Programação de Sistemas de Tempo Real
Laboratório 1 - Ambiente de programação

*Edson Nascimento Silva - 21350694, *Evaldo Patrik Cardoso - 21453640

I. Objetivos

Este relatório tem como objetivo a criação e descrição de um ambiente de Programação em C ANSI, no qual geramos um programa C que gera a soma e a diferença de dois números passados na linha de comando.

II. Introdução Teórica

ANSI C , ISO C e Padrão C referem-se aos padrões sucessivos para a linguagem de programação C publicados pelo American National Standards Institute (ANSI) e pela International Organization for Standardization (ISO). Historicamente, os nomes se referem especificamente à versão original e com melhor suporte do padrão (conhecida como C89 ou C90). Os desenvolvedores de software que escrevem em C são incentivados a obedecer aos padrões, pois isso ajuda a portabilidade entre os compiladores.

- C89 - Em 1983, o American National Standards Institute formou um comitê, X3J11, para estabelecer uma especificação padrão de C. A norma foi concluída em 1989 e ratificada como ANSI X3.159-1989 ”Linguagem de Programação C”. Esta versão do idioma é geralmente chamada de ”ANSI C”. Mais tarde, às vezes, o rótulo ”C89”é usado para distingui-lo do C99, mas usando o mesmo método de rotulagem.
- C90 - O mesmo padrão que C89 foi ratificado pela Organização Internacional de Padronização como ISO / IEC 9899: 1990, apenas com alterações de formatação, [2] que às vezes é chamado de C90. Portanto, os termos ”C89”e ”C90”se referem essencialmente ao mesmo idioma. Esta norma foi retirada por ANSI / INCITS [1] e ISO / IEC. [2]
-

O ANSI C agora é suportado por quase todos os compiladores amplamente usados. O GCC e o Clang são dois principais compiladores C populares hoje em dia, ambos são baseados no C11 com atualizações, incluindo alterações de especificações posteriores, como C17 e C18.

Qualquer programa escrito apenas no padrão C e sem nenhuma suposição dependente de hardware é virtualmente garantido para compilar corretamente em qualquer plataforma com uma implementação C em conformidade. Sem essas precauções, a maioria dos programas pode compilar apenas em uma determinada plataforma ou com um compilador específico, devido, por exemplo, ao uso de bibliotecas não padrão, como bibliotecas de GUI , ou à dependência de atributos específicos do compilador ou da plataforma como o tamanho exato de certos tipos de dados.

Alguns compiladores que suportam ANSI C:

- Kit de compilador de Amsterdã (C KR e C89 / 90)
- ARM RealView
- Clang , usando back-end LLVM
- GCC (completo C89 / 90, C99 e C11)
- Compilador HP C / ANSI C (C89 e C99)
- IBM XL C / C ++ (C11, começando com a versão 12.1)
- ICC da Intel
- LabWindows / CVI
- LCC
- OpenWatcom (C89 / 90 e alguns C99)
- Microsoft Visual C ++ (C89 / 90 e alguns C99)

III. Ambiente

Em nosso ambiente ultizaremos os seguintes arquivos:

- main.c - Arquivo de extensão .c onde são importados nossas dependências e bibliotrcas que estamos utilizando em nosso projetos, neste mesmo arquivo se encontra a função main(), *função principal do nosso projeto*.

```c
// imports
#include  <stdio.h>
#include  <stdlib.h>
#include  <string.h>
#include  "adicao.h"
#include  "subtracao.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 de (%d + %d) = % d \n \n " ,num1, num2, adicao(num1, num2));
      printf ( "A soma de (%d - %d) = % d \n \n " ,num1, num2, subtracao(num1, num2));
    }else{
      printf ( "Argumentos inválidos \nusage: somasub [int num1] [int num2]\n " );
    }
  }else {
    printf ( "É necessário 2 Argumentos \nusage: somasub [int num1] [int num2]\n" );
  }

  return  0 ;
}
```

Figura 1. main.c

- adicao.c - Arquivo de extensão .c onde se encontra a declaração da função *adicacao(int num1, int num2)*.

