



# コンピュータフォレンジクス技術解説

## Timeline Creation and Analysis

- ミクロビューによるタイムスタンプの特性検証 -

2009.12.4

株式会社 Ji2 山崎 輝

# 目次

1. タイムライン関連ツールと取り組み状況の紹介
2. コンピュータにおける時間の取り扱い
3. ミクロビューによるタイムスタンプの特性検証

# タイムライン調査の位置づけ

- タイムライン：歴史上の出来事を起こった年とともに順に並べたもの
- タイムスタンプ：メタデータとして記録される時間情報
- フォレンジック分野では対象の行動および付随する情報の収集、事象追跡に活用
- 従来はファイルシステムが管理するファイルのタイムスタンプをソート、表示

Encase Table View

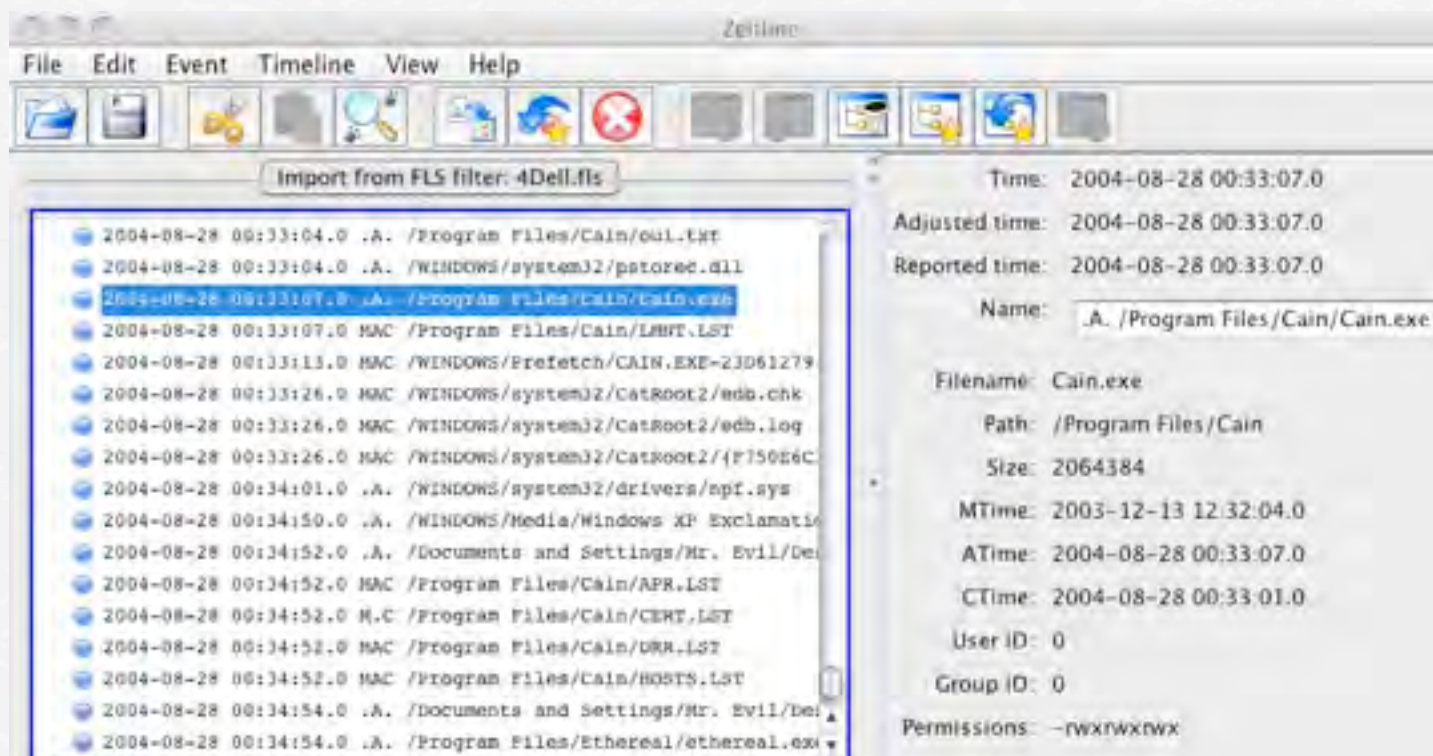
	Name	Last Written	Last Accessed	Entry Modified
<input type="checkbox"/> 11979	oui.txt	2003/11/23 03:23:46	2004/08/26 00:33:04	2004/08/21 00:05:59
<input type="checkbox"/> 11980	Calc.exe	2003/12/13 12:32:04	2004/08/28 00:33:07	2004/08/28 00:33:01
<input type="checkbox"/> 11981	LMNT.LST	2004/08/28 00:33:07	2004/08/28 00:33:07	2004/08/28 00:33:07
<input type="checkbox"/> 11982	CADN.EXE-Z3D61279.pif	2004/08/28 00:33:13	2004/08/26 00:33:13	2004/08/28 00:33:13
<input type="checkbox"/> 11983	edb.log	2004/08/28 00:33:26	2004/08/28 00:33:26	2004/08/28 00:33:26
<input type="checkbox"/> 11984	catdb	2004/08/28 00:33:26	2004/08/28 00:33:26	2004/08/28 00:33:26
<input type="checkbox"/> 11985	edb.chk	2004/08/28 00:33:26	2004/08/28 00:33:26	2004/08/28 00:33:26
<input type="checkbox"/> 11986	npf.sys	2003/06/14 04:06:32	2004/08/28 00:34:01	2004/08/28 00:15:16
<input type="checkbox"/> 11987	Windows XP Exclamation.wav	2001/08/24 03:00:00	2004/08/28 00:34:50	2004/08/20 02:02:14
<input type="checkbox"/> 11988	Can v2.5.lnk	2004/08/21 00:06:01	2004/08/28 00:34:52	2004/08/21 00:11:29
<input type="checkbox"/> 11989	HOSTS.LST	2004/08/28 00:34:52	2004/08/28 00:34:52	2004/08/28 00:34:52
<input type="checkbox"/> 11990	APR.LST	2004/08/28 00:34:52	2004/08/28 00:34:52	2004/08/28 00:34:52
<input type="checkbox"/> 11991	DRR.LST	2004/08/28 00:34:52	2004/08/28 00:34:52	2004/08/28 00:34:52
<input type="checkbox"/> 11992	etherd.exe	2004/08/13 11:15:35	2004/08/26 00:34:54	2004/08/28 00:35:04
<input type="checkbox"/> 11993	Etherd.lnk	2004/08/28 00:29:44	2004/08/28 00:34:54	2004/08/28 00:30:01

