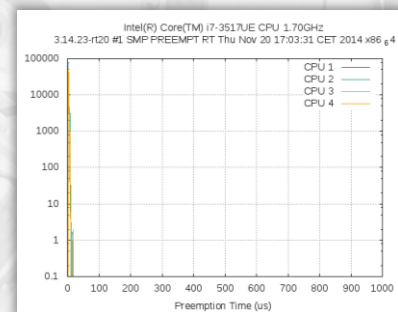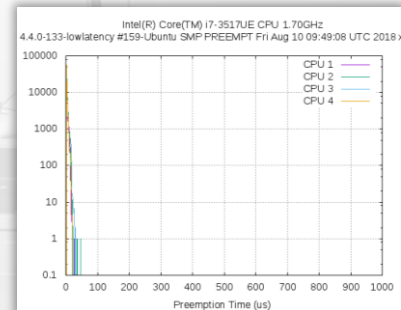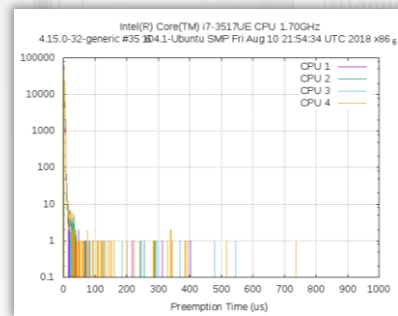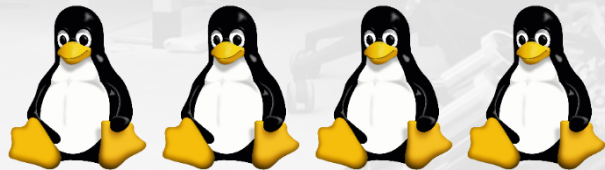# Installation guide

for

real-time

Linux kernel

Kaixian Qu

Email Address: QKX515@gmail.com

Supervisor: Prof. Alexander Leonessa

Terrestrial Robotics Engineering & Controls Lab

TREC
Terrestrial Robotics
Engineering & Controls

Please finish the "Real-time implementation for humanoid robots" before you modify your kernel
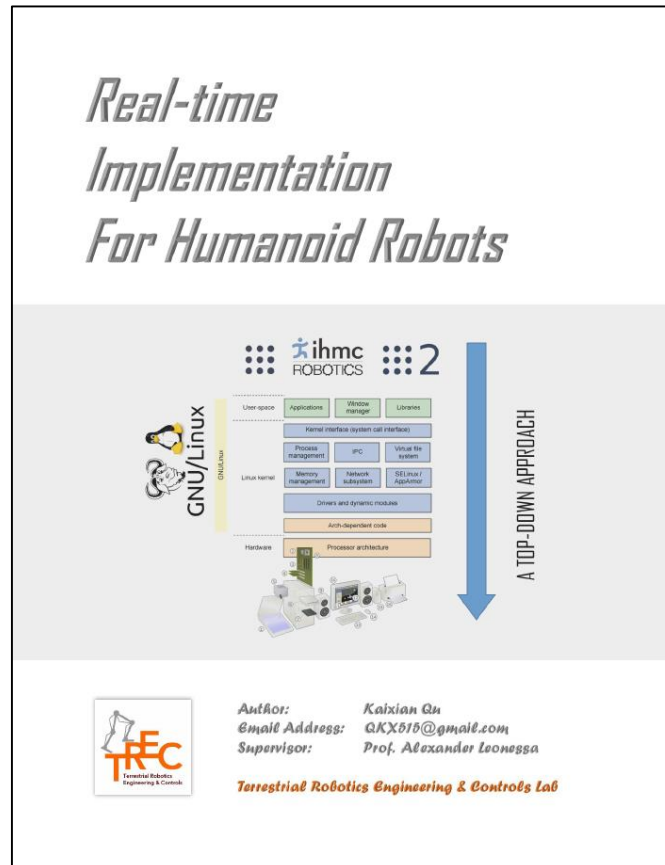
Real-time Implementation For Humanoid Robots

Author: Kaixian Qu
Email Address: QKX515@gmail.com
Supervisor: Prof. Alexander Leonessa

Terrestrial Robotics Engineering & Controls Lab

All the files about this part are hosted on https://github.com/TREC-LAB/real-time-linux-kernel. Please go see it online. Having a local copy is not a good idea as the online edition will always be the newest version.

1. **Install Ubuntu**

As we said before, we will use Ubuntu as our operating system. When we install Ubuntu, we will get a Linux kernel by default. If you already have Ubuntu in your computer, please skip this part. If you don't have Ubuntu, please read the following.

If you want to install Ubuntu without losing Windows, please see this following link.
https://www.tecmint.com/install-ubuntu-16-04-alongside-with-windows-10-or-8-in-dual-boot/

- Even though they used Ubuntu 16.04 in their instruction, you can always choose another edition, like 18.04, 14.04. Their instruction always works like a charm.

- 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 https://www.tecmint.com/ubuntu-16-04-installation-guide/

2. **Before installation**

2.1.1 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. :-(

2.1.2 Manage our Grub

GNU GRUB (short for **GNU GR**and **U**nified **B**ootloader) 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! -
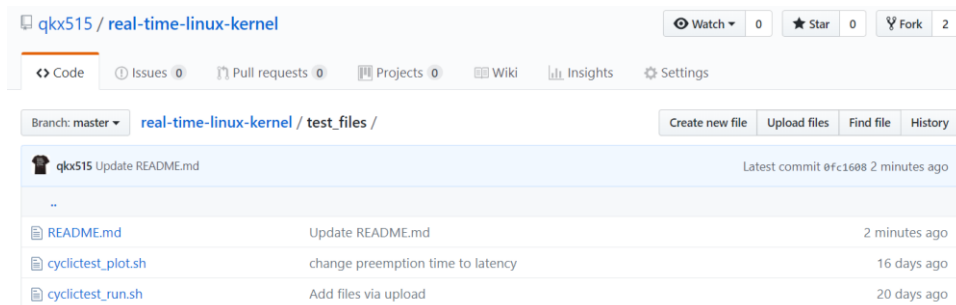
```
sudo update-grub

sudo reboot
```

