the rwobj package

Paolo De Donato

June 28, 2022

Contents

1 Introduction

The package rwobj introduces a new mechanism to create objects like the well known C structures. Usually an object in programming languages can be seen as a collection of variables (organized in different ways depending on the chosen language) treated as part of a single entity. Also in rwobj objects are collections of variables, called member variables, which can be retrieved from a string representing that object. Such string is the *address* of the object and act like the address of a structure in C.

An address is composed of two parts, the *module* in which variables are created and an *identifier* that identify uniquely the object inside its module. It's up to the caller that two different objects have different identifiers. The address of an object can be obtained with the `\object_address` function. Identifiers and module names should not contain numbers, `#` and `_` characters in order to avoid conflicts with automatically generated addresses.

In C each object/structure has a *type* that tells the compiler how each object should be organized and instantiated in the memory. So if you need to create objects with the same structure you should first create a new `struct` entity and then create object with such type.

In rwobj objects are created from an existing object with a particular structure that holds all the needed informations to organize their variables. Such objects that can be used to instantiate new objects are calles *proxies* and the proxy object used to instantiate an object is its *generator*. In order to create new objects with a specified proxy you can use the `\object_create` functions.

Since proxies are themself objects we need a proxy to instantiate user defined proxies, you can use the `proxy` object in the rwobj module to create you own proxy, which address is held by the `\c_proxy_address_str` variable. Proxies must be created from the `proxy` object otherwise they won't be recognized as proxies. Instead of using `\object_create` to create proxies you can directly use the function `\proxy_create`.

Once you've created you proxy object you should specify its member variables that will be created in each object initialized with such proxy. You can add a variable specification with the `\proxy_push_member` function. Once you've added all yor variables specifications you can use your proxy to create objects. You should never modify a proxy once you've used it to create at least one object, since these modifications won't be updated on already created objects, leading to hidden errors in subsequential code.

When you create a new variable specification with the `\proxy_push_member` you can notice the presence of ⟨*type*⟩ parameter. It represents the type of such variable and can be a standard type (like `tl`, `str`, `int`, `seq`, ...) or user defined types if the following functions are defined:

`\`⟨*type*⟩`_new:N` and `c` variant;

`\`⟨*type*⟩`_set_eq:NN` and `cN`, `Nc`, `cc` variants.

Every object, and so proxies too, is characterized by the following parameters:

- the *module* in which it has been created;
- the address of the proxy generator;
- a parameter saying if the object is *local* or *global*;
- a parameter saying if the object is *public* or *private*;
- zero or more member variables.

In a local/global/public/private object every member variable is declared local/global/public/private. Address of a member variable can be obtained with the `\object_member_adr` function, and you can instantiate new members that haven't been specified in its generator with the function `\object_new_member`. members created in this way aren't described by generator proxy, so its type can't be deduced and should be always specified in functions like `\object_member_adr` or `\object_member_use`.

2 Library functions

`\object_address:nn` ⋆

`\object_address:nn {`⟨*module*⟩`} {`⟨*id*⟩`}`

Expands to the object address.

`\object_if_exist_p:n` ⋆
`\object_if_exist_p:V` ⋆
`\object_if_exist:nTF` ⋆
`\object_if_exist:VTF` ⋆

`\object_if_exist_p:n {`⟨*address*⟩`}`
`\object_if_exist:nTF {`⟨*address*⟩`} {`⟨*true code*⟩`} {`⟨*false code*⟩`}`

Tests if exists an object at the specified address.

`\object_get_module:n` ⋆
`\object_get_module:V` ⋆
`\object_get_proxy_adr:n` ⋆
`\object_get_proxy_adr:V` ⋆

`\object_get_module:n {`⟨*address*⟩`}`
`\object_get_proxy_adr:n {`⟨*address*⟩`}`

Get the module and the generator proxy of specified object.

\object_if_local_p:n ⋆
\object_if_local_p:V ⋆
\object_if_local:nTF ⋆
\object_if_local:VTF ⋆
\object_if_global_p:n ⋆
\object_if_global_p:V ⋆
\object_if_global:nTF ⋆
\object_if_global:VTF ⋆

\object_if_local_p:n {⟨address⟩}
\object_if_local:nTF {⟨address⟩} {⟨true code⟩} {⟨false code⟩}

Tests if the object is local or global.