Encase Timeline View



## タイムライン調査ツール (1) - Zeitline -

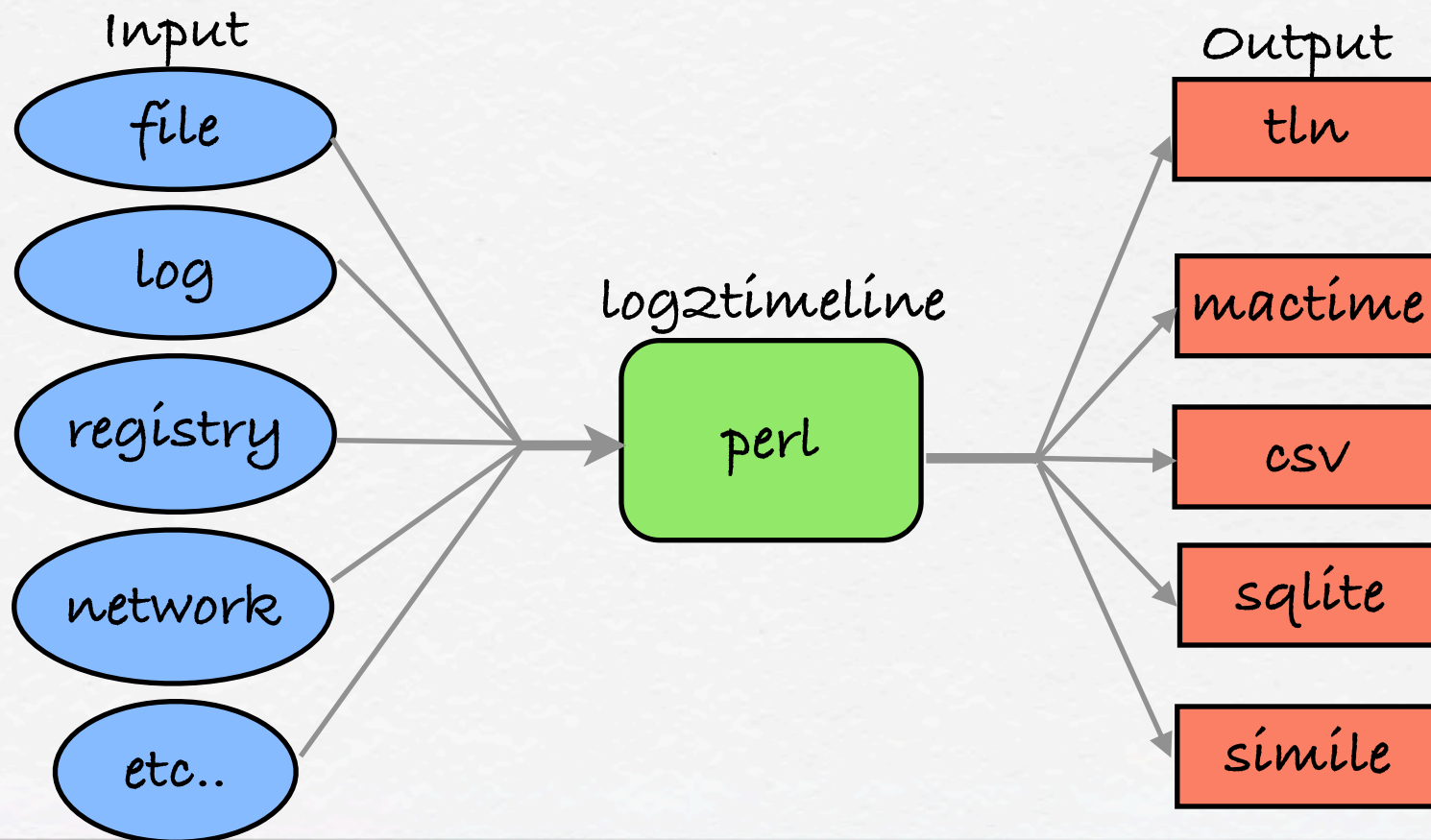
- <http://projects.cerias.purdue.edu/forensics/timeline.php>
- ファイルやログのタイムスタンプをマージして表示



2006年から更新はないが、タイムライン調査関連の論文ではZeitlineをベースに拡張方法が模索、提案されている。

## タイムライン調査ツール (2) - log2timeline -

- <http://log2timeline.net/>
- タイムライン用データを生成するパーサ+コンバータという位置づけ
- タイムラインビューアが扱う形式で出力可能





## タイムライン調査ツール (3) - CFTL -

- CyberForensics TimeLab (<http://cftl.rby.se>)
- 現在一般には公開されていないが、DFRWS 2009で発表された論文 (<http://www.dfrws.org/2009/proceedings/p78-olsson.pdf>) を見る限りでは高性能。

The screenshot displays the CyberForensics TimeLab Viewer interface. The main window shows a timeline view of system events from 2008-01-27 17:36:30 to 2008-01-27 22:08:43. The timeline is divided into several sections: WindowsRegistry, WindowsEventLog, WindowsLHM, FAT, JFD, MDOArchive, and NTFS. A detailed event log is visible at the bottom, listing events such as 'Entry was generated', 'File was created', and 'Key was last changed' with their respective timestamps and source information.

Time	SourceType	EventType	Information
2008-01-27 18:00:52.0	Zero	Entry was generated	ID: 1113194531
2008-01-27 18:00:52.0	Zero	Entry was written	ID: 1113194531
2008-01-27 18:00:52.330	Zero	File was created	\\.\C:\WINDOWS\Prefetch\MSDCOE.D-E-30411802.pf
2008-01-27 18:00:54.0	Zero	File was last modified	\\.\C:\WINDOWS\Prefetch\MSDCOE.D-E-30411802.pf
2008-01-27 18:00:54.0	Zero	File's Master Record was m...	\\.\C:\WINDOWS\Prefetch\MSDCOE.D-E-30411802.pf
2008-01-27 18:01:31.767	Zero	Key was last changed	\\.\C:\WINDOWS\Prefetch\MSDCOE.D-E-30411802.pf
2008-01-27 18:01:31.767	Zero	Key was last changed	\\.\C:\WINDOWS\Prefetch\MSDCOE.D-E-30411802.pf
2008-01-27 18:01:31.986	Zero	Key was last changed	\\.\C:\WINDOWS\Prefetch\MSDCOE.D-E-30411802.pf
2008-01-27 18:01:31.986	Zero	Key was last changed	\\.\C:\WINDOWS\Prefetch\MSDCOE.D-E-30411802.pf
2008-01-27 18:01:31.986	Zero	Key was last changed	\\.\C:\WINDOWS\Prefetch\MSDCOE.D-E-30411802.pf
2008-01-27 18:01:31.986	Zero	Key was last changed	\\.\C:\WINDOWS\Prefetch\MSDCOE.D-E-30411802.pf
2008-01-27 18:01:31.986	Zero	Key was last changed	\\.\C:\WINDOWS\Prefetch\MSDCOE.D-E-30411802.pf
2008-01-27 18:01:31.986	Zero	Key was last changed	\\.\C:\WINDOWS\Prefetch\MSDCOE.D-E-30411802.pf
2008-01-27 18:01:31.986	Zero	Key was last changed	\\.\C:\WINDOWS\Prefetch\MSDCOE.D-E-30411802.pf
2008-01-27 18:01:31.986	Zero	Key was last changed	\\.\C:\WINDOWS\Prefetch\MSDCOE.D-E-30411802.pf
2008-01-27 18:01:31.986	Zero	Key was last changed	\\.\C:\WINDOWS\Prefetch\MSDCOE.D-E-30411802.pf
2008-01-27 18:01:31.986	Zero	Key was last changed	\\.\C:\WINDOWS\Prefetch\MSDCOE.D-E-30411802.pf
2008-01-27 18:01:31.986	Zero	Key was last changed	\\.\C:\WINDOWS\Prefetch\MSDCOE.D-E-30411802.pf

