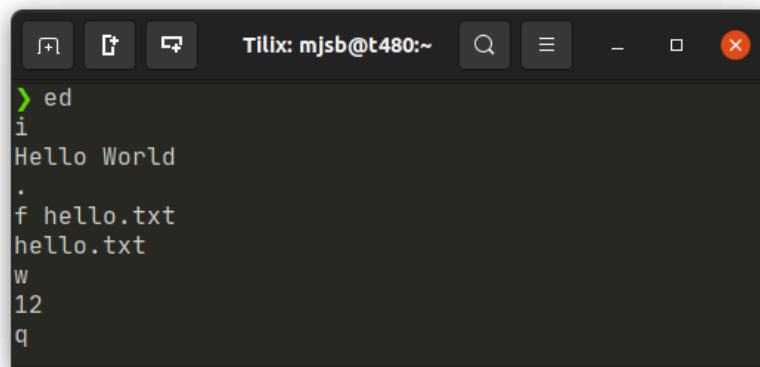


Arquitectura de Computadores I

Enunciado do Trabalho

Licenciatura em Engenharia Informática



```
Tilix: mjsb@t480:~  
> ed  
i  
Hello World  
.  
f hello.txt  
hello.txt  
w  
12  
q
```

1 Objectivo do trabalho

Pretende-se com este trabalho desenvolver um editor de linha que implemente os principais comandos de abertura, edição e gravação de ficheiros.

O trabalho está dividido em duas fases. Na primeira fase, o programa é desenvolvido em linguagem C com o objectivo de o estruturar em funções e escolher as estruturas de dados apropriadas. Na segunda fase é realizada a implementação em Assembly RISC-V.

O trabalho é realizado em grupos de 2 alunos.

2 Editores de linha

Os sistemas computacionais modernos têm tipicamente um interface de input/output composto por um teclado e um ecrã. O tipo de interface físico com o utilizador determina também o modo de funcionamento das aplicações. Um exemplo são os editores de texto usados no desenvolvimento de software, onde o texto é escrito no teclado e instantaneamente visualizado no ecrã. A característica dos ecrãs serem bidimensionais permite a navegação horizontal e vertical, com um cursor, e modificar o texto em qualquer posição do documento.

Antes dos computadores usarem ecrãs, eram usadas impressoras para visualizar o output do computador. As impressoras imprimiam apenas texto num rolo contínuo de papel. Este interface de output era muito diferente dos ecrãs, que só surgiram posteriormente, e obrigou ao desenvolvimento de maneiras diferentes de editar ficheiros de texto. Em particular, não é possível

mover o cursor para trás para modificar o texto. Uma vez impresso, o texto já não pode ser alterado no papel. Foi neste contexto que se desenvolveram os editores de linha.

Um editor de linha é um programa que permite editar ficheiros de texto e é útil quando o interface de output é uma impressora. Um editor de linha funciona em dois modos: modo de comando e modo de inserção. No modo de comando, o input introduzido no teclado é interpretado como sendo comandos para o editor, como por exemplo “apagar linha”, “inserir texto”, “gravar ficheiro”, “abrir ficheiro”, *etc.* No modo de inserção, o input introduzido no teclado é considerado como texto pertencente ao documento e é guardado em memória.

Na altura em que o sistema operativo UNIX foi desenvolvido, ainda se usavam impressoras como standard output. Por esse motivo, todos os sistemas UNIX incluem, ainda hoje, o editor de linha **ed**. É o caso dos sistemas Gnu/Linux, MacOS, FreeBSD, OpenBSD, Minix, Solaris, *etc.* O **ed** faz parte da especificação POSIX que um sistema operativo tem de obedecer para poder ser considerado um sistema UNIX.

Pode testar o editor de linha dando o comando **ed** no terminal. Não é apresentada nenhuma mensagem (para poupar papel...) mas o editor está a correr em modo de comando à espera de uma ordem. Durante toda a edição, o texto é mantido em memória até ser dado o comando para escrever num ficheiro.

Alguns dos comandos disponíveis são os seguintes:

Comando	Nome	Descrição
i	<i>insert</i>	insere texto antes da linha actual.
a	<i>append</i>	acrescenta texto depois da linha actual.
c	<i>change</i>	substitui a linha actual por novo texto.
d	<i>delete</i>	apaga a linha actual.
p	<i>print</i>	imprime a linha actual.
e fname	<i>edit</i>	abre ficheiro existente para edição.
f fname	<i>file</i>	define o nome do ficheiro.
w	<i>write</i>	escreve ficheiro no sistema de ficheiros.
q	<i>quit</i>	sai do editor se o documento estiver guardado.
Q	<i>Quit</i>	sai do editor sem guardar.

