



Universidade Federal de Uberlândia
FEELT – Faculdade de Engenharia Elétrica



Ambiente de Programação Linux
Prof. Éder Alves de Moura
Sistemas Embarcados 2

Aluno: Pedro Jacob Favoreto - 11721EAU003
Data: 18/12/2021

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

Análise léxica: A análise léxica é a primeira fase do compilador. A função do analisador léxico, também denominado *scanner*, é ler o código-fonte, caractere a caractere, buscando a separação e identificação dos elementos componentes do programa-fonte, denominados símbolos léxicos ou *tokens*. É também de responsabilidade desta fase a eliminação de elementos "decorativos" do programa, tais como espaços em branco, marcas de formatação de texto e comentários. Existem disponíveis uma série de geradores automáticos de analisadores léxicos, como por exemplo, o *lex*. O objetivo dos geradores automáticos é limitar o esforço de programação de um analisador léxico, especificando-se apenas os *tokens* a serem reconhecidos.

Análise sintática: A análise sintática, ou análise gramatical é o processo de se determinar se uma cadeia de símbolos léxicos pode ser gerada por uma gramática. O analisador sintático é o cerne do compilador, responsável por verificar se os símbolos contidos no programa fonte formam um programa válido, ou não. No caso de analisadores sintáticos *top-down*, temos a opção de escrevê-los à mão ou gerá-los de forma automática, mas os analisadores *bottom-up* só podem ser gerados automaticamente. A maioria dos métodos de análise sintática, cai em uma dessas duas classes denominadas *top-down* e *bottom-up*. Entre os métodos *top-down* os mais importantes são a análise sintática descendente recursiva e a análise sintática preditiva não-recursiva. Entre os métodos de análise sintática *bottom-up* os mais importantes são a análise sintática de precedência de operadores, análise sintática LR canônico, análise sintática LALR e análise sintática SLR. Existem disponíveis uma série de geradores automáticos de analisadores sintáticos, como por exemplo, o *Yacc*, o *Bison* e o *JavaCC*.

Análise semântica: As análises léxica e sintática não estão preocupadas com o significado ou semântica dos programas que elas processam. O papel do analisador semântico é prover métodos pelos quais as estruturas construídas pelo analisador sintático possam ser avaliadas ou executadas. As gramáticas livres de contexto não são suficientemente poderosas para descrever uma série de construções das linguagens de programação, como por exemplo regras de escopo, regras de visibilidade e consistência de tipos. É papel do analisador semântico assegurar que todas as regras sensíveis ao contexto da linguagem estejam analisadas e verificadas quanto à sua validade. Um exemplo de tarefa própria do analisador semântico é a checagem de tipos de variáveis em expressões. Um dos mecanismos comumente utilizados por implementadores de compiladores é a Gramática de Atributos, que consiste em uma gramática livre de contexto acrescentada de um conjunto finito de atributos e um conjunto finito de predicados sobre estes atributos.

Geração de código intermediário: Na fase de geração de código intermediário, ocorre a transformação da árvore sintática em uma representação intermediária do código fonte. Esta linguagem intermediária é mais próxima da linguagem objeto do que o código fonte, mas ainda permite uma manipulação mais fácil do que se código assembly ou código de máquina fosse utilizado. Um tipo popular de linguagem intermediária é conhecido como código de três endereços. Neste tipo de código uma sentença típica tem a forma $X := A \text{ op } B$, onde X , A e B são operandos e op uma operação qualquer. Uma forma prática de representar sentenças de três endereços é através do uso de quádruplas (operador, argumento 1, argumento 2 e, resultado). Este esquema de representação de código intermediário é preferido por diversos compiladores, principalmente aqueles que executam extensivas otimizações de código, uma vez que o código intermediário pode ser rearranjado de uma maneira conveniente com facilidade. Outras representações de código intermediário comumente usadas são as triplas, (similares às quádruplas exceto pelo fato de que os resultados não são nomeados explicitamente) as árvores, os grafos acíclicos dirigidos(DAG) e a notação polonesa.

