std::string 的 find 问题研究

一见 2018/12/19

目录

1. 前言

一次偶然，发现完全同一份代码，在不同机器上 find 出现两个不同执行结果，本文旨在研究 find 的“诡异”行为，找出背后的原因。

2. find 字符串

测试代码：

```
// g++ -g -o x x.cpp
#include <string>
#include <iostream>

extern "C" int main()
{
    std::string::size_type n = std::string::npos;
    std::string str = "123";
    std::string::size_type m = str.find("2", n); // 按照期望，m 值应为 npos
    std::cout << "n=" << n << ", m=" << m << std::endl;
    return 0;
}
```

i386 输出结果（gcc (GCC) 4.1.2）：

```
n=4294967295, m=1
```

这里 m 值为 1，是一个非期望的值。

i86_64 编译成 64 位输出结果（gcc (GCC) 4.8.5）：

```
n=18446744073709551615, m=18446744073709551615
```

i86_64 编译成 32 位输出结果（gcc (GCC) 4.8.5）：

```
n=4294967295, m=4294967295
```

i386 上编译放到 i86_64 上执行的输出结果：

```
n=4294967295, m=4294967295
```

i386 上编译成共享库后放到 i86_64 上执行的输出结果：

```
// g++ -g -o libx.so -fPIC -shared x.cpp
n=4294967295, m=4294967295
```

3. find 单个字符

测试代码：

```
// g++ -g -o x x.cpp
#include <string>
#include <iostream>

extern "C" int main()
{
    std::string::size_type n = std::string::npos;
    std::string str = "123";
    std::string::size_type m = str.find('2', n);
    std::cout << "n=" << n << ", m=" << m << std::endl;
    return 0;
}
```

i386 输出结果（gcc (GCC) 4.1.2）：

```
n=4294967295, m=4294967295
```

i86_64 编译成 64 位输出结果（gcc (GCC) 4.8.5）：

```
n=18446744073709551615, m=18446744073709551615
```

i86_64 编译成 32 位输出结果（gcc (GCC) 4.8.5）：

```
n=4294967295, m=4294967295
```

i386 上编译放到 i86_64 上执行的输出结果：

```
n=4294967295, m=4294967295
```

i386 上编译成共享库后放到 i86_64 上执行的输出结果：

```
n=4294967295, m=4294967295
```

4. 问题分析

对于字符串版本的 find，出现不同的结果。小技巧：加上编译选项“**-D_GLIBCXX_DEBUG**”，方可 DEBUG 进入 find。

4.1. gcc-4.1.2

以下为 i386 环境。

```
g++ -g -o x x.cpp -D_GLIBCXX_DEBUG
Breakpoint 2, main () at x.cpp:6
6           std::string::size_type n = std::string::npos;
(gdb) n
7           std::string str = "123";
(gdb)
8           std::string::size_type m = str.find("2", n);
(gdb) s
std::string::find (this=0xbfb54a10, __s=0x804b8f2 "2", __pos=4294967295) at
/usr/include/c++/4.1.2/bits/basic_string.h:1579
1579            return this->find(__s, __pos, traits_type::length(__s));
(gdb) s
std::char_traits<char>::length (__s=0x804b8f2 "2") at
/usr/include/c++/4.1.2/bits/char_traits.h:257
257           { return strlen(__s); }
(gdb) finish
Run till exit from #0 std::char_traits<char>::length (__s=0x804b8f2 "2") at
/usr/include/c++/4.1.2/bits/char_traits.h:257
0x0804a8d9 in std::string::find (this=0xbfb54a10, __s=0x804b8f2 "2", __pos=4294967295) at
/usr/include/c++/4.1.2/bits/basic_string.h:1579
1579            return this->find(__s, __pos, traits_type::length(__s));
Value returned is $1 = 1
(gdb) s
std::string::find (this=0xbfb54a10, __s=0x804b8f2 "2", __pos=4294967295, __n=1) at
/usr/include/c++/4.1.2/bits/basic_string.tcc:721
721           size_type __ret = npos;
```

```
723          if (__pos + __n <= __size)
(gdb) p __pos
$2 = 4294967295
(gdb) p __n
$3 = 1
(gdb) p __size
$4 = 3
(gdb) p __pos + __n
$5 = 0
```

