A Systematic Approach to Understanding MACB Timestamps on Unix-like Systems

DFRWS EU 2022

Aurélien Thierry[1], Tilo Müller[2]

March 30th, 2022

Hochschule Hof
University of Applied Sciences

[1]aurelien.thierry@telekom.de - Deutsche Telekom Security, Germany
[2]tilo.mueller@hof-university.de - Hof University of Applied Sciences, Germany

Motivation & Contribution

File timestamps: fundamental forensics artifact

1. What timestamps are modified by a given operation?
2. Given timestamps, what happened to the file(s)?

Motivation & Contribution

File timestamps: fundamental forensics artifact

1. What timestamps are modified by a given operation?
2. Given timestamps, what happened to the file(s)?

(SANS Institute)

Motivation & Contribution

File timestamps: fundamental forensics artifact

1. What timestamps are modified by a given operation?
2. Given timestamps, what happened to the file(s)?

(SANS Institute)

Our contributions:

- Study of timestamp updates on **Linux, OpenBSD, FreeBSD and macOS**
- Framework for automated testing and profiling
- Better understanding
- Tables for practitioners

Hof University

POSIX: Portable Operating System Interface

POSIX: Specifications for portability across Unix systems

- **Specifies:** utilities (cp, cat...), libc, kernel (system calls)
- Loosely applies to Linux, OpenBSD, FreeBSD, macOS
- Only Mac OS X 10.5 Leopard is POSIX-certified

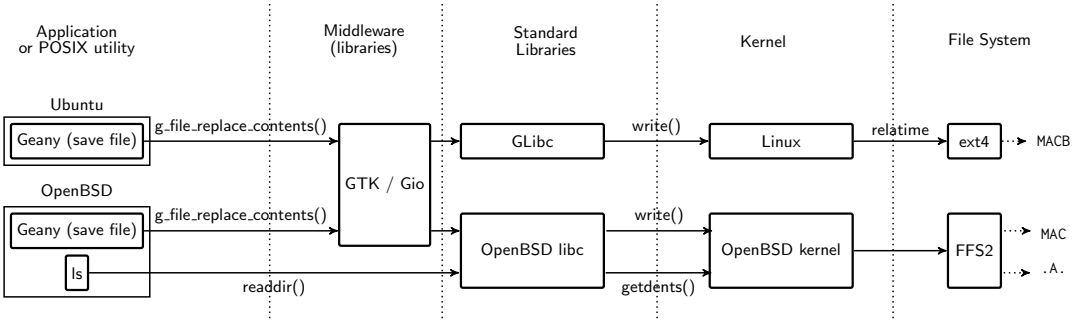
MACB Timestamps:

- `M`: Last data **M**odification
- `A`: Last data **A**ccess
- `C`: Last file status **C**hange (typically: `chmod`)
- `B`: File Creation (**B**irth)

```
u@h$ statx test.txt
test.txt: (stat + B)
M: Tue Mar 22 15:42:41 2022 - ns: 895045511
A: Tue Mar 22 15:42:46 2022 - ns: 571081872
C: Tue Mar 22 15:42:41 2022 - ns: 899045542
B: Wed Dec  1 19:09:40 2021 - ns: 295783662
```

POSIX: MAC only (**M**odify, **A**ccess, **C**hange) - no **B**irth timestamp

Unix: Software Stack



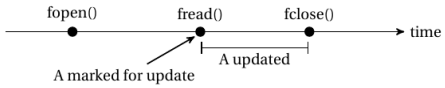
Each layer has to be considered.

