Converting File Times - 64 bit math using only built-in DOS commands

breazile 22 Feb 2008 4:03 PM 0

What are File Times?

A *File Time* is a 64-bit value that represents the number of 100-nanosecond intervals that have elapsed since 12:00 A.M. January 1, 1601 Coordinated Universal Time (UTC). I won't go into the details in this blog, but NTFS and Active Directory uses these, and you can find more information here: http://msdn2.microsoft.com/en-us/library/ms724290.aspx This blog talks about how you can convert this 64 bit number into a human readable date and time using only built-in DOS commands.

The Challenge

Convert a 64 bit NTTE number (***NT T****ime* ***E****poch*, or a *File Time* number) to a human readable date and time using only DOS commands, no other executables involved. Ok, I cheated a bit, I used regedit.exe to find the local time offset from UTC to show the local time as well. The UTC time calculation is pure DOS. Since the built in DOS math functions cannot handle 64 bit numbers, we have to calculate things manually.

Why? Dude, you have too much free time...

For fun, weeeee. Only a hard-core geek can appreciate this, and no one would want to do this in a real world application, but I used this exercise to reacquaint myself with DOS after a long absence from batch file scripting. I was in a training class and the instructor made it a challenge since he had not found a way to solve this problem. This scenario came up because some companies do not allow unapproved binaries to be executed in their environment, and w32tm was not available. My developer ego was at stake, since I had been a software architect and developer for more than 15 years. How hard could it be, and just how much can you really do using DOS commands?

How did you do it?

Since we can only use 32 bit signed numbers in DOS commands, we will need to break the number into a high order 32 bit number, and low order 32 bit number. We use the high order number to calculate the number of days since 1601, and then add the remainder to the low order number. From there we calculate days, hours, minutes, etc.

We take into account leap years, and start calculating the year. Here are the rules for leap years:

```
if (year % 4 == 0) {
   if (year % 100 == 0) {
      if (year % 400 == 0)
          We have a leap year
      else
          Not a leap year
   }
   else
      We have a leap year
}
else
   Not a leap year
```

Calling the Script

Simply pass the NTTE number as a parameter, and hit enter. Here is an example:

```
C:\>pntte 126036951652030000

UTC Time is 5/25/2000 2:26:05.2030000 AM
Local Time is 5/24/2000 6:26:05.2030000 PM
```

We can use w32tm.exe to check our work:

```
C:\>w32tm /ntte 126036951652030000
145876 02:26:05.2030000 - 5/24/2000 6:26:05 PM (local time)
```

Badda bing, badda boom. Whadda know, the same result.

