TRABALHO 02 - SISTEMAS OPERACIONAIS 24.1

Aluno: Vinícius Araujo
Matrícula: 18205320

- **Instruções para compilar:**
  - No terminal do seu Linux, digite o seguinte comando:
    gcc main.c -o main
  - O arquivo *main.c* tem que estar na mesma pasta do arquivo *main.h*, e a compilação tem que ser feita dentro dessa pasta em específico.
  - A versão do Ubuntu é o **Ubuntu 20.04.2 LTS**

- **Principais partes do código:**
  - Função de visualização da memória

```c
void ver_memoria() {
    int num_quadros = tamanho_memoria_fisica / tamanho_pagina;
    int quadros_livres = 0;
    for (int i = 0; i < num_quadros; i++) {
        if (memoria_fisica[i].ocupado) {
            printf("Quadro %d: Ocupado pelo Processo %d\n", i, memoria_fisica[i].id_processo  );
        } else {
            printf("Quadro %d: Livre\n", i);
            quadros_livres++;
        }
    }
    printf("Memoria LIVRE: %.2f%%\n", (quadros_livres / (float)num_quadros) * 100);
}
```

Exibe o estado de cada quadro da memória física (ocupado ou livre) e qual processo está ocupando. Além disso, calcula e exibe o percentual de memória livre.

  - Função para criar um processo

```c
void criar_processo(int id_processo, int tamanho_processo) {
    if (tamanho_processo > tamanho_maximo_processo) {
        printf("Erro: Tamanho do processo excede o limite de %d bytes por processo.\n", tamanho_maximo_processo);
        return;
    }

    if (verifica_ID_processo(id_processo)){
        printf("ERRO: ID do Processo %d, ja existe. Tente outro.\n", id_processo);
        return;
    }

    int numero_paginas = (tamanho_processo + tamanho_pagina - 1) / tamanho_pagina; // numero de paginas necessecarias
    int quadros_livres = conta_quadros_livres();
    if (quadros_livres < numero_paginas){
        printf("ERRO: Sem memoria suficiente para alocar o processo\n");
        return;
    }

    int *tabela_paginas = (int *)malloc(numero_paginas * sizeof(int));

    for (int i = 0; i < numero_paginas; i++) {
        int quadro_livre = encontrar_quadro_livre();
        memoria_fisica[quadro_livre].ocupado = 1;
        memoria_fisica[quadro_livre].id_processo = id_processo;
        tabela_paginas[i] = quadro_livre;
    }

    Processo novo_processo;
    novo_processo.id_processo = id_processo;
    novo_processo.tamanho_processo = tamanho_processo;
    novo_processo.tabela_paginas = tabela_paginas;

    processos[contador_processos++] = novo_processo;
    printf("Processo %d criado com SUCESSO.\n", id_processo);
}
```

Verifica se o tamanho do processo excede o máximo e depois se o ID do processo já existe. Calcula o número de páginas necessárias para o processo e verifica se há quadros livres suficientes para acomodar todas as páginas do processo. Se houver espaço suficiente, aloca memória para a tabela de páginas e mapeia as páginas para os quadros físicos. Cria e adiciona o novo processo à lista de processos.

```c
int numero_paginas = (tamanho_processo + tamanho_pagina - 1) / tamanho_pagina;
```

Na linha de código acima, adicionei *tamanho_pagina - 1* ao *process_size* para garantir que qualquer fração de uma página adicional resulta em uma página completa

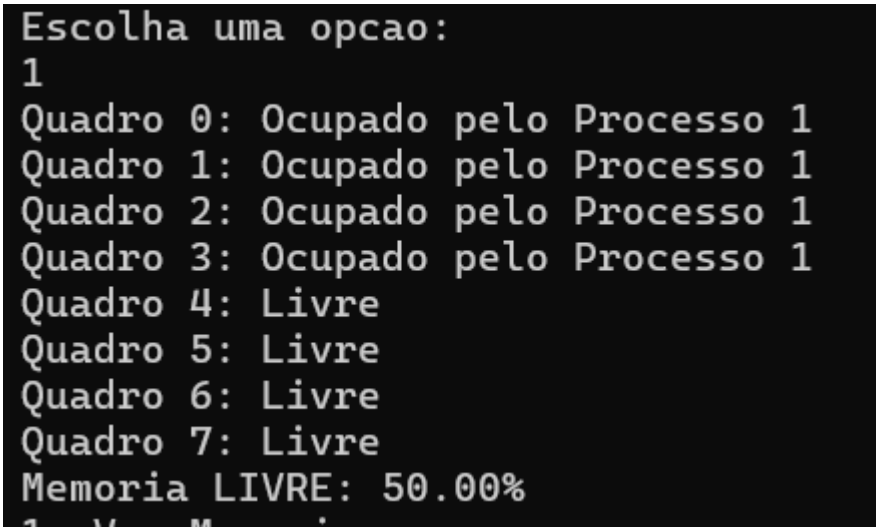
- Função para modificar os paramêtros:

```c
void modifica_configuracao(){
    printf("Tamanho da MEMORIA FISICA atual: %d bytes\n", tamanho_memoria_fisica);
    printf("Digite o novo tamanho da MEMORIA FISICA (multiplo de 2): ");
    scanf("%d", &tamanho_memoria_fisica);

    printf("Tamanho da PAGINA (QUADRO) atual: %d bytes\n", tamanho_pagina);
    printf("Digite o novo tamanho da PAGINA (QUADRO): ");
    scanf("%d", &tamanho_pagina);

    printf("Tamanho MAXIMO DE UM PROCESSO atual: %d bytes\n", tamanho_maximo_processo);
    printf("Digite o novo tamanho MAXIMO DE UM PROCESSO: ");
    scanf("%d", &tamanho_maximo_processo);

    free(memoria_fisica);
    inicializar_memoria();
}
```

