Shrinking C++ Executables

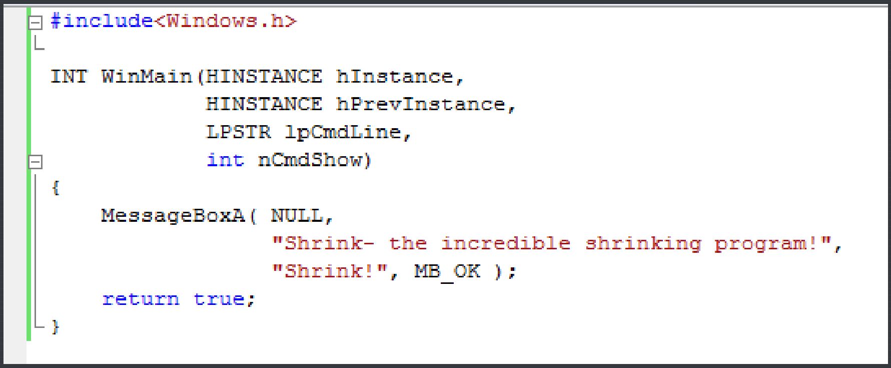
by R4ndom on Sep.10, 2012, under Intermediate, Reverse Engineering, Tutorials

Over the course of creating these tutorials, I have been confronted with attempting to make the compiled binaries small. Usually, after entering a three line program in C++, Visual Studio will assume I would like every DLL, API and function ever created by Microsoft to be included in my binary, and I end up having something close to a 6 meg file. (Don't even get me started on the fact that you can open a new Word document, type one letter, and the file no longer fits on a 32Gig USB key!)

Because you don't want the binary filled with a bunch of useless crap to detract from the learning process, the binary should ONLY contain the instructions you want used, and nothing else. You would think this would be easy- perhaps a button somewhere that says "De-crapify" or something, but this is Microsoft, so you actually have to do quite a bit of experimenting in Visual Studio to get the binary size even close to what it should actually be.

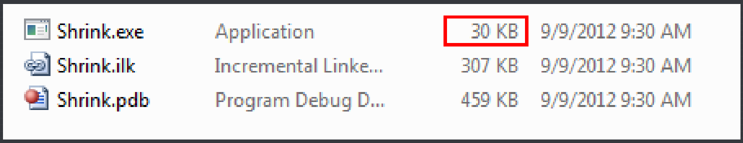
Over the weekend I did some experimenting, attempting to get the binary as small as possible and trying to figure out what all this crap is that gets inserted into our binary, and this tutorial covers what I learned. A lot of this info was performed by Zer0Flag, so many thanks (and kudos) go out to him for his hard work. If you would rather have the PDF of this tutorial, you can download it on the tutorials page. Otherwise, read on…

First off, here is the source code. It simply opens a message box with a string in it, then closes the app:



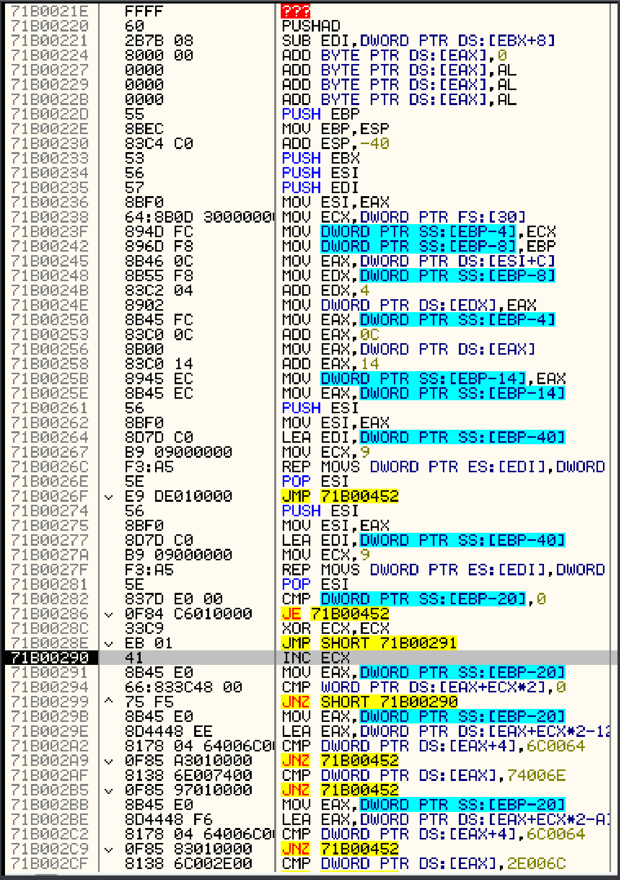
```
#include<Windows.h>

INT WinMain(HINSTANCE hInstance,
            HINSTANCE hPrevInstance,
            LPSTR lpCmdLine,
            int nCmdShow)
{
    MessageBoxA( NULL,
                 "Shrink- the incredible shrinking program!",
                 "Shrink!", MB_OK );
    return true;
}
```

Because this is basically two lines of code, you would guess that it should be like, oh, maybe 50 bytes? You guessed wrong:



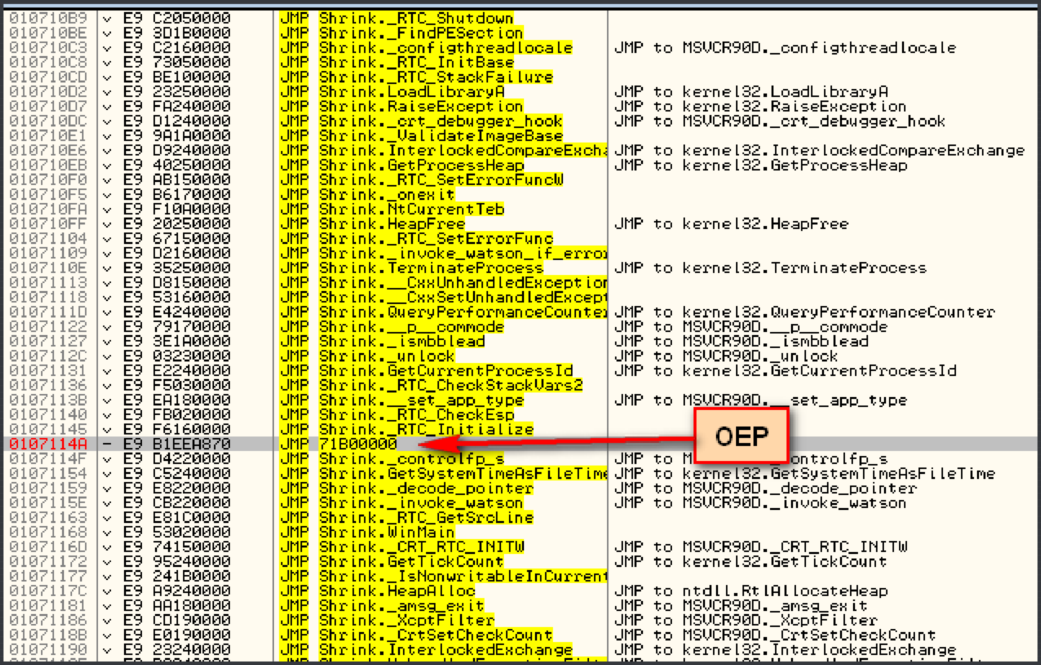
```
Shrink.exe    Application            30 KB   9/9/2012 9:30 AM
Shrink.ilk    Incremental Linke...  307 KB   9/9/2012 9:30 AM
Shrink.pdb    Program Debug D...    459 KB   9/9/2012 9:30 AM
```

Yes, try 30,000 bytes. For two lines of code? Oh, come on! Let's see what's going on in Olly:



```
71B0021E    FFFF              ???
71B00220    60                PUSHAD
71B00221    2B7B 08           SUB EDI,DWORD PTR DS:[EBX+8]
71B00224    8000 00           ADD BYTE PTR DS:[EAX],0
71B00227    0000              ADD BYTE PTR DS:[EAX],AL
71B00229    0000              ADD BYTE PTR DS:[EAX],AL
71B0022B    0000              ADD BYTE PTR DS:[EAX],AL
71B0022D    55                PUSH EBP
71B0022E    8BEC              MOV EBP,ESP
71B00230    83C4 C0           ADD ESP,-40
71B00233    53                PUSH EBX
71B00234    56                PUSH ESI
71B00235    57                PUSH EDI
71B00236    8BF0              MOV ESI,EAX
71B00238    64:8B0D 3000000   MOV ECX,DWORD PTR FS:[30]
71B0023F    894D FC           MOV DWORD PTR SS:[EBP-4],ECX
71B00242    896D F8           MOV DWORD PTR SS:[EBP-8],EBP
71B00245    8B46 0C           MOV EAX,DWORD PTR DS:[ESI+C]
71B00248    8B55 F8           MOV EDX,DWORD PTR SS:[EBP-8]
71B0024B    83C2 04           ADD EDX,4
71B0024E    8902              MOV DWORD PTR DS:[EDX],EAX
71B00250    8B45 FC           MOV EAX,DWORD PTR SS:[EBP-4]
71B00253    83C0 0C           ADD EAX,0C
71B00256    8B00              MOV EAX,DWORD PTR DS:[EAX]
71B00258    83C0 14           ADD EAX,14
71B0025B    8945 EC           MOV DWORD PTR SS:[EBP-14],EAX
71B0025E    8B45 EC           MOV EAX,DWORD PTR SS:[EBP-14]
71B00261    56                PUSH ESI
71B00262    8BF0              MOV ESI,EAX
71B00264    8D7D C0           LEA EDI,DWORD PTR SS:[EBP-40]
71B00267    B9 09000000       MOV ECX,9
71B0026C    F3:A5             REP MOVS DWORD PTR ES:[EDI],DWORD
71B0026E    5E                POP ESI
71B0026F  v E9 DE010000       JMP 71B00452
71B00274    56                PUSH ESI
71B00275    8BF0              MOV ESI,EAX
71B00277    8D7D C0           LEA EDI,DWORD PTR SS:[EBP-40]
71B0027A    B9 09000000       MOV ECX,9
71B0027F    F3:A5             REP MOVS DWORD PTR ES:[EDI],DWORD
71B00281    5E                POP ESI
71B00282    837D E0 00        CMP DWORD PTR SS:[EBP-20],0
71B00286  v 0F84 C6010000     JE 71B00452
71B0028C    33C9              XOR ECX,ECX
71B0028E  v EB 01             JMP SHORT 71B00291
71B00290    41                INC ECX
71B00291    8B45 E0           MOV EAX,DWORD PTR SS:[EBP-20]
71B00294    66:833C48 00      CMP WORD PTR DS:[EAX+ECX*2],0
71B00299  ^ 75 F5             JNZ SHORT 71B00290
71B0029B    8B45 E0           MOV EAX,DWORD PTR SS:[EBP-20]
71B0029E    8D4448 EE         LEA EAX,DWORD PTR DS:[EAX+ECX*2-12
71B002A2    8178 04 64006C0   CMP DWORD PTR DS:[EAX+4],6C0064
71B002A9  v 0F85 A3010000     JNZ 71B00452
71B002AF    8138 6E007400     CMP DWORD PTR DS:[EAX],74006E
71B002B5  v 0F85 97010000     JNZ 71B00452
71B002BB    8B45 E0           MOV EAX,DWORD PTR SS:[EBP-20]
71B002BE    8D4448 F6         LEA EAX,DWORD PTR DS:[EAX+ECX*2-A]
71B002C2    8178 04 64006C0   CMP DWORD PTR DS:[EAX+4],6C0064
71B002C9  v 0F85 83010000     JNZ 71B00452
71B002CF    8138 6C002E00     CMP DWORD PTR DS:[EAX],2E006C
```

