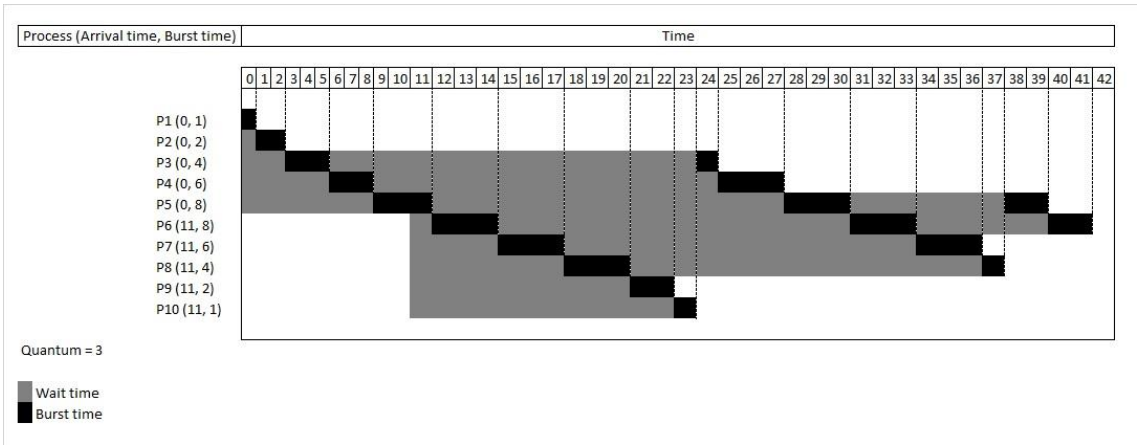
Universidade de Évora

Curso de Engenharia Informática

Sistemas Operativos 2019/2020

Trabalho 1 – Paginação e Particionamento Dinâmico BEST FIT



Trabalho realizado por:

Dinis Matos nº42738

José Lopes nº37861

Introdução

Este trabalho foi realizado com o objetivo de implementar um simulador de um Sistema Operativo, usando o modelo de 3 estados e implementando memória com Paginação e Particionamento Dinâmico BEST FIT.

Estrutura do trabalho

Implementação de filas

Para a execução do programa é feita uma implementação de filas para depois ser utilizada para o escalonamento.

Esta implementação tem uma estrutura de fila, cujo nome é *struct Queue*, tem um método que cria um estrutura de fila nova, um método boleano que diz se a fila está cheia, outra que diz se está vazia. Também tem um método de retorna o tamanho atual da fila, outro que coloca um valor no fim da fila e mais um que retira e retorna o primeiro valor da fila. Por fim, tem duas funções. Uma que retorna o início da fila e outra que retorna o fim da fila.

Implementação de double linked lists

Cada node da linked list vai corresponder a um processo, e dentro desse node vai estar uma struct com várias informações sobre esse processo. Também é possível existirem nodes “vazios” que simbolizam memória disponível.

Esta implementação tem um método que muda o conteúdo no node, um método que o apaga, um método para inserir um novo node no início da linked list, depois de um certo node e no final. Tem também um método para inserir um processo, para remover um processo e para imprimir o conteúdo da linked list. Depois tem também um método que retorna o

espaço total disponível em memória (através da contagem de nodes "vazios") e um método que encontra e retorna um node em específico.

Esta implementação é utilizada exclusivamente quando o programa está a utilizar a implementação em memória com Particionamento Dinâmico BEST FIT.

Includes e Defines

É declarado inicialmente as bibliotecas "stdio.h", "stdlib.h", "string.h", "stdbool.h", a implementação de filas, a implementação de double linked lists e oito variáveis iniciais, MEMSIZE (Tamanho do array mem), PAGINA (Tamanho de cada Página), PAGASSIST (Tamanho do array auxiliar pagingAssist), MAXPROCESSOS (Tamanho da quantidade máxima de processos), QUANTUM (Valor do Quantum), SIZE (Tamanho da quantidade máxima de cada linha), NOMEFICHEIRO (Nome do ficheiro a testar) e MODO (Nome do modo a utilizar). Depois declaramos os includes "IO.c", "GlobalFunctions.c", "MemoryBest.c" e "MemoryPaging.c".

