um modelo de 5 estados. Este modelo contém os seguintes estados: NEW, READY, RUNNING, BLOCKED e EXIT. O objetivo deste trabalho é aplicar a um determinado input dois algoritmos de escalonamento diferentes, First Come First Served (FCFS) e *Round Robin* (Q=3). O input irá conter o tempo em que o processo entrou no sistema operativo, como também os tempos de permanência nos estados RUN e BLOCKED.
No nosso trabalho utilizámos a estrutura de dados *Queue*, para ajudar no escalonamento dos processos, sendo que foram criadas duas *Queues*, uma para guardar os processos que estavam no estado READY, e outra para guardar os processos no estado BLOCKED.
Também criámos uma *Struct* que vai guardar a informação acerca de cada processo, isto é, o estado em que se encontra (state), o instante em que o processo entrou (start) e os tempos em que o processo se encontra no estado run e no estado blocked (all_times). Esta informação é guardada num *array* de inteiros. Por último também guarda o processo que está a ser corrido (CPU), também é guardado um contador, que vai ser usado no escalonamento Round Robin (Q). O grupo decidiu que cada estado é representado por um inteiro, para facilitar tanto o código, como a implementação do algoritmo em linguagem C.

Descrição do Programa

No nosso código foram criadas as seguintes funções:
all_times_each_process- esta função a partir do input introduzido irá preencher o *array all_times* com todos os tempos de RUNNING e BLOCKED para cada processo. Caso uma dada posição do *array* contenha o número “0”, esta vai ser colocada no *array* como sendo “-1”.
start_process- nesta função o valor da variável *start* vai ser atualizado, para indicar qual o tempo de início de cada processo.
STATE_NEW- esta função indica se o instante que está a ser executado é igual ao instante que o processo entra no sistema operativo (variável *start*).
READY_TO_RUN- serve para indicar se existe algum processo a ser corrido no CPU.
ALL_EXIT- indica se todos os processos já se encontram no estado exit.
line_time_RR- função que vai retornar a linha referente ao instante que está a ser executado, e que irá conter o estado em que se encontra cada processo na mesma, usando o algoritmo de escalonamento Round Robin (com um determinado quantum), neste caso sendo o quantum igual a 3, como estava referido no enunciado.
line_time_FCFS- função que vai retornar a linha referente ao instante que está a ser executado, e que irá conter o estado em que se encontra cada processo na mesma, usando o algoritmo de escalonamento First Come First Served.

Descrição dos Algoritmos de Escalonamento

First Come First Served (FCFS)

Para este algoritmo vamos verificar o estado em que se encontra cada processo num determinado instante. Em primeiro lugar averiguamos se o instante que está a ser executado é igual ao instante em que um processo se inicia, para isso utilizamos a função STATE_NEW. Caso a função retorne um inteiro “1” é introduzido na linha do instante o estado NEW e posteriormente este irá ser adicionado à Queue Ready e passará a estar no estado READY. Caso o instante não seja igual ao instante em que o processo se inicia e o instante que está a ser executado seja menor do que o instante em que o processo é iniciado é adicionado um espaço em branco, pois o processo ainda não foi iniciado. Caso contrário, iremos proceder aos seguintes testes: primeiro averiguamos se o inteiro que está na posição do *array all_times* do processo é igual a zero, se for verdade e a posição do processo for divisível por dois, iremos retirar o elemento que está à frente da fila “Ready” e, de seguida iremos adicionar esse mesmo processo à Queue Block, consequentemente o processo passará a estar no estado blocked (BLCK) e, por último o processo que estava a ser corrido será retirado do CPU. Caso a posição em que o processo se encontra no *array all_times* não seja divisível por dois, o processo será retirado da fila Block, e

consequentemente irá ser adicionado à Queue Ready e o processo passará a estar no estado ready (READY), por último será atualizada a posição do processo, que foi incrementada.
De seguida averiguamos se na posição em que se encontra o processo no *array all_times* é diferente de “-1”. Se for igual a “-1” e o processo já se encontrar no estado exit (EXIT) é adicionado um espaço em branco à linha, mas se o estado em que o processo se encontra for diferente de exit (EXIT), o processo passará a estar no estado exit (EXIT) e é adicionado à linha a palavra EXIT.
No caso em que este numero é diferente de “-1” iremos verificar se existe algum processo a ser corrido (função READY_TO_RUN), caso isso não se verifique e o processo atual que está a ser avaliado esteja no inicio da Queue Ready será atualizado o processo que está a ser corrido no CPU, e posteriormente será introduzido na linha que o processo está no estado run (RUN) e será decrementado o inteiro da posição em que o processo se encontra no *array all_times*. Caso o processo que está no CPU for igual ao processo atual será introduzido na linha que o processo está no estado run (RUN) e será decrementado o inteiro da posição em que o processo se encontra no *array all_times*. Se o estado do processo se encontrar no estado ready (READY) e ao mesmo tempo este processo for diferente ao que se encontra no CPU ou então o processo se encontrar no estado blocked (BLCK) e ao mesmo tempo o início da Queue Block for diferente ao processo será introduzido na linha que o processo está no estado ready (READY). Por último, se o processo estiver no início da Queue Block irá ser inserido na linha que o processo está no estado blocked (BLCK).

