FACULDADE: CENTRO UNIVERSITÁRIO DE BRASÍLIA – UniCEUB
CURSO: ENGENHARIA DE COMPUTAÇÃO
DISCIPLINA: SISTEMAS OPERACIONAIS
CARGA HORÁRIA: 75 H. A. **ANO/SEMESTRE**: 2016/01
PROFESSOR: EDUARDO FERREIRA DOS SANTOS
HORÁRIOS: Terça às 07h20 e Sexta às 09h40

LABORATÓRIO 02 – ALOCAÇÃO E ENDEREÇAMENTO

RESUMO

A memória principal (RAM) é um recurso de fundamental importância que deve ser administrado com cuidado. Apesar do aumento da capacidade computacional de acordo com a Lei de Moore[1], o crescimento do consumo de recursos pelos programas ainda é maior. Para utilizar o recurso com maior eficiência é preciso conhecer o sistema operacional e suas ferramentas. O laboratório apresenta as funções elementares de tratamento de memória e dispõe sobre sua utilidade na computação.

OBJETIVOS

Objetivo Geral
Conhecer e utilizar alocação dinâmica e endereçamento de memória no sistema operacional.

Objetivos Específicos

1. Adquirir familiaridade com a linguagem de programação C;
2. Conhecer os mecanismos de tratamento de dados da linguagem;
3. Aprender a controlar, na perspectiva do programador, a utilização da memória do programa.

PARTE 1 – LISTAS ENCADEADAS

Para exemplificar a alocação dinâmica de memória em sistemas operacionais, vamos utilizar a implementação de de listas encadeadas como estrutura de dados.

> Uma lista encadeada é uma representação de uma sequência de objetos, todos do mesmo tipo, na memória RAM (= random access memory) do computador. Cada elemento da sequência é armazenado em uma célula da lista: o primeiro elemento na primeira célula, o segundo na segunda e assim por diante. [FEOFILOFF, 2016]

Considere as seguintes definições para implementação da lista encadeada:

1 Mais em http://www.hardware.com.br/artigos/lei-moore

PARTE 1 – LISTAS ENCADEADAS

```
struct reg {
      int conteudo;
      struct reg *prox;
};
```

typedef struct reg celula;

Uma célula `c` e um ponteiro `p` para uma célula podem ser declarados assim:

```
celula  c;
celula *p;
```

O algoritmo produz a seguinte implementação:

Figura 1: Representação de lista encadeada [FEOFILOFF, 2016]

Podemos considerar ainda que "o endereço de uma lista encadeada é o endereço de sua primeira célula. Se `le` é o endereço de uma lista, convém dizer simplesmente le é uma lista encadeada".

EXERCÍCIO 01: Considerando a estrutura de dados da célula le, crie três instâncias do objeto célula (três valores na lista);

EXERCÍCIO 02: Construa uma função que imprima todos os valores da lista;

EXERCÍCIO 03: Calcule a quantidade de memória gasta por cada instância da célula.

EXERCÍCIO 2 – INSERÇÃO E REMOÇÃO EM LISTAS ENCADEADAS

Considera-se que a lista encadeada tem uma cabeça quando o primeiro elemento é definido da seguinte maneira:

```
celula *le;
le = malloc (sizeof (celula));
le->prox = NULL;
```

Perceba que, para a cabeça estar vazia, a seguinte condição é necessária:

```
le->prox == NULL
```

EXERCÍCIO 2 – INSERÇÃO E REMOÇÃO EM LISTAS ENCADEADAS

O exemplo a seguir realiza a inserção de um novo elemento na lista encadeada [FEOFILOFF, 2016]. Perceba que, diferentemente de vetores, não é necessário mover e reposicionar elementos:

```
// Esta função insere uma nova celula
// em uma lista encadeada. A nova celula
// tem conteudo x e é inserida entre a
// celula p e a célula seguinte.
// (Supõe-se que p != NULL.)

void insere (int x, celula *p) {
      celula *nova;
      nova = mallocc (sizeof (celula));
      nova->conteudo = x;
      nova->prox = p->prox;
      p->prox = nova;
}
```

Considerando os exemplos apresentados, resolva as seguintes questões:

EXERCICÍO 01: Construa uma função que remove os elementos da lista;

EXERCICÍO 02: Incremente sua função liberando a memória quando um elemento é liberado;

EXERCÍCIO 03: Qual o tamanho de memória utilizado por cada instância do objeto?

EXERCÍCIO 04: Calcule o máximo de elementos possíveis na fila, considerando a memória disponível no computador.

PARTE 3 – GERENCIAMENTO DE MEMÓRIA

EXERCÍCIO: Considere o arquivo CSV disponível no link https://owncloud.eduardosan.com/public.php?service=files&t=953dc55585d2e0b54b4f2aaf8a4b7925

Sabendo que toda estrutura de dados pode ocupar um tamanho máximo na memória, construa um exemplo utilizando listas encadeadas para carregar o arquivo no programa sem erros.

BIBLIOGRAFIA

SILBERSCHATZ, Abraham et al. **Operating system concepts**. Reading: Addison-Wesley, 1998.

TANENBAUM, Andrew S.; MACHADO FILHO, Nery. **Sistemas operacionais modernos**. Prentice-Hall, 1995.

FEOFILOFF, Paulo. **Listas Encadeadas**, 2016. Disponível em http://www.ime.usp.br/~pf/algoritmos/aulas/lista.html Acessado em 26/04/2016.