# Relatório (Simulador de SO)

Lucas Nimirio Maia Feng - 11915950
Patrick Marques de Barros Costa - 11257550
Vitor  Chinaglia - 11912363

## Resumo

Esse relatório tem como objetivo mostrar todo o funcionamento do projeto, trata-se de um programa que faz o gerenciamento de processos e alocação de memória de um sistema operacional (Simulador SO).

Basicamente ele opera em um loop onde o usuário pode inserir comandos para criar processos e matar processos, alocando memória (4 unidades) com base nos processos que foram criados. Ficou definido como 20 o tamanho máximo de memória.

## Funções

Para essa alocação, foi usada a função “allocate memory”. Ela procura um PID disponível na matriz process, atribui-o a um intervalo de blocos de memória e atualiza a matriz memory com o PID do processo.

Depois, temos a função “createProcess” que vai criar um processo encontrando um intervalo contíguo de blocos de memória. Ela percorre a matriz de memória para encontrar um intervalo de memória livre do tamanho necessário, e em seguida, chama a função allocateMemory() para alocar memória para o processo.

A função “clearMemory” limpa toda a memória, definindo todos os valores na matriz memory como 0.

Essa função “killProcess”recebe um PID (Process ID) como entrada e percorre a matriz de memória. Se encontrar algum bloco de memória com o PID correspondente, libera a memória (definindo o valor como 0) e também remove o processo da matriz process (definindo o valor como 0).

Essa função “printMemory” imprime o estado atual da matriz memory, mostrando quais blocos de memória estão alocados para quais processos e quais estão livres.

Utilizamos também essa função “printMemoryToFile” que imprime o estado atual da matriz memory em um arquivo temporário, que é usado para exibir o status da memória em uma janela de terminal separada.

## MAIN

Por fim, temos a função principal que contém um loop que lê os comandos do usuário e realiza ações correspondentes. Ela usa uma interface de linha de comando simples para interagir com o usuário. Ela suporta dois comandos: "create -m" para criar um novo processo e "kill" para finalizar um processo.

No loop principal, o programa executa as seguintes etapas:

Solicita ao usuário para inserir um comando.
Analisa o comando inserido para determinar qual ação executar: criar um processo ou finalizar um processo.
Realiza a ação apropriada chamando as funções createProcess ou killProcess.
Gera um arquivo temporário para registrar o estado da memória após a operação e exibe o status da memória em um terminal secundário (dependendo do sistema operacional).
O loop principal continua repetindo essas etapas, permitindo que o usuário insira mais comandos.