The Code

Here is the source for the script. The code is commented, so I won't go through it here. If you find yourself modifying this script, please stop what you are doing, and go outside from time to time. You are spending way too much time in front of the computer :)

```
::
:: PNTTE.CMD - Jon Breazile - July 2006
::
:: Converts a Windows FILETIME number (NT Time Epoch) to UTC and local
```

```
:: time using only DOS commands (at least in the case of UTC time)
:: for cases where W32tm may not be available
::

@echo off
setlocal ENABLEEXTENSIONS ENABLEDELAYEDEXPANSION

title NTTE Time Conversion

::  ***  Check for missing parameters or request for help
if "%1"=="" goto error
if "%1"=="/?" goto usage

::echo.
::echo Converting NTTE Time %1 to UTC and Local Time
::echo.


:: Calculate the length of the NTTE parameter
call :StrLen %1
set /a NTTE_LENGTH = %StrLen%

:: NTTE conversion primer
::
:: Since we can only use 32 bit signed numbers in a shell script, we will need to
:: break the number into a high order 32 bit number, and low order 32 bit number.
:: We use the high order number to calculate the number of days since 1601, and
:: then add the remainder to the low order number. From there we calculate days,
:: hours, minutes, etc.
::
:: We take into account leap years, and start calculating the year. Here are the
:: rules for leap years (I'm an old 'C' dev, with it):
::
:: if (year % 4 == 0) {
::    if (year % 100 == 0) {
::       if (year % 400 == 0)
::           We have a leap year
::       else
::           Not a leap year
::    }
::    else
::       We have a leap year
:: }
:: else
::    Not a leap year
::
:: Now we know how to convert NTTE to UTC. Grab the GMT offset from the registry,
:: and adjust the hours and recalculate to figure out UTC to local time.
::
:: Don't bother trying to convert NTTE to centuries, years, etc. It won't work, and
:: you get math errors. The key is to convert it to days, and then convert the days
:: to centuries, years, etc. Fair enough?

:: Split the NTTE number into high and low order numbers. Add the high order part
:: to the low order part if it is small enough.
::

:: High order part of number of ticks in a day (actually 864,000,000,000)
set TICK_DAY=864
:: High order part of number of ticks in an hour (actually 36,000,000,000)
set TICK_HOUR=36000000
:: High order part of number of ticks in a minute (actually 600,000,000)
set TICK_MINUTE=600000
:: Number of ticks in a second
set TICK_SECOND=10000000
:: Number of ticks in a millisecond
set TICK_MILLISEC=10000
:: Number of days in a year
set /a DAYS_IN_YEAR=365
:: Number of days in 4 years
set /a DAYS_IN_4YEARS=%DAYS_IN_YEAR% * 4 + 1
:: Number of days in 100 years
set /a DAYS_IN_100YEARS=%DAYS_IN_4YEARS% * 25 - 1
:: Number of days in 400 years
set /a DAYS_IN_400YEARS=%DAYS_IN_100YEARS% * 4 + 1

set NTTE_TIME=%1

:: Break the NTTE number into high order and low order parts
if %NTTE_LENGTH% GTR 9  (
```

```
    set NTTE_HIGH=%NTTE_TIME:~0,-9%
    set NTTE_LOW=%NTTE_TIME:~-9%
) else (
    set /a NTTE_HIGH=0
    set NTTE_LOW=%NTTE_TIME%
)
::echo high=%NTTE_HIGH%
::echo low=%NTTE_LOW%

:: Calculate days, and store remainder for later processing
if %NTTE_HIGH% GTR %TICK_DAY% (
    set /a UTC_DAYS=%NTTE_HIGH% / %TICK_DAY%