4.2. gcc-4.8.2

以下为 x86_64 环境。

```
// g++ -g -o x x.cpp -m32 -D_GLIBCXX_DEBUG
Breakpoint 1, main () at x.cpp:6
6           std::string::size_type n = std::string::npos;
Missing separate debuginfos, use: debuginfo-install glibc-2.17-196.tl2.3.i686
libgcc-4.8.5-4.el7.i686 libstdc++-4.8.5-4.el7.i686
(gdb) n
7           std::string str = "123";
(gdb)
8           std::string::size_type m = str.find("2", n);
(gdb) s
std::string::find (this=0xffffd300, __s=0x80499d8 "2", __pos=4294967295) at
/usr/include/c++/4.8.2/bits/basic_string.h:1864
1864            return this->find(__s, __pos, traits_type::length(__s));
(gdb) s
std::char_traits<char>::length (__s=0x80499d8 "2") at
/usr/include/c++/4.8.2/bits/char_traits.h:259
259           { return __builtin_strlen(__s); }
(gdb) finish
Run till exit from #0 std::char_traits<char>::length (__s=0x80499d8 "2") at
/usr/include/c++/4.8.2/bits/char_traits.h:259
0x0804931f in std::string::find (this=0xffffd300, __s=0x80499d8 "2", __pos=4294967295) at
/usr/include/c++/4.8.2/bits/basic_string.h:1864
1864            return this->find(__s, __pos, traits_type::length(__s));
Value returned is $1 = 1
(gdb) s
std::string::find (this=0xffffd300, __s=0x80499d8 "2", __pos=4294967295, __n=1) at
/usr/include/c++/4.8.2/bits/basic_string.tcc:740
740           const size_type __size = this->size();
```

5. a.cpp 源代码

```
// g++ -g -o a a.cpp -ldl -m32
#include <dlfcn.h>
#include <stdio.h>

int main()
{
    typedef int (*X)();

    void* p = dlopen("./libx.so", RTLD_NOW);
    if (!p)
        printf("dlopen failed: %s\n", dlerror());
    else {
        X x = (X)dlsym(p, "main");
        (*x)();
    }
    return 0;
}
```

6. 单个字符版本 find 源码

gcc-4.1.2 版本的 find 源码，gcc-4.8.2 的实现相同。

```
// basic_string.tcc
template<typename _CharT, typename _Traits, typename _Alloc>
typename basic_string<_CharT, _Traits, _Alloc>::size_type
basic_string<_CharT, _Traits, _Alloc>::
find(_CharT __c, size_type __pos) const _GLIBCXX_NOEXCEPT
{
    size_type __ret = npos;
    const size_type __size = this->size();

    if (__pos < __size)
    {
        const _CharT* __data = _M_data();
        const size_type __n = __size - __pos;
        const _CharT* __p = traits_type::find(__data + __pos, __n, __c);
        if (__p)
            __ret = __p - __data;
    }
```

```
    return __ret;
}
```

7. 字符串版本 find 源码

7.1. gcc-4.1.2

```
// /usr/include/c++/4.1.2/bits/basic_string.h
1575       size_type
1576       find(const _CharT* __s, size_type __pos = 0) const
1577       {
1578     __glibcxx_requires_string(__s);
1579     return this->find(__s, __pos, traits_type::length(__s));
1580       }

// /usr/include/c++/4.1.2/bits/basic_string.tcc
715   template<typename _CharT, typename _Traits, typename _Alloc>
716     typename basic_string<_CharT, _Traits, _Alloc>::size_type
717     basic_string<_CharT, _Traits, _Alloc>::
718     find(const _CharT* __s, size_type __pos, size_type __n) const
719     {
720       __glibcxx_requires_string_len(__s, __n);
721       size_type __ret = npos;
722       const size_type __size = this->size();
723       if (__pos + __n <= __size) // 这里溢出
724     {
725       const _CharT* __data = _M_data();
726       const _CharT* __p = std::search(__data + __pos, __data + __size,
727                       __s, __s + __n, traits_type::eq);
728       if (__p != __data + __size || __n == 0)
729         __ret = __p - __data;
730     }
731       return __ret;
732     }
```

7.2. gcc-4.8.2

实现和 gcc-4.1.2 不同了，新的实现不存在溢出漏洞。

```
// /usr/include/c++/4.8.2/bits/basic_string.h
1860       size_type
1861       find(const _CharT* __s, size_type __pos = 0) const
1862       {
```

```
      __glibcxx_requires_string(__s);
      return this->find(__s, __pos, traits_type::length(__s));
    }

// /usr/include/c++/4.8.2/bits/basic_string.tcc
 template<typename _CharT, typename _Traits, typename _Alloc>
   typename basic_string<_CharT, _Traits, _Alloc>::size_type
   basic_string<_CharT, _Traits, _Alloc>::
   find(const _CharT* __s, size_type __pos, size_type __n) const
   {
     __glibcxx_requires_string_len(__s, __n);
     const size_type __size = this->size();
     const _CharT* __data = _M_data();

     if (__n == 0)
       return __pos <= __size ? __pos : npos;

     if (__n <= __size) // 这里不存在溢出
       {
         for (; __pos <= __size - __n; ++__pos)
           if (traits_type::eq(__data[__pos], __s[0])
               && traits_type::compare(__data + __pos + 1,
                                       __s + 1, __n - 1) == 0)
             return __pos;
       }
     return npos;
   }
```

8. 结论

一些低版本的 find 实现存在 bug，存在溢出。注：std::string::size_type 实际为 size_t，是一个无符号整数类型，在 i386 上为 4 字节无符号整数类型，在 x86_84 上为 8 字节无符号整数类型，对应的有符号类型为 ssize_t。