POSIX: marked for update, updated

Timestamp updates are a two-step process: **Marked for update → Updated**
Update shall happen:

- When file is not open anymore (`fclose()`)
- Before timestamp manipulation occurs (`stat()`, `futimens()`)
- When the kernel decides (after some time)

File Read (libc):



1. `fopen("r")`: No impact on timestamps
2. `fread()`: Mark A for update
3. A shall be updated before `fclose()`

[redacted] The last data access timestamp shall be marked for update by the first successful execution of *fgetc*(), *fgets*(), *fread*(), *fscanf*(), *getc*(), *getchar*(), *getdelim*(), *getline*(), *gets*(), or *scanf*() [redacted]

Hof University

Tests against POSIX Compliance

Approach: tests run on the OS

	File Read (file.txt):
1 For each watched file: `stat(file.path)`	`stat("file.txt")`
2 `sleep(1.1s)`	
3 $t_1 =$ current_time()	
4 **Run the tested operation**	`fopen("r")`, `fread()`, `fclose()`
5 $t_2 =$ current_time()	
6 `sleep(1.1s)`	
7 For each watched file: $t_{MAC} =$ `stat(file.path)`	`stat("file.txt")`
8 Compare t_{MAC} with t_1 and t_2 and against the specification	A ✓

Compliance Tests & Automated Profiling

Compliance tests only verify expected behavior (POSIX Utilities, libc, Kernel, File System)

- C framework

Automated Profiling: Determine timestamp updates of any given operation

1. Typical Operations (New File, File Read, File Write, File Copy...):
   - Multiple implementations for each operation
   - **File Read**: `fopen("r")+fread()+fclose()`, or `cat`
   - C implementation
2. Middleware: GTK (GNOME) and Qt (KDE)
   - C implementation
3. Applications: Text Editors
   - python implementation with pyautogui (keystrokes)

Tested Operating Systems

System	Version	File System	Machine
Linux	Ubuntu 20.04.3 LTS (Linux 5.10.0)	ext4 mounted with `strictatime`	Lenovo P1
	Ubuntu 18.04.3 LTS		Lenovo P1
	ArchLinux 5.3.8		VirtualBox
OpenBSD	6.5, 6.8, 7.0	FFS1, FFS2	VirtualBox
FreeBSD	12.0, 13.0-RELEASE-p4	UFS2	VirtualBox
macOS	10.13.6 (Supported until 2020)	HFS+	Mac mini

Tools

- Specification and documentation (POSIX, man pages...)
- Automated compliance checks and profiling (C and python)

Challenges:

- Unexpected results, sometimes difficult to reproduce
- Interference from other processes

- **Source code analysis**
- **Debug** (`strace`...)

```
The worst of all namespace operations - renaming directory. "Perverted"
doesn't even start to describe it. Somebody in UCB had a heck of a trip...
```

Linux: comment on renaming directory (Local Dir Move)

```
$ strace cat test.txt

execve("/usr/bin/cat", ["cat", "test.txt"], 0x7ffedd5e24b8 /* 60 vars */) = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
[...]
openat(AT_FDCWD, "test.txt", O_RDONLY)  = 3
fstat(3, {st_mode=S_IFREG|0664, st_size=14, ...}) = 0
read(3, "lorem ipsum\n", 131072)        = 14
write(1, "lorem ipsum\n", 12)           = 14
read(3, "", 131072)                     = 0
close(3)                                = 0
```

File Systems & Kernels

File Systems:

	POSIX	ext4 (Linux)	FFS2 (OpenBSD)	UFS2 (FreeBSD)	HFS+ (macOS)
Birth timestamp (B)	(No)	Yes	Yes	Yes	Yes
Timestamp Resolution	$\leq$ 1s	1ns	1ns	1ns	1s

Kernels:

- OpenBSD (FFS2): The **B** field exists but is always 0
- FreeBSD (UFS2): Timestamp resolution can be configured (1s, **default: 1µs**, 1ns)

File Systems & Kernels

File Systems:

	POSIX	ext4 (Linux)	FFS2 (OpenBSD)	UFS2 (FreeBSD)	HFS+ (macOS)
Birth timestamp (B)	(No)	Yes	Yes	Yes	Yes
Timestamp Resolution	$\leq$ 1s	1ns	1ns	1ns	1s

Kernels:

- OpenBSD (FFS2): The **B** field exists but is always 0
- FreeBSD (UFS2): Timestamp resolution can be configured (1s, **default: 1µs**, 1ns)

Kernels: Mount Options

Mount options act as a filter for timestamp updates

- Disable A updates → improve performance

Linux:

- **relatime (default)**: A only updated if more than 1 day old, or if earlier than M or C
- strictatime: A updates are always performed
- noatime: A updates are never performed

Mount option	OpenBSD	FreeBSD	macOS
(default)	MAC updates are all performed, B is always 0	MACB updates are all performed	
noatime	A updates are performed only if M or C is also marked for update	A updates are never performed	

Timestomping

root can directly modify the file system and alter any timestamp

What can a *standard user* arbitrarily set? (`touch`, `futimensat()`)

	POSIX	Linux	OpenBSD	FreeBSD	macOS
M	Yes				
A	Yes				
C	No				
B		No	No	Yes	Yes

C can only be updated:

- Modifying M, A or B (with `touch`) updates C to the current time
- It cannot be arbitrarily set to any other time (past or future)

Timestamp Rules

	New File/Dir touch, mkdir	File Read /Execute cat, exec()	Symlink Read/Follow readlink		File Write >, >>	File/Dir Change chmod, chown	New/Delete Hardlink ln, rm	File/Dir Rename, Local File Move mv, rename()				Local Dir Move mv, rename()				Volume File/Dir Move mv				File/Dir Copy (new) cp	File Copy (existing) cp
			POSIX Linux	OpenBSD FreeBSD macOS				POSIX	Linux	OpenBSD FreeBSD	macOS	POSIX	Linux	OpenBSD FreeBSD	macOS	POSIX	Linux	OpenBSD	FreeBSD macOS		
M	M	.	.	.	M	.	.	.	.	.	.	.	.	M	.	m	m	m		M	M
A	A	A	A	.	.	.	.	.	.	.	.	.	.	.	.	a	a	a		A	.
C	C	.	.	.	C	C	C	*	C	.	.	*	C	C	.	*	C	C		C	C
B[1]	B	.	.	.	.	.	.		.	.	.		.	.	.		B	m		B	.

	Dir Traversal cd	**Dir** Listing ls			**Dir**: New/Rename/Delete Child (File/Dir/Hardlink), File Moved into (Local) touch, mkdir, ln, mv, cp, rm	**Dir**: Dir Moved into (Local) mv			**Dir**: Child Read/Exec /Write/Change cat, readlink, >>
		POSIX Linux OpenBSD macOS		FreeBSD		POSIX Linux FreeBSD macOS		OpenBSD	
M	.	.		.	M	M		M	.
A	.	A		.	.	.		A	.
C	.	.		.	C	C		C	.
B[1]	.	.		.	.	.			.

M/A/C/B	M/A/C/B is updated to current time
m/a/c/b	M/A/C/B is inherited from m/a/c/b of source file/dir
.	M/A/C/B is not modified
*	POSIX: Choice is left to the implementation

Legend

[1]: Linux, FreeBSD and macOS only; always 0 on OpenBSD; not specified by POSIX

Mount options have to be considered as filters

Hof University

Timestamp Rules

	New File/Dir `touch, mkdir`	File Read /Execute `cat, exec()`	Symlink Read/Follow `readlink`		File Write `>, >>`	File/Dir Change `chmod, chown`	New/Delete Hardlink `ln, rm`	File/Dir Rename, Local File Move `mv, rename()`				Local Dir Move `mv, rename()`				Volume File/Dir Move `mv`				File/Dir Copy (new) `cp`	File Copy (existing) `cp`
			POSIX Linux	OpenBSD FreeBSD macOS				POSIX	Linux	OpenBSD FreeBSD	macOS	POSIX	Linux	OpenBSD FreeBSD	macOS	POSIX	Linux	OpenBSD	FreeBSD macOS		
M	M	.	.	.	M	.	.	.	.	.	.	.	.	M	.	m	m	m		M	M
A	A	A	A	.	.	.	.	.	.	.	.	.	.	.	.	a	a	a		A	.
C	C	.	.	.	C	C	C	*	C	.	.	*	C	C	.	*	C	C		C	C
B[1]	B	.	.	.	.	.	.		.	.	.		.	.	.		B	m		B	.

