

Table of Contents

Extending C++ Classes with R functions.	1
The Interface Code	2
The R code	2
The C++ code	2

Extending C++ Classes with R functions.

In this article, we will see how we can create new C++ classes that extend a given C++ class and have some of the new class' methods implemented as R functions. This is quite powerful as it allows us to introduce C++ objects into an existing C++ computation and have our R functions invoked instead of the lower-level C++ methods. This allows us to intercept certain calls and customize behaviour to our needs. This mechanism allows us to more rapid prototyping than in creating these extended classes and their methods in C++ directly. Additionally, we can change the R function that implements a method at run time for a particular instance. Thus we have a very dynamic class mechanism mixed with C++'s static, strongly typed approach.

We will look at a simple example first. It is not entirely compelling as we are attempting to keep things simple. But it does illustrate how we can use some of the non-GUI facilities in wxWidgets from within R. We use the wxDir class to recursively process all the files and directories within that given directory. The Traverse method in this class does this. To call it, we need an implementation of the wxDirTraverser (abstract) class. For each file tha the Traverse method finds, it will call the wxDirTraverser's OnFile method. And similarly, for every directory it encounters, it will call the OnDir method. So we need to provide R functions that implement each of these.

The C++ and R interface code is automatically generated from the definition of wxDir and wxDirTraverser. It is given at the end of this document.

We can now see this in operation.

```
library(RwxWidgets)
wxInit()

d = wxDir(system.file( package = "RwxWidgets"))

tt = RwxDirTraverser(function(this, filename) {
    cat("File:", filename, "\n")
    wxDIR_CONTINUE
},
    function(this, dirname) {
cat("Directory:", dirname, "\n")
    wxDIR_CONTINUE
})

d$Traverse(tt)
```

Because these are pure methods, we omit one important detail, specifically how we call the inherited methods. This arises

The Interface Code

The R code

```
RwxDirTraverser =  
function(OnFile = NULL, OnDir = NULL, OnOpenError = NULL)  
{  
  .Call("R_RwxDirTraverser_new", OnFile, OnDir, OnOpenError)  
}  
  
wxDir =  
function(dir = character())  
{  
  .Call("R_wxDir_new", as.character(dir))  
}  
  
wxDir_Traverse =  
function(this, sink, filespec = "", flags = wxDIR_DEFAULT)  
{  
  flags = bitlist(flags)  
  .Call("R_wxDir_Traverse", this, sink, as.character(filespec), flags)  
}
```

The C++ code

```
#include <wx/wx.h>  
#include <wx/dir.h>  
  
#include "RwxUtils.h"  
  
//XXX had to do this manually as the docs and the code are out of sync.  
typedef wxDirTraverseResult wxOpenErrorTraverseResult;  
  
class RwxDirTraverser : public wxDirTraverser {  
  
public:  
  RwxDirTraverser(SEXP OnFile, SEXP OnDir, SEXP OnOpenError = NULL) {  
    if(OnFile && GET_LENGTH(OnFile) && TYPEOF(OnFile) == CLOXP)  
      R_PreserveObject(R_OnFile_m = Rf_duplicate(OnFile));  
  
    if(OnDir && GET_LENGTH(OnDir) && TYPEOF(OnDir) == CLOXP)
```

```

R_PreserveObject(R_OnDir_m = Rf_duplicate(OnDir));

if(OnOpenError && GET_LENGTH(OnOpenError) && TYPEOF(OnOpenError) == CLOSXP)
  R_PreserveObject(R_OnOpenError_m = Rf_duplicate(OnOpenError));
}

wxDirTraverseResult OnFile(const wxString &filename);
wxDirTraverseResult OnDir(const wxString &filename);
wxOpenErrorTraverseResult OnOpenError(const wxString& openerrorname);

protected:
  SEXP R_OnDir_m, R_OnFile_m, R_OnOpenError_m;
};

#include "RwxDirTraverser.h"

extern "C"
SEXP
R_RwxDirTraverser_new(SEXP OnFile, SEXP OnDir, SEXP OnError)
{
  RwxDirTraverser *ans = new RwxDirTraverser(OnFile, OnDir, OnError);
  return(R_make_wx_Ref(ans, "RwxDirTraverser"));
}

wxDirTraverseResult
RwxDirTraverser::OnFile(const wxString &filename)
{
  if(!R_OnFile_m) {
    PROBLEM "no method for OnFile in RwxDirTraverser"
    ERROR;
  }

  SEXP e, r_ans;
  wxDirTraverseResult ans;

  PROTECT(e = allocVector(LANGSXP, 3));
  SETCAR(e, R_OnFile_m);
  SEXP This = R_make_wx_Ref(this, "RwxDirTraverser");
  SETCAR(CDR(e), This);
  SETCAR(CDR(CDR(e)), wxStringToR(filename));

  r_ans = Rf_eval(e, R_GlobalEnv);
  ans = (wxDirTraverseResult) INTEGER(r_ans)[0];
  UNPROTECT(1);

  return(ans);
}

