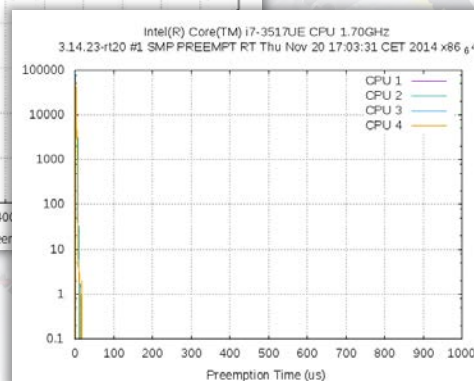
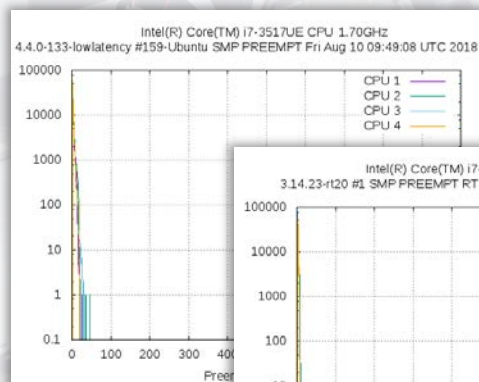
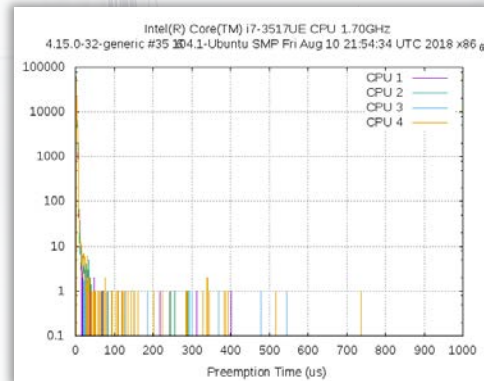


Achieving Real-time Performance In ubuntu®




Kaixian Qu

Email Address: QKX515@gmail.com

Supervisor: Prof. Alexander Leonessa

Terrestrial Robotics Engineering & Controls Lab

Contents

1. Introduction	2
2. What is real-time systems?	4
2.1 Hard real-time systems	4
2.2 Soft Real-time Systems	4
3. How to achieve real-time performance?	5
3.1 Linux kernel.....	5
3.2 Standard Linux	6
3.3 Official Kernel types.....	6
3.4 How to achieve hard real-time without modifying Linux kernel.....	8
3.4.1 Approach 1: Micro-kernel.....	8
3.4.2 Approach 2: Nano-kernel.....	9
[example]  Xenomai	9
4. Installation.....	11
4.1 Install Ubuntu	11
4.1.1 Download Ubuntu 16.04 LTS	11
4.1.2 Install Ubuntu 16.04 LTS.....	11
4.2 Before any modification	11
4.2.1 Use Timeshift to back up your whole disk.....	11
4.2.2 Use GRUB customizer to manager your kernel	12
4.2.3 Prepare for future test	13
4.3 Install the kernel	13
5. Test real-time performance.....	15
5.1 What is latency?.....	15
5.2 The performance of latency test on different kernels.....	15
6. References	18

1. Introduction

First of all, let's see what we are aiming to achieve.

Orcos RTT and ROS Integration: <https://www.youtube.com/watch?v=2-RhiPUcbGo>

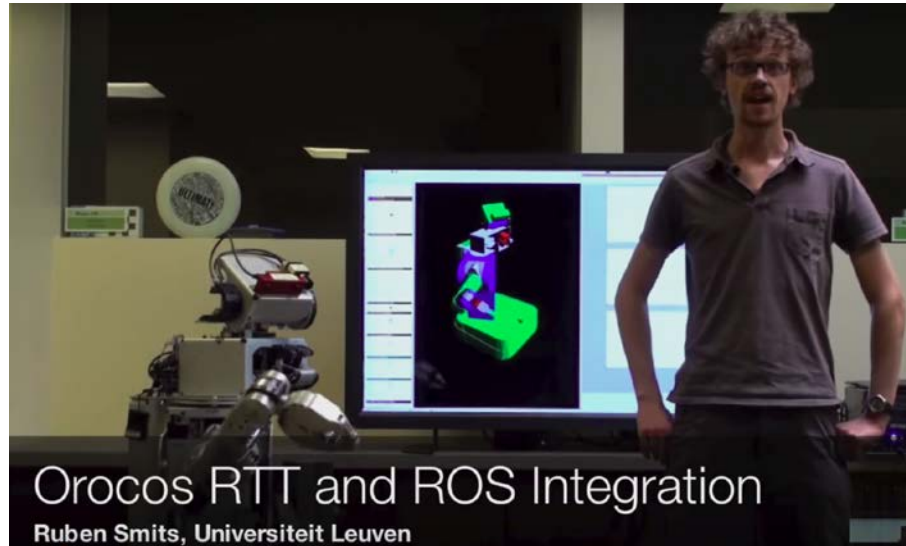


Figure 1.1 Screenshot of “Orcos RTT and ROS Integration”

There are some explanation about why they try to integrate Orcos_RTT and ROS on the following link. I copied their explanation here just in case this link may expire in the future.

<http://www.willowgarage.com/blog/2009/06/10/orocos-rtt-and-ros-integrated>

“Integrating ROS with other robot software frameworks, such as OpenRAVE, Player, and Euslisp, as been important to us as it allows developers to leverage the strengths of each. No framework can be the best at everything, so it's important to allow developers to choose the tools they need. This video shows the integration between the ROS and the Orocus Real Time Toolkit (RTT), which was done by Orocus developer Ruben Smits during a month-long visit to Willow Garage. Ruben became a vital member of our Milestone 2 team and still managed to have time to build this great, seamless integration.

