Calling "C" FFI with variable sized structure

help

rminsk July 17, 2025, 12:09am 1

windows-rs has the following struct

```
#[repr(C)]
#[derive(Clone, Copy, Debug, PartialEq)]
pub struct TOKEN_PRIVILEGES {
   pub PrivilegeCount: u32,
   pub Privileges: [LUID_AND_ATTRIBUTES; 1]
}
```

I want to call the following function with newstate having three privileges

```
pub unsafe fn AdjustTokenPrivileges(tokenhandle: HANDLE, disableallprivileges: bool, newstate: Option<*const TOKEN_PRIVILEGES>
```

How do I create a TOKEN_PRIVILEGES with more than one Privileges (LUID_AND_ATTRIBUTES).

```
let token_privileges = TOKEN_PRIVILEGES {
   PrivilegeCount: 3,
   Privileges: /* Need to create a "C" style array of 3 elements of type LUID_AND_ATTRIBUTES */
}
```

Do I need to create a `Vec<LUID_AND_ATTRIBUTES>` and somehow get a pointer to the contents? Maybe into_boxed_slice? How about for an arbitrary number of elements?

nerditation July 17, 2025, 12:56am 2

the bindings in `windows-rs` for structs with flex array member (FAM for short) is basically unusable in rust as is.

I suggest to write your own type declaration (or a wrapper type over the `windows-rs` `TOKEN_PRIVILEGES`):

```
const N: usize = 3;
// custom definition:
#[repr(C)]
#[derive(Clone, Copy, Debug, PartialEq)]
pub struct TOKEN_PRIVILEGES {
   pub PrivilegeCount: u32,
   pub Privileges: [LUID_AND_ATTRIBUTES; N]
}

// a wrapper
#[repr(C)]
#[derive(Clone, Copy, Debug, PartialEq)]
pub struct MyTokenPrivileges {
   pub header: TOKEN_PRIVILEGES,
   pub extra_privileges: [LUID_AND_ATTRIBUTES: N - 1]
}
```

see also my reply earlier about `bindgen` and FAM:

Memory allocation and writing to incomplete array field generated by bindgen

> let me try to give you a short summary. the definition for a struct with FAM consists of the fixed-sized "header"/"prefix" part of the packet, and the "payload" part as an array with either an unspecified size, or zero size. if you don't use the flexarray_dst option, then the type definition generated by bindgen represents ONLY the header, which means when you use this type alone, e.g. an owned (unboxed) value, or a (safe) reference, it is always UB to access the payload data array through i…

nerditation July 17, 2025, 1:19am 3

btw, you can work around the problem without creating a wrapper, e.g. you can just "emulate" C code with raw pointers in `unsafe` rust.

ultimately, it's no different than creating a wrapper type, it's just you do all the pointer arithemtic manually (and `unsafe`-ly) instead of letting the compiler do it for you.

below is a sketch using the `alloc` API

```
// the data
let privileges = todo!("initialize the privilage array");
```

```
let n = privileges.len();

// allocate memory to be used as arguments for the API
let header_layout = Layout::new::<TOKEN_PRIVILEGES>();
let extra_layout = Layout::array::<LUID_AND_ATTRIBUTES>(n-1).unwrap();
let (layout, _) = header_layout.extend(extra_layout).unwrap();
let layout = layout.pad_to_align();
let ptr = unsafe { alloc(layout) } as *mut TOKEN_PRIVILEGES;
assert!(!ptr.is_null());

// initialize the allocated struct
unsafe {
   (*ptr).PrivilegeCount = n as u32;
   std::ptr::copy_non_overlapping(privileges.as_ptr(), &raw mut (*ptr).Privileges, n);
}

// call the API
let result = unsafe {
   AdjustTokenPrivileges(
       ...,
       Some(ptr),
       ...,
   )
};

// free the memory
unsafe {
   dealloc(ptr, layout);
}
```

1 Like

rminsk July 17, 2025, 1:34am 4

Reading the Layout docs. Why do I not need `pad_to_align` on the layout?

nerditation July 17, 2025, 1:38am 5

oops, yes, you need to call `pad_to_align()`. I forgot about it.

rminsk July 17, 2025, 1:59am 6

If I call a function that returns a FAM it seems much easier.

```
let mut bytes_required = 0;
_ = GetTokenInformation(token, TokenPrivileges, None, 0, &mut bytes_required);

let buffer = Owned::new(LocalAlloc(LPTR, bytes_required as usize)?);

GetTokenInformation(
   token,
   TokenPrivileges,
   Some(buffer.0 as *mut _),
   bytes_required,
   &mut bytes_required,
)?;

let header = &*(buffer.0 as *const TOKEN_PRIVILEGES);

let privileges =  std::slice::from_raw_parts(header.Privileges.as_ptr(), header.PrivilegeCount as usize);
```

nerditation July 17, 2025, 2:05am 7

rminsk:

> If I call a function that returns a FAM it seems much easier.

that is correct. for FAM structs, ffi -> rust is relatively straight forward, while rust -> ffi is complicated.

creating a FAM in rust needs to deal with allocation, pointer arithmetics, uninitialized memory, etc, but **accessing** a FAM in rust only needs dereferencing raw pointers.

1 Like

system Closed October 15, 2025, 2:06am 8

This topic was automatically closed 90 days after the last reply. We invite you to open a new topic if you have further questions or comments.

Related topics

Topic		Replies	Views	Activity
Call C function with an array of struct as parameter help	5	955	November 5, 2020	
How can I solve it? something may error when I pass a struct to a FFI function help	4	195	August 19, 2024	
Passing vector of vectors buffer to C help	13	2494	April 22, 2020	
Memory allocation and writing to incomplete array field generated by bindgen	12	352	August 24, 2025	
Return C struct (pointer) from ffi help	4	4169	February 24, 2020	