    set /a REMAINDER=%NTTE_HIGH% - !UTC_DAYS! * %TICK_DAY%
) else (
    set /a UTC_DAYS=0
    set /a REMAINDER=%NTTE_HIGH%
)
::echo days=%UTC_DAYS%

:: Go ahead and add in some low order bits to the number
set /a REMAINDER=%REMAINDER%%NTTE_LOW:~0,-3%
::echo remainder=%REMAINDER%

:: Calculate hours, and store remainder for later processing
if %REMAINDER% GTR %TICK_HOUR% (
    set /a UTC_HOURS=%REMAINDER% / %TICK_HOUR%
    set /a REMAINDER=%REMAINDER% - !UTC_HOURS! * %TICK_HOUR%
) else (
    set /a UTC_HOURS=0
)
::echo hours=%UTC_HOURS%
::echo remainder=%REMAINDER%

:: Calculate minutes, and store remainder for later processing
if %REMAINDER% GTR %TICK_MINUTE% (
    set /a UTC_MINUTES=%REMAINDER% / %TICK_MINUTE%
    set /a REMAINDER=%REMAINDER% - !UTC_MINUTES! * %TICK_MINUTE%
) else (
    set /a UTC_MINUTES=0
)
::echo minutes=%UTC_MINUTES%

:: At this point, we need to add in the remaining low order bits
set /a REMAINDER=%REMAINDER%%NTTE_LOW:~-3%
::echo remainder=%REMAINDER%

:: Calculate seconds, and store remainder for later processing
if %REMAINDER% GTR %TICK_SECOND% (
    set /a UTC_SECONDS=%REMAINDER% / %TICK_SECOND%
    set /a REMAINDER=%REMAINDER% - !UTC_SECONDS! * %TICK_SECOND%
) else (
    set /a UTC_SECONDS=0
)
::echo seconds=%UTC_SECONDS%
::echo remainder=%REMAINDER%

:: Calculate milliseconds, and store remainder for later processing
if %REMAINDER% GTR %TICK_MILLISEC% (
    set /a UTC_MILLISEC=%REMAINDER% / %TICK_MILLISEC%
    set /a REMAINDER=%REMAINDER% - !UTC_MILLISEC! * %TICK_MILLISEC%
) else (
    set /a UTC_MILLISEC=0
)
::echo milliseconds=%UTC_MILLISEC%

set /a UTC_NANOSEC=%REMAINDER%
:echo nanoseconds=%UTC_NANOSEC%

:: OK, now we are ready to calculate the years from the days
set /a QUAD_CENTURY=%UTC_DAYS% / %DAYS_IN_400YEARS%
set /a REMAINDER=%UTC_DAYS% - %QUAD_CENTURY% * %DAYS_IN_400YEARS%
::echo QUAD_CENTURY:%QUAD_CENTURY%
::echo REMAINDER:%REMAINDER%

set /a CENTURY=%REMAINDER% / %DAYS_IN_100YEARS%
set /a REMAINDER=%REMAINDER% - %CENTURY% * %DAYS_IN_100YEARS%
::echo CENTURY:%CENTURY%
::echo REMAINDER:%REMAINDER%

set /a QUAD_YEAR=%REMAINDER% / %DAYS_IN_4YEARS%
```

```
set /a REMAINDER=%REMAINDER% - %QUAD_YEAR% * %DAYS_IN_4YEARS%
::echo QUAD_YEAR:%QUAD_YEAR%
::echo REMAINDER:%REMAINDER%

set /a YEARS=%REMAINDER% / %DAYS_IN_YEAR%
set /a REMAINDER=%REMAINDER% - %YEARS% * %DAYS_IN_YEAR%
::echo YEARS:%YEARS%
::echo REMAINDER:%REMAINDER%

set /a UTC_YEAR=1601 + (%QUAD_CENTURY% * 400) + (%CENTURY% * 100) + (%QUAD_YEAR% *
4) + %YEARS%
::echo %UTC_YEAR%

:: See if the year is a leap year, so we can calculate the month and day
set /a LEAP=%UTC_YEAR% %% 4
if %LEAP% == 0 (
    set /a LEAP=%UTC_YEAR% %% 100
    if %LEAP% == 0 (
        set /a LEAP=%UTC_YEAR% %% 400
        if %LEAP% == 0 (
           set /a LEAP=1
        ) else (
           set /a LEAP=0
        )
    ) else (
        set /a LEAP=1
    )
) else (
    set /a LEAP=0
)
::echo leap:%LEAP%

:: Now we are ready to figure out the month, REMAINDER contains the day
:: in the current year. We add 1 to is because the day starts at 1 not 0
set /a UTC_DAY=%REMAINDER% + 1
set /a UTC_MONTH=1
set /a LAST=0
set /a TMP_DAY=0
for %%a in (31,59,90,120,151,181,212,243,273,304,334) do (
   if %%a GTR 31 (set /a TMP_DAY=%%a + %LEAP%) else set /a TMP_DAY=%%a

   if %UTC_DAY% LEQ !TMP_DAY! goto day_calc_exit

