Implementing the NTFS filesystem in Rust

Reusable code from firmware level up to user-mode

Colin Finck
Working on the ReactOS Operating System for the past 15 years

Ported the HermitCore Unikernel from C to Rust in 2017

Reverse-Engineering industrial systems and writing Rust connectors at ENLYZE
MOTIVATION

Windows / ReactOS bootloader in Rust
MOTIVATION

Filesystem support for the growing number of Rust operating systems
MOTIVATION

Food for the Rust no_std ecosystem

No standard library

Crates that are able to function without the Rust standard library.

Crates

Displaying 1-10 of 3012 total results

Sort by Recent Downloads
Debian: CVE-2021-33285: ntfs-3g -- security update

Microsoft CVE-2021-33285 (Elevation of Privilege)

- **Severity:** 9
- **CVSS:** (AV:N/AC:M/Au:N/C:C/I:C)
- **Published:** 06/08/2021
- **Created:** 06/09/2021
- **Added:** 06/08/2021
- **Modified:** 09/14/2021

**Time to first bug**

<table>
<thead>
<tr>
<th>Filesystem</th>
<th>Time</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Btrfs</td>
<td>5s</td>
<td>BUG()</td>
</tr>
<tr>
<td>Ext4</td>
<td>2h</td>
<td>BUG()</td>
</tr>
<tr>
<td>F2fs</td>
<td>10s</td>
<td>BUG()</td>
</tr>
<tr>
<td>Gfs2</td>
<td>8m</td>
<td>Double free</td>
</tr>
<tr>
<td>Hfs</td>
<td>30s</td>
<td>Page Fault</td>
</tr>
<tr>
<td>Hfsplus</td>
<td>25s</td>
<td>Page Fault</td>
</tr>
<tr>
<td>Nilfs2</td>
<td>1m</td>
<td>Page Fault</td>
</tr>
<tr>
<td>Ntfs</td>
<td>4m</td>
<td>Soft lockup</td>
</tr>
<tr>
<td>Ocf2</td>
<td>15s</td>
<td>BUG()</td>
</tr>
<tr>
<td>Reiserfs</td>
<td>25s</td>
<td>BUG()</td>
</tr>
<tr>
<td>Xfs</td>
<td>1h45m</td>
<td>Soft lockup</td>
</tr>
</tbody>
</table>

- Linux 4.3 or newer
- 3 AFL instances on my laptop
- Don’t believe us?
  - Live crash demo
NTFS FACTS
NTFS FACTS

• Most popular desktop filesystem
• Initial release 1993, current version 3.1 from 2001
• Unicode filenames
• 64-bit numbers for sizes and timestamps
  • 16 EiB volume and file size, limited to 8 PiB by current Windows
  • Years 1601 – 60056, no Year-2038 problem
NTFS UNICODE STRINGS

• Windows NT first major OS to support Unicode
• Chose UCS-2 fixed-length 2 bytes per character encoding
• Moved to UTF-16 with Windows 2000

64K characters ought to be enough for anybody
struct NtfsString<'a>(&'a [u8])

impl<'a> Display for NtfsString<'a>
impl<'a> Eq for NtfsString<'a>
impl<'a> Ord for NtfsString<'a>

impl<'a> PartialEq<&str> for NtfsString<'a>
impl<'a> PartialEq<NtfsString<'a>> for &str
impl<'a> PartialOrd<&str> for NtfsString<'a>
impl<'a> PartialOrd<NtfsString<'a>> for &str

fn to_string_checked(&self) -> Option<String>
fn to_string_lossy(&self) -> String
NTFS TIMESTAMP
struct NtfsTime(u64)

/// Returns the number of 100-nanosecond intervals since January 1, 1601
fn nt_timestamp(&self) -> u64

#[cfg(feature = "chrono")]
impl TryFrom<DateTime<Utc>> for NtfsTime
impl From<NtfsTime> for DateTime<Utc>

