ChronOS Installation Guide

ECE 4550/5550G Real-Time Systems

This guide explains how to install ChronOS on your machine. ChronOS is a real-time operating system, and it consists of Linux and additional real-time patches. Ubuntu is used as the base operating system. If your machine has a different OS, you are encouraged to install Ubuntu OS as dual boot system. All programming exercises will require you to benchmark your code on your Ubuntu installation with the ChronOS patch.

1 Installing Linux

ChronOS requires an existing Linux distribution to be installed. We recommend Ubuntu 18.04.1 LTS server, although some earlier distribution should also work in theory. To install Ubuntu 18.04.1 LTS server version on your system, follow the instructions at:

http://old-releases.ubuntu.com/releases/bionic/ubuntu-18.04-live-server-amd64.iso

ChronOS supports both 32-bit and 64-bit distribution, but here the instructions are provided for 64-bit version and you are encouraged to use the 64-bit version.
Here we provide the instructions for installing Ubuntu 18.04.1 LTS server version. You will have to download Ubuntu 18.04.1 LTS ISO from Ubuntu website. Most of the installation steps are easy to follow, but you should ensure the following points.

1. You must not encrypt the home partition.
2. For partitioning, you should choose "Guided – use the largest continuous free space"
   Note: Selecting “Guided – use entire disk” will REMOVE your all hard disk data.
3. You can leave HTTP proxy information blank.

Note: check whether it is the correct kernel version /boot/config-4.15.0-xx-generic. It will be used in further procedures.

2 Preparing for Kernel Setup

In order to compile the kernel, the build tools need to be installed. On Ubuntu, these can be installed from the package manager with the following command:

```
$ sudo apt-get update -y && sudo apt-get upgrade -y # updates
$ sudo apt-get install build-essential libncurses5-dev bison flex \
    libssl-dev -y # install required software
```

Now you will need to first download the 4.16.8 kernel from the Linux kernel archives, found here:

https://mirrors.edge.kernel.org/pub/linux/kernel/v4.x/linux-4.16.8.tar.gz

Create a directory for your kernels somewhere in your home folder, and extract the archive there.

```
$ mkdir chronos && cd chronos
$ wget https://mirrors.edge.kernel.org/pub/linux/kernel/v4.x/linux-4.16.8.tar.gz
$ tar xjvf linux-4.16.8.tar.gz
$ mv linux-4.16.8 linux-4.16.8-chronos
```

Download the latest version of the ChronOS archive. It contains the latest `PREEMPT_RT` Linux patch and ChronOS kernel patches, associated libraries, and the test application called `sched_test_app`. Extract the Chronos and keep in the above created folder.BL: TODO: add tarball

```
$ wget tarball
$ tar xjvf tarball
```

Navigate into the renamed kernel folder, switch branches then apply the chronos patch.

```
$ cd linux-4.16.8-chronos
$ cp ../ChronOS-4.16.8/*.patch .
$ xzcat patch-4.16.8-rt3.patch.xz | patch -p1
$ git apply patch-4.16.8-chronos.patch
```

3 Configuring the Kernel

First, navigate into the kernel directory (if you're not already there) You may wish to copy over your previous configuration, since the default kernel config may not have the correct options selected for your distribution or hardware. You should copy the most recent config for your distribution to a *.config* file in this *linux-4.16.8-chronos* directory. For Ubuntu 18.04.1, this configuration is located at */boot/config-X.X.XX-XX-generic*, and you may copy it to *.config* in the kernel directory, like so:

```
$ cd linux-4.16.8-chronos
$ cp /boot/config-4.15.-generic .config
```

You should now update this `.config` file to work with the ChronOS kernel version. To update it with the defaults for every option which has been added between your distribution's kernel's release and the ChronOS kernel's release, and configure the kernel, run the following commands:

```
$ yes "" | make oldconfig
$ make menuconfig
```

This should bring up a set of menus. You need to mark/unmark the following options (If you do not see first option in your config then ignore the corresponding setting):

```
# here, we provide an easy way to do so.
$ cp ChronOS-4.16/config_chronos linux-4.16.8/.config
```

4 Compiling and Installing the Kernel

Compile the kernel and the modules and then install them.

```
$ cd linux-4.16.8-chronos
$ make bzImage -j‘nproc‘ # -j flag can use multiple core to compile the kernel
$ make modules -j‘nproc‘
$ sudo make modules_install
```

Install the kernel image to the `/boot`, create an **initramfs** image for the kernel and update grub

```
$ sudo cp arch/x86/boot/bzImage /boot/vmlinuz-4.16.8-chronos
$ sudo cp .config /boot/config-4.16.8-chronos
$ sudo cp System.map /boot/System.map-4.16.8-chronos
$ sudo update-initramfs -c -k 4.16.8-chronos
$ sudo update-grub
```

In order to ease the compilation and installation of ChronOS, we provide a bash script called *kinst* in the kernel folder. You will have to **chmod** this script before running it. This script will automate all above commands to compile and install the kernel.

```
$ cd linux-4.16.8-chronos
$ chmod +x kinst.sh
$ sudo ./kinst.sh
```

In recent versions of Ubuntu, boot options are disabled, as a result the system will always start with vanilla linux kernel. Since we need to boot into ChronOS, we can enable boot options by editing */etc/default/grub*(sudoer permission needed) file. You should change the existing settings as shown below:

```
#GRUB_HIDDEN_TIMEOUT=0
GRUB_TIMEOUT=10
```

Save changes and run `sudo update-grub` to apply changes.