	Dir Traversal `cd`	**Dir** Listing `ls`			**Dir**: New/Rename/Delete Child (File/Dir/Hardlink), File Moved into (Local) `touch, mkdir, ln, mv, cp, rm`	**Dir**: Dir Moved into (Local) `mv`		**Dir**: Child Read/Exec /Write/Change `cat, readlink, >>`
		POSIX Linux OpenBSD macOS		FreeBSD		POSIX Linux FreeBSD macOS	OpenBSD	
M	.	.		.	M	M	M	.
A	.	A		.	.	.	A	.
C	.	.		.	C	C	C	.
B[1]	.	.		.	.	.		.

M/A/C/B	M/A/C/B is updated to current time
m/a/c/b	M/A/C/B is inherited from m/a/c/b of source file/dir
.	M/A/C/B is not modified
*	POSIX: Choice is left to the implementation

Legend

[1]: Linux, FreeBSD and macOS only; always 0 on OpenBSD; not specified by POSIX

Mount options have to be considered as filters

Timestamp Rules

	New File/Dir touch, mkdir	File Read /Execute cat, exec()	Symlink Read/Follow readlink		File Write >, >>	File/Dir Change chmod, chown	New/Delete Hardlink ln, rm	File/Dir Rename, Local File Move mv, rename()				Local Dir Move mv, rename()				Volume File/Dir Move mv				File/Dir Copy (new) cp	File Copy (existing) cp
			POSIX Linux	OpenBSD FreeBSD macOS				POSIX	Linux	OpenBSD FreeBSD	macOS	POSIX	Linux	OpenBSD FreeBSD	macOS	POSIX	Linux	OpenBSD FreeBSD	macOS		
M	M	.	.	.	M	.	.	.	.	.	.	.	.	M	.	m	m	m		M	M
A	A	A	A	.	.	.	.	.	.	.	.	.	.	.	.	a	a	a		A	.
C	C	.	.	.	C	C	C	*	C	.		*	C	C	.	*	C	C		C	C
B[1]	B	.	.	.	.	.	.		.	.			.	.	.		B	m		B	.

	Dir Traversal cd	**Dir** Listing ls		**Dir**: New/Rename/Delete Child (File/Dir/Hardlink), File Moved into (Local) touch, mkdir, ln, mv, cp, rm	**Dir**: Dir Moved into (Local) mv		**Dir**: Child Read/Exec /Write/Change cat, readlink, >>
		POSIX Linux OpenBSD macOS	FreeBSD		POSIX Linux FreeBSD macOS	OpenBSD	
M	.	.	.	M	M	M	.
A	.	A	.	.	.	A	.
C	.	.	.	C	C	C	.
B[1]	.	.	.	.	.		.

M/A/C/B	M/A/C/B is updated to current time
m/a/c/b	M/A/C/B is inherited from m/a/c/b of source file/dir
.	M/A/C/B is not modified
*	POSIX: Choice is left to the implementation

Legend

[1]: Linux, FreeBSD and macOS only; always 0 on OpenBSD; not specified by POSIX

Mount options have to be considered as filters

Timestamp Rules

	New File/Dir touch, mkdir	File Read /Execute cat, exec()	Symlink Read/Follow readlink		File Write >, >>	File/Dir Change chmod, chown	New/Delete Hardlink ln, rm	File/Dir Rename, Local File Move mv, rename()				Local Dir Move mv, rename()				Volume File/Dir Move mv				File/Dir Copy (new) cp	File Copy (existing) cp
			POSIX Linux	OpenBSD FreeBSD macOS				POSIX	Linux	OpenBSD FreeBSD	macOS	POSIX	Linux	OpenBSD FreeBSD	macOS	POSIX	Linux	OpenBSD	FreeBSD macOS		
M	M	.	.	.	M	.	.	.	.	.	.	.	.	M	.	m	m	m		M	M
A	A	A	A	.	.	.	.	.	.	.	.	.	.	.	.	a	a	a		A	.
C	C	.	.	.	C	C	C	*	C	.		*	C	C	.	*	C	C		C	C
B[1]	B	.	.	.	.	.	.		.	.			.	.	.		B	m		B	.