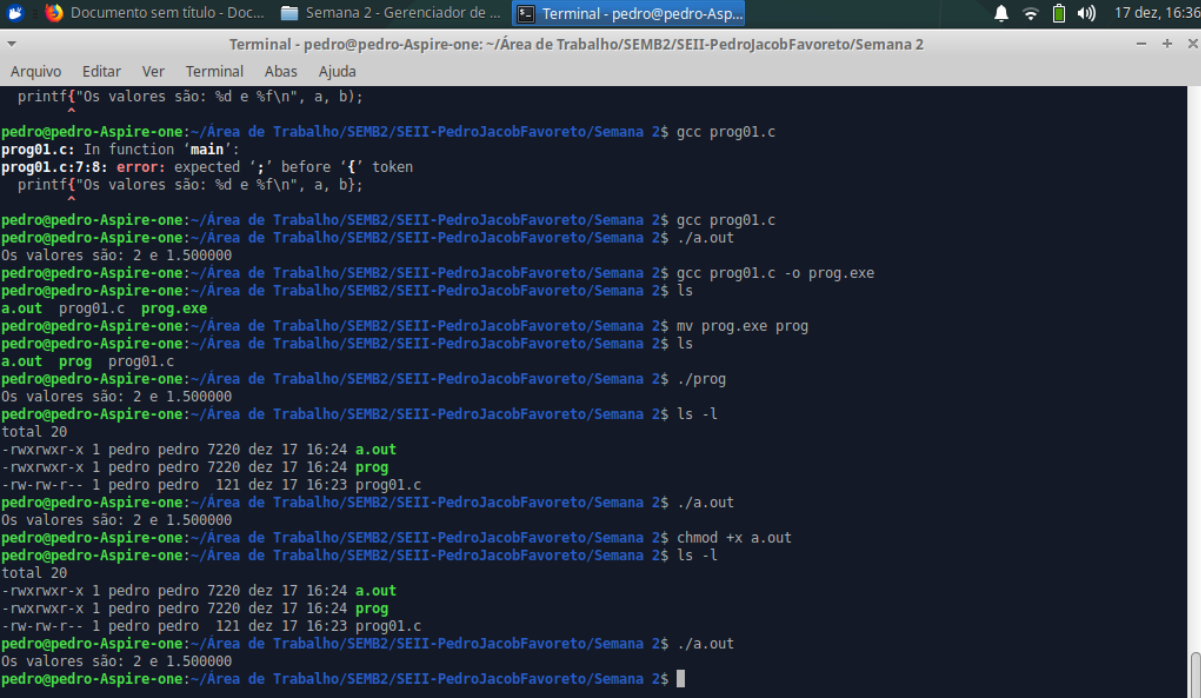
Otimização de código: A otimização de código é a estratégia de examinar o código intermediário, produzido durante a fase de geração de código com objetivo de produzir, através de algumas técnicas, um código que execute com bastante eficiência. O nome otimizador deve sempre ser encarado com cuidado, pois não se pode criar um programa que leia um programa P e gere um programa P' equivalente sendo melhor possível segundo o critério adotado. Várias técnicas e várias tarefas se reúnem sob o nome de Otimização. Estas técnicas consistem em detectar padrões dentro do código produzido e substituí-los por códigos mais eficientes. Entre as técnicas usadas estão a substituição de expressões que podem ser avaliadas durante o tempo de compilação pelos seus valores calculados, eliminação de sub expressões redundantes, desmembramento de laços, substituição de operações (multiplicação por *shifts*), entre outras. Uma das técnicas de otimização mais eficazes e independente de máquina é a otimização de laços, pois laços internos são bons candidatos para melhorias. Por exemplo, em caso de computações fixas dentro de laços, é possível mover estas computações para fora dos mesmos reduzindo o processamento.

Geração de código final: A fase de geração de código final é a última fase da compilação. A geração de um bom código objeto é difícil devido aos detalhes particulares das máquinas para os quais o código é gerado. Contudo, é uma fase importante, pois uma boa geração de código pode ser, por exemplo, duas vezes mais rápida que um algoritmo de geração de código ineficiente. Nem todas as técnicas de otimização são independentes da arquitetura da máquina-alvo. Otimização dependentes da máquina necessitam de informações tais como os limites e os recursos especiais da máquina-alvo a fim de produzir um código mais

compacto e eficiente. O código produzido pelo compilador deve se aproveitar dos recursos especiais de cada máquina-alvo. Segundo Aho, o código objeto pode ser uma sequência de instruções absolutas de máquina, uma sequência de instruções de máquina relocáveis, um programa em linguagem assembly ou um programa em outra linguagem.

