Technical Report

Building a real-time Debian distribution for embedded systems

Pedro Henrique Santana, Bruno Guilherme Amui, Felipe Brandão Cavalcanti,
Glauco Garcia Scandaroli, Geovany Araújo Borges.

Robotics and Automation Laboratory (LARA)
Department of Electrical Engineering, University of Brasília, Brazil

April 5, 2011

1 Introduction

Complex robotics projects usually demand real-time operational systems (RTOS) for control means. These OSs are normally installed in solid-state disks (e.g., CompactFlash), which do not have as much storage space as common hard disks neither can be written as many times. Also in the robotics universe, a real-time extension for the OS can be a really useful tool, since it is sometimes impossible to handle the non-determinism introduced by the OS during hard real-time control tasks. A real-time extension is a software layer that is capable of interrupting the operational systems's tasks for the sake of timing determinism.

This tutorial tries to describe the steps to build a Debian distribution designed to control a helicopter robot. The whole system is installed in a 1GB CompactFlash (CF) and received a Xenomai real-time extension for control purposes. The instructions found here are based in [1–8].

2 Installing a Debian base system

- First, create a directory or partition where you will install a new Debian base system

```
$ su
# mkdir /embedded
```

- Download and install the **debootstrap** package and use it to create your base system. Two options are presented.

```
# apt-get install debootstrap
```

1. Debian 4.0 Etch[1]

   ```
   # debootstrap --arch i386 etch /embedded http://ftp.debian.org
   ```

2. Debian 5.0 Lenny

   ```
   # debootstrap --arch i386 lenny /embedded http://ftp.debian.org
   ```

- After the download and install processes, change your root partition to your new Debian system's directory

```
# mount -t proc proc /embedded/proc/
# mount -o bind /dev/ /embedded/dev/
# LC_ALL=C chroot /embedded /bin/bash
```

- Install the necessary packages for development. The ones described below were useful in some of the lab's robotics projects [2].

```
# apt-get -f install
# apt-get install ssh gcc make libncurses-dev grub udev rsync
# apt-get install pciutils usbutils less build-essential kernel-package ncurses-dev
# apt-get install ifupdown ifrename initramfs-tools bzip2
```

  - If you are using Debian 4.0 Etch

    ```
    # apt-get install linux-source-2.6.24
    ```

  - If you are using Debian 5.0 Lenny

    ```
    # apt-get install linux-source-2.6.26
    ```

- Verify your *gcc* version.

```
# gcc -v
```

- IF, AND ONLY IF, you get a *gcc* version newer than *gcc-4.1*, it is recommended that you download *gcc-4.1* and change the symbolic links **gcc** (compiler) and **cpp** (preprocessor) at /**usr**/**bin**.

```
# apt-get install gcc-4.1
# cd /usr/bin
# rm gcc
# ln -s gcc-4.1 gcc
# rm cpp
# ln -s cpp-4.1 cpp
```

[1] There were some hardware compatibility problems with this distribution when trying to run it on a PCM-3350 board. If you are experiencing similar problems, try using the Debian 5.0 Lenny distribution.

[2] The versions for the Linux Kernel and patches shown in this section and in the next were already tested and, because of that, are presented as suggestions. Nevertheless, different ones may work as well.

3 Patching the Kernel with a Xenomai real-time extension

- Download the most recent Xenomai source at `http://www.xenomai.org/`. The commands below, which download Xenomai 2.5.3 to the folder /**usr**/**src**, are shown here as an example.

```
# cd /usr/src
# wget http://download.gna.org/xenomai/stable/xenomai-2.5.3.tar.bz2
```

- Uncompress your source files and, if desired, create symbolic links to them

```
# tar xvfj xenomai-2.5.3.tar.bz2
# ln -s xenomai-2.5.3 xenomai
```

  - If you are using Kernel 2.6.24

```
# tar xvfj linux-source-2.6.24.tar.bz2
# ln -s linux-source-2.6.24 linux
```

  - If you are using Kernel 2.6.26

```
# tar xvfj linux-source-2.6.26.tar.bz2
# ln -s linux-source-2.6.26 linux
```

- If there is no suitable patch for your Kernel in Xenomai's patch folder, download the most suitable one from `http://download.gna.org/adeos/patches/`. The patches shown here were previously tested and are presented as suggestions.