	Dir Traversal cd	**Dir** Listing ls			**Dir**: New/Rename/Delete Child (File/Dir/Hardlink), File Moved into (Local) touch, mkdir, ln, mv, cp, rm	**Dir**: Dir Moved into (Local) mv		**Dir**: Child Read/Exec /Write/Change cat, readlink, >>
		POSIX Linux OpenBSD macOS		FreeBSD		POSIX Linux FreeBSD macOS	OpenBSD	
M	.	.		.	M	M	M	.
A	.	A		.	.	.	A	.
C	.	.		.	C	C	C	.
B[1]	.	.		.	.	.		.

M/A/C/B	M/A/C/B is updated to current time
m/a/c/b	M/A/C/B is inherited from m/a/c/b of source file/dir
.	M/A/C/B is not modified
*	POSIX: Choice is left to the implementation

Legend

[1]: Linux, FreeBSD and macOS only; always 0 on OpenBSD; not specified by POSIX

Mount options have to be considered as filters

Timestamp Rules

	New File/Dir touch, mkdir	File Read /Execute cat, exec()	Symlink Read/Follow readlink		File Write >, >>	File/Dir Change chmod, chown	New/Delete Hardlink ln, rm	File/Dir Rename, Local File Move mv, rename()				Local Dir Move mv, rename()				Volume File/Dir Move mv				File/Dir Copy (new) cp	File Copy (existing) cp
			POSIX Linux	OpenBSD FreeBSD macOS				POSIX	Linux	OpenBSD FreeBSD	macOS	POSIX	Linux	OpenBSD FreeBSD	macOS	POSIX	Linux	OpenBSD	FreeBSD macOS		
M	M	.	.	.	M	.	.	.	.	.	.	.	.	M	.	m	m	m		M	M
A	A	A	A	.	.	.	.	.	.	.	.	.	.	.	.	a	a	a		A	.
C	C	.	.	.	C	C	C	*	C	.		*	C	C	.	*	C	C		C	C
B[1]	B	.	.	.	.	.	.	.	.	.		.	.	.	.		B	m		B	.

	Dir Traversal cd	**Dir** Listing ls			**Dir**: New/Rename/Delete Child (File/Dir/Hardlink), File Moved into (Local) touch, mkdir, ln, mv, cp, rm	**Dir**: Dir Moved into (Local) mv		**Dir**: Child Read/Exec /Write/Change cat, readlink, >>
		POSIX Linux OpenBSD macOS		FreeBSD		POSIX Linux FreeBSD macOS	OpenBSD	
M	.	.		.	M	M	M	.
A	.	A		.	.	.	A	.
C	.	.		.	C	C	C	.
B[1]	.	.		.	.	.		.

M/A/C/B	M/A/C/B is updated to current time
m/a/c/b	M/A/C/B is inherited from m/a/c/b of source file/dir
.	M/A/C/B is not modified
*	POSIX: Choice is left to the implementation

Legend

[1]: Linux, FreeBSD and macOS only; always 0 on OpenBSD; not specified by POSIX

Mount options have to be considered as filters

Hof University

Timestamp Rules

	New File/Dir touch, mkdir	File Read /Execute cat, exec()	Symlink Read/Follow readlink		File Write >, >>	File/Dir Change chmod, chown	New/Delete Hardlink ln, rm	File/Dir Rename, Local File Move mv, rename()				Local Dir Move mv, rename()				Volume File/Dir Move mv				File/Dir Copy (new) cp	File Copy (existing) cp
			POSIX Linux	OpenBSD FreeBSD macOS				POSIX	Linux	OpenBSD FreeBSD	macOS	POSIX	Linux	OpenBSD FreeBSD	macOS	POSIX	Linux	OpenBSD	FreeBSD macOS		
M	M	.	.	.	M	.	.	.	.	.	.	.	.	M	.	m	m	m		M	M
A	A	A	A	.	.	.	.	.	.	.	.	.	.	.	.	a	a	a		A	.
C	C	.	.	.	C	C	C	*	C	.	*	C	C	.	*	C	C			C	C
B[1]	B	.	.	.	.	.	.	.	.	.	.	.	.	.			B	m		B	.

