Relatório
Sistemas Operativos
Trabalho Final

Modelo de 5 estados
com Gestão de Memória

Trabalho realizado por:
- André Figueira, 31626
- Ricardo Benedito, 31643

Introdução

Com este trabalhamos pretendemos implementar um modelo funcional de 5 de estados, juntamente com gestão de memória. Usaremos 3 classes.

Class PCB
Class Scheduler
Class Main

E utilizaremos o conhecimento adquirido, nas aulas práticas e aulas teóricas para conseguir realizar este trabalho.

Classe PCB

Esta classe representa um processo incluíndo toda a sua informação.
A informacao de cada processo é:

- o id;
- o estado;
- suas instrucoes;
- paginas ocupadas;
- program counter;
- instante em que foi criado;

O “id” serve simplesmente para indicar qual o “numero” do processo

O estado indica nos o estado actual de cada processo. Quando o processo é criado o estado é “new”.

As paginas ocupadas indica o numero de paginas que o processo ocupa, as paginas ocupadas pelo processo dependem do numero de instruções de cada processo, sendo uma pagina por cada 10 instruções.

O program counter serve para nos indicar qual a instrução que está ser executada de momento.

Nota:

Como indicado, assumiu-se:

"A transição para o estado ‘ready’ faz-se verificando os processos que se encontram nos estados

‘blocked’, ‘new’ e ‘running’ (por esta ordem)."

Classe Scheduler

Este classe é composta por uma série de listas, cada uma representando os estados de um modelo de cinco estado à excepção do estado Run.

Quando um processo é criado, o scheduler transfere esse processo para a lista NEW onde aguarda pela sua vez, visto que as listas tem o comportamento FIFO.

O scheduler começa por verificar por uma certa ordem os processos que vão para a lista READY.
Começando por verificar os processos bloqueados. Se o disco estiver a ser utilizado (IO_LOCKED = TRUE) então não se enviam processos para a lista READY. Isto porque um processo bloqueado irá esperar pela sua vez de utilizar o disco e portanto não será trazido para a lista READY para ser bloqueado outra vez pois o disco ainda está a ser utilizado. Se o disco não estiver a ser utilizado (IO_LOCKED = FALSE) então o processo bloqueado a aguardar a mais tempo passa a ocupar o IO_BLOCKED e é transferido para a lista READY, e consequentemento altera-se o IO_LOCKED para true. Garante-se assim que so haja um processo a ocupar o disco de cada vez.

Depois de verificar os processos bloqueados verifica a lista NEW. O Scheduler verifica inicialimente se existem menos que X processos (default = 10) no sistema. Se houver, então o scheduler verifica se existem recursos suficientes para o processo (1 pagina por cada 10 instrucoes) . Se não existirem, o processo aguarda a sua vez até haver. Caso contrário, decrementa-se o numero de paginas ocupadas em PAGINAS_DISPONIVEIS e o processo é enviado para a lista READY.

A ultima verificação é o processo que esteve no estado RUN, antes de fazer a decisão de enviar para a lista READY verifica-se a instrucao que executou foi a ultima instrução. Se sim envia para a lista EXIT pois já terminou. Caso contrário, envia-se para a lista READY pois so pode estar no estado RUN um ciclo (quantum = 1).

Se o RUN for o IO_BLOCKED e for a ultima instrucao, então liberta-se chama-se newIOBlocked() para trazer um novo processo ocupar o disco, e altera-se o lock, e o processo passa para a lista EXIT, se não houver processos na lista BLOCKED o processo sai pois não tem nada a fazer. No fim liberta as paginas ocupadas. Caso não seja a ultima instrução, é interrompido pois so pode estar no estado RUN um ciclo (quantum = 1)

