Arquitetura de Computadores 2022/23

Ficha 3

Tópicos: *Programas em C. O utilitário make.*

O utilitário make

O utilitário `make`, que pode ser consultado nas páginas do manual com `$ man make`, é utilizado para descrever quaisquer tarefas onde alguns dos ficheiros necessitam de ser atualizados automaticamente a partir de outros ficheiros dos quais dependam, sempre que estes últimos ficheiros sejam alterados. Em particular, o `make` permite determinar, automaticamente, quais a partes de um programa de grande dimensão que necessitam de ser recompiladas, e submete os comandos para a sua recompilação. Tipicamente, a regras que definem as dependências entre ficheiros e as ações a realizar estão definidas num ficheiro de nome `Makefile` ou `makefile`, o qual é consultado por defeito pelo comando make. Cada regra é definida por duas linhas; a primeira indica o nome do alvo/*target* a produzir separado de dois pontos dos ficheiros dos quais depende; a segunda linha é sempre iniciada com o carácter `TAB`, seguida da ação/comando a realizar, de modo a produzir o alvo/*target* com base nos ficheiros dependentes.

Por exemplo, dado o seguinte Makefile:

```
ALL: progr1 progr2

progr1: prog1.c
	cc -o prog1 prog1.c

progr2: progr2.c progr3.o
	cc -o progr2 progr2.c progr3.o

progr3.o: progr3.c
	cc -c progr3.c

clean:
	rm *.o progr1 progr2
```

Se invocarmos o comando make

```
$ make
```

O `make` vai considerar a primeira regra encontrada no ficheiro que é a regra `ALL`. Esta regra indica que existem duas dependências, `progr1` e `progr2`. De seguida, o comando `make` procura e encontra a regra para gerar `progr1`, a qual define que é necessário compilar o ficheiro `progr1.c` para gerar o executável `progr1`. De seguida, procura e encontra a regra para gerar o ficheiro `progr2`, só que, agora, esta regra define que há uma dependência adicional, nomeadamente do ficheiro objecto progr3.o. Novamente o make procura e encontra a regra para gerar o ficheiro progr3.o, a qual define a utilização do comando cc apenas com a flag -c, para gerar o ficheiro objecto `progr3.o` a partir do ficheiro `progr3.c`. Caso o utilizador queira usar um ficheiro de nome diferente de `makefile` ou `Makefile`, pode utilizar o `make` com a opção `-f`:

```
$ make -f myMakefile
```

Cálculo do histograma de um conjunto de valores

Neste exercício deve conceber, implementar e testar uma função escrita na linguagem C que cria um histograma dos valores que ocorrem num vetor de inteiros. Essa função deve poder ser invocada de um programa em C onde é declarada da seguinte forma:

```
extern histograma( int valores[ ], int size, int histogr [ ] );
```

Admite-se que o vetor valores tem size posições e que contém apenas valores inteiros entre 0 e `MAX-1` com `MAX` positivo; o vetor `histogr[ ]` tem `MAX` posições e `histogr[ i ]` contém o número de ocorrências do valor `i` no vetor `valores`.

O código abaixo exemplifica o uso da função histograma. Este programa está contido no ficheiro main_hist.c que é disponibilizado no CLIP. Note que a função histograma não deve escrever nada no ecrã.

