

Re: Problems with redirect and format stdout and stderr for CreateProcess

Source: <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.mfc/2007-11/msg00716.html>

- *From:* Joseph M. Newcomer <newcomer@xxxxxxxxxxxxx>
 - *Date:* Wed, 14 Nov 2007 21:23:58 -0500
-

See below...

On Wed, 14 Nov 2007 05:13:28 -0800, luc.pop@xxxxxxxxxx wrote:

Hello,

I'm trying I'm trying to create a filter application that filters the stdout and stderr for a process.

It should work like this:

```
C:>filterOutput.exe "echo hello"
<p><span class="stdout">hello</span></p>
```

In other words it should run the program "echo hello" and concatenate strings in the beginning and end of stdout and stderr. This to facilitate creation of a log.htm file when compiling an application for example `c:> filterOutput.exe "make project" > log.htm`.

I'm using this approach:

Create two unnamed pipes with `CreatePipe`, assign them to `STARTUPINFO` and use `CreateProcess` to create the process. Use a never ending loop with `PeekNamedPipe` and `ReadFile` to extract the data from stdout and stderr of the process. Format the data and use `std::out` to show it.

I get suspicious about never-ending loops and `PeekNamedPipe`. Why doesn't the loop end when the pipe closes? Why do you need to peek?

The problem:

When I run the program from Visual Studio 7.1 it works like a charm. When I run the program from the command line nothing is outputted to the console. I traced the error to the call `PeekNamedPipe`. This does not fail but it always returns that there is no data in the pipe. How is this possible? Is it because the `HANDLE` inheritance rule are wrong?

Re: Problems with redirect and format stdout and stderr for CreateProcess

How can I make this work?

Thank you in advance
/Lucian

Here is the code:

```
// FilterOutput.cpp : Defines the entry point for the console
// application.
//

#include "stdafx.h"
#include <io.h>
#include <string>
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <iostream>
#include <tchar.h>
#include <afxwin.h>
#include <Windows.h>

#pragma hdrstop

#define bzero(a) memset(a,0,sizeof(a)) //easier --- shortcut

bool IsWinNT() //check if we're running NT
{
    OSVERSIONINFO osv;
    osv.dwOSVersionInfoSize = sizeof(osv);
    GetVersionEx(&osv);
    return (osv.dwPlatformId == VER_PLATFORM_WIN32_NT);
}

void ErrorMessage(char *str) //display detailed error info
{
    LPVOID msg;
    FormatMessage(
        FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM,
        NULL,
        GetLastError(),
        MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), // Default language
        (LPTSTR) &msg,
        0,
        NULL
    );
    printf("%s: %s\n",str,msg);
    LocalFree(msg);
}
```

Re: Problems with redirect and format stdout and stderr for CreateProcess

```
std::string& trim2(std::string& str)
{
    std::string::size_type pos = str.find_last_not_of('\n');
    if(pos != std::string::npos) {
        str.erase(pos + 1);
        pos = str.find_first_not_of('\n');
        if(pos != std::string::npos) str.erase(0, pos);
    }
    else str.erase(str.begin(), str.end());
    pos = str.find_last_not_of('\r');
    if(pos != std::string::npos) {
        str.erase(pos + 1);
        pos = str.find_first_not_of('\r');
        if(pos != std::string::npos) str.erase(0, pos);
    }
    else str.erase(str.begin(), str.end());
    return str;
}

HANDLE
newstdin,newstdout,read_stdout,newstderr,read_stderr,write_stdin; //
pipe handles
```

THis should be written as

```
HANDLE newstdin;
HANDLE newstdout;
HANDLE read_stdout;
HANDLE newstderr,
HANDLE read_stderr;
HANDLE write_stdin;
```

the above compressed mess, that doesn't even have whitespace after the commas, is unreadable

```
void close()
{
    CloseHandle(newstdin);
    CloseHandle(newstdout);
    CloseHandle(read_stdout);
    CloseHandle(newstderr);
    CloseHandle(read_stderr);
    CloseHandle(write_stdin);
}
//-----
void main(int argc, char** argv)
{
    char buf[1024]; //i/o buffer
```

Re: Problems with redirect and format stdout and stderr for CreateProcess

```
STARTUPINFO si;
SECURITY_ATTRIBUTES sa;
SECURITY_DESCRIPTOR sd; //security information for
pipes
PROCESS_INFORMATION pi;

if (IsWinNT()) //initialize security descriptor (Windows NT)
{
InitializeSecurityDescriptor(&sd,SECURITY_DESCRIPTOR_REVISION);
SetSecurityDescriptorDacl(&sd, true, NULL, false);
sa.lpSecurityDescriptor = &sd;
}
else sa.lpSecurityDescriptor = NULL;
```