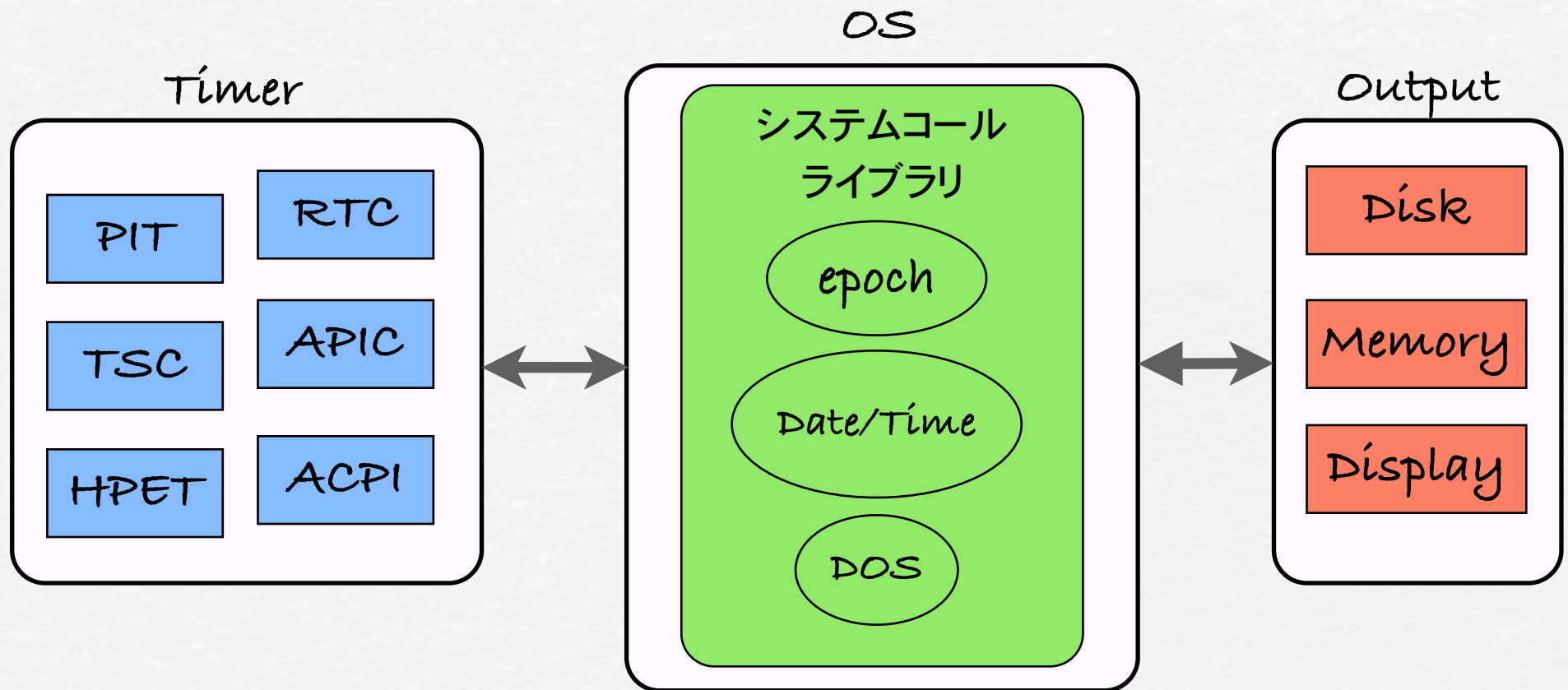
# タイムライン調査関連の主な取り組み状況

- 複数のソース(機器)をマージしたタイムライン調査
  - ソース間の時間ずれを推測して調整する試み
    - ➡ 2003年北米大停電は9秒以内に発生し、各地で発生したイベントからタイムラインを作成するために多大なりソースを要したことが背景
  - 世界中のPCに設定されている時間の正確さ(NTP同期の割合)調査
    - ➡ 74%のホストはUTCから10秒以内の誤差という結果
- ファイルのタイムスタンプの更新とメタ番号の割当ルールの発見、検証
- PCの時間情報をハンドリングする分解能の検証

# 目次

1. タイムライン関連ツールと取り組み状況の紹介
2. コンピュータにおける時間の取り扱い
3. ミクロビューによるタイムスタンプの特性検証

# PCの時間情報のハンドリングイメージ



# コンピュータのハードウェアクロック/タイマ

ハードウェア	内容	カウンタ	周波数
RTC	カレンダー形式で時間情報を保持する 唯一のデバイス	-	32.768kHz
PIT	Intel i8254チップ	16bit	1.193182MHz
APIC	CPU内蔵(タイマ割り込み可)	32bit	CPUクロックベース
TSC	CPU内蔵(タイマ割り込み不可)	64bit	CPUクロックベース
ACPI(PM)	ACPIのタイマ 省電力モードでも正常に動作	24bit	3.57945MHz
HPET	ICH4あたりから搭載 Windows Vista/ Linux Kernel 2.6/Intel Macから対応	32/64bit	10MHz以上

# コンピュータの時刻精度

- 論文(Capturing Timestamp Precision for Digital Forensics) [\*1]の検証結果より
  - 稼働中マシンの時間誤差は線形に増加する（検証では1日毎に実時間より約3974ms遅くなった）。
  - CPUの高負荷は時刻精度に影響を与えない。
  - NTP有効時は時間誤差はほとんど発生しない。
- その他の一般的な特性について
  - RTCは温度に関わらず実時間より遅くなる傾向にある。
  - PICは温度が低いと遅く、温度が高いと速くなる傾向にある。
  - 時刻精度は常時稼働の場合にI/Oのタイマ、電源On/Offが多いマシンはRTCに依存することになる。