   set /a UTC_MONTH=!UTC_MONTH! + 1

   if %%a GTR 31 (set /a LAST=%%a + %LEAP%) else set /a LAST=%%a
)

:day_calc_exit
set /a UTC_DAY=%UTC_DAY%-%LAST%
::echo month:%UTC_MONTH%
::echo day:%UTC_DAY%

:: Extract the GMT offset (in minutes) from the registry, so we can calculate local
time
::
call :get_gmt_offset
set /a LOCAL_OFFSET=%get_gmt_offset%
::echo local offset:%LOCAL_OFFSET%
set /a LOCAL_DAY=%UTC_DAY%
set /a LOCAL_MONTH=%UTC_MONTH%
set /a LOCAL_YEAR=%UTC_YEAR%

:: Calculate local time offset in hours and minutes
set /a OFFSET_HOURS=%LOCAL_OFFSET% / 60
set /a OFFSET_MINUTES=%LOCAL_OFFSET% - %OFFSET_HOURS% * 60
::echo offset %OFFSET_HOURS%:%OFFSET_MINUTES%

:: Adjust the minutes, and roll the hour back or forward if necessary
set /a LOCAL_MINUTES=%UTC_MINUTES%+%OFFSET_MINUTES%
if %LOCAL_MINUTES% LSS 0 (
   set /a OFFSET_HOURS=%OFFSET_HOURS%-1
   set /a LOCAL_MINUTES=60+%LOCAL_MINUTES%
) else (
   if %LOCAL_MINUTES% GTR 59 (
      set /a OFFSET_HOURS=%OFFSET_HOURS%+1
      set /a LOCAL_MINUTES=%LOCAL_MINUTES%-60
   )
)
```

```
:: Adjust the hours, and roll the day back or forward if necessary
set /a LOCAL_HOURS=%UTC_HOURS%+%OFFSET_HOURS%
::echo local hrs:%LOCAL_HOURS%
if %LOCAL_HOURS% LSS 0 (
   set /a LOCAL_DAY=%LOCAL_DAY%-1
   set /a LOCAL_HOURS=24+%LOCAL_HOURS%
) else (
   if %LOCAL_HOURS% GTR 23 (
      set /a LOCAL_DAY=%LOCAL_DAY%+1
      set /a LOCAL_HOURS=%LOCAL_HOURS%-23
   )
)

:: Do a final check on the day to see if we need to roll the month or year
call :fix_day_month %LOCAL_DAY% %LOCAL_MONTH% %LEAP%

echo.
set BANNER="UTC Time is"
call :print_utc_time %UTC_DAY% %UTC_MONTH% %UTC_YEAR% %UTC_HOURS% %UTC_MINUTES%
%UTC_SECONDS% %UTC_MILLISEC% %UTC_NANOSEC% %BANNER%

set BANNER="Local Time is"
call :print_utc_time %LOCAL_DAY% %LOCAL_MONTH% %LOCAL_YEAR% %LOCAL_HOURS%
%LOCAL_MINUTES% %UTC_SECONDS% %UTC_MILLISEC% %UTC_NANOSEC% %BANNER%


goto end


::
********************************************************************************
::  ***
::  *** print_utc_time - Print the specified time in a pretty format
::  ***                  interested in ActiveTimeBias which is the offset (in
minutes)
;:  ***                  from UTC time.
::  ***
::  *** Parameters:      %1 - Day, %2 - Month, %3 - Year, %4 - Hour, %5 - Min,
::  ***                  %6 - Sec, %7 - millisec, %8 - nanosec,
::  ***                  %9 - Text to print before the time (use quotes if spaces
in parameter)
::  *** Return:          None.
::  ***
::
********************************************************************************

:print_utc_time
setlocal
set DAY=%1
set MONTH=%2
set YEAR=%3
set HOURS=%4
set MINUTES=%5
set SECONDS=%6
set MILLISEC=%7
set NANOSEC=%8
if %HOURS% GTR 12 (
    set /a SHOW_HOURS=%HOURS%-12
    set AMPM=PM
) else (
    set /a SHOW_HOURS=%HOURS%
    set AMPM=AM
)

if %MINUTES% LSS 10 (
    set SHOW_MIN=0%MINUTES%
) else (
    set SHOW_MIN=%MINUTES%
)

if %SECONDS% LSS 10 (
    set SHOW_SEC=0%SECONDS%
) else (
    set SHOW_SEC=%SECONDS%
)

if %MILLISEC% LSS 10 (
    set SHOW_MILLISEC=00%MILLISEC%
) else (
    if %MILLISEC% LSS 100 (
```

```
        set SHOW_MILLISEC=0%MILLISEC%
    ) else (
        set SHOW_MILLISEC=%MILLISEC%
    )
)

if %NANOSEC% LSS 10 (
    set SHOW_NANOSEC=000%NANOSEC%
) else (
    if %NANOSEC% LSS 100 (