Have you heard of indentation and vertical whitespace? Don't jam things together to make them unreadable!

```
else
sa.lpSecurityDescriptor = NULL;
<blank link here>
*****
```

```
sa.nLength = sizeof(SEcurity_ATTRIBUTES);
sa.bInheritHandle = true; //allow inheritable handles

if (!CreatePipe(&newstdin,&write_stdin,&sa,0)) //create stdin pipe
{
ErrorMessage("CreatePipe");
//getch();
return;
}
if (!CreatePipe(&read_stdout,&newstdout,&sa,0)) //create stdout pipe
{
ErrorMessage("CreatePipe");
//getch();
close();
return;
}
if (!CreatePipe(&read_stderr,&newstderr,&sa,0)) //create stderr pipe
{
ErrorMessage("CreatePipe");
//getch();
close();
return;
}

GetStartupInfo(&si); //set startupinfo for the spawned process
/*
The dwFlags member tells CreateProcess how to make the process.
```

Re: Problems with redirect and format stdout and stderr for CreateProcess

STARTF_USESTDHANDLES validates the hStd* members.

No, it tells the CreateProcess call to USE the standard handles. Validation does not actually happen.

STARTF_USESHOWWINDOW
validates the wShowWindow member.

No, it tells CreateProcess to USE the wShowWindow member; it doesn't validate anything!

```
*/  
si.dwFlags = STARTF_USESTDHANDLES|STARTF_USESHOWWINDOW;  
si.wShowWindow = SW_HIDE;  
si.hStdOutput = newstdout;  
si.hStdError = newstderr; //set the new handles for the child  
process  
si.hStdInput = newstdin;  
char* app_spawn = argv[1]; //sample, modify for your
```

Why are you using obsolete data types like 'char'?

```
//system  
  
//spawn the child process  
printf("hej1\n");  
if (!  
CreateProcess(NULL,app_spawn,NULL,NULL,TRUE,CREATE_NEW_CONSOLE,  
NULL,NULL,&si,&pi)*/
```

Why is this useless piece of text even present? Delete it.

```
!CreateProcess( NULL, // No module name (use command line).  
app_spawn , // Command line.  
NULL, // Process handle not inheritable.  
NULL, // Thread handle not inheritable.  
TRUE, // Set handle inheritance to TRUE.  
CREATE_NEW_CONSOLE,  
NULL, // Use parent's environment block.  
NULL, // Use parent's starting directory.  
&si, // Pointer to STARTUPINFO structure.  
&pi )  
{  
ErrorMessage("CreateProcess");
```

Re: Problems with redirect and format stdout and stderr for CreateProcess

```
close();
return;
}
printf("hej2\n");

unsigned long exit=0; //process exit code
unsigned long bread; //bytes read
unsigned long avail; //bytes available
```

I do not recall any of these being defined as 'unsigned long'. For example the bytes read is DEFINITELY specified as DWORD. Never use hand-expansion of the proper types to something you think they might be.

```
bzero(buf);
```

What possible value is this? You're about to read data into the buffer, there is no reason to zero it out.

```
for(;;) //main program loop
{
  GetExitCodeProcess(pi.hProcess,&exit); //while the process is
  running
  if (exit != STILL_ACTIVE)
  break;
```

This is so wrong. Just wait for the pipe to break. Anything that polls GetExitCodeProcess for STILL_ACTIVE is suspect

```
if(!PeekNamedPipe(read_stdout,buf,1023,&bread,&avail,NULL))
{
  ErrorMessage("PeekNamedPipe stdout");
}
```

Why not just do a ReadFile here? There's no reason to do a PeekNamedPipe.

```
//check to see if there is any data to read from stdout
if (bread != 0)
{
  bzero(buf);
```

You don't need to zero out a buffer you are about to fill.

Re: Problems with redirect and format stdout and stderr for CreateProcess

Note that the ReadFile would work anyway. There's no reason for such complex code here

```
if (avail > 1023)
```

What is this value 1023? Have you never heard of #define? Lose every instance of 1023 and use a named constant in its place!

```
{
while (bread >= 1023)
{
ReadFile(read_stdout,buf,1023,&bread,NULL); //read the stdout
pipe
std::string tmp = trim2(std::string(buf));
```

This doesn't even make sense. You are reading some long string of bytes, perhaps eight or ten lines, and just randomly jamming HTML on each side of it. In fact, according to this code, if you get

Volume in drive C has no label.

