**Implementação de simulador de gerenciamento de memória**

**Roger Fagner Rothmund;**

Departamento de Computação

Curso de Ciência da Computação

Universidade de Santa Cruz do Sul (UNISC)

Av. Independência, 2293 - 96815-900 – Santa Cruz do Sul, RS

rrothmund@mx2.unisc.br

***Resumo.*** *Este trabalho tem como objetivo o desenvolvimento de um simulador de gerenciamento de memória na linguagem Python na plataforma Windows. Através de uma interface terminal, a implementação consiste em possibilitar que o usuário entre com processos através de arquivo ou pelo teclado utilizando os critérios de alocação first-fit, best-fit e worst-fit.*

**1. Introdução**

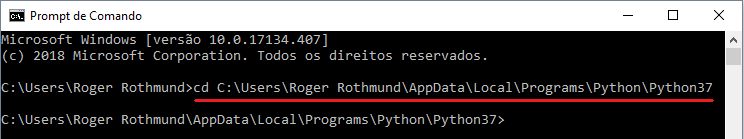
Teremos como ponto principal deste documento um passo a passo de como executar o simulador, com orientação do que é necessário para a sua execução e também como alguns padrões utilizados na leitura de arquivos.

**2. Requisitos para execução do Simulador**

Esta parte é destinada a especificar os requisitos para a execução do simulador em qualquer máquina, a partir de figuras e textos explicativos mostrarei como instalar os requisitos básicos para o correto funcionamento do simulador.

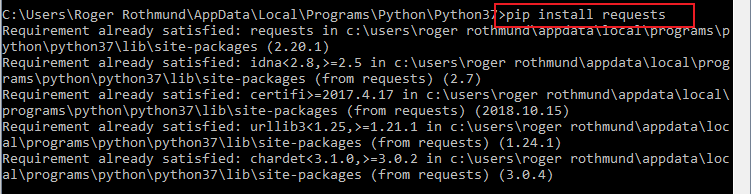
Para o funcionamento do arquivo “simuladorMMU.py” é essencial a execução na plataforma Windows e a instalação do Python 3.7 onde as seguintes bibliotecas “os” e “copy” são necessárias.

Caso não tenha o Python instalado faça o download através do link <https://www.python.org/downloads/> selecionando a plataforma de sua preferência. Após será necessário instalar as bibliotecas mencionadas acima, para isso localize onde foi feita a instalação do Python, por padrão se encontra em C:\Users\AppData\Local\Programs\Python\Python37, feito isso abra o terminal e acesse esta pasta com o comando “cd”.



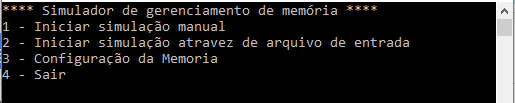
**Figura 1 – Acesso a pasta de instalação do Python**

Com o acesso a pasta feito basta digitar o comando “pip install” seguido do nome da biblioteca necessária, e aguardar a instalação.



**Figura 2 – Instalação de biblioteca**

A utilização é muito simples, uma vez que as bibliotecas e o Python estejam instalados, execute o simulador arquivo “simuladorMMU.py” com um duplo clique e o menu inicial vai aparecer.

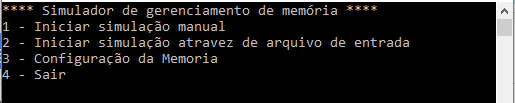


**Figura 3 – Menu inicial**

**3. Funcionalidades**

A seguir as funcionalidades implementadas no simulador conforme os requisitos estipulados no trabalho, a principal funcionalidade é a implementação dos algoritmos first-fit, best-fit e worst-fit para alocação de memória e para permitir que seja testado, o simulador também deve conter uma interface com o usuário oferecendo um menu com as seguintes opções:

1. Iniciar simulação e selecionar critério de alocação (first-fit, best-fit e worst-fit);
2. Adicionar um novo processo à memória (o ID e o tamanho do processo devem ser indicados pelo usuário);
3. Remover um processo da memória (o processo a ser removido deve ser indicado pelo usuário);
4. Apresentar a lista de espaços livres;
5. Apresentar o registrador de relocação de um processo (ao ser alocado, um processo recebe um endereço inicial; sempre que a memória for compactada, os endereços iniciais de todos os processos deverão ser corrigidos);
6. Apresentar o mapa de utilização da memória (quais posições estão sendo ocupadas por quais processos e quais posições estão livres), na tela e em um arquivo texto;
7. Compactar a memória movendo os processos para posições contíguas, eliminando os espaços entre eles(esse processo de compactação poderá ocorrer mediante solicitação do usuário –devendo aparecer como opção no menu, ou quando se tentar alocar um processo e não existir espaço suficiente para tanto–durante a adição de um novo processo à memória);
8. Ler de um arquivo uma sequência de processos para serem executados, com seus respectivos tamanhos, instantes de início e duração, e simular o gerenciamento de memória considerando os critérios first-fit, best-fit e worst-fit. Apresentar um relatório comparativo da aplicação do gerenciamento de memória considerando os três critérios.