        set SHOW_NANOSEC=00%NANOSEC%
    ) else (
        if %NANOSEC% LSS 1000 (
            set SHOW_NANOSEC=0%NANOSEC%
        ) else (
            set SHOW_NANOSEC=%NANOSEC%
        )
    )
)

echo %~9 %MONTH%/%DAY%/%YEAR%
%SHOW_HOURS%:%SHOW_MIN%:%SHOW_SEC%.%SHOW_MILLISEC%%SHOW_NANOSEC% %AMPM%
endlocal
goto :EOF


::
********************************************************************************
::  ***
::  *** get_gmt_offset - Extract the local time offset from the registry. We are
::  ***                  interested in ActiveTimeBias which is the offset (in
minutes)
;:  ***                  from UTC time.
::  ***
::  *** Parameters:      None
::  *** Return:          The adjusted value of ActiveTimeBias.
::  ***
::
********************************************************************************

:get_gmt_offset
setlocal
set junk=0x123
regedit /e pntte.tmp
"HKEY_LOCAL_MACHINE\system\currentcontrolset\control\timezoneinformation"
For /F "skip=3 tokens=1-3* delims=:=" %%a in ('type pntte.tmp') do (
 ::  *** Look for ActiveTimeBias which gives us the offset in seconds, make a valid
hex number
 if /i %%a=="ActiveTimeBias" set junk=0x%%c
)
:: Convert hex bias to decimal bias, and fix sign (offset FROM GMT instead of
offset TO GMT)
if exist pntte.tmp del pntte.tmp
set /a get_gmt_offset=0-!junk!
endlocal & set get_gmt_offset=%get_gmt_offset%
goto :EOF


::
********************************************************************************
::  ***
::  *** StrLen -      Calculate the number of characters in a variable
::  ***
::  *** Parameters:   %1 contains the variable which you want the length of
::  *** Return:       Returns the length of the string.
::  ***
::
********************************************************************************

:StrLen
setlocal & set TmpCnt=%*
if not defined TmpCnt (
  set StrLen=0
) else (
:Lenloop
  set TmpCnt=%TmpCnt:~1%
  set /a StrLen +=1
  if defined TmpCnt goto Lenloop
)
```

```
endlocal & set StrLen=%StrLen%
goto :EOF


::
********************************************************************************
::  ***
::  *** fix_day_month - Look at the day and month after rolling time, and fix it.
::  ***
::  *** Parameters:     %1 - day, %2 - month, %3 - leap year flag (1 = leap year)
::  *** Return:         Adjustment to year if necessary.
::  ***
::
********************************************************************************

:fix_day_month
set fix_day_month=0

if %2 == 1 (
   if %1 == 0 (
      set /a LOCAL_DAY=31
      set /a LOCAL_mONTH=12
      set /a fix_day_month=-1
   ) else (
     if %1 GTR 31 (
         set /a LOCAL_DAY=%1 - 31
         set /a LOCAL_mONTH=%LOCAL_mONTH% + 1
      )
   )
   goto fix_day_month_exit
)

if %2 == 2 (
   if %3 == 1 (
      if %1 == 0 (
         set /a LOCAL_DAY=31
         set /a LOCAL_mONTH=1
      ) else (
         if %1 GTR 29 (
            set /a LOCAL_DAY=%1 - 29
            set /a LOCAL_mONTH=%LOCAL_mONTH% + 1
         )
      )
   ) else (
      if %1 == 0 (
         set /a LOCAL_DAY=31
         set /a LOCAL_mONTH=1
      ) else (
         if %1 GTR 28 (
            set /a LOCAL_DAY=%1 - 28
            set /a LOCAL_mONTH=%LOCAL_mONTH% + 1
         )
      )
   )
   goto fix_day_month_exit
)

if %2 == 3 (
   if %3 == 1 (
      if %1 == 0 (
         set /a LOCAL_DAY=29
