ICC 2 – Trabalho 07

Prof. Rodrigo Mello/ Moacir Ponti

Estagiários PAE
Martha Dais Ferreira
Fausto Guzzo da Costa
Gabriela Salvador Thumé

Data Máxima para Perguntas: 10/11/2014

Data de Entrega do Trabalho: 13/11/2014

Fórum para Envio de Perguntas: https://groups.google.com/d/forum/icc2-2014

Descrição

Considere um sistema em que você recebe, como parâmetro, o nome de um **arquivo textual de metadados**. Por exemplo, abaixo é apresentado um exemplo de arquivo de metadados, com seu conteúdo abaixo do nome:

<<metadados.dat>>
filename: registros.reg
key-name: codigo
key-type: int
field-name: nome
field-type: char[80]
field-name: idade
field-type: int
field-name: bolsa
field-type: double

Esse arquivo contem a definição de um arquivo de dados. Seu programa deverá ler os metadados para entender como o arquivo de dados está organizado. Assim, após ler este arquivo, você deverá criar um outro **arquivo binário** chamado *registros.reg* que irá conter registros de acordo com a definição do arquivo de metadados.

O nome <<*metadados.dat*>> **não** é parte do conteúdo do arquivo. Neste caso em particular, seu arquivo deverá suportar registros que tenham os campos: código, nome, idade e bolsa, respectivamente dos tipos *int, char[80], int e double*, conforme descrito arquivo de metadados.

Em seguida, o programa poderá receber os diferentes comandos:

1) **insert**: utilizado para inserir dados no arquivo registros.reg tal como segue:

insert 4, “João Mário”, 15, 324.54
insert 1, “Pedro Carlos”, 21, 56.34

insert 2, “Marcelo Isidro”, 34, 673.23

Nesse exemplo, 3 registros são inseridos no final do arquivo, utilizando os dados descritos. Assuma inicialmente que os comandos insert estão sempre corretos, ou seja, que os campos e seus respectivos tipos estão corretamente descritos.

2) **index**: esse comando deve gerar um arquivo com o mesmo nome do arquivo de dados, ou seja, neste caso *registros.reg* contudo com outra extensão. Será usada a extensão *idx*. Sendo assim você deve criar um **arquivo binário** chamado *registros.idx* que conterá, para cada registro no arquivo *registros.reg:*

- os valores do campo chave (key) do arquivo *registros.reg,* e
- o offset (deslocamento em bytes) do registro nesse arquivo.

Neste exemplo, como cada registro é formado por um int, char[80], int e double, você terá registros com 96 bytes. Sendo assim, o primeiro registro inserido, i.e., o de chave 4 estará no offset 0, o registro de chave 1 estará no offset 96 e o registro de chave 2 estará no offset 192. Dessa maneira, temos inicialmente os seguintes pares <<chave>> e <<offset>>:

<<chave>>	*<<offset>>*
4	*0*
1	*96*
2	*192*

Note que os registros estão ordenados pelo offset, pois essa é a ordem original do arquivo de dados. O programa deverá ordenar esses dados, de maneira crescente, de acordo com as chaves, obtendo:

<<chave>>	*<<offset>*
1	*96*
2	*192*
4	*0*

Este conteúdo deverá ser gravado no arquivo *registros.idx*. OBS: o cabeçalho indicado no exemplo não existe no arquivo. Ele é usado aqui apenas para demonstrar o que significa cada coluna.

3) **search**: comando que realiza uma busca binária utilizando o arquivo *registros.idx*. Devido a isso a busca é sempre feita pela chave. Ao encontrar a chave no arquivo de índice, deve-se obter o offset do registro correspondente, abrir o arquivo *registros.reg* e imprimir seus dados. Por exemplo, ao receber:

search 2

Você irá procurar pela chave 2, utilizando busca binária, em *registros.idx* e localizar o offset 192. Em seguida deve abrir o arquivo *registros.reg* e carregar o registro relativo ao offset 192 e imprimí-lo como segue:

2\n
Marcelo Isidro\n
34\n
673.23\n

Em caso de busca sem sucesso não imprimir nada na tela. Note que não será possível fazer a busca sem a criação do índice antes. Caso uma busca seja requerida e não exista um índice, você deve automaticamente chamar o comando *index* para gerar o item.

Note ainda que o índice poderá ficar desatualizado. Por exemplo, se for feita uma nova inserção e o índice não for re-gerado, a busca pode resultar sem sucesso. Esse problema será tratado num próximo trabalho. Para esse trabalho, assuma que não serão feitas novas inserções após a geração do índice.

4) **exit**: Este comando indica que nenhum outro comando adicional será recebido. Assim seu programa poderá finalizar.

Dicas

1) Quais os tipos serão definidos no arquivo de metadados?

int, double, char, char[número] e float

2) Qual comando da linguagem C posso utilizar para saber qual o offset atual de um arquivo?

ftell

3) Qual comando da linguagem C posso utilizar para posicionar em um offset do arquivo?

fseek

4) O arquivo de metadados poderá ter qualquer formato de registro desde que tenha um campo chave?

Sim.

5) Mas como faço para criar dinamicamente um registro que represente o conteúdo do arquivo de metadados?

Isto é parte do trabalho. Pense um pouco.

Motivação

Este trabalho tem grande relação com as disciplinas de Organização de Arquivos e Banco de Dados que serão vistas nos próximos semestres. Ao implementá-lo, você estará melhor preparado para os próximos semestres.