  - If you are using Kernel 2.6.24

```
# cd /usr/src/xenomai/ksrc/arch/x86/patches/
# wget http://download.gna.org/adeos/patches/v2.6/x86/older/adeos-ipipe-2.6.24-x86-2.0-07.patch
```

  - If you are using Kernel 2.6.26

```
# cd /usr/src/xenomai/ksrc/arch/x86/patches/
# wget http://download.gna.org/adeos/patches/v2.6/x86/older/adeos-ipipe-2.6.26.7-x86-2.0-17.patch
```

- Copy a suitable **.config** file to your Linux's source tree (/**usr**/**src**/**linux**). If there is none, use your Kernel's current **config** file. In order to do that, open another terminal window and type the commands below. After that, return to the previous terminal

```
$ su
# cp /boot/config-‘uname -r‘ /embedded/usr/src/linux/.config
```

- Patch your Kernel with Xenomai's **prepare-kernel** script

  - If you are using Kernel 2.6.24

```
# /usr/src/xenomai/scripts/prepare-kernel.sh --linux=/usr/src/linux --arch=x86
--adeos=/usr/src/xenomai/ksrc/arch/x86/patches/adeos-ipipe-2.6.24-x86-2.0-07.patch
```

  - If you are using Kernel 2.6.26

```
# /usr/src/xenomai/scripts/prepare-kernel.sh --linux=/usr/src/linux --arch=x86
--adeos=/usr/src/xenomai/ksrc/arch/x86/patches/adeos-ipipe-2.6.26.7-x86-2.0-17.patch
```

4 Configuring the Kernel

- Configure your new Kernel

```
# cd /usr/src/linux
# make menuconfig
```

The following fields should be changed. Depending on the Kernel version, some of them may not be avaiable.

```
- General setup
     |--> Local version ---> -xenomai
- Loadable module support
     |--> Enable loadable module support ---> enabled
     |--> Module versioning support ---> disabled
- Processor type and features
     |--> Preemption Model ---> Preemptible Kernel (Low-Latency Desktop)
     |--> Interrupt pipeline ---> enabled
     |--> High Resolution Timer Support ---> enabled
     |--> HPET Timer Support ---> disabled
     |--> Toshiba Laptop support ---> disabled
     |--> Dell Laptop support ---> disabled
     |--> Enable -fstack-protector buffer overflow detection (EXPERIMENTAL) ---> disabled
     |--> Support sparse irq numbering ---> disabled
     |--> Check for P4 thermal throttling interrupt ---> disabled
- Power management options
     |--> Power Management support ---> disabled
     |--> CPU Frequency scaling ---> CPU Frequency scaling ---> disabled
     |--> CPU idle PM support ---> disabled
- Kernel hacking
     |--> KGDB: kernel debugging with remote gdb ---> disabled
- Device Drivers
     |--> Input device support ---> Miscellaneous devices ---> PC Speaker support ---> disabled
```

- For non-SMP systems, disable symmetric multi-processing support

```
- Processor type and features
     |--> Symmetric multi-processing support ---> disabled
```

- Choose the most suitable processor family for your machine. For example, for a Pentium-III processor

```
- Processor type and features
     |--> Processor family ---> Pentium-III / Celeron(Coppermine) / Pentium-III Xeon
```

- If you have a dual core CPU or SMP system, don't choose a processor family which has no TSC (time stamp counter). This means that, for example, you can't choose 586/K5/5x86/6x86/6x86MX as Processor family if you have a dual core CPU. In conclusion, choose the most suitable processor family for your machine.
- Check the **Real-time sub-system** Kernel option. If there are any conflicts between your Kernel's configuration and Xenomai, follow the instructions given in order to solve the problem. When the options menu appears, change the following

```
- Real-time sub-system
     |--> Machine ---> SMI workaround ---> Disable SMI detection ---> disabled
     |--> Machine ---> SMI workaround ---> Enable SMI workaround ---> enabled
     |--> Machine ---> SMI workaround ---> Globally disable SMI ---> enabled
```

5 Compiling the Kernel

- If there were no errors, you can start compiling your Kernel. Since we are dealing with a reduced distribution, this process probably will not take more then a few minutes

  ```
  # make && make modules
  ```

