

Forum:[Hack Forums](#)

Thread: [VAMemory edit pointer adress](#)

Jopemeister - May 22, 2017, 12:43 pm

How can i write to the pointer i found using cheatengine?
im using VAMemory and the normal method doesnt work
Code VAM((IntPtr)adress, value)

stefsot - May 22, 2017, 1:08 pm

a pointer points to the address of the object

Code var address = ReadInt32(pointer_address); \\ ReadInt64 for 64bit
VAM(new IntPtr(address), value)

gigajew - May 26, 2017, 10:05 am

If you're reading from the memory of your own application, do [Marshal.ReadIntPtr](#) or, check the architecture and do what above said

```
if(IntPtr.Size * 8 == 32) { //32 bit
    Marshal.ReadInt32(...)
}else { //64 bit
    Marshal.ReadInt64(...)
}
```

Edit: just re-read your post. Make sure you've got the offset right

d4ne - May 29, 2017, 4:22 am

Here's a memory class i've used in my old Call of duty infinite warfare trainer. Edit it for ur usage:

```
Code using System;
using System.Text;
using System.Diagnostics;
using System.Runtime.InteropServices;
```

```
namespace InfiniteTrainerRelease
{
    public class Memory
    {
        // Dll includes

        [DllImport("kernel32.dll")]
        private static extern Int32 WriteProcessMemory(IntPtr Handle, Int64 Address, byte[] buffer, int
Size, int BytesWritten = 0);
        [DllImport("kernel32.dll")]
        private static extern Int32 ReadProcessMemory(IntPtr Handle, Int64 Address, byte[] buffer, int
Size, int BytesRead = 0);

        // Variables
```

```

public static IntPtr pHandle;
static string ExeName = "iw7_ship";

// Process Functions

public static IntPtr GetProcessHandle()
{
    try
    {
        Process[] ProcList = Process.GetProcessesByName(ExeName);
        pHandle = ProcList[0].Handle;

        return pHandle;
    }
    catch
    {
        return IntPtr.Zero;
    }
}

public static long GetBaseAddress(string ProcessName, string ModuleName)
{
    try
    {
        Process[] processes = Process.GetProcessesByName(ProcessName);
        ProcessModuleCollection modules = processes[0].Modules;
        ProcessModule DLLBaseAddress = null;

        foreach (ProcessModule i in modules)
        {
            if (i.ModuleName == ModuleName)
            {
                DLLBaseAddress = i;
            }
        }

        return DLLBaseAddress.BaseAddress.ToInt64();
    }
    catch
    {
        return 0;
    }
}

public static long GetPointerAddress(long Pointer, int[] Offset)
{
    byte[] Buffer = new byte[8];

    ReadProcessMemory(GetProcessHandle(), Pointer, Buffer, Buffer.Length);

    for (int x = 0; x < (Offset.Length - 1); x++)
    {

```

```

        Pointer = BitConverter.ToInt64(Buffer, 0) + Offset[x];
        ReadProcessMemory(GetProcessHandle(), Pointer, Buffer, Buffer.Length);
    }

    Pointer = BitConverter.ToInt64(Buffer, 0) + Offset[Offset.Length - 1];

    return Pointer;
}

// Writing Functions

public static void WriteBytes(Int64 Address, byte[] Bytes)
{
    WriteProcessMemory(GetProcessHandle(), Address, Bytes, Bytes.Length);
}

public static void WriteFloat(Int64 Address, float Value)
{
    WriteProcessMemory(GetProcessHandle(), Address, BitConverter.GetBytes(Value), 4);
}

public static void WriteDouble(Int64 Address, double Value)
{
    WriteProcessMemory(GetProcessHandle(), Address, BitConverter.GetBytes(Value), 8);
}

public static void WriteInteger(Int64 Address, int Value, int size)
{
    WriteProcessMemory(GetProcessHandle(), Address, BitConverter.GetBytes(Value), size);
}

public static void WriteString(Int64 Address, string String)
{
    byte[] Buffer = new ASCIIEncoding().GetBytes(String);
    WriteProcessMemory(GetProcessHandle(), Address, Buffer, Buffer.Length);
}

// Reading Functions

public static byte[] ReadByte(Int64 Address, int Length)
{
    byte[] Buffer = new byte[Length];
    ReadProcessMemory(GetProcessHandle(), Address, Buffer, Length);
    return Buffer;
}

public static float ReadFloat(Int64 Address)
{
    byte[] Buffer = new byte[4]; ;
    ReadProcessMemory(GetProcessHandle(), Address, Buffer, 4);
    return BitConverter.ToSingle(Buffer, 0);
}

```

```

public static double ReadDouble(Int64 Address)
{
    byte[] Buffer = new byte[8];
    ReadProcessMemory(GetProcessHandle(), Address, Buffer, 8);
    return BitConverter.ToDouble(Buffer, 0);
}

public static int ReadInteger(Int64 Address, int Length)
{
    byte[] Buffer = new byte[Length];
    ReadProcessMemory(GetProcessHandle(), Address, Buffer, Length);
    return BitConverter.ToInt32(Buffer, 0);
}

public static string ReadString(Int64 Address, int size)
{
    byte[] Buffer = new byte[size]; ;
    ReadProcessMemory(GetProcessHandle(), Address, Buffer, size);
    return new ASCIIEncoding().GetString(Buffer);
}
}

```

Now you could update values in the following way using the class:

```

Code public void updatePrestige(string exeName, int value)
{
    long baseaddress = GetBaseAddress(exeName, exeName + ".exe");
    long pointer = GetPointerAddress(baseaddress + 0x041153A0, new int[] { 0x6E5 });

    byte[] intBytes = BitConverter.GetBytes(value);

    WriteBytes(pointer, intBytes);
}

```

Hope this helped a little.