Universidade de Coimbra
Faculdade de Ciências e Tecnologia
Departamento de Engenharia Electrotécnica e de Computadores

Sistemas de Tempo Real - 2017/2018
Practical Assignment Nº1
Scheduling, Priorities Assignment, and Measurement of Computation Times

Preparation

1. Assume that a particular application includes 3 tasks according to the following table:

Task	Maximum computation time	Activation frequency
T1	30 ms	5 Hz
T2	20 ms	12 Hz
T3	4 ms	50 Hz

2. For the case of temporal deadlines equal to the activation periods, indicate:
   (a) What are the priorities that should be assigned to the tasks (1=high, 3=low). Consider rate monotonic priority ordering (RMPO).
   (b) What is the response time of each task. Is it possible to meet all the temporal deadlines?
   (c) What are the response times of each of the tasks if it is used a prioritisation scheme inverse with respect to the one used in question 2a). The tasks can meet the temporal deadlines?

3. Write a pseudo-code to measure the computation time of a task, using POSIX routines.

Practical Implementation

The work to be performed consists on the implementation of an application that makes use of the POSIX routines, with the goal of studying the scheduling by priorities assignment and the measurement of the computation times. There are three functions, `f1(class,group)`, `f2(class,group)` and `f3(class,group)` assigned to three distinct periodic tasks. These functions have as input parameters two integer numbers that identify the class ("turma") and the group. The corresponding code is in the files `func.h` and `func.o`, and must be compiled with the application developed.

This work should be performed in POSIX [1], [2] with Real-Time Extension [1], [3]. The *kernels* of recent distributions already (and also) largely implement the real-time part of the POSIX *standard*. If, and only if, some part of the work may not be strictly performed with the POSIX functionalities, then for that part of the work it can also be used functions of the "Official Linux" [4], [5] (for example, functions of the Linux library).

All the *software* in this practical assignment should be developed/configured to run in only one processor. Even if the work is developed in a system with only one processor, the *software* should be developed in a manner that it executes in only one processor if later the *software* is compiled and executed in a computer with multiple processors.

The work should be preferably developed in a 64 bits Linux architecture. Alternately, it can be developed in a 32 bits architecture, but should then be able to be correctly complied, run, and tested in a 64 bits architecture.

The points of the work are as follows:

1. Using the POSIX functionalities that you may consider necessary, measure the computation time of each of the functions.

2. Using the results of 1 verify if the system is schedulable, for two priority orderings: RMPO (rate monotonic priority ordering) and its inverse. Present in the report the calculations performed. The three tasks should be periodically activated with periods of 0.1 [s], 0.2 [s] and 0.3 [s].

3. Write the code of an application that, during some seconds, maintains the three tasks jointly running using RMPO. This application should show if the tasks have met the temporal deadlines. All the tasks 1 to 3 should be periodically activated with periods of 0.1 [s], 0.2 [s] and 0.3 [s], respectively.

4. Change the code developed in 3 in order to change the priorities (between RMPO and the inverse) during the execution of the tasks. Comment the obtained results.

5. Develop the source code `func2.c` of a module `func2.o` that imitates the functioning of the module `func.o`.

6. Test the module `func2.o` using it jointly with the created application. For this purpose, you should in particular repeat the work of points 3 and 4 above, using `func2.o` in substitution of `func.o`.

7. Assigning equal priorities to the tasks, test the application with the Round Robin scheduling method. Comment the results.

8. Research and development work: using the POSIX functionalities, implement in POSIX, functions with functionalities similar to the following RTAI functions: `rt_task_make_periodic()`, `rt_task_make_periodic_relative_ns()`, `rt_task_wait_period()`.

9. Repeat point 6, using the functions developed in point 8. Note: the solutions of point 9 and point 6 should be different.

POSIX and Linux Documentation

Resources on the Internet

- **POSIX Specification** – `http://www.opengroup.org/onlinepubs/9699919799/`

- **POSIX and its Real-Time Extension** –
  Consult and study the references, investigating what are the parts of POSIX which are most important for the POSIX part of the work. Some topics to study: *pthreads*, their attributes (e.g. priorities) and scheduling options [1], functionalities for measurement and management of time. Consult [6], including the example in C. Consult the **POSIX specification** [1], including the following parts:
  - ▷ System Interfaces ▷ General Information ▷ Realtime;
  - ▷ Topic ▷ Realtime;
  - ▷ System Interfaces ▷ General Information ▷ Threads.

- **Linux *Manpages*** – `http://www.kernel.org/doc/man-pages/`
  The pages of the Linux manual may have relevance to the development of the work. Section "2: System calls" may have particular relevance.

