

**FUNDAÇÃO UNIVERSIDADE FEDERAL DO AMAZONAS  
FACULDADE DE TECNOLOGIA  
ENGENHARIA DA COMPUTAÇÃO**

**PROGRAMAÇÃO EM TEMPO REAL**

**AMBIENTE DE PROGRAMAÇÃO**

**Manaus – AM  
2017**

**FELIPE DE MENEZES SANTOS**

**AMBIENTE DE PROGRAMAÇÃO**

Primeiro Relatório da Disciplina de Programação em  
Tempo Real apresentado ao Curso de Engenharia da  
Computação.

**PROFESSOR: ANDRÉ CAVALCANTE**

**Manaus – AM  
2017**

## **SUMÁRIO**

<b>OBJETIVOS</b>	<b>4</b>
<b>INTRODUÇÃO TEÓRICA</b>	<b>5</b>
<b>RESULTADOS</b>	<b>6</b>
<b>CONCLUSÃO</b>	<b>9</b>
<b>REFERÊNCIAS</b>	<b>10</b>

## **OBJETIVOS**

A atividade tem como o propósito a criação de um ambiente de programação onde arquivos fontes compõe um projeto em que as dependências são resolvidas através da criação manual de um arquivo makefile que direciona a criação do executável da aplicação.

## FUNDAMENTAÇÃO TEÓRICA

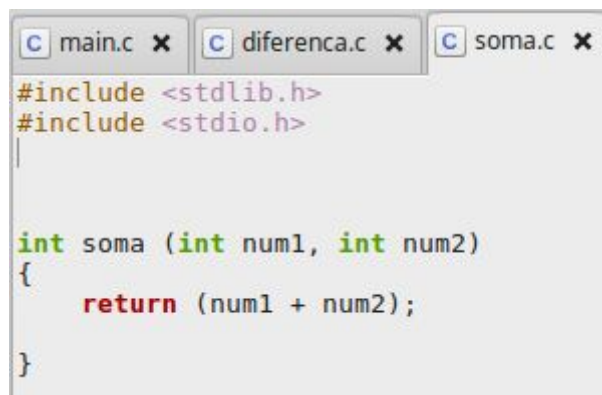
“Para programas escritos em C, antes da compilação propriamente dita, há um processo que chamamos de pré-compilação. O papel das diretivas é instruir o pré-processador (parte do compilador responsável pelo pré-processamento) a realizar determinadas tarefas. A diretiva define, por exemplo, instrui o pré-processador a substituir um texto por outro texto no código-fonte do programa. A diretiva include instrui o pré-processador a incluir num arquivo-fonte o conteúdo de um outro arquivo, geralmente chamado de header file(arquivo de cabeçalho). Estes arquivos de cabeçalho tem extensão .h, como em stdio.h.” [1]

“O programa make é uma maneira muito conveniente de gerir grandes programas ou grupos de programas. Quando se começa a escrever programas cada vez maiores e visível a diferença de tempo necessário para recompilar esses programas em comparação com programas menores. Por outro lado, normalmente trabalha-se apenas em uma pequena parte do programa (tal como uma simples função que se está depurando), e grande parte do resto do programa permanece inalterada. O programa make ajuda na manutenção desses programas observando quais partes do programa foram mudadas desde a última compilação e recompilando apenas essas partes. Para isso, é necessário que se escreva uma “makefile”, que é um arquivo de texto responsável por dizer ao programa make “o que fazer” e contém o relacionamento entre os arquivos fonte, objeto e executáveis.”[2]

## ARQUIVOS FONTES

A descrição da atividade solicita que os dois números a serem somados e subtraídos sejam passados na linha de comando, para isso utilizamos os parâmetros `argc` e `argv` na `main` do código e tratamos de forma que ele limite o usuário a passar apenas dois parâmetros válidos ou a aplicação irá exibir mensagens de erro.

As funções de soma e subtração estão declaradas e implementadas em arquivos separados e são solicitadas na exibição dos resultados na `main` do projeto, onde são incluídas as bibliotecas criadas que contém as assinaturas para tais funções.



```
C main.c x C diferenca.c x C soma.c x
#include <stdlib.h>
#include <stdio.h>

int soma (int num1, int num2)
{
    return (num1 + num2);
}
```