\object_if_public_p:n ⋆
\object_if_public_p:V ⋆
\object_if_public:nTF ⋆
\object_if_public:VTF ⋆
\object_if_private_p:n ⋆
\object_if_private_p:V ⋆
\object_if_private:nTF ⋆
\object_if_private:VTF ⋆

\object_if_local_p:n {⟨address⟩}
\object_if_local:nTF {⟨address⟩} {⟨true code⟩} {⟨false code⟩}

Tests if the object is public or private.

\object_member_adr:nnn ⋆
\object_member_adr:(Vnn|nnv) ⋆
\object_member_adr:nn ⋆
\object_member_adr:Vn ⋆

\object_member_adr:nnn {⟨address⟩} {⟨member name⟩} {⟨member type⟩}
\object_member_adr:nn {⟨address⟩} {⟨member name⟩}

Fully expands to the address of specified member variable. If type is not specified it'll be retrieved from the generator proxy, but only if member is specified in the generator.

\object_member_type:nn ⋆
\object_member_type:Vn ⋆

\object_member_type:nn {⟨address⟩} {⟨member name⟩}

Fully expands to the type of member ⟨*member name*⟩. Use this function only with member variables specified in the generator proxy, not with other member variables.

\object_new_member:nnn
\object_new_member:(Vnn|nnv)

\object_new_member:nnn {⟨address⟩} {⟨member name⟩} {⟨member type⟩}

Creates a new member variable with specified name and type. You can't retrieve the type of these variables with \object_member_type functions.

\object_member_use:nnn ⋆
\object_member_use:(Vnn|nnv) ⋆
\object_member_use:nn ⋆
\object_member_use:Vn ⋆

\object_member_use:nnn {⟨address⟩} {⟨member name⟩} {⟨member type⟩}
\object_member_use:nn {⟨address⟩} {⟨member name⟩}

Uses the specified member variable.

\object_member_set_eq:nnnN ⋆
\object_member_set_eq:(nnvN|VnnN|nnnc|Vnnc) ⋆
\object_member_set_eq:nnN ⋆
\object_member_set_eq:(VnN|nnc|Vnc) ⋆

\object_member_set_eq:nnnN {⟨address⟩} {⟨member name⟩}
{⟨member type⟩} ⟨variable⟩
\object_member_set_eq:nnN {⟨address⟩} {⟨member name⟩}
⟨variable⟩

Sets the value of specified member equal to the value of ⟨*variable*⟩.

`\object_if_proxy_p:n` ⋆
`\object_if_proxy_p:V` ⋆
`\object_if_proxy:nTF` ⋆
`\object_if_proxy:VTF` ⋆

`\object_if_proxy_p:n {⟨address⟩}`
`\object_if_proxy:nTF {⟨address⟩} {⟨true code⟩} {⟨false code⟩}`

Test if the specified object is a proxy object.

`\c_proxy_address_str`

The address of the `proxy` object in the rwobj module.

`\object_create:nnnNN`
`\object_create:VnnNN`

`\object_create:nnnNN {⟨proxy address⟩} {⟨module⟩} {⟨id⟩} ⟨scope⟩ ⟨visibility⟩`

Creates an object by using the proxy at ⟨*proxy address*⟩ and the specified parameters.

`\c_object_local_str`
`\c_object_global_str`

Possible values for ⟨*scope*⟩ parameter.

`\c_object_public_str`
`\c_object_private_str`

Possible values for ⟨*visibility*⟩ parameter.

`\object_create_set:NnnnNN`
`\object_create_set:NVnnNN`
`\object_create_gset:NnnnNN`
`\object_create_gset:NVnnNN`

`\object_create_set:NnnnNN ⟨str var⟩ {⟨proxy address⟩} {⟨module⟩} {⟨id⟩} ⟨scope⟩ ⟨visibility⟩`

Creates an object and sets its fully expanded address inside ⟨*str var*⟩.

`\proxy_create:nnN`
`\proxy_create_set:NnnN`
`\proxy_create_gset:NnnN`

`\proxy_create:nnN {⟨module⟩} {⟨id⟩} ⟨visibility⟩`
`\proxy_create_set:NnnN ⟨str var⟩ {⟨module⟩} {⟨id⟩} ⟨visibility⟩`

Creates a global proxy object.