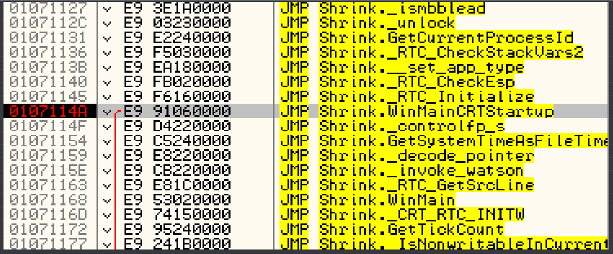
This is where Olly first breaks, at address 71B00220. After a little digging, though, I found this is not the real EP. Looking in the PE header, the real entry point is at107114A:



```
010710B9  v E9 C2050000   JMP Shrink._RTC_Shutdown
010710BE  v E9 3D1B0000   JMP Shrink._FindPESection
010710C3  v E9 C2160000   JMP Shrink._configthreadlocale      JMP to MSVCR90D._configthreadlocale
010710C8  v E9 73050000   JMP Shrink._RTC_InitBase
010710CD  v E9 BE100000   JMP Shrink._RTC_StackFailure
010710D2  v E9 23250000   JMP Shrink.LoadLibraryA             JMP to kernel32.LoadLibraryA
010710D7  v E9 FA240000   JMP Shrink.RaiseException           JMP to kernel32.RaiseException
010710DC  v E9 D1240000   JMP Shrink._crt_debugger_hook       JMP to MSVCR90D._crt_debugger_hook
010710E1  v E9 9A1A0000   JMP Shrink._ValidateImageBase
010710E6  v E9 D9240000   JMP Shrink.InterlockedCompareExcha  JMP to kernel32.InterlockedCompareExchange
010710EB  v E9 40250000   JMP Shrink.GetProcessHeap           JMP to kernel32.GetProcessHeap
010710F0  v E9 AB150000   JMP Shrink._RTC_SetErrorFuncW
010710F5  v E9 B6170000   JMP Shrink._onexit
010710FA  v E9 F10A0000   JMP Shrink.NtCurrentTeb
010710FF  v E9 20250000   JMP Shrink.HeapFree                 JMP to kernel32.HeapFree
01071104  v E9 67150000   JMP Shrink._RTC_SetErrorFunc
01071109  v E9 D2160000   JMP Shrink._invoke_watson_if_erro
0107110E  v E9 35250000   JMP Shrink.TerminateProcess         JMP to kernel32.TerminateProcess
01071113  v E9 D8150000   JMP Shrink.__CxxUnhandledExceptio
01071118  v E9 53160000   JMP Shrink.__CxxSetUnhandledExcep
0107111D  v E9 E4240000   JMP Shrink.QueryPerformanceCounte   JMP to kernel32.QueryPerformanceCounter
01071122  v E9 79170000   JMP Shrink.__p__commode             JMP to MSVCR90D.__p__commode
01071127  v E9 3E1A0000   JMP Shrink._ismbblead               JMP to MSVCR90D._ismbblead
0107112C  v E9 03230000   JMP Shrink._unlock                  JMP to MSVCR90D._unlock
01071131  v E9 E2240000   JMP Shrink.GetCurrentProcessId      JMP to kernel32.GetCurrentProcessId
01071136  v E9 F5030000   JMP Shrink._RTC_CheckStackVars2
0107113B  v E9 EA180000   JMP Shrink.__set_app_type           JMP to MSVCR90D.__set_app_type
01071140  v E9 FB020000   JMP Shrink._RTC_CheckEsp
01071145  v E9 F6160000   JMP Shrink._RTC_Initialize
0107114A  - E9 B1EEA870   JMP 71B00000
0107114F  v E9 D4220000   JMP Shrink._controlfp_s             JMP to M          ontrolfp_s
01071154  v E9 C5240000   JMP Shrink.GetSystemTimeAsFileTime  JMP to kernel32.GetSystemTimeAsFileTime
01071159  v E9 E8220000   JMP Shrink._decode_pointer          JMP to MSVCR90D._decode_pointer
0107115E  v E9 CB220000   JMP Shrink._invoke_watson           JMP to MSVCR90D._invoke_watson
01071163  v E9 E81C0000   JMP Shrink._RTC_GetSrcLine
01071168  v E9 53020000   JMP Shrink.WinMain
0107116D  v E9 74150000   JMP Shrink._CRT_RTC_INITW           JMP to MSVCR90D._CRT_RTC_INITW
01071172  v E9 95240000   JMP Shrink.GetTickCount             JMP to kernel32.GetTickCount
01071177  v E9 241B0000   JMP Shrink._IsNonwritableInCurrent
0107117C  v E9 A9240000   JMP Shrink.HeapAlloc                JMP to ntdll.RtlAllocateHeap
01071181  v E9 AA180000   JMP Shrink._amsg_exit               JMP to MSVCR90D._amsg_exit
01071186  v E9 CD190000   JMP Shrink._XcptFilter              JMP to MSVCR90D._XcptFilter
0107118B  v E9 E0190000   JMP Shrink._CrtSetCheckCount        JMP to MSVCR90D._CrtSetCheckCount
01071190  v E9 23240000   JMP Shrink.InterlockedExchange      JMP to kernel32.InterlockedExchange
```

This jumps to our initialization code. Interestingly, after we perform this code, the jump at 107114A will be dynamically changed to point to CRTMain later on. But for now, this jumps to the code in the picture above, starting at address 71B00220.

This initialization code looks for command line arguments and loads in DLLs for the application. At the end, we return to our original jump that is now changed to point to WinMainCRTStartup:

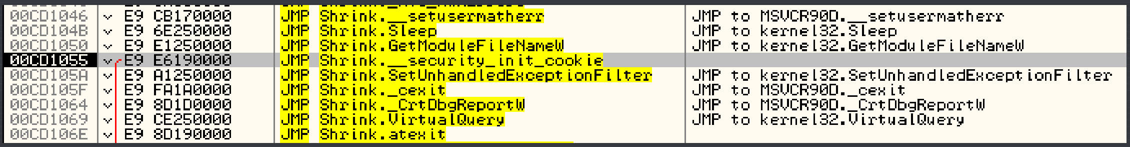


```
01071127  v E9 3E1A0000   JMP Shrink._ismbblead
0107112C  v E9 03230000   JMP Shrink._unlock
01071131  v E9 E2240000   JMP Shrink.GetCurrentProcessId
01071136  v E9 F5030000   JMP Shrink._RTC_CheckStackVars2
0107113B  v E9 EA180000   JMP Shrink.__set_app_type
01071140  v E9 FB020000   JMP Shrink._RTC_CheckEsp
01071145  v E9 F6160000   JMP Shrink._RTC_Initialize
0107114A  v E9 91060000   JMP Shrink.WinMainCRTStartup
0107114F  v E9 D4220000   JMP Shrink._controlfp_s
01071154  v E9 C5240000   JMP Shrink.GetSystemTimeAsFileTime
01071159  v E9 E8220000   JMP Shrink._decode_pointer
0107115E  v E9 CB220000   JMP Shrink._invoke_watson
01071163  v E9 E81C0000   JMP Shrink._RTC_GetSrcLine
01071168  v E9 53020000   JMP Shrink.WinMain
0107116D  v E9 74150000   JMP Shrink._CRT_RTC_INITW
01071172  v E9 95240000   JMP Shrink.GetTickCount
01071177  v E9 241B0000   JMP Shrink._IsNonwritableInCurrent
```

CRTStartup is used for loading the C RunTime libraries. The CRT provides the fundamental C++ runtime support, including:

- setup the C++ exception model
- making sure the constructor of global variables get called before entering main function
- parse command line arguments, and call the main function
- initialize the heap
- setup the atexit chain