#[cfg(feature = "std")]
impl TryFrom<SystemTime> for NtfsTime
<table>
<thead>
<tr>
<th>NTFS Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alternate Data Streams</td>
</tr>
<tr>
<td>B-Tree Indexes</td>
</tr>
<tr>
<td>Compression</td>
</tr>
<tr>
<td>Encryption</td>
</tr>
<tr>
<td>Hard links</td>
</tr>
<tr>
<td>Journaling</td>
</tr>
<tr>
<td>Quotas</td>
</tr>
<tr>
<td>Reparse points</td>
</tr>
<tr>
<td>Security descriptors</td>
</tr>
<tr>
<td>Sparse files</td>
</tr>
</tbody>
</table>
ANATOMY OF AN NTFS FILE
FILE Record

$STANDARD_INFORMATION

$FILE_NAME

$DATA
FILE Record

$STANDARD_INFORMATION

Creation Time
Access Time
Modification Time
Record Modification Time
DOS File Flags
...

$FILE_NAME

$DATA
<table>
<thead>
<tr>
<th>$STANDARD_INFORMATION</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$FILE_NAME</td>
<td></td>
</tr>
<tr>
<td>$DATA</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>43 6f 6e 67 72 61 74 73</th>
</tr>
</thead>
<tbody>
<tr>
<td>20 59 6f 75 20 66 6f 75</td>
</tr>
<tr>
<td>$STANDARD_INFORMATION</td>
</tr>
<tr>
<td>------------------------</td>
</tr>
<tr>
<td>FILE Record</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
FILE Record

$STANDARD_INFORMATION

$DATA

Data Run #0

$DATA

43 6f 6e 67 72 61 74 73
20 59 6f 75 20 6f 6e 64 20 74 68 65 20 65
FILE Record

$STANDARD_INFORMATION

$FILE_NAME

$DATA

Data Run #0

$DATA

43 6f 6e 67 72 61 74 73 20 59 6f 75 20 66 6f 75 6e 64 20 74 68 65 20 65 61 73 74 65 72 20 65 67
FILE Record

$STANDARD_INFORMATION
$FILE_NAME
$DATA

Data Run #0

43 6f 6e 67 72 61 74 73 20 59 6f 75 20 66 6f 75 6e 64 20 74 68 65 20 65 61 73 74 65 72 20 65 67

$DATA
FILE Record

$STANDARD_INFORMATION
$FILE_NAME

$DATA

Data Run #0
Data Run #1

$DATA

43 6f 6e 67 72 61 74 73 20 59 6f 75 20 66 6f 75 6e 64 20 74 68 65 20 65 61 73 74 65 72 20 65 67

Data Run #1

43 6f 6e 67 72 61 74 73 20 59 6f 75 20 66 6f 75 6e 64 20 74 68 65 20 65 61 73 74 65 72 20 65 67
TRAITS
enum NtfsAttributeValue<'n, 'f> {
    Resident(NtfsResidentAttributeValue<'f>),
    NonResident(NtfsNonResidentAttributeValue<'n, 'f>),
    AttributeListNonResident(...),
}

trait Read {
    fn read(&mut self, buf: &mut [u8]) -> Result<usize, std::io::Error>;
}

trait Seek {
    fn seek(&mut self, pos: SeekFrom) -> Result<u64, std::io::Error>;
}
trait Read {
    fn read(&mut self, buf: &mut [u8]) -> Result<usize, std::io::Error>;
}

trait Seek {
    fn seek(&mut self, pos: SeekFrom) -> Result<u64, std::io::Error>;
}

struct Error {
    repr: Repr,
}

enum Repr {
    Os(i32),
    Simple(ErrorKind),
    SimpleMessage(ErrorKind, &'static &'static str),
    Custom(Box<Custom>),
}
trait Read {
    fn read(&mut self, buf: &mut [u8]) -> Result<usize, std::io::Error>;
}

trait Seek {
    fn seek(&mut self, pos: SeekFrom) -> Result<u64, std::io::Error>;
}

