Universidade Federal de Pelotas

Conceitos de Linguagens de Programação

Relatório do Projeto

Misturando Linguagens: C e Fortran

Alunos	Renata Junges
	Matheus Neiverth
	Yan Soares
Professor	Gerson Cavalheiro

Pelotas, 13 de Outubro de 2015

Conteúdo

1 Introdução

Esse relatório tem como objetivo demonstrar os resultados obtidos e as etapas concluídas na implementação de quatro códigos diferentes para a resolução de um problema: um puramente em C, outro puramente em Fortran, outro em C com chamada de função em Fortran e o último em Fortran, com chamada de função em C.

1.1 Problema Proposto

O problema proposto no projeto visa avaliar as capacidades dos alunos em diferentes maneiras, tanto como na programação, como também nas etapas de compilação necessárias para diferentes fins (nesse caso, misturar linguagens). A tarefa era dividida em 2 atividades:

Atividade 1. Implementar dois programas um em C, outro em Fortran que leia de um arquivo que contenha duas linhas. Em cada linha há uma cadeia de caracteres (string). A primeira string tem de 2 a 10 caracteres. A segunda, entre 100 e 1000 caracteres. Os programas construídos devem retornar o número de vezes que a primeira string é encontrada na segunda. Observe que podem ocorrer sobreposições, ou seja "aba"ocorre três vezes em "abababa".

Atividade 2. Utilizar as implementações acima para gerar duas novas versões para os programas, uma fazendo a leitura dos arquivos e a impressão do resultado em C, mas as comparações em Fortran, o outro, ao contrário. O arquivo de entrada é especificado na seção 6.

2 Código em C

O código puramente em C tem uma função main, que chama uma função separada "numeroVezesC"(definida no arquivo biblioteca.c) responsável pela comparação das strings e a contagem do número de vezes que a substring da primeira linha se repete na string da segunda. Além disso, essa função é definida no header biblioteca.h e há também um makefile para compilação.

2.1 main.c

```c
#include <stdio.h>
#include <string.h>
//o número de vezes que a primeira string é encontrada na segunda
int main(void) {
        FILE *arquivo; // estrutura para usar arquivo
        int cont,i, tam1, tam2;
        char linha1[11], linha2[1001], ch ; // cria uma 'string' para a linha 1 e a linha 2
        arquivo = fopen("entrada.txt", "r+");// abre o programa para read-leitura
        if (arquivo == NULL){ // se o arquivo é NULL significa que ele não existe
                printf("Erro ao abrir o arquivo\n");
                return 0;//fecha o programa caso o arquivo seja NULO
        }
        i=0;
        // Le a primeira string, char a char até a quebra de linha
        while( (ch=fgetc(arquivo))!= '\n' ){
                linha1[i]=ch;
                i++;
        }
        linha1[i]='\0';//encerra a primeira string
        i=0;
        // Le a segunda string, char a char
        while( (ch=fgetc(arquivo))!= '\n' ){
                linha2[i]=ch;
                i++;
        }
        linha2[i]='\0';//encerra a segunda string
        fclose(arquivo); //fecha o arquivo
        tam1=strlen(linha1);
        tam2=strlen(linha2);
        cont = numeroVezesC(linha1, linha2, tam1, tam2);
            //Chama a funcão que verifica o número de vezes
        printf ("%d", cont);
        return 0;
}
```

2.2 biblioteca.c

```c
//biblioteca.c
#include <stdio.h>
#include <stdlib.h>

int numeroVezesC(char *linha1,char *linha2, int tam1,
   int tam2){
        int cont=0,i=0,j=0, n=0;
        char aux[tam1]; // string auxiliar para fazer
           comparacões

        while (j != tam2){
                n++;
                while (i!=tam1){
                        aux[i]=linha2[j];
                        j++;
                        i++;
                }
                j=n; //para garantir que encontrará
                   substrings que serão sobrescritas
                i=0;
                if (strcmp(linha1, aux)==0){ //strings
                   iguais, achou 1 substring
                        cont++; //achou a subpalavra
                }
        }
        return cont;
}
```

2.3 biblioteca.h

```c
#include <string.h>
#include <stdio.h>

int numeroVezesC(char *linha1,char *linha2, int tam1,
   int tam2)
```

2.4 makefile

```
#C
index:
        gcc -c biblioteca.c
        gcc -c trabalho1.c
        gcc trabalho1.o biblioteca.o -o trabalhoC
        ./trabalhoC
```