- Some warning messages may appear, but normally you should not worry about them. If no errors occur, install the Kernel's modules

  ```
  # make modules_install
  ```

- Install the Kernel's image

  ```
  # make install
  ```

- Create an initrd image and symbolic links to your Kernel's image and initrd image

  – If you are using Kernel 2.6.24

  ```
  # cd /boot
  # update-initramfs -c -k 2.6.24-xenomai
  # ln -s vmlinuz-2.6.24-xenomai vmlinuz
  # ln -s initrd.img-2.6.24-xenomai initrd
  ```

  – If you are using Kernel 2.6.26

  ```
  # cd /boot
  # update-initramfs -c -k 2.6.26-xenomai
  # ln -s vmlinuz-2.6.26-xenomai vmlinuz
  # ln -s initrd.img-2.6.26-xenomai initrd
  ```

6 Installing Xenomai

- Configure, compile and install your Xenomai real-time extension. By default, it will be placed in /**usr**/**xenomai**. Below, we present two possible ways to do that

  1. Using **configure**'s default options

     ```
     # cd /usr/src/xenomai
     # ./configure
     # make
     # make install
     ```

  2. As described in [9], you may get the error

     ```
     Incompatible feature set userland requires "tsc", kernel provides "set fastsynch smp" missing "tsc"
     ```

     when trying to execute Xenomai's test scripts under some OSs, like newer Ubuntu (9.04). To avoid that, change the installation procedure to

```
# cd /usr/src/xenomai
# ./configure --disable-x86-tsc
# make
# make install
```

- Create **xenomai.conf** file

  ```
  # vim /etc/ld.so.conf.d/xenomai.conf
  ```

  and add

  ```
  # xenomai libs
  /usr/local/lib
  /usr/xenomai/lib
  ```

  to it

- Update your **ld path** in order to be able to use Xenomai's libraries

  ```
  # ldconfig
  ```

- If you find it useful, you can add the Xenomai's binaries to your system's path

  ```
  # PATH=$PATH:/usr/xenomai/bin
  # export PATH
  ```

  This will change the search path just for the current terminal session. If you want to make this change permanent, add the last two lines to /**root**/**.bashrc** (**root** user) and/or /**home**/<**user**>/**.bashrc** (<**user**> user)

- You may have some troubles running Xenomai's scripts under Ubuntu and other recent Debian derivatives. This is due to the fact that these distributions use **dash** instead of **bash** [10]. To overcome this issue, you may proceed as follows

  1. Change the first line of **xeno-test** from #!/**bin**/**sh** to #!/**bin**/**bash**;
  2. Change the symbolic link $sh \rightarrow dash$ at /**bin** to $sh \rightarrow bash$

     ```
     # cd /bin
     # rm sh
     # ln -s bash sh
     ```

7 Modifying the OS

Operational systems designed to run from a CF must meet some restrictions. Besides having reduced size, they must also write as few times as possible on the CF to increase its lifetime. Hence, the real-time OS built so far needs some modifications to avoid writing unecessarily to the disk. The procedure was based on [2].

- Remove /**etc**/**mtab** and symlink it to /**proc**/**mounts**. This special file also describes the mounted filesystems, and can replace **mtab** without needing a write access on it

```
# rm /etc/mtab
# ln -s /proc/mounts /etc/mtab
```

- Another file in /**etc** that needs to be writeable is **resolv.conf**. We make it writeable by removing it, and creating a symlink to /**var**/**log**, which is writable

```
# mv /etc/resolv.conf /var/log
# ln -s /var/log/resolv.conf /etc/resolv.conf
```

- Create or modify some config files
  - /**etc**/**fstab**

```
/dev/hda1 / ext2 defaults,noatime 0 0
proc /proc proc defaults 0 0
tmpfs /var/run tmpfs defaults 0 0
tmpfs /var/lock tmpfs defaults 0 0
tmpfs /var/log tmpfs defaults 0 0
tmpfs /tmp tmpfs defaults 0 0
tmpfs /work tmpfs defaults 0 0
tmpfs /var/lib/dhcp3/ tmpfs defaults 0 0
```