While ROS has good support for distributing components over the network, RTT offers hard realtime communication between components. This demo combines the best of both worlds: the realtime low level control is handled by a set of RTT components that directly control the the PR2 robot hardware, but the communication between the controllers is visible to the whole ROS network.

The integration makes the RTT components appear as ROS components, without breaking the realtime communication between the RTT components. This allows RTT developers to use tools developed for ROS, such as rviz and rxplot, to visualize the communication between controllers.”

From their demonstration and explanation, you can see that ROS is not real-time and to make ROS real-time, you need to integrate Orocos RTT with it. However, is this enough?

ROS is an application, a software which is built upon the Ubuntu operating system. Even though you successfully patch ROS with Orocos RTT, you still cannot have real-time performance, because your operating system cannot perform real-time task!

I can give you a proof for my statement. Say, ROS is not real-time. But the developers has been trying to have a real-time ROS and so they launched the ROS2 which already has 2 stable editions. In one of their presentation about real-time ROS2, they said,

“First, you need to pick your operating system correctly. The default scheduler on Linux is not considered real-time safe. It uses a non-deterministic scheduler that optimized for fairness instead of determinism and priority.”

Besides, I copied one of their slides below to show you that it is required to “use an OS able to deliver the required determinism” to achieve real-time ROS2.

Requirements and best practices

Use an OS able to deliver the required determinism

– Linux variants

OS	real-time	max latency (μs)
Linux	no	10^4
RT PREEMPT	soft	10^1 - 10^2
Xenomai	hard	10^1

– **Proprietary:** e.g. QNX, VxWorks

POSIX compliant, **certified** to IEC 61508 SIL3 et.al.

Figure 1.2 it is required to “use an OS able to deliver the required determinism” to achieve real-time ROS2

Therefore, what I am trying to talk about in this documentation is how to make Ubuntu real-time. After that we can carry on to use this real-time Ubuntu to develop real-time ROS program.

2. What is real-time systems?

The first time you see the words “real-time performance”, you may think it suggest fast execution time which leads to better performance. That is correct in some way, otherwise there’s no point in achieving that. However, how fast is fast? Let me give you an accurate definition.

To be precise, it's all about **determinism**, the deterministic timing behavior! Given a reasonable deadline, it will strive to finish that before DDL.

- Correctness means execution at the correct time
- Missing the timeslot will lead to an error condition

Real-time systems in general are classified as hard real-time and soft real-time systems. Some may also say there is another Firm-time system.

-
- 1) **Hard** – missing a deadline is a total system failure.
 - 2) **Firm** – infrequent deadline misses are tolerable, but may degrade the system's quality of service. The usefulness of a result is zero after its deadline.
 - 3) **Soft** – the usefulness of a result degrades after its deadline, thereby degrading the system's quality of service.
-

2.1 Hard real-time systems

Hard real-time systems have strict timing requirements to meet deadlines. Missing a deadline is considered a system failure. Figure below shows the timing constraints in a hard real-time system using a time-utility function.

2.2 Soft Real-time Systems

For soft real-time systems, meeting a deadline is important, but it may not be catastrophic. Result becomes less useful after deadline, as the picture below shows. In addition, it is often related to Quality of Service.

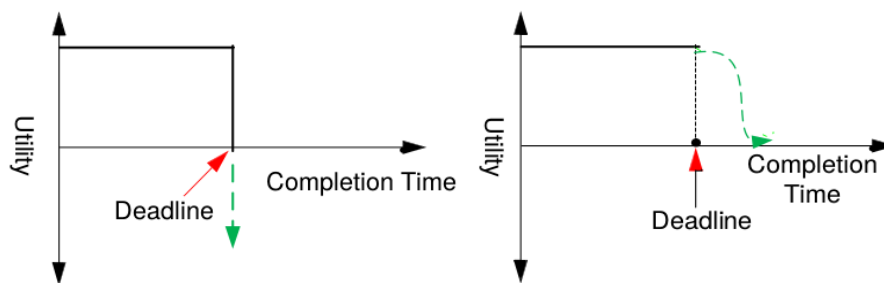


Figure 2.1 Hard real-time systems and Soft real-time systems

3. How to achieve real-time performance?

3.1 Linux kernel

The **kernel** is a computer program that is the core of a computer's operating system, with complete control over everything in the system. On most systems, it is one of the first programs loaded on start-up (after the bootloader).

```
[ 0.953693] serio: i8042 KBD port at 0x60,0x64 irq 1
[ 0.954816] serio: i8042 AUX port at 0x60,0x64 irq 12
[ 0.956069] mousedev: PS/2 mouse device common for all mice
[ 0.957743] input: AT Translated Set 2 keyboard as /devices/platform/i8042/ser
rio0/input/input0
[ 0.960144] rtc_cmos rtc_cmos: rtc core: registered rtc_cmos as rtc0
[ 0.961230] rtc0: alarms up to one day, 114 bytes nvram
[ 0.962307] cpuidle: using governor ladder
[ 0.963320] cpuidle: using governor menu
[ 0.964366] TCP cubic registered
[ 0.965316] NET: Registered protocol family 10
[ 0.967271] Mobile IPv6
[ 0.970938] NET: Registered protocol family 17
[ 0.972209] Registering the dns_resolver key type
[ 0.973334] Using IPI No-Shortcut mode
[ 0.974557] registered taskstats version 1
[ 0.976637] rtc_cmos rtc_cmos: setting system clock to 2011-09-09 20:32:52 UT
C (1315600372)
[ 0.982005] Initializing network drop monitor service
[ 0.983351] Freeing unused kernel memory: 404k freed
[ 0.984627] Write protecting the kernel text: 2768k
[ 0.985825] Write protecting the kernel read-only data: 1068k
[ 0.986935] NX-protecting the kernel data: 3376k
Loading, please wait...
```

Figure 3.1 Linux kernel 3.0.0 booting

It handles the rest of start-up as well as input/output requests from software, translating them into data-processing instructions for the central processing unit. It handles memory and peripherals like keyboards, monitors, printers, and speakers.