`\proxy_push_member:nnn`
`\proxy_push_member:Vnn`

`\proxy_push_member:nnn {⟨proxy address⟩} {⟨ member name ⟩} {⟨ member type ⟩}`

Updates a proxy object with a new member specification, so that every subsequential object created with this proxy will have a member variable with the specified name and type that can be retrieved with `\object_member_type` functions.

`\object_assign:nn`
`\object_assign:(Vn|nV|VV)`

`\object_assign:nn {⟨to address⟩} {⟨from address⟩}`

Assigns the content of each variable of object at ⟨*from address*⟩ to each correspective variable in ⟨*to address*⟩. Both the objects should be created with the same proxy object and only variables listed in the proxy are assigned.

3 Implementation

```
1 ⟨*package⟩
2 ⟨@@=objpriv⟩
```

`\c_object_local_str`
`\c_object_global_str`
`\c_object_public_str`
`\c_object_private_str`

```
3 \str_const:Nn \c_object_local_str {loc}
4 \str_const:Nn \c_object_global_str {glo}
5 \str_const:Nn \c_object_public_str {pub}
```

```
\str_const:Nn \c_object_private_str {pri}

\str_const:Nn \c__objpriv_const_str {con}
```

(*End definition for* `\c_object_local_str` *and others. These variables are documented on page 4.*)

`\object_address:nn` Get address of an object

```
\cs_new:Nn \object_address:nn {
  \tl_to_str:n { #1  _  #2 }
}
```

(*End definition for* `\object_address:nn`*. This function is documented on page 2.*)

```
\cs_new:Nn \__objpriv_object_modvar:n{
  c __ #1 _ MODULE _ str
}

\cs_new:Nn \__objpriv_object_pxyvar:n{
  c __ #1 _ PROXY _ str
}

\cs_new:Nn \__objpriv_object_scovar:n{
  c __ #1 _ SCOPE _ str
}

\cs_new:Nn \__objpriv_object_visvar:n{
  c __ #1 _ VISIB _ str
}

\cs_generate_variant:Nn \__objpriv_object_modvar:n { V }
\cs_generate_variant:Nn \__objpriv_object_pxyvar:n { V }
\cs_generate_variant:Nn \__objpriv_object_scovar:n { V }
\cs_generate_variant:Nn \__objpriv_object_visvar:n { V }
```

`\object_if_exist_p:n`
`\object_if_exist:nTF` Tests if object exists.

```

\prg_new_conditional:Nnn \object_if_exist:n { p, T, F, TF }
  {
    \cs_if_exist:cTF
      {
        \__objpriv_object_modvar:n { #1 }
      }
      {
        \prg_return_true:
      }
      {
        \prg_return_false:
      }
  }

\prg_generate_conditional_variant:Nnn \object_if_exist:n { V }
  { p, T, F, TF }

```

(*End definition for* `\object_if_exist:nTF`*. This function is documented on page 2.*)

\object_get_module:n
\object_get_proxy_adr:n

Retrieve the name, module and generating proxy of an object

```
\cs_new:Nn \object_get_module:n {
  \str_use:c { \__objpriv_object_modvar:n { #1 } }
}
\cs_new:Nn \object_get_proxy_adr:n {
  \str_use:c { \__objpriv_object_pxyvar:n { #1 } }
}

\cs_generate_variant:Nn \object_get_module:n { V }
\cs_generate_variant:Nn \object_get_proxy_adr:n { V }
```

(*End definition for* `\object_get_module:n` *and* `\object_get_proxy_adr:n`. *These functions are documented on page* 2.)