Structs

A estrutura "pagingElements" é inicializada com os seguintes argumentos: processo (número do processo), nextPage (próxima página do processo), backPage (página anterior do processo), lineToRead (linha a ser executada), terminou (saber se página terminou) e variáveis (saber se página contém as variáveis do processo). Esta estrutura é utilizada na memória com Paginação.

Métodos Globais

Estes vão ser os métodos usados quer na implementação com memória com paginação quer na implementação BEST FIT.

- **startIntArray**

  Este método vai inicializar as posições do array recebido com o valor integer '0'.

- **NextINI**

  Este método vai calcular o índice da próxima instrução "INI" no array "inputsave" e vai guardar esse valor no pointer "*INIAtual".

- **ready_run**

  Este método vai inserir no estado "Run" o primeiro processo no estado "Ready".

- **prints**

  Este método vai imprimir o instante da execução, o "stdout" e o conteúdo dos 3 estados ("Ready, Run e Blocked") em cada instante da execução do Sistema Operativo.

Métodos de I/O

Estes métodos estão todos relacionados com a leitura de ficheiro e tratamento de input.

- **contalinhas**

  Este método tem o simples objetivo de contar o número de linhas no ficheiro de input.

- **lerinput**

  Este método vai ler o input linha a linha, e vai guardar no array "inputsave" um valor específico que corresponde à instrução lida. Também vai guardar o valor da variável que essa instrução possui.

- **openfileteste**

  Este método vai simplesmente abrir o ficheiro que corresponde ao nome recebido como argumento.

Métodos de BEST FIT

Estes métodos vão ser usados exclusivamente quando o Sistema Operativo estiver a utilizar a implementação de memória com Particionamento Dinâmico BEST FIT.

- **ColocarNoMemBest**

  Este método vai colocar o conteúdo do processo recebido no array principal “mem[]”.

- **MemToMemBest**

  Este método auxilia a instrução “FRK”, copia o processo e coloca um semelhante (em que certos valores são trocados) no array “mem[]”.

- **entrarProcessoBest**

  Método que coloca um novo processo (linkedlist e queue) se o instante do processo corresponder ao instante atual.

- **blocked_readyBest**

  Método que vai passar (se possível) um processo do estado “Blocked” para o estado “Ready”.

- **Métodos executeINSTRUÇÃO**

  Sequência de métodos em que a sua função é executar a instrução que está no nome do próprio método.

- **checkFRKBest**

  Este método verifica se é possível a realização da instrução “FRK” sendo o fator limitante o espaço na memória.

- **checkJumpLimitFowardBest**

  Este método verifica se é possível executar o “Jump Foward” verificando se não ultrapassa os limites do próprio processo.

- **checkJizFowardBest**

  Esta verificação vai assegurar que se ao ser executada a instrução “JIZ” não vão ser ultrapassados os limites do processo atual.

- **checkJumpLimitBackBest**

  Este método verifica se é possível executar o “Jump Back” verificando se não ultrapassa os limites do próprio processo.

- **checkVarAccessBest**

  Este método verifica que o acesso à posição da variável requerida não tenta aceder posições que não estão designadas estritamente para variáveis.

- **checkLimitRangeBest**

  Esta verificação vai assegurar que a instrução atual não está fora dos limites do próprio processo.

- **checkExceptionBest**

  Método principal de verificações. Mediante da instrução recebida, vai encaminhar os checks necessários a serem realizados para essa instrução.

- **executeRunBeforeBest**

  Este método vai ler a instrução que tem que ser executada e vai executar o método que corresponde a essa mesma instrução. Tem a particularidade de executar as instruções antes da execução do método "prints" em cada ciclo.

- **executeRunAfterBest**

  Este método vai ler a instrução que tem que ser executada e vai executar o método que corresponde a essa mesma instrução. Tem a particularidade de executar as instruções depois da execução do método "prints" em cada ciclo.

- **executionBest**

  Método que executa o programa a partir do algoritmo BEST FIT.

Métodos de Paginação

Estes métodos vão ser usados exclusivamente quando o Sistema Operativo estiver a utilizar a implementação de memória com Paginação.

Esta implementação vai usar um array auxiliar, em que cada posição simboliza um processo. Cada posição vai ter uma struct com informações sobre esse mesmo processo e as páginas lá contidas.

- **nextLinePaging**

  Método que passa para a próxima linha do processo atual.

- **inicializePaging**

  Método que inicializa o array auxiliar “pagingAssist”.

- **countSpacePaging**

  Este método retorna o espaço livre do pagingAssist.

