Estatísticas e Informações .DIN – Parte 1

Andrey Mota, Bruno Caike e Gabriel Cardoso

FACOM – Faculdade de Computação

UFMS -Universidade Federal de Mato Grosso do Sul

RESUMO

Este relatório descreve um programa escrito na linguagem C, que analisa e extrai informações de um arquivo no formato DIN. Este formato consiste em registros que são escritos separadamente em cada linha do arquivo, sendo assim compostos por: um comando (que pode ser uma busca, uma leitura ou uma escrita que será realizada pelo processador) e um endereço (conjunto de bytes que indicam o espaço da memória que será utilizado). O retorno ao usuário apresenta a quantidade de leituras de dados, escritas de dados e buscas de instruções realizadas na memória, a quantidade total de acessos a ela (soma de todas os acessos) e a quantidade de total de acesso aos dados (Leituras + Escritas).

Palavras-chave: Programa. Formato. Registros. Comando. Endereço.

ABSTRACT

This report describes a program written in the C language, which analyzes and extracts information from a file in DIN format. This format consists of records that are written separately in each line of the file, being composed by: a command (which can be a search, a reading or a writing that will be performed by the processor) and an address (set of bytes that indicate the memory space to be used). The return to the user shows the number of data readings, data writes and instructions searches performed in the memory, the total number of accesses to it (sum of all accesses) and the total amount of access to data (Readings + Writings).

Keywords: Program. Format. Records. Command. Address.

INTRODUÇÃO

Um arquivo em formato DIN é capaz de representar registros de operações realizadas por um processador. Cada registro é escrito em uma linha, que é dividida através de um espaço, em dois campos: o tipo de acesso e o endereço.

- Tipo de acesso: Indica o tipo de operação que será realizada pelo processador através um numéro inteiro que varia entre 0 e 2. O número 0 define que será feita uma leitura de um dado já existente na memória, o numero 1 indica que será realizada uma escrita de um novo dado na memória, enquanto o numero 2 define uma busca de instrução que também está armazenada na memória.

- Endereço: É um conjunto de bytes na base hexadecimal que define o endereço de memória que será utilizado na operação a ser realizada. Este número vai variar entre 0 e o endereçamento máximo do processador, que é definido através da quantidade de endereços de memória que o mesmo possui. Logo, este campo é capaz de receber um valor imensurável, devido a quantidade de endereços existentes nos processadores atualmente e na necessidade de seus fabricantes em aumentá-la constantemente com o surgimento de novas tecnologias.

DESENVOLVIMENTO

Para entendermos o funcionamento do programa desenvolvido, é importante entender também a forma como um computador realiza a leitura de um arquivo qualquer, independente do formato e da linguagem de programação que será utilizada.

Primeiramente é necessário definir um espaço de memória que será utilizado pelo programa para armazenar temporariamente o arquivo, possibilitando a manipulação do conteúdo existente no mesmo. Portanto, durante o desenvolvimento foi utilizado um tipo de ponteiro especial para arquivos (FILE), disponibilizado em uma biblioteca padrão da linguagem C (stdio.h), que também foi utilizada em todas as interações com o usuário (através das funções printf e scanf).

Após alocar o espaço de memória que será utilizado, é preciso definir o arquivo que será aberto nele. Para isso foi utilizada uma função existente na mesma biblioteca (fopen), que recebe 2 paramêtros de entrada para abertura de um arquivo, o primeiro é o nome do mesmo e o segundo define a forma como ele será aberto, neste caso, sendo necessária a abertura apenas como leitura, inserindo o caractere “r” (reading), afinal não será realizada nenhuma outro tipo de operação no arquivo de entrada, apenas a manipulação dos dados que já existem nele.

A interação mais viável escolhida na hora de obter o nome do arquivo para conseguir manipular ele, foi pedir ao usuário que digite o nome do mesmo. No entanto, isso gera algumas complicações e erros que deverão ser tratados, afinal o programa irá exigir que, o arquivo a ser lido esteja no mesmo diretório em que o código fonte, o usuário digite corretamente o nome do arquivo e que o programa saiba reconhecer o formato do arquivo mesmo que uma extensão não sera inserida.

Os dois primeiros problemas encontrados durante a abertura do arquivo, dependem exclusivamente da atenção do usuário, já o terceiro foi resolvido com o auxílio de duas funções da biblioteca string.h, biblioteca padrão da linguagem C. A função strstr compara as duas strings e verifica se a extensão DIN foi digitada junto ao nome do arquivo, caso não tenha sido, a função strcat, que concatena duas Strings, irá adicionar a extensão ao final do nome dele. Caso o usuário digite um nome incorreto, ou ainda, o arquivo seja de uma extensão diferente da especificada e o nome dele passe a ter duas extensões, não será possível abri-lo. Isso é tratado pelo programa como um erro de nome de arquivo inválido e o usuário será alertado, podendo optar por encerrar o programa ou tentar digitar um nome diferente para ser lido, sendo necessário refazer todo o processo de abertura do mesmo.

Assim que encerrada a utilização do arquivo, o programa irá fechar o arquivo que foi aberto, para isso basta utilizar a função contrária a fopen, que se chama fclose e recebe como parâmetro apenas o nome do arquivo a ser fechado. No entanto isso será feito somente no final de cada execução do programa, após todas as informações necessárias forem utilizadas pelo mesmo, possibilitanto assim que caso o usuário solicite seja aberto um outro arquivo em sua próxima execução.

Entendendo então a forma como o arquivo é manipulado pelo computador, para ser possível que ele interprete um arquivo no formato DIN é necessário fazer com que ele armazene o conteúdo de cada linha que possua um registro em duas partes, conforme os dois campos existentes no mesmo (tipo de acesso e endereço).