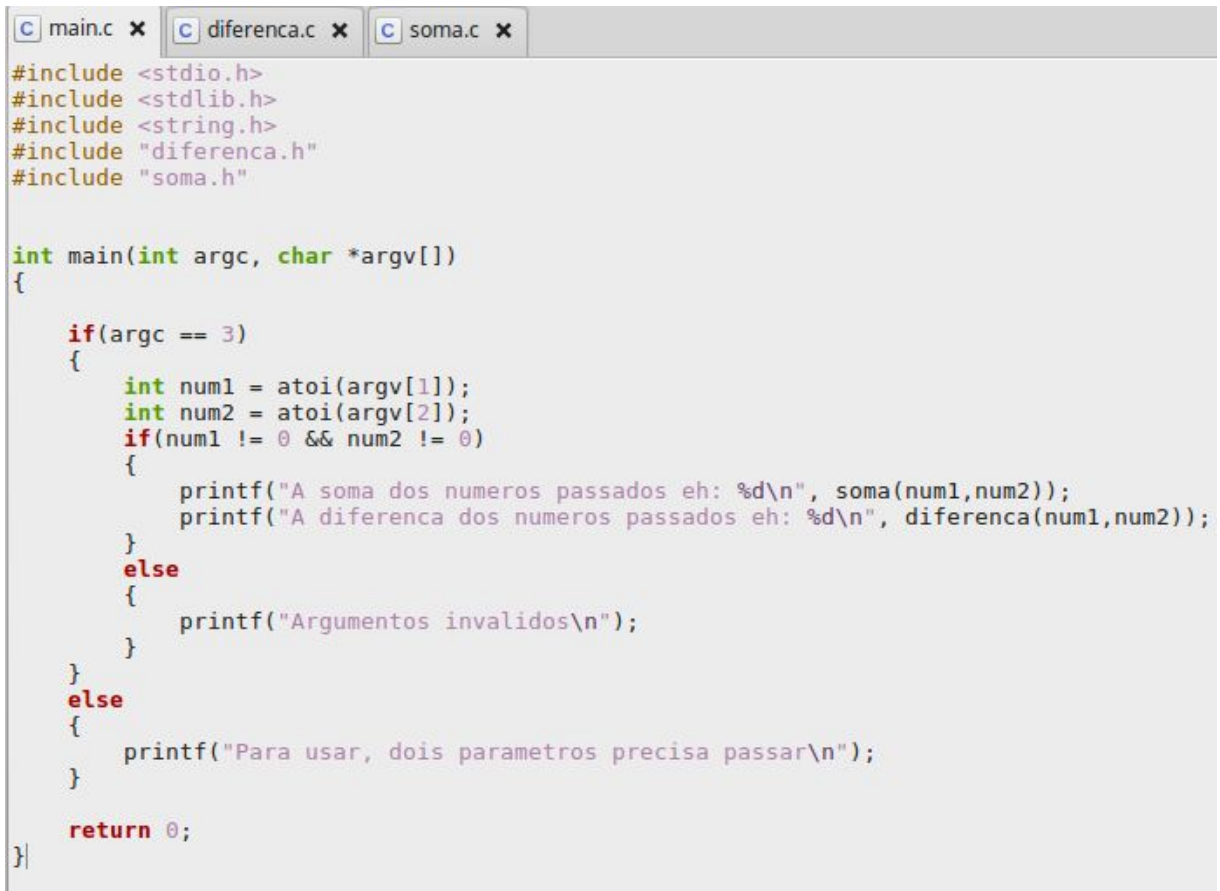
Figura 1: Arquivo soma.c



```
C main.c x C diferenca.c x C soma.c x
#include <stdlib.h>
#include <stdio.h>

int diferenca(int num1, int num2)
{
    return (num1 - num2);
}
```

Figura 2: Arquivo diferenca.c



```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "diferenca.h"
#include "soma.h"

int main(int argc, char *argv[])
{
    if(argc == 3)
    {
        int num1 = atoi(argv[1]);
        int num2 = atoi(argv[2]);
        if(num1 != 0 && num2 != 0)
        {
            printf("A soma dos numeros passados eh: %d\n", soma(num1,num2));
            printf("A diferenca dos numeros passados eh: %d\n", diferenca(num1,num2));
        }
        else
        {
            printf("Argumentos invalidos\n");
        }
    }
    else
    {
        printf("Para usar, dois parametros precisa passar\n");
    }

    return 0;
}
```

Figura 3: Arquivo main.c

## ESTRUTURA DE DIRETÓRIOS

O ambiente possui 5 arquivos fontes: main.c, soma.c, soma.h, diferenca.c, diferenca.h. A main.c é o arquivo principal do projeto e nele estão incluídas as bibliotecas soma.h e diferenca.h. O soma.c é o arquivo onde está descrita a função de soma e seu respectivo .h é onde está a assinatura da mesma o análogo se aplica para o arquivo diferenca.c. Além disso, o Makefile gera os arquivos de compilação .o de cada um dos arquivos fontes.

