UnB Departamento de Ciência da Computação

Relatório Grupo 8

Professor(a):

Aletéia Patrícia Favacho de Araújo

Aluno(s):

Daltro Oliveira Vinuto Mat: 160025966

Otávio Souza de Oliveira Mat: 150143401

Descrição das ferramentas/linguagens usadas:

Linguagem utilizada: Python - Devido ser interpretada e o que torna assim mais fácil o processo de desenvolvimento.

Ferramentas utilizadas: SO linux, interpretador do python versão 3.10 -
O SO linux(Ubuntu) é um SO de código aberto e é o SO mais utilizado no mundo tanto no mundo dos supercomputadores quando nos servidores e aparelhos móveis somente sendo superada no ambiente de computadores pessoais devido a sua curva de aprendizado mais curva em relação a outros SO de código fechado.

Bibliotecas utilizadas: mypy - Que é uma biblioteca que permite fazer correção estática de código e checar o type hinting do python, facilitando assim o processo de desenvolvimento, debugação e automatização a detecção de erros simples como atribuições entre objetos de tipos diferentes.

Descrição teórica e prática da solução dada:

1. Gerenciamento de Processos:

O Gerenciamento de Processos é uma parte crucial dos sistemas operacionais e refere-se à administração e coordenação de processos em execução no sistema. Um processo pode ser definido como um programa em execução, juntamente com seu contexto de execução, como registradores, memória e estado. O gerenciamento de processos envolve a criação, suspensão, retomada, escalonamento e término de processos. As principais responsabilidades incluem:

- Criação de Processos;
- Escalonamento;
- Gerenciamento de estados.

Com esse objetivo a tabela de processos e os respectivos PCB de cada processo foram implementados como sendo o estado interno de atributos dos objetos processos que estão em estado pronto ou executando. Não foi necessário atingir o estado bloqueado pois os processos somente obtêm os recursos ou não no momento da inicialização apenas. Esse gerenciamento utiliza o objeto Processo de processos.py e as funções e executa_processo() e escalona_processos() do arquivo kernel.py.

2. Gerenciamento de Memória:

O Gerenciamento de Memória lida com a alocação e desalocação eficiente de espaço de memória para processos em execução. As principais tarefas incluem:

- Alocação de memória;
- Mapeamento de endereços;
- Proteção de memória;
- Desalocação de memória.

Esse gerenciamento foi realizado pela classe Memoria em gerencia_de_memoria.py e para isso ele usa um mapa de bits para manter o controle das áreas de memória livres e usadas. É utilizado o algoritmo first fit.
A memória é alocada no momento em que o processo é inicializado e liberado no momento em que ele é liberado, isso é, seu estado muda para finalizado.

3. Gerenciamento do Sistema de arquivos:

O Gerenciamento do Sistema de Arquivos refere-se à administração e organização de dados armazenados em dispositivos de armazenamento de longo prazo, como discos rígidos. Principais aspectos incluem:

- Organização de dados;
- Operações de arquivos;
- Controle de acesso;
- Recuperação de falhas.

Esse gerenciamento é realizado pela classe Arquivo em gerencia_de_arquivos.py e para isso ele usa um mapa de bits para manter o controle das áreas de memória livres e usadas. Ele mantém os dados do disco na lista de string dados no disco que é inicializado pela função inicializa_sistema_arquivos(). Ele utiliza o algoritmo first fit.

4. Gerenciamento de E/S:

O Gerenciamento de E/S lida com as operações de entrada e saída no sistema, envolvendo a comunicação entre o sistema e dispositivos periféricos. As principais tarefas incluem:

- Buffering;
- Gerenciamento de dispositivos;
- Interrupções;
- Drivers.

Esse gerenciamento é realizado pelo objeto IO_Hierarquia no arquivo gerenciador_de_io.py e para tal se utiliza de uma lista de listas de booleanos sendo False e o recurso está disponível e sendo True se está sendo utilizado por algum processo. Os recursos são alocados quando os processos são inicializados(carregados na memória) e liberados quando o processo foi finalizado e portanto seu estado passou para finalizado.