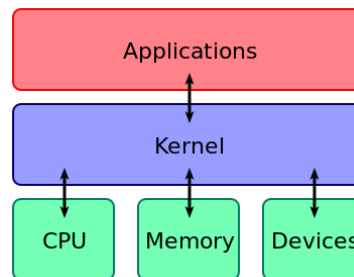


Figure 3.2 System structure

The Linux kernel is an open-source monolithic Unix-like computer operating system kernel. The Linux family of operating systems is based on this kernel and deployed on both traditional computer systems such as personal computers and servers, usually in the form of Linux distribution.

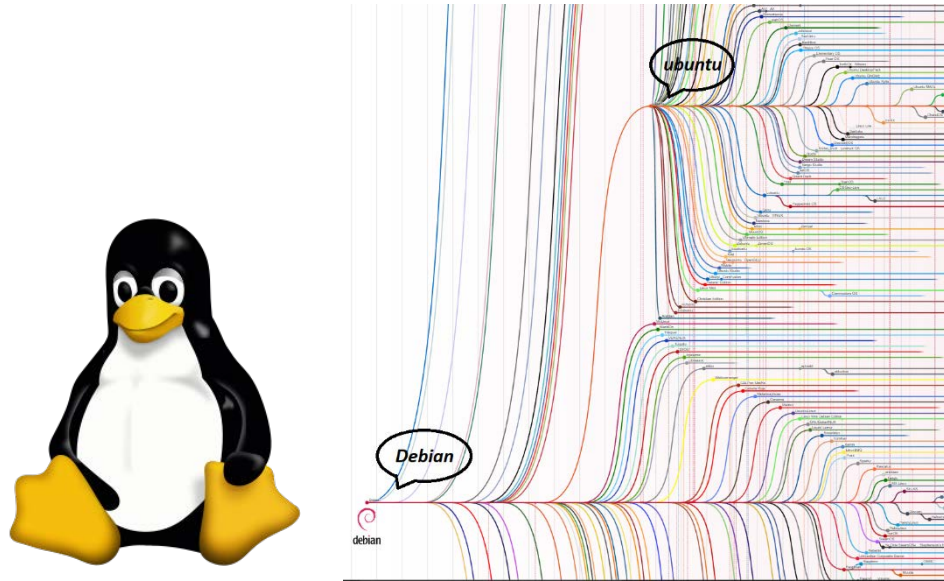


Figure 3.3 Linux logo and its distribution

As you can see from the picture above, Ubuntu is a child of debian. Therefore, if there is some installation that says for debian, you can use it safely.

3.2 Standard Linux

Linux is a standard time-sharing operating system.

- ☺ Good average performance and highly sophisticated services
- ☺ Hardware management layer dealing with event polling or processor/peripheral interrupts
- ☺ Scheduler classes dealing with process activation, priorities, time slice
- ☺ Communications between applications.

On the other hand, Linux is not a hard real-time operating system as it does not guarantee a task to meet strict deadlines.

- ☹ The kernel can suspend a task when its time slice has completed and it can remain suspended for an arbitrarily long time (for example, when an interrupt is getting serviced). And, the real-time task is not an exception.

3.3 Official Kernel types

First, let's see some official explanation on Ubuntu website.

<https://help.ubuntu.com/community/UbuntuStudio/RealTimeKernel>.

-generic kernel - this is the stock kernel that is provided by default in Ubuntu, which is non-realtime.

-preempt kernel - this kernel is based on the -generic kernel source tree but is built with different configurations (settings) to reduce latency. Also known as a soft

real-time kernel.

-rt kernel - is based on the Ubuntu kernel source¹ tree with Ingo Molnar maintained PREEMPT_RT patch applied to it. Also known as a hard real-time kernel. PREEMPT_RT patch converts Linux into a fully preemptible kernel!

-lowlatency kernel - very similar to the -preempt kernel and based on the -generic kernel source tree, but uses a more aggressive configuration to further reduce latency. Also known as a soft real-time kernel.

-realtime kernel - is based on the vanilla kernel source² tree with Ingo Molnar maintained PREEMPT_RT patch applied to it. Also known as a hard real-time kernel.

Personally, I like open source, because it is free and no worries about license. On the other hand, they are poorly maintained. You can see so much mismatching on their website. Their explanation about real-time kernel misled me for a long time.

<https://help.ubuntu.com/community/UbuntuStudio/RealTimeKernel>.

“At the moment, only the first three kernels are available through official Ubuntu archives.”

<https://wiki.ubuntu.com/RealTime>

“News: The -preempt and -rt kernels are no longer being developed due to lack of support. Focus has instead turned to the -lowlatency and -realtime kernels, particularly for the release of Ubuntu 11.04 Natty Narwhal. The long-term goal is to have -lowlatency in the official Ubuntu repositories, while maintaining -realtime in a dedicated PPA.”