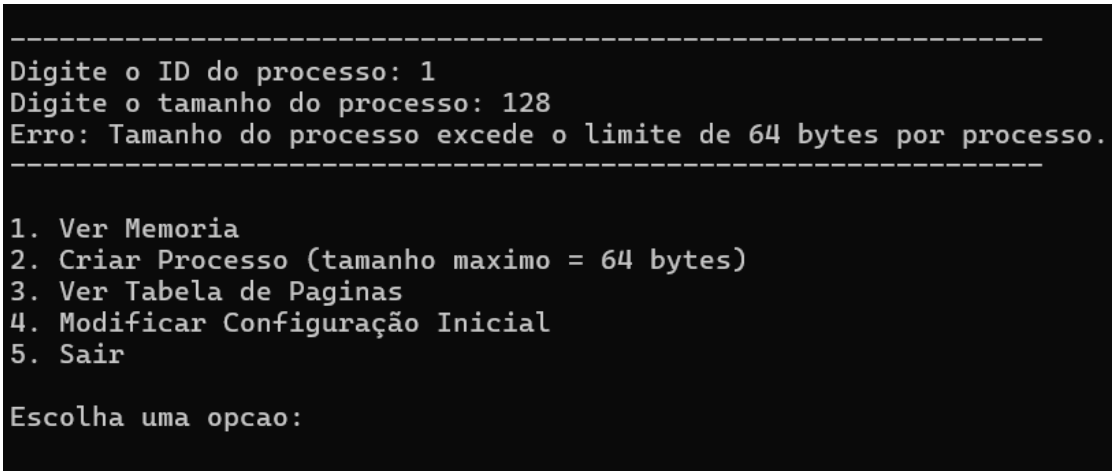
Permite ao usuário alterar os valores de *tamanho_memoria_fisica*, *tamanho_pagina*

e *tamanho_maximo_processo*. E após, libera a memória física e reinicializa a memória com os novos valores configurados.

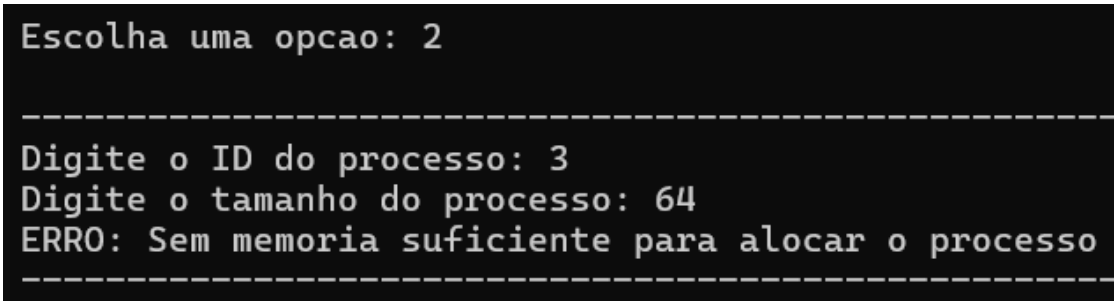
- **Testes e suas saídas:**
  - Alocando um processo de 64 bytes em uma memória fisíca de 128 bytes



  - Tentando criar um processo maior que o tamanho máximo permitido



  - Tentando criar um processo maior do que espaço disponível na memória física



- Exibindo da tabela de páginas de um processo

```
Escolha uma opcao: 3

--------------------------------
Digite o ID do processo: 1
ID do Processo: 1
Tamanho do Processo: 64 bytes
Pagina 0 -> Quadro 0
Pagina 1 -> Quadro 1
Pagina 2 -> Quadro 2
Pagina 3 -> Quadro 3
--------------------------------
```

- Modificando os valores iniciais da memória física, página e processo

```
------------------------------------------------------------
Tamanho da MEMORIA FISICA atual: 128 bytes
Digite o novo tamanho da MEMORIA FISICA (multiplo de 2): 512

Tamanho da PAGINA (QUADRO) atual: 16 bytes
Digite o novo tamanho da PAGINA (QUADRO): 32

Tamanho MAXIMO DE UM PROCESSO atual: 64 bytes
Digite o novo tamanho MAXIMO DE UM PROCESSO: 128

------------------------------------------------------------
1. Ver Memoria
2. Criar Processo (tamanho maximo = 128 bytes)
3. Ver Tabela de Paginas
4. Modificar Configuração Inicial
5. Sair

Escolha uma opcao: 1

------------------------------------------------------------
Quadro 0: Livre
Quadro 1: Livre
Quadro 2: Livre
Quadro 3: Livre
Quadro 4: Livre
Quadro 5: Livre
Quadro 6: Livre
Quadro 7: Livre
Quadro 8: Livre
Quadro 9: Livre
Quadro 10: Livre
Quadro 11: Livre
Quadro 12: Livre
Quadro 13: Livre
Quadro 14: Livre
Quadro 15: Livre

Memoria LIVRE: 100.00%
```