**CENTRO UNIVERSITÁRIO UNIDOMBOSCO**

**ANÁLISE E DESENVOLVIMENTO DE SISTEMAS**

**TRABALHO DE SISTEMAS OPERACIONAIS**

GUILHERME PENSO, MURILO LUSTOSA, EMANOEL ANDRE, MATHEUS GUILHERME

**GERENCIAMENTO DE MEMÓRIA**

**CURITIBA**

**2023**

GUILHERME PENSO, MURILO LUSTOSA, EMANOEL ANDRE, MATHEUS GUILHERME

Trabalho do 2º Semestre do Curso Tecnólogo Análise e Desenvolvimento de Sistemas do Centro Universitário UniDomBosco, envolvendo a matéria de Sistemas Operacionais com o objetivo de criar um gerenciador de memória funcional.

Orientador: Edson Pedro Ferlin

CURITIBA

2023

SUMÁRIO

1. **INTRODUÇÃO** **1**
2. **LÓGICA** **2**
3. **REQUISITOS** **2**
4. **FUNCIONAMENTO** **2**

4.1 TELAS ....................................................................................... **4**

1. **DESCRITIVO** **2**
2. **CONSIDERAÇÕES FINAIS** **2**
3. **REFERÊNCIAS** **2**
4. **INTRODUÇÃO**

O gerenciamento de memória é uma função essencial em qualquer sistema de computação, responsável por alocar, desalocar e, se necessário, realocar espaços de memória. Este trabalho se concentrará no “Gerenciador de Memória”, uma parte integral do sistema operacional que realiza essas funções.

O Gerenciador de Memória tem a capacidade de alocar e desalocar espaços de memória para diferentes processos. Ele também pode realocar esses espaços conforme necessário, garantindo que cada processo tenha a quantidade adequada de memória para funcionar eficientemente.

Além disso, o Gerenciador de Memória tem a capacidade de limpar todos os processos de uma vez, liberando toda a memória e permitindo que o sistema comece do zero. Isso pode ser útil em situações onde o sistema está sobrecarregado ou quando um novo processo precisa de uma grande quantidade de memória.

Este trabalho explorará essas funções em detalhes, discutindo os vários algoritmos e técnicas usados para gerenciar a memória de maneira eficiente. Também discutiremos os desafios associados ao gerenciamento de memória e as soluções para esses desafios.

Com este estudo, nosso objetivo é aprofundar nosso entendimento sobre o papel crucial do Gerenciador de Memória em sistemas computacionais.

1. **LÓGICA**

**Método allocate():**

Esse método é usado quando o usuário quer alocar memória para um processo. Ele pede ao usuário o tamanho do processo, verifica os blocos de memória livres e tenta encontrar o melhor ajuste para o processo. Se houver espaço suficiente, ele gera um nome e uma cor para o processo e atualiza os blocos de memória para mostrar que estão ocupados pelo novo processo. Se não houver espaço suficiente, ele informa ao usuário com uma mensagem de erro.

**Método reallocate():**

Esse método é acionado quando um processo precisa ser realocado. Ele encontra o primeiro processo que pode ser realocado, desaloca esse processo e, em seguida, procura blocos livres para realocar o processo. Este processo continua até que todo o processo tenha sido realocado ou não haja mais blocos livres suficientes. E por fim, o método então retorna a grade atualizada.

**Método deallocate():**

O método é acionado quando o usuário deseja desalocar memória de um processo específico. Ele solicita ao usuário o nome do processo. Em seguida, percorre todos os 100 blocos de memória e, se encontrar um bloco ocupado pelo processo, atualiza o bloco para mostrar que está livre. Isso é feito alterando o texto e a cor de fundo do bloco para branco na interface do usuário e atualizando o status do bloco para 0 (livre).

**Método full\_deallocate():**

O método é acionado quando o usuário deseja desalocar toda a memória. Ele percorre todos os 100 blocos de memória e, se encontrar um bloco ocupado, atualiza o bloco para mostrar que está livre. Isso é feito alterando o texto e a cor de fundo do bloco para branco na interface do usuário e atualizando o status do bloco para 0 (livre). No final, toda a memória estará desalocada.

**Método process\_name\_generator():**

O método é um gerador que produz nomes de processos únicos. Ele começa gerando todas as possíveis combinações de letras do alfabeto inglês de tamanho 1 (ou seja, ‘A’ até ‘Z’). Quando todas as combinações de tamanho 1 são esgotadas, ele passa para combinações de tamanho 2 (ou seja, ‘AA’ até ‘ZZ’), e assim por diante. Isso garante que cada nome de processo gerado seja único.

**Método create\_grid():**

O método é responsável por criar uma grade visual de blocos de memória no aplicativo. Ele cria um quadro e, em seguida, adiciona 100 rótulos a esse quadro. Cada rótulo representa um bloco de memória e é inicialmente configurado para ser branco e ter um certo tamanho.