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

2.1.3 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 gnuplot
```

There are two files that help you gather test data and visualize it, which are in the "test_files" folder.

Use the following command to download these files to your current folder

```
wget https://raw.githubusercontent.com/qkx515/real-time-linux-kernel/master/test_files/cyclictest_plot.sh

wget https://raw.githubusercontent.com/qkx515/real-time-linux-kernel/master/test_files/cyclictest_run.sh
```

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

```
chmod +x cyclictest_run.sh

chmod +x cyclictest_plot.sh
```

The test preparation is finished.

3. -lowlatency kernel

-lowlatency kernel is already in the official Ubuntu repositories. Therefore, it is extremely easy to install -lowlatency kernel.

3.1 Find them in the official Ubuntu repositories

```
apt-cache search linux-lowlatency
```

```
kai@heater:~$ apt-cache search linux-lowlatency
linux-image-4.4.0-21-lowlatency - Linux kernel image for version 4.4.0 on 64 bit x86 SMP
linux-lowlatency - Complete lowlatency Linux kernel
linux-lowlatency-lts-utopic - Complete lowlatency Linux kernel (dummy transitional package)
linux-lowlatency-lts-vivid - Complete lowlatency Linux kernel (dummy transitional package)
linux-lowlatency-lts-wily - Complete lowlatency Linux kernel (dummy transitional package)
linux-lowlatency-lts-xenial - Complete lowlatency Linux kernel (dummy transitional package)
linux-image-4.10.0-14-lowlatency - Linux kernel image for version 4.10.0 on 64 bit x86 SMP
linux-image-4.10.0-19-lowlatency - Linux kernel image for version 4.10.0 on 64 bit x86 SMP
linux-image-4.10.0-20-lowlatency - Linux kernel image for version 4.10.0 on 64 bit x86 SMP
linux-image-4.10.0-21-lowlatency - Linux kernel image for version 4.10.0 on 64 bit x86 SMP
linux-image-4.10.0-22-lowlatency - Linux kernel image for version 4.10.0 on 64 bit x86 SMP
linux-image-4.10.0-24-lowlatency - Linux kernel image for version 4.10.0 on 64 bit x86 SMP
linux-image-4.10.0-26-lowlatency - Linux kernel image for version 4.10.0 on 64 bit x86 SMP
linux-image-4.10.0-27-lowlatency - Linux kernel image for version 4.10.0 on 64 bit x86 SMP
linux-image-4.10.0-28-lowlatency - Linux kernel image for version 4.10.0 on 64 bit x86 SMP
linux-image-4.10.0-30-lowlatency - Linux kernel image for version 4.10.0 on 64 bit x86 SMP
```

```
apt-cache search linux-headers-lowlatency
```

```
kai@heater:~$ apt-cache search linux-headers-lowlatency
linux-headers-lowlatency - lowlatency Linux kernel headers
linux-headers-lowlatency-lts-utopic - lowlatency Linux kernel headers (dummy transitional package)
linux-headers-lowlatency-lts-vivid - lowlatency Linux kernel headers (dummy transitional package)
linux-headers-lowlatency-lts-wily - lowlatency Linux kernel headers (dummy transitional package)
linux-headers-lowlatency-lts-xenial - lowlatency Linux kernel headers (dummy transitional package)
linux-headers-lowlatency-hwe-16.04 - lowlatency Linux kernel headers
linux-headers-lowlatency-hwe-16.04-edge - lowlatency Linux kernel headers
kai@heater:~$
```

Both packages we need are in the official repositories. Great!

3.2 Install them

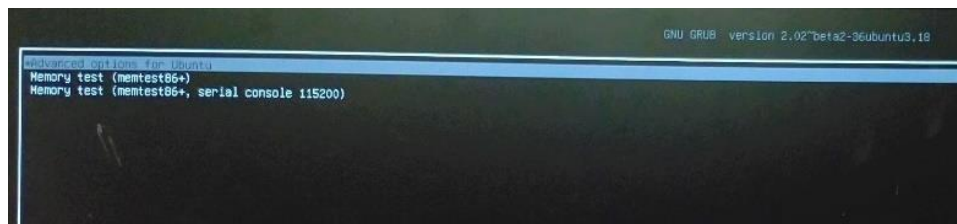Two commands below will be enough.

```
sudo apt-get update
sudo apt-get install linux-lowlatency linux-headers-lowlatency
```
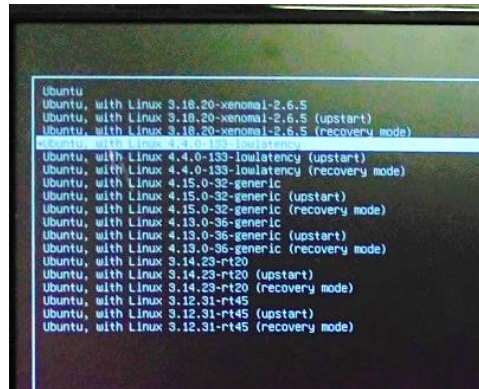
3.3 Reboot Ubuntu system

```
sudo reboot
```

After reboot, select "Advanced options for Ubuntu". See the picture below.

Under this directory, you should be able to see one option end with "-lowlatency". Use ↑ ↓ to select it and press "ENTER" key. Now, wait for it to finish the rebooting.

After the reboot, let's see your current kernel with the following command.

```
uname -a
```


If it matches the one you chose, then your installation succeeds! Congratulations!

3.4 Test your latency

To test your latency, first make sure that you have cyclictest_run.sh and cyclictest_plot.sh and made them executable using chmod +x command.

Now let's run it.