3 Código em Fortran

3.1 main.f90

,

```
PROGRAM trabalho1 !Inicio do trabalho de clp em
   fortran

IMPLICIT NONE !Todas as variáveis terão que ter seus
    tipos definidos EXPLICITAMENTE
CHARACTER *10 first !Primeira linha-> equivalente a
   first[10]
CHARACTER *1000 second !Segunda linha-> equivalente
   a second[1000]
INTEGER cont,tam1,tam2
INTEGER, EXTERNAL :: numero !Funcão q retorna o
   numero de substrings encontradas

!Abertura de arquivo
OPEN(UNIT=50,FILE="entrada.txt") !Leitura de arquivo
   , demais atributos não são obrigados (status,
   action)

!leitura do arquivo
READ(50, *) first  !Le a primeira linha do arquivo
READ(50,*) second !Le a segunda liha do arquivo
CLOSE(50) !Fecha o arquivo
tam1=LEN_TRIM (first)
tam2=LEN_TRIM (second)
cont =numero(first,tam1, second, tam2)
print *, cont !Informa o numero de vezes que a
   string1 está na string2
```

```
20 END PROGRAM trabalho1
21
22 FUNCTION numero(first,tam1,second,tam2) !Parametros
      passados
23                 CHARACTER *10, INTENT(IN) :: first !
                      Não altera o valor de first
24                 CHARACTER *1000, INTENT (IN) ::
                      second !Não altera o valor de
                      second
25                 INTEGER :: cont, aux, inicio
26                 INTEGER, INTENT (IN):: tam1, tam2
27                         cont=0
28                         aux=1
29                         inicio=1
30                 !Para não comparar inutilmente os
                      ultimos chars da string2,
                      descontamos o tamanho da string1
31                 DO WHILE(aux <= (tam2 -tam1+1))
32                         !Compara por posicões como
                              se fossem posicões de um
                              vetor.
33                         IF(first(inicio: tam1) ==
                              second(aux:(tam1 + aux-1)
                              )) THEN
34                                 cont=1+cont
35                         END IF
36                         aux = aux + 1
37                 END DO
38                 numero= cont
39 END FUNCTION numero
```

4 C com Fortran

Nesse caso, o programa principal é escrito em C e é chamada a função "numero" (escrita em Fortran), que está presente no arquivo numeroVezes-Fortran.f90. Conta também com makefile.

4.1 main.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
//o número de vezes que a primeira string é encontrada na segunda
extern int numero(char *linha1, int *tam1, char *linha2, int *tam2);

int main(void) {
        FILE *arquivo; // estrutura para usar arquivo
        int cont,i, tam1, tam2;
        char linha1[11], linha2[1001], ch ; // cria uma 'string' para a linha 1 e a linha 2
        arquivo = fopen("entrada.txt", "r+");// abre o programa para read-leitura
        if (arquivo == NULL){ // se o arquivo é NULL significa que ele não existe
                printf("Erro ao abrir o arquivo\n");
                return 0;//fecha o programa caso o arquivo seja NULO
        }
        i=0;
        // Le a primeira string, char a char até a quebra de linha
        while( (ch=fgetc(arquivo))!= '\n' ){
                linha1[i]=ch;
                i++;
        }
        linha1[i]='\0';//encerra a primeira string
        i=0;
        // Le a segunda string, char a char
        while( (ch=fgetc(arquivo))!= '\n' ){
                linha2[i]=ch;
                i++;
        }
        linha2[i]='\0';//encerra a segunda string
        fclose(arquivo); //fecha o programa*/

        tam1=strlen(linha1);
        tam2=strlen(linha2);
```

```
34          cont = numero(linha1, &tam1, linha2, &tam2); //
               Chama a funcão que verifica o número de vezes
35          printf ("%d", cont);
36          return 0;
37
38  }
```

4.2 numeroVezesFortran.f90

,

```
1   FUNCTION numero(first,tam1,second,tam2) !Parametros
       passados
2           CHARACTER *10, INTENT(IN) :: first !Não
               altera o valor de first
3           CHARACTER *1000, INTENT (IN) :: second !Não
               altera o valor de second
4           INTEGER :: cont, aux, inicio
5           INTEGER, INTENT (IN):: tam1, tam2
6                   cont=0
7                   aux=1
8                   inicio=1
9           !Para não comparar inutilmente os ultimos
               chars da string2, descontamos o tamanho
               da string1
10          DO WHILE(aux <= (tam2 -tam1+1))
11                  !Compara por posicões como se fossem
                       posicões de um vetor.
12                  IF(first(inicio: tam1) == second(aux
                       :(tam1 + aux-1))) THEN
13                          cont=1+cont
14                  END IF
15                  aux = aux + 1
16          END DO
17          numero= cont
18  END FUNCTION numero
```

4.3 makefile