[\*1] <http://www.infosec.jmu.edu/reports/jmu-infosec-tr-2009-002.pdf>

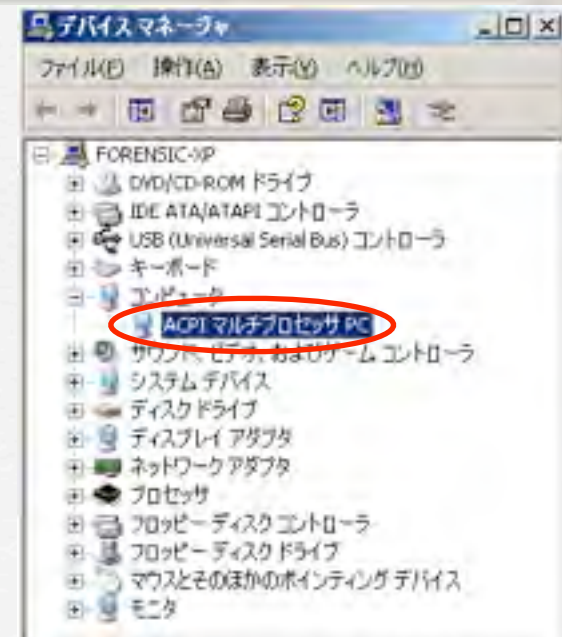
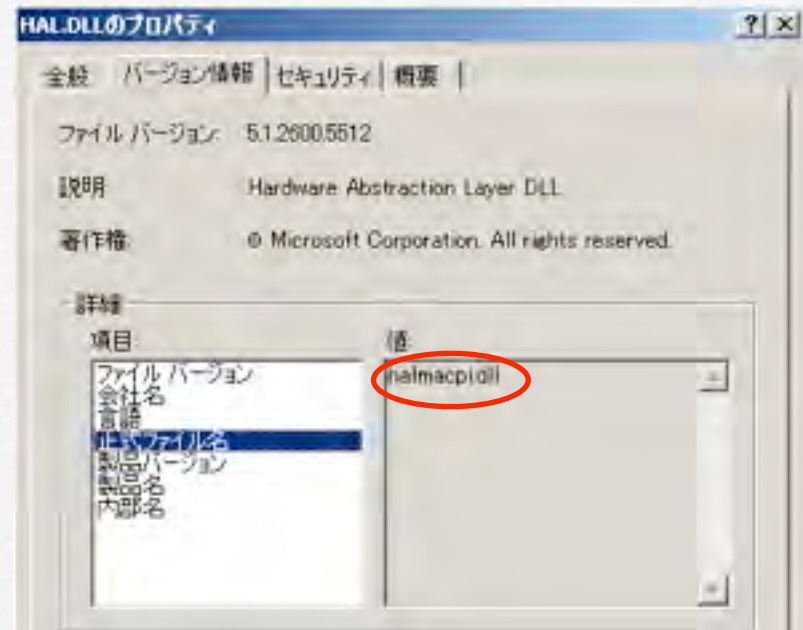
# Windowsのタイマと時刻取得API

## □ タイマの選択肢

- 標準 PC - hal.dll
- MPS ユニプロセッサ PC - halapic.dll
- MPS マルチプロセッサ PC - halmps.dll
- ACPI PC - halacpi.dll
- ACPI ユニプロセッサ PC - halaacpi.dll
- ACPI マルチプロセッサ PC - halmacpi.dll

## □ 時刻取得API - GetSystemAsFileTime()

- 最小単位は100ns
- 実測値は15.625ms前後で、ACPI, MPS非対応のタイマの場合は10.0144ms, 8.38ms前後になる。



# Linuxのタイマと時刻取得API

- 起動時に現在時刻を読み取る際にRTCを参照し、その後利用するタイマが選択される。

- タイマに関する情報の取得

- `/sys/devices/system/clocksource/clocksource0/`

- `available_clocksource` (優先順で記載)

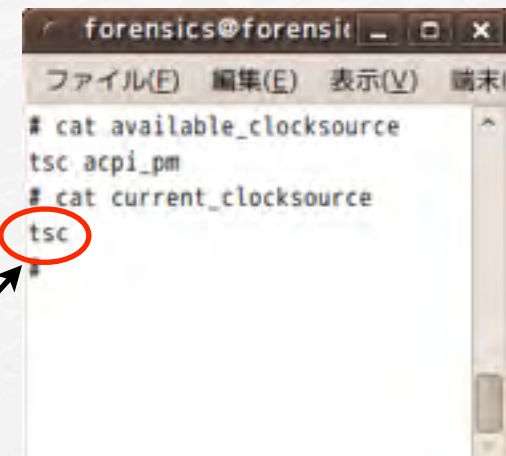
- `current_clocksource`

- `/proc/timer_list` (resolutionがタイマ精度)

- 時刻取得API

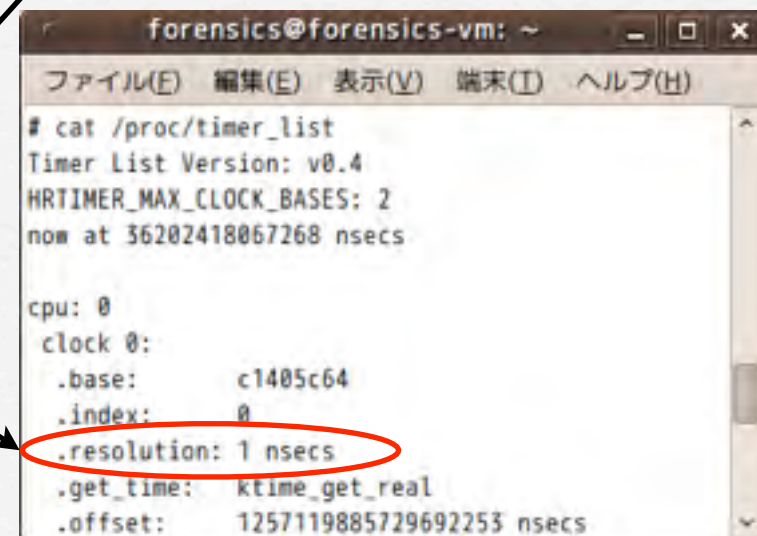
- `gettimeofday()` - 最小単位は $1\mu\text{s}$

- `clock_gettime()` - 最小単位は $1\text{ns}$  (実測は $300\text{ns}$ 程度)



```
forensics@forensic: ~  
# cat available_clocksource  
tsc acpi_pm  
# cat current_clocksource  
tsc
```

A terminal window titled 'forensics@forensic' showing the output of two 'cat' commands. The first command, 'cat available\_clocksource', lists 'tsc' and 'acpi\_pm'. The second command, 'cat current\_clocksource', lists 'tsc'. The 'tsc' output of the second command is circled in red. An arrow points from this circle to the 'current\_clocksource' entry in the list below.



```
forensics@forensics-vm: ~  
# cat /proc/timer_list  
Timer List Version: v0.4  
HRTIMER_MAX_CLOCK_BASES: 2  
now at 36202418067268 nsecs  
  
cpu: 0  
clock 0:  
  .base:      c1405c64  
  .index:     0  
  .resolution: 1 nsecs  
  .get_time:  ktime_get_real  
  .offset:    1257119885729692253 nsecs
```

A terminal window titled 'forensics@forensics-vm' showing the output of 'cat /proc/timer\_list'. The output includes version information, current time in nanoseconds, and details for 'clock 0'. The line '.resolution: 1 nsecs' is circled in red. An arrow points from this circle to the 'resolutionがタイマ精度' text in the list above.

