Laboratório 21
Ligação e Compilação

1. Crie o programa “vetor.c” abaixo:

```
#include <stdio.h>

#define SIZE 1048576

const int size = SIZE;
char vetor[SIZE];

int main()
{
   printf("%ld\n", sizeof(vetor));
   return 0;
}
```

Agora traduza o programa “vector.c” em Assembly usando o GCC:

```
gcc -S -o vector.s vector.c
```

a) Em que seção a string “%ld\n” do printf() e o a constante *size* foram colocadas?
b) Em que seção SIZE foi colocado?

2. Compile e execute o arquivo “vector.c”.

O programa *objdump* para fazer a decompilação de um programa. A opção “-d” decompila somente as funções. A opção “-D” decompila funções e variáveis.

a) Usando a opção “-D” diga em que seção foi colocada a variável “vetor”?

```
objdump -D vetor
```

b) Qual é o tamanho do executável?

3. Crie um arquivo “vetor2.c”, que é uma cópia de “vector.c”, mas com a seguinte modificação na definição do vetor:

char vetor[SIZE] = { [0] = 1, [512] = 2, [1024] = 3 };

ou seja, a posição [0] do vetor é inicializada com “1”, a posição [512] com “2” e a posição [1024] com “3”.

a) Usando a opção “-D” diga em que seção foi colocada a variável “vetor”?
b) Qual é o tamanho do executável?
c) Qual a diferença das seções “.bss” e “.data” ?

4. Considere o programa formado pelos arquivos abaixo:

- Arquivo temp1.h

```
extern int a;
```

- Arquivo temp1.c

```
#include <stdio.h>
#include "temp1.h"
#include "temp2.h"

int a = 1024;

int main (void) {
  foo();
  printf("%d\n", a);
  return 0;
}
```

- Arquivo temp2.h

```
extern int b;
void foo(void);
```

- Arquivo temp2.c

```
#include "temp1.h"
#include "temp2.h"

int b = 10;

void foo (void) {
  a = -3;
}
```

Note que os arquivos “temp1.h” e “temp2.h” contém declarações dos símbolos globais exportados, respectivamente, por “temp1.c” e “temp2.c”. A inclusão desses arquivos garante a consistência dessas declarações entre os módulos, tanto para os arquivos que definem os símbolos quanto para os que os importam.

Compile, ligue e execute esse programa, seguindo os passos abaixo:

```
gcc -c -Wall temp1.c
gcc -c -Wall temp2.c
gcc -o prog  temp1.o temp2.o
```

4.1. Use o programa *nm* para inspecionar os símbolos usados por cada módulo:

```
nm temp1.o
nm temp2.o
```

Você consegue inferir o significado das letras que aparecem na saída deste programa (U, T, etc.)?

Dica: use o comando “man nm”

4.2. Troque no arquivo “temp1.c” a linha

```
int a = 1024;
```

por

```
char a = 1;
```

Recompile esse módulo. O que acontece? Há algum erro? (compilação? linkedição?)

Você percebe como a inclusão dos arquivos de *header*, inclusive pelos módulos que definem os símbolos, garante a consistência das declarações.

4.3. Desfaça a mudança no arquivo “temp1.c”.

a) Troque no arquivo “temp2.c” a linha

```
#include "temp1.h"
```

por

```
extern char a;
```

Recompile os dois módulos, gere um novo executável e execute o programa.
O que aconteceu? Você consegue explicar?

b) Troque no arquivo “temp2.c” a declaração

```
extern char a;
```

por

```
int a = 1;
```

Recompile este módulo e tente gerar um novo executável.

O que acontece agora? Algum erro ocorreu? (se sim, em um passo?)
Inspecione novamente os símbolos dos módulos objeto usando o programa nm.

c) E se a troca fosse por

```
static int a = 1;
```

o que acontece? Por quê?