Plugins

Daniel Westerberg

COLLABORATORS			
	TITLE : Plugins		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Daniel Westerberg	July 10, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Plugins

1.1 Plugins for MultiRen

```
                  How to make MultiRen Plugins
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

This guide was released together with MultiRen v1.4 in 2000-07-04.
If you use this guide you will make a plugin with revision #2.


 This guide is giving you a precise description on how to create a
 plugin to use with MultiRen.
 If you are only in the interest of using plugins you should look
 at the Renplacer/Plugins-page in the main MultiRen guide instead.
 But if you are interested in how it works, sure read on :)

 If you have decided to create a plugin you should (must) read all
 of this guide to avoid confusion MultiRen<->Your_Plugin and to avoid
 taking things for granted that are not to be taken for granted!


                First step
                -> How MultiRen plugins work, read this!

                Communication
                -> How MultiRen communicates with the plugins.

                Finalization
                -> How to make reality of this you just have read.

                Distribution
                -> What to do with your final product.


                The Substitute list
                -> This is a neat little invention!

                Changes in rev.#2
                -> What has changed from rev.#1
```

1.2 First steps..

How it works and what to think of

A MultiRen Plugin is a program that is going to be executed the first time you flip to the Plugins page in the Renplacer tool, or anytime the user feel like reloading the plugins, and then supposed to be Wait()'ing in the background for a signal from MultiRen telling your plugin what to do.

I have made 2 examples. Both in Amiga-E and C using StormC. I'm actually not a C-programmer so bare with me if the examples in C is made in an incommon way of something ;)

The first example I have made, TestPlugin, demonstrates a plugin that returns 2 strings. These 2 strings will contain the first 4 bytes of the file that is to be renamed, in hexadecimal and as a plain 4 character ASCII-string. It features no settings.

The second example, SwapName, will return one string which will be the filename turned backwards. This features one setting that will control if the extension will be part of the swapping or left untouched.

Note: You should not make a plugin from scratch but use the Template.e|c and then you don't need (must not) change anything in the main()-procedure/function but only in the functions that are called when you get the different commands. (See below.)

Now study the examples a bit!

....

..ok, you've done that, now read on.

When your plugin is loaded by MultiRen, it is then immediately executed with an argument containing a decimal number which is the address of a shared object (C; struct) holding (among other things) an ID-long that always has to contain these characters; "MRPO" which stands for MultiRen Plugin Object, when you get the object from MultiRen. The argument string ends with a linefeed and then null-termination. But as this is all taken care of in the main()-procedure/function in the Template.e|c you don't need to worry about this.

You will get one of 5 different commands everytime your plugin gets a signal.

Generally what they are for:

ASK ; Is used only once, and that is first of all.
It asks for info about the plugin and gives the plugin an

```
    opportunity to allocate resources for itself so it doesn't
    have to do that for every filename which would slow things
    down.
    Remember that the things you are doing here must not take
    longer than 2 seconds because then MultiRen will think you
    are not a plugin because you didn't respond and you might
    be ignored, depending on the user.

  EXTRACT ; Is used when MultiRen want your data from a file.
    When you get this command, a filename is pointed to in one
    of the fields in the object and MultiRen want you to do your
    stuff and return the string(s) it manage to extract from the
    file(-name). Make this as fast as possible, otherwise it might
    take time to process 1000 filenames!!
    Also, don't open the config-file, if you are using one, and
    read it here, do that when you get ASK instead!

  CONFIGURE ; MultiRen gives you an opportunity to let the user
              configure eventual settings that your plugin has.
    You may get this anytime between you get ASK and QUIT.
    If you don't have any settings in your plugin you should
    just return with proper return value. (See below)

  ABOUT ; User wants to know what this is.
    Here you should open a requester telling version, and e-mail
    is also good so the user can get in touch with you for
    questions and suggestions and so. Some info about what
    this plugin really does might also be in place here.

  QUIT ; User quits MultiRen or flushes plugins.
    Is used only once and that is right before MultiRen quits, or
    if user decided to flush plugins. Now you should deallocate
    all resources you might have allocated.


If you would get some other command, you should return
ERR_UNKNOWN to tell MultiRen that this is an old plugin not
supporting this new feature that your plugin (or you and me
right now) do not know about. It's already done in main() though.

There are some different return values that you must use after
every command:

  ERR_OK ; Everytime is OK.
    This should always be returned unless you got an error like
    a file didn't open or you run out of memory. If you do not
    return this when you got the command ASK, your plugin will
    be unloaded and ignored and you will never get another
    command.

  ERR_NOMEM ; Return this if you run out of memory.
    This should only be happening during ASK, CONFIGURE or ABOUT
    as you should not allocate any memory or resources while
    EXTRACTing or QUITing.

  ERR_NOFILE ; The file for renaming didn't open.
    Return this if the file that you were supposed to work on
```

```
    didn't open. This should only be return if you get command
    EXTRACT.

  ERR_NOSIG ; Could not allocate signal.
    Return this if you could not allocate a signal for your plugin.
    Don't worry about this, it's taken care of in the main()-proc/func.

  ERR_NOTIMPL ; Tells MultiRen that this thing is not implemented.
    Return this if you get the command CONFIGURE or ABOUT but
    have no settings or no about-requester in your plugin.

  ERR_UNKNOWN ; Tells MultiRen that you don't know what this
    command is for. You should not return this if you get some
    error that you don't know what it is, then use ERR_OTHER,
    this is only for commands that is not listed here. Don't
    worry about this, it's taken care of in the main()-proc/func.

  ERR_OTHER ; Some other error occoured.
    Return this if you encounter some other error than Out of
    memory, Could not open file to rename or so. Maybe if
    your plugin wants to open a window but could not. Or if
    you were trying to open another file than the one supplied
    for renaming. Maybe you are using some kind of database
    stuff or something that didn't open..

  ERR_WRONGFORMAT ; The file you was supposed to examine was in
                    the wrong format.
    Return this if your plugin is supposed to read information
    from a file of a specified format (like f.ex IFF-8SVX) but
    the file was not and 8SVX file.

  ERR_NOINFO ; This file has the right format but lack the
               information you were supposed to get.
    Return this if say you were to extract a string from a certain
    filetype but this particular file did not have this string
    in it.

  ERR_FATAL ; For total malfunctioning.
    Return this if say, your plugin is started by itself from
    f.ex CLI. You should also always return this if the ID is
    not "MRPO", which usually indicates that it wasn't started
    properly by MultiRen.
    Don't worry about this, it's taken care of in main().

Never use any other return values than these above, especially
not if you are returning a value to MultiRen (i.e. not to f.ex
DOS if the plugin was started wrong).

As you probably already understand, the commands and the return
values are constants that is defined in the examples.
The command-constants starts with COM_. So when you get f.ex the
command ASK, it is really only a field in the object that equals
the value of the constant COM_ASK. (See examples)


                List of constants
                -> In case you have misplaced the examples..
```

Ok, now you have a breaf overlook of what to think of and how
to use and interpretate events that will happen. On the next page
there is a detailed description of the object and how to use it
along with the commands and so.

1.3 Constants

The values of the constants used in a plugin
~~~~~~

The commands:

```
 COM_ASK       = 1
 COM_EXTRACT   = 2
 COM_CONFIGURE = 3
 COM_ABOUT     = 4
 COM_QUIT      = 5
```

The return values:

```
 ERR_OK          = 0
 ERR_NOMEM       = 1
 ERR_NOFILE      = 2
 ERR_NOSIG       = 3
 ERR_NOTIMPL     = 4
 ERR_UNKNOWN     = 5
 ERR_OTHER       = 6
 ERR_WRONGFORMAT = 7
 ERR_NOINFO      = 8
 ERR_FATAL       = 20
```

The maximum length of a filename in MultiRen is currently 512
characters.
The maximum length of a filename with path is 1024 characters.
The current maximum length of files handled by AmigaDOS is 107
characters and the FastFileSystem 31 characters.

1.4 Communication..

As I mentioned earlier, all communication is made through a
shared object. This chapter describes this object in detail,
showing you how to use it.

All strings I talk about must be null-terminated.
(E-strings is ok)

Below is the object you will get:
~~~~~~

```
OBJECT multiren_plugin          ( struct multiren_plugin { )  bits:
 id:LONG                        (   long )                    32
 task:LONG                      (   Task* )                   32
 sig:LONG                       (   long )                    32
 return:INT                     (   short )                   16
 command:CHAR                   (   char )                    8
 numstrings:CHAR                (   char )                    8
 stringlist[256]:ARRAY OF LONG  (   char*[256] )              256 * 32
 name:PTR TO CHAR               (   char* )                   32
 newname:CHAR                   (   char )                    8
ENDOBJECT                       ( } )
```

```
 id
   ID-long, this shall always be "MRPO" ($77828079) when your
   plugin is started. If not, print error message to stdout or
   something and return ERR_FATAL. (See examples)
   It's already taken care of in main().

 task
   When command is COM_ASK this will point to the task of
   MultiRen. You need this to be able to signal to MultiRen.
   Set it to your own task after you have saved the pointer to
   MultiRen.
   Do not mess with this, it's allready taken care of in the
   main()-procedure/function in the Template.e|c!

 sig
   When command is COM_ASK this will contain a signal mask
   allocated by MultiRen that you need to be able to signal
   MultiRen. You need to allocate a signal yourself and set this
   variable to your allocated and left-shifted sigbit after you
   have saved the one that came, so that MultiRen will be able to
   signal your plugin.
   Do not mess with this, it's allready taken care of in the
   main()-procedure/function in the Template.e|c!

 command
   Tells you what to do.
   You can get COM_ASK, COM_EXTRACT, COM_CONFIGURE, COM_ABOUT or COM_QUIT.

   If you get COM_ASK;
    Set numstrings to the number of strings you want to return.
     You can never change this value later on. (1 to 255 is legal,
     0 would make no sense)
    Fill stringlist from position 0 to numstrings-1 with informative
     strings telling what will be returned here. This is what the
     user will see.
    Set name to point to a string telling the name of the plugin.
     MultiRen will not show the filename to the user but only this
     string.
    Set newname to 0 or 1. 0 means that you plugin prefers to get
     Old name as name parameter. 1 means New name.
    This is also the time for allocating things that you will need
     later on instead of doing these allocations for every filename
     which would be very inefficient.
    Return ERR_NOMEM if you run out of memory or ERR_OTHER if some
```

```
     other resource could not be allocated, like a library, else
     return ERR_OK.
     If you do not return ERR_OK here, you will never get anoher
     command, not even QUIT.
    Function ask() is defined for this command in Template.e|c.

   If you get COM_EXTRACT;
    Fill stringlist from position 0 to numstrings-1 with the strings
     that your plugin was made to deliver.
     The filename with path will be in name.
    Return ERR_OTHER if something went wrong other then opening
     the file and recognizing it, else ERR_OK.
     You should not return ERR_NOMEM as you should not allocate
     memory here.
    Return ERR_NOFILE if you could not open the file whos name
     you got in name.
    Return ERR_WRONGFORMAT if the file in name is not in the format
     you expected.
    Function extract() is defined for this command in Template.e|c.

   If you get COM_CONFIGURE;
    If you recieve this command you can popup a GUI with configurations
     if you want.
     Return ERR_OTHER if something went wrong, else ERR_OK.
    If your plugin doesn't have any need for configuration you should
     return ERR_NOTIMPL to tell MultiRen that you have no options to
     set. Returning ERR_OK works too but then the user might get
     confused as nothing appeared to happen.
    Function configure() is defined for this command in Template.e|c.

   If you get COM_ABOUT;
     Now you should popup a requester or something telling the
      user the author and version and such. Maybe some information
      about what this plugin does too.
     If you don't have an about-reuester you should return
      ERR_NOTIMPL.
     Function about() is defined for this command in Template.e|c.

   If you get COM_QUIT;
    Deallocate anything you have allocated during COM_ASK, this is
     the last command you will ever get because now MultiRen is
     quitting.
    Function quit() is defined for this command in Template.e|c.

numstrings
   This keeps information of how many strings MultiRen will use
   from you after an EXTRACT command was executed. This value can
   only be altered if command is COM_ASK, ie. when you plugin is
   started.

stringlist
   When command is COM_ASK you must fill numstrings number of
   positions in this array with informative information (strings)
   that will be showed to the user telling him what he is going
   to get if he uses this string.
   When command is COM_EXTRACT you must fill these numstrings
   number of positions with the strings you got from the file or
```

```
  filename, that is, the strings that your plugin was designed
  to generate.
  If you want to leave a string empty, NEVER set
  it to NIL (C; NULL)! Just set a zero-length string to it
  instead like: mrp.stringlist[0]:=''  (C; ="";).

  Note: No memory is allocated to the pointers that this array of
  longs consist of, you have to provide strings and just
  set their pointers to these positions. (See examples)
  Static strings is fine as the programs code and data stays
  resident in memory. But no StrCopy() directly to these pointers
  without first allocate memory to them.
  Note: If you plan to set static string to the positions, you must
  not use local strings, only global. Local string is allocated
  on the stack and is flushed in the instance you return from the
  function in question!

name
  When command is COM_ASK you must set this pointer to a
  string that tells this plugins name.
  When command is COM_EXTRACT this will point to a string
  containing the filename with path of the file that you should
  get some info from. Or maybe you only want to manipulate the
  filename and return it in stringlist. I don't care, only
  it returns some kind of string that will make the user happy :)
  Never set this to NIL either, it goes with the same rules
  as stringlist when it come to empty strings.

newname
  When command is COM_ASK you must set this to a value which
  currently is 0 or 1. 0 means that your plugin would prefer
  to receive Old name in the name field when you get COM_EXTRACT.
  1 means that you want New name. You should request Old name (0)
  if your plugin is supposed to open the file that later is to
  be renamed. You should request New name (1) if you intend to
  manipulate the filename only and not want the file it stands for.
  The user can change this later if he want or need to.
  If you do not set this, maybe you are a rev.#1 plugin, then
  MultiRen will set this for you.


 Ok, now we have covered the communications a bit.
 Now take a look at the examples again.
 Then go on to the finalization..
```