O próximo passo no scheduler é passar um processo para o estado RUN. Neste caso o primeiro passo a verificar é se a instrução actual do processo a aguardar é 3 ou seja um NO OPERATION. Neste caso não faz sentido trazer para processador pois não executada nada. Portanto incrementa-se o pc e envia-se para o fim da fila. Caso a instrucao actual do proximo processo seja a mesma. Então volta-se a repetir o mesmo

passo. Se por acaso for a ultima instrução então é removido. Após este ciclo, verifica-se se a lista READY ainda tem processos, caso tenha, envia-se o que está a aguardar a mais tempo para o estado RUN

No estado RUN:

Se for 0, o programa mete o program counter igual ao tamanho da lista para que no proximo instante, o scheduler reconheça que esse processo esta na sua ultima instrução e portanto envia-o para a lista EXIT.

Se for 1, o scheduler simplesmente incrementa o program counter.

Se for 2, o scheduler bloqueia o processo, e mete o disco locked (altera a flag de IO_LOCKED) significante que o disco está a ser usado por um processo e esse processso passa a ser IO_BLOCKED. Caso o disco esteja locked quando é executado esta instrução, este é enviado para a lista BLOCKED onde aguarda a sua vez de utilizar o disco.

Se for 3, o scheduler não faz nada, esta verificação é feita quando o scheduler verifica qual o proximo processo a ocupar o CPU, isto para não ficar um instante sem fazer nada. Neste caso é enviado para o fim da fila READY e incrementa-se o program counter.

Se for 4, o scheduler faz reset ao program counter.

Se for 5, o scheduler executa um fork sobre o processo. Este fork consiste em copiar as instruções do processo pai, e envia lo para a lista READY.

NOTA:
Qualquer instrução excepto DISK, é feita uma verificação. Verifica-se se o processo RUN e igual ao processo IO_BLOCKED, pois, se esta condição verificar então significa que o lock do disco pode ser alterado (IO_LOCKED = FALSE)

Classe Main

Esta classe é a principal, visto que é esta que contém o command prompt e o metodo main.

Na main desta classe, inicializa-se as flags, GOT_INPUT e CYCLE

GOT_INPUT é a flag que é alterada na thread COMMAND_PROMPT quando há um ficheiro novo. Esta flag permite com que o mesmo ficheiro não seja inserido enquanto se está à espera de input.

CYCLE é a flag que é alterada pelo COMMAND_PROMPT quando o utilizador deixa de crer introduzir input e portanto, termina o programa, incluindo a thread.

Antes da thread ser inicializada é pedido ao utilizador que introduza o numero maximo de processos a correr e o numero maximo de paginas disponiveis. Se o utilizador introduzir -1 numa destas opções então inicializa-se o scheduler com valores default no que o utilizador introduziu -1.

Quando a thread se inicializa entra em ciclo. O scheduler não começa a funcionar até ser introduzido pelo menos um programa. Nesta thread, só está presente um Scanner, pois assim evitou-se o bloqueio de output pelo scheduler. Nesta thread altera-se o valor de String FICHEIRO, que é lido por READ_INPUT(FICHEIRO).

Quando o programa lê o ficheiro, verifica a existência da instrução 0 (EXIT), se encontrar, ignora, copia as instrucoes para um array e cria um novo pcb que vai ser adicionado a lista NEW. Se não encontrar, adiciona a instrucao ao fim do array e segue o procedimento normal.

Assim que é introduzido um programa a thread altera a flag GOT_INPUT e a String FICHEIRO e então o scheduler pode começar a funcionar. Enquanto a thread espera por input, o scheduler não para de produzir output (escreve para um ficheiro scheduler.out após cada ciclo) .

Se for inserido um programa incorrecto, o scheduler continua a funcionar normalmente mas é escrito no ficheiro scheduler.out uma mensagem de erro. E a thread volta a pedir input.

Quando o utilizador quiser terminar toda o programa, basta introduzir stop, e a thread altera as flags para terminar o ciclo. E espera pela thread terminar.

Em cada ciclo o scheduler verifica se todos os estados estão no estado EXIT. Se sim, o programa termina. Caso contrário continua.