Based on voluminous search I did on the internet; I must say I cannot agree with these statements. Instead, I list my understanding as follows. I cannot guarantee that they are true, but they really make a lot more sense and help me with the installation.

- ☐ **-preempt** kernel is no longer supported.
- ☐ **-lowlatency** kernel is already in the official Ubuntu repositories. Install -lowlatency kernel is extremely easy. Only two commands will be enough.
- ☐ **-rt kernel** is still supported. You can find the list from the link below.

¹ **Ubuntu kernel source** - Linux kernel source code used in Ubuntu and modified for offer all Ubuntu features.

² **vanilla kernel source** - Linux kernel source code than you can download at www.kernel.org.

<https://mirrors.edge.kernel.org/pub/linux/kernel/projects/rt/>

However, since this page is maintained by kernel.org, operated by the Linux Kernel Organization, Inc., I believe this kernel is for vanilla linux kernel instead of Ubuntu linux kernel. In other words, we should patch it to vanilla linux kernel from <https://mirrors.edge.kernel.org/pub/linux/kernel/>.

- ❑ **-realtime kernel** is a mystery. I don't know what it is and how to find it. Let's forget about it.

3.4 How to achieve hard real-time without modifying Linux kernel

The basic idea of making standard Linux hard real-time is that a small high-priority real-time kernel runs between the hardware and standard Linux.

The real-time tasks are executed by this real-time kernel (run to completion) and normal Linux processes are suspended during this duration. The real-time scheduler of the real-time kernel treats the standard Linux kernel as an idle task, which when given a chance to run executes its own scheduler to schedule normal Linux processes. But since the real-time kernel runs at a higher priority, the normal Linux processes can at any time be preempted by a real-time task. Interrupt management is another factor handled by the real-time kernel.

- 1) When an interrupt gets triggered during the execution of a real-time task, it is first received by the real-time kernel and stored.
- 2) When the real-time kernel is done, the interrupt is handed over to the standard Linux kernel.
 - ❑ If there is an associated real-time handler for the interrupt, it is executed by the real-time kernel.
 - ❑ Otherwise, if there are no more real-time tasks to run, the stored interrupt is passed to normal Linux.

3.4.1 Approach 1: Micro-kernel

A hard real-time kernel called micro-kernel is provided between the standard Linux kernel and the hardware. The Micro-kernel is responsible for intercepting all the hardware interrupts and ensures that normal Linux kernel cannot suspend high-priority interrupts or tasks currently running on the micro-kernel. Besides, the standard Linux kernel runs at a lower priority in the background of micro- kernel. The micro-kernel also comes with a RT scheduler that schedules higher-priority tasks with minimal latency (for example RTLinux or RTAI).

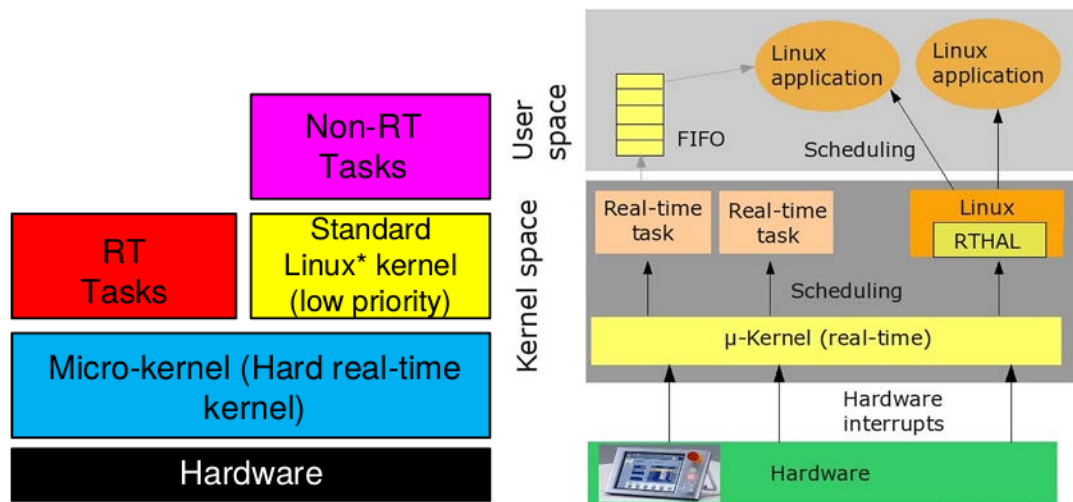


Figure 3.4 The structure of Micro-kernel Approach

3.4.2 Approach 2: Nano-kernel

The nano-kernel approach is similar to micro-kernel in that it also provides a real-time kernel layer between the standard Linux kernel and hardware, with standard Linux kernel running in the background as a low-priority task. Nano-kernel is basically a **Hardware Abstraction Layer (HAL)** that unlike micro-kernel, provides the facility to run multiple real-time and non-real-time operating systems on top of the nano-kernel (for example, ADEOS*).

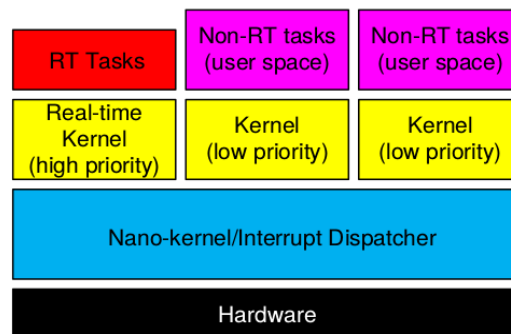


Figure 3.5 The structure of Nano-kernel Approach

[example]  Xenomai

Xenomai implements a **nano-kernel (ADEOS/I-pipe)** between the hardware and the Linux kernel. This nano-kernel is responsible for executing hard real-time tasks and intercepts interrupts, blocking them from reaching the Linux kernel, hence preventing the Linux kernel from preempting the hard real-time micro-kernel. Intrinsically, the Linux kernel executes in the background of this nano-kernel as a low-priority task that runs non-real-time tasks.