## コンピュータが扱う時間情報の形式 (1) - epoch -

- ある時間を基点とした経過時間で表現。人間にはわかりにくいだが日時の変換処理等コンピュータにとって扱いやすい。通常うるう秒は考慮しない。

種類	基点	最小単位	型 (サイズ[*1])
Unix (POSIX)	1970/01/01 00:00:00	1s	time_t (4byte)
Windows	1601/01/01 00:00:00	100ns	FILETIME (8byte)
NTP	1900/01/01 00:00:00	約200ps	固定小数点数 (8byte)

[\*1] 32bit環境の場合

# epoch magic number

- epoch形式で格納される10進数の値や16進パターンをある程度覚えておくと、調査や検証時に役立つことがある。

	unix (dec)	unix(hex)	windows (dec)	windows(hex)
1970/01/01 00:00:00 (UTC)	0	00 00 00 00	116,444,736,000,000,000	00 80 3e d5 de d1 9d 01
2000/01/01 00:00:00 (UTC)	946,684,800	80 43 6D <u>38</u>	125,911,584,000,000,000	00 40 6d 25 eb 53 <u>bf 01</u>
2010/01/01 00:00:00 (UTC)	1,262,304,000	00 3B 3D <u>4B</u>	129,067,776,000,000,000	00 00 6e 5c 75 8a <u>ca 01</u>

hexはリトルエンディアンで記載

## コンピュータが扱う時間情報の形式 (2) - Date/Time -

- 年月日、時分秒、曜日、秒未満、タイムゾーン等を別々の形式(型)で記録。  
人間にはわかりやすい。

種類	最小単位	型 (サイズ[*1])
ANSI C	1s	tm (36byte)
Windows	1ms	SYSTEMTIME (16byte)
ISO 8601 [*2]	∞(?)	-

- OS, 言語によって定義される種類は様々だが、基本的にはISO 8601に準拠した定義になっていると思われる。

[\*1] 32bit環境の場合

[\*2] 日時表記に関する国際標準で、形式は“YYYY-MM-DD[T]HH:MM:SS,sss..±hh:mm”

# epoch, Date/Time形式の代表例

FILETIME [\*1]

```
typedef struct _FILETIME {  
    DWORD dwLowDateTime;  
    DWORD dwHighDateTime;  
} FILETIME;
```

SYSTEMTIME [\*2]

```
typedef struct _SYSTEMTIME {  
    WORD wYear;  
    WORD wMonth;  
    WORD wDayOfWeek;  
    WORD wDay;  
    WORD wHour;  
    WORD wMinute;  
    WORD wSecond;  
    WORD wMilliseconds;  
} SYSTEMTIME;
```

[\*1] [http://msdn.microsoft.com/ja-jp/library/x3399a54\(vs.80\).aspx](http://msdn.microsoft.com/ja-jp/library/x3399a54(vs.80).aspx)

[\*2] [http://msdn.microsoft.com/ja-jp/library/tc6fd5zs\(vs.80\).aspx](http://msdn.microsoft.com/ja-jp/library/tc6fd5zs(vs.80).aspx)

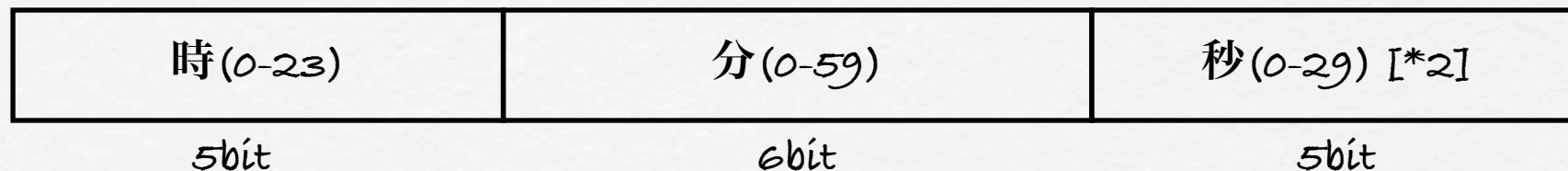
## コンピュータが扱う時間情報の形式 (3) - DOS -

- Date/Time形式に近いが、省スペース設計で人間にはわかりにくい。
- 最小単位は2秒。
- FAT, ZIP, LZHなどで使われている。

Date (2byte)



Time (2byte)



[\*1] 年は1980加算した値

[\*2] 秒は2倍した値

# 目次

1. タイムライン関連ツールと取り組み状況の紹介
2. コンピュータにおける時間の取り扱い
3. ミクロビューによるタイムスタンプの特性検証

## マイクロビューによる検証の背景

- タイムスタンプを格納する型(箱、器)は、秒未満の単位も保存するようになってきている(FILETIME型など)。
- ただし主要なフォレンジック調査ツールでも秒までしか表示しない。
  - 秒未満は無視してソートを行っているツールも多い。
  - ➡ 並びが正確でないために場合によっては解釈を誤る可能性がある。
- 考えられる理由
  - ライブラリが用意されておらず実装が大変。
  - 情報としてそれほど価値がない。
  - ➡ 今後は有用になる可能性あり。

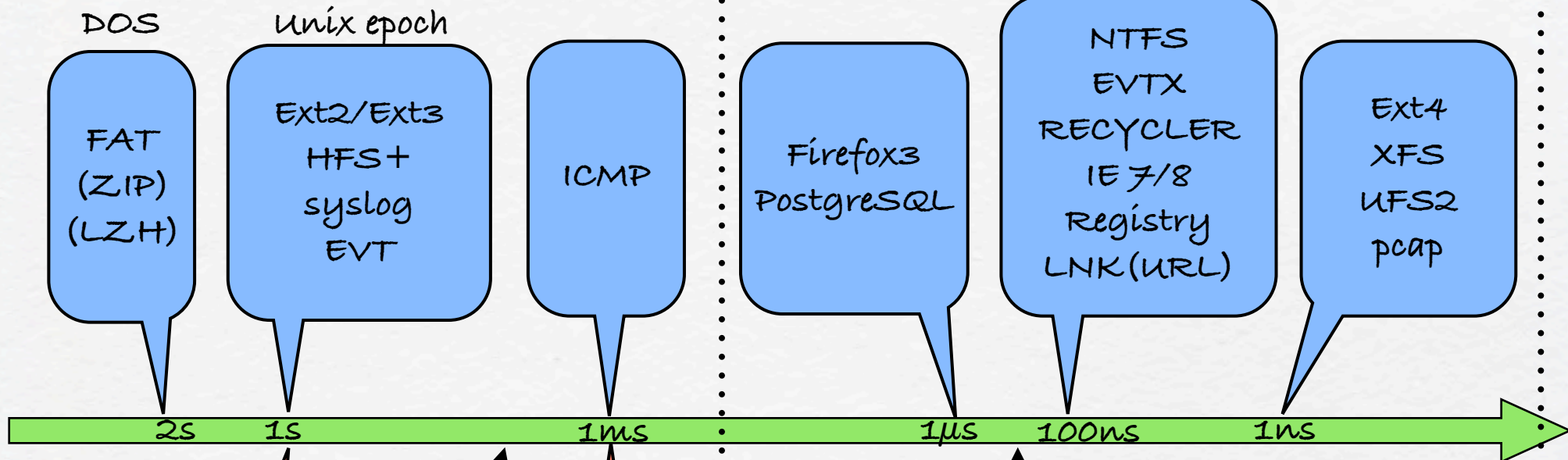
## 秒未満の表現で使う単位（接頭辞）

名称	記号	表記
ミリ	m	0.001
マイクロ	$\mu$	0.000 001
ナノ	n	0.000 000 001
ピコ	p	0.000 000 000 001
フェムト	f	0.000 000 000 000 001
...		
ヨクト	y	0.000 000 000 000 000 000 000 001