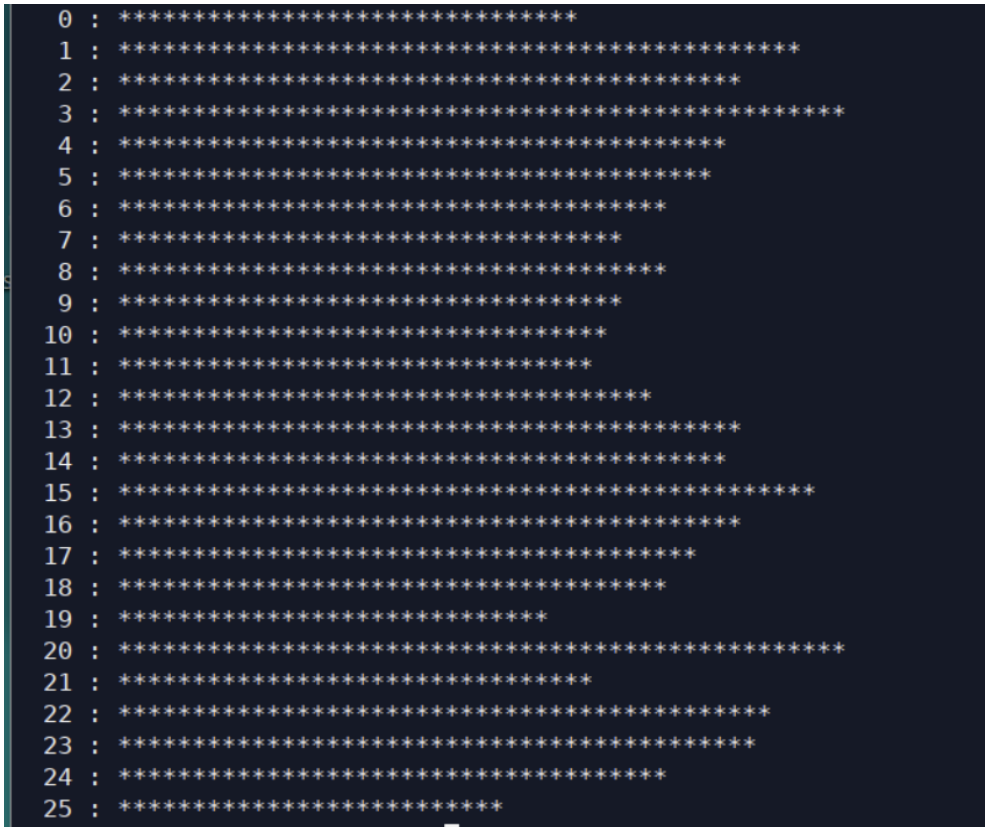
```
#include <stdio.h>
#include <stdlib.h>

#define SIZE 1000
#define MAX 26

void printHistograma( int histo[], int size ) {
   int i, j;
   for( i = 0; i < size; i++) {
        printf(" %3d : ", i);
        for( j = 0; j < histo[i]; j++ )
            printf("*");
        printf("\n");
    }
}

extern void histograma( int valores[ ], int size, int histo[] );

int main( ) {
   int vals[SIZE], hist[MAX], i;
   for (i=0; i < SIZE; i++)
             vals[ i ] = random() % MAX;
   for (i=0; i < MAX; i++)
             hist[ i ] = 0;

   histograma( vals, SIZE, hist );
   printHistograma( hist, MAX );
   return 0;
}
```

Supondo que o código fonte da função histograma está no ficheiro `hist.c`, o ficheiro executável `histograma` pode ser obtido com os seguinte comandos no shell :

```
$ cc -c  hist.c
$ cc -o histograma  main_hist.c hist.o
```

Em alternativa, altere o ficheiro `Makefile` e use o utilitário `make`, referidos anteriormente, para fazer a compilação dos dois ficheiros.

O resultado da execução do programa `histograma` é o seguinte (o histograma dependerá dos valores aleatórios em `vals`):



Manipulação de strings

Faça um programa de nome `echostring` que fica em ciclo a ler uma *`string`* do teclado e a escrevê-la no ecrã prefixada com "`->`". O programa só terminará quando o utilizador escrever a *`string`* "`fim!`". Veja o seguinte exemplo (a **negrito** o que o utilizador escreveu):

```
$ ./echostring
ola
->ola
boa tarde
->boa tarde
fim
->fim
termina!!!
->termina!!!
fim!
->fim!
$
```

Para realizar este programa necessita de utilizar as funções: `fgets`, `printf`, e `strcmp`. Para saber como se usam estas funções leia as páginas de manual correspondentes executando no terminal o comando "`man 3 função`", onde função deverá ser o nome da função de que pretende obter informação.

Tipo dos argumentos passados a um programa

Copie o programa C de nome "`args.c`" a partir do sistema CLIP. Este programa recebe um conjunto de argumentos na linha de comando e, para cada um dos argumentos, tenta adivinhar

o seu tipo e depois invoca a função `printf`. No caso das `strings`, também usa a função `maiusculas`.
Complete o programa implementando a função `maiusculas` que deve devolver a `string` em parâmetro com todas as letras passadas a maiúsculas. Veja a função `toupper` da biblioteca de C (use por exemplo o comando: `man 3 toupper`).

Mais informação

Sobre a linguagem de programação C:

Livro *Dive into Systems*, capítulos 1 e 2, https://diveintosystems.org/book/C1-C_intro/index.html e https://diveintosystems.org/book/C2-C_depth/index.html

http://publications.gbdirect.co.uk/c_book/

http://www.cprogramming.com/tutorial/c-tutorial.html