Simulador Sistema Operativo 2º Trabalho

Trabalho realizado por:

Rafael Silva 45813

Docente: Luís Rato

- Objetivo:

Pretende-se implementar um simulador de Sistema Operativo com um modelo de 5 estados (New, Ready , Running , Blocked , Exit) que simula programas constituídos por um conjunto instruções .

- Desenvolvimento:

Em primeiro lugar , o programa recebe como input um ficheiro de tipo "txt" com o nome "input.txt" . O programa ao receber o mesmo , vai criar uma matriz[NumeroProcessos][Máximo de instruções] e atribuir os respetivos valores . Por exemplo, um input de texto : INI 00 , ZER 100 , PRT 00 , HLT 99 , INI 01 , ZER 50 , CPY 5 , HLT -99 ,vai criar uma matriz :
programas[1][7] = {0,3,0,100,8,0,9,99} , {1,3,0,50,1,5,9,-99} .

Após ser criada a matriz correspondente ao input , o programa vai primeiro colocar os processos que contêm 0 instantes de espera diretamente no running caso não exista nenhum programa a correr , caso exista será colocado na fila de espera Ready . Os restantes processos que têm mais tempo de espera serão colocados na fila de espera New . Para criar um processo , o programa calcula primeiro o valor maximo de var_x , que ficará guardado numa variável , por exemplo , no exemplo anterior o primeiro processo so vai precisar de um espaço em memória que será o var_0 e mais (3 * 2) para as instruções , o segundo processo já vai necessitar de 5 espaços em memória para as variáveis , pois vai precisar da variável var_5 e mais (3*2) para as instruções . Neste programa temos um array "posições" que guarda onde está a informação de cada processo , por exemplo ,

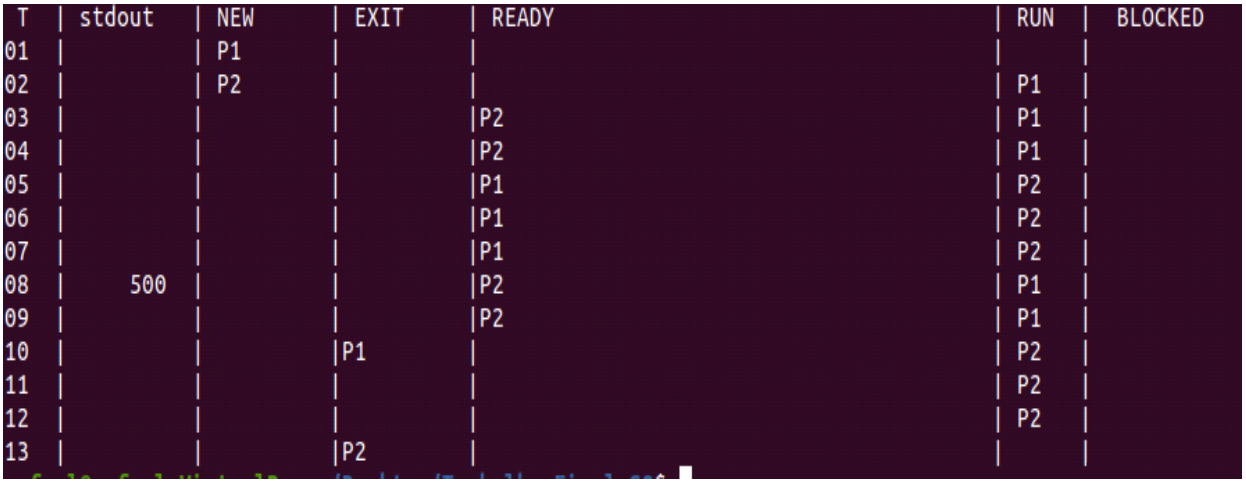
no primeiro processo posições[0] = 0 ,posições[1] = 0 , posições[2] = 1, posições[3] = 6 e com o segundo processo posições[4] = 7 , posições[5] = 11 , posições[6] = 12 , posições[7] = 17 , e para encontrar informação relativamente a um processo utiliza-se posições[4 * NumeroDoProcesso - 1) + 1 ou +2 ou +3] , por exemplo se quiser saber em que posição de mem[] está a primeira instrução do primeiro processo será mem[posições[4 * 0 + 2]]. O programa corre de processo em processo e existem 5 if/else if em que um processo pode entrar ,sendo que entra só em um . Sendo esses , caso o processo esteja para correr , seja estar no blocked ,seja estar a decrementar de new , seja a inicialização de um novo processo ou seja estar a sair do programa. Quando um processo está a correr é utilizado um switch(código) em que o código corresponde ao código da instrução e dependendo desse código o programa executará a instrução .

Existem duas funções muito importantes para o funcionamento deste programa sendo uma delas a função bestfit() que têm como objetivo gerir a memória, essa função começa por verificar as posições vazias em memória , guardando-as de 2 em 2 num array , quando todos os conjuntos de posições vazias estejam no array , ocorre a verificação de todos esses conjuntos e encontra o espaço mais pequeno e que é possível alocar , retornando a primeira posição que se pode alocar. Caso retorne -5 significa que não há espaço suficiente para alocar . Outra função é a função frk(fork) que têm como objetivo duplicar um processo , esta função começa por averiguar qual o número do processo novo tendo em conta os processos que já existem e os que já terminaram, depois utiliza a função bestfit para saber se existe espaço suficiente para alocar o novo processo e se houver onde o alocar . Copia a informação do processo pai para o processo filho tendo este as suas próprias variáveis , mas utiliza as instruções do processo pai . Retorna -5 caso não haja espaço ou o numero do novo processo.

- Input de teste:

Ficheiro "input.txt" :
INI 0
ZER 500
CPY 13
CPY 7
PRT 13
HLT -999
INI 1
ZER 100
CPY 5
DEC 5
JFW 3
HLT 99
PRT 5
JBK 2

Output:



T	stdout	NEW	EXIT	READY	RUN	BLOCKED
01		P1				
02		P2			P1	
03				P2	P1	
04				P2	P1	
05				P1	P2	
06				P1	P2	
07				P1	P2	
08	500			P2	P1	
09				P2	P1	
10			P1		P2	
11					P2	
12					P2	
13			P2			