The \texttt{lt3rawobjects} package

Paolo De Donato

Released 2022/06/30 Version 1.0.2

Contents

1 Introduction 1
2 To do 1
3 Objects and proxies 2
4 Library functions 3
5 Examples 5
6 Implementation 5

1 Introduction

First to all notice that \texttt{lt3rawobjects} means “raw object(s)”, indeed \texttt{lt3rawobjects} introduces a new mechanism to create objects like the well known C structures. The functions exported by this package are quite low level, and many important mechanisms like member protection and name resolution aren’t already defined and should be introduced by intermediate packages.

2 To do

- Introduce member functions in objects and member function specifications in proxies;
- Uniform declarations for templated proxies;
- Introduce constant objects.
3 Objects and proxies

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 lt3rawobjects 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 \texttt{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 \texttt{struct} entity and then create object with such type.

In lt3rawobjects 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 called 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 \texttt{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 \texttt{lt3rawobjects} module to create you own proxy, which address is held by the \texttt{c_proxy_address_str} variable. Proxies must be created from the proxy object otherwise they won’t be recognized as proxies. Instead of using \texttt{object_create} to create proxies you can directly use the function \texttt{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 \texttt{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 subsequental code.

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

\texttt{\langle type \rangle\_new:N and c variant;}
\texttt{\langle type \rangle\_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`.

## 4 Library functions

```
\object_address:nn \{module\} \{(id)\}
```
Expands to the object address.

```
\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 \{address\}
\object_get_proxy_adr:n \{address\}
```
Get the module and the generator proxy of specified object.

```
\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 \{address\}
\object_if_public:nTF \{address\} \{true code\} \{false code\}
```
Tests if the object is public or private.

```
\object_member_adr:nn \{address\} \{member name\} \{member type\}
```
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.
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>\object_member_type:nn</code></td>
<td><code>\object_member_type:Vn</code></td>
</tr>
<tr>
<td>Fully expands to the type of member <code>⟨member name⟩</code>. Use this function only with member variables specified in the generator proxy, not with other member variables.</td>
<td></td>
</tr>
<tr>
<td><code>\object_new_member:nn</code></td>
<td><code>\object_new_member:nnn (⟨address⟩) {⟨member name⟩} {⟨member type⟩}</code></td>
</tr>
<tr>
<td>Creates a new member variable with specified name and type. You can’t retrieve the type of these variables with <code>\object_member_type</code> functions.</td>
<td></td>
</tr>
<tr>
<td><code>\object_member_use:nn</code></td>
<td><code>\object_member_use:Vn</code></td>
</tr>
<tr>
<td>Uses the specified member variable.</td>
<td></td>
</tr>
<tr>
<td><code>\object_member_set_eq:nnnN</code></td>
<td><code>\object_member_set_eq:nnN (⟨address⟩) {⟨member name⟩} {⟨member type⟩} ⟨variable⟩</code></td>
</tr>
<tr>
<td>Sets the value of specified member equal to the value of <code>⟨variable⟩</code>.</td>
<td></td>
</tr>
<tr>
<td><code>\object_if_proxy_p:n</code></td>
<td><code>\object_if_proxy:nTF (⟨address⟩) {⟨true code⟩} {⟨false code⟩}</code></td>
</tr>
<tr>
<td>Test if the specified object is a proxy object.</td>
<td></td>
</tr>
<tr>
<td><code>\c_proxy_address_str</code></td>
<td>The address of the proxy object in the lt3rawobjects module.</td>
</tr>
<tr>
<td><code>\object_create:nnnNN</code></td>
<td><code>\object_create:nnnNN {⟨proxy address⟩} {⟨module⟩} {⟨id⟩} {⟨scope⟩} {⟨visibility⟩}</code></td>
</tr>
<tr>
<td>Creates an object by using the proxy at <code>⟨proxy address⟩</code> and the specified parameters.</td>
<td></td>
</tr>
<tr>
<td><code>\c_object_local_str</code></td>
<td>Possible values for <code>⟨scope⟩</code> parameter.</td>
</tr>
<tr>
<td><code>\c_object_global_str</code></td>
<td>Possible values for <code>⟨visibility⟩</code> parameter.</td>
</tr>
<tr>
<td><code>\object_create_set:NnnnNN</code></td>
<td><code>\object_create_set:NnnnNN {⟨str var⟩} {⟨proxy address⟩} {⟨module⟩} {⟨id⟩} {⟨scope⟩} {⟨visibility⟩}</code></td>
</tr>
<tr>
<td>Creates an object and sets its fully expanded address inside <code>⟨str var⟩</code>.</td>
<td></td>
</tr>
</tbody>
</table>
\proxy_create:nnN \proxy_create_set:NnnN \proxy_create_gset:NnnN

