Configuration and Implementation of RTOS under Linux as Co-Operating System on x86 Dual Core Processor

Amol Kashinath Boke
Electronics and Communication Engineering Department
G. H. Raisoni Academy of Engineering and Technology
Nagpur, India
amol.boke@gmail.com

***Abstract*— The core idea in this project is to have two operating systems for both cores of dual core processor individually. General purpose operating system on one core and Real time operating system (RTOS) on second core. Implementation of same will be done by making one of core independent from General purpose operating system and install RTOS in that core so that both will work as one system i.e. General purpose operating system will look after the non-real time applications and RTOS will ensure that all real time application should not miss their deadline and executed as fast as they can. As by installing general purpose operating system and RTOS in both the cores of dual core processors of system and running them in parallel way, dual core processors is being used to its maximum efficiency. The advantage is that if one of OS gets crashed, whole system will not crash instead one of OS which didn't crash hold the system to safe mode till damaged OS is recovered.**

Keywords*— *Core processor, RTOS, x86 processor, SMT, RT Linux, kernel compilation

I. INTRODUCTION

Nowadays many uses the dual core or rather multi core processor technology in their laptops and personal computers for accomplishment of their day to day computing needs. In dual core processors there are two cores i.e. two processors working simultaneously as each core handles the incoming data strings simultaneously to improve efficiency. To utilize a dual core processor full to its strength, operating systems must be able to recognize multi-threading and software must be Simultaneous Multi-core Threading (SMT) technology written into its code. Since most of us using Windows and Linux as their operating system, these operating systems are capable of providing efficient or rather more than efficient results. But both of them fail to utilize dual core processor computing fully.

In this project the idea is to have two operating systems for both cores of dual core processor individually. General purpose operating system on one core and Real time operating system (RTOS) on second core. General purpose operating system was taken as Linux because it is popular OS and many of us use this one in their device. RTOS for dual core processor is nothing but the guarantees for real time applications to be executed faster than ever while host OS (General purpose operating system) is doing other computing in background.

In this paper further discussion will be done on what challenges were put upon us during this project and what solution we introduce to overcome those challenges. Topic of discussion will also include the hardware and software utilities that are going to use or already used for accomplishments of goal of this paper.

II. RELATED WORK

The basic idea behind configuring the RTOS for dual core x86 processors is to utilize a dual core processor full to its strength. Operating systems must be able to recognize multi-threading and software must be Simultaneous Multi-core Threading (SMT) technology [1] written into its code. Since most of us using Windows and Linux as their operating system, these operating systems are capable of providing efficient or rather more than efficient results. But both of them fail to utilize power of dual core processor computing fully.

As discussed above, there no free RTOS available for the dual core processors which comes in the x 86 families of the processors. Before this paper, general purpose operating system was combined with the RTOS by Windows and introduced the product called on time rtos32win. The embedded operating system from On Time computer science is optimized for x86 CPUs meets hard real-time requirements and implements a Windows subset kernel. On Time RTOS-32 is one of the fastest real-time operating systems that are available on the x 86 platforms. Scheduler performance data can be found here. The development is fully integrated into Microsoft Visual Studio; it supports about 290 Win32 API functions. since this operating system costs a lot for the beginners who just want to experience the RTOS and explore the areas of the real time systems on their x86 processors which the general purpose operating system alongside.

This was the inspiration to configure a RTOS which avoids the cost and focuses on implementing and application of RTOS.

978-1-5386-1887-5/17/$31.00 ©2017 IEEE

III. CONFIGURING THE KERNEL

The kernel is one most fundamental component of all of Linux or any operating system which enables hardware to understand software instructions given via operating system and vice versa[2]. In other words it is Communication Bridge between hardware and software of the system. Resource management is one of the important tasks done by kernel which include scheduling of system processor for important programs use of available RAM, indirect access to many hardware devices including those custom to target choose etc. kernel is central most important component in operating system , capabilities of which solemnly dictates capabilities of entire system.