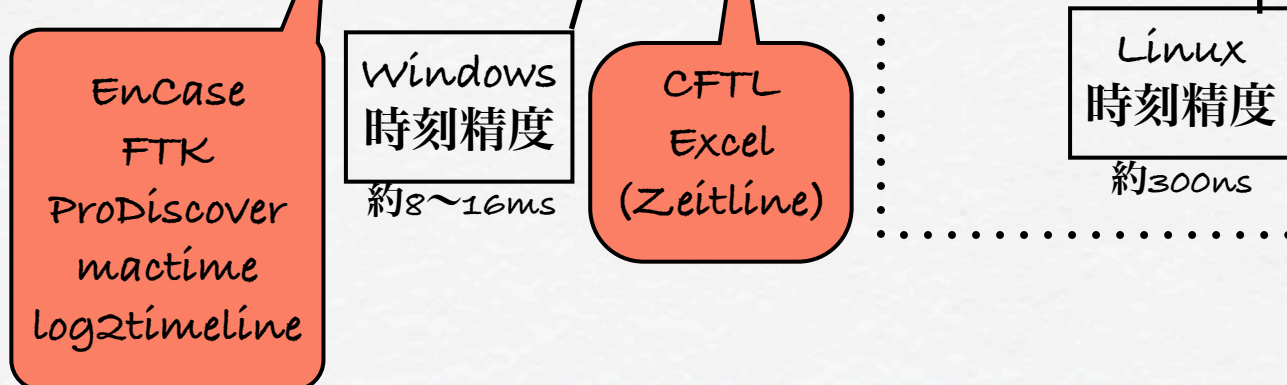
← 現在

# 秒未満の記録/表示精度に関する現状

データ (被調査側)



ツール (調査側)



# ファイルのタイムスタンプを正確に表示/取得する方法

□ `ls --full-time`オプション

□ `stat`コマンド

□ GNU coreutils

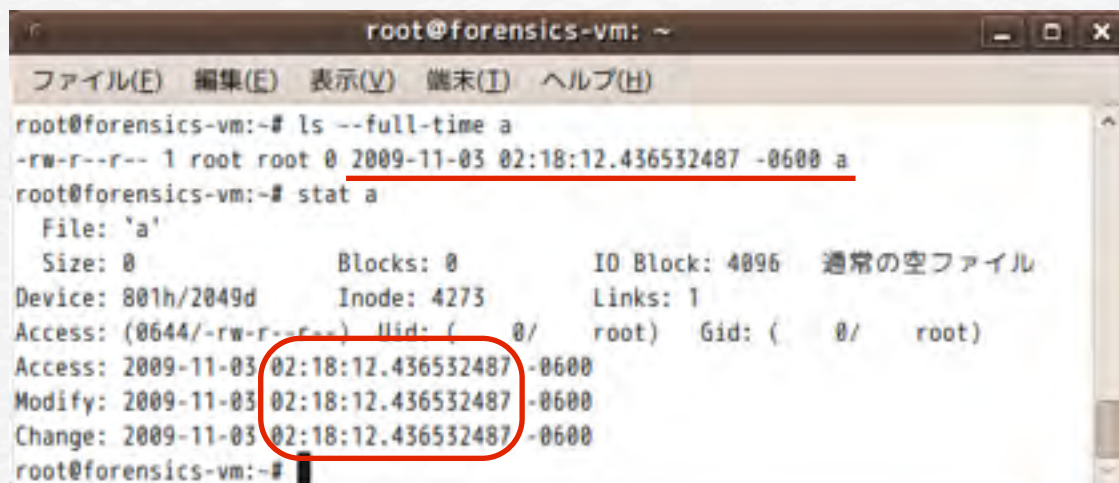
□ cygwinもOK

□ BSD系(Mac OS X含む)はNG(秒までしか表示しない)

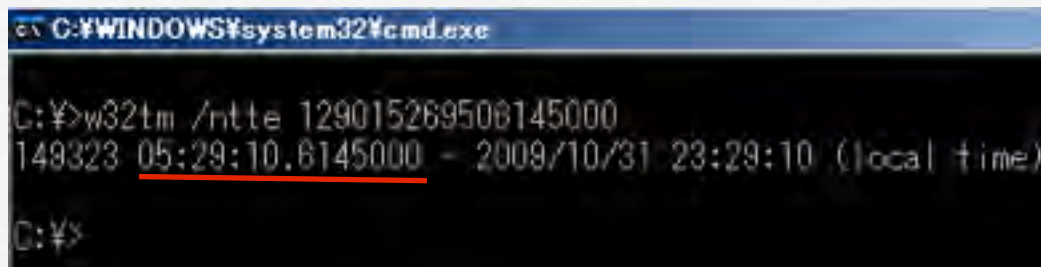
□ ファイル作成日時情報は取得/表示しない

□ w32tm

□ `/ntte`オプションの引数にFILETIME型を10進数で入力



```
root@forensics-vm: ~
ファイル(E) 編集(E) 表示(V) 端末(T) ヘルプ(H)
root@forensics-vm:~# ls --full-time a
-rw-r--r-- 1 root root 0 2009-11-03 02:18:12.436532487 -0600 a
root@forensics-vm:~# stat a
  File: 'a'
  Size: 0          Blocks: 0          IO Block: 4096   通常の空ファイル
Device: 801h/2049d Inode: 4273       Links: 1
Access: (0644/-rw-r--r--)  Uid: (   0/   root)   Gid: (   0/   root)
Access: 2009-11-03 02:18:12.436532487 -0600
Modify: 2009-11-03 02:18:12.436532487 -0600
Change: 2009-11-03 02:18:12.436532487 -0600
root@forensics-vm:~#
```



```
C:\WINDOWS\system32\cmd.exe
C:\>w32tm /ntte 129015269500145000
149323 05:29:10.6145000 - 2009/10/31 23:29:10 (local time)
C:\>
```