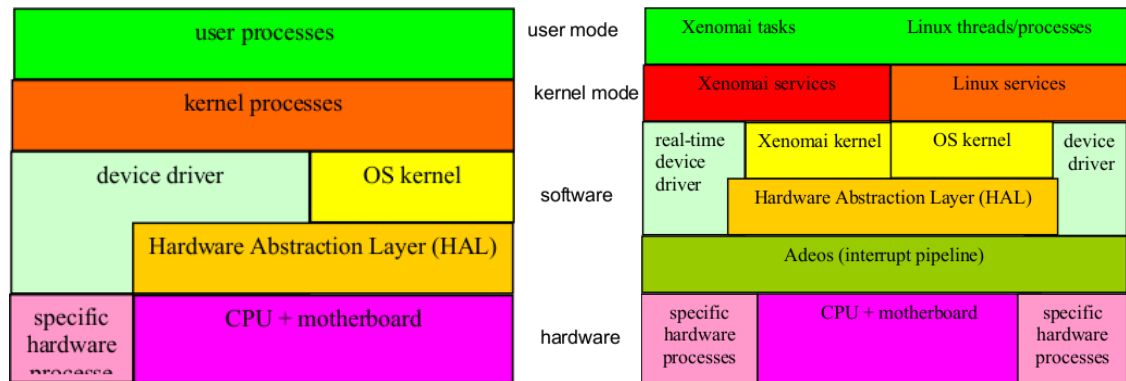


Figure 3.6 Linux Kernel V.S. Xenomai's nano-kernel Approach(ADEOS/I-pipe)

The Xenomai core, **Xenomai Nucleus**, is an abstract **RTOS Core** that provides operating-system resources and generic building blocks for building different RTOS interfaces called **skins**. These skins mimic the different RTOS APIs, allowing straightforward porting of existing applications to Xenomai. The building blocks form the Xenomai nucleus. Xenomai Nucleus makes Xenomai **real-time in user space**.

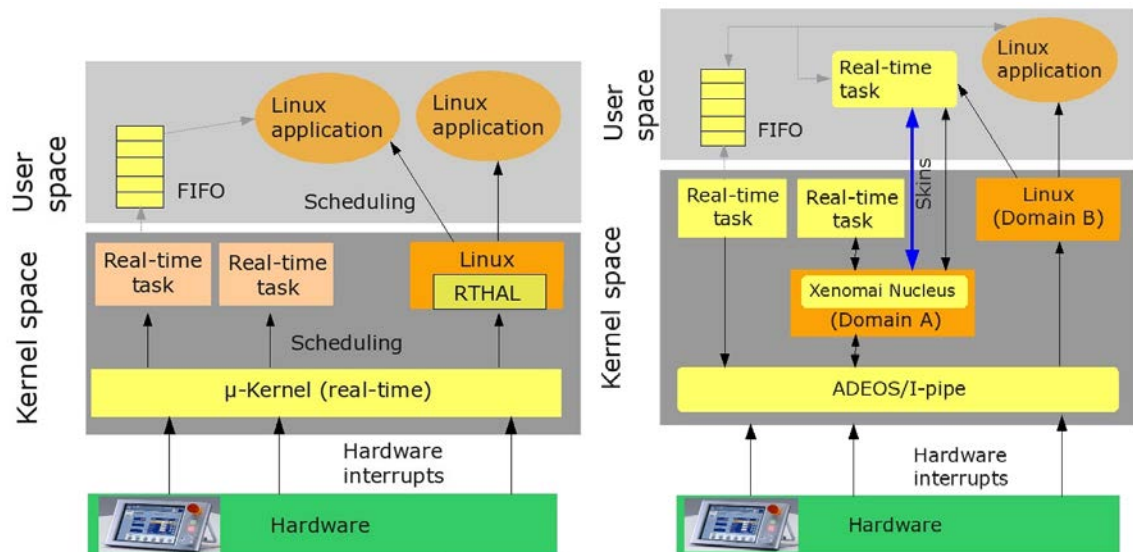


Figure 3.7 Micro-kernel approach VS. Xenomai

4. Installation

4.1 Install Ubuntu

Ubuntu 16.10 released with a support of 9 months until July 2017 and Ubuntu 16.04 has been released in wild by Canonical with a life circle of 5 years support.

4.1.1 Download Ubuntu 16.04 LTS

Download Ubuntu 16.04 LTS from [Ubuntu resources downloads:](https://www.ubuntu.com/download/alternative-downloads)
<https://www.ubuntu.com/download/alternative-downloads>

4.1.2 Install Ubuntu 16.04 LTS

If you want to install Ubuntu 16.04 LTS alongside with your Windows, please follow the instruction from [How to Install Ubuntu 16.10/16.04 Alongside With Windows 10 or 8 in Dual-Boot:](https://www.tecmint.com/install-ubuntu-16-04-alongside-with-windows-10-or-8-in-dual-boot/)
<https://www.tecmint.com/install-ubuntu-16-04-alongside-with-windows-10-or-8-in-dual-boot/>

- This guide assumes that your machine comes pre-installed with **Windows 10 OS or an older version of Microsoft Windows, such as Windows 8.1 or 8.**
- In case your hardware uses UEFI then you should **modify the EFI settings and disable Secure Boot feature. Please be sure to do this, or you won't be able to achieve dual-boot.**
- Burn the image to a DVD or create a bootable USB stick using a utility such as **Universal USB Installer (BIOS compatible)** or **Rufus (UEFI compatible).** **Please be sure to choose the one that compatible with your computer.**
- If your computer has no other Operating System already installed and you plan to use a Windows variant alongside Ubuntu 16.04/16.10, you should **first install Microsoft Windows and then proceed with Ubuntu 16.04 installation.**
- In this particular case, on Windows installation steps, when formatting the hard disk, you should allocate a free space on the disk with at least **20 GB** in size in order use it later as a partition for Ubuntu installation.