*Engenharia da Computação, Departamento de Eletrônica e Computação, Faculdade de Tecnologia, Universidade Federal do Amazonas, Manaus, AM. E-mail:, netosilvan78@gmail.com, evaldo.cardoso@icomp.ufam.edu.br

```c
// imports
#include  <stdio.h>
#include  <stdlib.h>
#include  "adicao.h"

// Função para somar dois números
int  adicao( int num1, int num2) {
  return (num1 + num2);
}
```

Figura 2. adicao.c

- adicao.h - Arquivo de extensão .h onde se encontra o protótipo da função *adicacao(int num1, int num2);*.

```c
// DEFINES
#ifndef ADICAO_H
#define ADICAO_H

int  adicao( int num1, int num2);

#endif
```

Figura 3. adicao.h

- subtracao.c - Arquivo de extensão .c onde se encontra a declaração da função *subtracao(int num1, int num2).*

```c
// imports
#include  <stdio.h>
#include  <stdlib.h>
#include "subtracao.h"

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

Figura 4. subtracao.h

- subtracao.h - Arquivo de extensão .h onde se encontra o protótipo da função *subtracao(int num1, int num2);*.

```c
// DEFINES
#ifndef SUBTRACAO_H
#define SUBTRACAO_H

int subtracao( int num1, int num2);

#endif
```

Figura 5. subtracao.h

- Makefile.mk - Arquivo de extensão .mk onde estão nossos comandos de execução do nosso projeto.

```
all: somasub

somasub: adicao.c subtracao.c main.c  #arquivos de  dependencia
    clear
    rm -rf somasub
    clear
    gcc -o somasub adicao.c subtracao.c main.c
clean: #remove arquivo somasub
    rm -rf somasub
    clear
```

Figura 6. Makefile.h

No arquivo de compilação Makefile temos as seguinte opções:

- make all ou make somasub: este comando irá limpar o console *$clear* , remover o arquivo executavel somasub caso ele exista com o comando *$rm -rf somasub*, e fazer a compilação do projeto *$gcc -o somasub adicao.c subtracao.c main.c*
- make exec: este comando irá executar o arquivo somasub caso ele exista com o comando *$./somasub 3 5* e exibirá como resultado a adição e subtração dos números inteiros 3 e 5. Os quais vemos que estão sendo passados como argumentos na execução do projeto.
- make clean: este comando irá remover o arquivo executavel somasub caso ele exista com o comando *$rm -rf somasub* e depois limpar a tela com o comando *$clear*.

IV. Conclusão

O objetivo do artigo era estabelecer um ambiente de programação utilizando padrão ANSI C. As evidências características que apareceram foram:

- A estruturação de arquivos.
- A separação entre arquivos fontes e cabeçalhos ajudam na boa prática de programação.
- A Estruturação de diretórios.
- Portabilidade.
- Grande Quantidade de compiladores os quais aceitam o padrão ANSI C.

Além disso, como mensionado anteriormente Os desenvolvedores de software que escrevem em C são incentivados a obedecer aos padrões, pois isso ajuda a portabilidadade de compiladores.

Por fim, a linguagem de programação ANSI C, às vezes chamado de linguagem de programação ANSI, continua a ser um poderoso exemplo da importância do ANSI no mundo de hoje. Este padrão de ”consenso voluntário ”permiti a compatibilidade entre compiladores de diversar instituições e arquiteturas, logo ANSI tem desempenhado um papel importante a normalização no mundo da informática , supervisionando tudo, de linguagens de programação à caracteres básicos de informática.

V. Referências

[1] INCITS/ISO/IEC 9899 Disponível em: ¡https://www.techstreet.com/standards/incits-iso-iec-9899?doc_no=incits_iso_iec—9899product_id=232462¿.Acesso em 27/09/2019

[2] ISO/IEC 9899: 1990 Linguagens de programação - C. Disponível em : ¡https://www.iso.org/standard/17782.html¿. Acesso em 27/09/2019