UNIVERSIDADE FEDERAL DE UBERLÂNDIA

FACULDADE DE ENGENHARIA MECÂNICA

Curso de Graduação em Engenharia Mecatrônica

Semana 02:

Prof. Edér Alves de Moura

Rafael de Lima Costa- 11611EMT011

UBERLÂNDIA

2023

1) Liste e descreva o que são as 4 etapas do processo de compilação.

a) No pré-processamento, o compilador processa as diretivas que começam com "#", como #include e #define, e remove os comentários.

b) Durante a compilação, o compilador analisa o código pré-processado e traduz o código-fonte gerado anteriormente para a linguagem de máquina. O compilador interpreta-o como um conjunto de instruções a serem executadas, podendo ocorrer erros de sintaxe durante esse processo.

c) Na etapa de montagem, é gerado um arquivo binário em código de máquina que está pronto para ser interpretado pelo processador.

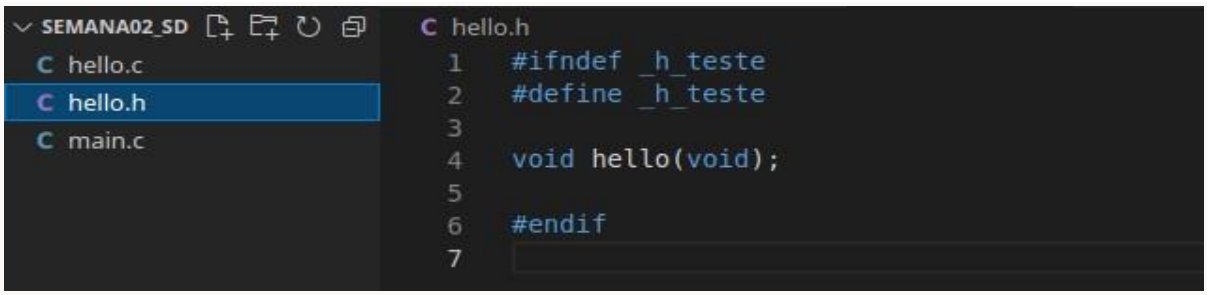
d) Por fim, na etapa de ligação, os arquivos binários (objetos) da etapa anterior são combinados para criar um executável autônomo que pode ser distribuído aos usuários.

2) Desenvolva uma aplicação simples que demonstre o uso de múltiplos arquivospara a construção de uma aplicação em C.

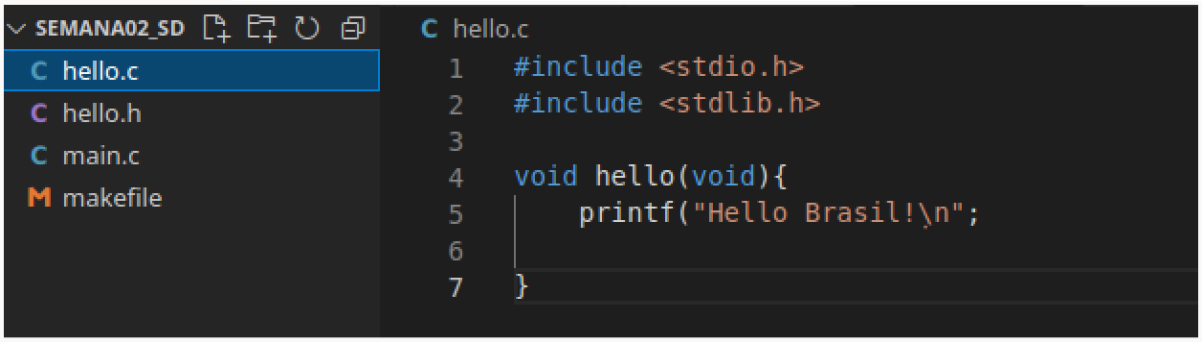
Para mostrar o uso multiplo de arquivos de uma ação simples como printar “HelloBrasil!”, foram criados três arquivos;
Um main.c