struct Error {
    repr: Repr,
}

enum Repr {
    Os(i32),
    Simple(ErrorKind),
    SimpleMessage(ErrorKind, &'static &'static str),
    Custom(Box<Custom>),
}
trait Read {
    fn read(&mut self, buf: &mut [u8]) -> Result<usize, binread::io::Error>;
}

trait Seek {
    fn seek(&mut self, pos: SeekFrom) -> Result<u64, binread::io::Error>;
}
trait Read {
    fn read(&mut self, buf: &mut [u8]) -> Result<usize, binread::io::Error>;
}

trait Seek {
    fn seek(&mut self, pos: SeekFrom) -> Result<u64, binread::io::Error>;
}
struct NtfsFile<'a, T: Read + Seek> {
    fs: &'a mut T,
}

```rust
struct NtfsFile<'a, T: Read + Seek> {
    fs: &'a mut T,
}

trait ReadAt { fn read_at(&self, pos: u64, buf: &mut [u8]) -> Result<usize>; }
```
ADDITIONAL CONSTRAINTS

• `no_std` ⇒ No OS-specific synchronization

• All borrow checks at compile-time
1. Only immutable references in base structs

```rust
struct Ntfs {
    cluster_size: u32,
    sector_size: u16,
    serial_number: u64,
}

struct NtfsFile<'n> {
    ntfs: &'n Ntfs,
}
```
2. Pass mutable references when needed

```rust
impl<'n> NtfsFile<'n> {
    fn data<T>(&self, fs: &mut T) -> ...
    where
        T: Read + Seek,
    }
}```
3. Provide helper structs with attached mutable reference

```rust
pub struct NtfsAttributesAttached<'n, 'f, 'a, T: Read + Seek> {
    fs: &'a mut T,
    attributes: NtfsAttributes<'n, 'f>,
}

impl<'n, 'f, 'a, T> Iterator for NtfsAttributesAttached<'n, 'f, 'a, T>
where
    T: Read + Seek,
{
    type Item = Result<NtfsAttributeItem<'n, 'f>>;

    fn next(&mut self) -> Option<Self::Item> {
        self.attributes.next(self.fs)
    }
}
```
3. Provide helper structs with attached mutable reference

Not an option for

```rust
fn next<'a, T>(&'a mut self, fs: &mut T) -> Option<Entry<'a>>
where
  T: Read + Seek,
{ ... }
```

Waiting for GATs and StreamingIterator
RECORD FIXUPS
FILE Record

$STANDARD_INFORMATION

$FILE_NAME

$DATA
One morning, when Gregor Samsa woke from trouble he found himself transformed into a horrible vermin. He lay on his armour-like back, and if he lifted his head a little he could see his brown belly, slightly domed and di
One morning, when Gregor Samsa woke from trouble d dreams, he found himse
One morning, when Gregor Samsa woke from trouble d dreams, he found himse

ed his head a little he

could see his brown bell y, slily domed and di
One morning, when Gregor Samsa woke from trouble, he found himself transformed in his bed into a horrible vermin. He lay on his armour-like back, and if he lifted his head a little he could see his brown belly, slightly domed and di
One morning, when Gregor Samsa woke from trouble d dreams, he found himse

1f transformed in his be

d into a horrible vermin

He lay on his armour-

like back, and if he lift

ed his head a little he

could see his brown bell

y, sli..tly domed and di
One morning, when Gregor Samsa woke from trouble dreams, he found himself transformed in his bed into a horrible vermin. He lay on his armour皮肤 like back, and if he lifted his head a little he could see his brown belly tiely domed and di
One morning, when Gregor Samsa woke from trouble dreams, he found himself transformed in his bed into a horrible vermin. He lay on his armour-like back, and if he lifted his head a little he could see his brown belly under the domed and di...
One morning, when Gregor Samsa woke from trouble
d dreams, he found himse
lf transformed in his be
d into a horrible vermin
. He lay on his armour-
lke back, and if he lift
ed his head a little he
could see his brown bell
y, sli. tly domed and di
One morning, when Gregor Samsa woke from trouble dreams, he found himself transformed in his bed into a horrible vermin. He lay on his armour-like back, and if he lifted his head a little he could see his brown belly slily domed and di
One morning, when Gregor Samsa woke from trouble dreams, he found himself transformed in his bed into a horrible vermin. He lay on his armour-like back, and if he lifted his head a little he could see his brown belly beneath the domed and di
One morning, when Gregor Samsa woke from trouble
d dreams, he found himself
transformed in his bed
to a horrible vermin
He lay on his arm-
like back, and if he lifted
ed his head a little he
could see his brown bell
y, slightly domed and di
pub(crate) struct Record<'n> {
    ntfs: &'n Ntfs,
    data: Vec<u8>,
    position: u64,
}
B-TREE INDEXES
AN NTFS B-TREE