	Dir Traversal cd	**Dir** Listing ls		**Dir**: New/Rename/Delete Child (File/Dir/Hardlink), File Moved into (Local) touch, mkdir, ln, mv, cp, rm	**Dir**: Dir Moved into (Local) mv		**Dir**: Child Read/Exec /Write/Change cat, readlink, >>
		POSIX Linux OpenBSD macOS	FreeBSD		POSIX Linux FreeBSD macOS	OpenBSD	
M	.	.	.	M	M	M	.
A	.	A	.	.	.	A	.
C	.	.	.	C	C	C	.
B[1]	.	.	.	.	.		.

M/A/C/B	M/A/C/B is updated to current time
m/a/c/b	M/A/C/B is inherited from m/a/c/b of source file/dir
.	M/A/C/B is not modified
*	POSIX: Choice is left to the implementation

Legend

[1]: Linux, FreeBSD and macOS only; always 0 on OpenBSD; not specified by POSIX

Mount options have to be considered as filters

Hof University

Timestamp Rules

	New File/Dir touch, mkdir	File Read /Execute cat, exec()	Symlink Read/Follow readlink		File Write >, >>	File/Dir Change chmod, chown	New/Delete Hardlink ln, rm	File/Dir Rename, Local File Move mv, rename()				Local Dir Move mv, rename()				Volume File/Dir Move mv				File/Dir Copy (new) cp	File Copy (existing) cp
			POSIX Linux	OpenBSD FreeBSD macOS				POSIX	Linux	OpenBSD FreeBSD	macOS	POSIX	Linux	OpenBSD FreeBSD	macOS	POSIX	Linux	OpenBSD	FreeBSD macOS		
M	M	.	.	.	M	.	.	.	.	.	.	.	.	M	.	m	m	m		M	M
A	A	A	A	.	.	.	.	.	.	.	.	.	.	.	.	a	a	a		A	.
C	C	.	.	.	C	C	C	*	C	.	.	*	C	C	.	*	C	C		C	C
B[1]	B	.	.	.	.	.	.		.	.	.		.	.	.		B	m		B	.

	Dir Traversal cd	**Dir** Listing ls			**Dir**: New/Rename/Delete Child (File/Dir/Hardlink), File Moved into (Local) touch, mkdir, ln, mv, cp, rm	**Dir**: Dir Moved into (Local) mv		**Dir**: Child Read/Exec /Write/Change cat, readlink, >>
		POSIX Linux OpenBSD macOS		FreeBSD		POSIX Linux FreeBSD macOS	OpenBSD	
M	.	.		.	M	M	M	.
A	.	A		.	.	.	A	.
C	.	.		.	C	C	C	.
B[1]	.	.		.	.	.		.

M/A/C/B	M/A/C/B is updated to current time
m/a/c/b	M/A/C/B is inherited from m/a/c/b of source file/dir
.	M/A/C/B is not modified
*	POSIX: Choice is left to the implementation

Legend

[1]: Linux, FreeBSD and macOS only; always 0 on OpenBSD; not specified by POSIX

Mount options have to be considered as filters

POSIX Compliance: Results

279 tests implemented:

- Including 177 tests against mandatory POSIX (C181, 2018) behavior
- **No implementation is fully compliant**
- Linux (with `strictatime`) is mostly compliant

Failed tests:

- Irrelevant edge-cases (`stdin`, `abort()`, identical `chmod`, missing `gets()`)

Bugs (?) to be investigated:

- FreeBSD: Missing some A updates
- macOS: `futimens()` sometimes fails to modify timestamps

Gio (GLib, GTK/GNOME): File Copy

`g_file_copy()`: Tested on Linux (Ubuntu)