First step in building of kernel for chosen target i.e. x86 processor is to configure latest kernel. Kernel generates a '.config' file at the end of configuration regardless of of method of configuration.In addition to '.config' file kernel also generates number of symbolic links and file headers is was used by rest of the build. Since target processor is x86 processor, kernel was downloaded from website Keeping in mind that while selecting the kernel one need to be sure that selected kernel is fully stable, otherwise it can cause problems in future proceedings[5].
Before downloading this kernel system needs to be updated and it is done by typing following command in the terminal,

$ sudo apt-get update ; sudo apt-get upgrade

This will take some time for downloading and extracting the files. To know which kernel version system was running on following command is executed in terminal ,

$ uname –a

After knowing version installed on system , latest stable version of linux kernel is chosen from website , downloaded and extracted as follows

$ wget http://www.kernel.org/pub/linux/kernel/v3.0/linux-3.2.17.tar.bz2

$ tar xjf linux-3.2.17.tar.bz2

It will take time to extract, once kernel get extracted package needs to be installed called 'ncurses-dev'.This package will allow view the menuconfig options,

$ sudo apt-get install ncurses-dev

After the installation of above package directory of downloaded kernel is changed and configuration is carried out .

$ cd linux-3.2.17/

$ make menuconfig

After getting menu configuration window linux kernel was configured as the real time linux kernel by making appropriate changes to the configuration and selecting necessary options.

IV. KERNEL COMPILATION AND INSTALLATION

Changes made in kernel were saved under different name other than '.config' because .config is default kernel setting that are already present in our linux system. Changes were saved under name '.config_sample' for simplicity.Next step was to compile the saved file but before that '.config_sample' needs to be moved to '.config'.

$ mv .config_sample .config

$ make

OR

$ make –j2
(in case of multiple cores where 2 indicates no. of cores)

After finishing the compilation , output file can be seen in same directory named 'vmlinux' and 'vmlinux.o' . Now system was ready for installation of real time linux kernel which just now configured. For installing this kernel following instructions were executed in terminal ,

$ sudo make modules_install

$ sudo make install

After completion of installation system needs rebooting for real time Linux to work .

$ sudo reboot

After rebooting the system one can see the new version of kernel running on system.

$ uname -a

So RT Linux has been configured , compiled and being installed on system and from looks of it it is running very nicely.

V. REAL TIME EVALUATION

Even after successfully running RT Linux one of the cores there is no guaranty that it will work i.e. it is needed to make sure that RT Linux is taking less time to execute the process than general purpose Linux. If not then its inadequate because one of the goals of this paper was to make sure RT Linux is faster than usual general purpose Linux and will take

less time in executing any process if compared to general purpose Linux.

So to check the working of this setup, some processes were made to run on both the operating systems and the time taken by both operating systems to complete the process was calculated. To make things more accurate C language programs with both simple (static) and complicated (dynamic) characteristic was taken for compilation as process to execute on both the operating systems. Name given to this simple program as exampleS.c and it is a simple Fibonacci series program in C language. There was complicated program also taken into consideration involving application of data structures and it is named as 'exampleC.c'. After compiling and making sure that there is no error in program TIME command was executed in terminal as shown.

$ timegccexample.c

The above command will calculate the three types of time real time, system time and user time. Now user time is nothing but the time taken by the user to input the data and is depending on the typing speed of user. System time is time taken by the processor itself to calculate the output of process and is depending on the type and speed of processor. But the interest of this paper lies on 3rd type of time i.e. real time taken by the operating system itself for executing the process.

TABLE I. REAL TIME TAKEN FOR EXECUTING SIMPLE PROCESS

Trial no.	Time taken by general Linux	Time taken by RT Linux
1	1.483	1.037
2	1.932	1.142
3	1.333	0.926
4	1.237	0.874
5	1.537	1.005
6	1.784	1.067
7	1.564	1.071
8	1.705	0.962
9	1.1	0.882
Average	**1.519444**	**0.99622**

TABLE II. REAL TIME TAKEN FOR EXECUTING COMPLICATED PROCESS

Trial no.	Time taken by General Linux	Time taken by RT Linux
1	1.983	1.484
2	2.332	1. 659
3	2.333	1.648
4	1.837	1.354
5	2.587	1.655
6	2.458	1.667
7	2.948	1.891
8	3.008	1.782
9	1.596	1.092
Average	**2.342**	**1.581**

The real time taken by RT Linux and General pupose linux for executing both simple and complicated process is shown in tabular format in Table I and II respectively. As fig.1, real time taken by general purpose Linux is a bit more than that of RT Linux for simple process. Real time taken for complicated process is little bit on higher side as compared to that for simple process by RT Linux. But real time taken by RT Linux is much more less than real time taken by general purpose Linux for executing complicated processes which is shown in fig. 2 This is now proof of RT Linux functioning properly and fulfilling real time requirements.