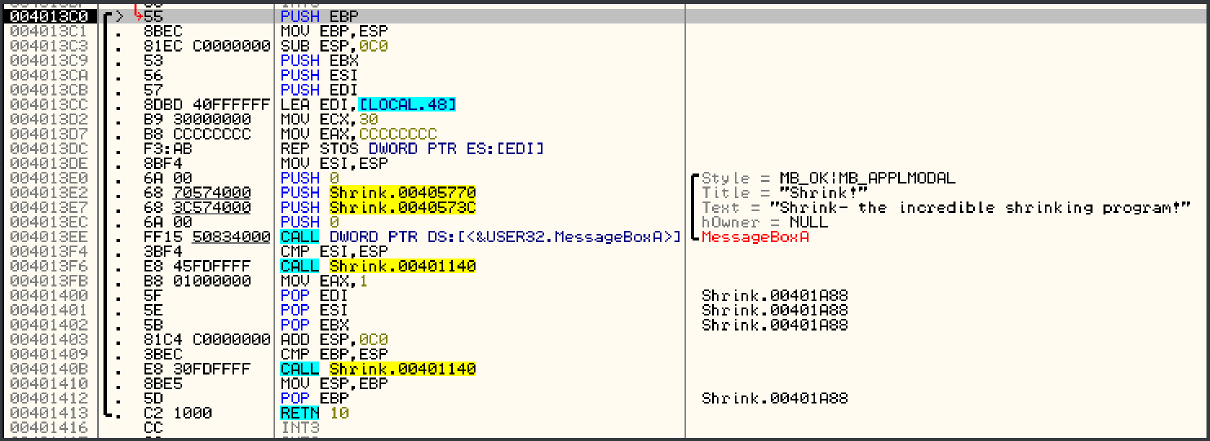
After the runtime is initialized, CRTStartup calls the __security_init_cookie function:



```
00CD1046  v E9 CB170000   JMP Shrink.__setusermatherr          JMP to MSVCR90D.__setusermatherr
00CD104B  v E9 6E250000   JMP Shrink.Sleep                     JMP to kernel32.Sleep
00CD1050  v E9 E1250000   JMP Shrink.GetModuleFileNameW        JMP to kernel32.GetModuleFileNameW
00CD1055  v E9 E6190000   JMP Shrink.__security_init_cookie
00CD105A  v E9 A1250000   JMP Shrink.SetUnhandledExceptionFilter JMP to kernel32.SetUnhandledExceptionFilter
00CD105F  v E9 FA1A0000   JMP Shrink._cexit                    JMP to MSVCR90D._cexit
00CD1064  v E9 8D1D0000   JMP Shrink._CrtDbgReportW            JMP to MSVCR90D._CrtDbgReportW
00CD1069  v E9 CE250000   JMP Shrink.VirtualQuery              JMP to kernel32.VirtualQuery
00CD106E  v E9 8D190000   JMP Shrink.atexit
```

This function detects some buffer overruns that overwrite a function’s return address, exception handler address, or certain types of parameters. Causing a buffer overrun is a technique used by hackers to exploit code that does not enforce buffer size restrictions.

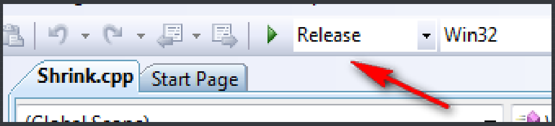
After this function checks the code for potential buffer overruns, we finally get to our actual code:



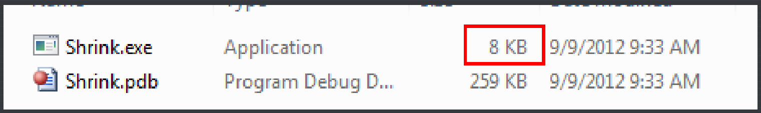
```
004013C0  > 55              PUSH EBP
004013C1  . 8BEC            MOV EBP,ESP
004013C3  . 81EC C0000000   SUB ESP,0C0
004013C9  . 53              PUSH EBX
004013CA  . 56              PUSH ESI
004013CB  . 57              PUSH EDI
004013CC  . 8DBD 40FFFFFF   LEA EDI,[LOCAL.48]
004013D2  . B9 30000000     MOV ECX,30
004013D7  . B8 CCCCCCCC     MOV EAX,CCCCCCCC
004013DC  . F3:AB           REP STOS DWORD PTR ES:[EDI]
004013DE  . 8BF4            MOV ESI,ESP
004013E0  . 6A 00           PUSH 0                                   Style = MB_OK|MB_APPLMODAL
004013E2  . 68 70574000     PUSH Shrink.00405770                     Title = "Shrink!"
004013E7  . 68 3C574000     PUSH Shrink.0040573C                     Text = "Shrink- the incredible shrinking program!"
004013EC  . 6A 00           PUSH 0                                   hOwner = NULL
004013EE  . FF15 50834000   CALL DWORD PTR DS:[<&USER32.MessageBoxA>] MessageBoxA
004013F4  . 3BF4            CMP ESI,ESP
004013F6  . E8 45FDFFFF     CALL Shrink.00401140
004013FB  . B8 01000000     MOV EAX,1
00401400  . 5F              POP EDI                                  Shrink.00401A88
00401401  . 5E              POP ESI                                  Shrink.00401A88
00401402  . 5B              POP EBX                                  Shrink.00401A88
00401403  . 81C4 C0000000   ADD ESP,0C0
00401409  . 3BEC            CMP EBP,ESP
0040140B  . E8 30FDFFFF     CALL Shrink.00401140
00401410  . 8BE5            MOV ESP,EBP
00401412  . 5D              POP EBP                                  Shrink.00401A88
00401413  . C2 1000         RETN 10
00401416    CC              INT3
```

Changing the Build

The first thing we should notice is that Visual Studio defaults to debug mode, so we should definitely change to Release:



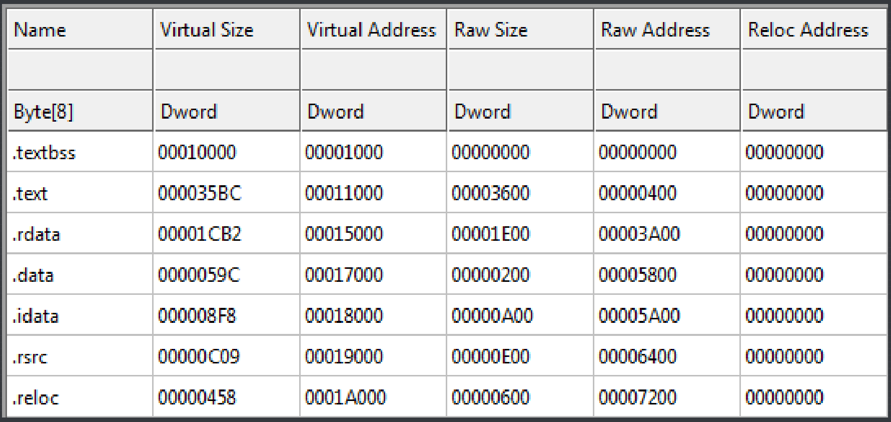
Now when we check the size, we see already a big difference:



Shrink.exe	Application	8 KB	9/9/2012 9:33 AM
Shrink.pdb	Program Debug D...	259 KB	9/9/2012 9:33 AM

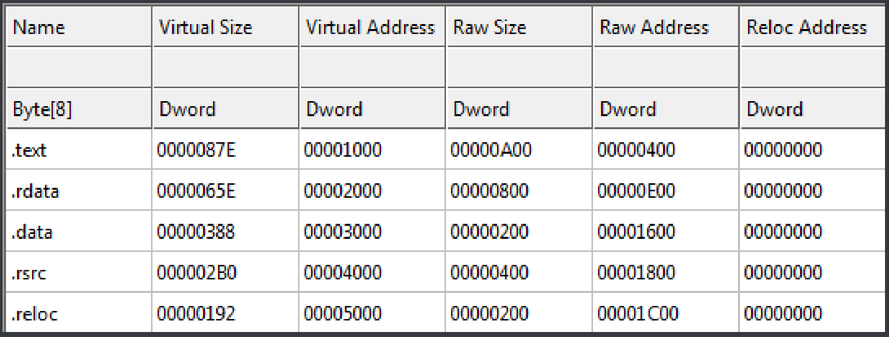
Wow, that debug information was almost 75% of the binary’s size! Loading this in CFF Explorer, we see that we lost the .textbss and .idata sections, and the other sections have been reduced drastically.

Debug version:



Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address
Byte[8]	Dword	Dword	Dword	Dword	Dword
.textbss	00010000	00001000	00000000	00000000	00000000
.text	000035BC	00011000	00003600	00000400	00000000
.rdata	00001CB2	00015000	00001E00	00003A00	00000000
.data	0000059C	00017000	00000200	00005800	00000000
.idata	000008F8	00018000	00000A00	00005A00	00000000
.rsrc	00000C09	00019000	00000E00	00006400	00000000
.reloc	00000458	0001A000	00000600	00007200	00000000

Release version:

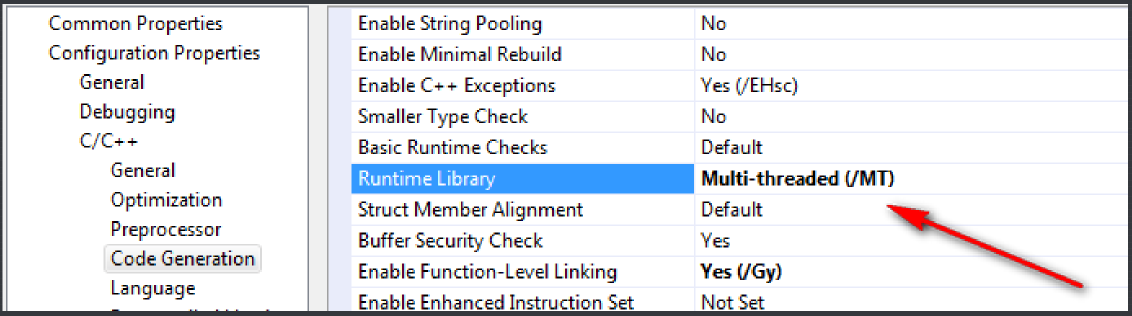


Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address
Byte[8]	Dword	Dword	Dword	Dword	Dword
.text	0000087E	00001000	00000A00	00000400	00000000
.rdata	0000065E	00002000	00000800	00000E00	00000000
.data	00000388	00003000	00000200	00001600	00000000
.rsrc	000002B0	00004000	00000400	00001800	00000000
.reloc	00000192	00005000	00000200	00001C00	00000000

Removing the C Runtime

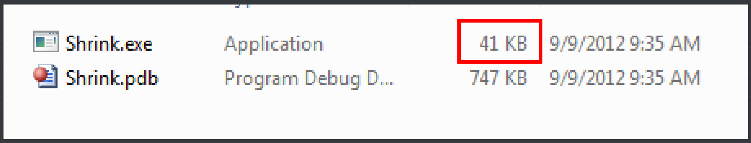
Of course, 8,000 bytes is already pretty good, but who wants to stop at "pretty good"?

