What is the maximum size of a process environment block?

devblogs.microsoft.com/oldnewthing/20230404-00

April 4, 2023

Raymond Chen

A customer was getting this error from their Web server:

Server Error in '/' Application.

The environment block used to start a process cannot be longer than 65535 bytes. Your environment block is 70009 bytes long. Remove some environment variables and try again.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Details: System.InvalidOperationException: The environment block used to start a process cannot be longer than 65535 bytes. Your environment block is 70009 bytes long. Remove some environment variables and try again.

Windows imposes no hard-coded limit on the size of a Unicode environment block. Here's a test program that shows that a 256KB environment block is no problem:

```
#include <windows.h>
#include <stdio.h>

int main()
{
    auto n = 65535*2;
    auto p = new WCHAR[n];
    for (int i = 0; i < n; i++) {
        p[i] = L'x';
    }
    p[1] = L'=';
    p[n-1] = 0;
    p[n-2] = 0;
    STARTUPINFO si = { sizeof(si) };
    PROCESS_INFORMATION pi;
    wchar_t cmdline[] = L"notepad.exe";
    printf("CreateProcess result is %d\n",
        CreateProcessW(nullptr, cmdline, nullptr, nullptr, false,
        CREATE_UNICODE_ENVIRONMENT, p, nullptr, &si, &pi));
    return 0;
}
```

This program creates an environment block that holds only one variable, named `x` whose value is 131,067 copies of the letter `x`. It's not glamourous, and certainly not useful, but it does show that Windows is fine with it.

This claimed 65535 byte limit must therefore be coming from somewhere else. After some investigation, the customer found it:

```
namespace System.Diagnostics
{
    /* ... */

    internal static class EnvironmentBlock {
        public static byte[] ToByteArray(StringDictionary sd, bool unicode) {
            /* ... */

            if (unicode) {
                envBlock = Encoding.Unicode.GetBytes(stringBuff.ToString());
            } else {
                envBlock = Encoding.Default.GetBytes(stringBuff.ToString());

                if (envBlock.Length > UInt16.MaxValue)
                    throw new InvalidOperationException(
                        SR.GetString(SR.EnvironmentBlockTooLong,
                                     envBlock.Length));
            }

            return envBlock;
        }
    }
}
```

The exception is coming from the C# base class library.

Note that the check is inside the `else` branch, which means that somebody is passing an ANSI environment block to the CreateProcess function, and ANSI environment blocks are documented as supported up to 32,767 characters.

Chasing back to the caller, the character set is selected here:

```
    bool unicode = false;
#if !FEATURE_PAL
    if (ProcessManager.IsNt) {
        creationFlags |= NativeMethods.CREATE_UNICODE_ENVIRONMENT;
        unicode = true;
    }
#endif // !FEATURE_PAL

    byte[] environmentBytes = EnvironmentBlock.ToByteArray(
            startInfo.environmentVariables, unicode);
```

So it appears that if you are compiled as `FEATURE_PAL`, the environment block is limited to 32,767 characters and must be mappable to ANSI characters.