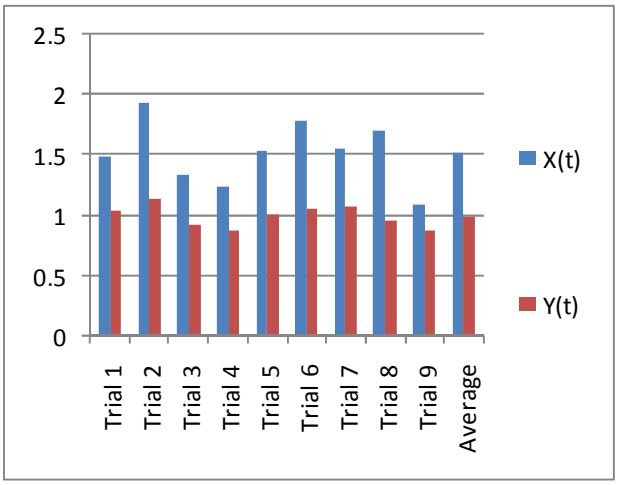

Fig. 1 Comparison of execution time(in sec) taken by general Linux X(t) to that of RT Linux Y(t) for exampleS.c (simple Process)

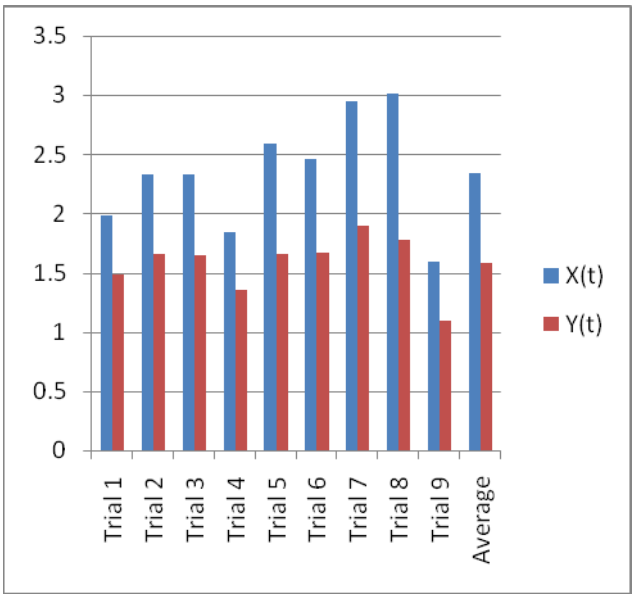

Fig. 2 Comparison of execution time(in sec) taken by general Linux X(t) to that of RT Linux Y(t) for exampleC.c (Complecated Process)

VI. CONCLUSION AND FURTHER WORK

Configuration, compilation and installation of RT Linux on one of the core of dual core processor using virtual machine has been done successfully. While doing so it was

needed to make sure that right type and version of Linux kernel is been selected for the project as there are many types present (i.e. different Linux kernel for different processors/controllers.). Success came in not only installing RT Linux on one core and general purpose Linux on other core of dual core processor using virtual machine, but making sure that RT Linux taking fewer real times in executing the processes in comparison with general purpose Linux also which is kind of one of objective from starting for building this project.

The only thing which was not achieved during above process was system wide communication between two operating systems. In future one can try to overcome this disadvantage and build a single common buffer for the communication between two operating systems involving systems involve the exchange of data like system information, file system, system logs, access to administrative tasks form one operating system to another etc.

References

[1] Copyright © Intel Corporation an abstract on "Dual Core Processors LowersSystem Costs for Embedded Real-Time Applications" 2006.

[2] Karim Yaghmour, Jon Masters, Gilad Ben-Yossef, and Philippe GerumI. "Building Embedded Linux Systems"O'Reilly 2nd edition 2006

[3] M. CvijoviC & M. Kunc. "A Distributed Real-Time Operating Systems"in IEEE pp 260-266.

[4] Michael B. KarboY. "The evolution of the Pentium 4", Chap. 15. PC Architecture

[5] Matt Welsh, Lar Kaufman, Terry Dawson, and Matthias Kalle Dalheimer "Running Linux" , O'Reilly

[6] Arnold S. Berger," Embedded Systems Design", CMP Books

[7] Daniel Bovet and Marco Cesati "Understanding the Linux Kernel",O'Reilly