Projeto II: Utilizando Arquivos Invertidos para Suporte à Busca Textual Indexada sobre Manpages Unix

Requisitos

Como foi visto em aula, as formas de indexação de arquivos utilizando listas são muito úteis para indexação por chave secundária. Quando os dados se encontram na forma de arquivos-texto contendo palavras-chave ou referências por metadados, então a Lista Invertida ou Arquivo Invertido é a forma de representação de nossa escolha.

Neste trabalho você implementará um sistema de recuperação de informações de um banco de dados contendo *MANPAGES* do Unix/Linux/FreeBSD. Para tanto você utilizará o arquivo .tgz linkado adiante, o qual contém todas as *MANPAGES* do SunOS/Solaris (um Unix) na forma de arquivos-texto (a formatação e os comandos de indentação, etc foram excluídos com *troff -a*).

O software que você vai desenvolver deverá satisfazer os seguintes requisitos:

Dados:

- Todo o conteúdo de todas as *MANPAGES* será armazenadas na forma de registros em um arquivo de dados que será chamado de **`manpages.dat`**.
    - O texto completo de cada *MANPAGE* será armazenado em um campo de um registro, o campo **`conteudo`**.
    - O registro deverá conter, além disso, um campo para a **chave primária**, o campo **`comando`**, que será o COMANDO descrito na manpage (nome do arquivo menos a extensão .txt).
    - O registro poderá conter outros campos que o programador considerar necessários.
- Todos os índices secundários referenciarão o campo texto como um todo, sem se preocupar com onde no texto esteja a palavra-chave indexada.

Indexação

- Manter um índice primário em arquivo separado e que referencia as MANPAGES por nome, ex: **`"fseek"`**.
    - Este índice deve ser baseado em uma estrutura em árvore para indexação primária que você já programou como a **árvore-B, R-B** ou **AVL**.
    - Observe que a chave primária será um string, significando que você deverá ser capaz de calcular precedência entre strings, indicando, por exemplo, se `fclose` *precede* ou *sucede* `fseek`.

- Manter um conjunto de índices de chave secundária na forma de **arquivo invertido**.
    - Este índice indexará o conjuntode todas as palavras contidas em todas as manpages que não caiam nas seguintes categorias léxicas: *artigo, conjunção, preposição e pronome* (chamadas em seu conjunto de *conetivos*).
- Você deverá percorer todas as manpages (o que pode ser durante a inclusão de uma manpage no arquivo de dados ou a posteriori, durante um processo de indexação) e:
    - incluir no índice invertido toda palavra que ainda não estiver nele
    - criar uma entrada para as palavras que já estiverem indexadas

O software desenvolvido deverá ser capaz de realizar as seguintes buscas:

- Por chave primária, retornando o texto da MANPAGE para um nome-de-comando ou a mensagem de que uma MANPAGE com aquele nome não existe (3 pontos)
- Busca simples por chave secundária, retornando todas as MANPAGES que contém uma determinada chave secundária, ex.: *filename* (4 pontos)
- Busca conjuntiva com duas chaves secundárias, retornando todas as MANPAGES que contenham ambas as chaves. Ex.: *filename* e *string* (3 pontos)

Para satisfazer estes requisitos o sistema deverá gerenciar um arquivo de dados, um arquivo de índices de chave primária e tantos arquivos de índices de chave secundária quantos forem necessários. Toda a indexação deverá obrigatoriamente ser realizada em arquivos em disco.

Não será aceito trabalho que utilizar a memória principal do computador para qualquer outro gerenciamento de dados de indexação que não a geração de tabelas temporárias em buscas conjuntivas.

Lembre-se que para a pesquisa conjuntiva você deve gerar uma tabela temporária. Esta pode ser gerada na memória como um vetor. Para realizar a pesquisa comparativa você pode usar o algoritmo de Pesquisa de Co-Ocorrência em duas Listas Não-Ordenadas.

A entrada dos dados deverá ser realizada da seguinte forma:

- Cada arquivo-texto de uma MANPAGE é lido e armazenado em um registro, cujo endereço é guardado
- Percorre-se o texto (na memória) palavra a palavra e verifica-se, para cada palavra, se ela se encontra na lista de palavras-chave.
- Para cada palavra-chave encontrada, atualiza-se uma lista de endereços de chave secundária com o endereço do registro
- Finalmente o índice de chave primária e alimentado.

