Atividade de Laboratório
Paginação (Estruturas de Dados)

💻 Laboratório de Sistemas Operacionais

Prof. Lucas Figueiredo e Prof. Jamilson Bispo

Nome do(a) aluno(a)	**RA**	**Turma**

Em sistemas operacionais modernos, a memória virtual permite que programas utilizem mais memória do que está fisicamente disponível. Isso é alcançado através da paginação, onde a memória é dividida em páginas que podem ser armazenadas tanto na memória física quanto no disco.

Neste laboratório, você irá implementar as estruturas básicas necessárias para simular um sistema de memória virtual simples. Este é o primeiro passo para o desenvolvimento do simulador completo de paginação que será o projeto final.

Componentes a serem implementados:

1. **Memória Física**: Represente a RAM do sistema, dividida em frames de tamanho fixo.
2. **Memória Virtual**: Represente o espaço de endereçamento virtual, dividido em páginas do mesmo tamanho dos frames.
3. **Tabela de Páginas**: Crie uma estrutura que mapeia páginas virtuais para frames físicos.
4. **Processo**: Represente um processo simples com seu próprio espaço de endereçamento virtual.
5. **Função de Tradução de Endereço**: Implemente uma função que converta endereços virtuais em físicos.

TAREFA

1. **Projete e implemente estruturas de dados para representar a memória física, virtual e a tabela de páginas.**

   Abaixo está um exemplo (alguns atributos são opcionais) de como a construção da ED para um frame pode ser implementada em C:

```c
1 la1proj2.c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

// Definição do tamanho de cada frame (em bytes)
#define TAMANHO_FRAME 4096  // 4KB, um tamanho típico de página/frame

// Estrutura que representa um frame individual na memória física
typedef struct {
    int id;              // Identificador único do frame
    bool ocupado;        // Indica se o frame está em uso
    int processo_id;     // ID do processo que está usando este frame (-1 se livre)
    int pagina_id;       // ID da página que está armazenada neste frame (-1 se livre)
    char *dados;         // Ponteiro para os dados armazenados no frame
} Frame;
```

   A paginação divide a memória em unidades de tamanho fixo. Na memória física, essas unidades são chamadas de frames, e na memória virtual são chamadas de páginas. Além dos frames, você precisará implementar:

   - Estrutura para Memória Virtual (páginas)
   - Estrutura para Tabela de Páginas
   - Estrutura para gerenciar a Memória Física (conjunto de frames)

   Lembrando que a memória virtual permite que programas utilizem um espaço de endereçamento maior que a memória física disponível. A tabela de páginas mantém o mapeamento entre páginas virtuais e frames físicos.

2. **Crie funções para inicializar essas estruturas e manipulá-las.**

   Você precisará implementar funções para:

   - Inicializar a memória física (conjunto de frames).
   - Inicializar a memória virtual (conjunto de páginas).
   - Criar e gerenciar a tabela de páginas.
   - Alocar/desalocar páginas e frames.

   O sistema operacional precisa manter registro de quais frames estão livres e quais estão ocupados. Quando ocorre uma falta de página (page fault), o sistema precisa encontrar um frame livre ou escolher um frame para

substituição **(o tratamento de page faults será abordado no próximo laboratório).**

3. **Implemente a função de tradução de endereço virtual para físico.**

   Para traduzir um endereço virtual em físico:

   1. Extraia o número (id) da página do endereço virtual
   2. Use a tabela de páginas para encontrar o frame correspondente
   3. Combine o número do frame com o offset para formar o endereço físico

   Lembre que um endereço virtual é dividido em duas partes:

   - Número da página virtual (bits mais significativos)
   - Offset dentro da página (bits menos significativos).

A tradução mantém o offset inalterado e apenas converte o número da página em número do frame usando a tabela de páginas.

4. **(ENTREGA) Desenvolva um programa de teste que demonstre o funcionamento básico do seu simulador.**

   Seu programa de teste deve demonstrar:

   - Criação e inicialização das estruturas
   - Alocação de páginas para um processo. A estrutura utilizada para os processos deve contemplar também quais endereços (virtuais) cada processo irá tentar acessar. Exemplo:

```c
typedef struct {
    int pid;                    // ID do processo
    int *enderecos;             // Array com endereços que o processo vai acessar
    int num_enderecos;          // Quantidade de endereços a serem acessados
    int tamanho_processo;       // Tamanho total do processo em bytes
} Processo;
```

   - Tradução de alguns endereços virtuais para físicos

   O programa deve simular como um sistema operacional real gerencia a memória virtual, incluindo:

   - Mapeamento de páginas para frames
   - Tradução de endereços

- Detecção de páginas não presentes na memória física (indicando necessidade de page fault).
    - Por ora, basta gerar um log relatando a falha da página. O tratamento dessa situação será tópico do próximo laboratório.

Observações Importantes

1. Mantenha as estruturas de dados simples inicialmente.
2. Use constantes para definir tamanhos (TAMANHO_PAGINA, NUM_FRAMES, etc.).
3. Implemente verificações de erro apropriadas.
4. Lembre de documentar suas decisões de projeto.