	Exemplo 1	Exemplo 2	Exemplo 3
Conteúdo da linha	0 1014d5b4	1 7fff009c	2 430cfc
Tipo de Acesso definido	0 (Leitura)	1 (Escrita)	2 (Busca)
Endereço definido	1014d5b4	7fff009c	430cfc

Esse vai ser portanto o primeiro objetivo do programa com o intuito de interpretar as informações do arquivo. Para fazer isso, pode ser utilizado um vetor de caracteres (String) para armazenar todo o conteúdo de cada uma das linhas. Com isso, utilizando outra função da biblioteca stdio.h (fgets), podemos obter o conteúdo de cada uma das linhas de um arquivo e armazená-las temporiamente.

No entanto, nos deparamos com um problema ao tentar fazer isso, afinal como já observado, o valor que um endereço receberá é imensurável, logo não é possível definir um tamanho para o vetor que receberá o conteúdo de cada linha (quantidade de espaços de memória que irá utilizar).

Conseguimos resolver isso através do *BUFSIZE,* que é um conjunto de dados que especificará o tamanho do vetor que armazenará o conteúdo das linhas. Isso acontece porque quando o limite definido inicalmente for atingido, o BUFSIZE aumentará seu valor e consequentemente o espaço de memória reservado para o vetor. A desvantagem de fazer isso é que o valor reservado inicialmente por ele, na maioria dos exemplos apresentados será muito maior do que o utilizado verdadeiramente, causando assim um grande desperdício de memória.

Com o conteúdo de cada linha sendo armazenado temporariamente, ainda é necessário dividi-los nos dois campos existentes. Utilizando outra função da biblioteca string.h (strtok), quando for encontrado um espaço (“ “) podemos dividir o conteúdo em duas Strings distintas e assim armazená-las em um espaço da memória reservado por dois ponteiros, nomeados conforme a nomenclatura padrão de definição de variáveis (tipoDeAcesso e endereco).

Por fim, ao analisar cada linha do arquivo, o programa irá verificar o tipo de acesso definido e realizará um incremento na contagem de leituras, escritas e buscas. A partir destes valores é possível obter também o número de acessos aos dados (soma de leituras e escritas) e o total de acessos à memória (soma de leituras, escritas e buscas). Após isso apresenta as informações ao usuário e verifica se ele deseja realizar outra leitura ou encerrar o programa

Alem de tudo, é importante considerar a entrada de arquivos no formato DIN escritos de maneira incorreta, não seguindo as especificações do formato. Podemos indicar erros como:

- **Quantidade incorreta de campos:** Caso exista alguma linha do arquivo que tenha menos do que dois campos, significa que um deles não foi inserido ou que possa ter ocorrido um erro de escrita e não tenha sido dado espaço entre eles.

- **Tipo de acesso inválido:** Mesmo que uma linha possua dois campos não significa que ela seja correta, afinal o tipo de acesso que será realizado deve estar representado como um número inteiro entre 0 e 2.

- **Linhas vazias:** A existência de linhas vazias em um arquivo podem ser uma complicação na hora de interpretá-lo, afinal elas não podem ser consideradas uma linha comum pois não existem campos em seu conteúdo, no entanto também não podem ser consideradas como um término de leitura do arquivo pois podem existir linhas abaixos dela que deverão ser interpretadas.

Buscando evitar ao máximo a apresentação de erros ao usuário, estes casos apresentados serão considerados em cada linha do arquivo, e mesmo que sejam encontrados, toda a leitura até a última linha será realizada. O usuário será alertado de todas as linhas que estejam incorretas e nenhum valor encontrado nelas será incrementado ao resultado das contagens. Caso um erro diferente seja encontrado, o usuário será alertado e o programa será finalizado.

CONCLUSÃO

O desenvolvimento de um protótipo capaz de ler arquivos no formato DIN sem interações com o usuário e interpretação de erros foi realizado sem muitas dificuldades devido a conhecimentos anteriores com relação a algoritmos e programação.

No entanto, para garantir um funcionamento correto do programa independente das entradas de seu utilizador ou do arquivo que for inserido (que pode conter erros em sua estrutura) foi necessário analisar cuidadosamente os possíveis erros e procurar soluções viáveis para resolvê-los, sendo também utilizados conceitos teóricos da computação adquiridos em matérias anteriores.

Apesar dessas dificuldades encontradas durante o processo de desenvolvimento do programa, considerando que ele esteja escrito como o formato exige e seu nome seja passado corretamente, o processo de interpretação e apresentação dos resultados será um sucesso. Caso contrário, os erros conhecidos serão tratados e apresentados ao usuário, ainda assim possibilitando que o mesmo utilize-o sem dificuldades.

Mesmo o programa tendo sido escrito em uma linguagem estruturada (linguagem C) e sendo utilizadas bibliotecas padrões da mesma, suas definições podem ser facilmente implementadas utilizando outras linguagens e bibliotecas, e inclusive serem facilitadas e aprimoradas por elas.

REFERÊNCIAS BIBLIOGRÁFICAS

Bibliotecas de funções em C, *Paulo Feofiloff. Disponível em: <https://www.ime.usp.br/~pf/algoritmos/apend/interfaces.html>*

Declaração de ponteiros, Alexandre Tachard Passos. Disponível em: <https://www.ic.unicamp.br/~tachard/mc102/0921.html>

Lidando com arquivos na programação, *Eduardo Casavella. Disponível em:* <http://www.amelek.gda.pl/avr/uisp/srecord.htm>

Programming Tips in C (BUFSIZE). Disponível em:

<https://www.cs.uky.edu/~raphael/programming.html>

Exemplos de arquivos no formato DIN, Dinero-IV_traces. Disponível em:

<https://github.com/priyankarroychowdhury3/Dinero-IV_traces>