PerSeVerE: Persistency Semantics for Verification Under Ext4

Michalis Kokologiannakis [1] Ilya Kaysin [2] Azalea Raad [3] Viktor Vafeiadis [1]

January, 2021

[1]MPI-SWS
[2]National Research University, Higher School of Economics, Russia/JetBrains Research
[3]Imperial College London

Writing a Text Editor

Save buffer as "`foo.txt`"

$d_f = \text{open}(\text{"}\texttt{foo.txt}\text{"}, \texttt{O\_WRONLY}|\texttt{O\_CREAT}|\texttt{O\_TRUNC});$
$\text{write}(d_f, \texttt{buffer});$
$\text{close}(d_f);$
$\text{printf}(\text{"}\texttt{File saved}\backslash\texttt{n}\text{"});$

Does replace-via-truncate successfully write `foo.txt`?

- Consistency: ✓
- Persistency: ???

What Does `man` Say?

"If `auto_da_alloc` is enabled, ext4 will detect the [...] replace-via-truncate [...] and [...] the data blocks of the new file are forced to disk [on close] [...]."

man 5 ext4

replace-via-truncate is used by e.g., `nano` for writing files and backups

Upon a crash, both the edited file and its backup might be empty!

Desiderata

Formal Semantics

Effective Model Checking

Challenges:

- POSIX written in prose
- `ext4` not fully POSIX-compliant
- `ext4` and weak memory?

- State Space Explosion

Contributions

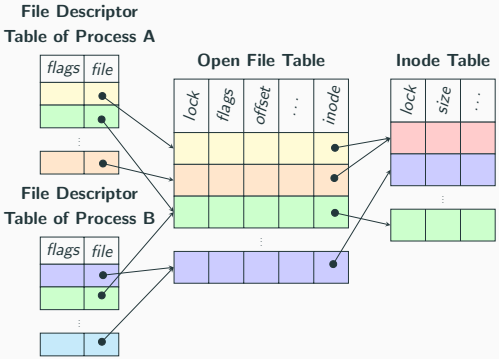
- **PerSeVerE**: framework for verification under `ext4`
  - formal model integrated with C/C++ consistency semantics
  - based on kernel's code and stress testing
  - effective model checking algorithm
- Report bugs in commonly used text editors like `emacs`, `vim` and `nano`

`ext4` Semantics

Filesystem operations

Opening a file:

- $d_f = \text{open}(\text{“}\texttt{foo.txt}\text{”}, \texttt{O\_APPEND})$



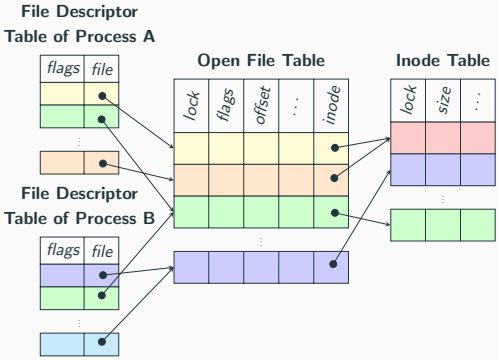
Filesystem operations

Opening a file:

- $d_f = \text{open}(\text{“}\texttt{foo.txt}\text{”}, \texttt{O\_APPEND})$

Reading from a file:

- $r = \text{read}(d_f, 3)$, $r = \text{pread}(d_f, 3, 42)$



Filesystem operations

Opening a file:

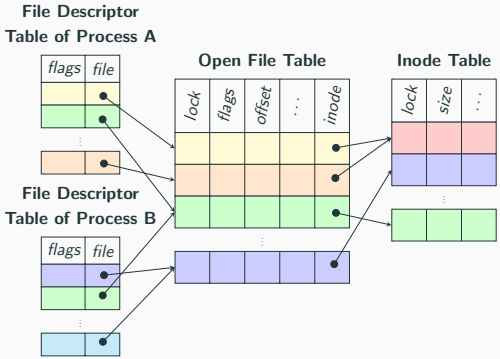
- $d_f = \text{open}(\text{“}\texttt{foo.txt}\text{”}, \texttt{O\_APPEND})$