```
sudo -s #we need root permission

./cyclictest_run.sh 100000 > result
```

Wait for a few minutes. This command creates a file named "result" containing the testing result. Then,

```
./cyclictest_plot.sh result
```

This command visualizes your data and put it in result.png in your current directory.

The following picture shows the latency test result for -lowlatency kernel.
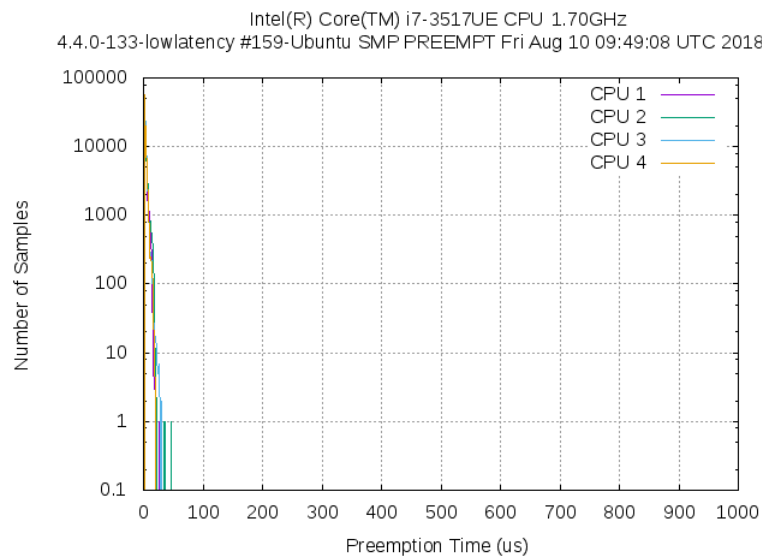
And you can compare it to latency result for -generic kernel
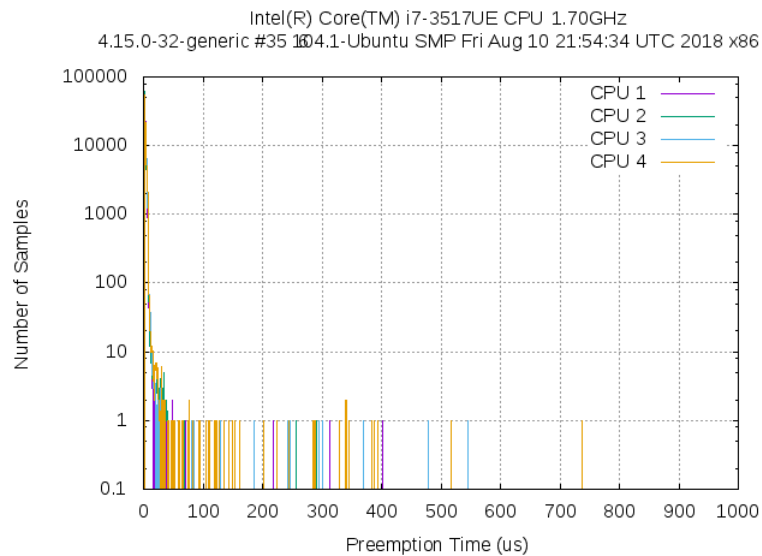
The preemption time here actually stands for latency. As you can see, the latency dropped dramatically!

4. –rt kernel

"Controlling a laser with Linux is crazy, but everyone in this room is crazy in his own way. So if you want to use Linux to control an industrial welding laser, I have no problem with your using Preempt RT"

- Linus Torvalds (Founder of Linux)

Before we get started, please make sure that your current kernel is –generic kernel. If not, please reboot and select –generic kernel.

Let's change to super user to avoid future permission issues.

```
sudo -s
```

4.1 Get the sources

First, we need to find our patch and the compatible Linux kernel edition.

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

⬇

Index of /pub/linux/kernel/projects/rt/

```
../
2.6.22/                                08-Aug-2013 18:24       -
2.6.23/                                08-Aug-2013 18:26       -
2.6.24/                                08-Aug-2013 18:27       -
2.6.25/                                08-Aug-2013 18:27       -
2.6.26/                                08-Aug-2013 18:28       -
2.6.29/                                08-Aug-2013 18:28       -
2.6.31/                                04-Nov-2014 14:19       -
2.6.33/                                08-Aug-2013 18:29       -
3.0/                                   19-Nov-2013 22:02       -
3.10/                                  23-Nov-2017 05:44       -
3.12/                                  08-Jun-2017 13:40       -
3.14/                                  13-Feb-2017 22:26       -
3.18/                                  17-Aug-2018 04:15       -
3.2/                                   23-Nov-2017 05:53       -
3.4/                                   16-Nov-2016 19:26       -
3.6/                                   19-Nov-2013 22:01       -
3.8/                                   04-Nov-2014 13:35       -
4.0/                                   13-Jul-2015 21:06       -
4.1/                                   29-Nov-2017 22:12       -
4.11/                                  17-Oct-2017 13:42       -
4.13/                                  17-Nov-2017 17:03       -
4.14/                                  31-Jul-2018 21:12       -
4.16/                                  03-Aug-2018 07:39       -
4.18/                                  07-Aug-2018 14:27       -
4.4/                                   16-Aug-2018 20:09       -
4.6/                                   30-Sep-2016 21:37       -
4.8/                                   23-Dec-2016 15:26       -
4.9/                                   06-Aug-2018 09:04       -
```

⬇

⬇

⬇

⬇

↓

Index of /pub/linux/kernel/projects/rt/4.16/

```
../
incr/                                            03-Aug-2018 07:39      -
older/                                           03-Aug-2018 07:39      -
patch-4.16.18-rt12.patch.gz                      03-Aug-2018 07:39    292K
patch-4.16.18-rt12.patch.sign                    03-Aug-2018 07:39    566
patch-4.16.18-rt12.patch.xz                      03-Aug-2018 07:39    225K
patches-4.16.18-rt12.tar.gz    Open link in new tab              455K
patches-4.16.18-rt12.tar.si                                           566
patches-4.16.18-rt12.tar.xz    Open link in new window           329K
sha256sums.asc                                                       1261
                               Open link in incognito window

                               Save link as...
                               Copy link address

                               Inspect              Ctrl+Shift+I
```

```
    wget https://mirrors.edge.kernel.org/pub/linux/kernel/projects/rt/4.16/patch-
4.16.18-rt12.patch.xz
```

After downloading our patch, we need to download the Linux kernel which goes with our patch – **linux-4.16.18.**

https://mirrors.edge.kernel.org/pub/linux/kernel/v4.x/

↓