\object_if_local_p:n
\object_if_local:nTF
\object_if_global_p:n
\object_if_global:nTF
\object_if_public_p:n
\object_if_public:nTF
\object_if_private_p:n
\object_if_private:nTF

Test the specified parameters.

```
\prg_new_conditional:Nnn \object_if_local:n {p, T, F, TF}
{
  \str_if_eq:cNTF { \__objpriv_object_scovar:n {#1} } \c_object_local_str
  {
    \prg_return_true:
  }
  {
    \prg_return_false:
  }
}

\prg_new_conditional:Nnn \object_if_global:n {p, T, F, TF}
{
  \str_if_eq:cNTF { \__objpriv_object_scovar:n {#1} } \c_object_global_str
  {
    \prg_return_true:
  }
  {
    \prg_return_false:
  }
}

\prg_new_conditional:Nnn \object_if_public:n {p, T, F, TF}
{
  \str_if_eq:cNTF { \__objpriv_object_visvar:n { #1 } } \c_object_public_str
  {
    \prg_return_true:
  }
  {
    \prg_return_false:
  }
}

\prg_new_conditional:Nnn \object_if_private:n {p, T, F, TF}
{
  \str_if_eq:cNTF { \__objpriv_object_visvar:n {#1} } \c_object_private_str
  {
    \prg_return_true:
  }
```

```
  {
    \prg_return_false:
  }
}

\prg_generate_conditional_variant:Nnn \object_if_local:n { V }
  { p, T, F, TF }
\prg_generate_conditional_variant:Nnn \object_if_global:n { V }
  { p, T, F, TF }
\prg_generate_conditional_variant:Nnn \object_if_public:n { V }
  { p, T, F, TF }
\prg_generate_conditional_variant:Nnn \object_if_private:n { V }
  { p, T, F, TF }
```

(*End definition for* `\object_if_local:nTF` *and others. These functions are documented on page* 3.)

You can retrieve the address of a member variable with the following function:

`\object_member_adr:nnn`
`\object_member_adr:nn`

Get the address of a member variable

```

\cs_new:Nn \__objpriv_scope:n
  {
    \object_if_global:nTF { #1 }
      {
        g
      }
      {
        \str_if_eq:cNTF { \__objpriv_object_scovar:n { #1 } }
          \c__objpriv_const_str
          {
            c
          }
          {
            l
          }
      }
  }

\cs_new:Nn \object_member_adr:nnn
  {
    \__objpriv_scope:n { #1 }
    \object_if_private:nTF { #1 }
      {
        __
      }
      {
        _
      }
    #1 \tl_to_str:n { _ MEMBER _ #2 _ #3 }
  }

\cs_generate_variant:Nn \object_member_adr:nnn { Vnn, vnn, nnv }

\cs_new:Nn \object_member_adr:nn
  {
```

```
    \object_member_adr:nnv { #1 }{ #2 }
      {
        \object_member_adr:vnn { \__objpriv_object_pxyvar:n { #1 } }
          { #2 _ type }{ str }
      }
  }

\cs_generate_variant:Nn \object_member_adr:nn { Vn }
```

(*End definition for* `\object_member_adr:nnn` *and* `\object_member_adr:nn`*. These functions are documented on page 3.*)

`\object_member_type:nn` Deduce the member type from the generating proxy.

```
\cs_new:Nn \object_member_type:nn
  {
    \object_member_use:vnn { \__objpriv_object_pxyvar:n { #1 } }
      { #2 _ type }{ str }
  }

```

(*End definition for* `\object_member_type:nn`*. This function is documented on page 3.*)

```
\msg_new:nnnn { rwobj }{ scoperr }{ Nonstandard ~ scope }
  {
    Operation ~ not ~ permitted ~ on ~ object ~ #1 ~
    ~ since ~ it ~ wasn't ~ declared ~ local ~ or ~ global
  }

\cs_new_protected:Nn \__objpriv_force_scope:n
  {
    \bool_if:nF
      {
        \object_if_local_p:n { #1 } || \object_if_global_p:n { #1 }
      }
      {
        \msg_error:nnx { rwobj }{ scoperr }{ #1 }
      }
  }

```

`\object_new_member:nnn` Creates a new member variable

```
\cs_new_protected:Nn \object_new_member:nnn
  {
    \__objpriv_force_scope:n { #1 }
    \cs_if_exist_use:cT { #3 _ new:c }
      {
        { \object_member_adr:nnn { #1 }{ #2 }{ #3 } }
      }
  }

\cs_generate_variant:Nn \object_new_member:nnn { Vnn, nnv }

```

(*End definition for* `\object_new_member:nnn`. *This function is documented on page 3.*)