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

Para o desenvolvimento dessa questão segui os passos que foram passados pelo professor durante a aula e pude perceber que em sua explicação que o ambiente linux não a diferença entre as extensões utilizadas para determinado arquivo. Desta forma o que determina um executável é um atributo daquele arquivo (rwx). Sendo assim foi feito o passo-a-passo que foi mostrado pelo professor durante a aula de laboratório.



```
Documento sem título - Doc... Semana 2 - Gerenciador de ... Terminal - pedro@pedro-Asp... 17 dez, 16:36
Terminal - pedro@pedro-Aspire-one: ~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2
Arquivo Editar Ver Terminal Abas Ajuda
printf("Os valores são: %d e %f\n", a, b);
^
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ gcc prog01.c
prog01.c: In function 'main':
prog01.c:7:8: error: expected ';' before '{' token
printf("Os valores são: %d e %f\n", a, b);
^
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ gcc prog01.c
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ ./a.out
Os valores são: 2 e 1.500000
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ gcc prog01.c -o prog.exe
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ ls
a.out prog01.c prog.exe
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ mv prog.exe prog
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ ls
a.out prog prog01.c
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ ./prog
Os valores são: 2 e 1.500000
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ ls -l
total 20
-rwxrwxr-x 1 pedro pedro 7220 dez 17 16:24 a.out
-rwxrwxr-x 1 pedro pedro 7220 dez 17 16:24 prog
-rw-rw-r-- 1 pedro pedro 121 dez 17 16:23 prog01.c
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ ./a.out
Os valores são: 2 e 1.500000
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ chmod +x a.out
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ ls -l
total 20
-rwxrwxr-x 1 pedro pedro 7220 dez 17 16:24 a.out
-rwxrwxr-x 1 pedro pedro 7220 dez 17 16:24 prog
-rw-rw-r-- 1 pedro pedro 121 dez 17 16:23 prog01.c
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ ./a.out
Os valores são: 2 e 1.500000
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$
```

```
Terminal - pedro@pedro-Aspire-one: ~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2
gcc: error: static: Arquivo ou diretório inexistente
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ gcc prog01.c -o prog -static
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ ls
prog prog01.c
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ gcc prog01.c -o prog_static -static
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ ls
prog prog01.c prog_static
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ ls -la
total 1308
drwxrwxr-x 2 pedro pedro 4096 dez 17 16:39 .
drwxrwxr-x 4 pedro pedro 4096 dez 17 13:06 ..
-rwxrwxr-x 1 pedro pedro 661764 dez 17 16:39 prog
-rw-rw-r-- 1 pedro pedro 121 dez 17 16:23 prog01.c
-rwxrwxr-x 1 pedro pedro 661764 dez 17 16:39 prog_static
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ ls -lh
total 1,3M
-rwxrwxr-x 1 pedro pedro 647K dez 17 16:39 prog
-rw-rw-r-- 1 pedro pedro 121 dez 17 16:23 prog01.c
-rwxrwxr-x 1 pedro pedro 647K dez 17 16:39 prog_static
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ ./prog
Os valores são: 2 e 1.500000
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ ./prog_static
Os valores são: 2 e 1.500000
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ ldd prog
não é um executável dinâmico
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ ldd prog_static
não é um executável dinâmico
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$
```

```
/home/pedro/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2/prog01.c - Mousepad
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include<string.h>

int main()
{
    char s1[20] = "12345";
    char s2[10] = "Pedro";
    char s3[10] = "PedroJacob";
    long int res;

    res = pow(9, 3);
    printf("Usando math.h, "
           "O valor é: %ld\n", res);

    long int a = atol(s1);
    printf("Usando stdlib.h, a string");
    printf("para long int: %ld\n", a);

    strcpy(s2, s3);
    printf("Usando string.h, as strings"
           "s2 e s3: %s %s\n",
           s2, s3);

    return 0;
}
```

3)

-Static: A opção static do compilador copia as funções externas das bibliotecas para o executável.

```
Terminal - pedro@pedro-Aspire-one: ~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2
Arquivo Editar Ver Terminal Abas Ajuda
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ gcc -static prog01.c -o prog01
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ ./prog01
Usando math.h, 0 valor é: 729
Usando stdlib.h, a stringpara long int: 12345
Usando string.h, as stringss2 e s3: PedroJacob12345 12345
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$
```

-Pedantic: Serve para avisar de erros em potencial no seu programa, como tentar usar uma atribuição em vez de comparação e muitos outros erros comuns. Como o próprio nome já diz, deixa mais pedante o compilador. Não permite funções com extensões proibidas e com o que não for compatível com ansi C.

```
Terminal - pedro@pedro-Aspire-one: ~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2
Arquivo Editar Ver Terminal Abas Ajuda
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ gcc prog01.c -pedantic -o prog01_pedantic
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ ./prog01_pedantic
Usando math.h, 0 valor é: 729
Usando stdlib.h, a stringpara long int: 12345
Usando string.h, as stringss2 e s3: PedroJacob12345 12345
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$
```