\proxy_push_member:nnn \proxy_push_member:Vnn

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

5 Examples

Example 1

Create a public proxy with id \texttt{myproxy} with the specification of a single member variable with name \texttt{myvar} and type \texttt{tl}, then set its address inside \texttt{\l_myproxy_str}.

\str_new:N \l_myproxy_str
\proxy_create_set:NnnN \l_myproxy_str \example \myproxy
\c_object_public_str
\proxy_push_member:Vnn \l_myproxy_str \myvar \tl

Then create a new object with name \texttt{myobj} with that proxy, assign then token list \texttt{\c_dollar_str{}} - dollar - \texttt{\c_dollar_str{}} to \texttt{myvar} and then print it.

\str_new:N \l_myobj_str
\object_create_set:NVnnNN \l_myobj_str \l_myproxy_str
\example \myobj \c_object_local_str \c_object_public_str
\tl_set:cn
{ \object_member_adr:Vn \l_myobj_str \myvar }
{ \c_dollar_str{} - dollar - \c_dollar_str{} }
\object_member_use:Vn \l_myobj_str \myvar

Output:
$ dollar $

6 Implementation

\c_object_local_str \c_object_global_str \c_object_public_str \c_object_private_str
\str_const:Nn \c_object_local_str \loc

5
\str_const:Nn \c_object_global_str {glo}
\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:n Get address of an object
\cs_new:Nn \object_address:n { \tl_to_str:n { #1 _ #2 } }

(End definition for \object_address:n. This function is documented on page 3.)

\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 Tests if object exists.
\cs_new:Nn \object_if_exist:n { p, T, F, TF }
{ \cs_if_exist:cTF
  { \__objpriv_object_modvar:n { #1 } }
  \prg_return_true: }
{ \prg_return_false: }
}
\cs_new:Nn \object_if_exist:n { p, T, F, TF }
{ \prg_generate_conditional_variant:Nnn \object_if_exist:n { V } { p, T, F, TF } }

6
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 }

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: }

(End definition for \object_if_exist:nTF. This function is documented on page 3.)
\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 }
{
\str_if_eq:cTF { \__objpriv_object_scovar:n { #1 } }
\c__objpriv_const_str

#1 \tl_to_str:n { MEMBER #2 #3 }
}
\cs_generate_variant:Nn \object_member_adr:nnn { Vnn, vnn, nnv }

8
\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 4.)

\msg_new:nnnn { lt3rawobjects }{ 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 { lt3rawobjects }{ 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 } }

\texttt{\textbackslash cs\_generate\_variant:Nn \textbackslash object\_new\_member:nnn \{ Vnn, nnv \}}

\textit{(End definition for \textbackslash object\_new\_member:nnn. This function is documented on page 4.)}

\textbf{\texttt{\textbackslash object\_member\_use:nnn}}

\textbf{\texttt{\textbackslash object\_member\_use:nnn}}

\texttt{\textbackslash cs\_new:Nn \textbackslash object\_member\_use:nnn}

\texttt{\{}

\texttt{\cs\_if\_exist\_use:cT \{ #3 _ use:c \}}

\texttt{\{}

\texttt{\object\_member\_adr:nnn \{ #1 \}{ #2 \}{ #3 \}}

\texttt{\}}

\texttt{\}}

\texttt{\cs\_new:Nn \textbackslash object\_member\_use:nnn}

\texttt{\{}

\texttt{\object\_member\_use:nnv \{ #1 \}{ #2 \}}

\texttt{\{}

\texttt{\object\_member\_adr:vnn \{ \_\_objpriv\_object\_pxyvar:n \{ #1 \} \}}

\texttt{\{ #2 _ type \}{ str \}}

\texttt{\}}

\texttt{\}}

\texttt{\cs\_generate\_variant:Nn \textbackslash object\_member\_use:nnn \{ Vnn, vnn, nnv \}}

\texttt{\cs\_generate\_variant:Nn \textbackslash object\_member\_use:nnn \{ Vn \}}