1.5 Finalization..

```
So, now you should have enough information along with the
examples so that you could create your plugin.
But let's go through some basics:

 You have set the extension ".mrp" (MultiRen Plugin) on your
 plugin otherwise it will be ignored by MultiRen. ProgDir:
 of the plugins is MultiRen's directory, so to access the
```

```
 plugin-directory (if you want to save a config or something)
 is PROGDIR:Plugins/.

 The first time your plugin is run you will get the command ASK.
 And this is when you should allocate resources, which maybe
 will involve allocation of strings. You should also read the
 config-file now (if you have any).
 You will only get the command ASK once.

 Then you will get none or alot of EXCTRACT-commands and this
 is when you will 'do your job', and do make it fast too.
 Maybe you also will get one or more CONFIGURE- and ABOUT-
 commands sometime if the user feels like it.

 The last command you will ever get is QUIT and now you
 should deallocate anything you allocated when getting ASK
 because now your plugin will not be run anymore and
 MultiRen will deallocate the signal.
 You will only get the command QUIT once.


And now for some guidelining notes..:

 Note #1
  If something goes wrong in the process when command is
  EXTRACT, don't put up a requester or something else that
  will freeze the process until the user interacts, return
  an appropriate error-code instead and let MultiRen handle
  your problems.

 Note #2
  Don't put up any configuration or requesters when you get
  the command ASK or QUIT, this will only irritate the user.
  But if you make a shareware plugin, then sure this is the
  right place for nag requesters. However, if you wait more
  than 2 seconds in ASK, MultiRen will start to wonder what
  happened to you and ask the user if he wants to ignore you.

 Note #3
  Put the extension ".mrp" on your plugin.

 Note #4
  Do not change the values of the constants, your plugin
  wouldn't work if you do, ofcource.

 Note #5
  Never return a null-pointer as a string, return a zero-length
  string instead.
  Not like this:
    mrp.stringlist[1]:=NIL     ( C; mrp->stringlist[1]=NULL; )
  but like this:
    mrp.stringlist[1]:=''      ( C; mrp->stringlist[1]=""; )
  instead.
  The same rules apply for the name-variable.


Ok, now we should be finished!
```

```
You should use the Template.e|c as a base to make your plugin
from. That way the handling of the object and task/signal will
be correct. Do not mess with main() in the template or the
examples.

 Happy coding!

And don't forget to give me a copy :)
```