Next, we have to take a step backward in order to take a couple steps forward. Right-clicking the main project's name in the Project Explorer and selecting Properties, we have the main properties window. Open the C/C++ tree and select the "Code Generation" item. We want to change the "Runtime Library" to "Multi-threaded (/MT)". This will make the binary load the C++ runtime files when the executable is loaded. The reason we want to do this is so we can manually delete it later.



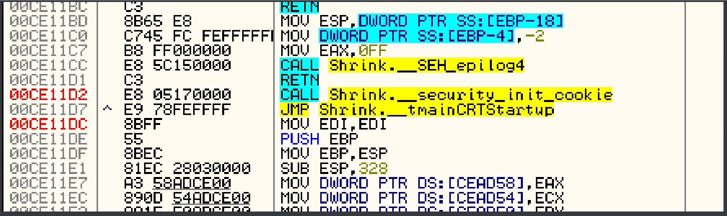
Enable String Pooling	No
Enable Minimal Rebuild	No
Enable C++ Exceptions	Yes (/EHsc)
Smaller Type Check	No
Basic Runtime Checks	Default
Runtime Library	**Multi-threaded (/MT)**
Struct Member Alignment	Default
Buffer Security Check	Yes
Enable Function-Level Linking	**Yes (/Gy)**
Enable Enhanced Instruction Set	Not Set

Changing this adds a significant amount back in, but will allow us to delete it (and more) later:



Shrink.exe	Application	41 KB	9/9/2012 9:35 AM
Shrink.pdb	Program Debug D...	747 KB	9/9/2012 9:35 AM

One thing you will notice in OllyDBG is that our jump table has all but disappeared:

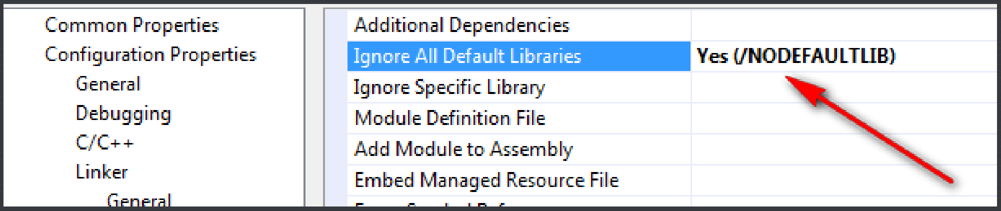


```
00CE11BC     C3              RETN
00CE11BD     8B65 E8         MOV ESP,DWORD PTR SS:[EBP-18]
00CE11C0     C745 FC FEFFFFFF MOV DWORD PTR SS:[EBP-4],-2
00CE11C7     B8 FF000000     MOV EAX,0FF
00CE11CC     E8 5C150000     CALL Shrink.__SEH_epilog4
00CE11D1     C3              RETN
00CE11D2     E8 05170000     CALL Shrink.__security_init_cookie
00CE11D7   ^ E9 78FEFFFF     JMP Shrink.__tmainCRTStartup
00CE11DC     8BFF            MOV EDI,EDI
00CE11DE     55              PUSH EBP
00CE11DF     8BEC            MOV EBP,ESP
00CE11E1     81EC 28030000   SUB ESP,328
00CE11E7     A3 58ADCE00     MOV DWORD PTR DS:[CEAD58],EAX
00CE11EC     890D 54ADCE00   MOV DWORD PTR DS:[CEAD54],ECX
```

This is because our DLLs have been inserted into our binary, so they will be called directly.

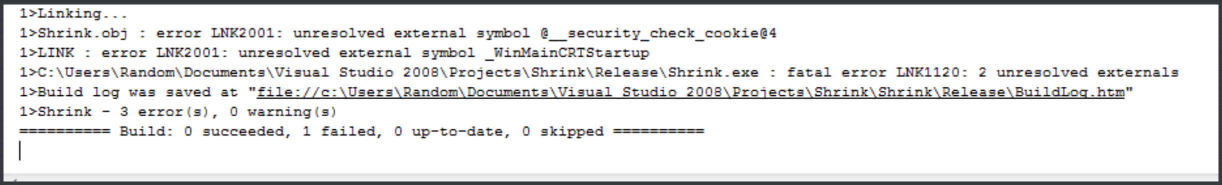
Ignore Default Libraries

Clicking on the “Input” label under the “Linker” tree, we can force Visual Studio to ignore all the default libraries usually automatically loaded.



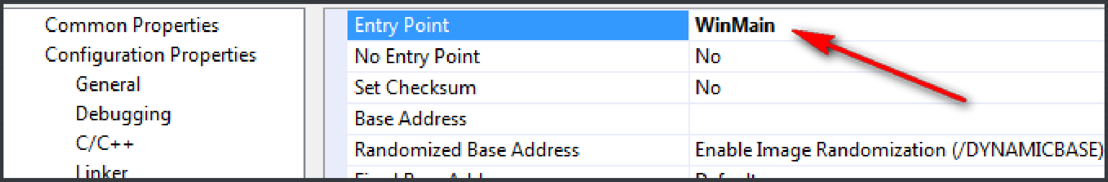
Additional Dependencies	
Ignore All Default Libraries	Yes (/NODEFAULTLIB)
Ignore Specific Library	
Module Definition File	
Add Module to Assembly	
Embed Managed Resource File	

Changing this to “Yes” and trying to build the program gives us an error though:



```
1>Linking...
1>Shrink.obj : error LNK2001: unresolved external symbol @__security_check_cookie@4
1>LINK : error LNK2001: unresolved external symbol _WinMainCRTStartup
1>C:\Users\Random\Documents\Visual Studio 2008\Projects\Shrink\Release\Shrink.exe : fatal error LNK1120: 2 unresolved externals
1>Build log was saved at "file://c:\Users\Random\Documents\Visual Studio 2008\Projects\Shrink\Shrink\Release\BuildLog.htm"
1>Shrink - 3 error(s), 0 warning(s)
========== Build: 0 succeeded, 1 failed, 0 up-to-date, 0 skipped ==========
```

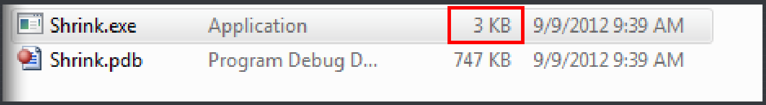
To fix this we must change the entry point of our program. The reason for this is that Visual Studio incorporates several function calls before our program actually starts, namely the CRTStartup and security_cookie calls. That means the entry point is set to these functions instead of the true beginning of our app. Since we just told Visual Studio to ignore these functions, if we don’t change the entry point it is still pointing to these functions, that are now being ignored. Clicking on the “Advanced” label under “Linker” we can change this to our actual entry point, WinMain:



Entry Point	WinMain
No Entry Point	No
Set Checksum	No
Base Address	
Randomized Base Address	Enable Image Randomization (/DYNAMICBASE)

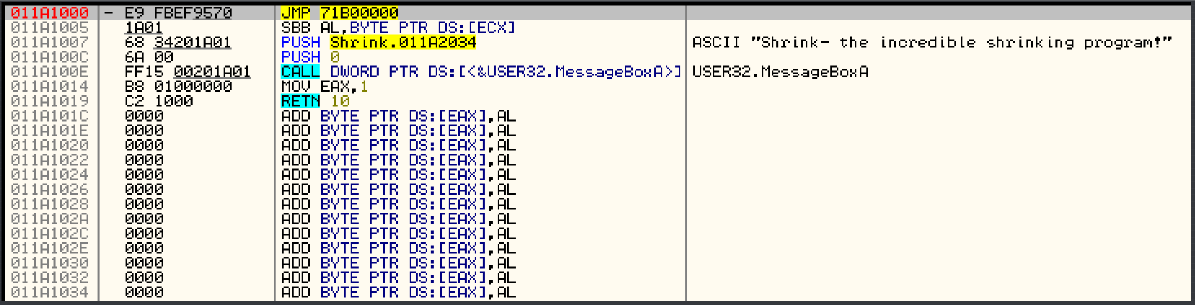
**** You may also need to change the “Buffer Security Check” option to “No (/GS-)” under C/C++ in the Code Generation tab to make it build properly. ****

Now when we build it we get no errors and also a file size of 3,000 bytes:



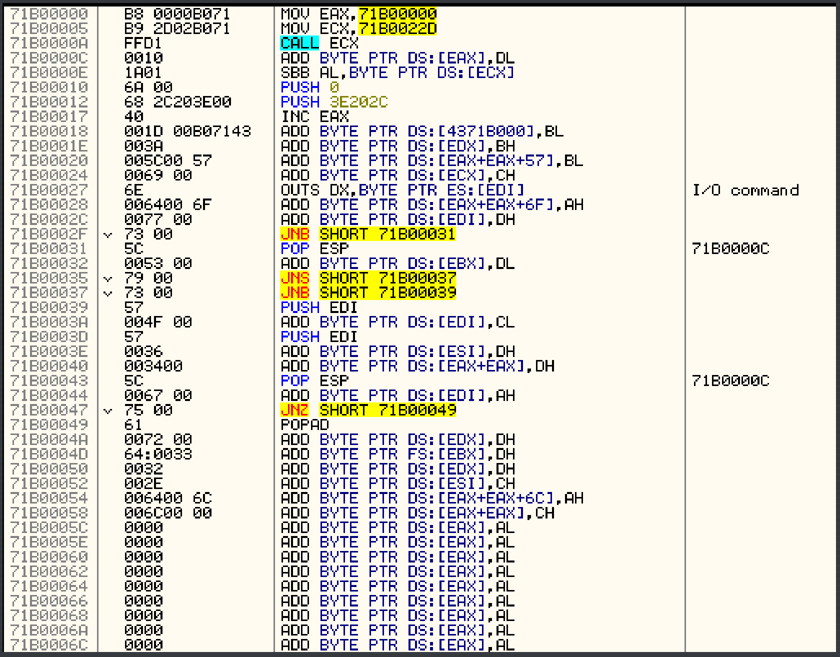
Shrink.exe	Application	3 KB	9/9/2012 9:39 AM
Shrink.pdb	Program Debug D...	747 KB	9/9/2012 9:39 AM

