Laboratório 01

Prof Bruno Silvestre

1. Crie o programa em C, *print.c*, utilizando um editor de texto:

```
#include <stdio.h>

int main()
{
  printf("Laboratório 01\n");
  return 0;
}
```

Utilizando um terminal de comandos, compile o programa:

```
$ gcc –o print print.c
```

E depois o execute:

```
$ ./print
```

(Este exercício é apenas para lembrar o uso do compilador)

2. Digite o comando "*man gcc*" no terminal.
Segundo a descrição, quando o *gcc* é invocado, o que ele faz?

3. Quantas opções o GCC possui?
   (a) De 1 a 5 opções.
   (b) De 5 a 10 opções.
   (c) Mais de 10 opções.

4. No entanto, o GCC não realiza todo o processo sozinho, ele delega parte do processo de criação do executável para dois outros programas: "*as*" e "*ld*". O que cada um deles faz?
Dica: utilize, no terminal do Linux, o comando *man*:

```
$ man as
$ man ld
```

5. De acordo com o trecho abaixo retirado do manual do *gcc*:

> *The "overall options" allow you to stop this process at an intermediate stage. For example, the -c option says not to run the linker. Then the output consists of object files output by the assembler.*

Podemos pedir que o processo de compilação pare em uma determinada fase da compilação.
Utilizando o comando "gcc --help", quais opções são responsáveis por:
   (a) Parar o processo logo após o pré-processador.
   (b) Parar o processo logo após gerar as instruções assembly (e antes de compilar).
   (c) Parar o processo logo depois de compilar e antes de gerar o executável.

6. Execute o comando abaixo para ativar o modo *verbose*, ou seja, o GCC mostrará o que ele está fazendo para compilar o programa:

```
$ gcc -v  -Wl,-v  -o print  print.c
```

É possível identificar as fases de compilação nos *logs* exibidos?

7. Seja o programa C, *macro.c*, que possui alguns macros:

```
#define inc(x, y)   x++; y++
#define dobro(x)    (x + x)
#define calc(x, y) (x + y * 2)

int main()
{
  int a = 10;
  int b = 20;
  if (a > 5)
    inc(a, b);

  a = dobro(++b);
  b = calc(a, 1+5);

  return 0;
}
```

Após executar este comando, o que podemos dizer da diferença entre o arquivo *macro.c* e *macro.i*?

```
$ gcc -E -o macro.i macro.c
```

8. Execute o comando abaixo no terminal:

```
$ gcc -S -o print.s print.c
```

Qual é o conteúdo do arquivo *print.s*? Conseguimos fazer uma correlação com o conteúdo do arquivo *print.c*?

9. Salve os dois arquivos C abaixo no disco:

main.c

```
#include <stdio.h>

void sayHello();

int main()
{
  sayHello();
  return 0;
}
```

say.c

```
#include <stdio.h>

void sayHello()
{
   printf("Hello!!!\n");
}
```

A diretiva “-c” faz com que o GCC transforme o arquivo C em código objeto (código binário, mas que não pode ser executado diretamente). Depois, podemos criar o executável a partir dos objetos.

Execute os comandos abaixo como exemplo:

```
$ gcc –c –o main.o  main.c
$ gcc –c –o say.o   sayc
$ gcc –o main  main.o say.o
$ ./main
```

Atividade:

- Altere o arquivo “say.c”, modificando o texto do printf().
  - Não altere “main.c”, nem “main.o”.
- Transforme “say.c” novamente em código objeto (segundo comando mostrado acima).
- Crie e execute o programa “main” com o novo “say.o” e o antigo “main.o” (terceiro comando mostrado acima).
- Execute novamente o programa “main”.