

[什么是万能按键call](#)
[切入点](#)
[CE搜索背包状态](#)
[通过内存关系找万能按键call](#)
[万能按键call参数分析](#)
[总结](#)

什么是万能按键call

当我们在游戏中敲击键盘的按键时，游戏会对键盘事件做出响应，去执行某些功能。

以口袋西游为例，按B键会打开人物背包，通过B键打开背包的这个call就是万能按键call。

只要找到这一个call，就相当于找到了所有的按键call，因为剩下的所有的快捷键功能都会走这一个call。而且这个call实用性也比较强，可以用来替代很多的游戏功能。

切入点

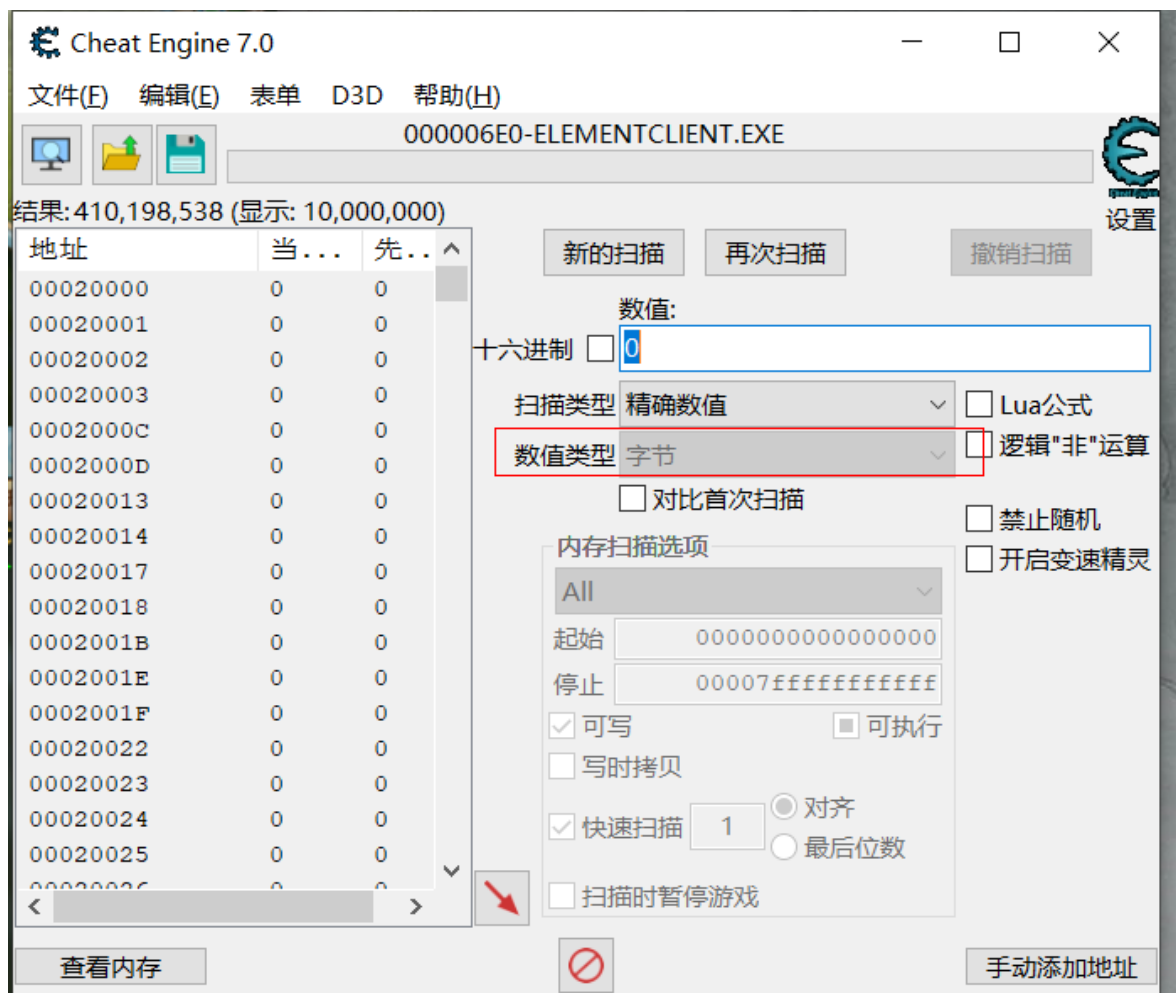
当我们点击B键时，背包会被打开，再次点击，背包会被关闭。



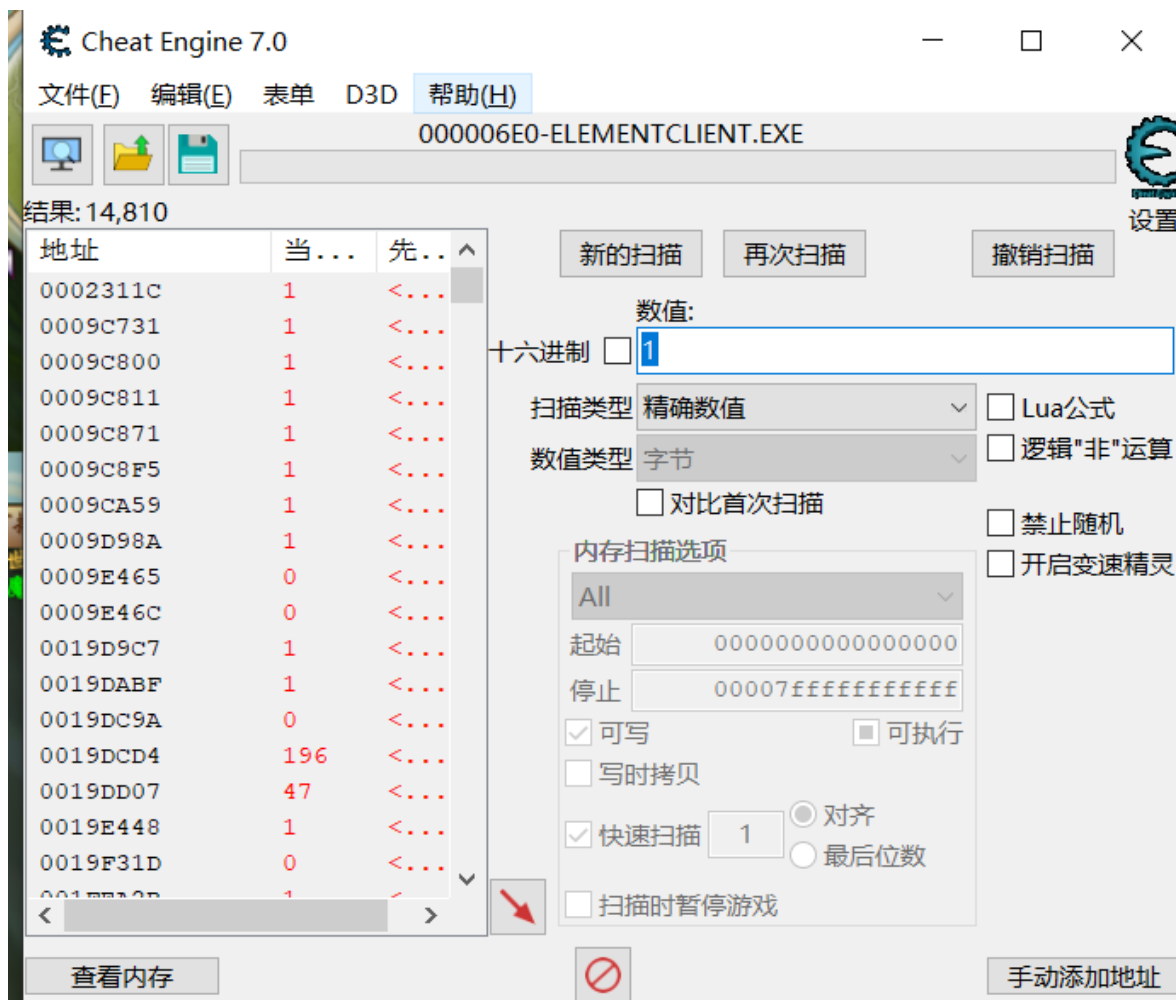
那我们就可以先用CE搜索到背包打开和关闭的状态，之后再对这个状态下一个写入断点，通过栈回溯的方式就能找到这个万能按键call。