```
linux-4.16.16.tar.gz                   16-Jun-2018 07:46    152M
linux-4.16.16.tar.sign                 16-Jun-2018 07:46    833
linux-4.16.16.tar.xz                   16-Jun-2018 07:46    98M
linux-4.16.17.tar.gz                   20-Jun-2018 19:04    152M
linux-4.16.17.tar.sign                 20-Jun-2018 19:04    833
linux-4.16.17.tar.xz                   20-Jun-2018 19:04    98M
linux-4.16.18.tar.gz                   25-Jun-2018 23:59    152M
linux-4.16.18.tar.sign                 25-Jun-2018 23:59    833
linux-4.16.18.tar.xz                   25-Jun-2018 23:59    98M
linux-4.16.2.tar      Open link in new tab          2018 10:33    152M
linux-4.16.2.tar                                    2018 10:33    833
linux-4.16.2.tar      Open link in new window       2018 10:33    98M
linux-4.16.3.tar      Open link in incognito window 2018 06:57    152M
linux-4.16.3.tar                                    2018 06:57    833
linux-4.16.3.tar                                    2018 06:57    98M
linux-4.16.4.tar      Save link as...               2018 07:46    152M
linux-4.16.4.tar      Copy link address             2018 07:46    833
linux-4.16.4.tar                                    2018 07:46    98M
linux-4.16.5.tar                                    2018 09:03    152M
linux-4.16.5.tar      Inspect          Ctrl+Shift+I 2018 09:03    833
linux-4.16.5.tar.xz                    26-Apr-2018 09:03    98M
linux-4.16.6.tar.gz                    29-Apr-2018 19:47    152M
linux-4.16.6.tar.sign                  29-Apr-2018 19:47    833
linux-4.16.6.tar.xz                    29-Apr-2018 19:47    98M
linux-4.16.7.tar.gz                    02-May-2018 15:04    152M
```

```
    wget https://mirrors.edge.kernel.org/pub/linux/kernel/v4.x/linux-4.16.18.tar.xz
```

After downloading, we need to unpack the archives and patch the Linux kernel.

```
xz -cd linux-4.16.18.tar.xz | tar xvf -

cd linux-4.16.18

xzcat ../patch-4.16.18-rt12.patch.xz | patch -p1
```

4.2 Configure the kernel

There are some packages required to help us configure the kernel.

```
sudo apt-get install qt5-default libssl-dev libelf-dev

sudo apt-get install bison flex
```

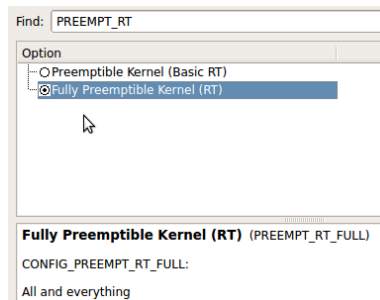Take the actual working config, which is the configuration for generic kernel:

```
yes "" | make oldconfig
```

Change your configuration.

```
make xconfig
```

The only necessary configuration for real-time Linux kernel is the choice of the "Fully Preemptible Kernel" preemption model (CONFIG_PREEMPT_RT_FULL). All other kernel configuration parameters depend on system requirements. For detailed information about how to configure a kernel have a look at Linux kernel documentation.

CONFIG_PREEMPT_RT_FULL (Enable)

Find: PREEMPT_RT

Option
- ○ Preemptible Kernel (Basic RT)
- ◉ Fully Preemptible Kernel (RT)

Fully Preemptible Kernel (RT) (PREEMPT_RT_FULL)

CONFIG_PREEMPT_RT_FULL:

All and everything

Recommendation Configuration **(if you cannot change some configuration, just forget about it)**

* General setup
   --> Timers subsystem
       --> High Resolution Timer Support (Enable)
* Power management and ACPI options
   --> ACPI (Advanced Configuration and Power Interface) Support
       --> Processor (Disable)
   --> CPU Frequency scaling
       --> CPU Frequency scaling (Disable)
   --> CPU Idle
       --> CPU idle PM support (Disable)
* Processor type and features
   --> Enable maximum number of SMP processors and NUMA nodes (Disable)
   --> Processor family
       --> Core 2/newer Xeon   if "cat /proc/cpuinfo | grep family" returns 6,
       --> set as Generic   otherwise
   --> Transparent Hugepage Support (Disable)
   --> Allow for memory compaction (Disable)
   --> Contiguous Memory Allocation (Disable)
   --> Allow for memory compaction
     --> Page Migration (Disable)

* Device Drivers
   --> Staging drivers
       --> Unisys SPAR driver support
           --> Unisys visorbus driver (Disable)

Save configuration `CTRL`+`S`**.**
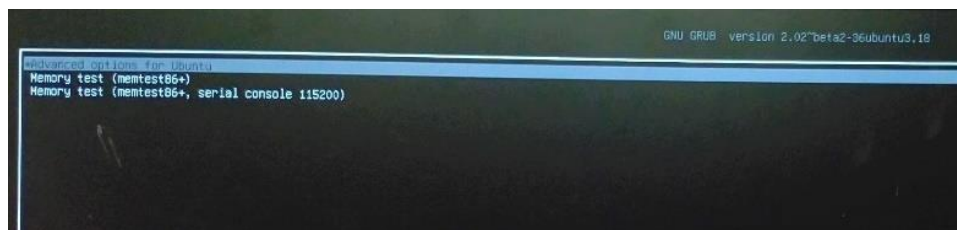
4.3 Build the kernel

"-j4" is for my quad-core CPU. This can take a long time.

```
make –j4

make –j4 modules

make –j4 modules_install

make -j4 install
```
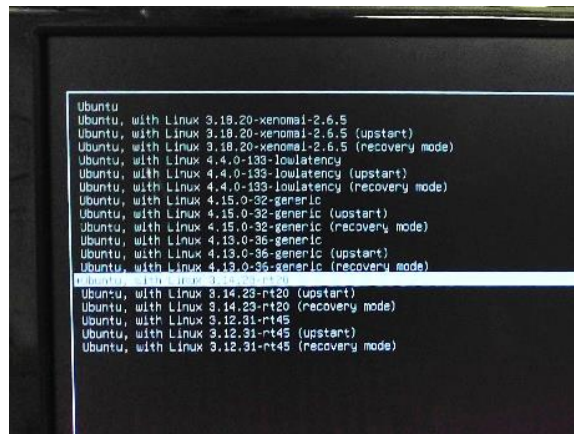
4.4 Reboot Ubuntu system

```
sudo reboot
```

After reboot, select "Advanced options for Ubuntu". See the picture below.

Under this directory, you should be able to see one option end with "–rtXX". Use `↑` `↓` to select it and press "ENTER" key. Now, wait for it to finish the rebooting.

After the reboot, let's see your current kernel with the following command.

