Projeto de Arquitetura de Computadores

28 de junho de 2021

UNIVERSIDADE DE ÉVORA

Discentes:

Sara Amaral nº48563
João Condeço nº48976

Docentes:

Miguel Barão
Pedro Salgueiro

Departamento de Informática, Universidade de Évora

Conteúdo

1 Introdução

Este trabalho prático tem como objetivo conciliar os conhecimentos adquiridos na cadeira de Arquitetura de Computadores 1 para desenvolver o editor de texto da terminal "ed".

Este programa contempla 8 funções que operam com base na informação informação armazenada num ficheiro de apontadores. Estas funções podem acrescentar, alterar e/ou remover informação e, após efetuadas todas as operações desejadas, se o utilizador desejar, salvam a informação num ficheiro txt.

Numa primeira instância iremos fazer uma detalhada explicação do funcionamento do programa, do ponto de vista do utilizador, bem como das funções utilizadas para a realização do mesmo. Serão também abordadas as escolhas feitas ao longo da elaboração deste trabalho bem como das dificuldades sentidas na mesma.

Para este trabalho o software utilizado foi:

- Simulador RARS

2 O programa e tomadas de Decisão

Inicialmente existiu a necessidade de escolher entre as sugestões dadas no enunciado relativamente à estrutura de dados a usar para armazenar o conteúdo sob o qual o programa opera. Devido a uma maior versatibilidade e facilidade para minimizar o movimento de dados ao longo do array, foi seguida a segunda sugestão. Assim implementámos uma estrutura de dados com um intervalo vazio no meio (gap) onde serão inseridas todas as linhas novas, sendo a linha atual (pos) a ultima linha inserida antes da gap. Foi também criado uma variável (end) com o primeiro endereço vazio no array a contar do fim, com a finalidade de saber para onde mover os dados que se passam para a zona de baixo do array.

Em termos de funcionamento generalizado o programa opera através de um ciclo while existente na função main que recebe o input do user, analisa-o através da função FindDaWae e, através desta, invoca a função desejada (detetada através da função FindFunction). Na função FindDaWae, a função stop analisa se o utilizador inseriu valores numéricos (ou os caracteres especiais '%' e '$') relativamente às linhas desejadas e armazena os mesmos em variáveis globais. No caso das funções edit, write e file é também invocada a função GetOrigin para armazenar o diretório do ficheiro inserido. A função FindDaWae garante a constante atualização das variáveis pos, end, saved consoante o tipo de função invocada.

Para minimizar erros de utilização por parte do utilizador sempre que o programa encontra um input inválido ou é solicitada uma operação não disponivel nas circunstancias atuais (como por exemplo a função quit quando o ficheiro não está guardado) o programa dá output de "?".

Para terminar o programa, caso seja detetado como input as letras 'q' ou 'Q' é invocada a função auxiliar exit (no caso do "q", isto apenas ocorre se o ficheiro tiver guardado) e o programa termina.

3 Funcionalidades

3.1 Insert

A função insert (invocada pelo comando "i") tem como objetivo inserir texto antes da linha atual do ficheiro ou antes da linha pretendida pelo utilizador (caso indicada nos argumentos) e define a linha atual como a ultima linha inserida.

Esta função recebe como argumentos:

1. O endereço do array onde é guardado o conteúdo do ficheiro;
2. O endereço da linha atual no array;
3. O endereço da primeira posição vazia no array a contar do fim;
4. O endereço do valor X;
5. O endereço do valor Y;
6. O valor da variável global TRUECHAR;

O insert pode ser utilizado das seguintes maneiras:

- 'i' - Será inserido o texto desejado pelo user antes da linha armazenada na variável POS, para tal é movimentada a linha do endereço indicado por pos para o final do array .
- 'xi' - Onde x corresponde a um número inserido pelo utilizador. Serão inseridas linhas antes da linha número x. Deste modo ter-se-á que deslocar as linhas entre x e pos para a zona de baixo do array caso a linha x se localize acima de pos, deslocar para baixo a linha atual caso x corresponda a pos ou deslocar para cima as linhas entre end+1 e x-1 para a parte de cima do array caso a linha x se localize abaixo de pos. Seguidamente faz-se a inserção de texto no endereço a seguir à nova POS que terá sido alterada corretamente com os movimentos feitos previamente.
- 'x,yi' - Onde x e y correspondem ao um intervalo de linhas pretendido pelo utilizador. Neste caso deverá ser ignorado x e proceder como se o input fosse "yi", caso desenvolvido num ponto anterior.
- '$i' - Insere antes da última linha escrita. Para identificar qual é esta linha é utilizada a função auxiliar LastIndex, seguidamente far-se-ão os movimentos necessários através da função moveUp.
- 'x,$i' - Caso semelhante ao anterior.
- '%i' - É, de igual modo, inserido texto antes da última linha. Semelhante aos casos anteriores.

  É de realçar que para diferenciar entre as linhas número 36 e 37 e o código ASCII de $ e % é utilizada a variável TRUECHAR que possuirá o valor 1 caso se trate dos referidos caractéres especiais.

3.2 Append

A função append (invocada pelo comando "a") tem como objetivo inserir texto depois da linha atual do ficheiro ou depois da linha pretendida pelo utilizador (caso indicada nos argumentos) e define a linha atual como a ultima linha inserida.

Esta função recebe como argumentos:

1. O endereço do array onde é guardado o conteúdo do ficheiro;
2. O endereço da linha atual no array;
3. O endereço da primeira posição vazia no array a contar do fim;
4. O endereço do valor X;
5. O endereço do valor Y;
6. O valor da variável global TRUECHAR;

O append pode ser utilizado das seguintes maneiras:

- 'a' - Será inserido o texto desejado pelo user depois da linha armazenada na variável pos.
- 'xa' - Onde x corresponde a um número inserido pelo utilizador. Serão inseridas linhas depois da linha número x. Deste modo ter-se-á que deslocar as linhas entre x+1 e pos para a zona de baixo do array caso a linha x se localize acima de pos, caso x corresponda a pos não serão feitos movimentos ou deslocar para cima as linhas entre END+1 e x para a parte de cima do array caso a linha x se localize abaixo de pos. Seguidamente faz-se a inserção de texto a seguir a pos que terá sido alterada corretamente com os movimentos feitos previamente.
- 'x,ya' - Onde x e y correspondem ao intervalo de linhas pretendido pelo utilizador. Neste caso deverá ser ignorado x e proceder como se o input fosse "yi", caso desenvolvido no ponto anterior.
- '$a'- Insere depois da última linha escrita. Para identificar qual é esta linha é utilizada a função auxiliar LastIndex, seguidamente far-se-ão os movimentos necessários através da função moveUp (movimentar-se-ão todas as linhas para a zona de cima do array).
- 'x,$a' - Caso semelhante ao anterior.
- '%a' - É, de igual modo, inserido texto depois da última linha. Semelhante aos casos anteriores.

3.3 Change

A função change (invocada pelo comando "c") tem como objetivo modificar um intervalo de linhas inserido pelo utilizador e define a linha atual como a ultima linha inserida.

Esta função recebe como argumentos:

1. O endereço do array onde é guardado o conteúdo do ficheiro;
2. O endereço da linha atual no array;
3. O endereço da primeira posição vazia no array a contar do fim;
4. O endereço do valor X;
5. O endereço do valor Y;
6. O valor da variável global TRUECHAR;

O change pode ser utilizado das seguintes maneiras:

- 'c' - Será modificada a linha atual (POS) pelo texto inserido pelo user. É atualizada a posição atual para a última linha inserida.
- 'xc' - Onde x corresponde a um número inserido pelo utilizador. Será modificada a linha x pelo texto inserido pelo user. Antes do utilizador inserir o testo desejado serão feitos os movimentos de linhas devidos. É atualizada a posição atual para a última linha inserida.
- 'x,yc' - Onde x e y correspondem ao um intervalo de linhas pretendido pelo utilizador. Neste caso dever-se-á fazer delete do intervalo de linhas apresentado e, caso este intervalo inclua a última linha fazer um append do texto a ser inserido pelo utilizador, caso contrário deverá ser feito um insert antes da linha atual. Isto deve-se a propriedades da função delete: quando se faz delete da última linha a posição atual passa a ser a linha anterior àquela(s) apagada(s), quando não se faz delete da última linha a posição atual passa a ser a linha a seguir àquela(s) apagada(s). Isto justifica a necessidade de diferenciar entre as situações onde se deverá realizar um append e aquelas onde se deverá realizar um insert.
- '$' - Modifica a última linha apenas, deverá ser feito um moveUp e inserido texto após a nova posição atual.
- 'x,$c' - Change de um intervalo que inlcui a última linha, abordado em casos anteriores.
- '%c' - Change de todas as linhas, deverá ser feito um delete de todas as linhas e, seguidamente, inserir o texto no início do array, atualizando a posição atual para a última linha inserida.

3.4 Delete

A função delete (invocada pelo comando "d") tem como objetivo apagar um intervalo de linhas inserido pelo utilizador e define a linha atual como: a linha que se segue à(s) apagada(s) caso o delete não inclua a última linha ou a linha antes da(s) apagada(s) caso isso se verifique.

Esta função recebe como argumentos:

1. O endereço do array onde é guardado o conteúdo do ficheiro;
2. O endereço da linha atual no array;
3. O endereço da primeira posição vazia no array a contar do fim;
4. O endereço do valor X;
5. O endereço do valor Y;
6. O valor da variável global TRUECHAR;

O Delete pode ser utilizado das seguintes maneiras:

- 'd' - Será apagada a linha atual (endereço indicado por pos). É atualizada a posição atual conforme explicado anteriormente.
- 'xd' - Onde x corresponde a um número inserido pelo utilizador. Será apagada a linha x. É atualizada a posição atual e feitos os devidos movimentos de linhas.
- 'x,yd' - Onde x e y correspondem ao um intervalo de linhas pretendido pelo utilizador.Tal como mencionado, é necessário verificar se este intervalo inclui ou não a última linha. É atualizada a posição atual e feitos os devidos movimentos de linhas.
- '$' - Apaga a última linha apenas, deverá ser feito um moveUp e será atualizada a posição atual e feitos os devidos movimentos de linhas.
- 'x,$d' - Delete de um intervalo que inlcui a última linha.
- '%d' - Delete de todas as linhas. A posição atual passará a ser o endereço que precede a primeira posição do array e end o último endereço do array e feitos os devidos movimentos de linhas.

3.5 Print

A função print (invocada pelo comando "p") tem como objetivo, como o nome indica, fazer print de um intervalo de linhas inserido pelo utilizador e define a linha atual como a última linha sobre a qual foi feito o print.

Esta função recebe como argumentos:

1. O endereço do array onde é guardado o conteúdo do ficheiro;
2. O endereço da linha atual no array;
3. O endereço da primeira posição vazia no array a contar do fim;
4. O endereço do valor X;
5. O endereço do valor Y;
6. O valor da variável global TRUECHAR;

O Print pode ser utilizado das seguintes maneiras:

- 'p' - É feito print da linha atual pelo que não é necessário atualizar POS.
- 'xp' - Onde x corresponde a um número inserido pelo utilizador. É feito print da linha x e será utilizada a função changePos para alterar a posição atual.
- 'x,yp' - Onde x e y correspondem ao um intervalo de linhas pretendido pelo utilizador.Tal como mencionado, é necessário verificar se este intervalo inclui ou não a última linha. É atualizada a posição atual.
- '$' - Print da última linha, será feito um moveUp de todas as linhas que se encontram na zona de baixo do array e é atualizada POS.
- 'x,$p' - Print de um intervalo que inclui a última linha, caso semelhante ao anterior.
- '%p' - Print de todas as linhas corresponde a fazer print de um intervalo que inclui a última linha pelo que segue o mesmo princípio do caso anterior.

