Path forward for the C-API

We move slowly and don’t break things

The NumPy C-API barely changes!

- NumPy 2.0 will be compatible with Python 3.9 or 3.10 (but we can choose 3.9).
- Oldest NumPy compatible with Python 3.9 is NumPy 1.19!
- Our additions since 1.19:
  - `'PyDataMem_SetHandler':                    (304,),`
  - `'PyDataMem_GetHandler':                    (305,),`
  - (A bit more in PRs and 1-2 struct fields)

Otherwise, only things guarded as internal build were changed!

A Brave New World (Part I): *C-API UX improvement*

I propose the following changes to our C UX experience. These can already be part of NumPy 1.25!

- Allow users to specify `NUMPY_TARGET_VERSION=0x01013000 (1.19)`
  - Default to oldest! (if you want to use newer API, specify higher)
  - There are always about 7 NumPy versions back
  - Same as Python Limited API Setup! (and they have 10 versions already)

- Mark API functions with `/*NUMPY_API 1.22`
  - #if guards functionality that is newer.
  - Manually add the guard for struct fields

- Huge advantage: Retire “oldest-supported-numpy”

C-API UX changes – Details

- If a symbol is not defined for the older target version, given error:

```
<source>:5:17: error: 'Old_NumPy_target' undeclared (first use in this function)
5 | #define symbol (Old_NumPy_target: see
https://numpy.org/devdoc/howto_build#MinimumVersion)
 <source>:9:5: note: in expansion of macro 'symbol'
```

  - *By using:* `#define Symbol (Old_NumPy_target: see https://numpy.org/devdoc/howto_build#MinimumVersion)`

- Missing fields cannot contain hint, but should OK to track down.

This means that user can always use our newest headers!

Premise for a major release (NumPy 2.0)

- It is OK to expect downstream to recompile.
  - The recompiled version must work on both 1.x and 2.x!
  - Most libraries should be able to so this in bug-fix release
  - Yes, this will slow down adoption a bit.

- We *can* get away with asking users to do simple code changes
  - (Based on typical assessment, the fewer users it affects the more is OK)
  - We can definitely rely on things that are part of the “deprecated API”

Premise for a major release (NumPy 2.0)

- It is OK to expect downstream to recompile.
    - The recompiled version must work on both 1.x and 2.x!
    - Most libraries should be able to so this in bug-fix release
    - Yes, this will slow down adoption a bit.

- We *can* get away with asking users to do simple code changes
    - (Based on typical assessment, the fewer users it affects the more is OK)
    - We can definitely rely on things that are part of the “deprecated API”

- Users are guaranteed be using our 2.0+ headers:
    - **ABI is irrelevant** as long as we can write a compatibility shim in a header
    - Struct layout changes are an **API** break: Users have to convert to a macro.
- Larger changes may be awkward in headers (e.g. changing our API table)
    - I *propose* a **`numpy2_compat`** package: SciPy compiled against 2.0 will ask you to install it if run against 1.x

Example: Possibly change `PyArray_Descr`

Projects do not use dtype/descriptors directly much. Can we make the struct (mostly) opaque or change it?

- Main relevant field: **`descr->elsize`**
    - It would be nice if this was change from "**`int`**" to "**`ssize_t`**"
    - (not used a lot because `PyArray_ITEMSIZE` is more common)
    - Force you to change it to `PyDataType_ITEMSIZE(descr)`
    - ***`Done:`*** by forcing a few projects to change the above we can freely change struct layout!
    - Both pandas and SciPy have very few, very simple uses.

- A dozen users create extension dtypes, but this is no worse:
    - We change struct to: `PyArray_DescrProto`
    - The real instance will be created by NumPy (user adds one line to get it)
    - ***`Done:`*** NumPy headers can easily take care of everything else.
    - (This API has to go away, but besides a warning that can wait I think)

What about Cython?

- Users compile with NumPy 2, so we know they are using the newest `.pxd` file

- Hope we can rely on Cython 3!
    - Cython is even *less* of a problem!
    - Thanks to Matti, struct field access can be converted macro already in Cython 3 in the `.pxd`

- Caveats:
    - new classes that do not exist in old versions will probably have to be in a different `.pxd` file (The reason is that Cython will insist on initializing them probably and that part would fail)
    - NumPy ships pxd files since 1.19, but oldest-supported-numpy still is 1.17, so cython-supplied files are still used.

How brave are we?

Before I give an impossibly large list of potentially slashed/changed API:

- We can change ABI freely by asking users to adapt to small **API** changes
- This does require asking forcing downstream to recompile
- Useful to ask users to install a **`numpy2_compat`** if running 1.x
- Otherwise: I believe the new setup is better for *everyone*
  - (i.e. no more "oldest support numpy")

I would like to be brave in **major** releases:

- Hiding the descriptor struct will open up some nice things (like more flag space, no integer limit).
- Other changes are not a gamble (e.g. bumping **`NPY_MAXDIMS`**)
- … many things I didn't even think of yet.

This is the way…

… and here is a much too long list of thoughts on what to change (where i will continue less structured:

https://hackmd.io/6XFJ_VnxRtitw2Wy-VXiJg