Reading from a file:

- $r = \text{read}(d_f, 3)$, $r = \text{pread}(d_f, 3, 42)$

Writing to a file:

- $\text{write}(d_f, \text{“}\texttt{foo}\text{”})$, $\text{pwrite}(d_f, \text{“}\texttt{bar}\text{”}, 0)$



Filesystem operations

Opening a file:

- $d_f = \text{open}(\text{“}\texttt{foo.txt}\text{”}, \texttt{O\_APPEND})$

Reading from a file:

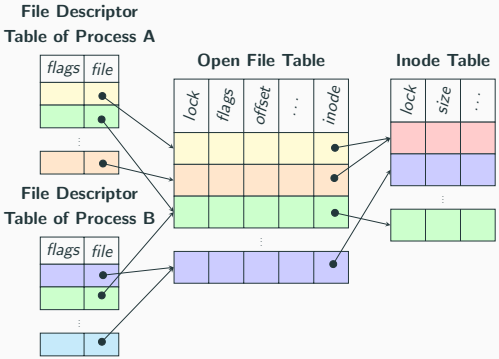
- $r = \text{read}(d_f, 3)$, $r = \text{pread}(d_f, 3, 42)$

Writing to a file:

- $\text{write}(d_f, \text{“}\texttt{foo}\text{”})$, $\text{pwrite}(d_f, \text{“}\texttt{bar}\text{”}, 0)$

Closing a file:

- $\text{close}(d_f)$



Filesystem operations

Opening a file:

- $d_f = \text{open}(\text{“}\texttt{foo.txt}\text{”}, \texttt{O\_APPEND})$

Reading from a file:

- $r = \text{read}(d_f, 3)$, $r = \text{pread}(d_f, 3, 42)$

Writing to a file:

- $\text{write}(d_f, \text{“}\texttt{foo}\text{”})$, $\text{pwrite}(d_f, \text{“}\texttt{bar}\text{”}, 0)$

Closing a file:

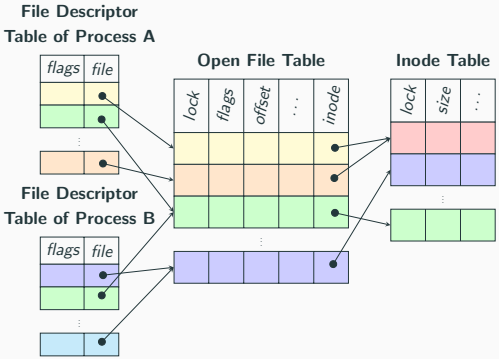
- $\text{close}(d_f)$

Directory operations:

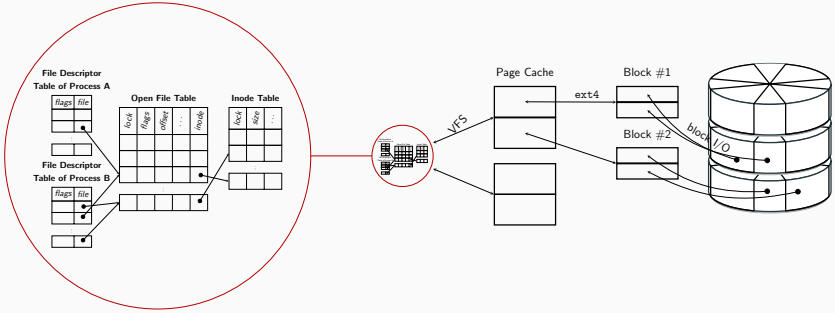
- creat, link, unlink, rename

Synchronization operations:

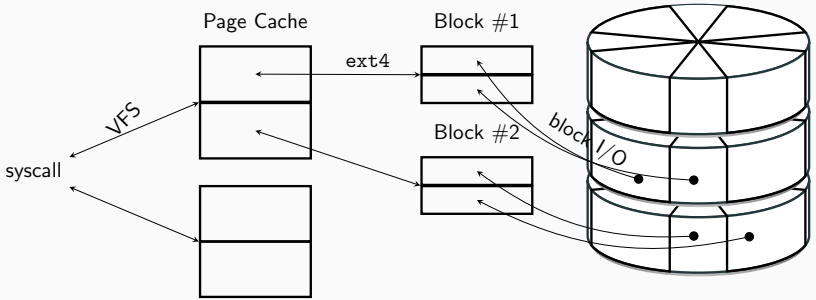
- fsync, sync



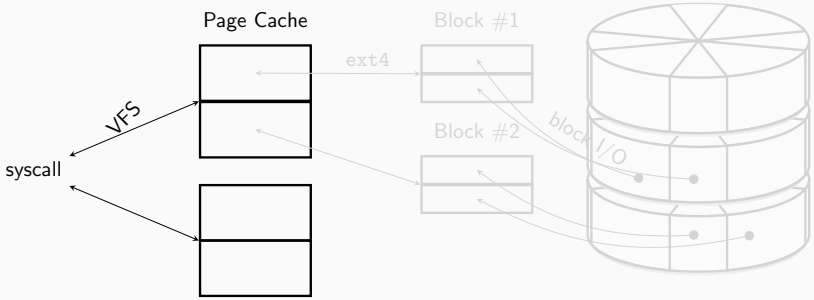
I/O Stack


File Descriptor
Table of Process A
flags
file
Open File Table
Inode Table
lock
flags
offset
...
inode
lock
size
...
File Descriptor
Table of Process B
flags
file
VFS
Page Cache
ext4
Block #1
Block #2
block I/O