- **variablesPaging**

  Método que retorna a página onde estão situadas as variáveis do processo.

- **blocked_readyPaging**

  Método que coloca um processo do estado “Blocked” para o estado “Ready” se o tempo do estado “Blocked” acabou.

- **Métodos executeINSTRUÇÃO**

  Sequência de métodos em que a sua função é executar a instrução que está no nome do próprio método.

- **removerProcessoPaging**

  Este método tem a simples função de remover um processo.

- **checkFRKPaging**

  Método que verifica se a instrução “FRK” é possível ser executada.

- **checkJumpLimitFowardPaging**

  Este método verifica se é possível executar o “Jump Back” verificando se não ultrapassa os limites do próprio processo.

- **checkJizFowardPaging**

  Esta verificação vai assegurar que se ao ser executada a instrução “JIZ” não vão ser ultrapassados os limites do processo atual.

- **checkJumpLimitBackPaging**

  Este método verifica se é possível executar o “Jump Back” verificando se não ultrapassa os limites do próprio processo.

- **checkVarAccessPaging**

  Este método verifica que o acesso à posição da variável requerida não tenta aceder posições que não estão designadas estritamente para variáveis.

- **checkLimitRangePaging**

  Método que verifica se a instrução pertence ao processo atual.

- **findPaginaAtual**

  Método que verifica se a instrução “FRK” é possível ser executada.

- **checkExceptionPaging**

  Este método executa a verificação que corresponde instrução que vai ser possivelmente executada.

- **executeRunBeforePaging**

  Este método vai ler a instrução que tem que ser executada e vai executar o método que corresponde a essa mesma instrução. Tem a particularidade de executar as instruções antes da execução do método “prints” em cada ciclo.

- **executeRunAfterPaging**

  Este método vai ler a instrução que tem que ser executada e vai executar o método que corresponde a essa mesma instrução. Tem a particularidade de executar as instruções depois da execução do método “prints” em cada ciclo.

- **executionPaging**

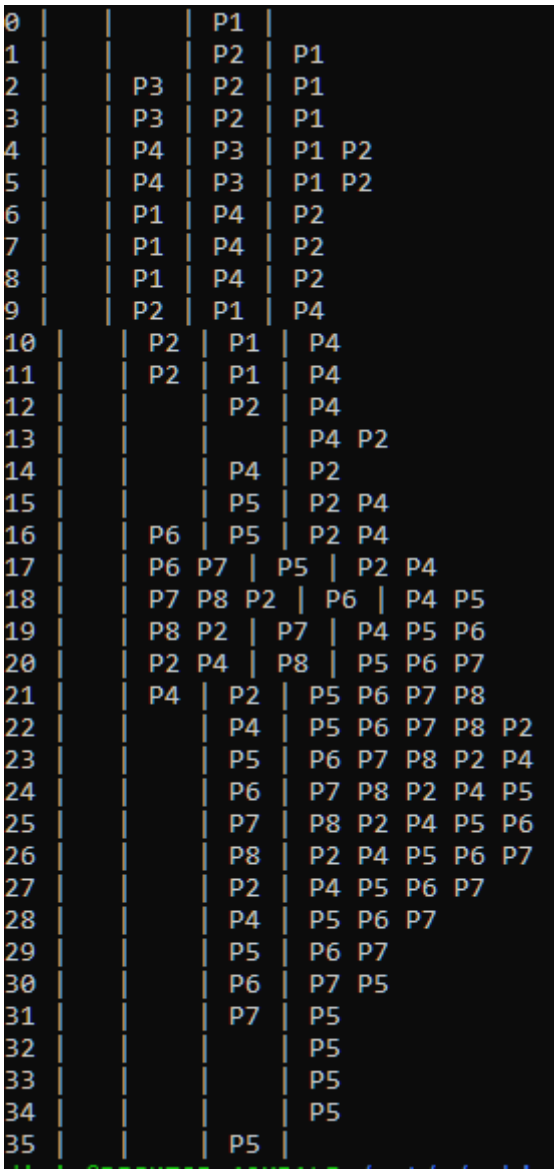
  Método que executa o programa com paginação.

Main

Método principal que inicializa o programa. Varia entre o modo “so_best” que corre o programa com o algoritmo BEST FIT e “so_pag” que corre o programa com paginação.

Output – teste1

Output Paginação



Output Best Fit