```
  4  9  END
  subnode subnode subnode

  1  3  END
  subnode subnode END

  2  END

  5  7  END
  subnode subnode END

  6  END

  10 END
```
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Offset</td>
<td>Data Offset</td>
</tr>
<tr>
<td>Data Length</td>
<td>Data Length</td>
</tr>
<tr>
<td>Padding</td>
<td>Padding</td>
</tr>
<tr>
<td>Index Entry Length</td>
<td>Index Entry Length</td>
</tr>
<tr>
<td>Key Length</td>
<td>Key Length</td>
</tr>
<tr>
<td>Flags</td>
<td>Flags</td>
</tr>
<tr>
<td>Key</td>
<td>Key</td>
</tr>
<tr>
<td>Data</td>
<td>Data</td>
</tr>
<tr>
<td>Subnode</td>
<td>Subnode</td>
</tr>
<tr>
<td>Field</td>
<td></td>
</tr>
<tr>
<td>-----------------------</td>
<td></td>
</tr>
<tr>
<td>File Reference</td>
<td></td>
</tr>
<tr>
<td>Index Entry Length</td>
<td></td>
</tr>
<tr>
<td>Key Length</td>
<td></td>
</tr>
<tr>
<td>Flags</td>
<td></td>
</tr>
<tr>
<td>Key</td>
<td></td>
</tr>
<tr>
<td>Subnode</td>
<td></td>
</tr>
</tbody>
</table>
trait NtfsIndexEntryType

trait NtfsIndexEntryHasData: NtfsIndexEntryType
trait NtfsIndexEntryHasFileReference: NtfsIndexEntryType
trait NtfsIndexEntryType {
    type KeyType: NtfsIndexEntryKey
}

trait NtfsIndexEntryHasData: NtfsIndexEntryType {
    type DataType: NtfsIndexEntryData
}

trait NtfsIndexEntryHasFileReference: NtfsIndexEntryType

trait NtfsIndexEntryKey
trait NtfsIndexEntryData
struct NtfsFileName {
    header: FileNameHeader,
    name: ArrayVec<u8, NAME_MAX_SIZE>,
}

impl NtfsIndexEntryKey for NtfsFileName

struct NtfsFileNameIndex

impl NtfsIndexEntryType for NtfsFileNameIndex {
    type KeyType = NtfsFileName
}

impl NtfsIndexEntryHasFileReference for NtfsFileNameIndex
impl<'s, E> NtfsIndexEntry<'s, E> where
    E: NtfsIndexEntryType,
{
    fn key(&self) -> Option<Result<E::KeyType>> { ... }

    fn file_reference(&self) -> NtfsFileReference
where
    E: NtfsIndexEntryHasFileReference,
    { ... }
WHAT’S MISSING?

• Caching

• Write support

• Features
  • Compression
  • Journaling
  • Reparse Points
  • Security Descriptors
Thank you for your attention!

Github: [ColinFinck](https://github.com/ColinFinck/ntfs)
Email: colin@reactos.org
Twitter: [ColinFinck](https://twitter.com/ColinFinck)

Shoutout to the Linux-NTFS Documentation at [https://flatcap.github.io/linux-ntfs/ntfs](https://flatcap.github.io/linux-ntfs/ntfs)