

# **Relatório do trabalho sobre Gerência de Memória com Segmentação e Memória Virtual**

**Grupo:** Bruno Behnken, Hugo Faria, Lucas Murakami, Rafael Mendes, Thales Nathan  
**DREs:** 111222925, 112177343, 112177864, 112031288, 111179740

## **Introdução:**

O objetivo do nosso trabalho é fazer um sistema que simule uma gerência de memória principal com segmentação e memória virtual.

Abaixo seguem as estratégias escolhidas e uma explicação de como funciona a aplicação.

## **Estratégia de Alocação de Segmentos:**

Nosso trabalho utiliza estratégia First-fit para alocação de segmentos na Memória Principal. Para guardar os espaços livres dela, é utilizada uma lista encadeada.

## **Estratégia de Realocação de Segmentos:**

Para a estratégia de Realocação, usamos o algoritmo Least Recently Used (LRU). Para isso, temos uma estrutura de dados que guarda os segmentos que estão presentes na Memória Principal. Lá são guardadas, além de outras informações para a aplicação, a “idade” de cada segmento. Esta é renovada a cada vez que seu segmento correspondente é referenciado enquanto os outros têm sua idade incrementada.

Na hora da decisão de qual segmento retirar da M.P. para dar espaço a outro que chega, o segmento que tem a maior idade é o que irá ser retirado.

## **Como funciona a simulação:**

Para iniciar o programa use o comando “./principal Memoria”, onde “Memoria” será um inteiro que representará o tamanho total da Memória Principal na simulação.

Ao iniciar o programa, um menu informa ao usuário as opções de ação que são:

### **1 – Inserir Processo no Sistema:**

Inclui um novo processo no sistema, que passará a ter seus segmentos passíveis de alocação em M.P.

### **2 – Excluir Processo do Sistema:**

Exclui um processo do sistema, de forma que seus segmentos já não poderão mais ser acessados. Caso algum segmento do processo que está sendo excluído esteja alocado na M.P., este será desalocado.

### **3 – Acessar Endereço Real:**

É informado ao sistema qual o PID do processo que tem o segmento que se deseja acessar e o Endereço Virtual que se deseja acessar (o número desse segmento na tabela de segmentos e o offset).

Após passar essas informações, o sistema aloca, se necessário, o segmento na M.P. e atualiza as “idades” de cada segmento alocado de acordo com a estratégia do

LRU. O sistema também mantém atualizadas as tabelas de segmentos envolvidas nas possíveis realocações, setando bits de presença e as bases em M.P. de cada segmento alocado.

Além de guardar bits de presença e base em M.P., a tabela de segmentos também guarda o tamanho de cada segmento, para diversos fins como checar se o offset dado é válido (dentro dos limites do segmento).

Finalmente, é exibido para o usuário o endereço real em M.P. da base do segmento desejado assim como a mapeação do endereço virtual fornecido para real fazendo "base + offset".

#### 4 – Ver os Intervalos de Espaço Livre na M.P.:

Mostra a lista encadeada de espaços livres na M.P.

Ex: (300,500) → (600, 700), representando que na M.P. existem espaços livres entre as posições de memória 300 e 500 assim como entre 600 e 700.

#### 5 – Ver os Processos e seus Segmentos:

Mostra informações dos processos que estão incluídos no sistema no momento.

#### 6 – Ver as Idades dos Segmentos em Memória:

Mostra informações dos segmentos alocados em memória na seguinte forma:  
(Número do Segmento ; PID do Processo ; Idade)

#### 0 – Encerrar Programa:

Encerra simulação.

#### Conclusão:

Tivemos muitos bugs para corrigir e implementações para repensar, mas chegamos a um programa que é capaz de simular um gerenciamento de memória com segmentação e memória virtual.