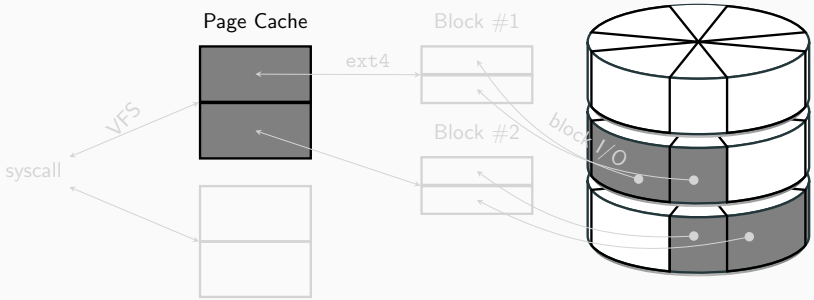
I/O Stack



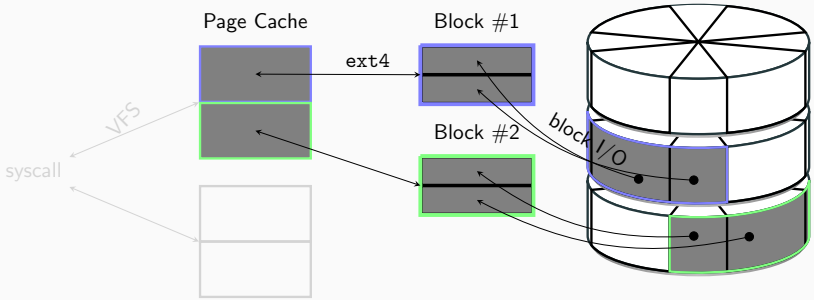
I/O Stack



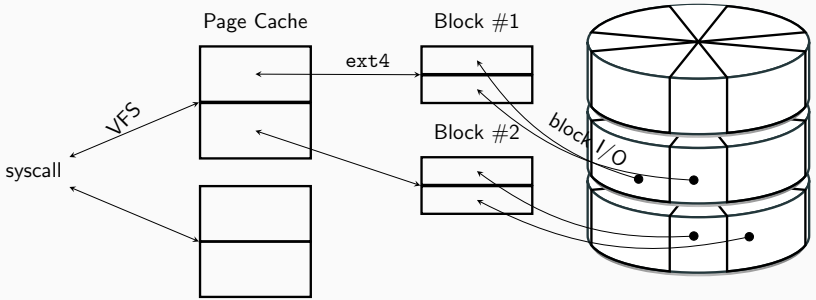
I/O Stack



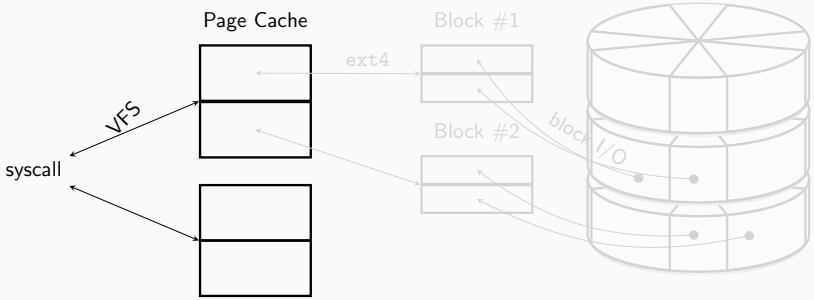
I/O Stack



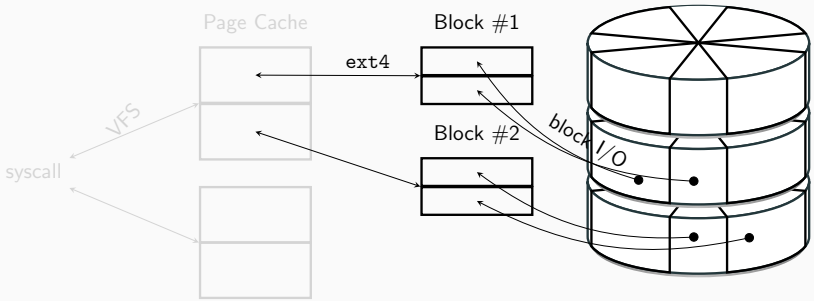
I/O Stack



I/O Stack



I/O Stack



Consistency Semantics

Writes vs Writes : c-atomic, c-ordered

$$d_f \rightsquigarrow \text{“foo”}$$

$$\text{pwrite}\,(d_f, \text{“bar”}, 0); \;\Big\|\; \text{pwrite}\,(d_f, \text{“qux”}, 0);$$
$$r = \text{pread}\,(d_f, 3, 0);$$

Possible outcomes: “bar”, “qux”

Consistency Semantics

Writes vs Writes : c-atomic, c-ordered

Reads vs Overwrites :

$$d_f \rightsquigarrow \text{“}\texttt{foobar}\text{”}, \mathit{offset} = 0$$

$$\text{write}\,(d_f, \text{“}\texttt{bar}\text{”}); \;\Big\|\; r = \text{read}\,(d_f, 3);$$

Possible outcomes: “`foo`”, “`bar`”

Consistency Semantics

Writes vs Writes : c-atomic, c-ordered

Reads vs Overwrites :

$d_f \rightsquigarrow$ "foo"

write $(d_f,$ "bar"$);$ $\Big\|$ $r =$ pread $(d_f, 3, 0);$

Possible outcomes: "foo", "bar", "far", "fao", etc

Consistency Semantics

Writes vs Writes	:	c-atomic, c-ordered
Reads vs Overwrites	:	c-nonatomic, c-unordered

$$d_f \rightsquigarrow \text{“foo”}$$

$$\text{write}\,(d_f, \text{“bar”}); \;\Big\|\; r = \text{pread}\,(d_f, 3, 0);$$

Possible outcomes: “foo”, “bar”, “far”, “fao”, etc

Consistency Semantics

Writes vs Writes	:	c-atomic, c-ordered
Reads vs Overwrites	:	c-nonatomic, c-unordered
Reads vs Appends	:	

$$d_f \rightsquigarrow \text{“}\texttt{foo}\text{”}, \texttt{O\_APPEND}, \mathit{page\_size} = 3$$

$$\text{write}\,(d_f, \text{“}\texttt{barqux}\text{”}); \;\Big\|\; r = \text{pread}\,(d_f, 42, 0);$$

