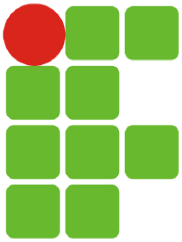

INSTITUTO FEDERAL DE EDUCAÇÃO
CURSO SUPERIOR DE TECNOLOGIA EM SISTEMAS PARA INTERNET
DISCIPLINA: SISTEMAS OPERACIONAIS
PROFESSOR: ANDRÉ LUÍS DE LUCENA TORRES

SIMULADOR FIFO: ALGORITMO DE ESCALONAMENTO DE PROCESSOS

EQUIPE

Product Owner e Desenvolvedor: Douglas Carneiro - 20231370002

Desenvolvedor e Testador: Jonata Barbosa - 20222370034

Relator e Roteirista: Luan Pimenta - 20231370042

Apresentador e Roteirista: Pedro Arthur - 20231370041

João Pessoa - PB

2024.1

1. Introdução

- **Objetivo do Projeto:** O objetivo deste projeto é desenvolver um simulador que implemente o algoritmo de escalonamento de processos FIFO (First In, First Out), analisando seu comportamento e desempenho.

2. Descrição do Projeto

- **O que é o Simulador FIFO:** O simulador FIFO tem como função simular o comportamento de um sistema operacional no escalonamento de processos.

- **Principais Funcionalidades:**
  - **Leitura de Processos via Arquivo CSV:** Os processos são carregados de um arquivo CSV que contém suas respectivas informações de tempo de chegada e tempo de execução.
  - **Execução Simulada dos Processos:** Cada processo é executado na ordem de chegada, e o tempo de execução é simulado com a possibilidade de incluir pausas (ou pulá-las, conforme parâmetro).
  - **Cálculo de Tempos (Espera, Turnaround e Resposta):** Durante a execução, o simulador calcula e exibe o tempo de espera, o turnaround, e o tempo de resposta de cada processo.
  - **Relatório de Resultados:** Após a execução dos processos, é exibido um resumo com os tempos médios de espera, turnaround e resposta.

- **Requisitos Específicos:**
  - **Arquivo CSV de Processos:** O simulador exige um arquivo CSV contendo os processos a serem escalonados.

3. Arquitetura e Design

- **Visão Geral da Arquitetura:** O simulador é composto por quatro módulos principais: `main.py`, `carregar_arquivo.py`, `processo.py` e `run.py`. A interação entre esses módulos permite o carregamento de processos, execução do algoritmo FIFO, e exibição dos resultados.

- **Estruturas de Dados:** Cada processo é representado por uma instância da classe `Processo`, que contém atributos como nome, tempo de chegada, tempo de execução, tempo de espera, tempo de término, e tempo de resposta.

- **Funcionamento do Algoritmo FIFO:** No algoritmo FIFO, os processos são ordenados com base no tempo de chegada. O primeiro processo na fila é executado até o final, seguido pelo próximo, até que todos os processos tenham sido executados. Durante a execução, são calculados o tempo de espera, tempo de resposta, e o turnaround de cada processo.

4. Implementação

- **Ambiente de Desenvolvimento**: A linguagem de programação utilizada foi Python. O desenvolvimento ocorreu no Visual Studio Code, e os testes foram realizados via terminal, utilizando o arquivo `processos.csv` para carregar os processos.

- **Detalhes do Código:**
  - **carregar_arquivo.py:** Responsável por carregar os processos a partir de um arquivo CSV.
  - **main.py:** Contém o algoritmo FIFO, que gerencia a execução dos processos, calcula tempos de espera e turnaround, e exibe os resultados.
  - **processo.py:** Define a estrutura de um processo, com atributos relevantes para o cálculo dos tempos.
  - **run.py:** Utilizado para executar o simulador via terminal, permitindo passar parâmetros como o arquivo CSV e a opção de pular o tempo de espera entre execuções.

- **Desafios e Soluções:**
  - **Desafio:** Garantir que os processos fossem executados na ordem correta, respeitando o tempo de chegada. **Solução:** Implementação de uma ordenação com base no tempo de chegada dos processos, antes da execução.