```
uname -a
```


If it matches the one you chose, then your installation succeeds! Congratulations！

4.5 Test your latency

To test your latency, first make sure that you have cyclictest_run.sh and cyclictest_plot.sh and made them executable using chmod + x command.

Now let's run it.

```
sudo -s #we need root permission
./cyclictest_run.sh 100000 > result
```

Wait for a few minutes. This command creates a file named "result" containing the testing result. Then,
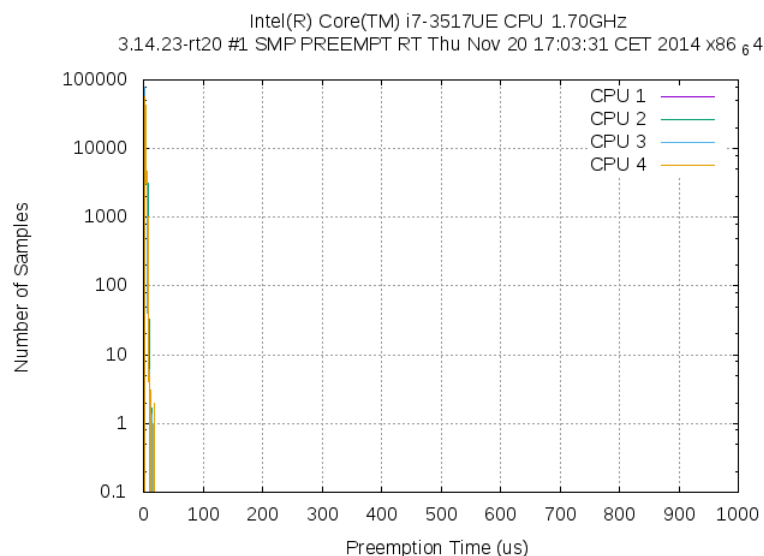
```
./cyclictest_plot.sh result
```

This command visualizes your data and put it in result.png in your current directory.

```
root@heater:~# ./cyclictest_run.sh 100000 > result
0.52user 1.20system 0:25.07elapsed 6%CPU (0avgtext+0avgdata 35332maxresident)k
104inputs+72outputs (1major+8890minor)pagefaults 0swaps
root@heater:~# ./cyclictest_plot.sh result
./cyclictest_plot.sh: line 59: [: -lt: unary operator expected
./cyclictest_plot.sh: line 64: [: -l: integer expression expected
CPUS: 4  Title:  Intel(R) Core(TM) i7-3517UE CPU @ 1.70GHz Kernel:  3.14.23-rt20
 #1 SMP PREEMPT RT Thu Nov 20 17:03:31 CET 2014 x86_64 L-Max:  X-Max Y-Max
Drawing ...

        Rectangular grid drawn at x y tics
        Major grid drawn with lt 0 linewidth 0.500
        Minor grid drawn with lt 0 linewidth 0.500
        Grid drawn at default layer

Histogram created: result.png
0
```
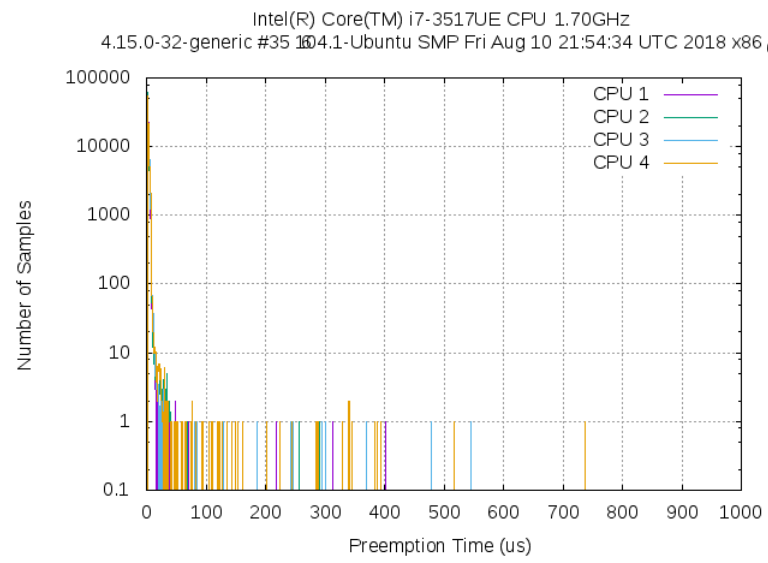
The following picture shows the latency test result for –rt kernel.

And you can compare it to latency result for –generic kernel

Intel(R) Core(TM) i7-3517UE CPU 1.70GHz
4.15.0-32-generic #35 104.1-Ubuntu SMP Fri Aug 10 21:54:34 UTC 2018 x86_6

The preemption time here actually stands for latency. As you can see, the latency dropped dramatically!

5. ~~Install Xenomai (Not recommended)~~

If this installation guide fails, please go check these 2 websites to see their guides.

https://rtt-lwr.readthedocs.io/en/latest/rtpc/xenomai3.html

https://code.google.com/archive/p/atrias/wikis/XenomaiSetup.wiki

---

5.1 Check your hardware

Hardware is a big issue in this installation. Since this is really low-level thing, it does not provide support for many different hardwares.

- ☑ **Recommended Hardware**
    - ☑ **Intel/AMD Processor i5/i7** (2016 is recommended to guarantee full 16.04 support)
    - ☑ **Dedicated Ethernet controller for RTnet**, with either e1000e/e1000/r8169 driver (Intel PRO/1000 GT recommended)
- ☒ **Warning**
    - ☒ **Nvidia/Ati Drivers** are **NOT** supported (creates a lot of interruptions that breaks the real-time constraints). Please consider removing the dedicated graphic card and use the integrated graphics (Intel HD graphics)

5.2 Get a Linux kernel compatible with I-pipe

Xenomai 2 and Xenomai 3 over the Cobalt core (i.e. dual kernel mode) both need special kernel support to deliver fast and deterministic response time to external interrupts, and services seamlessly integrated with the standard Linux kernel.

This support is provided by the interrupt pipeline (aka I-pipe) in the form of a kernel patch you have to apply against a vanilla kernel tree, before the Xenomai co-kernel can be built into Linux.
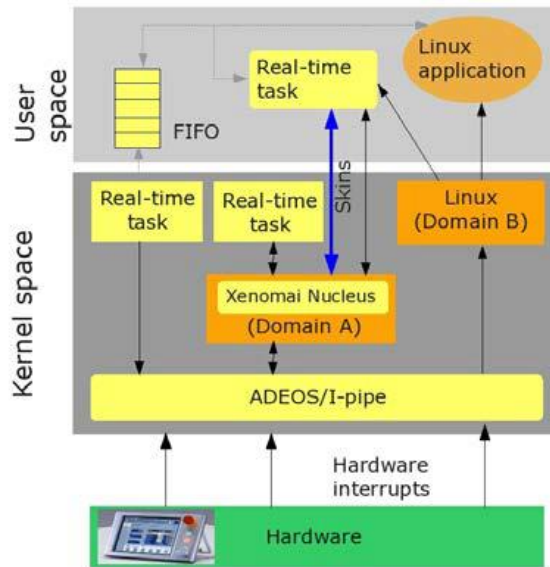
I chose Linux-4.9.90 because it is the latest kernel compatible with ADEOS/I-pipe. You can find the ADEOS/I-pipe patches and their compatibility at https://xenomai.org/downloads/ipipe/

- The architecture of my CPU is x86_64, the 64-bit edition of x86 architecture. so my patch are find in https://xenomai.org/downloads/ipipe/v4.x/x86/

Index of /downloads/ipipe/v4.x/x86

Name	Last modified	Size
Parent Directory		-
ipipe-core-4.1.18-x86-9.patch	2017-05-25 11:47	451K
ipipe-core-4.4.43-x86-8.patch	2017-06-14 11:47	486K
ipipe-core-4.4.71-x86-10.patch	2017-10-03 12:35	446K
ipipe-core-4.9.24-x86-2.patch	2017-06-12 11:06	491K
ipipe-core-4.9.38-x86-4.patch	2017-10-03 12:41	452K
ipipe-core-4.9.51-x86-5.patch	2018-03-26 09:17	456K
ipipe-core-4.9.90-x86-6.patch	2018-03-26 17:36	453K
older/	2018-03-26 09:17	-

As you can see here, the **ipipe-core-4.9.90-x86-6.patch** is the latest. So I chose to install **Linux-4.9.90** for my Xenomai.

Go to the website: https://mirrors.edge.kernel.org/pub/linux/kernel/v4.x/ , and search for the Linux kernel you are looking for. (tips: If you don't know how to search on website, Ctrl + F normally should work.)

