**Universidade Federal do Ceará**
**Disciplina: Sistemas Operacionais**
**Prof. Ms. Rafael Ivo**

# Trabalho 2: Paginação

## Regras gerais:

- O trabalho deve ser em grupos de até 3 pessoas. Os grupos devem ser formados até dia 05/09 (quarta-feira) e informados ao professor após o término da aula. Não será permitida a inserção de novo(s) integrante(s) em um grupo formado após esta data.
- O envio do trabalho será exclusivamente no SIGAA, no prazo estipulado.
- O código deve ser muito bem comentado, explicando as ideias utilizadas. Deve ser possível entender todo o funcionamento do código apenas lendo os comentários.
- O início do código deve conter o nome dos integrantes do trabalho. Se o código possuir mais de um arquivo, a identificação deve estar em cada um dos arquivos.
- A implementação pode ser feita em qualquer linguagem.

## Descrição Geral

Escreva um programa que simule o gerenciamento da memória RAM. Para isto, um vetor de caracteres com uma quantidade bem alta de posições será a única estrutura de armazenamento utilizada e este vetor será gerenciado por paginação. Os índices desse vetor serão os endereços físicos e cada posição do vetor será uma célula de memória com capacidade para armazenar exatamente um único caractere. Este vetor deve ser particionado em partes iguais denominadas ***quadros***. As primeiras informações pedidas pelo seu programa devem ser o tamanho da memória e o tamanho do quadro. (Não se preocupem que o último quadro fique com menos elementos, sempre pedirei potências de 2 para evitar este problema.) Várias *strings* serão armazenadas neste vetor. Cada string fará o papel de um processo qualquer na memória. Seu programa fará o papel do gerenciador de memória com as seguintes funcionalidades (juntamente com sua interpretação, no caso de um SO de verdade):

1 Funções de usuário
  - 1.1 Inserir uma string (inicializar um processo)
  - 1.2 Remover uma string (finalizar um processo)
  - 1.3 Editar uma string (um processo aloca mais memória)
  - 1.4 Acessar um determinado caractere da string (acessar uma célula de memória)

2 Funções de administrador (Usadas apenas para visualizar o status do sistema)
  - 2.1 Exibir o conteúdo de uma string (processo)
  - 2.2 Exibir o conteúdo completo do vetor (memória RAM)
  - 2.3 Exibir a tabela de páginas de uma string (processo)
  - 2.4 Exibir a lista de quadros livres

Seu gerenciador de memória deve manter para cada string (processo) uma tabela de páginas. Deve manter também uma lista dos quadros livres. Alocação e liberação de memória devem ser feitas apropriadamente assim como a tradução de endereço lógico para endereço físico. O gerenciador não deve utilizar o conteúdo do vetor (memória RAM) para tomar qualquer decisão.

## Exemplo de Execução

***Obs:*** *conteúdo em vermelho é a entrada feita em uma possível execução.*

```
→ Qual o tamamho da memória?
→ 128
→ Qual o tamanho do quadro?
→ 8
→ (U)suario ou (A)dministrador?
→ U
→ [1] Inicializar processo
→ [2] Finalizar processo
→ [3] Alocar memória para processo
→ [4] Acessar conteudo
→ Opção: 1
→ Conteúdo: GameOfThrones
```

*Esta string possui 13 caracteres. Dois quadros são necessários. Uma possível decisão é alocar os quadros 0 e 1. Atribua um id ao processo para poder identificá-lo depois nas outras funcionalidades.*

```
→ Processo de id 0 inicializado com sucesso.
```

*Suponha que eu repeti o processo acima de inicializar processo para mais duas strings: "Chaves" (id=1) e "UniversidadeFederalDoCeará" (id=2).*

```
→ (U)suario ou (A)dministrador?
→ A
→ [1] Exibir conteudo de um processo
→ [2] Exibir conteúdo da memória
→ [3] Exibir tabela de páginas de um processo
→ [4] Exibir lista de quadros livres
→ Opção: 2
→ GameOfTh
→ rones000
→ Chaves00
→ Universi
→ dadeFede
→ ralDoCea
→ rá000000
→ 00000000
…
→ 00000000
```