```
1   #C com Fortran
2   index:
```

```
3         gfortran -c numeroVezesFortran.f90 -fno-
              underscoring
4         gcc -c CcomFortran.c
5         gcc numeroVezesFortran.o CcomFortran.o -o
              codigoCFortran -lgfortran
6         ./codigoCFortran
```

5 Fortran com C

O programa principal é escrito em Fortran, com chamada a função "numero", definida no arquivo biblioteca.c e também conta com makefile.

5.1 main.f90

,

```
1  PROGRAM trabalho1 !Inicio do trabalho de clp em
       fortran
2
3  IMPLICIT NONE !Todas as variáveis terão que ter seus
       tipos definidos EXPLICITAMENTE
4  CHARACTER *10 first !Primeira linha-> equivalente a
       first[10]
5  CHARACTER *1000 second !Segunda linha-> equivalente
       a second[1000]
6  INTEGER cont,tam1,tam2
7  INTEGER, EXTERNAL :: numero !Funcão q retorna o
       numero de substrings encontradas
8
9  !Abertura de arquivo
10 OPEN(UNIT=50,FILE="entrada.txt") !Leitura de arquivo
       , demais atributos não são obrigados (status,
       action)
11
12 !leitura do arquivo
13 READ(50, *) first  !Le a primeira linha do arquivo
14 READ(50,*) second !Le a segunda liha do arquivo
15 CLOSE(50) !Fecha o arquivo
16 tam1=LEN_TRIM (first)
17 tam2=LEN_TRIM (second)
18 cont =numero(first,tam1, second, tam2)
```

```
19 print *, cont !Informa o numero de vezes que a
      string1 está na string2
20 END PROGRAM trabalho1
```

5.2 biblioteca.c

```
1  //biblioteca.c
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <string.h>
5
6  int numero(char *linha1,int *tam1,char *linha2, int *
      tam2){
7          int cont=0,i=0,j=0, n=0;
8          char aux[*tam1]; // string auxiliar para fazer
              comparacões
9          //Como o código vem do fortran precisamos
              colocar um \0 no fim para se adaptar ao có
              digo C
10         linha1[*tam1]='\0';
11         linha2[*tam2]='\0';
12         i=0;
13         while (j != *tam2){
14                 n++;
15                 while (i!=*tam1){
16                         aux[i]=linha2[j];
17                         j++;
18                         i++;
19                 }
20                 aux[i]='\0';
21                 j=n; //para garantir que encontrará
                      substrings que serão sobrescritas
22                 i=0;
23                 if (strcmp(linha1, aux)==0){ //strings
                      iguais, achou 1 substring
24                         cont++; //achou a subpalavra
25                 }
26         }
27         return cont;
28 }
```

5.3 makefile

```
#Fortran com C
index:
        gfortran -c trabalho1.f90 -fno-underscoring
        gcc -c biblioteca.c
        gfortran trabalho1.o biblioteca.o -o trabalho1
        ./trabalho1
```

6 Arquivo de entrada

O arquivo de entrada usado para testes (entrada.txt), conta com o seguinte conteúdo:

```
aba
abababaababababaababababaababababaababababaababababaababababaababababaababababaababababa
abababaababababaababababaababababaababababaababababaababababaababababaababababaababababa
abababaababababaababababaababababaababababa
```

7 Adaptações de código

7.1 Passagem de parâmetros

Houve a necessidade de adaptarmos as funções de cada código, pois Fortran necessita que os argumentos das funções sejam passados por referência (endereço), já em C a cópia é feita por conteúdo (valor).

7.2 Processo de Compilação

As instruções -fno-underscoring e -lgfortran são diretrizes de compilação, impedindo a inserção de caracteres nos nomes das funções.

7.3 Função Externa

Nos códigos que chamam a função em outra linguagem, há a necessidade de declarar a função como External (Fortran) ou Extern (C).

8 Referências Bibliográficas

[1] GCC. The GNU Compiler Collection. Disponível em:
https://gcc.gnu.org
[2] GFORTRAN. The GNU Fortran. Disponível em:
https://gcc.gnu.org/fortran
[3] Makefile. GNU Makefile. Disponível em:
http://www.gnu.org/software/make/manual/make.html
[4] Introdução ao Fortran 90/95. Departamento de Física - Universidade Federal de Pelotas. 2010. Disponível em: http://minerva.ufpel.edu.br/~rudi/grad/ModComp/Apostila/Apostila_links.pdf
[5] Apostila de Fortran. Curso de Física - Universidade Estadual do Ceará. Disponível em: http://www.dma.ufv.br/tutorial/fortran.pdf
[6] Guia Básico de programação em linguagem Fortran 77 e 90. PEREIRA CRISTO, Helder. Junho/2003. Disponível em: http://www.inf.ufes.br/~thomas/fortran/tutorials/helder/fortran.pdf