`\object_member_use:nnn` `\object_member_use:nn`

Uses a member variable

```
\cs_new:Nn \object_member_use:nnn
  {
    \cs_if_exist_use:cT { #3 _ use:c }
      {
        { \object_member_adr:nnn { #1 }{ #2 }{ #3 } }
      }
  }

\cs_new:Nn \object_member_use:nn
  {
    \object_member_use:nnv { #1 }{ #2 }
      {
        \object_member_adr:vnn { \__objpriv_object_pxyvar:n { #1 } }
          { #2 _ type }{ str }
      }
  }

\cs_generate_variant:Nn \object_member_use:nnn { Vnn, vnn, nnv }
\cs_generate_variant:Nn \object_member_use:nn { Vn }
```

(*End definition for* `\object_member_use:nnn` *and* `\object_member_use:nn`. *These functions are documented on page 3.*)

`\object_member_set_eq:nnnN` `\object_member_set_eq:nnN`

Set the value of a variable to a member.

```
\cs_new_protected:Nn \object_member_set_eq:nnnN
  {
    \__objpriv_force_scope:n { #1 }
    \cs_if_exist_use:cT
      {
        #3 _ \object_if_global:nT { #1 }{ g } set _ eq:cN
      }
      {
        { \object_member_adr:nnn { #1 }{ #2 }{ #3 } } #4
      }
  }

\cs_generate_variant:Nn \object_member_set_eq:nnnN { VnnN, nnnc, Vnnc, nnvN }

\cs_new_protected:Nn \object_member_set_eq:nnN
  {
    \object_member_set_eq:nnvN { #1 }{ #2 }
      {
        \object_member_adr:vnn { \__objpriv_object_pxyvar:n { #1 } }
          { #2 _ type }{ str }
      } #3
  }

\cs_generate_variant:Nn \object_member_set_eq:nnN { VnN, nnc, Vnc }
```

(*End definition for* `\object_member_set_eq:nnnN` *and* `\object_member_set_eq:nnN`. *These functions are documented on page 3.*)

`\c_proxy_address_str` The address of the `proxy` object.

```
\str_const:Nx \c_proxy_address_str
  { \object_address:nn { rwobj }{ proxy } }
```

(*End definition for* `\c_proxy_address_str`. *This variable is documented on page 4.*)

Source of `proxy` object

```
\str_const:cn { \__objpriv_object_modvar:V \c_proxy_address_str }
  { rwobj }
\str_const:cV { \__objpriv_object_pxyvar:V \c_proxy_address_str }
  \c_proxy_address_str
\str_const:cV { \__objpriv_object_scovar:V \c_proxy_address_str }
  \c__objpriv_const_str
\str_const:cV { \__objpriv_object_visvar:V \c_proxy_address_str }
  \c_object_public_str

\cs_generate_variant:Nn \seq_const_from_clist:Nn { cx }

\seq_const_from_clist:cn
  {
    \object_member_adr:Vnn \c_proxy_address_str { varlist }{ seq }
  }
  { varlist }

\str_const:cn
  {
    \object_member_adr:Vnn \c_proxy_address_str { varlist_type }{ str }
  }
  { seq }
```

`\object_if_proxy_p:n` Test if an object is a proxy.
`\object_if_proxy:nTF`

```

\prg_new_conditional:Nnn \object_if_proxy:n {p, T, F, TF}
  {
    \str_if_eq:cNTF { \__objpriv_object_pxyvar:n { #1 } } \c_proxy_address_str
    {
      \prg_return_true:
    }
    {
      \prg_return_false:
    }
  }

```

(*End definition for* `\object_if_proxy:nTF`. *This function is documented on page 4.*)