```

```

wxDirTraverseResult
RwxDirTraverser::OnDir(const wxString &filename)
{
    if(!R_OnDir_m) {
        PROBLEM "no method for OnFile in RwxDirTraverser"
        ERROR;
    }

    SEXP e, r_ans;
    wxDirTraverseResult ans;

    PROTECT(e = allocVector(LANGSXP, 3));
    SETCAR(e, R_OnDir_m);
    SEXP This = R_make_wx_Ref(this, "RwxDirTraverser");
    SETCAR(CDR(e), This);
    SETCAR(CDR(CDR(e)), wxStringToR(filename));

    r_ans = Rf_eval(e, R_GlobalEnv);
    ans = (wxDirTraverseResult) INTEGER(r_ans)[0];
    UNPROTECT(1);

    return(ans);
}

wxOpenErrorTraverseResult
RwxDirTraverser::OnOpenError(const wxString& openerrorname)
{
    if(!R_OnOpenError_m) {
        return(wxDirTraverser::OnOpenError(openerrorname));
    }

    SEXP e, r_ans;
    wxDirTraverseResult ans;

    PROTECT(e = allocVector(LANGSXP, 3));
    SETCAR(e, R_OnOpenError_m);
    SEXP This = R_make_wx_Ref(this, "RwxDirTraverser");
    SETCAR(CDR(e), This);
    SETCAR(CDR(CDR(e)), wxStringToR(openerrorname));

    r_ans = Rf_eval(e, R_GlobalEnv);
    ans = (wxOpenErrorTraverseResult) INTEGER(r_ans)[0];
    UNPROTECT(1);
}

```

```

    return(ans);
}

#if 0
extern "C"
SEXP
R_RwxTextDropTarget_RsetOnDropText_method(SEXP r_target, SEXP fun)
{
    RwxTextDropTarget *target = (RwxTextDropTarget *) R_get_wxWidget_Ref(r_target,
    return(target->R_setOnDropText_method(fun));
}
#endif

#include <wx/wx.h>
#include <wx/dir.h>

#include "RwxUtils.h"

extern "C"
SEXP
R_wxDir_new(SEXP r_dir)
{
    wxString dir;
    wxDir *ans;

    if(GET_LENGTH(r_dir)) {
        dir = R_to_wxString(r_dir);
        ans = new wxDir(dir);
    } else
        ans = new wxDir();

    return(R_make_wx_Ref(ans, "wxDir"));
}

extern "C"
SEXP
R_wxDir_Traverse(SEXP r_dir, SEXP r_traverser, SEXP r_filespec, SEXP r_flags)
{
    DECLARE_RWX_REF(dir, wxDir);
    wxDirTraverser *traverser = (wxDirTraverser *) R_get_wxWidget_Ref(r_traverser,
    wxString filespec = R_to_wxString(r_filespec);
    int flags = INTEGER(r_flags)[0];

    size_t ans = dir->Traverse(*traverser, filespec, flags);
    return(ScalarReal(ans));
}

```

}