5187/31 e 6897/1 – Organização e Recuperação de Dados
Profa. Valéria D. Feltrim

2º Trabalho Prático

O 2º trabalho prático relativo à disciplina **Organização e Recuperação de Dados** constitui-se do desenvolvimento de um programa conforme as especificações abaixo.

O programa deverá ser desenvolvido na linguagem C (compilador gcc) e deverá ser feito individualmente ou em duplas.

A entrega do trabalho será feita pelo *moodle* e deverão ser entregues como parte do trabalho:

- relatório (.pdf) de uma página contendo informações sobre: S.O., compilador e IDE utilizados no desenvolvimento, bem como outras decisões de projeto e configurações específicas necessárias para o correto funcionamento do programa;
- código-fonte do programa (arquivos .c e .h (se houver)).

Especificação

O objetivo desde trabalho é criar um sistema de *hashing* extensível (conforme visto em aula) em memória para armazenar chaves numéricas. Para auxiliar no desenvolvimento, o arquivo chaves.txt será disponibilizado no *moodle* junto com esta especificação.

O arquivo chaves.txt contém uma lista de chaves numéricas, uma por linha. Embora todas as chaves do arquivo tenham seis dígitos, não deve haver um tamanho mínimo (em número de dígitos) pré-estabelecido para as chaves. Já tamanho máximo (em número de dígitos) é fixado em seis. Assim, há garantia de que qualquer chave poderá ser armazenada como um número inteiro. Como as chaves são numéricas, usaremos a própria chave como endereço *hash*, i.e., não há necessidade de transformar as chaves antes de fazer a extração dos bits usados para o endereçamento. Embora o arquivo chaves.txt contenha 250 chaves, o sistema deverá ser capaz de lidar com arquivos de chaves de qualquer tamanho, desde que se respeite o formato de uma por linha em modo texto.

O programa deverá ler o arquivo de chaves e fazer a inserção das mesmas, na sequência em que aparecem no arquivo, no *hashing* extensível. O sistema não deve ser permitir a inserção de chaves repetidas (chaves repetidas são ignoradas). Como o programa deverá funcionar para arquivos de chaves de qualquer tamanho, faça a codificação do programa de modo que o tamanho dos *buckets* possa ser facilmente alterado. Por ex., (1) você pode definir uma constante MAX_BK_TAM e usá-la ao longo do código; (2) você pode pedir que o usuário digite o tamanho dos *buckets* antes da inicialização do *hashing*; (3) você fazer com que o programa receba essa informação pela linha de comando. O sistema será testado com arquivos e *buckets* de diferentes tamanhos.

Após todas as inserções, o programa deverá imprimir (em tela ou em arquivo no modo texto) o conteúdo do diretório e dos *buckets*. Para facilitar a visualização, crie um identificador para cada *bucket*. Abaixo é mostrado um exemplo de como essa impressão poderia feita para um *hashing* extensível com *buckets* de tamanho 2 no qual foram inseridas as 10 primeiras chaves do arquivo chaves.txt.

```
Diretory:
dir[0] = bucket #0
dir[1] = bucket #0
dir[2] = bucket #3
dir[3] = bucket #4
dir[4] = bucket #1
dir[5] = bucket #1
dir[6] = bucket #2
dir[7] = bucket #2

Diretory current size = 8
Number of buckets = 5

== Bucket #0 ==
#id = 0   depth = 2
chave[0] = 135220
chave[1] = 123316
```

```
== Bucket #3 ==
#id = 3   depth = 3
chave[0] = 113602
chave[1] = 147242

== Bucket #4 ==
#id = 4   depth = 3
chave[0] = 223462
chave[1] = 184590

== Bucket #1 ==
#id = 1   depth = 2
chave[0] = 213705
chave[1] = 106345

== Bucket #2 ==
#id = 2   depth = 2
chave[0] = 153603
chave[1] = 113603
```

BOM TRABALHO!