`\object_create:nnnNN` Creates an object from a proxy
`\object_create_set:NnnnNN`
`\object_create_gset:NnnnNN`

```

\msg_new:nnn { aa }{ mess }{ #1 }

\msg_new:nnnn { rwobj }{ notproxy }{ Fake ~ proxy }
  {
```

```
    Object ~ #1 ~ is ~ not ~ a ~ proxy.
  }

\cs_new_protected:Nn \__objpriv_force_proxy:n
  {
    \object_if_proxy:nF { #1 }
      {
        \msg_error:nnn { rwobj }{ notproxy }{ #1 }
      }
  }

\cs_new_protected:Nn \__objpriv_create_anon:nnnNN
  {

    \__objpriv_force_proxy:n { #1 }

    \str_const:cn { \__objpriv_object_modvar:n { #2 } }{ #3 }
    \str_const:cx { \__objpriv_object_pxyvar:n { #2 } }{ #1 }
    \str_const:cV { \__objpriv_object_scovar:n { #2 } } #4
    \str_const:cV { \__objpriv_object_visvar:n { #2 } } #5

    \seq_map_inline:cn
      {
        \object_member_adr:nnn { #1 }{ varlist }{ seq }
      }
      {
        \object_new_member:nnv { #2 }{ ##1 }
          {
            \object_member_adr:nnn { #1 }{ ##1 _ type }{ str }
          }
      }
  }

\cs_new_protected:Nn \object_create:nnnNN
  {
    \__objpriv_create_anon:nnnNN { #1 }{ \object_address:nn { #2 }{ #3 } }
      { #2 } #4 #5
  }

\cs_new_protected:Nn \object_create_set:NnnnNN
  {
    \object_create:nnnNN { #2 }{ #3 }{ #4 } #5 #6
    \str_set:Nx #1 { \object_address:nn { #3 }{ #4 } }
  }

\cs_new_protected:Nn \object_create_gset:NnnnNN
  {
    \object_create:nnnNN { #2 }{ #3 }{ #4 } #5 #6
    \str_gset:Nx #1 { \object_address:nn { #3 }{ #4 } }
  }

\cs_generate_variant:Nn \object_create:nnnNN { VnnNN }
\cs_generate_variant:Nn \object_create_set:NnnnNN { NVnnNN }
\cs_generate_variant:Nn \object_create_gset:NnnnNN { NVnnNN }
```

```
334
```

(*End definition for* `\object_create:nnnNN`, `\object_create_set:NnnnNN`, *and* `\object_create_gset:NnnnNN`. *These functions are documented on page* 4.)

`\proxy_create:nnN`
`\proxy_create_set:NnnN`
`\proxy_create_gset:NnnN`

Creates a new proxy object

```
335
336 \cs_new_protected:Nn \proxy_create:nnN
337   {
338     \object_create:VnnNN \c_proxy_address_str { #1 }{ #2 }
339       \c_object_global_str #3
340   }
341
342 \cs_new_protected:Nn \proxy_create_set:NnnN
343   {
344     \object_create_set:NVnnNN #1 \c_proxy_address_str { #2 }{ #3 }
345       \c_object_global_str #4
346   }
347
348 \cs_new_protected:Nn \proxy_create_gset:NnnN
349   {
350     \object_create_gset:NVnnNN #1 \c_proxy_address_str { #2 }{ #3 }
351       \c_object_global_str #4
352   }
353
```

(*End definition for* `\proxy_create:nnN`, `\proxy_create_set:NnnN`, *and* `\proxy_create_gset:NnnN`. *These functions are documented on page* 4.)

`\proxy_push_member:nnn`

Push a new member inside a proxy.

```
354 \cs_new_protected:Nn \proxy_push_member:nnn
355   {
356     \__objpriv_force_scope:n { #1 }
357     \object_new_member:nnn { #1 }{ #2 _ type }{ str }
358     \str_set:cn
359       {
360         \object_member_adr:nnn { #1 }{ #2 _ type }{ str }
361       }
362       { #3 }
363     \seq_gput_left:cn
364       {
365         \object_member_adr:nnn { #1 }{ varlist }{ seq }
366       }
367       { #2 }
368   }
369
370 \cs_generate_variant:Nn \proxy_push_member:nnn { Vnn }
371
```

(*End definition for* `\proxy_push_member:nnn`. *This function is documented on page* 4.)

`\object_assign:nn`

Copy an object to another one.

```
372 \cs_new_protected:Nn \object_assign:nn
373   {
374     \seq_map_inline:cn
```

```
      {
        \object_member_adr:vnn
          {
            \__objpriv_object_pxyvar:n { #1 }
          }
          { varlist }{ seq }
      }
      {
        \object_member_set_eq:nnc { #1 }{ ##1 }
          {
            \object_member_adr:nn{ #2 }{ ##1 }
          }
      }
  }

\cs_generate_variant:Nn \object_assign:nn { nV, Vn, VV }
```

(*End definition for* `\object_assign:nn`. *This function is documented on page* 4.)

⟨/package⟩