- **STR 2013/2014** – `http://www.isr.uc.pt/~rui/str/str1314.html`
  Former/old course page.

Clues

Use of pthreads

Some relevant functions:
`pthread_create()`
`pthread_join()`
`pthread_exit()`
`mlockall()`
Compilation flags: `-D_REENTRANT -lpthread`

Compilation with the POSIX real-time extension

Compilation flags: `-lrt`

RTAI Documentation

Internet Resources

- **RTAI official web page** – `https://www.rtai.org/`
  In “Documentation ▹ Reference Documents ▹ RTAI User Manual” there is the official manual. Chapters 5 and 6 are the most important. It is the best available reference. This manual may contain some typos.
  In the upper right corner of the page there is an engine to make searches in the mailing list.
  The “CVS (gna)” (left side of the page) is the CVS repository and includes, among other things, the RTAI *showroom*.

- **API Documentation** – `https://www.rtai.org/documentation/magma/html/api/`
  Documentation for the most recent version of RTAI.
  Key points:
  ▹ Modules ▹ LXRT module
  ▹ Related Pages ▹ An overview of RTAI schedulers
  ▹ Related Pages ▹ LXRT and hard real time in user space
  ▹ Related Pages ▹ LXRT-INFORMED FAQs
  ▹ Search for ▹ . . .

- **API Documentation (alternate)** –
  `http://download.gna.org/rtai/documentation/vesuvio/html/api/`
  Documentation for a closed RTAI *branch*.
  Contains some extra documentation that is almost all valid for the most recent version.
  Key points:
  ▹ Modules ▹ RTAI schedulers modules ▹ Timer functions
  ▹ Search for ▹ . . .

- **Rtai.Dk** – `http://www.rtai.dk/cgi-bin/gratiswiki.pl`
  Informations and tutorials for the RTAI.
  Key points: “1. 3. Start with LXRT”, and “1. 4. LXRT tips and tricks”

- **From Linux to RTAI** – `http://people.mech.kuleuven.be/~psoetens/portingtolxrt.html`
  Tutorial about how to port `C++` applications from Linux to RTAI/LXRT.
  May contain outdated parts.

- **Beginner’s Guide** – `http://www.aero.polimi.it/~rtai/documentation/articles/guide.html`
  Introductory guide to RTAI.
  May contain outdated parts.

- **Real-Time in Linux** – `http://www.isd.mel.nist.gov/projects/rtlinux/`
  Page about real-time for Linux.
  Contains some examples in RTAI.
- **Real-Time in Linux** – `https://gna.org/cvs/?group=rtai`
  Page about real-time for Linux.
  Contains some examples in RTAI.

Report and Material to Deliver

A report should be delivered to the professor in `PDF` format. The report should contain the following information in the first page: name of the course ("disciplina"), title and number of the practical assignment, names of the students, number of the class ("turma"), number of the group. It should be submitted through the Nónio system (`https://inforestudante.uc.pt/`) area of the "Sistemas de Tempo Real" course in a single file in `zip` format that should contain all the relevant files that have been involved in the realisation of the practical assignment. The name of the submitted `zip` file should be "`tXgYrZ-str18.zip`", where "`X`", "`Y`" e "`Z`" are characters that represent the numbers of the class ("turma"), group, and practical assignment, respectively.

References

[1] The IEEE and The Open Group. [POSIX Specification] The Open Group Base Specifications Issue 7; IEEE Std 1003.1^{TM}-2008, 2008. [Online]. Available: `http://www.opengroup.org/onlinepubs/9699919799/` .

[2] Wikipedia. POSIX. [Online]. Available: `http://en.wikipedia.org/wiki/POSIX` .

[3] The Realtime Extension: A White Paper from the X/Open Base Working Group, 1997. [Online]. Available: `http://www.unix.org/version2/whatsnew/realtime.html` .

[4] The Linux Man-Pages Project. The Linux Man-Pages Project. [Online]. Available: `http://www.kernel.org/doc/man-pages/` .

[5] Linux Kernel Documentation. Linux Kernel Documentation Index. [Online]. Available: `http://www.kernel.org/doc/` .

[6] [Linux] RT PREEMPT HOWTO. [Online]. Available: `http://rt.wiki.kernel.org/index.php/RT_PREEMPT_HOWTO` .

[7] Rui Araújo. Sistemas de Tempo Real - 2014/2015. [Online]. Available: `http://home.isr.uc.pt/~rui/str/str1415.html` (Antiga Página da Cadeira).