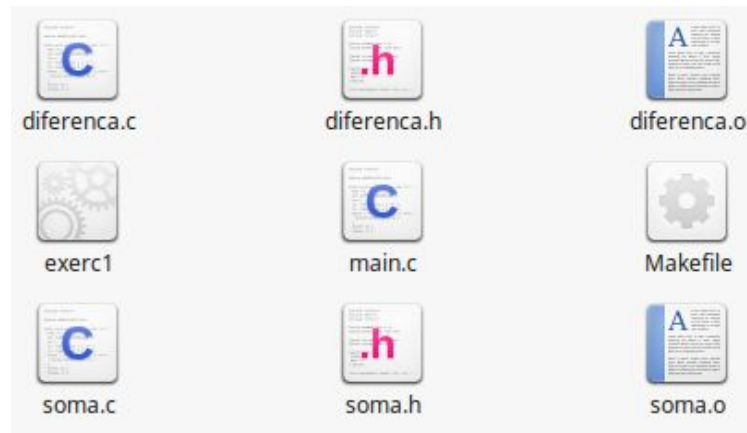


Figura 4: Todos os arquivos dentro do diretório da aplicação.

## ARQUIVO DE COMPILAÇÃO

No arquivo Makefile foram declaradas todas as dependências para cada arquivo de dentro do projeto, dessa forma possibilitando a sua execução abrindo-se um prompt de comando dentro do diretório do projeto. O nome do executável é “exerc1” e pode ser aberto como mostra nas figuras que seguem.

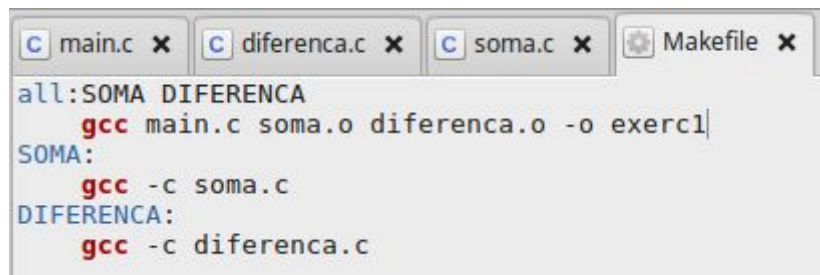


Figura 5: Arquivo Makefile criado.

```
felipedmsantos@billypc ~/Área de Trabalho/TrabalhoPTR1 $ make
gcc -c soma.c
gcc -c diferenca.c
gcc main.c soma.o diferenca.o -o exerc1
felipedmsantos@billypc ~/Área de Trabalho/TrabalhoPTR1 $ ./exerc1 1 2
A soma dos numeros passados eh: 3
A diferenca dos numeros passados eh: -1
felipedmsantos@billypc ~/Área de Trabalho/TrabalhoPTR1 $ ./exerc1 4
Para usar, dois parametros precisa passar
felipedmsantos@billypc ~/Área de Trabalho/TrabalhoPTR1 $ ./exerc1 4 f
Argumentos invalidos
felipedmsantos@billypc ~/Área de Trabalho/TrabalhoPTR1 $
```

Figura 6: Exemplos de teste para o projeto.



## **CONCLUSÃO**

Para a resolução do exercício passado a criação de um Makefile manualmente foi um processo relativamente simples e didático para o entendimento do funcionamento da estrutura por trás da compilação de um projeto. A simplicidade do procedimento se deu pelo exemplo se tratar de um ambiente com poucos arquivos, dessa forma numa ocasião onde muitos arquivos precisam ser gerenciados são necessárias ferramentas, como IDE's que criem esses Makefiles automaticamente, na medida que são adicionados novos arquivos ao projeto.

Um vantagem do uso de Makefiles se dá pelo ganho de tempo na hora da compilação, pois, em uma alteração feita em apenas um arquivo, não é necessário recompilar toda a aplicação, pois as dependências são checadas e a compilação apenas do que foi modificado fica mais simples.

## **REFERÊNCIAS**

- [1] GOMES G. F. C - Include e Makefile . Disponível em <<https://www.vivaolinux.com.br/artigo/C-Include-e-Makefile>>. Acesso em: 17 de ago. de 2017
  
- [2] DARCAMO Tutorial: Aprenda a criar seu próprio Makefile. Disponível em <<https://www.mat.uc.pt/~pedro/lectivos/ProgramacaoOrientadaObjectos/tutorialMakefilesPT.pdf>> . Acesso em: 17 de ago. de 2017.