```
linux-4.9.88.tar.xz      18-Mar-2018 19:44    89M
linux-4.9.89.tar.gz      22-Mar-2018 08:21   135M
linux-4.9.89.tar.sign    22-Mar-2018 08:21    833
linux-4.9.89.tar.xz      22-Mar-2018 08:21    89M
linux-4.9.9.tar.gz       09-Feb-2017 07:19   135M
linux-4.9.9.tar.sign     09-Feb-2017 07:19    833
linux-4.9.9.tar.xz       09-Feb-2017 07:19    89M
linux-4.9.90.tar.gz      25-Mar-2018 07:53   135M
linux-4.9.90.tar.sign    25-Mar-2018 07:53    833
linux-4.9.90.tar.xz      25-Mar-2018 07:53    89M
linux-4.9.91.tar.gz      28-Mar-2018 16:46   135M
linux-4.9.91.tar.sign    28-Mar-2018 16:46    833
linux-4.9.91.tar.xz      28-Mar-2018 16:46    89M
linux-4.9.92.tar.gz      31-Mar-2018 16:15   135M
linux-4.9.92.tar.sign    31-Mar-2018 16:15    833
linux-4.9.92.tar.xz      31-Mar-2018 16:15    89M
```

Now, download linux-4.9.90.tar.gz to your home directory. **Remember put it in the home directory and extract here!**
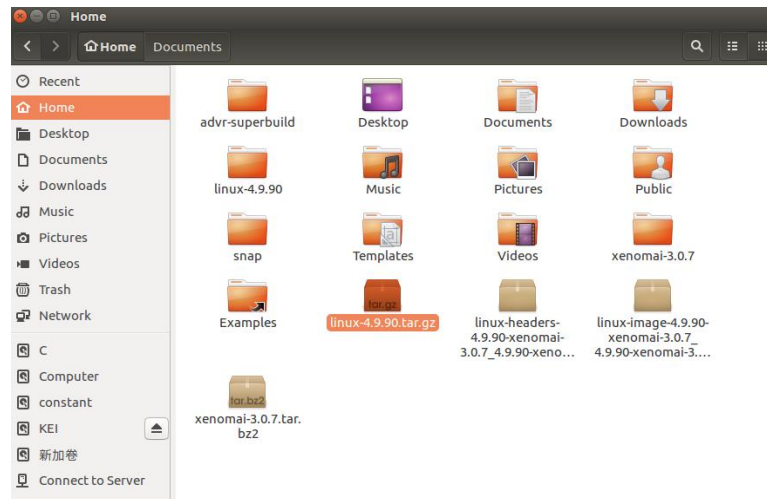

5.3 Get Xenomai kernel

Go to the website: https://xenomai.org/downloads/xenomai/stable/