3.6 Write

A função write (invocada pelo comando "w") tem como objetivo, através do diretório indicado pelo user (guardado na variável FN), copiar a informação do array com o conteúdo desejado para o ficheiro indicado (se este não existir cria um novo).

Esta função recebe como argumentos:

1. O endereço da string com o diretório do ficheiro;
2. O endereço do array onde é guardado o conteúdo do ficheiro;
3. O endereço da linha atual no array;
4. O endereço da primeira posição vazia no array a contar do fim;

O Write pode ser utilizado das seguintes maneiras:

- 'w' - copia a informação do array para o ficheiro localizado no diretório em fname.

3.7 Edit

A função edit (invocada pelo comando "e") tem como objetivo, através do diretório indicado pelo user (guardado na variável FN), copiar a informação do ficheiro desejado para o array com o auxilio da função Distri.

Esta função recebe como argumentos:

1. O endereço da string com o diretório do ficheiro
2. O endereço do array onde é guardado o conteúdo do ficheiro;
3. O endereço da linha atual no array;
4. O endereço da primeira posição vazia no array a contar do fim;

O Edit pode ser utilizado das seguintes maneiras:

- 'e fname' - copia a informação do ficheiro localizado no diretório fname para o array pré-definido.

3.8 File

Quando inserido o caracter 'f' a função auxiliar FindDaWae irá colocar na variável FN (filename) a string com o diretório do ficheiro através da função copy.

- 'f fname' - define o valor da variável FN como o diretório do ficheiro.

4 Funções Auxiliares

4.1 mallocArray

A função mallocArray é uma função auxiliar bastante importante pois irá alocar espaço em cada índice do array para uma string de tamanho 100 bytes, colocando em cada índice o endereço do espaço alocado para cada string.

Esta função recebe como argumentos:

1. O endereço do array onde é guardado o conteúdo do ficheiro.

Valor de retorno:

- Nenhum.

4.2 isNumber

A função isNumber verifica se o argumento é um número.

Esta função recebe como argumentos:

1. ASCII do possível número.

Valor de retorno:

- 1/0 caso seja ou não número.

4.3 exit

A função exit faz uma syscall para sair do programa.

- Esta função não recebe argumentos nem tem valor de retorno.

4.4 exist_numbers

A função exist_numbers verifica se existem números na string recebida como argumento.

Esta função recebe como argumentos:

1. Endereco da string em questão.

Valor de retorno:

- 1/0 caso haja ou não números.

4.5 copy

A função copy copia uma string para outra.

Esta função recebe como argumentos:

1. endereço da string destino; endereço da string origem.

Valor de retorno:

- Nenhum.

4.6 GetOrigin

A função GetOrigin serve para obter localização do ficheiro.

Esta função recebe como argumentos:

1. O endereço da string onde estará a localização a obter.

Valor de retorno:

- Nenhum.

4.7 LenString

A função LenString calcula o comprimento de uma string, sem contar com o caracter nulo.

Esta função recebe como argumentos:

1. O endereço da string a analisar.

Valor de retorno:

- Comprimento da string (sem contabilizar o nulo).

4.8 CompareStrings

A função CompareStrings

Esta função recebe como argumentos:

1. Endereço da string número 1.
2. Endereço da string número 2.

Valor de retorno:

- 1/0 se as strings são iguais ou não.

4.9 isLetter

A função isLetter verifica se o byte recebido corresponde a uma letra.

Esta função recebe como argumentos:

1. ASCII do char a analisar.

Valor de retorno:

- 1/0 caso seja ou não letra.

4.10 NumberToInt

A função NumberToInt tem por objetivo passar um algarismo em char para int.

