DOpus_C_Programming

Helmut Hummel

COLLABORATORS

	TITLE : DOpus_C_Programming	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Helmut Hummel	December 28, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

DOpus_C_Programming

1.1 Welcome

DOpus C Programming guide

Even if DOpus support greatly ARexx and other methods to do some programming work, it is often useful to write in C. Sometimes is a little bit speed needed or you need some features which are in ARexx not (or only difficult) available. Additionally you may be pleased by using DOpus library functions, which are very handy to use.

This guide should be not a replacement for the docs of the SDK, but it should show you well commented, how you can create easily a module by yourself - we do some examples here :-). But at all you may not expect that all stuff is here, this would take a little bit too much time of me (or better: it would not possible for me to write such a lot of stuff without getting bloody fingers...:)).

Introduction

 Chapter 1: What is a module / for which things must be taken care of

Preparing section (Directory "Simple_Module1")

 Chapter 2A: Needed stuff / supplied files (SDK and here) / localization
 Chapter 2B: How to recompile, the files in the SC directory

Beginners section (Directory "Simple_Module2")

 Chapter 3A: Simple module with a requester, a window and a hidden command
 Chapter 3B: DOpus notification, argument parsing
 Chapter 3C: A small menu for our window
 Chapter 3D: The most important DOpus DOS routines
 Chapter 3E: 'Autostart' of a module and detaching of our own process
 Chapter 3F: Passing arguments to a detached process - IPC_Commands

Average section (Directory "Average_Module1")

 Chapter 4A: An exchange.module using popup menu and saving of preferences

```
Code fragments (Directory "Code_fragments")
(It does contain also not direct DOpus related sources, since I saw already
some strange stuff done by average coders, especially for filerequesters...,
all right Leo ? :) )

 Chapter 5A:  Getting selected entries of the source lister (callbacks)
 Chapter 5B:  Choose destination by SelectionList(), ASL filerequester
 Chapter 5C:  IPC_Launch()..., but simplier to use...:)
 Chapter 5D:  Reading and writing the names of a whole Att_List
 Chapter 5E:  How to get the free space of any device
 Chapter 5F:  Scanning of a complete device for files without to recurse

Things left to mentioning

 Hints and other stuff
 Resources and troubleshooting
 Author, corrector(s)
```

1.2 What is a module / ...

```
A module is nothing else than a library. This has at least 2 functions
for DOpus to use and only one of these is important for us.

Note:
If you use a different compiler than SAS (or Storm), you must take care
of the positions of both functions. In this case you may refer also
CLib37x.lha (AmiNet) and your compiler manual.

--

If you have never created a shared library before, you must know some
things before about:

- because a library can not expect some IO channels (stdin, stdout),
  you can not include the "stdio.h". You can also not use a simple
  printf() without to have a valid (checked and locked) output window.

- your code must always end with a "return". "exit" is not allowed in
  libraries (, because stderr is also not available).

- due the possibility of more than one process running in this library,
  you should not use global variables, which may be modified by a
  process. If you need this, use a SignalSemaphore to allow only one
  access at the same time.

- you can not use to recursing functions since you have a fixed stacksize.
  There is always a way around...
  (In Module_fragments/ScanTree.c you can see an example how to scan a
   complete device for files without to recurse...)
```