Os comandos **i**, **a**, **c** passam para modo de inserção para se introduzir novo texto. Para terminar a inserção e regressar ao modo de comando, escreve-se uma linha composta só por um ponto final.

Alguns comandos aceitam argumentos para representar linhas ou sequências de linhas a que o comando se aplica. O cifrão representa a última linha e a percentagem representa todas as linhas. A primeira linha tem o número 1.

Comando	Descrição
5i	insere texto antes da linha 5.
\$a	acrescenta texto no fim (o cifrão representa o fim).
3c	substitui a linha 3 por uma nova.
%p	imprime todas as linhas.
5,\$p	imprime da linha 5 até à última.
1,10d	apaga da primeira linha até à décima.
27	a linha 27 passa a ser a linha actual.

Mostra-se em seguida um exemplo de uma sessão **ed**. É introduzida uma linha de texto, inserida uma nova linha antes da anterior, e finalmente substituída a segunda linha (introduzida inicialmente) por uma nova. No fim é mostrado o conteúdo do documento.

```

i
Linha de texto
.
1i
Esta vai ser inserida no inicio
.
2c
Esta substitui a linha de texto inicial
.
%p
Esta vai ser inserida no inicio
Esta substitui a linha de texto inicial

```

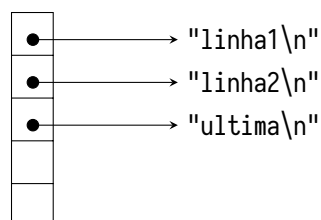
3 Trabalho a desenvolver

Implementação em linguagem C Nesta fase deverá implementar o editor de linha escrito em linguagem C. O programa deverá estar devidamente estruturado em funções e comentado. Cada comando do editor deve estar implementado numa função separada. Esta implementação deverá ser acompanhada de um pequeno relatório \LaTeX com a descrição das funções.

Implementação em Assembly RISC-V Nesta fase deverá implementar o mesmo programa, mas agora em Assembly para correr no simulador RARS num sistema do tipo UNIX. O programa deverá seguir a mesma estrutura (funções e estruturas de dados) usadas na implementação em C e seguir as convenções da ABI RISC-V estudadas neste curso. O programa também deverá ser acompanhado de um relatório com a descrição da organização do programa, indicado as diferenças relativamente ao programa em C (caso haja). As funções devem estar documentadas de forma semelhante às *man pages* acessíveis no terminal com o comando `man`.

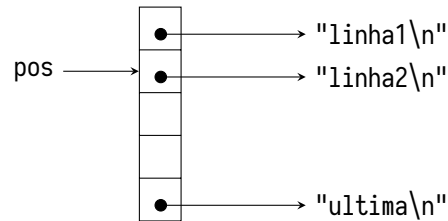
É livre de usar a estrutura de dados que achar mais conveniente. Deixam-se de seguida algumas sugestões.

Uma estrutura possível para guardar o texto consiste em usar um array de ponteiros para as linhas (strings). Deste modo, sempre que for necessário inserir ou remover linhas, basta deslocar os ponteiros no array, sem necessitar de mover as strings em memória.



De cada vez que for necessário inserir uma linha nova no meio das outras é necessário deslocar os ponteiros para a frente de modo a abrir espaço para a linha nova. Em documentos pequenos isso não é problemático, mas em ficheiros muito grandes, cada inserção ou remoção duma linha perto do início obriga a deslocar a maioria dos ponteiros para a frente, o que pode ser ineficiente.

Para evitar estes deslocamentos, pode-se usar o array com um intervalo vazio no meio (*gap*) onde são inseridas todas as linhas novas. Neste caso a linha actual representada pela seta **pos** corresponde à última linha do bloco de linhas inicial antes do *gap*.



Estas duas estruturas são apenas sugestões. Pode usar outras diferentes se preferir.

4 Submissão do trabalho

O trabalho é composto por um relatório em formato pdf e pelo código assembly. Os vários ficheiros devem ser comprimidos num único ficheiro em que o nome contém os números dos alunos, *e.g.* 12345-67890.tar.xz, e submetidos no moodle até à data limite de submissão. Para comprimir os ficheiros pode usar o comando **tar** com um dos algoritmos de compressão disponíveis: xz, gz ou bz2.

Bom trabalho!
Miguel Barão