\textit{(End definition for \textbackslash object\_member\_use:nnn and \textbackslash object\_member\_use:nn. These functions are documented on page 4.)}

\textbf{\texttt{\textbackslash object\_member\_set\_eq:nnN}}

\textbf{\texttt{\textbackslash object\_member\_set\_eq:nnN}}

\texttt{\cs\_new\_protected:Nn \textbackslash object\_member\_set\_eq:nnnN}

\texttt{\{}

\texttt{\_\_objpriv\_force\_scope:n \{ #1 \}}

\texttt{\cs\_if\_exist\_use:cT}

\texttt{\{}

\texttt{\#3 _ \_objpriv\_set\_eq:v \{ #1 \}{ g \} set _ eq:cN}

\texttt{\}}

\texttt{\}}

\texttt{\object\_member\_adr:n \{ #1 \}{ #2 \}{ #3 \} #4}

\texttt{\}}

\texttt{\cs\_generate\_variant:Nn \textbackslash object\_member\_set\_eq:nnnN \{ VnnN, nnnNc, Vnnc, nnvN \}}

\texttt{\cs\_new\_protected:Nn \textbackslash object\_member\_set\_eq:nnnN}

\texttt{\{}

\texttt{\object\_member\_set\_eq:nnvN \{ #1 \}{ #2 \}}

\texttt{\{}

\texttt{\object\_member\_adr:vnn \{ \_\_objpriv\_object\_pxyvar:n \{ #1 \} \}}

\texttt{\{ #2 _ type \}{ str \}}

\texttt{\}}

\texttt{\}}
\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 4.)

\c_proxy_address_str The address of the proxy object.

\str_const:Nx \c_proxy_address_str
\{ \object_address:nn \{ lt3rawobjects \} \{ 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 \}
\{ lt3rawobjects \}
\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 \object_if_proxy:nTF

Test if an object is a proxy.

\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
\object_create_set:NNNN
\object_create_gset:NNNN

Creates an object from a proxy
\msg_new:nnn { aa }{ mess }{ #1 }
\msg_new:nnn { lt3rawobjects }{ 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 { lt3rawobjects }{ 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 } }

\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 } }

\endinput
\cs_generate_variant:Nn \object_create:nnn { VnnNN } \\
\cs_generate_variant:Nn \object_create_set:Nnnn { NVnnNN } \\
\cs_generate_variant:Nn \object_create_gset:Nnnn { NVnnNN } 

\textit{(End definition for \object_create:nnnNN, \object_create_set:nnnNN, and \object_create_gset:nnnNN. These functions are documented on page 4.)}

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

Creates a new proxy object

\cs_new_protected:Nn \proxy_create:nnN 
{ 
\object_create:nnn \c_proxy_address_str { #1 }{ #2 } \\
\c_object_global_str #3 
}

\textit{(End definition for \proxy_create:nnN, \proxy_create_set:NnnN, and \proxy_create_gset:NnnN. These functions are documented on page 5.)}

\proxy_push_member:nnn

Push a new member inside a proxy.

\cs_new_protected:Nn \proxy_push_member:nnn 
{ 
\__objpriv_force_scope:n { #1 } 
\object_new_member:nnn { #1 }{ #2 _ type }{ str } 
\str_set:cn 
{ 
\object_member_adr:nnn { #1 }{ #2 _ type }{ str } 
} 
{ #3 } 
\seq_gput_left:cn 
{ 
\object_member_adr:nnn { #1 }{ varlist }{ seq } 
} 
{ #2 } 
}

\cs_generate_variant:Nn \proxy_push_member:nnn { Vnn }

\textit{(End definition for \proxy_push_member:nnn. This function is documented on page 5.)}
\texttt{\textbackslash object\_assign:nn} Copy an object to another one.

\begin{verbatim}
\cs_new_protected:Nn \object_assign:nn
{
  \seq_map_inline:cn
  {
    \object_member_adr:vnn
    {
      \__objpriv_object_pxyvar:n { #1 }
    }
    \seq \varlist
  }

  \object_member_set_eq:nnc { #1 }{ ##1 }
  {
    \object_member_adr:nn{ #2 }{ ##1 }
  }
}
\cs_generate_variant:Nn \object_assign:nn { nV, Vn, VV }
\end{verbatim}

(End definition for \texttt{\object_assign:nn}. This function is documented on page 5.)