1.3 What is needed / supplied files (modinit.c)/ localization

```
You need:

CatComp  - it is the offical catalog compiler of the Amiga
           You can find it on the "Amiga Developer CD 1.1" or ???

Compiler - Suggested to use (and I refer also only to this) is SAS 6.58,
           also possible may be StormC and some others.
           If you use a different Compiler, it is possible you must change
           a lot of the DOpus include files, so I would suggest to you to
           buy a SAS one (they are currently cheap).

           If you want anyway try to work with a different compiler, it
           may needed to know for you, what some SAS keywords mean:
           __saveds = keep the register A4
           __asm    = function does expect the arguments in specified registers

Additional you may use anyone of the GUI-Builders for trying a GUI - only
to get an impression how it may look later. The code of these is NOT
needed...


How do I work myself :

I use the editor of MaxonC 3.0 (Edward) to write my stuff and have with
this also the online help of HotHelp (a special format of the Autodocs).
To compile I turn to the DOpus Workbench and simply click on the "Build"
icon. I use also the GadToolsBox 2.1 for a quick arrangment for a GUI.

--

Supplied files

The important supplied files of the SDK you can find also here in the
directory "Simple_Module1" - with some small changes :-). The files you
may change are well commented. You may use the contents of this
directory as basic for your own project.

Take a look in:

 modules.fd
 modinit.c
 smakefile
 SCOptions

 includes/buildin.cd
 includes/Project.h
 ModuleEntry.c

To rebuild the other projects you must copy the whole directory of
"xxx_Module?", the subdirectories you want (and the directory "sc",
if you not install the contents to your sc: drawer - but here you must
modify the smakefile) to a disc. Then you may simply use the "Build"
icon (needs also properly installed SAS-C).

The source of the rndpic.module and of the time.module are only for
average programmers and are not so heavy commented. I have only supplied
```

```
them for the case you want to screw it up...
--

Localization

The way how DOpus modules are localized is really easy:
If you need a string which should be localized, you should give it
an ID (a kind of define) - ie. MSG_TEST - and write the ID and the
real text in your "includes/buildin.cd". If you need later the string
in you code, you write:

DOpusGetString( locale, MSG_TEST );

On other places may also be the ID enough (ie. in a popup menu).

Of course you'll get only a pointer and you can't modify the contents.
```

1.4 How to recompile

```
Since all this stuff is on CD, you can not recompile this things
directly from here. So I would suggest to you that you copy the whole
directory of this guide to your HD.

As minimum is required:
                        - the complete project directory of you choice
                        - and the directory SC.

These both must be placed in one drawer!

--

The "SC" directory contains a headerfile (sdi_std.h) and an objectfile
(sdi_std.o) to use RawDoFmt(),... as replacement for the mostly used
string functions and some ctype functions. You may use them as normal
if you define SDI_TO_ANSI, but you can not include the files then which
does contain the real ANSI functions and you have to take care then if
you use a xprintf function that you only supply formatting signs of
RawDoFmt() (see Autodoc/HotHelp what is allowed there).
If you want take a look in the  sdi_std.h...
Even if you does install these files to you SC: drawer within this
 Install button , you have to copy the drawer SC as well, or you must
change the files of the project (smakefile, include files) to set the
right path to this files.

This files are created by Dirk Stoecker - Author of XPK,...
(and forgive me... - renamed by me :) ) and are supplied within the
source of XPK (Public Domain).
```

1.5 Simple module with a requester, a window and a hidden command

```
Let's do some work... :-)
```

```
In every chapter you will find the links to our standard files, which
are a little bit depending from our current project (Simple_Module2):

 includes/Project.h
 includes/buildin.cd
 ModuleEntry.c
 smakefile

--

Now let's open a simple requester :
 includes/Requester.h
 Requester.c

You're well ?
Ready to view a little bit more stuff ?
 includes/Window.h
 Window.c

A little bit easier please ?
We make it very easy ... :)

 HiddenCommand.c

Like you have seen now, it is really simple to write something. But
I give you some more examples, so you can "steal" what you need... ;-)
```

1.6 DOpus notification, argument parsing

```
Just before we have opened a window. But what should happen with this
window if DOpus tries to quit or hide ?

This is the point the DOpus notification comes in, but you may use
also the other events, if you need them.

So let's adding to our window program a little bit stuff...

 NotifyWindow/NotifyWindow.c


Like you can see, there are additional signs in the arguments. Jon did
here also preparing a little bit the argument parsing for us. So let's
do it...
(for overview we do not install a notify here)

 ParseWindow/ParseWindow.c

---

Note: Use the smakefile or "Build" icon in the subdirectory to rebuild.
They are just the same like on beginning, only the paths are fitted to
this directory.
```

1.7 A small menu for our window

Creating a menu for a window with the dopus5.library is easier than to do this with the standard Amiga routines. The only needed thing is a MenuDef structure and add this later to the window.