# オリジナルプログラム

- FILETIME Extractor (<http://www.kazamiya.net/fte>) - 正確にタイムスタンプを取得するコマンドラインツールを作成
- ファイルのタイムスタンプ以外の時間情報を取得するプログラムを作成
  - ODESSAX - ODESSA (<http://odessa.sourceforge.net/>) をベースに正確な時間情報を表示するように修正
    - pascox (index.dat)
    - galletax (Cookie)
    - rifiutix (INFO2)
- tsconv (<http://www.kazamiya.net/tsconv>) - FILETIMEの16進パターンを正確な時間に変換

今回の検証では出番なし

## 検証内容

- 主にファイルのタイムスタンプ(秒未満の値)から有用な特徴を発見することを目的とし、以下の4項目の検証を実施
  1. `atime`更新の設定(`NtfsDisableLastAccessUpdate`)による差分
  2. Windowsシステムファイルのタイムスタンプ
  3. タイムスタンプ変更ツール
  4. 圧縮形式のタイムスタンプ処理
  
- 検証環境に32bit版のWindows XP Professional SP3とWindows 7 Ultimateを使用
  
- 以降では`atime`(最終アクセス)、`mtime`(最終更新)、`ctime`(エントリ更新)、`ctime`(ファイル作成)という用語で記述する

# 1. atime更新有効時(NtfsDisableLastAccessUpdate=0)

- Windows XP/2003のデフォルト設定
- 以下はWindows XPで各タイムスタンプが更新されるかを検証した結果

	atime	mtime	ctime	ctime
新規作成	√ (2)	√ (2)	√ (2)	√ (1)
実行(プログラム)	√			
内容参照	√			
内容変更	√	√	√	
ファイル名変更	√[*1]		√	
ファイル移動	√ [*1]		√	
ファイルコピー	√		[*2]	√
ファイル削除			[*3]	

[\*1] rename, moveコマンドなどで処理した場合は更新しない

()内の数字は処理順

[\*2] エクスプローラで同一フォルダ上にコピーした場合は更新する

[\*3] ゴミ箱への移動の場合は更新する

# 1. atime更新無効時(NtfsDisableLastAccessUpdate=1)

- Windows Vista/7/2008のデフォルト設定
- 以下はWindows 7で各タイムスタンプが更新されるかを検証した結果

	atime	mtime	ctime	ctime
新規作成	√ (2)	√ (2)	√ (2)	√ (1)
実行(プログラム)				
内容参照				
内容変更		√	√	
ファイル名変更			√	
ファイル移動			√	
ファイルコピー	√		√	√
ファイル削除			[*1]	

- 網掛け部分がatime更新の差分 ( )内の数字は処理順

- ファイルコピー時のctimeの更新は、xpと7の差と思われる(7で常に更新されるようになった)

[\*1] ゴミ箱への移動の場合は更新する

## 2. Windowsシステムファイルのタイムスタンプ

- Windows XP, 7をそれぞれインストールした直後のディスクイメージが対象
- 以下は秒未満が0(hh:mm:ss.0000000)になっているファイル数を計測した結果 (ただしインストール中の時間帯はのぞく)

	Windows XP	Windows 7
全ファイル/フォルダ数	10,711	59,921
atime	0	274
mtime	7934	650
ctime	1	0
crtime	5510	11

XPの上記ファイルはすべて偶数秒→DOS形式で記録されていたと推測される  
7は奇数、偶数秒の両方が存在する

### 3. タイムスタンプ変更ツール (1) - timestomp -

- <http://www.metasploit.com/research/projects/antiforensics/>
- `atime`, `mtime`, `ctime`, `crtime`を任意の日時に変更できる → 秒未満は指定できない
- 他のファイルの日時情報をセットできる → 秒未満も引き継ぐ

```
C:\WINDOWS\system32\cmd.exe
C:\Ytest>timestomp.exe タイムスタンプ.txt -z "Thursday 8/9/2007 10:20:30 AM"

C:\Ytest>stat --printf="%x |%n|atime %n%y |%n|mtime %n%z |%n|ctime %n" タイムスタ
ンプ.txt && getctime.exe タイムスタンプ.txt
2007-08-09 10:20:30.000000000 +0000 |タイムスタンプ.txt|atime
2007-08-09 10:20:30.000000000 +0000 |タイムスタンプ.txt|mtime
2007-08-09 10:20:30.000000000 +0000 |タイムスタンプ.txt|ctime
2007-08-09 10:20:30.000000000 +0000 |タイムスタンプ.txt|crtime

C:\Ytest>stat --printf="%x |%n|atime %n%y |%n|mtime %n%z |%n|ctime %n" C:\WINDOWS
\system32\calc.exe && getctime.exe C:\WINDOWS\system32\calc.exe
2009-11-22 03:17:09.343750000 +0000 |C:\WINDOWS\system32\calc.exe|atime
2001-08-27 12:00:00.000000000 +0000 |C:\WINDOWS\system32\calc.exe|mtime
2009-11-22 03:17:09.343750000 +0000 |C:\WINDOWS\system32\calc.exe|ctime
2009-10-17 05:00:33.624375000 +0000 |C:\WINDOWS\system32\calc.exe|crtime

C:\Ytest>timestomp.exe タイムスタンプ.txt -f C:\WINDOWS\system32\calc.exe

C:\Ytest>stat --printf="%x |%n|atime %n%y |%n|mtime %n%z |%n|ctime %n" タイムスタ
ンプ.txt && getctime.exe タイムスタンプ.txt
2009-11-22 03:17:09.343750000 +0000 |タイムスタンプ.txt|atime
2001-08-27 12:00:00.000000000 +0000 |タイムスタンプ.txt|mtime
2009-11-22 03:17:09.343750000 +0000 |タイムスタンプ.txt|ctime
2009-10-17 05:00:33.624375000 +0000 |タイムスタンプ.txt|crtime
```