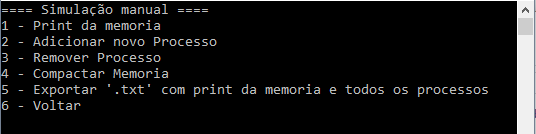


**Figura 4 – Menu inicial**

A opção 1 do menu principal do simulador cobre do requisito 1 até o 7, onde foi considerado como opções referentes a uma simulação manual, figura 5.

Já na opção 2 do menu principal do simulador cobre o requisito 8 do trabalho. Para este é necessário um arquivo de entrada que deve ter o nome “entrada.txt”. Este arquivo contém os processos e seus atributos para execução. Nesta implementação o arquivo de entrada deve ser disposto conforme a figura 6, com seus atributos separados por vírgula e uma linha para cada processo.

Por fim na opção 3 é possível alterar o tamanho da memória do simulador, que por padrão é 256 bytes.

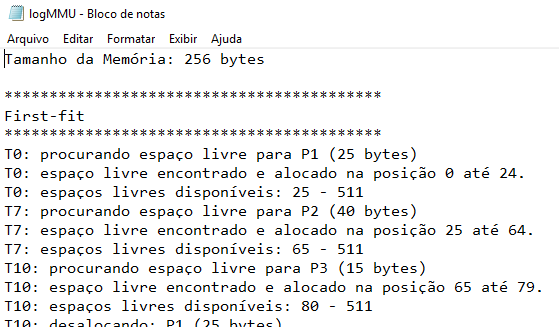


**Figura 5 – Opções da Simulação Manual**

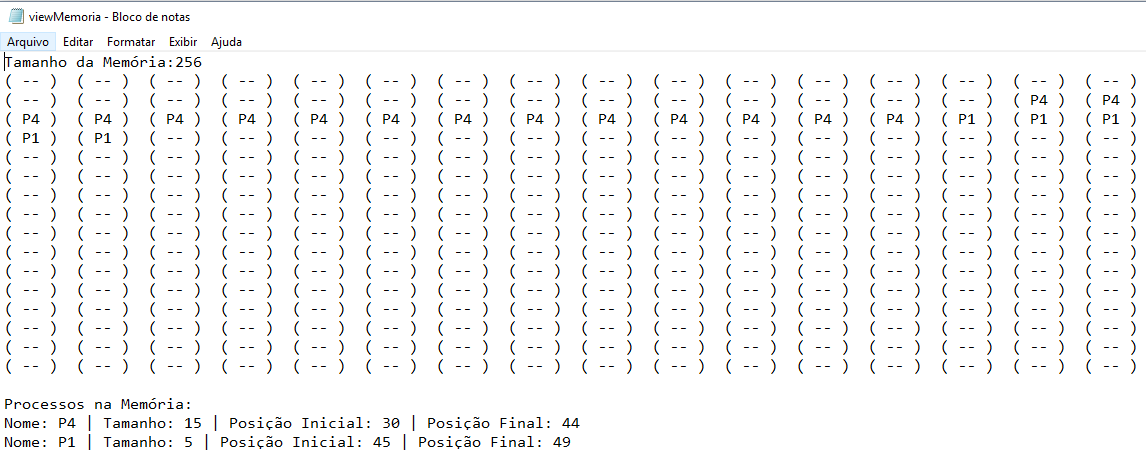


**Figura 6 – Arquivo “entrada.txt”**

O simulador gera dois relatórios, um proveniente da simulação por um arquivo de entrada chamado “logMMU.txt” e outro a partir da simulação manual, opção 5 conforme figura 5, chamado “viewMemoria.txt”. Estes arquivos são salvos na pasta onde o simulador é executado.



**Figura 7 – Arquivo “logMMU.txt”**



**Figura 8 – Arquivo “viewMemoria.txt”**

**4. Conclusão**

Este trabalho foi uma ótima experiência na utilização da linguagem Python na criação do simulador de gerenciamento de memória, o que também proporcionou um bom aprendizado na disciplina de Sistemas Operacionais.

Na parte de desenvolvimento problemas ocorreram como no caso da simulação por um arquivo de entrada, onde num primeiro momento não consegui entender de qual forma a simulação ocorreria, neste caso foi necessário adicionar uma fila para escalonar os processos a serem executados e assim simular a execução completa dos processos.