```
xenomai-3.0.5.tar.bz2       2017-05-21 18:13   11M
xenomai-3.0.5.tar.bz2.sig   2017-05-21 18:13   72
xenomai-3.0.6.tar.bz2       2017-11-19 19:24   2.2M
xenomai-3.0.6.tar.bz2.sig   2017-11-19 19:24   72
xenomai-3.0.7.tar.bz2       2018-06-25 10:02   2.2M
xenomai-3.0.7.tar.bz2.sig   2018-06-25 10:41   72
xenomai-3.0.tar.bz2         2015-10-08 09:00   12M
xenomai-3.0.tar.bz2.sig     2015-10-08 09:15   72
```

I would suggest you download the newest version. For instance, I downloaded the xenomai-3.0.7.tar.bz2. **Remember to put it in the home directory and extract here!**
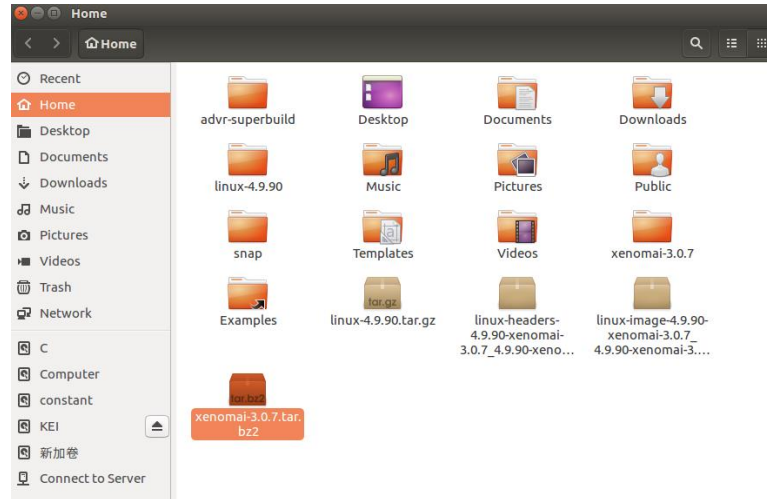
5.4 Get I-pipe

Now we have Linux and Xenomai, we still need I-pipe. To do this, let's go to the terminal. From now on, we will just use terminal to install Xenomai.
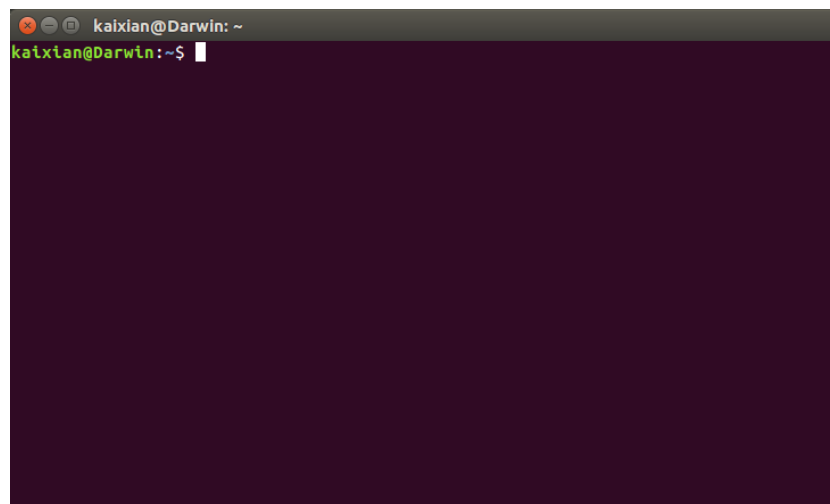
When you launch your terminal, you will be in your home directory automatically, just as the "~" indicates.

Now get I-pipe in our Linux kernel directory.

```
cd ~/linux-4.9.90

wget https://xenomai.org/downloads/ipipe/v4.x/x86/ipipe-core-4.9.90-x86-6.patch

../xenomai-3.0.7/scripts/prepare-kernel.sh --arch=x86_64 -ipipe=ipipe-core-4.9.90-x86-6.patch
```

5.5 Config the kernel

Take the actual working config:

```
yes "" | make oldconfig
```

And, then choose one from the following and change your configuration. For some reason, some may not work. In my opinion, anyone working should suffice to configure it successfully)

1) GUI version :

```
make xconfig

# normally, this won't work because you don't have qt platform. See the error message and use the following command to install it. Take qt5 as an example

apt-get install qt5-default

make xconfig
```

2) GTK+ version :

```
sudo apt install gtk+-2.0 glib-2.0 libglade2-dev

make gconfig
```

3) Without GUI :

```
sudo apt install libncurses5-dev

make menuconfig
```

Recommended options for configuration

Actually, these options are correlated, which is to say, some may not be activated unless you change another option. I believe I cannot disable CPU idle PM support until I disabled Processor in the directory of ACPI (Advanced Configuration and Power Interface) Support. So Good Luck on your configuration.

Remember not move on to next step before everything below has been done. CHECK TWICE!

* General setup
  --> Local version - append to kernel release: -xenomai-3.0.5
  --> Timers subsystem
      --> High Resolution Timer Support (Enable)
* Xenomai/cobalt
  --> Sizes and static limits
    --> Number of registry slots (512 --> 4096)
    --> Size of system heap (Kb) (512 --> 4096)
    --> Size of private heap (Kb) (64 --> 256)
    --> Size of shared heap (Kb) (64 --> 256)
    --> Maximum number of POSIX timers per process (128 --> 512)
  --> Drivers
    --> RTnet
        --> RTnet, TCP/IP socket interface (Enable)
            --> Drivers
                --> New intel(R) PRO/1000 PCIe (Enable)
                --> Realtek 8169 (Enable)
                --> Loopback (Enable)
        --> Add-Ons
            --> Real-Time Capturing Support (Enable)
* Power management and ACPI options
  --> CPU Frequency scaling
      --> CPU Frequency scaling (Disable)
  --> ACPI (Advanced Configuration and Power Interface) Support
      --> Processor (Disable)
  --> CPU Idle
      --> CPU idle PM support (Disable)
* Pocessor type and features
  --> Enable maximum number of SMP processors and NUMA nodes (Disable)
  // Ref : http://xenomai.org/pipermail/xenomai/2017-September/037718.html
  --> Processor family
      --> Core 2/newer Xeon (if "cat /proc/cpuinfo | grep family" returns 6, set as Generic otherwise)
  // Xenomai will issue a warning about CONFIG_MIGRATION, disable those in this order
  --> Transparent Hugepage Support (Disable)
  --> Allow for memory compaction (Disable)
  --> Contiguous Memory Allocation (Disable)
  --> Allow for memory compaction
    --> Page Migration (Disable)
* Device Drivers
  --> Staging drivers
      --> Unisys SPAR driver support
        --> Unisys visorbus driver (Disable)

5.6 Let's build the real-time kernel!

Now we have everything we need, let's build it

```
sudo apt install kernel-package

CONCURRENCY_LEVEL=$(nproc) make-kpkg --rootcmd fakeroot --initrd kernel_image
kernel_headers

cd ..