FAQ

Pesquisa de Co-Ocorrência em duas Listas Não-Ordenadas

O algoritmo abaixo pressupõe que as listas a serem pesquisadas são armazenadas em vetores, sendo chamadas `lista1` e `lista2`, sendo que `lista1` é a maior das duas. Se você usar listas encadeadas a referência `lista_n[posição]` deve ser substituída pela função que retorna o próximo elemento da lista.

```
tipoLista: função coOcorrencia( lista1, lista2: tipoLista )
VARIÁVEIS
    INTEIRO i, j;
    tipoLista saida;
INÍCIO
    PARA i de 1 até tamanho(lista1) FAÇA
        procurado <- lista1[i];
        PARA j de 1 até tamanho(lista2) FAÇA
            SE lista2[j] = procurado ENTÃO
                adicione(saida, procurado);
            FIM SE
        FIM PARA
    FIM PARA
    RETORNE( saída );
FIM coOcorrencia
```

Lendo arquivo texto até o fim em C++

OPÇÃO 1: Você pode determinar o tamanho do arquivo do arquivo usando stat e depois usar fread para ler um bloco do tamanho do arquivo para dentro de um vetor de char.
Use uma variável st definida como
struct stat st;
Invoque stat assim:
stat(filename, &st) Use o campo st.st_size;

OPÇÃO 2: Você pode ler o arquivo linha a linha para dentro de um vetor de char grande o suficiente

usando fgets, colocando a leitura dessas linhas dentro de um laço onde você testa com feof o fim do arquivo
para ver se chegou ao fim antes de ler a próxima linha.

OPÇÃO MENOS INTELIGENTE: Use `fseek` com a constante SEEK_END para posicionar o file pointer no
fim do arquivo. Aí use `ftell` para determinar em que posição está. Depois posicione no começo

de novo e
leia um número de bytes igual ao resultado obtido com `ftell` usando

fread: fseek(file, 0L, SEEK_END); tamanho = ftell(file);

Como saber os nomes de todos os arquivos de MANPAGES?

Use o sistema operacional para isso: Existem dois parâmetros-padrão da função main, argc e argv:

- argc (**argument count**) indica quantos argumentos de linha de comando foram passados ao programa. O valor default é 1 pois o programa em C sempre recebe seu próprio nome como parâmetro. Se o valor for maior do que um é porque foi passada alguma coisa a mais.
- argv (**argument vector**) é um vetor de ponteiros para strings que contém cada um dos argumentos passados como parâmetro. Os critérios de separação de argumentos são os do scanf: tudo o que for separador gera argumentos separados. Tipicamente utilizamos espaços em branco.
- **Como eu posso entender o argv?** Simples: é uma cópia da linha de comando passada ao programa. Se você passou uma linha de comando com wildcards (*) ou uma expressão regular, o sistema operacional vai expandir a linha de comando substituindo os wildcards ou resolvendo a expressão regular antes de passá-la ao programa "C"/C++. Assim se você tem um programa chamado banana e chama o programa com a linha de comando "banana *.txt" o Unix/Linux vai resolver esta expressão substituindo na linha de comando o "*.txt" pelo nome de todos os arquivos .txt daquele direório e passando esta lista como string separado por espaços ao programa. ARGV pode ter a seguinte cara: "banana abacaxi.txt melao.txt limao.txt", se houver três arquivos, por exemplo.

Um típico código usando argc e argv é esse (programa testeparam.c):

```
#include <stdio.h>
main (int argc, char* argv[])
{
   int i;
   printf("\n Arquivos: %d \n", argc - 1);
   for (i=1; i < argc; i++)
     { printf("--> %s\n", argv[i]); }
}
```

Esse código serve para imprimir todos os nomes de arquivo de um diretório ou para imprimir qualquer coisa que você passe como parâmetro. Observe que eu estou iniciando o contador i em 1 porque argv[0] é o nome do programa e não me interessa.

Se eu fizer: `venus{92}/home/professores/awangenh> ./testeparam *.gz`

Arquivos: 8 --> Cyclops-DMI.st.gz--> Cyclops-ErrorHandling.st.gz--> Cyclops-funca-DMI.st.gz--> Cyclops.ppt.gz--> cyclopsDicomUS.xml.gz--> cyclopsErrorMessagesBR.xml.gz--> materialalden.html.gz--> Page.ws.gzvenus{93}/home/professores/awangenh> Se eu fizer:

venus{93}/home/professores/awangenh> ./testeparam banana abacaxi

Arquivos: 2
--> banana
--> abacaxi
venus{94}/home/professores/awangenh>