Esta função recebe como argumentos:

1. O endereço do byte com o char do algarismo.

Valor de retorno:

- Valor em inteiro do char em questão.

4.11 CheckChar

A função CheckChar verifica se determinado caractere está presente numa string.

Esta função recebe como argumentos:

1. O endereço da string onde se deverá procurar o caracter;
2. ASCII do caracter procurado.

Valor de retorno:

- 1/0 caso exista ou não na string.

4.12 stop

A função stop atualiza as variáveis globais x e y com os devidos valores.

Esta função recebe como argumentos:

1. Endereço da string que será analisada.

Valor de retorno:

- Nenhum.

4.13 FindFunction

A função FindFunction deteta se se trata da função e, p, w, etc... Para isso analisa uma string em busca do caractere correspondente à função desejada.

Esta função recebe como argumentos:

1. Endereço da string (inserida pelo user);
2. ASCII com o char correspondente ao possível comando.

Valor de retorno:

- 1/0 caso o caractere desejado esteja ou não presente na string.

4.14 FindIndex

A função FindIndex encontra a posição no array de determinada linha.

Esta função recebe como argumentos:

1. O endereço da linha atual no array;
2. O endereço da primeira posição (endereço) vazia no array a contar do fim;
3. Valor de X (número da linha cujo endereço se vai calcular).

Valor de retorno:

- Endereço da linha número x.

4.15 LastIndex

A função LastIndex encontra a posição no array da última linha escrita.

Esta função recebe como argumentos:

1. O endereço da linha atual no array;
2. O endereço da primeira posição (endereço) vazia no array a contar do fim.

Valor de retorno:

- Endereço da última linha escrita.

4.16 moveUp

A função moveUp movimenta um intervalo de linhas no array para cima (e atualiza a posição atual).

Esta função recebe como argumentos:

1. Endereço da linha ate a qual se fará o move (inclusivé);
2. Endereço do array (da primeira posição do array);
3. Endereço da linha atual no array;
4. Endereço da primeira posição (endereço) vazia no array a contar do fim.

Valor de retorno:

- Nenhum.

4.17 moveDown

A função moveDown movimenta um intervalo de linhas no array para baixo (e atualiza a posição atual).

Esta função recebe como argumentos:

1. Endereço da linha ate a qual se fará o move (inclusivé);
2. Endereço do array (da primeira posição do array);
3. Endereço da linha atual no array;
4. Endereço da primeira posição (endereço) vazia no array a contar do fim.

Valor de retorno:

- Nenhum.

4.18 changePos

A função changePos modifica a posição atual do array.

Esta função recebe como argumentos:

1. Endereço do array (da primeira posição do array);
2. Endereço da linha atual no array;
3. Endereço da primeira posição (endereço) vazia no array a contar do fim;
4. Endereço da linha número x.

Valor de retorno:

- Endereço da string correspondente à nova posição atual. Este valor de retorno serve apenas para que na função auxiliar FindDaWae se possa chamar a função puts para fazer print desta mesma string.

4.19 insertLines

A função insertLines insere o input do utilizador no array e atualiza a posição atual para corresponder à última linha inserida.

Esta função recebe como argumentos:

1. Endereço do array (da primeira posição do array);
2. Endereço da linha atual no array.

Valor de retorno:

- Nenhum.

4.20 Distri

A função Distri, auxilia a função edit dividindo os conteúdos do ficheiro analisado pelo array com base nas linhas.

Esta função recebe como argumentos:

1. O endereço da string com os conteúdos do ficheiro;
2. O endereço da linha do array onde guardar os conteúdos.

Valor de retorno:

- Nenhum.

4.21 FindDaWae

A função FindDaWae é uma função auxiliar responsável por analisar o input do user, identificar a função pretendida, colocar os argumentos corretos nos registos a e invocar a função em questão.