5 Installing Libraries and Utilities

To define a set of tasks and convey their deadlines, we need **libchronos**, a userspace shared library that provides the hooks and headers files to interface with the ChronOS kernel. You need to install **libchronos** before installing any other userspace components. To install, simply run following commands inside the ChronOS directory (ChronOS_4.16).

```
$ cd libchronos
$ make all
$ sudo make install
```

To test the schedulability of the scheduling algorithms, you will need a userspace application `sched_test_app` written in C and designed to provide an interface for testing real-time schedulers. Follow the commands given below to compile it. More details can be found in the *README* file.

```
$ cd sched_test_app
$ make
$ sudo make install
```

Now, you can restart the machine, for example with:

```
$ sudo reboot
```

On restarting the machine, when you see boot options, go into Advanced Options and select ChronOS. If you have the following error while running `sched_test_app` or if this is your first time install and run chronOS:

“`Error: Could not open slope file to read average slope.`”

Please run the following commands from the *chronos/ChronOS_4.16/sched_test_app/* directory:

```
$ sudo mkdir /usr/local/chronos/slope/
$ sudo ./find_slopes -f
```

The second command can take more than ten minutes to complete.
Now you can test existing scheduling algorithms (RMA and FIFO), by the following commands (task sets are in the Chronos_4.16 directory):

```
$ sudo sched_test_app -s RMA -f chronos/ChronOS_4.16/Task\ Sets/5t_nl \
  -c 100 -r 20
```

If you find an error message indicating that the scheduler is not loaded, you can load it manually by following command (example for RMA):

```
$ sudo modprobe rma
```

You should see the RMA module in the list of active module returned by the `lsmod` command. Now you should be able to test `sched\_test\_app` with the RMA scheduling algorithm. To switch between algorithms you have to first remove the current one with the following command:

```
$ sudo rmmod rma
```

`sched_test_app` should now run fine.

Appendix A Use QEMU for debugging

In this section, we will go through the steps of installing ChronOS inside a virtual machine (VM). QEMU will be used as the hypervisor to launch the VM. The VM will be used for the development but cannot be used for the evaluations, you are expected to install ChronOS on your host Ubuntu installation for the evaluations.
Instructions to install QEMU and installing an OS in a QEMU virtual machine using an iso file are as follows:

1. On an Ubuntu system you can install QEMU by running the command:
   `$ sudo apt-get install qemu-system`

2. Download an iso image of ubuntu 18.04.1 LTS (Ubuntu server advised, Ubuntu desktop is also possible). Following instructions are provided for server image. Instructions to install Ubuntu 18.04.1 server are provided in section 1.

3. Create a folder for keeping the image of guest OS and change your current directory into the new folder.
   ```
   $ mkdir qemu_image
   $ cd qemu_image
   ```

4. You should now create a virtual disk which will be needed for the virtual machine. This will create a file of size big enough to contain the guest OS. Create an image at least 30GB large.
   `$ qemu-img create -f qcow2 rtos_image.img 30G`

5. Now use the Ubuntu OS image to install an Ubuntu instance in the virtual image created in earlier step. You can start installing the Ubuntu instance by executing `qemu-system-x86_64` command.

```
$ qemu-system-x86_64 -enable-kvm \
                     -boot d \
                     -cdrom ./ubuntu-18.04.1-live-server-amd64.iso \
                     -m 2048 \
                     -hda rtos_image.img \
                     -smp 8 \
```

6. Steps of installation of Ubuntu are the same as section 1. After Ubuntu is installed and it reboots, kill the QEMU session by either typing **ctrl-c** in terminal window (in which QEMU was launched) or by closing the QEMU window.

7. Start the installed ubuntu instance with the following command:
   `$ qemu-system-x86_64 -enable-kvm -m 2048 -hda rtos_image.img -smp 8`

   - **-enable-kvm** is an optional flag which speeds up the VM by using CPU virtualization.
   - **-m** flag denotes the amount of RAM allocated to QEMU VM. Depending on your machine's resources more RAM could be allocated to speed up the VM.
   - **-hda** flag describes the disk image used to boot the system.
   - **-smp** is to use multiple CPUs to run. The usage is `-smp <the number of CPUs on your host machine>`.

You can debug the QEMU VM with the gdb debugger. For more information consult the following link:

https://wiki.osdev.org/Kernel_Debugging#Use_gdb_with_Qemu