Descrição das principais dificuldades encontradas durante a implementação:

Uma das dificuldades encontradas foi a definição da linguagem de programação a ser usada no projeto, houve uma indefinição entre c++ e Python, entre usar uma linguagem de tipagem forte(c++) ou tipagem dinâmica(python).

Outra dificuldade encontrada foi na implementação de algumas funcionalidades do sistema como a estrutura de filas dos processos. Inicialmente foi feita uma fila ordenada pela prioridade do processo. Porém na tentativa de padronizar a estrutura de filas para o

que o projeto pede na especificação (fila de processos de tempo real, fila de processos de usuário), houveram muitos problemas pois a saída do programa foi afetada.

Outra dificuldade foi a necessidade de usar threads e sincronizar a execução das funções inicializa_processos() e escalona_processos().

Outra dificuldade encontrada foi manter o processo na fila de prontos enquanto o processo não tinha memória suficiente para ser alocado.

Para todas as dificuldades encontradas, explicar as soluções usadas:

1.Estrutura de Filas: Foram utilizadas duas listas na inicialização do Classe **Fila_Global**, chamadas fila_tempo_real para armazenar processos com prioridade igual a 0, e filas de usuário para prioridades maiores que zero. Sendo que o método **adiciona_processo** é responsável por adicionar um processo na fila correta, o método **obtem_prox_processo** é um método usado pelo escalonador para selecionar o próximo processo a ser executado.

2.Gerência de Processos: Foi utilizado uma lista de novos processos para armazenar os processos que seriam executados no tempo apropriado e outra lista de processos prontos para conter os próximos processos a serem carregados na memória principal e enviados para a fila global dos escalonados do kernel.

3.Gerência de Memória: Para o controle da alocação e desalocação da memória foi utilizado um mapa de bits para o controle das posições livres da memória. Foi utilizado a estratégia "first fit" para encontrar a próxima posição livre da memória.

4.Gerência de Arquivos: Para a alocação dos arquivos foi utilizado uma lista de strings contendo o nome(letra) do arquivo para cada posição dos dados e foi utilizado uma lista de bits(lista de inteiros) para controlar se as posições estão vazias ou ocupadas. Foi utilizada a estratégia "first fit" para encontrar o próximo conjunto de blocos do disco livre.

5.Gerência Entrada/Saída: Para a alocação e liberação dos recursos foi utilizado uma lista de listas de booleanos contendo a informação referente se o recurso está alocado(True) ou não(False).
Que é controlado pela classe IO_Hierarquia.

6. Sincronização entre o procedimento de ficar contando o tempo e esperar para que seja hora de inicializar o procedimento(criar o procedimento ao carregar o programa na memória): Foram utilizadas duas threads, uma para contar o tempo e inicializar os procedimentos no tempo adequado e outra para escalonar e executar os procedimentos nas filas de pronto.

Descrever o papel/função de cada aluno na realização do trabalho:

Otávio: Responsável pela estrutura das filas. Criação lógica e estrutural da fila de processos de tempo real, e das filas de usuários. Auxiliar na implementação de outras funcionalidades do programa, como, leitura e mapeamento dos processos.

Daltro: Responsável pela arquitetura do projeto. Contribuiu na especificação e implementação e teste de múltiplos módulos do projeto. Contribuiu para a elaboração do relatório.

Bibliografia:

1. Aulas presenciais. UnB. DF.

2. Slides da professora. URL: https://aprender3.unb.br/

3. Silberschat. A., Galvin P. B., Gagne, G. Operating System Concepts. LSC,, Rosewood Drive, Danvers, 10° edition, 2018.

4. Tanenbaum, A. S., Bos, W. Modern Operating Systems . Person, River Street, Hoboken, 5° edition., 2023.

5. Stallings, William. Operating Systems: internals and Design Principles. Person, global ed., 2018.
6. Martin, R. C. Clean Code. A Handbook of Agile Software Craftsmanship, Prentice Hall, 2008.