请看如下代码

```
// declare symbol of method to be update in run-time
extern "C" {

//#ifndef EXT_RELEASE
    // extern function symbol, we will change an instruction in this method later.
    int _ZN18YLPhraseCandidates10GetPhrasesEPt(unsigned short*);
//#endif //EXT_RELEASE

}

    // Function pointer points to method to change
    int (*methodPointer)(unsigned short*)       = NULL;
    // Get method pointer
    methodPointer = _ZN18YLPhraseCandidates10GetPhrasesEPt;
    // Seek target instruction by method address
    instruction = (unsigned char*)(methodPointer) + Instructionoffset;
```

C 语言的函数指针与 C++方法指针相比，使用起来更直接，更明了，更符合“指针”的思维。

3. 修改内存访问权限

在缺省情况下，代码段所对应的内存页的访问权限是可读，可执行，任何的写操作都会对应产生访问违例。因此，在修改指令前，必须先将指令所在内存页的访问权限设置为可写。这个任务可以由 ***mprotect()*** 函数来完成，通过使用该函数，应用程序员可以修改指定内存页的访问权限。

不过需要特别注意的是，***mprotect()*** 可能会失败，因此必须检查其返回值，只有当权限修改成功的情况下才能修改指令，否则会导致程序崩溃。

另外，***mprotect()*** 可接受的地址必须是按页对齐的，因此要把指令和方法地址转换为相应的页对齐地址。

```
// align the instruction address to page size
    pagePointer = instruction;
    pagePointer = (unsigned char*)(((int)pagePointer + PAGESIZE-1) &
~(PAGESIZE-1));
    pagePointer = pagePointer - PAGESIZE;
```