

Re: why I can not write to the file after initialize the MFC in a service program

## Re: why I can not write to the file after initialize the MFC in a service program

---

*Source:* <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.mfc/2007-02/msg00689.html>

---

- *From:* "zhang" <makefriend8@xxxxxxx>
  - *Date:* Thu, 8 Feb 2007 11:14:48 +0800
- 

Problem still exist

"Joseph M. Newcomer" <newcomer@xxxxxxxxxxxx>  
?????:cnqjs2lma6ce9ud757u3n6sq7tnn6sl7s6@xxxxxxxxxxx

I have no idea what "cannot work" means. Do you get an error? This is a SERVICE!!!! This means that

- (a) you don't use strcpy, strcat, or other unsafe mechanisms
  - (b) you don't use char, an obsolete data type
  - (c) you check EVERY return from a call that can fail, and any file system call can fail,
- and when logging errors you report EVERY fact you can. [Take a look at my ErrorString function described in my article on FormatMessage]

When debugging, and reporting a bug, never use meaningless terms like "cannot work". Use descriptions like

"It failed to write, and the GetLastError code was xxxxxxxxx after the <call that failed> in the line indicated below <show which line failed>"

As far as I can tell,

Why do you need an intermediate buffer to write literal strings anyway?

```
WriteToError(_T("Place 1 OK"));
```

works fine and doesn't need some strange and pointless intermediate buffer, particularly not one of fixed size! Assume that fixed-size arrays are evil anyway.

Re: why I can not write to the file after initialize the MFC in a service program

Re: why I can not write to the file after initialize the MFC in a service program

```
You could have
done
LPCTSTR buffer = _T("Place 1 OK");
WriteToError(buffer);
```

which is only slightly silly, but the use of strcpy to a fixed buffer screams "I want to fail horrendously! PLEASE LET MY CODE FAIL! PLEASE!"

For example, if AfxWinInit fails, you copy a 45-character string into a 30-character buffer. Your wish has been granted. Your code failed, because you used what are now thought of as irresponsible programming techniques. Had you written WriteToError(\_T("Fatal Error: MFC initialization failed")); it would have worked, but no, you had to do a gratuitous copy to a buffer. Why? There is no need for the variable 'buf2' to exist; it certainly does not need to be an array of fixed size, and there is certainly no need to copy to it to log a literal string.

joe

On Wed, 7 Feb 2007 16:54:06 +0800, "zhang" <makefriend8@xxxxxxx> wrote:

```
service can run.
void ServiceMain(int argc, char** argv)
{
.....
while (ServiceStatus.dwCurrentState ==
SERVICE_RUNNING)
{
```

\*\*\*\*

ServiceMain should not have a loop in it like this.

\*\*\*\*

```
char buf2[30]="place 1 is ok";
```

\*\*\*\*\*

There are so many things wrong with the above line it is mind-boggling that it could have ever been written

- (a) the variable does not need to exist
- (b) if it does exist, why is there a fixed bound  
const char buf2[] = "place 1 is ok";  
but that is about as silly as having the variable
- (c) why is it not const?
- (d) why is it so much bigger than the string?

Re: why I can not write to the file after initialize the MFC in a service program

\*\*\*\*\*

```
WriteToError(buf2);//It's a function write to a file. and here
is ok!!!!!!
////////////////////////////////////
if (!AfxWinInit(::GetModuleHandle(NULL), NULL,
::GetCommandLine(), 0))
```

\*\*\*\*

So you are going to try to initialize MFC EACH TIME THROUGH THE LOOP?  
WHY?

\*\*\*\*

```
{
strcpy(buffer,"Fatal Error: MFC initialization failed");
WriteToError(buffer);//they do not work
nRetCode = 1;
}
else
{
// TODO: code your application's behavior here.
InitDatabase();
```

\*\*\*\*

AND, you are going to initialize your databases each time through the  
loop, too! WHY?

\*\*\*\*

```
strcpy(buffer,"Init OK");
WriteToError(buffer);//they do not work either
```

\*\*\*\*

```
WriteToError(_T("Init OK"));
what good does an intermediate buffer do here, other than introduce the
possibility of
total disaster?
```

\*\*\*\*

```
.....////
}
.....
strcpy(buffer,"why can not write to file??");
WriteToError(buffer);//can not work
```

\*\*\*\*

Why a strcpy? Silly.

\*\*\*\*

Service can start ,can stop.

Re: why I can not write to the file after initialize the MFC in a service program

```
int WriteToError(char* str)
```

\*\*\*\*

Since the string is not modified, the correct declaration should be

```
BOOL WriteToError(LPCTSTR str)
```

see below about the BOOL. The parameter should be const. Never use a non-const parameter

when a const parameter would work. At the very least, assuming you want a silly int value

returned, and insist on using obsolete concepts like char, you should have written

```
int WriteToError(const char * str)
```

\*\*\*\*

```
{  
    log = fopen("C:\\MyServices\\error.txt", "a+");
```

\*\*\*\*

Why do you presume your service is in this directory? Hardwiring values like this into a

service can be disastrous. A service should not assume that the directory exists in such

a case; if it has a specific place to write to, it should try to create the directory.

Presuming that the C: drive is the only drive also makes no sense. In a multiboot system

it should probably be written to a directory in the drive where the system booted. Look

at GetFolderPath to see about useful and interesting places to find things.

\*\*\*\*

```
    if (log == NULL)  
        return -1;  
    fprintf(log, "%s\n", str);  
    fclose(log);  
    return 0;  
}
```

\*\*\*\*

Given you return only two values, success and failure, why isn't this a BOOL? return

FALSE or TRUE! Why use weird values like -1 (a true value) to indicate failure and 0 (a false value) to indicate success?

Look at strsafe.h if you are working in earlier versions of VS (< VS2005), or use the \_s versions of the functions in VS2005. These are the only string functions you should be

Re: why I can not write to the file after initialize the MFC in a service program

using.

But the whole notion that ServiceMain is in a loop is deeply flawed.

Once-only

initialization should be done outside ANY loop, and it is very rare in a service to put a

loop in ServiceMain; the more typical action is to spin off some number of threads that

implement the service, and have ServiceMain block waiting for a notification that the

service is being shut down.

joe

\*\*\*\*

Joseph M. Newcomer [MVP]

email: [newcomer@xxxxxxxxxxxx](mailto:newcomer@xxxxxxxxxxxx)

Web: <http://www.flounder.com>

MVP Tips: [http://www.flounder.com/mvp\\_tips.htm](http://www.flounder.com/mvp_tips.htm)