*Note que as células sem conteúdo ainda gravado possuem conteúdo inicial igual a zero. Este é o lixo de memória inicial. Graficamente, a memória estaria assim:*

| G | a | m | e | O | f | T | h | r | o | n | e | s | 0 | 0 | 0 | C | h | a | v | e | s | 0 | 0 | U | n | i | v | e | r | s | i |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| d | a | d | e | F | e | d | e | r | a | l | D | o | C | e | a | r | á | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

```
→ [1] Inicializar processo
→ [2] Finalizar processo
→ [3] Alocar memória para processo
→ [4] Acessar conteudo
→ Opção: 3
→ Id do processo: 1
→ Conteúdo: EmAcapulco
```

*Este novo conteúdo deve ser inserido ao final do processo 1. Como o espaço que estava sobrando no seu último quadro não é suficiente para armazenar este novo conteúdo, novos quadros devem ser alocados para ele. Graficamente a memória ficaria assim:*

| G | a | m | e | O | f | T | h | r | o | n | e | s | 0 | 0 | 0 | C | h | a | v | e | s | E | m | U | n | i | v | e | r | s | i |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| d | a | d | e | F | e | d | e | r | a | l | D | o | C | e | a | r | á | 0 | 0 | 0 | 0 | 0 | 0 | A | c | a | p | u | l | c | o |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

```
→ [1] Inicializar processo
→ [2] Finalizar processo
→ [3] Alocar memória para processo
→ [4] Acessar conteudo
→ Opção: 2
→ Id do processo: 0
→ Processo 0 finalizado com sucesso
```

*Ao finalizar um processo (desalocar) você não deve apagar o conteúdo das células, apenas marcar os quadros daquele processo como livres. Note que o conteúdo não foi sobreescrito, mas os quadros tornaram-se livres. Graficamente a memória ficaria assim:*

| G | a | m | e | O | f | T | h | r | o | n | e | s | 0 | 0 | 0 | C | h | a | v | e | s | E | m | U | n | i | v | e | r | s | i |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| d | a | d | e | F | e | d | e | r | a | l | D | o | C | e | a | r | á | 0 | 0 | 0 | 0 | 0 | 0 | A | c | a | p | u | l | c | o |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

```
→ [1] Inicializar processo
→ [2] Finalizar processo
→ [3] Alocar memória para processo
→ [4] Acessar conteudo
→ Opção: 4
→ Id do processo: 1
→ Endereço lógico: 10
→ Conteúdo: ‘a’
→ (U)suario ou (A)dministrador?
→ U
→ [1] Inicializar processo
→ [2] Finalizar processo
→ [3] Alocar memória para processo
→ [4] Acessar conteudo
→ Opção: 4
→ Id do processo: 2
→ Endereço lógico: 50
→ Alerta: Acesso ilegal de memória!
```

*O endereço lógico 10 do processo 1 corresponde ao endereço físico 58. Lá está armazenado o caractere ‘a’. O processo 2 possui apenas 26 caracteres (células de memória), logo ele não possui endereço lógico 50, caracterizando acesso ilegal de memória.*

## Considerações Finais

Este trabalho é uma simulação de uma parte do gerenciador de memória. Caso a memória se esgote, não é necessário implementar memória virtual (área de swapping) e os algoritmos de substituição de páginas. Simplesmente mostre uma mensagem de erro indicando que não é possível iniciar o processo ou alocar mais memória. Tenha em mente também que em situações reais a tabela de páginas e a lista de quadros livres seriam armazenadas na memória, mas isso também deve ser desprezado neste exercício.

Será avaliado se a técnica foi implementada usando os conceitos do algoritmo, ou seja, sem "gambiarras" ou que funcione apenas em casos específicos. O código deve estar claro e entendível. Ao final do semestre, durante a apresentação dos trabalhos, este código será explicado pelos integrantes da equipe. Todos do grupo devem explicar uma parte específica e terão suas notas de apresentação individualmente. A nota dada pela implementação é idêntica a todos. A nota do trabalho será a média destas duas notas. Trabalhos copiados receberão nota zero, indepentende se comprove ser o trabalho original. A nota do trabalho vale 25% da nota da Avaliação Parcial 2.