So let's take a look in the headerfile

MenuWindow/MenuWindow.h

and in the program

MenuWindow/MenuWindow.c

1.8 The most important DOpus DOS routines

Here I use now our hidden command and again a requester to demonstrate some of this routines.

You can use the command to do different things - hmmm..., why should I write this double - just take a look in

DOS/Dos.c

Really easy stuff, or ?

1.9 'Autostart' of a module and detaching of our own process

Autostart of a module is only a question to set the MODULEF_CALL_STARTUP flag in our ModuleEntry.c. See in the next section the link.

--

Now it is going harder..., but it stays solveable :)

There a two ways to launch an own process (similar to detaching in CLI)

ARexx solution:

after the module does get a command, it does send a command to the DOpus ARexx port and launch with this the real command (this is detached since the sending of the ARexx command will return before the real command is finished)

This is done in the Autostart case (FUNCID_STARTUP) of the DetachWindow/ModuleEntry.c

C like solution:

simply using the IPC routines of the dopus5.library - hmm..., good, not sooo simple, but ... :)

```
 DetachWindow/DetachWindow.c

This are also the basic routines we need for our next work, so I hope
you have understand all.

Note:
If you want to use the IPC routines, but you want not to take care
about all needed stuff, you may also have a look in
 LaunchIPC.h
 LaunchIPC.c .
This routines does make retrieving you own process so easy as possible.
```

1.10 Passing arguments to a detached process - IPC_Commands

```
If you create a command which does a longer work with "waitstates" and
which should not be launched twice, you can use this wait phases to
receive additional work/options,...
All you have to do is to detach this command and send a message with
the options to it. This can be done multiple times and so we get what
we want. For example you may make a try with the time.module - even
if the clock is already running, you can change the background within
the DOpus CLI...

Enough talked, let's do it... :)

At first we need a globally indicator to detect if the command is
already running. It is needed to setup this in the

 IPCWindow/modinit.c

And now let's make some heavier things...

IPCWindow/IPCWindow.c

So..., that it was on extrem simple programs. Now you are ready to
take off ;). Next level: Average... :)
```

1.11 An exchange module using popup menu and saving of preferences

```
Warning: This module is a HACK !!!

Nothing of the needed things of the commodities.library to create such
a module is really documented and may be they have a good reason not to
do this. However..., this module seems to work OK ... :)
--

DOpus popup routines are really a nice thing and also easy to use
(I had a look in some sources with the popmenu.library, ohhh...,
what a difference :) ). You have only to create a list of the
entries you want, another list if one entry should have a submenu,
and must only set the right flags. No need to take care about
VisualInfo() or something else, all is already done by this
```

routines.

Sometimes you may have to read or write IFF files, what you can do by using the iffparse.library. But this is not much comfortable, you have to do a lot around. So you should make a try with the DOpus IFF routines and you will notice then, that is so simple like to read a normal file.

--
The only reason that this module is average level, is that here is a lot of code needed to get the CX data and sort it.

So let's go... :

```
 modinit.c                (we need a globally pointer...)

 includes/Project.h       only a minor change...
 ModuleEntry.c            only a minor change...

 includes/DOExchange.h
 DOExchange.c
```

1.12 AttList's, DOpus listviewgadget, popup menus

AttList routines are a cheap way to do some List handling. If you want to sort something or even to set the value for DOpus listviewgadgets, they provide an easy handling of the list and the node names. Some of this routines does also work on "normal" list's and so they remain interesting for other things.

The DOpus listviewgadget is a powerful replacement for the standard listviewgadget. You may make the entries checkable, show them in different colour or even use doubleclick events for additional work. Of course you may also use the standard actions/events within this gadget. Take a look here in the include file dopus/layout.h - it is really worth to do it since not all of the tags for the DOpus listview are mentioned in the SDK...