If you just want to install Ubuntu on your computer without Windows, read articles from [Ubuntu 16.04 Desktop Installation Guide](#)

4.2 Before any modification

4.2.1 Use Timeshift to back up your whole disk

Like Richard Hendricks in “Silicon Valley” once said,



In Ubuntu, it is so easy to install package, yet it is extremely difficult to uninstall something. If you don't want your system to crash down, don't forget to back up your whole disk. **Timeshift** is a great tool to help you backup and restore. You can use the following command to install it.

```
sudo apt-add-repository -y ppa:teejee2008/ppa
sudo apt-get update
sudo apt-get install timeshift
```

Please open the following link to see more details.

<https://itsfoss.com/backup-restore-linux-timeshift/>

I highly recommend you back up your system after a new kernel has been installed successfully. Trust me, I've been there ☹.

4.2.2 Use GRUB customizer to manager your kernel

GNU GRUB (short for **GNU GR**and **Unified Boot**loader) is a boot loader package from the GNU Project. GRUB provides a user the choice to boot one of multiple operating systems installed on a computer or select a specific kernel configuration available on a particular operating system's partitions.

GRUB customizer can help us see our kernel list and manage our GRUB easily. Here's how to install it.

```
sudo add-apt-repository ppa:danielrichter2007/grub-customizer
sudo apt-get update
sudo apt-get install grub-customizer
```

Run GRUB customizer. You can see your current kernel list under the directory of "advanced option for Ubuntu system".

We also need to do the following grub configuration ourselves.

Edit the grub config :

```
sudo nano /etc/default/grub
```

We need to change first three lines.

```
GRUB_DEFAULT="Advanced options for Ubuntu"
# Comment the following lines
```

```
#GRUB_HIDDEN_TIMEOUT=0
#GRUB_HIDDEN_TIMEOUT_QUIET=true
GRUB_DISTRIBUTOR='lsb_release -i -s 2> /dev/null || echo Debian'
GRUB_TIMEOUT=10
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
GRUB_CMDLINE_LINUX=""
```

Finally, we need to update our GRUB configuration and reboot!

```
sudo update-grub
sudo reboot
```

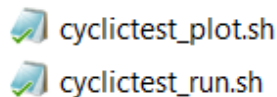
After reboot, see whether you can make selection for your kernel!

4.2.3 Prepare for future test

After future installation, we will test its real-time performance. Here's some preparation.

```
sudo -s
apt-get install rt-tests
apt-get install gnuplot
```

There are two files that help you gather test data and visualize it. There are in the "test_files" folder. Copy them to your home directory, so it is much easier to run them.



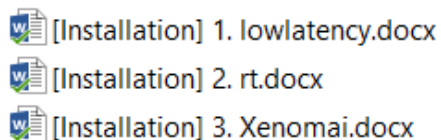
We need to use following command to make these files executable.

```
chmod +x cyclicttest_run.sh
chmod +x cyclicttest_plot.sh
```

The test preparation is finished.

4.3 Install the kernel

There are three types of kernel, the linux-lowlatency one, the linux-rt kernel and the Xenomai patched linux kernel. I have written three files for them respectively.



I suggest that you should follow the order.

- 1) Install linux-lowlatency kernel first, since it is the easiest one. After installation, test its real-time performance.
- 2) Install linux-rt kernel. It is officially supported. In addition, it gives the best performance in my test. After installation, test its real-time performance. See whether it meets hard-real time criterion.
- 3) If you want, you can try to install Xenomai patched kernel. But to be frank, its performance is

worse than the linux-lowlatency kernel. It is against our knowledge that hard real-time system can be beaten by soft real-time system. Maybe it is because xenomai kernel lacks proper maintenance. Actually, the installation guide I found is not even on their website!

5. Test real-time performance

5.1 What is latency?

Latency is the time taken from external interrupt till a process to react to the interrupt.

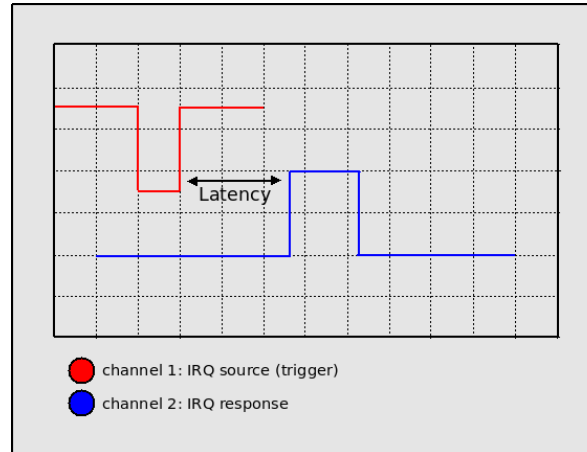


Figure 5.1 Latency

The picture below shows what make up the kernel latency. The kernel latency consists of interrupt latency, handler duration, scheduler latency and scheduler duration.

