Arquitetura e Sistemas de Computadores I
2015/2016

Departamento de Informática, Universidade de Évora

Trabalho prático
15 de maio de 2015 (v1.0)

– Cifra de Vigenère –

1 Introdução

A cifra de Vigenère é um método de encriptar texto alfabético usando uma série de diferentes codificações César com base nas letras de uma palavra-chave. É uma forma simples de substituição polialfabética.

A cifra de Vigenère foi reinventada muitas vezes; o método foi originalmente descrito por Giovan Battista Bellaso em 1553 mas foi mais tarde atribuída erroneamente a Blaise de Vigenère, no século 19.

Embora seja fácil de entender e implementar, durante três séculos a cifra resistiu a todas as tentativas de quebra ganhando a descrição de "a cifra indecifrável"[1].

2 Descrição

Numa cifra de César cada letra do alfabeto é deslocada um certo número de posições; por exemplo, numa cifra de César de deslocação 3, um "A"seria um "D", um "B"torna-se um "E"e um "Y"é representado por um "B". A cifra de Vigenère consiste numa sequência de cifras de César com diferentes valores de deslocamento. A deslocação em cada posição depende de uma palavra-chave de repetição.

Por exemplo, suponha que o texto a ser criptografado é "atacar de madrugada"e a pessoa que envia a mensagem escolhe a palavra-chave "laranja". Assim, a primeira letra do texto "A", é emparelhado com a letra "L", sendo transformada num "L"; a segunda letra "T"é emparelhada com com "A", mantendo o seu valor. O resto do texto original é cifrado de forma semelhante:

- texto: "`atacardemadrugada`"
- palavra-chave: "`laranja`"
- texto cifrado: "`ltrcnedpmrdedgldr`"

A desencriptação é realizada da mesma forma "subtraindo"a palavra chave ao texto cifrado.

A cifra de Vigenère pode ser calculada de forma algébrica, transformando as letras A-Z nos números 0-25 e fazendo a adição módulo 26. Pode incorporar o espaço e sinais de pontuação atualizando estes limites.

A execução do programa deverá seguir os seguintes passos:

1. Pedir ao utilizador para introduzir o nome de um ficheiro de texto;
2. Abrir o ficheiro, caso exista, ou apresentar mensagem de erro;
3. Encriptar o texto para um novo ficheiro com extensão `.vig`; se o ficheiro introduzido tiver extensão `.vig`, então deve desencriptar para um ficheiro sem a extensão.

[1] texto adaptado da Wikipedia: `https://en.wikipedia.org/wiki/Vigenere_cipher`

3 Implementação em MIPS

O programa deverá estar devidamente estruturado em funções que **executem tarefas simples** e seguir as convenções MIPS para o uso dos registos, chamada de funções, passagem de argumentos, pilha, variáveis locais, etc.

Deve implementar obrigatoriamente as seguintes funções:

- `fopen( name, mode )`: esta função abre o ficheiro `name` em modo `mode` e devolve o descritor do ficheiro. O modo de abertura é "`r`"para leitura e "`w`"para escrita;
- `fclose( descriptor )`: esta função fecha o ficheiro com descritor `descriptor`;
- `encripta( car1, car2 )`: esta função devolve o valor de deslocação da encriptação do caracácter `car1` pelo carácter `car2`;
- `desencripta( car1, car2 )`: esta função devolve o valor de deslocação da desencriptação do caracácter `car1` pelo carácter `car2`;

Vai necessitar das seguintes funções sobre strings[2] (terminadas por `null`):

- `strcat( s1, s2 )`: concatena uma cópia da string `s2` no final da string `s1` adicionando um `'\0'`. A string `s1` deve ter espaço suficiente para conter o resultado. A função devolve a string `s1`.
- `strstr( big, little )`: esta função localiza a primeira ocorrência da string `little` na string `big` devolvendo o seu endereço.

4 Manipulação de ficheiros de texto

Em MIPS a leitura de ficheiros é efectuada da mesma forma que em sistemas UNIX/LINUX com as funções `open`, `read`, `write` e `close`.