Caso a função pretendida seja a função file, todo o processo mencionado anteriormente relativamente a esta ocorre no FindDaWae. É também nesta função que termina o programa através dos comandos "q"ou "Q", invocando a função auxiliar exit (no caso do "q", isto apenas ocorre se o ficheiro tiver guardado).

Esta função recebe como argumentos:

1. Endereço de string com o input do utilizador;
2. Endereço do array;
3. Endereço da linha atual no array;
4. Endereço de end;
5. Endereço do inteiro que indica se ficheiro está salvo ou não;
6. Endereço da string com o nome do ficheiro de input.

Valor de retorno:

- Nenhum.

4.22 puts

A função puts mostra uma string da consola.

Esta função recebe como argumentos:

1. O endereço da string sobre a qual se pretende fazer print.

Valor de retorno:

- Nenhum.

4.23 gets

A função gets obtem uma string da consola (input do utilizador).

Esta função recebe como argumentos:

1. O endereço da string onde se irá guardar o input recebido;
2. O tamanho máximo dessa string.

Valor de retorno:

- Nenhum.

5 Dificuldades

Durante a elaboração deste trabalho surgiram várias dificuldades, a sua maioria acabou por ser resolvida, porém persistem erros para os quais não encontrámos solução no programa final. Uma vez que não conseguimos colocar o programa em C a funcionar acabou por ser mais difícil fazê-lo em Assembly.

A função FindIndex revelou-se complicada uma vez que nem sempre retornava o valor devido. Inicialmente esta função calculava o "gap"entre as variáveis pos e end e utilizava esta variável para calcular o endereço de linhas que se localizassem na zona de baixo do array, porém existiam casos em que o valor de retorno correspondia a um endereço para além do último endereço do array, estando portanto errado. Para corrigir isto utilizámos dois ciclos que percorrem o array desde o seu início até ao endereço da posição atual e desde o endereço que segue o endereço indicado pela variável end até ao final do array.

A função insert não funciona devidamente quando o array se encontra vazio. Portanto, na função FindDaWae que iria chamar a função i é testada uma condição que verifica se este é o caso (o array encontra-se vazio quando o endereço indicado por pos correspondender ao endereço que precede o endereço da primeira posição do array). Quando isto acontece em vez de ser chamada a função insert é chamada a função append, que nestas condições irá originar um output semelhante ao que seria suposto originar a função i.

A função delete teve de sofrer alterações para que seja possível executar comandos como "x,$c". Uma vez que a função change faz uso da função delete quando se pretende alterar um intervalo de linhas, e já que esta última não não estava a funcionar corretamente para comandos do tipo "x,$d", não era possível utilizar o comando mencionado para alterar um intervalo de linhas que incluísse a última. Porém, após algumas modificações no código da função delete este problema foi resolvido.

Um problema ao qual não arranjámos solução prende-se com a inserção de palavras que ocupem mais do que 3 bytes. Isto pois quando se utilizam as funções moveUp e moveDown, ao copiar as trings para a sua nova posição, provavelmente devido a como está organizada a memória nesta arquitetura, são copiados bytes a mais (por exemplo, ao inserir no input "adeus", se para a posição desta string copiarmos outra de menor tamanho o caractere de mudança de linha e o caractere "s"continuam válidos). Tentámos modificar a função copy para colocar um caractere nulo no final de cada string copiada porém isto não resultou, já que por razões desconhecidas este era colocado depois do caracter de mudança de linha.

6 Conclusão

Com este trabalho pudémos aprofundar o conhecimento que adquirimos ao longo do semestre relativo à organização da memória e, ainda que não tenha ficado resolvido, fomos capazes de identificar um problema a ela relativo no programa final. Foi também possível adquirir uma maior agilidade no que toca às diferentes operações suportadas pela arquitetura bem como em que situações é que cada uma deverá ser utilizada e quais os benefícios e possíveis problemas com que teremos de lidar caso as usemos. Consideramos que com uma melhor gestão de tempo teria sido possível elaborar um programa final mais eficiente e totalmente funcional.