CE搜索背包状态

背包打开和关闭的状态是一个标志位，可能是0和1，也可能不是。这里只能进行尝试，如果0和1没找到就需要利用未知的初始值和变化的值来找到这个标志位。



在背包打开状态下扫描1，注意这里需要用1字节进行扫描，4字节是扫不出来的



关闭状态下扫描0

查看内存					手动添加地址
激活	描述	地址	类型	数值	
<input type="checkbox"/>	无描述	0D9FA265	字节	1	
<input type="checkbox"/>	无描述	11A90290	字节	1	

高级选项

附加注释

重复这个过程(没啥技术含量),直到筛选出唯一一个背包状态标志位,我这里过滤剩下两个。

通过内存关系找万能按键call

接着我们在这个地址下一个字节的硬件写入断点,这里有两个要注意的点:

第一个点是吾爱的OD硬件断点有点问题,建议换个OD;

第二个点如果下了硬件断点之后这个地方断的很频繁,而且数值的变化范围超出了0和1说明标志位找错了。

然后点击B键,让游戏断下

地址	HEX	数据	反汇编	注释
008DC61E	8B06		mov eax,dword ptr ds:[esi]	ELEMENTC.00BF1F5C
008DC620	53		push ebx	
008DC621	FF50 24		call dword ptr ds:[eax+0x24]	
008DC624	8A8424 1C080000		mov al,byte ptr ss:[esp+0x81C]	
008DC62B	C786 28010000		mov dword ptr ds:[esi+0x128],0x0	
008DC635	84DB		test bl,bl	
008DC637	8886 91000000		mov byte ptr ds:[esi+0x91],al	
008DC63D	75 0E		jnz short ELEMENTC.008DC64D	
008DC63F	8A8E 90000000		mov cl,byte ptr ds:[esi+0x90]	
008DC645	84C9		test cl,cl	
008DC647	0F84 04030000		je ELEMENTC.008DC951	
008DC64D	3A9E 90000000		cmp bl,byte ptr ds:[esi+0x90]	
008DC653	0F95C1		setne cl	
008DC656	84DB		test bl,bl	
008DC658	884C24 13		mov byte ptr ss:[esp+0x13],cl	
008DC65C	0F84 AB010000		je ELEMENTC.008DC80D	
008DC662	84C0		test al,al	
008DC664	0F85 81000000		je ELEMENTC.008DC6F4	

删除硬件断点,打开调用堆栈

Jiack - ELEMENTCLIENT.EXE - [调用堆栈: 主线程]				
文件(F) 查看(V) 调试(D) 插件(P) 选项(T) 窗口(W) 帮助(H) [+] 快捷菜单 Tools BreakPoint-> 工具				
暂停				
地址	堆栈	函数过程 / 参数	调用来自	结构
0019FD8	00903DA3	ELEMENTC.0080C600	ELEMENTC.00903D9E	
0019F020	005D4E43	包含 ELEMENTC.00903DA3	ELEMENTC.005D4E40	
0019F038	0062A01D	ELEMENTC.005D4DD0	ELEMENTC.0062A018	
0019F054	00629698	ELEMENTC.006296F0	ELEMENTC.00629693	
0019F0A0	0064E5A0	包含 ELEMENTC.00629698	ELEMENTC.0064E59D	
0019F0B0	00457598	包含 ELEMENTC.0064E5A0	ELEMENTC.00457594	
0019F0D0	00449039	ELEMENTC.00457570	ELEMENTC.00449034	
0019F1E4	00469E73	ELEMENTC.00448A40	ELEMENTC.00469E6E	
0019F208	755848EB	包含 ELEMENTC.00469E73	user32.755848E9	
0019F234	7556613C	user32.755848C0	user32.75566137	
0019F318	7556528E	user32.75565D90	user32.75565289	0019F314
0019F38C	75565070	user32.75565080	user32.7556506B	0019F388
0019F398	004690D7	user32.DispatchMessageW	ELEMENTC.004690D1	0019F394
0019F39C	0019F40C	pMsg = WM_CHAR hw = C0806 ("口袋西		
0019FED8	00B7573D	? ELEMENTC.00468610	ELEMENTC.00B75738	0019FED4