Volume Serial Number is 64D1-F51D

Directory of C:\Documents and Settings\email

```
11/11/2007 02:18 AM <DIR> .
11/11/2007 02:18 AM <DIR> ..
10/19/2005 11:13 PM 22 52.zip
12/24/2004 10:27 PM 212,808 a.prn
05/08/2007 09:02 AM 0 abc
05/08/2007 09:01 AM 0 abc#
05/08/2007 09:02 AM 0 abc%
05/08/2007 09:02 AM 0 abc&
06/19/2007 11:40 AM <DIR> Desktop
10/15/2007 12:19 PM <DIR> Favorites
11/02/2007 03:04 PM <DIR> My Documents
11/11/2007 02:20 AM 3,407,872 NTUSER.DAT
11/06/2003 11:47 AM 0 ping
09/30/2005 12:38 PM 19,662,407 ProView 52.zip
10/19/2005 11:13 PM 22 ProvView 52.zip
03/03/2003 07:14 PM <DIR> Start Menu
```

I would expect to get output of the form

```
<p><span class="stdout"> Volume in drive C has no label.\r\n Volume Serial
Number\r\n is 64D1-F51D\r\n\r\n Directory of C:\Documents and
Settings\email\r\n\r\n11/11/2007 02:18 AM <DIR></span></p><p><span
class="stdout">>\r\n\r\n\r\n .\r\n11/11/2007 02:18 AM <DIR> ..\r\n10/19/2005
11:13 PM 22 52.zip\r\n12/24/2004 10:27 PM 212,808
```

Re: Problems with redirect and format stdout and stderr for CreateProcess

```
a.prn\r\n05/08/2007 09:02 AM 0 abc\r\n05/08/2007 09:01 AM 0  
ab\r\n
```

and so on. You could even get a sequence like

```
\r\n
```

since you have done nothing to parse lines or look for linefeeds.

```
std::cout << tmp << std::endl;  
std::cerr << "<p><span class=\"stdout\">" << tmp << "</span></p>"  
<< std::endl;  
bzero(buf);
```

What is this doing? Why do you need to zero out anything at all?

```
printf("hej3\n");  
}  
}  
else {  
ReadFile(read_stdout,buf,1023,&bread,NULL);  
std::string tmp = trim2(std::string(buf));
```

I'm not sure what trim2 is doing, but it looks like you have assumed that ReadFile gives you a single line without the \r\n. It doesn't. It gives you a bunch of bytes, period.

```
std::cout << tmp << std::endl;  
std::cerr << "<p><span class=\"stdout\">" << tmp << "</span></p>"  
<< std::endl;  
printf("hej4\n");  
}  
}/*else{  
printf("not stdout read\n");  
}*/
```

This is all very strange. I'd rewrite this from scratch to use multiple threads and/or asynchronous I/O.

```
if(!PeekNamedPipe(read_stderr,buf,1023,&bread,&avail,NULL))  
{  
ErrorMessage("PeekNamedPipe stderr");  
}  
//check to see if there is any data to read from stderr  
if (bread != 0)
```

Re: Problems with redirect and format stdout and stderr for CreateProcess

```
{
bzero(buf);
if (avail > 1023)
{
while (bread >= 1023)
{
ReadFile(read_stderr,buf,1023,&bread,NULL); //read the stderr
pipe
std::string tmp = trim2(std::string(buf));
std::cout << tmp << std::endl;
std::cerr << "<p><span class=\"stdout\">" << tmp << "</span></p>"
<< std::endl;
bzero(buf);
printf("hej5\n");
}
}
else {
ReadFile(read_stderr,buf,1023,&bread,NULL);
std::string tmp = trim2(std::string(buf));
std::cout << tmp << std::endl;
std::cerr << "<p><span class=\"stdout\">" << tmp << "</span></p>"
<< std::endl;
printf("hej6\n");
}
}/*else{
printf("not stderr read\n");
}*/
if (kbhit()) //check for user input.
{
bzero(buf);
*buf = (char)getche();
//printf("%c",*buf);
WriteFile(write_stdin,buf,1,&bread,NULL); //send it to stdin
if (*buf == '\r') {
*buf = '\n';
printf("%c",*buf);
WriteFile(write_stdin,buf,1,&bread,NULL); //send an extra newline
char,
//if necessary
}
}
}
close();
}
```

Joseph M. Newcomer [MVP]

email: newcomer@xxxxxxxxxxxx

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

MVP Tips: http://www.flounder.com/mvp_tips.htm