5. Testes e Resultados

- **Casos de Teste:**
    - O simulador utiliza, por padrão, o arquivo `processos.csv` com os seguintes dados:

Processo	Tempo de Chegada	Tempo de Execução
P1	6	5
P2	3	3
P3	5	2
P4	4	4
P5	0	2

Esse arquivo contém os processos usados para os testes, onde cada processo possui um tempo de chegada e um tempo de execução, e o simulador segue a política FIFO (First In, First Out) para escalonamento.

- **Execução dos processos:**

  Processo P5:
  Como o processo P5 chegou primeiro (no tempo 0), ele foi o primeiro a ser executado, sem necessidade de esperar.
  1. Tempo de Espera: 0
  2. Tempo de Resposta: 0
  3. Tempo de Término: 2
  4. Turnaround: 2

  Processo P2:
  O processo P2 chegou no tempo 3, após o término de P5, então pôde ser executado logo em seguida, sem esperar.
  1. Tempo de espera: 0
  2. Tempo de Resposta: 0
  3. Tempo de Término: 6
  4. Turnaround: 3

  Processo P4:
  O processo P4 chegou no tempo 4, mas como P2 estava sendo executado, ele precisou esperar até o tempo 6 para começar.
  1. Tempo de espera: 2
  2. Tempo de Resposta: 2
  3. Tempo de Término: 10
  4. Turnaround: 6

Processo P3:
O processo P3 chegou no tempo 5, mas precisou esperar até o término de P4 no tempo 10 para ser executado.

1. Tempo de espera: 5
2. Tempo de Resposta: 5
3. Tempo de Término: 12
4. Turnaround: 7

Processo P1:
Por fim, o processo P1, que chegou no tempo 6, só pôde ser executado depois do término de todos os processos anteriores, começando no tempo 12.

1. Tempo de espera: 6
2. Tempo de Resposta: 6
3. Tempo de Término: 17
4. Turnaround: 11

- **Resumo da Execução:**

Processo	Tempo de Execução	Tempo Espera	Turnaround	Tempo Resposta
P5	2	0	2	0
P2	3	0	3	0
P4	4	2	6	2
P3	2	5	7	5
P1	5	6	11	6

- **Cálculo Finais:**
  - **Tempo Médio de Espera:**
    - 0 + 0 + 2 +5 + 6 / 5 = 2.6
  - **Tempo Médio de Turnaround:**
    - 2 + 3 + 6 +7 + 11 / 5 = 5.8
  - **Tempo Médio de Resposta:**
    - 0 + 0 + 2 +5 + 6 / 5 = 2.6

- **Análise de Resultados:**
  O simulador FIFO processou os processos corretamente, sempre respeitando a ordem de chegada. Como esperado, os processos que chegaram mais cedo foram executados primeiro, e os cálculos de tempo de espera, turnaround e resposta foram consistentes com a teoria do algoritmo.
  Os tempos médios de espera e turnaround confirmam que o comportamento do simulador está correto, sem inversões ou erros na ordem de execução.

- **Alterando os Dados ou Arquivo Padrão:**
  Caso o usuário deseje testar outros processos, ele pode modificar diretamente os dados no arquivo CSV padrão `processos.csv`, ou carregar outro arquivo CSV utilizando o argumento `--file ou -f` ao executar o simulador.

  Exemplo de como alterar o arquivo:

  ```
  python simulador-fifo.py --file ./meus_processos.csv
  ```

  Dessa forma, o simulador carregará os processos a partir do arquivo utilizado em vez do padrão
  Também é possível pular a espera real do tempo de execução dos processos utilizando o argumento `--skip ou -s`, caso o usuário deseje acelerar a simulação:

  ```
  python simulador-fifo.py --file ./meus_processos.csv --skip
  ```

6. Conclusão

- **Resumo dos Resultados:** O simulador FIFO desenvolvido conseguiu simular corretamente o comportamento de um sistema de escalonamento de processos, fornecendo tempos médios de espera, turnaround e resposta adequados aos processos testados.

- **Aprendizados:**
  - Melhor compreensão do funcionamento do algoritmo FIFO.
  - Desenvolvimento de habilidades de manipulação de arquivos e tratamento de dados em Python.
  - Aplicação prática dos conceitos de escalonamento de processos estudados em sala de aula.

7. Referências

- Silberschatz, Abraham. Operating System Concepts. 10th Edition.
- Documentação oficial do Python: https://docs.python.org