Generating different types of timestamps from quite a long way away

devblogs.microsoft.com/oldnewthing/20150713-00

July 13, 2015

Raymond Chen

Today's Little Program does the reverse of what we had last time. It takes a point in time and then generates timestamps in various formats.

```
using System;

class Program
{
 static void TryFormat(string format, Func<long> func)
 {
  try {
   long l = func();
   if ((ulong)l > 0x00000000FFFFFFFF) {
       Console.WriteLine("{0} 0x{1:X16}", format, l);
   } else {
       Console.WriteLine("{0} 0x{1:X08}", format, l);
   }
  } catch (ArgumentException) {
   Console.WriteLine("{0} - invalid", format);
  }
 }
```

Like last time, the `TryFormat` method executes the passed-in function inside a try/catch block. If the function executes successfully, then we print the result. There is a tiny bit of cleverness where we choose the output format depending on the number of bits in the result.

```
 static long DosDateTimeFromDateTime(DateTime value)
 {
  int result = ((value.Year - 1980) << 25) |
               (value.Month << 21) |
               (value.Day << 16) |
               (value.Hour << 11) |
               (value.Minute << 5) |
               (value.Second >> 1);
  return (uint)result;
 }
```

The `DosDateTimeFromDateTime` converts the `DateTime` into a 32-bit date/time stamp in MS-DOS format. This is not quite correct because MS-DOS format date/time stamps are in local time, but we are not converting the incoming `DateTime` to local time. It's up to you to understand what's going on.

```
 public static void Main(string[] args)
 {
  int[] parts = new int[7];
  for (int i = 0; i < 7; i++) {
   parts[i] = args.Length > i ? int.Parse(args[i]) : 0;
  }

  DateTime value = new DateTime(parts[0], parts[1], parts[2],
                                parts[3], parts[4], parts[5],
                                parts[6], DateTimeKind.Utc);

  Console.WriteLine("Timestamp {0} UTC", value);

  TryFormat("Unix time",
    () => value.ToFileTimeUtc() / 10000000 - 11644473600);
  TryFormat("UTC FILETIME",
    () => value.ToFileTimeUtc());
  TryFormat("Binary DateTime",
    () => value.ToBinary());
  TryFormat("MS-DOS Date/Time",
    () => DosDateTimeFromDateTime(value));
  TryFormat("OLE Date/Time",
    () => BitConverter.DoubleToInt64Bits(value.ToOADate()));
 }
}
```

The parameters on the command line are the year, month, day, hour, minute, second, and millisecond; any omitted parameters are taken as zero. We create a UTC `DateTime` from it, and then try to convert that `DateTime` into the other formats.

Raymond Chen

Follow