Para abrir um ficheiro para leitura pode usar-se o código seguinte:

```
la $a0, FILE     # string com o nome do ficheiro
li $a1, 0
li $a2, 0        # read only
li $v0, 13       # open
syscall          # v0 = descritor do ficheiro
```

No final, o registo `$v0` contém o descritor do ficheiro que será usado para as operações de leitura seguintes. A leitura do ficheiro é efectuada com um `read` da seguinte maneira:

```
# a0 = descritor do ficheiro
# a1 = buffer onde vai ser escrita a informacao lida
# a2 = numero de bytes a ler
li $v0, 14       # read
syscall
```

Para abrir um ficheiro para escrita pode usar-se o código seguinte:

```
la $a0, FILE     # string com o nome do ficheiro
li $a1, 577      # flags O_CREAT|O_TRUNC|O_WRONLY
li $a2, 0x1ff    # permissões de escrita 777
li $v0, 13       # open
syscall          # v0 = descritor do ficheiro (-1 em caso de falha)
```

A escrita neste ficheiro é efectuada com um `write` da seguinte maneira:

[2] O comando `man funcao` permite ver a descrição pormenorizada destas funções.

```
# a0 = descritor do ficheiro
# a1 = endereço do buffer
# a2 = numero de bytes a escrever
li $v0, 15       # write
syscall
```

No final os ficheiros abertos devem ser fechados com a função `close`:

```
# a0 = descritor do ficheiro
li $v0, 16       # close
syscall
```

5 Desenvolvimento do trabalho

O trabalho inclui a implementação do código e a escrita de um relatório em PDF.

5.1 Implementação do código

Utilize os seguintes passos para a implementação do trabalho:

1. Comece por estruturar o programa em funções, com especificação clara dos argumentos e valor de retorno de cada função;
2. Desenhe um fluxograma que descreva os principais passos executados dentro de cada função;
3. Implemente e teste uma função de cada vez, começando pelas mais simples e que não chamam outras funções.

É essencial **comentar o código**. Utilize o seguinte formato:

```
#######################################################
# xpto - Esta funcao calcula a area de um retangulo
#
# Argumentos:
#  a0 - largura
#  a1 - comprimento
#
# Valor de retorno:
#  v0 - area calculada
#######################################################
xpto:
    mult $a0, $a1
    ...
    jr $ra
    nop
```

O programa deverá ser desenvolvido em **Linux** (ou MacOS)[3] por grupos de **2 alunos** e correr no simulador MARS 4.5.

5.2 Relatório

O relatório deverá incluir uma listagem de todas as funções implementadas, especificando os argumentos recebidos, o valor retornado e uma descrição detalhada da funcionalidade implementada. Para ter uma ideia do que se pretende, veja as *man pages* em Linux:

[3] A representação de strings no Windows é distinta o que não permite a sua correta execução em ambientes UNIX.

```
$ man puts
```

O texto deve ser escrito de forma impessoal, isto é, usando voz passiva e focando-se nos aspectos técnicos e não nos autores. Exemplos:

- Certo: Foi encontrado um erro...
  Errado: Encontrámos um erro...
- Certo: Nos testes efectuados verificou-se que o programa não funciona correctamente se...
  Errado: Reparámos que o programa não funciona correctamente se...
- Certo: A função xpto foi usada para...
  Errado: Usámos a função xpto para...

A organização do relatório e a qualidade do texto também são avaliadas. Sugere-se a escrita sua escrita em LaTeX e utilizar os editores online Sharelatex[4] ou Overleaf[5].

5.3 Prazos e Entrega

O trabalho deve ser desenvolvido em duas fases e entregue via moodle em duas datas distintas (especificadas no moodle):

Fase 1 – Tarefas (mínimas):

- Estruturar o programa em funções, com especificação clara dos argumentos e valor de retorno de cada função;
- Desenhar um fluxograma que descreva os principais passos executados dentro de cada função.

Fase 2 – Tarefas:

- Implementar a solução proposta, com alterações se necessário;
- Completar o relatório.

No moodle é submetido apenas um ficheiro com nome definido pelos números dos alunos (por exemplo `12345-67890.tar.gz`).

Na fase 2, o ficheiro deve incluir o código implementado e o relatório; o código do programa deve ser um ficheiro com a extensão `.asm`. Os ficheiros podem ser comprimidos com o comando

```
$ tar cvfz 12345-67890.tar.gz programa.asm relatorio.pdf
```

Bom trabalho!

[4] `https://www.sharelatex.com/`
[5] `https://www.overleaf.com/`