Possible outcomes: “`foo`”, “`foobar`”, “`foobarqux`”

Consistency Semantics

Writes vs Writes	:	c-atomic, c-ordered
Reads vs Overwrites	:	c-nonatomic, c-unordered
Reads vs Appends	:	page-c-atomic, page-c-ordered

$$d_f \rightsquigarrow \text{“}\texttt{foo}\text{”}, \texttt{O\_APPEND}, \mathit{page\_size} = 3$$

$$\text{write}\,(d_f, \text{“}\texttt{barqux}\text{”}); \;\Big\|\; r = \text{pread}\,(d_f, 42, 0);$$

Possible outcomes: “`foo`”, “`foobar`”, “`foobarqux`”

Consistency Semantics

Writes vs Writes	:	c-atomic, c-ordered
Reads vs Overwrites	:	c-nonatomic, c-unordered
Reads vs Appends	:	page-c-atomic, page-c-ordered
Directory vs All	:	

“foo.txt” $\mapsto$ “foo”

$d_b = \text{creat}(\text{“foo.tmp”});$ $\text{write}(d_b, \text{“bar”}); \text{close}(d_b);$ $\text{rename}(\text{“foo.tmp”}, \text{“foo.txt”});$	$d_f = \text{open}(\text{“foo.txt”}, \texttt{O\_RDONLY});$ $r = \text{read}(d_f, 3);$

Possible outcomes: “foo”, “bar”

Consistency Semantics

Writes vs Writes	:	c-atomic, c-ordered
Reads vs Overwrites	:	c-nonatomic, c-unordered
Reads vs Appends	:	page-c-atomic, page-c-ordered
Directory vs All	:	c-atomic, c-ordered

“foo.txt” $\mapsto$ “foo”

$d_b = \text{creat}(\text{“foo.tmp”});$ $\text{write}(d_b, \text{“bar”}); \text{close}(d_b);$ $\text{rename}(\text{“foo.tmp”}, \text{“foo.txt”});$	‖	$d_f = \text{open}(\text{“foo.txt”}, \texttt{O\_RDONLY});$ $r = \text{read}(d_f, 3);$

Possible outcomes: “foo”, “bar”

Persistency Semantics

Overwrites :

$d_f \rightsquigarrow$ "`foo`", $sector\_size = 1, block\_size = 3$

pwrite $(d_f,$ "`bar`", $0)$;

Possible outcomes: "`foo`", "`boo`", "`bao`", "`bar`" (not "`fao`", "`far`")

Persistency Semantics

Overwrites :

$d_f \rightsquigarrow$ "`foo`", $sector\_size = 1, block\_size = 1$

pwrite $(d_f,$ "`bar`", $0)$;

Possible outcomes: "`foo`", "`boo`", "`bao`", "`bar`" (also "`fao`", "`far`")

Persistency Semantics

Overwrites : sector p-atomic, p-unordered

$d_f \rightsquigarrow$ "foo", $sector\_size = 1, block\_size = 1$

pwrite $(d_f,$ "bar", $0);$

Possible outcomes: "foo", "boo", "bao", "bar" (also "fao", "far")

Persistency Semantics

Overwrites : sector p-atomic, p-unordered

Appends :

$d_f \rightsquigarrow$ "`foo`", `O_APPEND`, $block\_size = 3$

pwrite (d_f, "`bar`", 0);

Possible outcomes: "`foo`", "`foobar`"

Persistency Semantics

Overwrites : sector p-atomic, p-unordered

Appends : prefix-p-atomic *,

$d_f \rightsquigarrow$ "`foo`", `O_APPEND`, $block\_size = 1$

pwrite (d_f, "`bar`", 0);

Possible outcomes: "`foo`", "`foob`", "`fooba`", "`foobar`"

Persistency Semantics

Overwrites : sector p-atomic, p-unordered

Appends : prefix-p-atomic *,

$d_f \rightsquigarrow$ "`foo`", `O_APPEND`, $block\_size = 2$

pwrite (d_f, "`bar`", 0);