--

Here we open a window (command "GetDirEntries") with a listview, which does show the contents of the supplied directory - directories and files are shown in different colours. If you doubleclick on a directory this will be scanned in (means also a kind of a lister in name mode...). The list is normally sortet, but you may use the buttons Up, Top, Down and Bottom to manipulate entries in this list - this shows the using of the AttList functions in the source.

So go on to this heavy source... :)

```
Project.h
ModuleEntry.c

GetDirEntries.h
```

GetDirEntries.c

1.13 Hints and other stuff

1. If you open a window with a stringgadget, you can modify the IDCMP - add there IDCMP_ACTIVATE.
   If you have done this you can activate the stringgadget again, if it was active before the window was deactivated.

2. Take care, that you'll get mostly no valid screenpointer for your module, if it is called on DOpus startup (especially in WBR mode). If you install a notify, use the screenpointer of the "DN_OPUS_STARTUP" message instead.

3. You does not get in DOpus a really borderless window even if you use the right (needed) flags... (no workaround possible...)

4. Use only the matching routines to reply messages !

5. Before you start to code, you should really know exactly, what you want to reach (how should it work). It will save a lot of work...

6. Try to test your module with the Enforcer before you make it public. If you can not run it, because you have no MMU, search someone. If you know nobody else, you may also ask Greg to ask a Beta tester of DOpus, who can run it (, hopefully me too in the next time :), then you may also ask me...).

7. If you want to use for your own window the DOpus feature "Mousebuttons over inactive buttonbanks", you should do :

```
WindowID id; // must exist as long the window itself
...
if( (window_pointer = OpenWindow(...)) )
  {
     SetWindowID( window_pointer, &id, 0x1400, 0 );
     ...
  }
...
```

   This does work properly for all gadgets of your window. If you have no gadgets and want to watch out for a simple mouseclick on your window (IDCMP_MOUSEBUTTONS) , see Chapter 3A for a simple workaround...

8. Do not forget to read also the ARexx related stuff, even if you want to write a C program. Some things are more easily to do by using a DOpus callback to send a ARexx command and the right command in than doing all needed stuff itself in C.

9. It is always also a good idea to read the include files. You may find there things which are not mentioned in the SDK (and here).

1.14 Resources and troubleshooting

```
You may also take a look in the sources of the rndpic.module, or even
the time.module (this source is not optimized and is unfinished, I am
current working on it...). These are not in the same style like here,
but I swear I am on the way to do it better... ;)

A good idea is to look in the AmiNet biz/dopus directory, there you
may find also some sources and other interesting stuff.

If you need help on DOpus related problems (no programming problems),
you should send a EMail to
                listserv.lss.com.au
with
                subscribe dopus5 <your email address>
in the text.

You can send also
                subscribe dopus5-dev <your email address>
in the body to subscribe you to the developers list (for programming
related problems).

You will recieve a welcome message detailing the commands available, and
then the crap...eeerrr, the 'help' should start rolling in :)

Updates, announcements, and other sundry items of interest can be found on
the GPSoftware WWW page, located at http://www.gpsoft.com.au.

Bug reports and requests for help can also be be sent to Dr Greg Perry at
gperry@gpsoft.com.au, you will need to send your registration number
in the message if you expect a response.

If you have troubleshooting while creating you new module, you may use
the developer mailing list first. There you may get in contact with
Jonathan Potter, Greg Perry, Andrew Dunbar, other developers or even me,
who does try their best to help you to solve your programming problem.
```

1.15 Author, corrector(s)


```
This guide was created by

Helmut Hummel

Donathstr. 9
08451 Crimmitschau

Germany

EMail (preferred): hhummel@t-online.de
WWW: http://home.t-online.de/home/hhummel/  (not too much updated, sorry...)
```


```
My sometimes terrible english was corrected by

...
```

Small parts may be stolen from Dave Clark's DOpus ARexx tutorial...:)

Tests of the useful modules done by

Dave Clark, Trevor Morris, James Hayes, Jens Weyer and Greg Perry
(, surely also by Jonathan Potter :), who did me also support on some
questions).

Big THANKS to the last both mentioned people for this wonderful program
called Directory Opus ! Also to Andrew Dunbar...

1.16 Index

INDEX