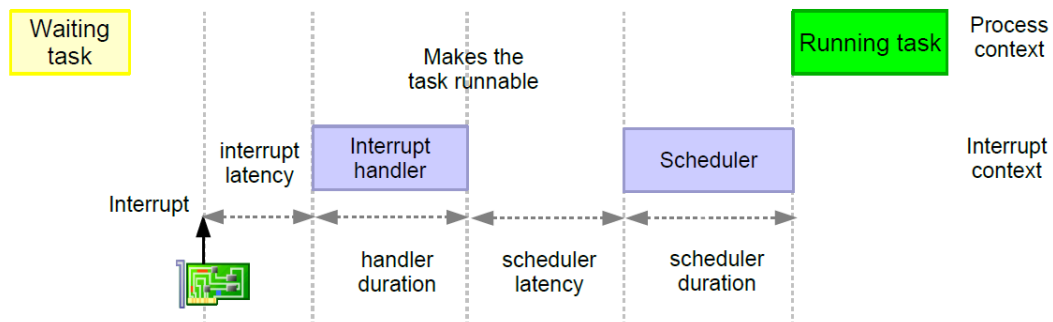


Figure 5.2 The composition of kernel latency

5.2 The performance of latency test on different kernels

If your installation was correct, and you have finished your latency test. You can compare their performance. I run the latency test on 4 different configurations on two computers respectively, and the result are as follows.

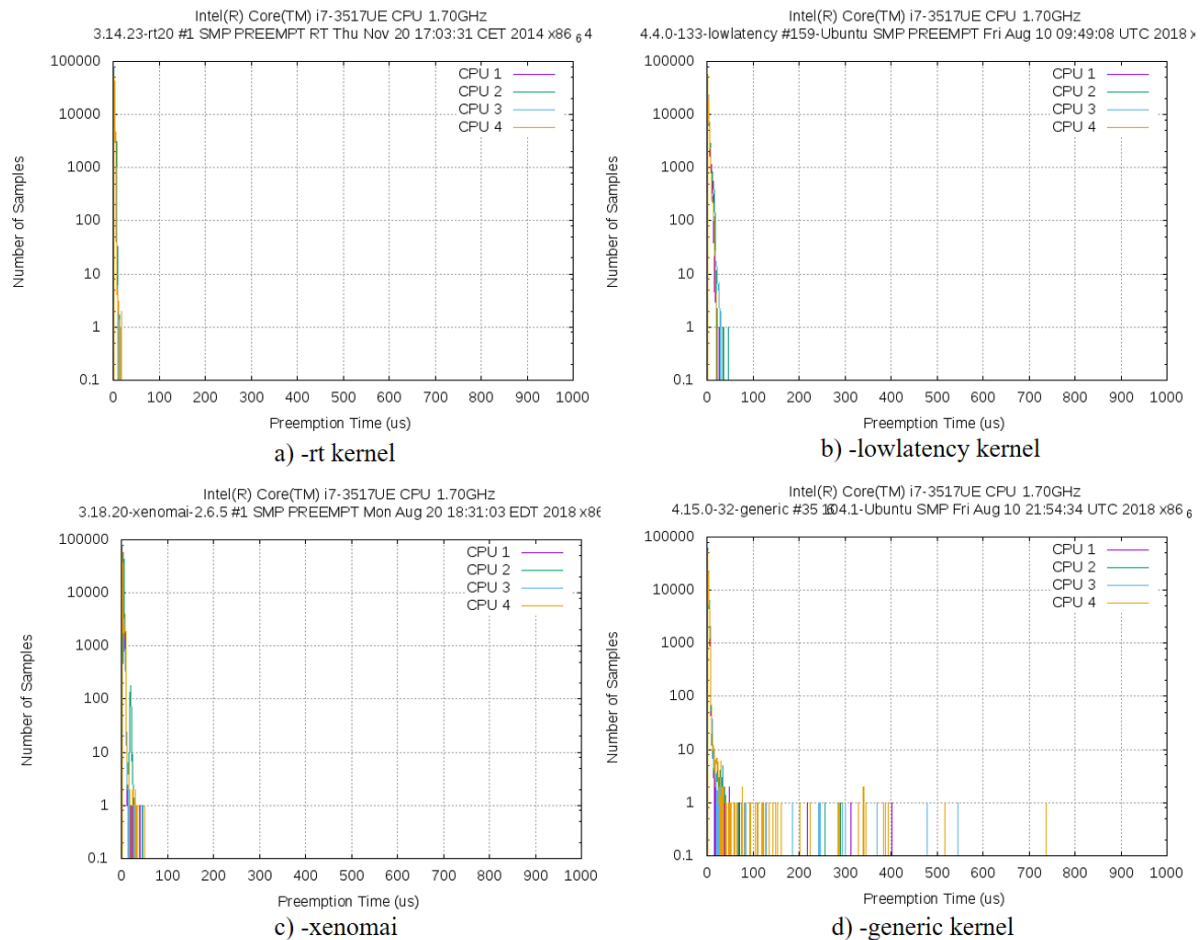


Figure 5.3 latency test results on SECURECORE TIANO (The ordinate is of logarithmic scale)

- d) -generic kernel : most of the latency is below 50 us, but there are still some of the test beyond that.
- c) -Xenomai : all of the latency data is below 50 us.
- b) -lowlatency kernel : most of the latency is below 20us, but there are still some beyond that and they never exceed 50us.
- a) -rt kernel : all of the latency data is below 20 us. It gives the best performance!