Possible outcomes: "`foo`", "`foob`", ~~"`fooba`"~~, "`foobar`" (also "`foo0`")

Persistency Semantics

Overwrites : sector p-atomic, p-unordered

Appends : prefix-p-atomic *,

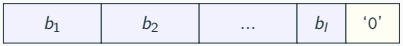
$d_f \rightsquigarrow$ "`foo`", `O_APPEND`, *block_size* = 2

pwrite (d_f, "`bar`", 0);

Possible outcomes: "`foo`", "`foob`", ~~"`fooba`"~~, "`foobar`" (also "`foo0`")

Before crash:

After crash:



Persistency Semantics

Overwrites : sector p-atomic, p-unordered

Appends : prefix-p-atomic *, same-file-p-ordered

$d_a \rightsquigarrow \varnothing, d_b \rightsquigarrow \varnothing$

write(d_a, "foo");
write(d_b, "bar");

One possible outcome: $d_a \rightsquigarrow \varnothing, d_b \rightsquigarrow$ "bar"

Persistency Semantics

Overwrites : sector p-atomic, p-unordered
Appends : prefix-p-atomic *, same-file-p-ordered
{f}sync

$d_a \rightsquigarrow \varnothing, d_b \rightsquigarrow \varnothing$

write(d_a, "foo");
fsync(d_a);
write(d_b, "bar");

An impossible outcome: $d_a \rightsquigarrow \varnothing, d_b \rightsquigarrow$ "bar"

Persistency Semantics

Overwrites	:	sector p-atomic, p-unordered
Appends	:	prefix-p-atomic *, same-file-p-ordered
{f}sync	:	p-nonatomic, p-ordered

$d_a \rightsquigarrow \varnothing, d_b \rightsquigarrow \varnothing$

write(d_a, "foo");
fsync(d_a);
write(d_b, "bar");

An impossible outcome: $d_a \rightsquigarrow \varnothing, d_b \rightsquigarrow$ "bar"

Full model

- Axiomatic model in the style of RC11
- Assumes the consistency model contains an `hb` relation
- Includes a `pb` relation denoting the order in which disk accesses persist

Full model

- Axiomatic model in the style of RC11
- Assumes the consistency model contains an hb relation
- Includes a pb relation denoting the order in which disk accesses persist

$$(\mathtt{I} \cap \mathtt{D}) \times (\mathtt{D} \setminus \mathtt{I}) \subseteq \mathtt{pb} \qquad \text{(PB-INIT)}$$

$$[\mathtt{DW}]; (\mathtt{hb} \cap \mathtt{ssec}); [\mathtt{DW}] \subseteq \mathtt{pb} \qquad \text{(PB-SECTOR)}$$

$$[\mathtt{DW}]; (\mathtt{hb} \cap \mathtt{bseq}); [\mathtt{DW}] \subseteq \mathtt{pb} \qquad \text{(PB-BLOCK)}$$

$$[\mathtt{DW}_{\mathtt{Floc}}]; (\mathtt{hb} \cap \mathtt{sf}); [\mathtt{DW}_{\mathtt{Dsizeloc}}] \subseteq \mathtt{pb} \qquad \text{(PB-META)}$$

$$[\mathtt{S} \cup \mathtt{FS}]; \mathtt{hb}; [\mathtt{D}] \cup [\mathtt{D}]; \mathtt{hb}; [\mathtt{S}] \cup [\mathtt{DW}]; (\mathtt{hb} \cap \mathtt{sf}); [\mathtt{FS}] \subseteq \mathtt{pb} \qquad \text{(PB-SYNC)}$$

$$[\mathtt{DW}_{\mathtt{Dnameloc}} \cup \mathtt{DW}^{\mathtt{trunc}}]; \mathtt{hb}; [\mathtt{D} \setminus \mathtt{DW}_{\mathtt{Floc}}] \subseteq \mathtt{pb} \qquad \text{(PB-DIROPS)}$$

$$(\mathtt{atom}; \mathtt{pb}) \cup (\mathtt{pb}; \mathtt{atom}) \subseteq \mathtt{pb} \qquad \text{(PB-ATOM)}$$

where

$$\mathtt{atom} \triangleq ([\mathtt{DW} \setminus \mathtt{DW}^{\mathtt{zero}}]; (\mathtt{ssec} \cap \mathtt{sid}); [\mathtt{DW} \setminus \mathtt{DW}^{\mathtt{zero}}]) \cup ([\mathtt{DW}^{\mathtt{rename}}]; \mathtt{sid}; [\mathtt{DW}^{\mathtt{rename}}])\,.$$