sudo dpkg -i linux-headers-4.9.90-xenomai-3.0.7_4.9.90-xenomai-3.0.7-
10.00.Custom_amd64.deb linux-image-4.9.90-xenomai-3.0.7_4.9.90-xenomai-3.0.7-
10.00.Custom_amd64.deb
```
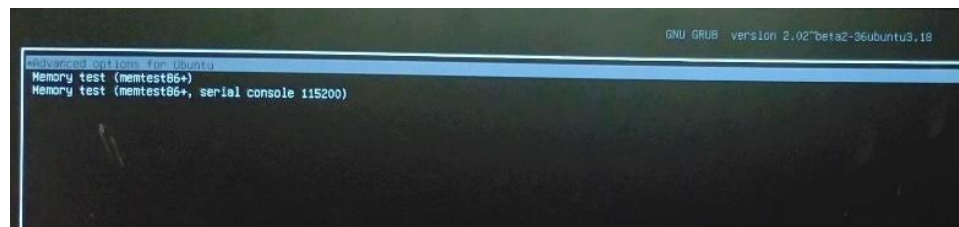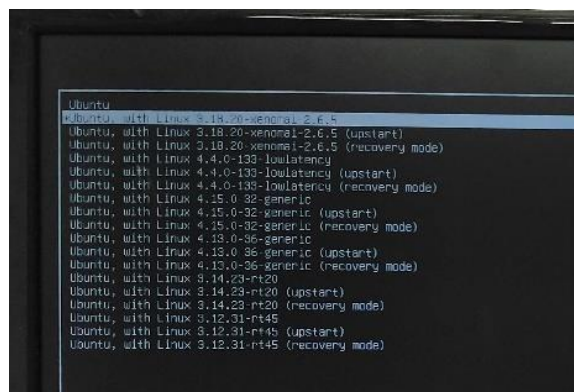
5.7 Configure GRUB and reboot

```
sudo update-grub

sudo reboot
```

After reboot, select "Advanced options for Ubuntu". See the picture below.

Under this directory, you should be able to see one option end with "–xenomai-x.x.x". Use ↑ ↓ to select it and press "ENTER" key. Now, wait for it to finish the rebooting.

After the reboot, let's see your current kernel with the following command.

```
uname -a
```


If it matches the one you chose, then your installation succeeds! Congratulations!

5.8 Installing User space libraries

```
cd xenomai-3.0.7

./configure --with-pic --with-core=cobalt --enable-smp --disable-tls --enable-
dlopen-libs --disable-clock-monotonic-raw

make -j`nproc`

sudo make install
```

There are also some potential errors in the future, let's avoid that!

```
git clone https://git.xenomai.org/xenomai-3.git

cd xenomai-3

./scripts/bootstrap

./configure --with-pic --with-core=cobalt --enable-smp --disable-tls --enable-
dlopen-libs

make -j`nproc`

sudo make install
```

5.9 Update your bashrc

```
echo '
### Xenomai

export XENOMAI_ROOT_DIR=/usr/xenomai

export XENOMAI_PATH=/usr/xenomai

export PATH=$PATH:$XENOMAI_PATH/bin:$XENOMAI_PATH/sbin

export PKG_CONFIG_PATH=$PKG_CONFIG_PATH:$XENOMAI_PATH/lib/pkgconfig

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$XENOMAI_PATH/lib

export OROCOS_TARGET=xenomai
' >> ~/.xenomai_rc


echo 'source ~/.xenomai_rc' >> ~/.bashrc

source ~/.bashrc
```

5.10 Test your latency

To test your latency, first make sure that you have cyclictest_run.sh and cyclictest_plot.sh and make them executable using chmod +x command.

```
./cyclictest_run.sh 100000 > result
```

Wait for a few minutes. This command creates a file named "result" containing the testing result. Then,

```
./cyclictest_plot.sh result
```

This command visualizes your data and put it in result.png in your current directory.
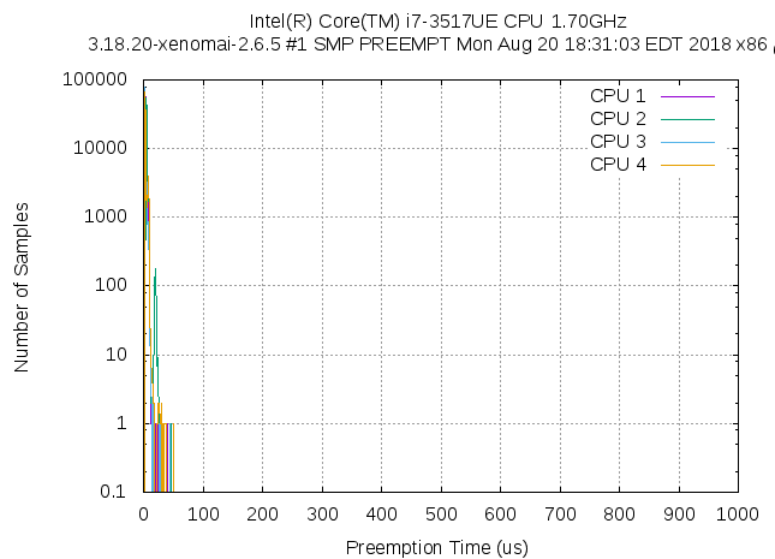
```
root@heater:~# ./cyclictest_run.sh 100000 > result
1.41user 0.84system 0:25.08elapsed 8%CPU (0avgtext+0avgdata 35332maxresident)k
104inputs+72outputs (1major+8359minor)pagefaults 0swaps
root@heater:~#  ./cyclictest_plot.sh result
./cyclictest_plot.sh: line 59: [: -lt: unary operator expected
./cyclictest_plot.sh: line 64: [: -l: integer expression expected
CPUS: 4  Title:  Intel(R) Core(TM) i7-3517UE CPU @ 1.70GHz Kernel:  3.18.20-xeno
mai-2.6.5 #1 SMP PREEMPT Mon Aug 20 18:31:03 EDT 2018 x86_64 L-Max:  X-Max Y-Max
Drawing ...

        Rectangular grid drawn at x y tics
        Major grid drawn with lt 0 linewidth 0.500
        Minor grid drawn with lt 0 linewidth 0.500
        Grid drawn at default layer

Histogram created: result.png
0
```

The following picture shows the latency test result for –xenomai kernel.

And you can compare it to latency result for –generic kernel