关于内存消毒器的使用

现在大家都已经开始学习指针了。指针使用是有很多需要注意的地方。否则，会引起一些不容察觉甚至灾难性的错误。

这里列举一些指针使用的错误。

1. 野指针。

例如：

```
/* wild-pointer.c */
#include <stdio.h>

int main()
{
    int *p;
    *p = 2;
    printf("%d\n", *p);
    return 0;
}
```

这里的 p 就是一个野指针(wild pointer)。它没有被初始化，因此没有指向任何有效地址。此时直接使用它就出现致命错误。

然而，如果你不是在集成环境中编译执行这个程序，那么这个程序极有可能不会导致错误，并且有可能得到想要的结果！

2. 指针分配内存后未释放。

例如：

```
/* no-free.c */
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int *p;
    p = (int *)malloc(sizeof(int));
    *p = 2;
    printf("%d\n", *p);
    return 0;
}
```

这里，指针 p 被分配了内存，但它不再使用后没有及时释放。这会导致内存泄漏 memory-leak 的错误。

不过，这个错误非常不明显，并且也极有可能不会导致致命错误发生。

3. 悬空指针。

```
/* dangling-pointer.c */
#include <stdio.h>
```

```
#include <stdlib.h>

int main()
{
    int *p, *q;
    p = (int *)malloc(sizeof(int));
    q = p;
    *p = 2;
    free(p);
    printf("%d\n", *q);
    return 0;
}
```

这里，指针 p 和 q 都指向了相同的内存。然后，p 释放了这段内存。但是，指针 q 不可能意识到它指向的内存已经被释放了（此时，指针 q 是一个悬空指针 dangling-pointer），此后对它的使用（如例中所示的赋值）会导致致命错误发生。类似地，如果用 free(q)释放它会发生 double-free 的错误。

同样的，如果你不是在集成环境中编译执行这个程序，那么你**极有可能不会**看到错误发生。

实际上，与上述指针非法使用的情况类似的，还有数组下标越界。

综上，就是如果无法发现指针使用错误，那么你的程序可能在错误的道路上越走越远。

为了能诊断出指针（包括数组）的内存使用违例，可以借助于内存消毒器(Sanitizer)的功能。Sanitizer 是一个独立的 C 运行时支持库。

1. 如果你在 Linux 下使用 gcc，那么可以这么用：

```
$gcc wild-pointer.c -o w -fsanitize=address
```

这个命令行将生成一个名为 w 的可执行文件。如果运行它，将会得到类似于这样的输出：

```
=================================================================
==12168==ERROR: AddressSanitizer: access-violation on unknown address
0x7ff6e7d103d0 (pc 0x7ff6e7c8106a bp 0x000000000000 sp 0x00aa95f7f7c0 T0)
==12168==The signal is caused by a WRITE memory access.
    #0 0x7ff6e7c81069 in main+0x69 (w+0x140001069)
    #1 0x7ff6e7cc2c17 in __scrt_common_main_seh
//这里省略了一些内容
==12168==ABORTING
```

通过这些信息，你可以诊断出哪里出了错误。

2. 如果你在 Windows 环境下使用 VS 2017，那么

1) 请在集成环境中编译运行程序。这可以使你看到错误的发生。常见情况是系统弹出一个窗口，其中说明了一些信息。
2) 如果要使用命令行，那么很遗憾，目前它还不支持 Sanitizer，因此极有可能无法复现错误。

3. 如果你在 Windows 环境下使用 MinGW（C-Free 的编译器就是它），那么很遗憾，

Sanitizer 不支持它，因此极有可能无法复现错误。

4. 但如果你在 Windows 环境下使用 Clang target msvc，那么就拥有了 Sanitizer 支持。不过，你得先安装 VS 2017，然后再安装 Clang。

如果已经安装好，那么请这样使用：

```
D:>clang-cl wild-pointer.c /Few.exe -fsanitize=address
```

这个命令行将会生成 w.exe 可执行代码，同时还有 w.lib、w.pdb、w.exp 三个诊断支持文件。运行 w.exe 将会导出与 Linux 下类似的错误信息出现。

不过即使这样，在 Windows 环境下还是无法检测内存泄漏问题。要检测这个错误，可以在 Linux 下这样做（如果你的系统支持的话）：

```
$gcc no-free.c -o n -fsanitize=leak
```