RODOS

How to begin

Version: 2.0
Date: Sept. 2019
Author: S. Montenegro

1. RODOS Tutorial: How to Begin

The directory structure of the current RODOS distribution is divided into following directories:

- api API and implementations for many different hardware platforms
- build Compiled libraries for the PC
- doc Documentation
- scripts Scripts for compilations on many different hardware platforms
- src Implementations for various platforms
- support Programmes and libs often used in space applications
- tutorials Examples, for you

To begin, you will need first the api and the tutorials.

You can find general documentation, introduction and programming directives in the doc directory. In this distribution you will find following implementations: for Linux-x86, AMR V*, Cortext-*, LEON, PPC, Sparc, and on the top of other operating systems like Posix (Linux), Free-Rtos, Rtems, Windows. For ARM, PPC, LEON, etc. you will need the corresponding cross compiler. You may download from our homepage a Virtual-Box or VmWare boot image which include already many of these compilers. As development computer we assume a **Linux** installation. All compile scripts are bash scripts.

In each directory you will find a README_* file. Please read this first, then continue.

Please read first the documentation found in

1. doc/wikipedia-page.pdf

2. doc/doc/intro-details.pdf

Then you may try some tutorials to get the feeling. Please begin with

tutorials/first_steps

We recommend you to read:

doc/doc/source-docs/codingdirectives10b.odt

doc/doc/how_to_and_warnings

You will find Doxygen code documentation, which you may use as developer reference. The best is to find an example. We added many for almost everything in tutorials/core and in all others tutorials (search with grep).

To compile the RODOS libraries and applications (test programs, tutorials, your applications) we use shell (bash) scripts (no makefiles). All scripts (for many different target CPUs) are in the directory named "make".

1. Please start a bash shell.

First we have to set some shell variables.

2. Go to *your* RODOS directory.

3. Type "source setenvs.sh".

This will set many shell variables, e.g. RODOS_ROOT, and it will extend to search path to $RODOS_ROOT/rodos-core/make .

Now you create the RODOS libraries.

4. For Linux type "rodos-lib.sh linuxMC" (in which directory you are is not important)

To remove older compilations, you may call the “rodos-clear.sh” script (do not do it now, else repeat step 4). You do not need to be in the rodos-root directory you may call it from anywhere.

To go to your rodos-root you may just type “cdrodos”.

By the way it would not be bad to take a look to all shell scripts in scripts/* .

Now go to the tutorials, read the README* files, compile and execute the delivered programs, then experiment with modifications and with your own programmes.

5. Go to the tutorials

Type

cdrodos

cd tutorials

Please follow this sequence:

10-first-steps how to begin, your first steps with RODOS

20-core ref. basic functions: Threads, time, synchronisation

30-communication-and-bbs simple intern communication using the middleware

In each tutorial directory, please read the README_*.pdf file.

For each file in the tutorial read the code, compile it using the command:

6. For linux: rodos-executable.sh linuxMC <file-name> [<file-name> <file-name> <file-name>]

Important Note: RODOS hat some thing very special. Active objects like Threads, Subscribers, Initiators, etc will be activated autonomously. You do not need a reference/call in a main() or so, like in (all) other operating systems. Just link the object-file (*.o or compiled *.cpp) to your executable (see "list for file-names to compile") and the active objects will do their job without any further action from you.

A very popular development board is the discovery-board which uses the CPU stm32f4. You may compile RODOS for this board (if you have the cross compiler) using

5. (alternative) for discovery board:

rodos-lib.sh stm32f4

rodos-executable.sh sm32f4 <file-name>

In this distribution you will find these ports:
linuxMC
linux
posix64
posixmac
posix
stm32f4
We have many others, If you need some thing else (Sparc, PPC, LEON, etc etc), please contact us

To execute the Linux version please type

tst

To load the programme in an ARM board and to execute it, please connect the ARM board to your host computer (in an USB-UART Port) and type

stm32f4-load

Please be aware: To execute your programs you shall use only one core. When using a posix port please execute you compiled program as follow:

sudo taskset -c 0 ./tst.

Please be aware: All RODOS classes, functions, variables, etc. are in the namespace RODOS.

Please Note, some thing very special in RODOS: You may compile several independent programs, they will be executed together. You do not need to modify source code or implement a "main" which calls other modules in order to integrate many applications. In our way you may develop applications totally independent from each other, and then just compile them together. Try this! It is total different to what you have used until now!