Executar comandos de consola no servidor de acordo com as instruções do professor.

1)

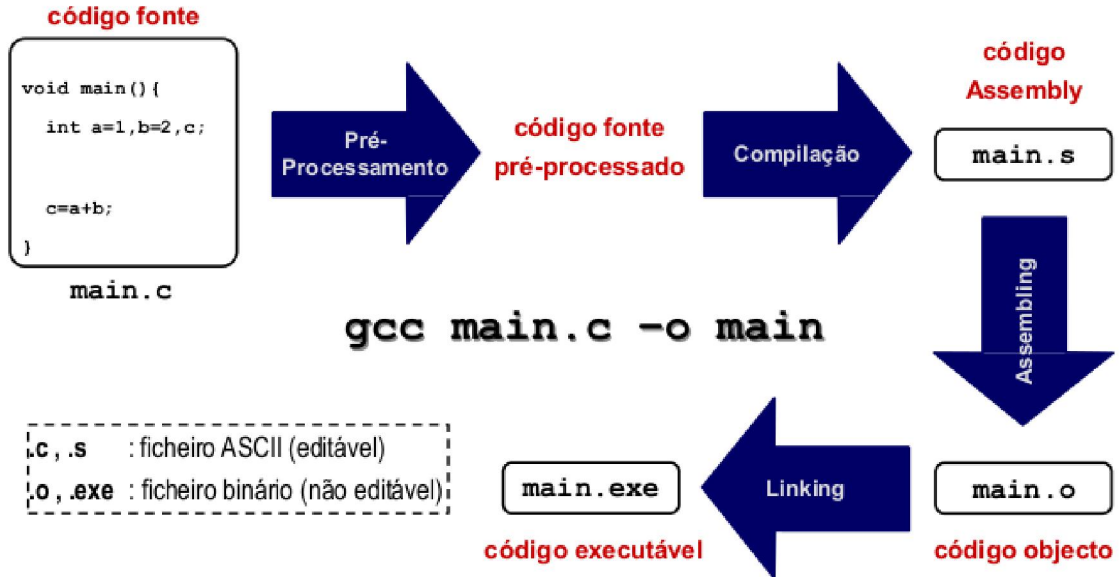
Execução:

Abrir consola na pasta onde está guardado sum_v1.c
Compilar código C num executável -> comando: gcc sum_v1.c -o nome_executavel
Correr executável criado anteriormente -> comando: ./nome_executavel

2)

Explicação:

Consultar slides de apoio para detalhes sobre cada um dos processos abaixo indicados.
Os comandos pegam no código fonte em C e correm as fases de compilação incrementalmente, parando na fase descrita. Ex: "gcc -c sum_v1.c" faz o pré-processamento, traduz para assembly e traduz o assembly para linguagem máquina, não fazendo a fase final de linkagem.



Execução:

Pré-processamento de includes (info mostrada no ecrã) -> comando: gcc -E sum_v1.c
Compilação (tradução em assembly, cria ficheiro "sum_v1.s") -> comando: gcc -S sum_v1.c
Assembling (tradução em linguagem máquina não executável) -> comando: gcc -c sum_v1.c
(Comando acima cria ficheiro "sum_v1.o")
Linkagem (com nome default: "a.out" pode ser executado) -> gcc sum_v1.c

3)

Explicação:

No exercício anterior foi executado o comando "gcc -S sum_v1.c" que traduziu o código C em código assembly, guardando no ficheiro sum_v1.s na mesma pasta. Agora vamos abri-lo, alterar uma instrução do assembly e compilar o assembly. A alteração no código deve ser visível quando testado o programa executável.

Execução:

(Feito no exercício anterior): gcc -S sum_v1.c´
Abrir "sum_v1.s" -> encontrar instrução addu, substituir por instrução subu -> guardar e fechar
Executar "gcc sum_v1.s" ou "gcc sum_v1.s -o nome_executavel"
Executar programa com "./a.out" ou "./nome_executavel" dependendo do comando usado acima.

4)

Explicação:

Um número inteiro é sempre um binário na máquina. A forma como é apresentado no ecrã depende da representação escolhida no código c para esse número. Enquanto estamos habituados a usar %d para que seja representado em decimal, é possível representar em octal %o ou hexadecimal %x. Por defeito em hexadecimal a representação é feita em complementos de 2. Assim sendo, números negativos são representados pela inversão dos bits do seu simétrico e posterior adição de 1. Ex: Convertendo 2 (0010) em -2 (1110) de acordo com as etapas anteriores. (0010 -> 1101 + 0001-> 1110)

Execução:

Abrir sum_v1.c -> localizar linha de printf, mudar ultimo "%d" para "%x" -> guardar e fechar
Executar "gcc sum_v1.c" ou "gcc sum_v1.c -o nome_executavel"
Executar programa com "./a.out" ou "./nome_executavel" dependendo do comando usado acima.

5)

Explicação:

Códigos diferentes podem ter o mesmo resultado. Se traduzirem o código fonte de "sum_v1.c "e "sum_v2.c" para assembly (comando "gcc -S sum_vx.c") irão verificar que o código resultante é diferente nos 2 casos, mas mais uma vez o resultado da execução é o mesmo.

Execução:

Abrir sum_v2.c e sum_v1.c, verificar diferenças no código, fechar.
Executar "gcc sum_v2.c" ou "gcc sum_v2.c -o nome_executavel"
Executar programa com "./a.out" ou "./nome_executavel" dependendo do comando usado acima.
Verificar que tem funcionamento idêntico a sum_v1. Nota: lembrar que sum_v1 foi alterado para mostrar a representação hexadecimal e sum_v2 não.