         set /a LOCAL_mONTH=2
      ) else (
         if %1 GTR 31 (
            set /a LOCAL_DAY=%1 - 31
            set /a LOCAL_mONTH=%LOCAL_mONTH% + 1
         )
      )
   ) else (
      if %1 == 0 (
         set /a LOCAL_DAY=28
         set /a LOCAL_mONTH=2
      ) else (
         if %1 GTR 31 (
            set /a LOCAL_DAY=%1 - 31
            set /a LOCAL_mONTH=%LOCAL_mONTH% + 1
         )
      )
   )
```

```
    goto fix_day_month_exit
)

if %2 == 4 (
    if %1 == 0 (
        set /a LOCAL_DAY=31
        set /a LOCAL_mONTH=3
    ) else (
        if %1 GTR 30 (
            set /a LOCAL_DAY=%1 - 30
            set /a LOCAL_mONTH=%LOCAL_mONTH% + 1
        )
    )
    goto fix_day_month_exit
)

if %2 == 5 (
    if %1 == 0 (
        set /a LOCAL_DAY=30
        set /a LOCAL_mONTH=4
    ) else (
        if %1 GTR 31 (
            set /a LOCAL_DAY=%1 - 31
            set /a LOCAL_mONTH=%LOCAL_mONTH% + 1
        )
    )
    goto fix_day_month_exit
)

if %2 == 6 (
    if %1 == 0 (
        set /a LOCAL_DAY=31
        set /a LOCAL_mONTH=5
    ) else (
        if %1 GTR 30 (
            set /a LOCAL_DAY=%1 - 30
            set /a LOCAL_mONTH=%LOCAL_mONTH% + 1
        )
    )
    goto fix_day_month_exit
)

if %2 == 7 (
    if %1 == 0 (
        set /a LOCAL_DAY=30
        set /a LOCAL_mONTH=6
    ) else (
        if %1 GTR 31 (
            set /a LOCAL_DAY=%1 - 31
            set /a LOCAL_mONTH=%LOCAL_mONTH% + 1
        )
    )
    goto fix_day_month_exit
)

if %2 == 8 (
    if %1 == 0 (
        set /a LOCAL_DAY=31
        set /a LOCAL_mONTH=7
    ) else (
        if %1 GTR 31 (
            set /a LOCAL_DAY=%1 - 31
            set /a LOCAL_mONTH=%LOCAL_mONTH% + 1
        )
    )
    goto fix_day_month_exit
)

if %2 == 9 (
    if %1 == 0 (
        set /a LOCAL_DAY=31
        set /a LOCAL_mONTH=8
    ) else (
        if %1 GTR 30 (
            set /a LOCAL_DAY=%1 - 30
            set /a LOCAL_mONTH=%LOCAL_mONTH% + 1
        )
    )
    goto fix_day_month_exit
```

```
)

if %2 == 10 (
   if %1 == 0 (
      set /a LOCAL_DAY=30
      set /a LOCAL_mONTH=9
   ) else (
      if %1 GTR 31 (
         set /a LOCAL_DAY=%1 - 31
         set /a LOCAL_mONTH=%LOCAL_mONTH% + 1
      )
   )
   goto fix_day_month_exit
)

if %2 == 11 (
   if %1 == 0 (
      set /a LOCAL_DAY=31
      set /a LOCAL_mONTH=10
   ) else (
      if %1 GTR 30 (
         set /a LOCAL_DAY=%1 - 30
         set /a LOCAL_mONTH=%LOCAL_mONTH% + 1
      )
   )
   goto fix_day_month_exit
)

if %2 == 12 (
   if %1 == 0 (
      set /a LOCAL_DAY=30
      set /a LOCAL_mONTH=11
   ) else (
      if %1 GTR 31 (
         set /a LOCAL_DAY=%1 - 31
         set /a LOCAL_mONTH=1
         set /a fix_day_month=1
      )
   )
)

:fix_day_month_exit
set /a LOCAL_YEAR=%LOCAL_YEAR% + %fix_day_month%
::set fix_day_month=%fix_day_month%
goto :EOF


::  *** Error Jmp, no parameters, or bad parameter
:error
echo.
echo ERROR: no NTTE parameter was defined (try %0 /?)
echo.
goto end


::  *** Print usage
:usage
echo.
echo %0 FILETIME
echo.
echo     Where FILETIME is a 64 bit number representing a Windows FILETIME
echo.
echo Example:
echo.
echo     %0 126036951652030000
echo.

::  *** That's all folks, make sure your seat backs are up, and tray tables are put
away...
:end

title Command Prompt

endlocal
```

Comments