Full model

- Axiomatic model in the style of RC11
- Assumes the consistency model contains an hb relation
- Includes a pb relation denoting the order in which disk accesses persist

$$(\mathtt{I} \cap \mathtt{D}) \times (\mathtt{D} \setminus \mathtt{I}) \subseteq \mathtt{pb} \qquad \text{(PB-INIT)}$$

$$[\mathtt{DW}]; (\mathtt{hb} \cap \mathtt{ssec}); [\mathtt{DW}] \subseteq \mathtt{pb} \qquad \text{(PB-SECTOR)}$$

$$[\mathtt{DW}]; (\mathtt{hb} \cap \mathtt{bseq}); [\mathtt{DW}] \subseteq \mathtt{pb} \qquad \text{(PB-BLOCK)}$$

$$[\mathtt{DW}_{\mathtt{Floc}}]; (\mathtt{hb} \cap \mathtt{sf}); [\mathtt{DW}_{\mathtt{Dsizeloc}}] \subseteq \mathtt{pb} \qquad \text{(PB-META)}$$

$$[\mathtt{S} \cup \mathtt{FS}]; \mathtt{hb}; [\mathtt{D}] \cup [\mathtt{D}]; \mathtt{hb}; [\mathtt{S}] \cup [\mathtt{DW}]; (\mathtt{hb} \cap \mathtt{sf}); [\mathtt{FS}] \subseteq \mathtt{pb} \qquad \text{(PB-SYNC)}$$

$$[\mathtt{DW}_{\mathtt{Dnameloc}} \cup \mathtt{DW}^{\mathtt{trunc}}]; \mathtt{hb}; [\mathtt{D} \setminus \mathtt{DW}_{\mathtt{Floc}}] \subseteq \mathtt{pb} \qquad \text{(PB-DIROPS)}$$

$$(\mathtt{atom}; \mathtt{pb}) \cup (\mathtt{pb}; \mathtt{atom}) \subseteq \mathtt{pb} \qquad \text{(PB-ATOM)}$$

where

$$\mathtt{atom} \triangleq ([\mathtt{DW} \setminus \mathtt{DW}^{\mathtt{zero}}]; (\mathtt{ssec} \cap \mathtt{sid}); [\mathtt{DW} \setminus \mathtt{DW}^{\mathtt{zero}}]) \cup ([\mathtt{DW}^{\mathtt{rename}}]; \mathtt{sid}; [\mathtt{DW}^{\mathtt{rename}}])\,.$$

$$[\mathtt{S} \cup \mathtt{FS}]; \mathtt{hb}; [\mathtt{D}]$$

Results

Editor Bugs

Save buffer as "`foo.txt`"

d_f = open ("`foo.txt`", `O_WRONLY|O_CREAT|O_TRUNC`);
write (d_f, `buffer`); fsync (d_f);
close (d_f);
printf("`File saved\n`");

The bug depends on the manifestation of a race

We reported the bug to the developers of `nano`
$\rightsquigarrow$ proposed a fix and verified it with PerSeVerE
$\rightsquigarrow$ our fixes were subsequently merged

We reproduced the same buggy pattern in `emacs` and `vim`

Conclusions

Summary

- **PerSeVerE**: framework for verification under `ext4`
  - formal model integrated with C/C++ consistency semantics
  - based on kernel's code and stress testing
  - effective model checking algorithm
- PerSeVerE is available at `github.com/MPI-SWS/genmc`

Future work

- Formalize other aspects of `ext4`
- Extend PerSeVerE for other filesystems