Now we’re talking! Loading this in Olly, we start to see some improvements:



```
011A1000  - E9 FBEF9570       JMP 71B00000
011A1005    1A01              SBB AL,BYTE PTR DS:[ECX]
011A1007    68 34201A01       PUSH Shrink.011A2034                      ASCII "Shrink- the incredible shrinking program!"
011A100C    6A 00             PUSH 0
011A100E    FF15 00201A01     CALL DWORD PTR DS:[<&USER32.MessageBoxA>] USER32.MessageBoxA
011A1014    B8 01000000       MOV EAX,1
011A1019    C2 1000           RETN 10
011A101C    0000              ADD BYTE PTR DS:[EAX],AL
011A101E    0000              ADD BYTE PTR DS:[EAX],AL
011A1020    0000              ADD BYTE PTR DS:[EAX],AL
011A1022    0000              ADD BYTE PTR DS:[EAX],AL
011A1024    0000              ADD BYTE PTR DS:[EAX],AL
011A1026    0000              ADD BYTE PTR DS:[EAX],AL
011A1028    0000              ADD BYTE PTR DS:[EAX],AL
011A102A    0000              ADD BYTE PTR DS:[EAX],AL
011A102C    0000              ADD BYTE PTR DS:[EAX],AL
011A102E    0000              ADD BYTE PTR DS:[EAX],AL
011A1030    0000              ADD BYTE PTR DS:[EAX],AL
011A1032    0000              ADD BYTE PTR DS:[EAX],AL
011A1034    0000              ADD BYTE PTR DS:[EAX],AL
```

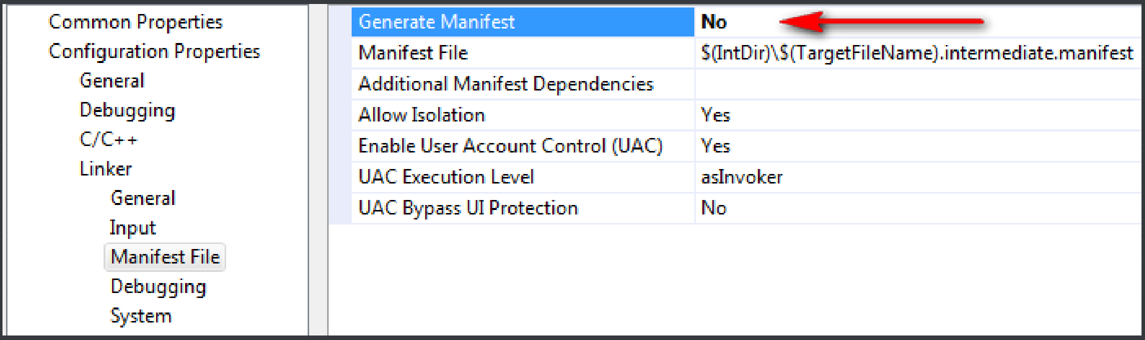
The setup code has also shrunk:



```
71B00000    B8 0000B071       MOV EAX,71B00000
71B00005    B9 2D02B071       MOV ECX,71B0022D
71B0000A    FFD1              CALL ECX
71B0000C    0010              ADD BYTE PTR DS:[EAX],DL
71B0000E    1A01              SBB AL,BYTE PTR DS:[ECX]
71B00010    6A 00             PUSH 0
71B00012    68 2C203E00       PUSH 3E202C
71B00017    40                INC EAX
71B00018    001D 00B07143     ADD BYTE PTR DS:[4371B000],BL
71B0001E    003A              ADD BYTE PTR DS:[EDX],BH
71B00020    005C00 57         ADD BYTE PTR DS:[EAX+EAX+57],BL
71B00024    0069 00           ADD BYTE PTR DS:[ECX],CH
71B00027    6E                OUTS DX,BYTE PTR ES:[EDI]                I/O command
71B00028    006400 6F         ADD BYTE PTR DS:[EAX+EAX+6F],AH
71B0002C    0077 00           ADD BYTE PTR DS:[EDI],DH
71B0002F  v 73 00             JNB SHORT 71B00031
71B00031    5C                POP ESP                                  71B0000C
71B00032    0053 00           ADD BYTE PTR DS:[EBX],DL
71B00035  v 79 00             JNS SHORT 71B00037
71B00037  v 73 00             JNB SHORT 71B00039
71B00039    57                PUSH EDI
71B0003A    004F 00           ADD BYTE PTR DS:[EDI],CL
71B0003D    57                PUSH EDI
71B0003E    0036              ADD BYTE PTR DS:[ESI],DH
71B00040    003400            ADD BYTE PTR DS:[EAX+EAX],DH
71B00043    5C                POP ESP                                  71B0000C
71B00044    0067 00           ADD BYTE PTR DS:[EDI],AH
71B00047  v 75 00             JNZ SHORT 71B00049
71B00049    61                POPAD
71B0004A    0072 00           ADD BYTE PTR DS:[EDX],DH
71B0004D    64:0033           ADD BYTE PTR FS:[EBX],DH
71B00050    0032              ADD BYTE PTR DS:[EDX],DH
71B00052    002E              ADD BYTE PTR DS:[ESI],CH
71B00054    006400 6C         ADD BYTE PTR DS:[EAX+EAX+6C],AH
71B00058    006C00 00         ADD BYTE PTR DS:[EAX+EAX],CH
71B0005C    0000              ADD BYTE PTR DS:[EAX],AL
71B0005E    0000              ADD BYTE PTR DS:[EAX],AL
71B00060    0000              ADD BYTE PTR DS:[EAX],AL
71B00062    0000              ADD BYTE PTR DS:[EAX],AL
71B00064    0000              ADD BYTE PTR DS:[EAX],AL
71B00066    0000              ADD BYTE PTR DS:[EAX],AL
71B00068    0000              ADD BYTE PTR DS:[EAX],AL
71B0006A    0000              ADD BYTE PTR DS:[EAX],AL
71B0006C    0000              ADD BYTE PTR DS:[EAX],AL
```

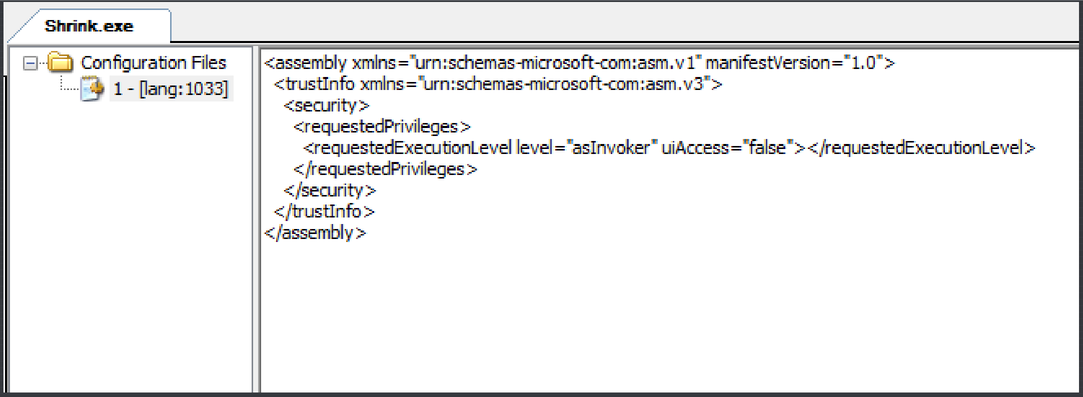
Removing the Manifest

Next we want to ditch the manifest as it's never used (at least not in our case). Under Linker, click Manifest File and change "Generate Manifest" to "No":



Generate Manifest	**No**
Manifest File	$(IntDir)\$(TargetFileName).intermediate.manifest
Additional Manifest Dependencies	
Allow Isolation	Yes
Enable User Account Control (UAC)	Yes
UAC Execution Level	asInvoker
UAC Bypass UI Protection	No

Doing this only saves about 200 bytes, but hey, that's something 😀 Here we can see exactly what the manifest looks like (in CFF Explorer):

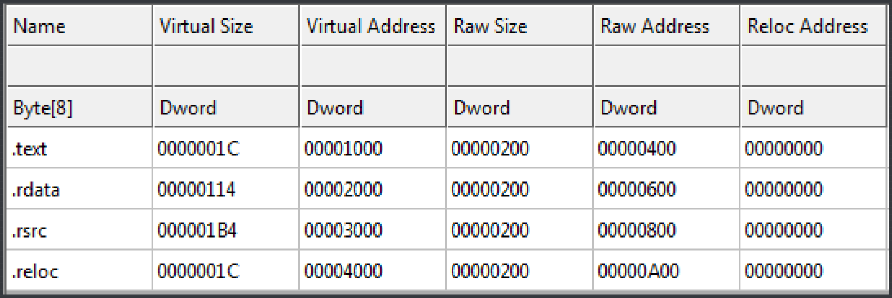


Shrink.exe

Configuration Files
1 - [lang:1033]

```
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
 <trustInfo xmlns="urn:schemas-microsoft-com:asm.v3">
  <security>
   <requestedPrivileges>
    <requestedExecutionLevel level="asInvoker" uiAccess="false"></requestedExecutionLevel>
   </requestedPrivileges>
  </security>
 </trustInfo>
</assembly>
```

The next thing we may notice is that our binary has four sections:

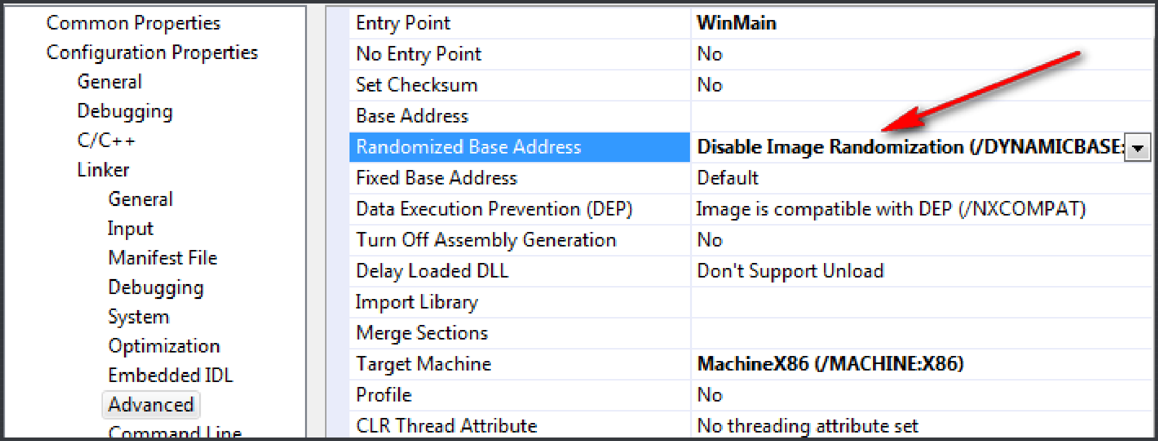


Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address
Byte[8]	Dword	Dword	Dword	Dword	Dword
.text	0000001C	00001000	00000200	00000400	00000000
.rdata	00000114	00002000	00000200	00000600	00000000
.rsrc	000001B4	00003000	00000200	00000800	00000000
.reloc	0000001C	00004000	00000200	00000A00	00000000

One that we could potentially lose is the .reloc section…

Removing Randomized Base Addresses

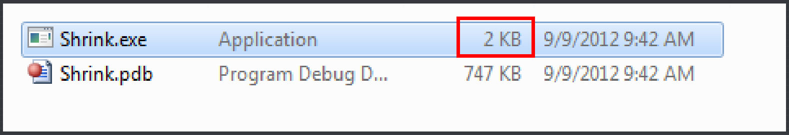
We don't need a relocations section if we never relocate code, so let's turn random relocations off:



Common Properties
Configuration Properties
General
Debugging
C/C++
Linker
General
Input
Manifest File
Debugging
System
Optimization
Embedded IDL
Advanced
Command Line

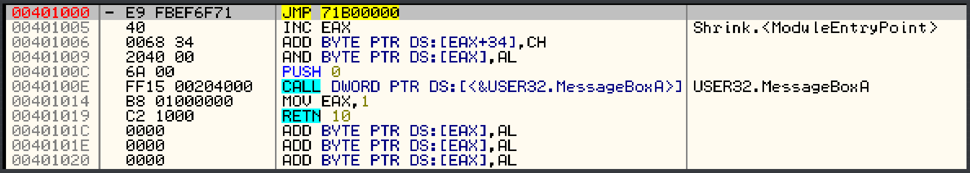
Entry Point	**WinMain**
No Entry Point	No
Set Checksum	No
Base Address	
Randomized Base Address	**Disable Image Randomization (/DYNAMICBASE:**
Fixed Base Address	Default
Data Execution Prevention (DEP)	Image is compatible with DEP (/NXCOMPAT)
Turn Off Assembly Generation	No
Delay Loaded DLL	Don't Support Unload
Import Library	
Merge Sections	
Target Machine	**MachineX86 (/MACHINE:X86)**
Profile	No
CLR Thread Attribute	No threading attribute set

Doing that and rebuilding automatically removes our .reloc section, shaving off another 1,000 bytes:



Shrink.exe	Application	2 KB	9/9/2012 9:42 AM
Shrink.pdb	Program Debug D...	747 KB	9/9/2012 9:42 AM

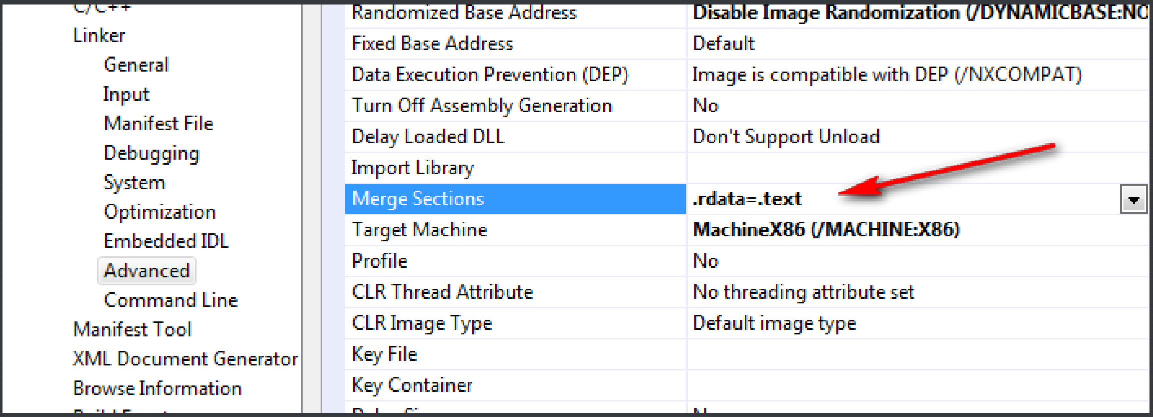
This also has the nice quality of loading our binary in at the usual address of 401000:



```
00401000  - E9 FBEF6F71      JMP 71B00000
00401005    40               INC EAX                                 Shrink.<ModuleEntryPoint>
00401006    0068 34          ADD BYTE PTR DS:[EAX+34],CH
00401009    2040 00          AND BYTE PTR DS:[EAX],AL
0040100C    6A 00            PUSH 0
0040100E    FF15 00204000    CALL DWORD PTR DS:[<&USER32.MessageBoxA>]  USER32.MessageBoxA
00401014    B8 01000000      MOV EAX,1
00401019    C2 1000          RETN 10
0040101C    0000             ADD BYTE PTR DS:[EAX],AL
0040101E    0000             ADD BYTE PTR DS:[EAX],AL
00401020    0000             ADD BYTE PTR DS:[EAX],AL
```

Combining Sections

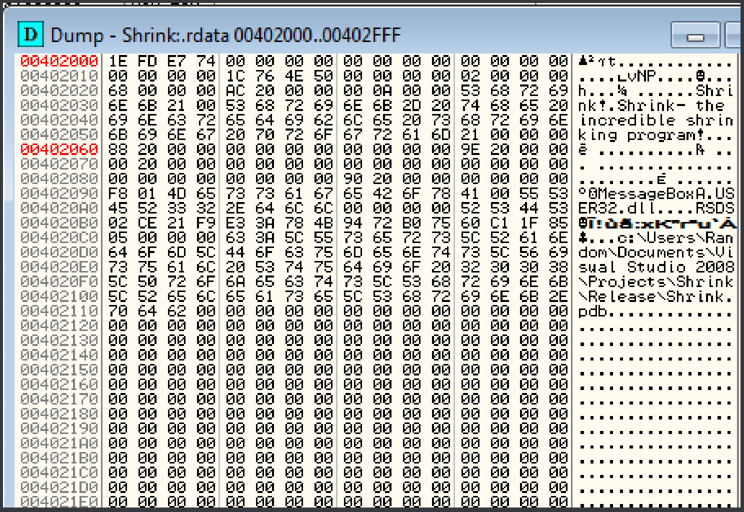
Next, we don't necessarily need both sections, especially since the second section only needs a couple dozen bytes but takes up 1,000. Here, we can set the .rdata section to share the .text section by merging them. Still in the Advanced tab, enter this for "Merge Sections":



- C/C++
- Linker
  - General
  - Input
  - Manifest File
  - Debugging
  - System
  - Optimization
  - Embedded IDL
  - Advanced
  - Command Line
- Manifest Tool
- XML Document Generator
- Browse Information

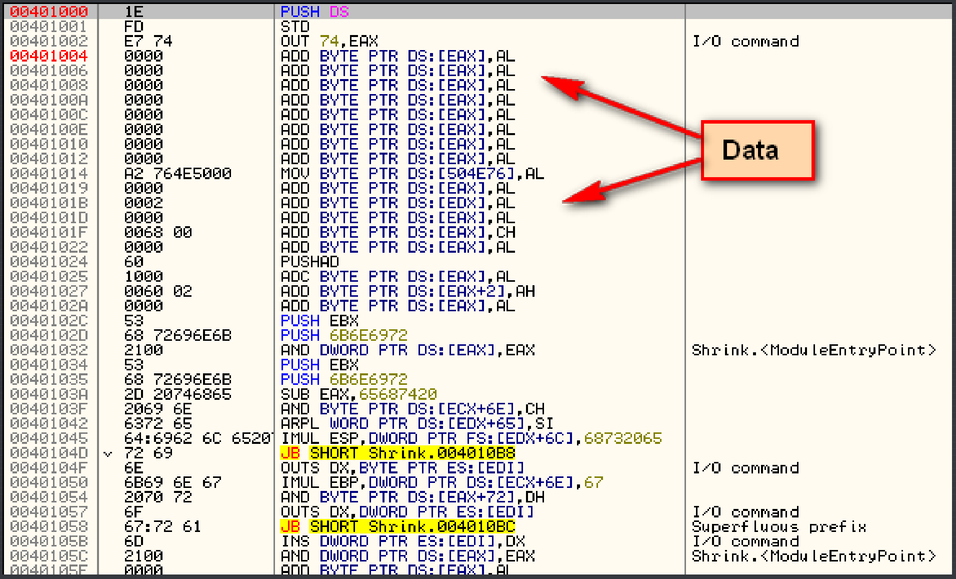
Randomized Base Address	Disable Image Randomization (/DYNAMICBASE:NO
Fixed Base Address	Default
Data Execution Prevention (DEP)	Image is compatible with DEP (/NXCOMPAT)
Turn Off Assembly Generation	No
Delay Loaded DLL	Don't Support Unload
Import Library	
Merge Sections	.rdata=.text
Target Machine	MachineX86 (/MACHINE:X86)
Profile	No
CLR Thread Attribute	No threading attribute set
CLR Image Type	Default image type
Key File	
Key Container	

Here is our original .rdata section:



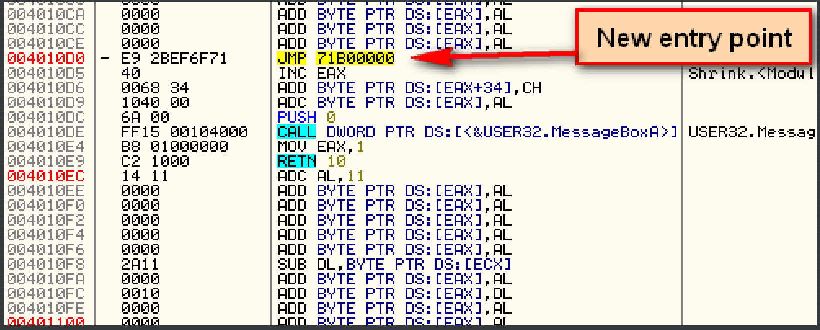
```
Dump - Shrink:.rdata 00402000..00402FFF
00402000 1E FD E7 74 00 00 00 00 00 00 00 00 00 00 00 00 ▲²γt............
00402010 00 00 00 00 1C 76 4E 50 00 00 00 00 02 00 00 00 ....∟vNP....☻...
00402020 68 00 00 00 AC 20 00 00 00 0A 00 00 53 68 72 69 h...¼ ......Shri
00402030 6E 6B 21 00 53 68 72 69 6E 6B 2D 20 74 68 65 20 nk!.Shrink- the 
00402040 69 6E 63 72 65 64 69 62 6C 65 20 73 68 72 69 6E incredible shrin
00402050 6B 69 6E 67 20 70 72 6F 67 72 61 6D 21 00 00 00 king program!...
00402060 88 20 00 00 00 00 00 00 00 00 00 00 9E 20 00 00 ê ..........₧ ..
00402070 00 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 . ..............
00402080 00 00 00 00 00 00 00 00 90 20 00 00 00 00 00 00 ........É ......
00402090 F8 01 4D 65 73 73 61 67 65 42 6F 78 41 00 55 53 °☺MessageBoxA.US
004020A0 45 52 33 32 2E 64 6C 6C 00 00 00 00 52 53 44 53 ER32.dll....RSDS
004020B0 02 CE 21 F9 E3 3A 78 4B 94 72 B0 75 60 C1 1F 85 ☻╬!∙π:xKör░u`┴▼à
004020C0 05 00 00 00 63 3A 5C 55 73 65 72 73 5C 52 61 6E ♣...c:\Users\Ran
004020D0 64 6F 6D 5C 44 6F 63 75 6D 65 6E 74 73 5C 56 69 dom\Documents\Vi
004020E0 73 75 61 6C 20 53 74 75 64 69 6F 20 32 30 30 38 sual Studio 2008
004020F0 5C 50 72 6F 6A 65 63 74 73 5C 53 68 72 69 6E 6B \Projects\Shrink
00402100 5C 52 65 6C 65 61 73 65 5C 53 68 72 69 6E 6B 2E \Release\Shrink.
00402110 70 64 62 00 00 00 00 00 00 00 00 00 00 00 00 00 pdb.............
00402120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00402130 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00402140 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00402150 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00402160 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00402170 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00402180 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00402190 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
004021A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
004021B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
004021C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
004021D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
004021E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
```

and after combining sections, we can see that this data was inserted into the beginning of our .text section in the binary:



```
00401000    1E              PUSH DS
00401001    FD              STD
00401002    E7 74           OUT 74,EAX                                I/O command
00401004    0000            ADD BYTE PTR DS:[EAX],AL
00401006    0000            ADD BYTE PTR DS:[EAX],AL
00401008    0000            ADD BYTE PTR DS:[EAX],AL
0040100A    0000            ADD BYTE PTR DS:[EAX],AL
0040100C    0000            ADD BYTE PTR DS:[EAX],AL
0040100E    0000            ADD BYTE PTR DS:[EAX],AL
00401010    0000            ADD BYTE PTR DS:[EAX],AL
00401012    0000            ADD BYTE PTR DS:[EAX],AL
00401014    A2 764E5000     MOV BYTE PTR DS:[504E76],AL
00401019    0000            ADD BYTE PTR DS:[EAX],AL
0040101B    0002            ADD BYTE PTR DS:[EDX],AL
0040101D    0000            ADD BYTE PTR DS:[EAX],AL
0040101F    0068 00         ADD BYTE PTR DS:[EAX],CH
00401022    0000            ADD BYTE PTR DS:[EAX],AL
00401024    60              PUSHAD
00401025    1000            ADC BYTE PTR DS:[EAX],AL
00401027    0060 02         ADD BYTE PTR DS:[EAX+2],AH
0040102A    0000            ADD BYTE PTR DS:[EAX],AL
0040102C    53              PUSH EBX
0040102D    68 72696E6B     PUSH 6B6E6972
00401032    2100            AND DWORD PTR DS:[EAX],EAX                Shrink.<ModuleEntryPoint>
00401034    53              PUSH EBX
00401035    68 72696E6B     PUSH 6B6E6972
0040103A    2D 20746865     SUB EAX,65687420
0040103F    2069 6E         AND BYTE PTR DS:[ECX+6E],CH
00401042    6372 65         ARPL WORD PTR DS:[EDX+65],SI
00401045    64:6962 6C 6520 IMUL ESP,DWORD PTR FS:[EDX+6C],68732065
0040104D  v 72 69           JB SHORT Shrink.004010B8
0040104F    6E              OUTS DX,BYTE PTR ES:[EDI]                 I/O command
00401050    6B69 6E 67      IMUL EBP,DWORD PTR DS:[ECX+6E],67
00401054    2070 72         AND BYTE PTR DS:[EAX+72],DH
00401057    6F              OUTS DX,DWORD PTR ES:[EDI]                I/O command
00401058    67:72 61        JB SHORT Shrink.004010BC                  Superfluous prefix
0040105B    6D              INS DWORD PTR ES:[EDI],DX                 I/O command
0040105C    2100            AND DWORD PTR DS:[EAX],EAX                Shrink.<ModuleEntryPoint>
0040105E    0000            ADD BYTE PTR DS:[EAX],AL
```

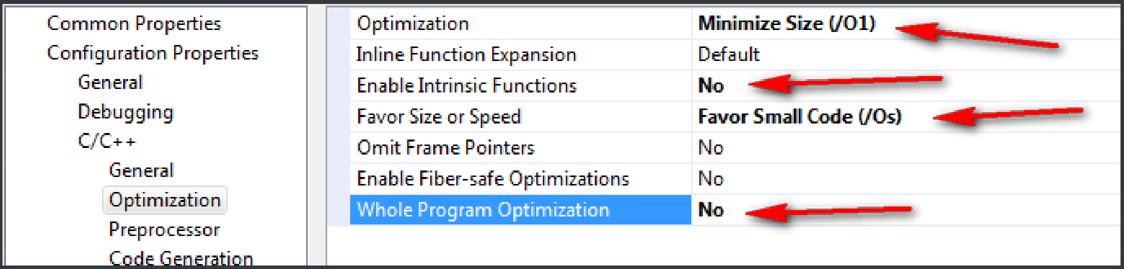
and that our entry point has been changed to 4010D0:



```
004010CA    0000            ADD BYTE PTR DS:[EAX],AL
004010CC    0000            ADD BYTE PTR DS:[EAX],AL
004010CE    0000            ADD BYTE PTR DS:[EAX],AL
004010D0  - E9 2BEF6F71     JMP 71B00000                               Shrink.<Modul
004010D5    40              INC EAX
004010D6    0068 34         ADD BYTE PTR DS:[EAX+34],CH
004010D9    1040 00         ADC BYTE PTR DS:[EAX],AL
004010DC    6A 00           PUSH 0
004010DE    FF15 00104000   CALL DWORD PTR DS:[<&USER32.MessageBoxA>]  USER32.Messag
004010E4    B8 01000000     MOV EAX,1
004010E9    C2 1000         RETN 10
004010EC    14 11           ADC AL,11
004010EE    0000            ADD BYTE PTR DS:[EAX],AL
004010F0    0000            ADD BYTE PTR DS:[EAX],AL
004010F2    0000            ADD BYTE PTR DS:[EAX],AL
004010F4    0000            ADD BYTE PTR DS:[EAX],AL
004010F6    0000            ADD BYTE PTR DS:[EAX],AL
004010F8    2A11            SUB DL,BYTE PTR DS:[ECX]
004010FA    0000            ADD BYTE PTR DS:[EAX],AL
004010FC    0010            ADD BYTE PTR DS:[EAX],DL
004010FE    0000            ADD BYTE PTR DS:[EAX],AL
```

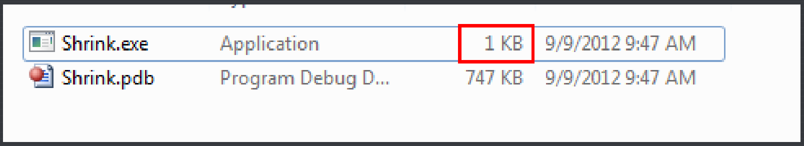
Changing Optimizations

Lastly, we can change the optimizations that Visual Studio uses, telling it to optimize for size over speed. Under C/C++, in the Optimization tab, change these four settings:



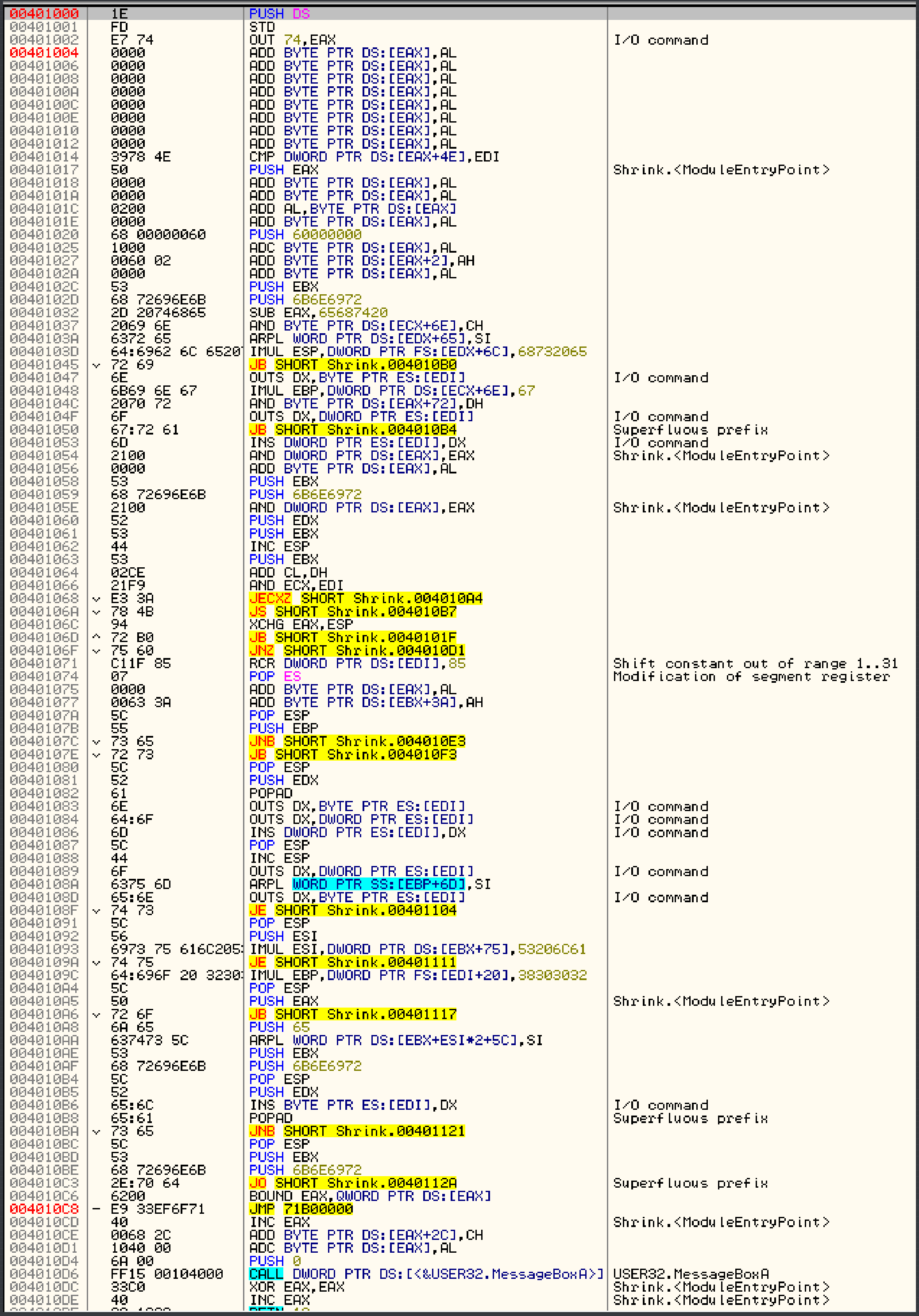
Optimization	**Minimize Size (/O1)**
Inline Function Expansion	Default
Enable Intrinsic Functions	**No**
Favor Size or Speed	**Favor Small Code (/Os)**
Omit Frame Pointers	No
Enable Fiber-safe Optimizations	No
Whole Program Optimization	**No**

One last look at our file size and we see we've done quite a nice job:



Shrink.exe	Application	1 KB	9/9/2012 9:47 AM
Shrink.pdb	Program Debug D...	747 KB	9/9/2012 9:47 AM

And this is the complete disassembly in Olly (the RETN instruction is a little cut off at the bottom):



```
00401000     1E               PUSH DS
00401001     FD               STD
00401002     E7 74            OUT 74,EAX                                  I/O command
00401004     0000             ADD BYTE PTR DS:[EAX],AL
00401006     0000             ADD BYTE PTR DS:[EAX],AL
00401008     0000             ADD BYTE PTR DS:[EAX],AL
0040100A     0000             ADD BYTE PTR DS:[EAX],AL
0040100C     0000             ADD BYTE PTR DS:[EAX],AL
0040100E     0000             ADD BYTE PTR DS:[EAX],AL
00401010     0000             ADD BYTE PTR DS:[EAX],AL
00401012     0000             ADD BYTE PTR DS:[EAX],AL
00401014     3978 4E          CMP DWORD PTR DS:[EAX+4E],EDI
00401017     50               PUSH EAX                                    Shrink.<ModuleEntryPoint>
00401018     0000             ADD BYTE PTR DS:[EAX],AL
0040101A     0000             ADD BYTE PTR DS:[EAX],AL
0040101C     0200             ADD AL,BYTE PTR DS:[EAX]
0040101E     0000             ADD BYTE PTR DS:[EAX],AL
00401020     68 00000060      PUSH 60000000
00401025     1000             ADC BYTE PTR DS:[EAX],AL
00401027     0060 02          ADD BYTE PTR DS:[EAX+2],AH
0040102A     0000             ADD BYTE PTR DS:[EAX],AL
0040102C     53               PUSH EBX
0040102D     68 72696E6B      PUSH 6B6E6972
00401032     2D 20746865      SUB EAX,65687420
00401037     2069 6E          AND BYTE PTR DS:[ECX+6E],CH
0040103A     6372 65          ARPL WORD PTR DS:[EDX+65],SI
0040103D     64:6962 6C 6520  IMUL ESP,DWORD PTR FS:[EDX+6C],68732065
00401045  v  72 69            JB SHORT Shrink.004010B0
00401047     6E               OUTS DX,BYTE PTR ES:[EDI]                   I/O command
00401048     6B69 6E 67       IMUL EBP,DWORD PTR DS:[ECX+6E],67
0040104C     2070 72          AND BYTE PTR DS:[EAX+72],DH
0040104F     6F               OUTS DX,DWORD PTR ES:[EDI]                  I/O command
00401050     67:72 61         JB SHORT Shrink.004010B4                    Superfluous prefix
00401053     6D               INS DWORD PTR ES:[EDI],DX                   I/O command
00401054     2100             AND DWORD PTR DS:[EAX],EAX                  Shrink.<ModuleEntryPoint>
00401056     0000             ADD BYTE PTR DS:[EAX],AL
00401058     53               PUSH EBX
00401059     68 72696E6B      PUSH 6B6E6972
0040105E     2100             AND DWORD PTR DS:[EAX],EAX                  Shrink.<ModuleEntryPoint>
00401060     52               PUSH EDX
00401061     53               PUSH EBX
00401062     44               INC ESP
00401063     53               PUSH EBX
00401064     02CE             ADD CL,DH
00401066     21F9             AND ECX,EDI
00401068  v  E3 3A            JECXZ SHORT Shrink.004010A4
0040106A  v  78 4B            JS SHORT Shrink.004010B7
0040106C     94               XCHG EAX,ESP
0040106D  ^  72 B0            JB SHORT Shrink.0040101F
0040106F  v  75 60            JNZ SHORT Shrink.004010D1
00401071     C11F 85          RCR DWORD PTR DS:[EDI],85                   Shift constant out of range 1..31
00401074     07               POP ES                                      Modification of segment register
00401075     0000             ADD BYTE PTR DS:[EAX],AL
00401077     0063 3A          ADD BYTE PTR DS:[EBX+3A],AH
0040107A     5C               POP ESP
0040107B     55               PUSH EBP
0040107C  v  73 65            JNB SHORT Shrink.004010E3
0040107E  v  72 73            JB SHORT Shrink.004010F3
00401080     5C               POP ESP
00401081     52               PUSH EDX
00401082     61               POPAD
00401083     6E               OUTS DX,BYTE PTR ES:[EDI]                   I/O command
00401084     64:6F            OUTS DX,DWORD PTR ES:[EDI]                  I/O command
00401086     6D               INS DWORD PTR ES:[EDI],DX                   I/O command
00401087     5C               POP ESP
00401088     44               INC ESP
00401089     6F               OUTS DX,DWORD PTR ES:[EDI]                  I/O command
0040108A     6375 6D          ARPL WORD PTR SS:[EBP+6D],SI
0040108D     65:6E            OUTS DX,BYTE PTR ES:[EDI]                   I/O command
0040108F  v  74 73            JE SHORT Shrink.00401104
00401091     5C               POP ESP
00401092     56               PUSH ESI
00401093     6973 75 616C205  IMUL ESI,DWORD PTR DS:[EBX+75],53206C61
0040109A  v  74 75            JE SHORT Shrink.00401111
0040109C     64:696F 20 3230  IMUL EBP,DWORD PTR FS:[EDI+20],38303032
004010A4     5C               POP ESP
004010A5     50               PUSH EAX                                    Shrink.<ModuleEntryPoint>
004010A6  v  72 6F            JB SHORT Shrink.00401117
004010A8     6A 65            PUSH 65
004010AA     637473 5C        ARPL WORD PTR DS:[EBX+ESI*2+5C],SI
004010AE     53               PUSH EBX
004010AF     68 72696E6B      PUSH 6B6E6972
004010B4     5C               POP ESP
004010B5     52               PUSH EDX
004010B6     65:6C            INS BYTE PTR ES:[EDI],DX                    I/O command
004010B8     65:61            POPAD                                       Superfluous prefix
004010BA  v  73 65            JNB SHORT Shrink.00401121
004010BC     5C               POP ESP
004010BD     53               PUSH EBX
004010BE     68 72696E6B      PUSH 6B6E6972
004010C3     2E:70 64         JO SHORT Shrink.0040112A                    Superfluous prefix
004010C6     6200             BOUND EAX,QWORD PTR DS:[EAX]
004010C8  -  E9 33EF6F71      JMP 71B00000
004010CD     40               INC EAX                                     Shrink.<ModuleEntryPoint>
004010CE     0068 2C          ADD BYTE PTR DS:[EAX+2C],CH
004010D1     1040 00          ADC BYTE PTR DS:[EAX],AL
004010D4     6A 00            PUSH 0
004010D6     FF15 00104000    CALL DWORD PTR DS:[<&USER32.MessageBoxA>]   USER32.MessageBoxA
004010DC     33C0             XOR EAX,EAX                                 Shrink.<ModuleEntryPoint>
004010DE     40               INC EAX                                     Shrink.<ModuleEntryPoint>
```

From 31,000 bytes to less than 1,000 (620 bytes to be exact). I guess the real question we should be asking is “Why didn’t Microsoft just start here and then add things as we need them?” I’m sure they’re crying all the way to the bank.

R4ndom