- M is inherited from source, ACB are updated
  - ≠ Standard File Copy (cp): MACB updated
- MA truncated to the microsecond resolution
  - Known Gio bug from 2010[3] (low priority)
  - Could be used to identify files copied with Nautilus[4] (default file manager for GNOME)

```
u@h$ statx test.txt
test.txt: (stat + B)
M: Mon Mar 14 14:34:34 2022 - ns: 977007889
A: Mon Mar 14 14:34:33 2022 - ns: 672996104
C: Mon Mar 14 14:34:36 2022 - ns: 381020578
B: Mon Mar 14 14:33:14 2022 - ns: 840282157
u@h$ # Nautilus: copy "test.txt" to "test (copy).txt"
u@h$ statx test\ \(copy\).txt
test (copy).txt: (stat + B)
M: Mon Mar 14 14:34:34 2022 - ns: 977007000
A: Mon Mar 14 14:34:48 2022 - ns: 233127000
C: Mon Mar 14 14:34:48 2022 - ns: 233127654
B: Mon Mar 14 14:34:48 2022 - ns: 233127654
```

[3] https://gitlab.gnome.org/GNOME/glib/-/issues/369

[4] https://bugs.launchpad.net/ubuntu/+source/nautilus/+bug/642596

Text editors - Modifying files: Two strategies

1. Direct write: MC updated
2. Write to a new temporary file then replace the original file with it: MACB updated

Editor	IO Middleware	Read	Modify
Vim nowritebackup	POSIX	A	MC
nano	POSIX	A	MC
Leafpad	POSIX	A	MC
Visual Studio Code	Electron	A	MC
Notepadqq	Qt	A	MC
Sublime Text	(Unknown)	A	MC
Kate	KTextEditor (KIO)	A	MAC
Atom	Electron	A	MAC
Vim –clean	POSIX	A	MACB
Emacs	POSIX	A	MACB
Code::Blocks	POSIX	A[1]	MACB
gedit	GTK	A	MACB
Bluefish	Gio	A	MACB
Geany	Gio	A	MACB
TeXstudio	Qt	A	MACB
JED	S-Lang	A	MACB

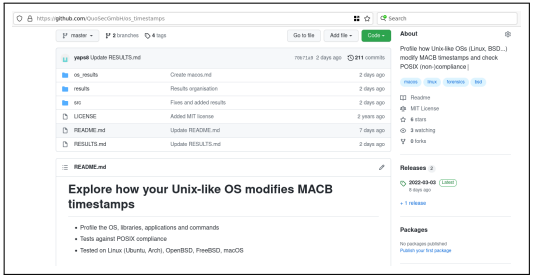
[1] Code::Blocks, when reading an empty file, does not modify any timestamp.

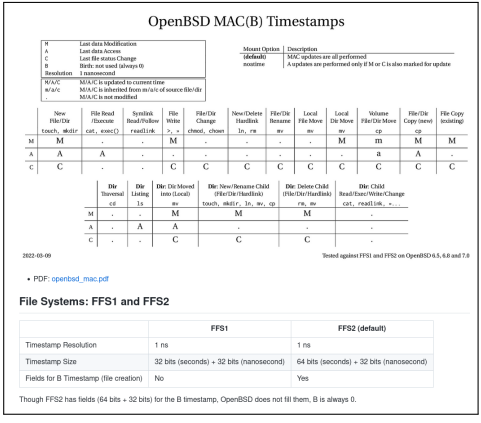
Tested on Linux (Ubuntu)

os_timestamps (Github)

https://github.com/QuoSecGmbH/os_timestamps/

- Open-Source (MIT License)
- Up-to-date results and tables
- Open to **feedback**, discussion, contributions (new tests?)...





Hof University

Conclusion & Next Steps

Open-source framework with tables `https://github.com/QuoSecGmbH/os_timestamps/`

Understanding timestamp updates:

- Need to consider the software stack, especially **mount options**
- Many variations despite POSIX (Symlink, Directory Listing, Dir Move, B...)

Next:

- Investigate unexpected results (kernel bugs?)
- Up-to-date macOS with APFS, File Managers, Mobile (Android, iOS), Windows?

Questions ?

(aurelien.thierry@telekom.de, Twitter: @yaps8)