Formato para os Casos de Teste

Formato da entrada:
<nome do arquivo de metadados>
<comando 1>
<comando 2>
...
<comando n>
<exit>

Formato da saída:
<impressão 1>
<impressão 2>
...
<impressão k>

Exemplo de Caso de Teste

Entrada fornecida:
metadados.dat
insert 4, "João Mário", 15, 324.54
insert 1, "Pedro Carlos", 21, 56.34
insert 2, "Marcelo Isidro", 34, 673.23
search 2
search 1
exit

Saída esperada:
2
Marcelo Isidro
34
673.23
1
Pedro Carlos
21
56.34

Informações Importantes

1) Um dos objetivos da disciplina de ICC2 é o aprendizado individual dos conceitos de programação. A principal evidência desse aprendizado está nos trabalhos, que são individuais neste curso. Você deverá desenvolver seu trabalho sem copiar trechos de código de outros alunos ou da Internet, nem codificar em conjunto. Portanto, compartilhem ideias, soluções, modos de resolver o problema, mas não o código.

1.1) O plágio vai contra o código de ética da USP;

1.2) Quando autores e copiadores combinam, estão ludibriando o sistema de avaliação, por isso serão utilizadas ferramentas de análise plágio tais como o MOSS (http://moss.stanford.edu/);

1.3) O trabalho em grupo e a cooperação entre colegas é em geral benéfico e útil ao aprendizado. Para ajudar um colega você pode lhe explicar estratégias e ideias. Por exemplo, pode explicar que é preciso usar dois loops para processar os dados, ou que para poupar memória basta usar uma certa estrutura de dados, etc. O que você não deve fazer é mostrar o seu código. Mostrar/compartilhar o código pode prejudicar o aprendizado de seu colega:

1.3.1) Depois seu colega ver seu código, será muito mais difícil para ele imaginar uma solução original e própria;

1.3.2) O seu colega não entenderá realmente o problema: a compreensão passa pela prática da codificação e não pela imitação/cópia.

1.4) Um colega que tenha visto a sua solução pode eventualmente divulgá-la a outros colegas, deixando você numa situação muito complicada, por tabela;

1.5) O texto acima foi baseado e adaptado da página http://www.ime.usp.br/~mac2166/plagio/, da qual recomendo a leitura completa.

2) Todos os códigos fontes serão comparados por um (ou mais) sistema(s) de detecção de plágio, e os trabalhos com alta similaridade detectada terão suas notas zeradas, tanto aqueles relativos ao código de origem quanto do código copiado. A detecção de plágio será reportada à Seção de Graduação para providências administrativas;

3) A avaliação incluirá a porcentagem de acertos verificada pelo Sistema de Submissão de Trabalhos e também a análise do seu código, incluindo identação, comentários, bom uso da memória e práticas de programação. Portanto faça seu código com cuidado, da melhor forma possível.

Sobre o sistema de submissão:

1. Seu código deverá incluir arquivo fonte .c .h e Makefile – todos os arquivos deverão obrigatoriamente conter no início um comentário com seu nome, número USP, turma e data da entrega do trabalho;

2. A data/hora de entrega do trabalho é aquela estipulada no sistema. Trabalhos entregues por email NÃO serão aceitos, mesmo que dentro da data/hora estipulada. Faça seu trabalho com antecedência para evitar entregar em cima da hora e ter problemas de submissão;

2.1. A submissão é de responsabilidade do aluno, e os problemas comuns à entrega próxima ao fechamento do sistema também. Portanto: problemas de acesso à rede não serão aceitos como desculpa para entrega por email ou fora do prazo;

3. A compilação e execução do código é feita no sistema pelos comandos:

make all
make run

4. A saída do seu programa deve ser exatamente igual à saída esperada, incluindo: espaços em branco, quebras de linha e precisão decimal;

5. Há um limite em segundos para a execução dos casos de teste e um limite de memória total para ser utilizado. Você deverá gerenciar bem o tempo de execução e o espaço ocupado para que seu programa fique dentro desses limites, para evitar uso excessivo, pois o sistema irá invalidar o caso em que o limite foi excedido;

6. Ao enviar, aguarde sem recarregar a página nem pressionar ESC, para obter a resposta. Caso demore mais do que 2 minutos para dar uma resposta, feche e abra novamente a página do servidor. Verifique se seu programa tem algum problema de entrada de dados (ou se tem algum loop infinito ou parada para pressionamento de tecla). Caso não tenha, aguarde alguns minutos e tente novamente;

7. O erro de “Violação de Memória” significa acesso indevido a arranjo ou arquivo. Use compilação

com g e o programa valgrind para tentar detectar a linha de código com erro.

7.1. Exemplo 1 de violação de memória:

```
int **mat = (int **)malloc(sizeof(int *) * 3);
mat[0] = (int *)malloc(sizeof(int)*10);
for (i = 1; i < 3; i++) {
   free(mat[i]);
}
// apenas a posicao 0 de mat foi alocada as outras nao
// portanto esta liberando regiao nao alocada
// gerando violacao de memoria
```

7.2) Exemplo 2 de violação de memória:

```
int B[5] = {5, 6, 7, 8, 9};
int N = 5;
int *A = malloc(N*sizeof(int));
int j = 0, i = 1; // 'j' inicializado em 0, 'i' inicializado em 1
while (j < N){
A[i] = B[j]; // 'j' e 'i' sao indices diferentes
j++;         // quando j = (N1) i = (N1)+1 e
i++;
     // havera escrita indevida em A
}
```