挨个下断点分析参数排查，6296F0这个call就是我们要的万能按键call

万能按键call参数分析

这个call和我们之前找的call有一定的区别。正常找的call一般是调用一次，而这个地方会断下来两次。

原因在于这个call是用来响应处理按下键盘事件的，按下键盘事件实际上分为两个动作，一个是按键按下，另一个是按键抬起，对应Windows的键盘事件就是WM_KeyDown和WM_KeyUp。

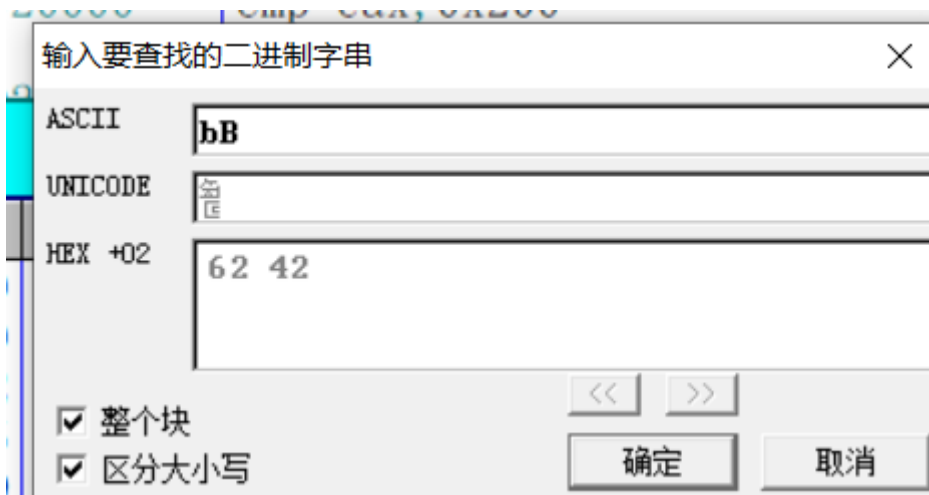
接下来分析这个call的参数

Jiack - ELEMENTCLIENT.EXE - [LCG - 主线程, 模块 - ELEMENTC]				
文件(F) 查看(V) 调试(D) 插件(P) 选项(T) 窗口(W) 帮助(H) [+] 快捷菜单 Tools BreakPoint-> 工具				
暂停				
地址	HEX 数据	反汇编	注释	寄存器 (FPU)
00629681	E8 DA512B00	call ELEMENTC.008DE860		EAX 00000100
00629686	47	inc edi	msvcrt._stricmp	ECX 2214E318
00629687	83FF 0A	cmp edi,0xA		EDX 00BEEAB4 ASCII "病Q"
0062968A	7C D5	j1 short ELEMENTC.00629661		EBX 00300001
0062968C	EB 0E	jmp short ELEMENTC.0062969C		ESP 0019F058
0062968E	53	push ebx		EBP 00000042
0062968F	55	push ebp		ESI 2214E318
00629690	50	push eax		EDI 76436F20 msvcrt._str
00629691	8BCE	mov ecx,esi		EIP 00629693 ELEMENTC.006:
00629693	E8 58000000	call ELEMENTC.006296F0	万能按键call	C 0 ES 002B 32位 0(FFFFF)
00629698	884424 13	mov byte ptr ss:[esp+0x13],al		P 1 CS 0023 32位 0(FFFFF)
0062969C	8B4424 40	mov eax,dword ptr ss:[esp+0x40]		A 0 SS 002B 32位 0(FFFFF)
006296A0	8B8E 90090000	mov ecx,dword ptr ds:[esi+0x990]		Z 1 DS 002B 32位 0(FFFFF)
006296A6	85C9	test ecx,ecx		S 0 FS 0053 32位 3D3000(
006296A8	74 2A	jg short ELEMENTC.006296D4		T 0 GS 002B 32位 0(FFFFF)
006296AA	3D 00020000	cmp eax,0x200		D 0
006296AF	74 1E	jg short ELEMENTC.006296CF		