1.6 Distribution

```
 Ok, now you are finished with your plugin and want the world
 to have it but you wonder how!

 You should send me your plugin with the source code.
 I will include the compiled program into the archive of
 MultiRen and I will keep the source for myself so that I
 can make fast changes to it if I happen to change the
 structure of how the plugins work or something without the
 need to contact you for doing it, but I will ofcource
 notify you and ask you to to the changes first. But in
 say three years you might not be interested or reachable
 anymore and then it's quite good for me to have the source.

 You will be mentioned in the main guide of MultiRen with
 name and e-mail if you want.

 You could also release it to Aminet to get it out as fast
 as possible as it might take some time before MultiRen is
 updated and released again..

 It would be good if you could send me a readme too,
 containing some info of what the plugin does and how to
 use it (if you have config or so..). If you release it to
 Aminet, the Aminet-readme is good enough, providing it
 contains basic info. It will be included in the archive
 along with your plugin.

 My address to send programs and ask me things and so is:
  deniil@algonet.se

 See the main guide, link Author for more info about me.

 For fresh updates you could also browse in on:
   http://www.onyxsoft.nu/


 Allright! That's it!
 I hope something good came out of this guide :-)
```

1.7 The Substitute list..

There is now a new program in the MultiRen archive called SubstituteGenerator - The General Substitute list Generator!

This program generates a little textfile with 2 columns of characters where the first character, if found in a string that is to be returned by a plugin that supports this substitute list, will be replaces by the second one. If the second one does not exists, i.e. it is a linefeed character then the first character was supposed to be removed and not replaced.

This probably sounded messy but it's really simple. There is a substitute.list supplied already in the plugins directory. Just start the SubstituteGenerator and it will immediately get clearer.

Currently the ID3-Tag Exctractor and the IFF-8SVX plugin supports this list.

If you make a plugin that returns strings picked right out of a file then you might consider using this list so that you can control what will end up in your newly renamed files later. ?*#()/: is not really allowed in AmigaDOS filenames f.ex. The substitute.list that comes with MultiRen has these characters already defined. You can change, remove or add your own replacements if you want.

The format of the substitute list is:

  The first line can be up to 200 characters and filled with settings. Currently only on setting is defined. That is the first number in this line which tells plugins if they should use the list or not. 0 means do not use the list and 1 means to use it. My plugins also has their own setting which overrides this one though..

  Then the substitute lines follows. They consists of 1 or 2 charcters followed by a linefeed. If there are 2 characters then the first one should be replaced by the second one if found anywhere in the strings you are returning to MultiRen. If there is only 1 character then this should be removed from the strings you are to return if it's found.

  Look at the file Substitute.list and you will understand.

You can start the SubstituteGenerator from the rightmost page in the Prefs-window in MultiRen too.

1.8 Changes..

Revision #1:

```
    Initial release with MultiRen v1.3.

  Revision #2:

    Added a new return value: ERR_NOINFO.
    Added a new field in the object: newname.

    Released with MultiRen v1.4.
```