-m, a, c, e, zオプションで時間を指定した場合、秒未満は0で揃えられる

-fオプションで別ファイルのタイムスタンプをセットした場合、秒未満も正確にコピーされる。ただし、現在Windows 7では動作しない

### 3. タイムスタンプ変更ツール (2) - Change File Time Stamp -

- <http://www.novell.com/cool solutions/tools/18707.html>
- *atime, mtime, ctime*を任意の日時に変更できる → 元のタイムスタンプのミリ秒(秒未満<sub>3桁</sub>)の情報はそのまま引き継ぐ

```
G:\WINDOWS\system32\cmd.exe
C:\test>stat --print="%x |%n|atime %n%y |%n|mtime %n%z |%n|ctime %n" タイムスタンプ.txt && setcrttime.exe タイムスタンプ.txt
2007-08-09 14:08:09.593750000 +0000 |タイムスタンプ.txt|atime
2007-08-09 14:08:09.593750000 +0000 |タイムスタンプ.txt|mtime
2007-08-09 14:08:09.593750000 +0000 |タイムスタンプ.txt|ctime
2007-08-09 14:08:09.593750000 +0000 |タイムスタンプ.txt|crttime
```

```
G:\WINDOWS\system32\cmd.exe
C:\test>ChangeFileTime.exe -c タイムスタンプ.txt

*****

Filename           : タイムスタンプ.txt
Creation Time      : 09-08-2007 14:08:09
Last Access Time  : 09-08-2007 14:08:09
Last Modified Time: 09-08-2007 14:08:09

*****

Note : You can press <ENTER> if you want to retain previous date/time

Creation Time      : 09-08-2007 14:08:09
New Date( dd-mm-yyyy) : 05-05-2005
New Time( hh-mm-ss)  : 08-08-08
```

```
G:\WINDOWS\system32\cmd.exe
C:\test>stat --print="%x |%n|atime %n%y |%n|mtime %n%z |%n|ctime %n" タイムスタンプ.txt && setcrttime.exe タイムスタンプ.txt
2007-08-09 14:08:09.593000000 +0000 |タイムスタンプ.txt|atime
2007-08-09 14:08:09.593000000 +0000 |タイムスタンプ.txt|mtime
2007-08-09 14:10:59.062500000 +0000 |タイムスタンプ.txt|ctime
2005-05-05 08:08:08.593000000 +0000 |タイムスタンプ.txt|crttime
```

元の秒未満の値"593750.."のうち、変更後もミリ秒"593"までは引き継いでいる

### 3. タイムスタンプ変更ツール (3) - FileTouch -

- <http://www.softreetech.com/24x7/archive/47.htm>
- *atime, mtime, ctime*を任意の日時に変更できる → 秒未満は指定できない
- 時間指定しない場合、現在時刻のミリ秒までの値がセットされる

```
C:\Ytest>FileTouch.exe /W /A /C /T 01:23:45.44 タイムスタンプ.txt
FileTouch version 1.03
Copyright (c) 2003-2005 SoftTree Technologies, Inc. All rights reserved.

Search base path:
Search mask: タイムスタンプ.txt
New file date/time will be 09/08/2007 01:23:45

Processing files...
File タイムスタンプ.txt date/time changed successfully

Done.

C:\Ytest>stat --printf='%x |%n|atime %n&y |%n|mtime %n&z |%n|ctime %n
ンブ.txt 88 zetcrtime.exe タイムスタンプ.txt
2007-08-09 01:23:45.000000000 +0000 |タイムスタンプ.txt|atime
2007-08-09 01:23:45.000000000 +0000 |タイムスタンプ.txt|mtime
2007-08-09 13:32:01.000000000 +0000 |タイムスタンプ.txt|ctime
2007-08-09 01:23:45.000000000 +0000 |タイムスタンプ.txt|ctime

C:\Ytest>FileTouch.exe /W /A /C タイムスタンプ.txt
FileTouch version 1.03
Copyright (c) 2003-2005 SoftTree Technologies, Inc. All rights reserved.

Search base path:
Search mask: タイムスタンプ.txt
New file date/time will be 09/08/2007 14:01:48

Processing files...
File タイムスタンプ.txt date/time changed successfully

Done.

C:\Ytest>stat --printf='%x |%n|atime %n&y |%n|mtime %n&z |%n|ctime %n
ンブ.txt 88 zetcrtime.exe タイムスタンプ.txt
2007-08-09 14:01:48.531000000 +0000 |タイムスタンプ.txt|atime
2007-08-09 14:01:48.531000000 +0000 |タイムスタンプ.txt|mtime
2007-08-09 14:01:48.531250000 +0000 |タイムスタンプ.txt|ctime
2007-08-09 14:01:48.531000000 +0000 |タイムスタンプ.txt|ctime
```

現在時刻設定(/Tオプション無し)の場合、ミリ秒まで(例では531)をセットしている  
(*ctime*の値より、本来の秒未満の値は"53125"であることがわかる)

## 4. 圧縮形式(ZIP/LZH)のタイムスタンプ

- ZIPやLZHなどの一般的な圧縮形式では、圧縮対象ファイルのmtimeをヘッダ情報として保持するように設計されている。
- ZIPの仕様(<http://www.pkware.com/documents/casestudies/APPNOTE.TXT>)
  - ヘッダ部にmtimeをDOS形式で保存する領域がある。
- LZH(LHA)の仕様(<http://www2m.biglobe.ne.jp/~dolphin/lha/lha-header.htm>)
  - ヘッダのバージョンは3種類あり、レベル0,1はDOS形式で、レベル2はunix epochでmtimeを格納する。
- ZIP, LZHどちらも拡張用のフィールドにunix epochやFILETIMEなどの型でatime, mtime, ctimeをそれぞれ格納できるような規定がある。ツールが独自にこの領域を使うことも可能。

## 4. 圧縮/展開ツールの比較

- ツールにより圧縮/展開時のタイムスタンプをセットする処理は異なる。
- 以下は圧縮時のタイムスタンプの格納形式/丸め方と、展開時にタイムスタンプをどこに復元するかを検証した結果

	ZIP圧縮	ZIP展開	LZH圧縮	LZH展開
Windows標準 (XP/7)	DOS/切上	mtime ⇒atime, ctime	-	-
Lhaplus	DOS/切上	mtime	DOS/切上	mtime
7zip	DOS/切上	mtime	-	mtime [*2]
Lhaz	DOS/四捨五入 unix epoch/切捨 [*1]	mtime⇒atime (atime, mtime, ctime) [*3]	unix epoch/切捨	mtime⇒atime
EXPLZH	DOS/四捨五入	mtime (atime, mtime, ctime) [*3]	unix epoch/切捨 FILETIME [*1]	mtime⇒atime (mtime, atime, ctime) [*3]

[\*1] - 拡張フィールドに格納する。 表内の $x \Rightarrow y$ は復元したタイムスタンプ $x$ の値が $y$ にもコピーされることを示す

[\*2] - EXPLZHで圧縮されたファイルを展開した際には、 $mtime \Rightarrow atime$ となった。原因は不明。

[\*3] - 拡張フィールドに格納されていれば、それらの値(atime, mtime, ctime)を復元する

## まとめ

- 時系列のソート機能は必ずしも正確でない場合がある。
- ➡ 複数のソース、データをマージして解析する際には、時間のずれだけでなく様々な処理により生じる欠落により、誤差が拡大する点に注意
- 秒未満の記録情報を調べることにより、有用な手がかりが得られる可能性がある。
- 別媒体/リモートからコピーされたファイルの特定
- タイムスタンプ変更ツール使用の特定
- 圧縮ファイルから展開されたファイルの特定

## 参考情報

- Guidelines For Providing Multimedia Timer Support

<http://www.microsoft.com/whdc/system/sysinternals/mm-timer.mspx>

- Timestamp evidence correlation by model based clock hypothesis testing

<http://www.willassen.no/svein/pub/efor08.pdf>

- An Improved Clock Model for Translating Timestamps

<http://www.infosec.jmu.edu/reports/jmu-infosec-tr-2007-001.pdf>