UFSC - Sistemas de Informação - INE5611-04238A/B (20241) - Sistemas Operacionais

Grupo 1 -
Cristiane Rodrigues Maragno (22200494);
Jenifer Sofia Ovejero (22200500);
Mariana Amaral Steffen (22200511).

Documentação do Código: Gerenciamento de Memória com Paginação

O programa implementa um sistema básico de um gerenciador de memória com paginação, proporcionando um exemplo prático de como sistemas operacionais modernos gerenciam a memória.

Estrutura do Código

MAX_PROCESSES: Define o número máximo de processos que o gerenciador de memória pode lidar simultaneamente.
BINARY_SIZE: Define o tamanho do endereço binário.

- **Estruturas de Dados**
  Process: Representa um processo no sistema.

  process_id: ID do processo.
  process_size: Tamanho do processo em bytes.
  page_table: Tabela de páginas que mapeia as páginas lógicas para os quadros físicos.
  num_pages: Número de páginas do processo.
  logical_memory: Memória lógica do processo.
  PhysicalMemory: Representa a memória física do sistema.

  memory: Vetor que representa a memória física.
  size: Tamanho da memória física.
  page_size: Tamanho de uma página.
  free_frames: Vetor que indica quais quadros estão livres.
  num_frames: Número de quadros na memória física.
  MemoryManager: Estrutura que gerencia os processos e a memória física.

  processes: Vetor de processos.
  num_processes: Número de processos no sistema.
  max_process_size: Tamanho máximo permitido para um processo.
  physical_memory: Instância de PhysicalMemory.

- **Funções**

  initialize_memory

  Inicializa a memória física e define os quadros como livres.

  Parâmetros: MemoryManager *mm, int physical_size, int page_size, int max_process_size
  Funcionalidade:
  Configura o tamanho da memória física.
  Define o tamanho da página.
  Calcula o número de quadros.

Inicializa todos os quadros como livres.

create_process

Cria um novo processo e aloca suas páginas na memória física.

Parâmetros: MemoryManager *mm, int process_id, int process_size
Funcionalidade:
Verifica se o processo cabe na memória física.
Calcula o número de páginas necessárias.
Aloca as páginas na memória física.
Atualiza a tabela de páginas do processo.

view_memory

Exibe o estado atual da memória física.

Parâmetros: MemoryManager *mm
Funcionalidade:
Calcula a porcentagem de memória utilizada.
Exibe o conteúdo da memória física.
view_page_table
Exibe a tabela de páginas de um processo específico.

Parâmetros: MemoryManager *mm, int process_id
Funcionalidade:
Exibe a tabela de páginas do processo identificado pelo process_id.

view_logical_memory

Exibe a memória lógica de um processo específico.

Parâmetros: MemoryManager *mm, int process_id
Funcionalidade:
Exibe o conteúdo da memória lógica do processo identificado pelo process_id.
display_menu
Exibe o menu de opções para o usuário.

Sem parâmetros.

get_binary

Converte um número inteiro para sua representação binária.

Parâmetros: int num, char *binaryStr
binaryStringToInt
Converte uma string binária para um número inteiro.

Parâmetros: char *binaryStr

getIndex

Extrai o índice de um endereço binário.

Parâmetros: int index

- **Função Principal**
  A função main realiza as seguintes operações:

  Solicita ao usuário os tamanhos da memória física, da página e do processo máximo.
  Inicializa a memória física.
  Exibe um menu de opções ao usuário para visualizar a memória física, criar um processo, visualizar a tabela de páginas de um processo ou sair do programa.

Como Compilar e Executar

- Compilação
  Assumindo que o **compilador GCC** esteja instalado, compile o programa com o seguinte comando:

  ```
  gcc -o <nome_arquivo_binario> pagination.c
  ```

- Execução
  Execute o programa com o comando:

  ```
  ./<nome_arquivo_binario>
  ```

Observações

Os endereços têm um limite de um byte.
.