Relatório Trabalho Final

Sistemas Operativos I

Professor:
Luís Rato
Nuno Miranda

Trabalho elaborado por:

Fernando Prates nº15407
Miguel Azevedo nº36975
Miguel Portugal nº38128

Desenvolvimento

Neste trabalho começamos por iniciar os cinco estados diferentes (*new, ready, run, blocked e exit*) sendo cada um destes uma *queue .* Posto isto, iniciamos também a memoria em que os 10 primeiros espaços irão conter as variáveis partilhadas entre todos os processos e o restante espaço irá estar as instruções dos processos.

Começamos por abrir um ficheiro de input, que irá conter os processos para o programa. O instante de entrada é lido (primeiro elemento do ficheiro) e após isso irá ser guardado num *buffer* todas as suas instruções. O processo aguarda no buffer até ser o seu instante de entrada no programa. Quando for o seu instante de entrada, as instruções são carregadas para memoria, e o processo entra no estado *NEW* onde irá ficar 1 ciclo de relógio.

Após não existirem mais processos para serem lidos, delimitamos um ciclo para continuar garantir que todos os processos terminam as suas instruções.

Funções

*int *inicioMEM (int *pag)* -> esta função irá receber um *array* de inteiros, que representam as páginas ocupadas da memoria. A função irá retornar um *array* de inteiros de 300 posições, sendo elas as 10 primeiro as para variáveis (partilhadas pelos processos) e as restantes as instruções de cada processo.

*void print_MEM(int *array, int *pag)* -> esta função irá receber a memória e as páginas ocupadas da memória, e irá fazer um print no terminal e em ficheiro do *output*.

*void to_MEM(int *MEM, pcb *temp,int *buffer, int nr_pag, int *paginas)* -> esta função irá receber o *array* memória, o processo (*pcb*) que irá ser inserido na memoria, um buffer que irá conter as instruções do processo, o numero de paginas ocupadas no processo e o *array* de páginas ocupadas pelo programa. A função percorre todas as posições da memória ate encontrar espaço para inserir o processo.

*void print_instante (int *MEM, int instante)* -> Esta função recebe a memória e o instante, percorre todos os processos printando os estados em que eles se encontram.

*void cicle(int *MEM)* -> Esta função começa por ler o processo que esta no RUN, lendo e guardando a próxima instrução a ser executada. Retorn 0 se a instrução a ser executada for um fork e 1 caso contrario.

*void cicle(int *MEM)* -> Esta função lida com as instruções do estado blocked, que são executadas após o processo estar 3 ciclos no blocked.

*void cicle(int *MEM)* -> Esta funções começa por ver se o estado blocked esta ou não vazio, caso não esteja verifica se estes processos já lá estão a três ciclos e lida com eles ou incrementa o blocked_counter. Após isto lida com o estado RUN, verifica se está vazio, caso não esteja, verifica se realizou o QUANTUM ou se existe programCounter fora do espaço reservado pelo processo. Incrementa o run_counter do processo. Após isto chama a função admin() e dispatch() para o caso de haver um processo no NEW ou no READY a precisar de ir para o RUN.

Int main() -> Esta função contem três ciclos, começa por um ciclo para ir lendo os processos dos ficheiros e outro para esperar para entrar no seu instante. O terceiro ciclo é necessário para quando não houver mais processos para ler e para o programa continuar a correr. Na main começamos por ler o processo do ficheiro para um buffer, e calcular o espaço necessário em memoria para ele. Após isto entramos noutro ciclo que vai correndo até ser o instante do processo entrar. Quando o processo está no instante de entrar este é carregado para a memória chamando a função to_MEM e de seguida chama o resto as funções necessárias ao funcionamento de cada ciclo dos cinco estados. O ultimo ciclo só acaba quando todos os 5 estados estiverem vazios terminando assim o programa.