6)

Explicação:

Vamos aprender a criar bibliotecas de funções como o <stdio.h> ou <string.h> usados em C. Precisamos de um ficheiro header "nome.h" com a declaração de todas as funções da biblioteca e um ficheiro "nome.c" com a definição das funções declaradas no header. Ex:

soma.h:

```
int soma(int  , int  );
```

soma.c:

```
int soma(int a, int b){
  return a+b;
};
```

Para utilizar esta função soma em qualquer programa deve-se no programa alvo adicionar o include «#include "soma.h"». E por fim compilar esta biblioteca juntamente com o programa alvo na fase de linkagem (ver imagem na explicação do exercício 2).

Execução:

Abrir 2 documentos de texto, escrever em cada um deles os conteúdos na imagem acima e guardar com os nomes também acima indicados, fechar.

7)

Explicação:

Vamos criar um programa com a mesma funcionalidade que sum_v1 e sum_v2 mas que recorre à função da nossa biblioteca em vez de definir a soma no seu próprio código. Vamos compilar separadamente o nosso "sum_v1.c" e "soma.c " (criado no exercício anterior) até à fase de Assembling e de seguida fazer a linkagem em conjunto.

Execução:

Criar ficheiro "sum_v3.c" com o código na imagem. Podem copiar a maior parte da estrutura de "sum_v2.c".

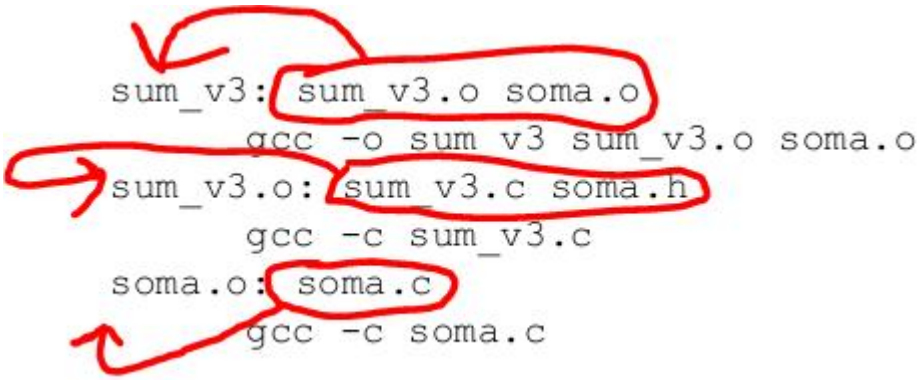
```c
#include <stdio.h>
#include "soma.h"

int main(){
  int a,b,c;

  printf("Enter the first number: ");
  scanf("%d",&a);
  printf("Enter the second number: ");
  scanf("%d",&b);

  c=soma(a,b);

  printf("The sum of %d with %d is %d \n",a,b,c);
}
```

Executar: "gcc -c sum_v3.c" e "gcc -c soma.c" (criação de ficheiros sum_v3.o e soma-o)
Executar: "gcc sum_v3.o soma.o -o sum_v3.exe" (linkagem de sum_v3.o e soma.o resultando no executável de nome "sum_v3.exe")
Executar: "./sum_v3.exe"

8)

Explicação:

De forma a encurtar o tempo de compilação de cada vez que há uma alteração aos ficheiros vamos criar um makefile, com a estrutura da imagem, que corre todas as instruções de compilação de uma vez. Estão circundados os ficheiros necessários à criação do ficheiro para que apontam, sendo "sum_v3" o executável final. *Nos exercícios 9 e 10 vamos continuar a treinar o que fizemos em 8, portanto as explicações terminam aqui, deixando apenas a execução de ambos.*



Execução:

Abrir novo documento de texto, escrever o código como acima representado, guardar com o nome "makefile" sem extensão.

Efectuar uma qualquer alteração no ficheiro "sum_v3.c" (sem alterar a funcionalidade), guardar e fechar.

Executar comando "make" na consola e correr o programa com "./sum_v3".

Verificar a funcionalidade.

9 e 10)

Execução:

Alterar os ficheiros "sum_v3.c" e makefile para o abaixo mostrado. A vermelho as alterações que devem estar presentes APENAS no exercício 10. Por favor não utilizem prints e nomes de funções iguais. Executar comando "make" e "./sum_v3", confirmar funcionalidade. Fazer alterações para exercício 10, mais uma vez executar comando "make" e "./sum_v3", confirmar funcionalidade, ez 20, ide em paz e que nosso senhor vos acompanhe.

```c
#include <stdio.h>
#include "soma.h"
#include "adicional.h"
#include "funcao.h"

int main(){
  int a,b,c;

  printf("Enter the first number: ");
  scanf("%d",&a);
  printf("Enter the second number: ");
  scanf("%d",&b);

  c=soma(a,b);

  printf("The sum of %d with %d is %d \n",a,b,c);
  adicional(c);
  printf("The result of the function with FORs is: %d \n", contafor(a, b));
  printf("The result of the function with WHILEs is: %d \n", contawhile(a, b));
  printf("The result of the function with DOs is: %d \n", contado(a, b));
}
```

```
sum_v3: sum_v3.o soma.o adicional.o funcao.o
	gcc sum_v3.o soma.o adicional.o funcao.o -o sum_v3
adicional.o: adicional.c
	gcc -c adicional.c
sum_v3.o: sum_v3.c soma.h adicional.h funcao.h
	gcc -c sum_v3.c
soma.o: soma.c
	gcc -c soma.c
funcao.o: funcao.c
	gcc -c funcao.c
```