In a nutshell, the comparison of these kernels are as follows.

- rt kernel (hard real-time, needs a little configuration to install)
- > -lowlatency kernel (soft real-time, extremely easy to install)
- > -xenomai (hard real-time supposedly, installation takes a long time and errors may still exist)
- > -generic (default one, you all have this)

Therefore, as I said, your installation order should be

[-lowlatency] -> [-rt] -> [xenomai/ or not]

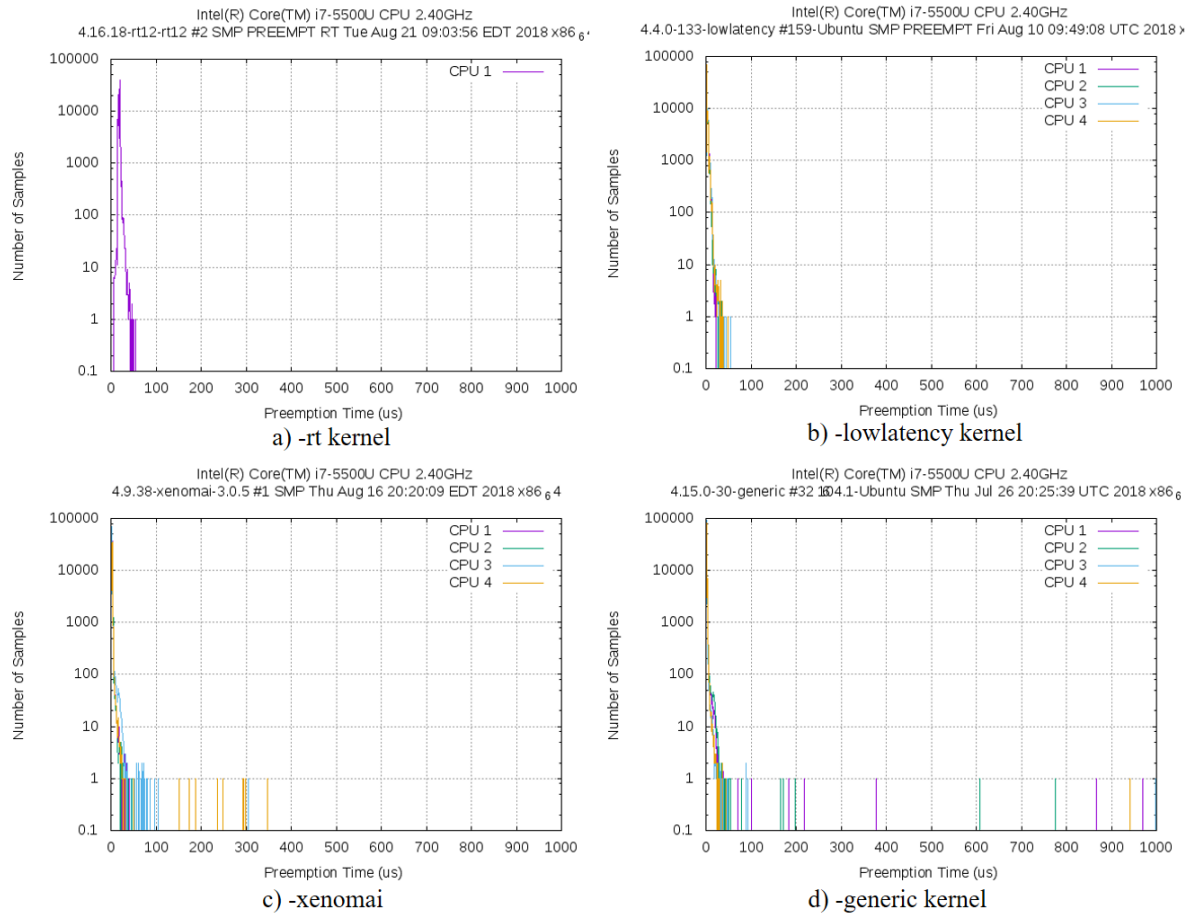


Figure 5.4 latency test results on Dell Vostro 5480 (The ordinate is of logarithmic scale)

This test on my computer gave the same result. Besides, it also shows that SECURECORE TIANO is a really good computer, doing real-time task better than mine!

6. References

- [1]. https://en.wikipedia.org/wiki/Real-time_computing
- [2]. <http://www.willowgarage.com/blog/2009/06/10/orocos-rtt-and-ros-integrated>
- [3]. <https://www.youtube.com/watch?v=2-RhiPUcbGo>
- [4]. <https://answers.ros.org/question/206052/making-ros-real-time-is-it-magic/>
- [5]. <https://answers.ros.org/question/217131/standard-way-to-use-ros-with-real-time-os/>
- [6]. <https://answers.ros.org/question/61162/set-up-for-real-time-control-in-ros/>
- [7]. [https://en.wikipedia.org/wiki/Kernel_\(operating_system\)](https://en.wikipedia.org/wiki/Kernel_(operating_system))
- [8]. <https://xenomai.org/>
- [9]. https://gitlab.denx.de/Xenomai/xenomai/wikis/Supported_Hardware#x86
- [10]. https://gitlab.denx.de/Xenomai/xenomai/wikis/Configuring_For_X86_Based_Dual_Kernels
- [11]. <http://caxapa.ru/thumbs/833391/multicore-real-time-linux-xenomai-paper.pdf>
- [12]. <https://rtt-lwr.readthedocs.io/en/latest/rtpc/xenomai3.html>
- [13]. <https://roscon.ros.org/2015/presentations/RealtimeROS2.pdf>