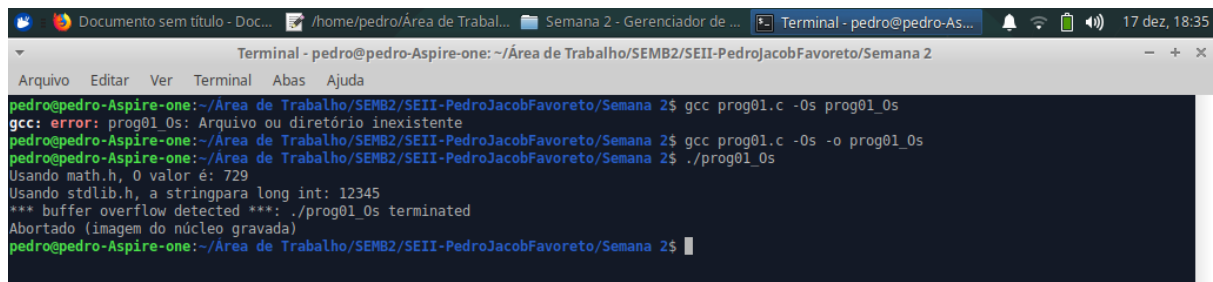
-Wall: Permite todos os warnings. Caso esteja usando funções "perigosas", coisa fora do padrão ou algo do tipo, o compilador imprimirá um aviso.

```
Terminal - pedro@pedro-Aspire-one: ~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2
Arquivo Editar Ver Terminal Abas Ajuda
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ gcc prog01.c -wall -o prog01_wal
gcc: error: unrecognized command line option '-wall'; did you mean '-Wall'?
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ Wall
Command 'Wall' not found, did you mean:
  command 'wall' from deb bsdtails
Try: sudo apt install <deb name>
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ gcc prog01.c -Wall -o prog01_wal
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ ./prog01_wal
bash: ./prog01 wall: Arquivo ou diretório inexistente
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ ./prog01_wal
Usando math.h, 0 valor é: 729
Usando stdlib.h, a stringpara long int: 12345
Usando string.h, as stringss2 e s3: PedroJacob12345 12345
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$
```

-g:

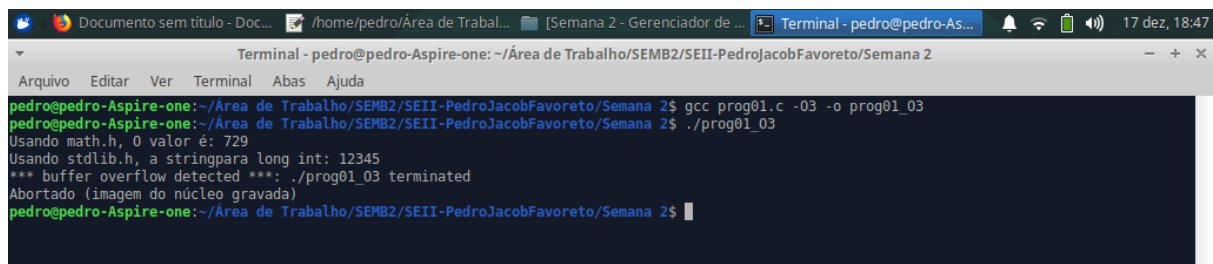
```
Terminal - pedro@pedro-Aspire-one: ~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2
Arquivo Editar Ver Terminal Abas Ajuda
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ gcc prog01.c -os -o prog01_wal
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ gcc prog01.c -Wall -o prog01_wal
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ gcc prog01.c -g -o prog01_g
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ ./prog01_g
Usando math.h, 0 valor é: 729
Usando stdlib.h, a stringpara long int: 12345
Usando string.h, as stringss2 e s3: PedroJacob12345 12345
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$
```

-Os: Realiza a otimização com o objetivo de reduzir o tamanho do código gerado. Em alguns casos, isso também pode melhorar o desempenho porque há menos código para executar.



```
Terminal - pedro@pedro-Aspire-one: ~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2
Arquivo Editar Ver Terminal Abas Ajuda
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ gcc prog01.c -O0 prog01_0s
gcc: error: prog01_0s: Arquivo ou diretório inexistente
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ gcc prog01.c -O0 -o prog01_0s
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ ./prog01_0s
Usando math.h, 0 valor é: 729
Usando stdlib.h, a stringpara long int: 12345
*** buffer overflow detected ***: ./prog01_0s terminated
Abortado (imagem do núcleo gravada)
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$
```

-O3: O objetivo deste nível de otimização é aumentar a velocidade de execução. Ele faz isso ao custo de aumentar o tamanho da imagem binária. Em alguns casos, o aumento no código pode ter o efeito de reduzir o desempenho.



```
Terminal - pedro@pedro-Aspire-one: ~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2
Arquivo Editar Ver Terminal Abas Ajuda
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ gcc prog01.c -O3 -o prog01_03
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$ ./prog01_03
Usando math.h, 0 valor é: 729
Usando stdlib.h, a stringpara long int: 12345
*** buffer overflow detected ***: ./prog01_03 terminated
Abortado (imagem do núcleo gravada)
pedro@pedro-Aspire-one:~/Área de Trabalho/SEMB2/SEII-PedroJacobFavoreto/Semana 2$
```