main.c

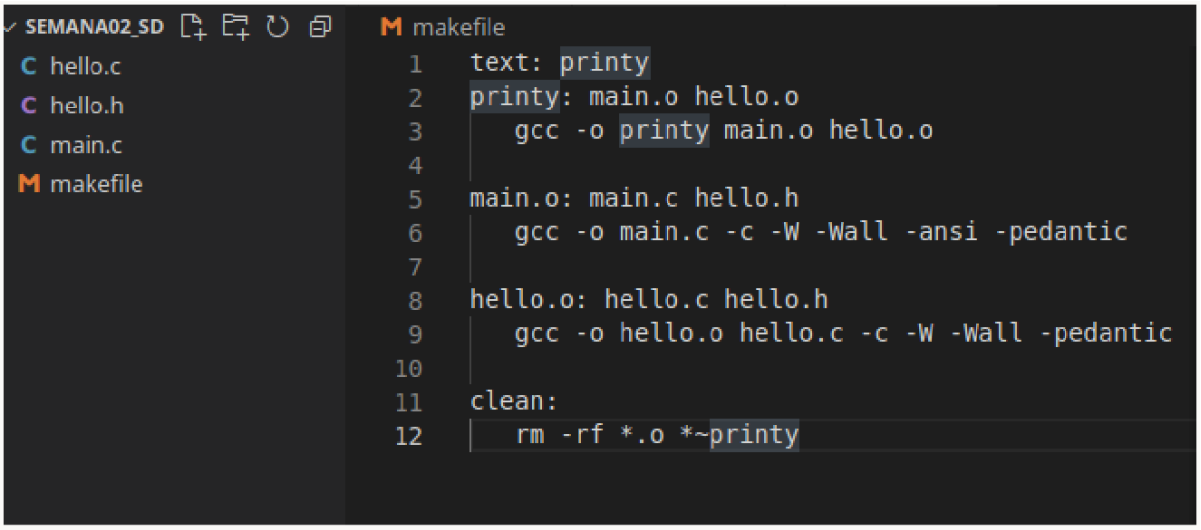
```c
#include <stdio.h>
#include <stdlib.h>
#include "hello.h"

int main(){
    hello();
    return(0);

}
```



Um e um hello.c



e logo nosso makefile



Para executar essa aplicação usamos os comendos

$ make text

$ ./printy

3) O compilador gcc permite fornecer parâmetros extras, que modificam desde a emissão de erros até o binário final, o otimizando para determinados comportamentos. Explique a função e crie um exemplo para demonstrar a funcionalidade dos seguintes parâmetros:

a) -static

Parâmetro que especifica que a vinculação deve ser estática.

Para realizar comandos utilizando a opção -static é necessário possuir o pacote glib-static. Utilizando o pacote build-essential, pode-se checar a versão utilizando o comando ldd --version. A seguir estão disponíveis os códigos e os comandos utilizados.

```c
/* Filename: lib_mylib.c

Código Adaptado da geeksforgeeks
https://www.geeksforgeeks.org/static-vs-dynamic-libraries/
*/

#include <stdio.h>
void fun_print(void)
{
  printf("Funcao executada pela biblioteca estatica\n\n");
}
```

```c
/* Filename: lib_mylib.h */

/* Código Adaptado da geeksforgeeks
https://www.geeksforgeeks.org/static-vs-dynamic-libraries/
*/

void fun_print(void);
```

```c
/* Filename: main.c  */

/* Código Adaptado da geeksforgeeks
https://www.geeksforgeeks.org/static-vs-dynamic-libraries/
*/

#include "lib_mylib.h"
void main()
{
  fun_print();
}
```

b) -g

O parâmetro “-g” gera informações de debug a serem utilizadas no GDB debugger, com algumas opções na tabela a seguir:

-g0	Sem informações de Debug
-g1	Mínimas informações de Debug
-g3	Máximas informações de Debug
-g	Informações padrão de Debug

c) -pedantic

Parâmetro que emite todos os avisos exigidos pelo padrão ANSI/ISO C.

```c
int main(void){

    int length = (int)strlen("Hello SEII\n");
    int lenght2 = (float)length;
    printf("%i", length);

    return 0;
}
```

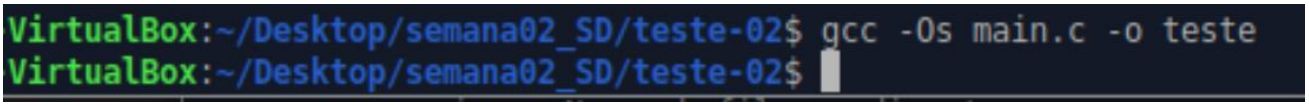
d) -Wall

Parâmetro que emite todos os avisos que o gcc pode fornecer.

```c
#include <stdio.h>

int value1 = 10, value2 = 20;

int main(void){

    int unused = 25;
    printf("O valor da multiplicacao entre %i e %i = %i\n", value1, value2,
mult(value1, value2));


    return 0;
}
```

e) -Os

Parâmetro que otimiza a compilação considerando o tamanho.



f) -O3

Parâmetro que otimiza a compilação, habilita a otimização -O2 e mais algumas funções