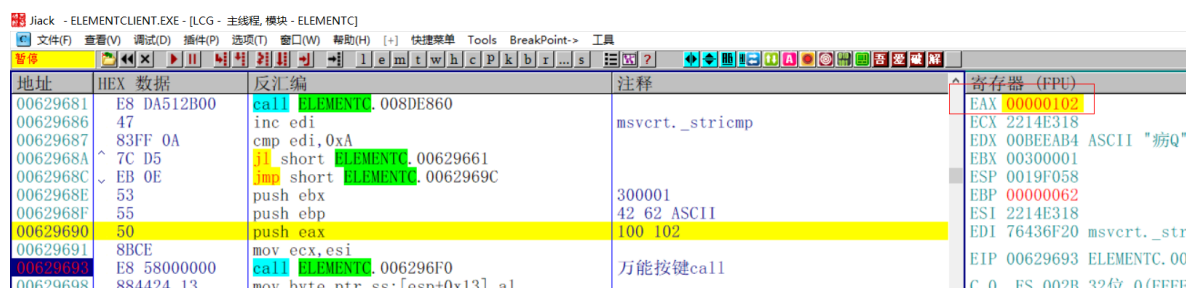
ebx是300001，无论是按下还是抬起都是一样的，具体含义暂时未知

Jiack - ELEMENTCLIENT.EXE - [LCG - 主线程, 模块 - ELEMENTC]				
文件(F) 查看(V) 调试(D) 插件(P) 选项(T) 窗口(W) 帮助(H) [+] 快捷菜单 Tools BreakPoint-> 工具				
暂停				
地址	HEX 数据	反汇编	注释	寄存器 (FPU)
00629681	E8 DA512B00	call ELEMENTC.008DE860		EAX 00000102
00629686	47	inc edi	msvcrt._stricmp	ECX 2214E318
00629687	83FF 0A	cmp edi,0xA		EDX 00BEEAB4 ASCII "病Q"
0062968A	7C D5	j1 short ELEMENTC.00629661		EBX 00300001
0062968C	EB 0E	jmp short ELEMENTC.0062969C		ESP 0019F058
0062968E	53	push ebx		EBP 00000062
0062968F	55	push ebp	42 62	ESI 2214E318
00629690	50	push eax		EDI 76436F20 msvcrt._str
00629691	8BCE	mov ecx,esi		EIP 00629693 ELEMENTC.00
00629693	E8 58000000	call ELEMENTC.006296F0	万能按键call	C 0 ES 002B 32位 0(FFFF
00629698	884424 13	mov byte ptr ss:[esp+0x13],al		P 1 CS 0023 32位 0(FFFF
0062969C	8B4424 40	mov eax,dword ptr ss:[esp+0x40]		A 0 SS 002B 32位 0(FFFF
006296A0	8B8E 90090000	mov ecx,dword ptr ds:[esi+0x990]		Z 1 DS 002B 32位 0(FFFF
006296A6	85C9	test ecx,ecx		

ebp在第一次断下的时候是0x42，第二次断下的时候是62



正好是对应按键B的ASCII值，大写的B表示按键被按下，小写的b表示键盘抬起



再看eax，按下的时候eax等于100，，抬起的时候eax等于102

```
#define WM_KEYDOWN 0x0100
#define WM_KEYUP 0x0101
```

这个参数对应的是当前的键盘状态是按下还是抬起，区别在于把KeyUp改成了102。

而ecx是一个数据结构，需要往上追ecx的来源，直到把基址追出来。这里就不一步一步追数据，本次的重点是找call，各位有兴趣可以自己再去追一下ecx数据的基址。

总结

在api断点不方便逆向的时候，可以通过内存的访问关系下断，然后利用栈回溯来找到我们需要的call。

另外这个万能按键call，如果是在实际写代码应用的时候，需要写两个call，一次是按下，一次是抬起，在按下和抬起的中间还应该加一个延迟，尽量去模拟真实的键盘事件。

相关工具：

<https://github.com/TonyChen56/GameReverseNote>