Round Robin (Q=3)

Em primeiro lugar neste algoritmo averiguamos se no instante em que está a ser averiguado é igual ao instante em que o processo se inicia, se for o caso
é escrito na linha que o processo está em new (NEW), sendo que o processo será adicionado à Queue Ready e o estado do processo será ready(READY).
Caso o instante for diferente do instante que inicial e maior que o instante atual será adicionado um espaço em branco à linha. Se nenhuma das condições anteriores não se verificar será feito feita a averiguação se o contador do processo é menor que o Quantum recebido do parâmetro, se for verdade, primeiro averiguamos se o inteiro que está na posição do *array all_times* do processo é igual a zero, se for verdade e a posição do processo for divisivel por dois, iremos retirar o elemento que está à frente da fila “Ready” e, de seguida iremos adicionar esse mesmo processo à Queue Block, consequentemente o processo passará a estar no estado blocked (BLCK) e, por ultimo o processo que estava a ser corrido será retirado do CPU. Caso a posição em que o processo se encontra no *array all_times* não seja divisível por dois, o processo será retirado da fila Block, e consequentemente irá ser adicionado à Queue Ready e o processo passará a estar no estado ready (READY), por último será atualizada a posição do processo, que foi incrementada e o contador do processo será igual a "0".
Posteriormente averiguamos se na posição em que se encontra o processo no *array all_times* é diferente de “-1”. Se for igual a “-1” e o processo já se encontrar no estado exit (EXIT) é adicionado um espaço em branco à linha, mas se o estado em que o processo se encontra for diferente de exit (EXIT), o processo passará a estar no estado exit (EXIT) e é adicionado à linha a palavra EXIT.
Caso o número que se encontra na posição do *array all_times* do processo for diferente que "-1" serão feitos os seguintes testes:
Verificar se existe algum processo a ser corrido (função READY_TO_RUN), caso isso não se verifique e o processo atual que está a ser avaliado esteja no inicio da Queue Ready será atualizado o processo que está a ser corrido no CPU, e posteriormente será introduzido na linha que o processo está no estado run (RUN) e será decrementado o inteiro da posição em que o processo se encontra no *array all_times* e incrementamos o contador do processo;
Caso o processo que está no CPU for igual ao processo atual será introduzido na linha que o processo está no estado run (RUN) e será decrementado o inteiro da posição em que o processo se encontra no array all_times e sera incrementado o contador do processo. Se o estado do processo for ready (READY) e ao mesmo tempo este processo for diferente ao que se encontra no CPU ou então o processo se encontrar no estado blocked (BLCK) e ao mesmo tempo o início da fila Block for diferente ao processo será introduzido na linha que o processo está no estado ready (READY). Por último, se o processo estiver no início da Queue Block irá ser inserido na linha que o processo está no estado blocked (BLCK) e é incrementado o contador do processo.
Se o contador nao for menor que o Quantum será feito os seguintes testes:
Se o estado do processo for run (RUN) o contador vai ser igual a "0" e o processo passará para

o estado ready (READY), o CPU Não terá nenhum processo a ser corrido, será retirado o processo da Queue Ready e de seguida será introduzido à Queue Ready. Caso o estado do processo nao for run (RUN) o contador vai ser igual a "0" e o processo passará para o estado block (BLCK), será retirado o processo da Queue Block e de seguida será introduzido à Queue Block.

instante	proc1	proc2	proc3
1	NEW		
2	RUN	NEW	
3	RUN	READY	
4	RUN	READY	NEW
5	BLCK	RUN	READY
6	READY	RUN	READY
7	READY	RUN	READY
8	READY	RUN	READY
9	READY	BLCK	RUN
10	READY	BLCK	RUN
11	READY	READY	BLCK
12	RUN	READY	READY
13	RUN	READY	READY
14	BLCK	RUN	READY
15	BLCK	RUN	READY
16	READY	RUN	READY
17	READY	RUN	READY
18	READY	BLCK	RUN
19	READY	READY	RUN
20	READY	READY	RUN
21	READY	READY	RUN
22	READY	READY	RUN
23	READY	READY	RUN
24	READY	READY	BLCK
25	RUN	READY	READY
26	RUN	READY	READY
27	RUN	READY	READY
28	RUN	READY	READY
29	EXIT	RUN	READY
30		EXIT	RUN
31			RUN
32			RUN
33			BLCK
34			RUN
35			EXIT

instante	proc1	proc2	proc3
1	NEW		
2	RUN	NEW	
3	RUN	READY	
4	RUN	READY	NEW
5	BLCK	RUN	READY
6	READY	RUN	READY
7	READY	RUN	READY
8	READY	READY	RUN
9	READY	READY	RUN
10	READY	READY	BLCK
11	RUN	READY	READY
12	RUN	READY	READY
13	BLCK	RUN	READY
14	BLCK	READY	RUN
15	READY	BLCK	RUN
16	READY	BLCK	RUN
17	READY	READY	READY
18	RUN	READY	READY
19	RUN	READY	READY
20	RUN	READY	READY
21	READY	RUN	READY
22	READY	RUN	READY
23	READY	RUN	READY
24	READY	READY	RUN
25	READY	READY	RUN
26	READY	READY	RUN
27	READY	READY	BLCK
28	RUN	READY	READY
29	EXIT	RUN	READY
30		READY	RUN
31		READY	RUN
32		READY	RUN
33		READY	BLCK
34		BLCK	RUN
35		READY	EXIT
36		RUN	
37		EXIT	