Aluno: Yago Castro
Disciplina: Sistemas Operacionais
Data: 18/05/22

Inicialização de programas

1. Introdução

Antes de iniciar os estudos na disciplina de Sistemas Operacionais nos foi convencionado que os programas iniciam na função *main*, porém na verdade existe uma série de processos que são realizados antes de chegar nesta etapa. Por meio do experimento proposto será possível verificar na prática a relevância que estes processos têm na compilação do programa e, consequentemente, na correta execução da função *main*.

2. Falha apresentada e correção

O programa C++ disponibilizado pelo professor possui uma função que inicializa um ponteiro do tipo inteiro com o valor “1” e uma função main escreve na tela o valor atribuído por esta variável. Entretanto, quando compilado o código utilizando o G++ ele apresentou a seguinte mensagem: “Falha de segmentação (imagem do núcleo gravada)”.

Analisando o código percebi que o problema estava no valor atribuído para o ponteiro, já que os ponteiros podem conter um endereço de memória ou o endereço de outra variável. No caso deste programa, ele estava armazenando o endereço de memória “1”, que pode estar sendo utilizado por um processo vital do sistema ou simplesmente é uma memória inválida, então o processo de ler esta variável pode ser proibido e, caso seja, irá causar uma falha de segmentação.

A correção deste sintoma é simples: bastou criar uma variável inteira inicializada com o valor “1”, fazer com que o ponteiro aponte para esta variável e no final foi possível mostrar o número “1” na tela utilizando a programação adequada. Desta forma, o problema foi resolvido e a mensagem apresentada anteriormente no terminal não voltou a aparecer.

3. Ponto de inicialização do problema

Para ter mais detalhes sobre até onde o programa pôde ser executado, executei os comandos “run” e “bt” do gdt. Para isso, realizei as seguintes instruções:

1. g++ -g test.cpp -o test
2. gdb ./test
3. run

Desta forma, me foi informado que o programa recebeu o sinal “SIGSEGV” que sinaliza falha de segmentação. Analisando os detalhes, foi possível constatar que a falha é ocasionada pelo comando “** _bar = 1*” que estava referenciando uma área de memória inválida.

```
Starting program: /home/yago/Downloads/foobar

Program received signal SIGSEGV, Segmentation fault.
Foo::Foo (this=0x555555558158 <foo>) at foobar.cxx:6
6               * _bar = 1;
```

Captura de tela do terminal contendo o comando “run”

De posse dessa informação, executei o comando “bt” (ainda estando no prompt do gdb) e obtive as informações presentes na captura de tela abaixo:

```
(gdb) bt
#0  Foo::Foo (this=0x555555558158 <foo>) at foobar.cxx:6
#1  0x000055555555528f in __static_initialization_and_destruction_0 (
    __initialize_p=1, __priority=65535) at foobar.cxx:14
#2  0x00005555555552a9 in _GLOBAL__sub_I_foo () at foobar.cxx:25
#3  0x000055555555532d in __libc_csu_init ()
#4  0x00007ffff7c04010 in __libc_start_main (
    main=0x5555555551c9 <main(int, char**)>, argc=1, argv=0x7fffffffdf48,
    init=0x5555555552e0 <__libc_csu_init>, fini=<optimized out>,
    rtld_fini=<optimized out>, stack_end=0x7fffffffdf38)
    at ../csu/libc-start.c:264
#5  0x000055555555510e in _start ()
```

Captura de tela do terminal contendo o comando “bt”

Já durante o início da compilação (*loader*) o programa detectou que o comando iria causar uma falha de segmentação e com isso não avançou do “*_libc_csu_init*” para o “*_init*” e nem do “*_libc_star_main*” para o *main*. Por conta disso, o programa não conseguiu chegar ao *main*.

4. Referências

http://dbp-consulting.com/tutorials/debugging/linuxProgramStartup.html

https://0xax.gitbooks.io/linux-insides/content/Misc/linux-misc-4.html

https://www.inf.pucrs.br/~pinho/PRGSWB/Ponteiros/ponteiros.html

https://www.ime.usp.br/~pf/algoritmos/aulas/pont.html

https://br.ccm.net/faq/10323-linguagem-c-c-c-falha-de-segmentacao