  - /**sbin**/**dhclient-script**
    - Set **new_resolv_conf** to **"/tmp/resolv.conf.dhclient-new"**
    - Change **"mv -f $new_resolv_conf /etc/resolv.conf"** to **"cat $new_resolv_conf > /etc/resolv.conf"**
  - /**etc**/**network**/**interfaces**

```
#Network configurations

#Loads the loopback (lo) and eth0 network interfaces automatically
auto lo eth0
#Enables hotplug for eth0
allow-hotplug eth0
#Configures the loopback interface
iface lo inet loopback

#Configures eth0 using DHCP (Automatic)
iface eth0 inet dhcp

#Uncomment the next lines if you want to configure
#eth0 manually (choose appropriate values)

#iface eth0 inet static
#address 192.168.0.95
#netmask 255.255.255.0
#gateway 192.168.0.253
```

```
#network  192.168.0.0
#broadcast 192.168.0.255
```

- **/etc/hosts**

  ```
  127.0.0.1 <YOUR_HOSTNAME>.<YOUR_LOCALDOMAIN> localhost <YOUR_HOSTNAME>
  ```

- If you are using Debian 4.0 Etch
  - **/etc/syslog.conf**
    - Comment the lines where **/dev/xconsole** is mentioned
  - **/etc/init.d/bootclean**
    - Add the following lines before the line stateing **[-f /tmp/.clean] && ...** (located at the end of the file)

      ```
      touch /var/log/resolv.conf
      touch /var/log/dmesg
      ```
  - **/etc/apt/sources.list**
    - Modify the file's content so it looks like this

      ```
      deb http://ftp.debian.org etch main contrib non-free
      deb http://ftp.us.debian.org/debian etch main contrib non-free
      ```
- If you are using Debian 5.0 Lenny
  - **/etc/rsyslog.conf**
    - Comment the lines where **/dev/xconsole** is mentioned
  - **/lib/init/bootclean.sh**
    - Add the following lines before the line stateing **[-f /tmp/.clean] && ...** (located at the end of the file)

      ```
      touch /var/log/resolv.conf
      touch /var/log/dmesg
      ```
  - **/etc/apt/sources.list**
    - Modify the file's content so it looks like this

      ```
      deb http://ftp.debian.org lenny main contrib non-free
      deb http://ftp.us.debian.org/debian lenny main contrib non-free
      ```

- Create users for your OS

  ```
  # adduser <YOUR_USERNAME>
  ```

- Set the root's password

  ```
  # passwd
  ```

- Create two special directories

  ```
  # mkdir /work
  # mkdir /home/load
  ```

The /**work** directory, which is mounted on RAM according to /**etc**/**fstab**, will be your workspace. Use it to store all your editable files, i.e., files that will be constantly changed, since writing on /**work** do not affect the CF. The /**home**/**load** will be used as a persistent storage space for the files in /**work** and should not be directly edited by the user

- Create a backup script that saves all the information in /**work** to /**home**/**load**

```
# nano /etc/init.d/write-files-cf.sh
```

and add

```
#!/bin/sh
#Transfers the files from the work directory mounted in RAM to the CF
echo "Writing files from /work to /home/load"
rsync -a --delete /work/ /home/load
echo "Writing complete"
exit 0
```

- Create a script to restore the information stored in /**home**/**load** to /**work**

```
# nano /etc/init.d/load-files-cf.sh
```

and add

```
#!/bin/sh
#Transfers the files from the CF to the work directory mounted in RAM
echo "Loading /home/load files to RAM"
rsync -a --delete /home/load/ /work
echo "Loading complete"
exit 0
```

- Render the scripts executable

```
# chmod ugo+x /etc/init.d/write-files-cf.sh
# chmod ugo+x /etc/init.d/load-files-cf.sh
```

- Create OS services that automatically write the files in /**work** to /**home**/**load** when the system shuts down or restarts and restore them at boot time

```
# ln -s /etc/init.d/write-files-cf.sh /etc/rc0.d/S01writefilestocf
# ln -s /etc/init.d/write-files-cf.sh /etc/rc6.d/S01writefilestocf
# ln -s /etc/init.d/load-files-cf.sh /etc/rc2.d/S99loadfilesfromcf
```

- Remove all unnecessary packages and files to reduce the OS's size

```
# apt-get remove <package name>
# rm -r <directory name>
# rm <file>
```

Some **tmpfs** entries we created in /**etc**/**fstab**. These point to directories where files should be writeable at all times (/**var**/**lock**, /**var**/**run**). These directories are mapped to memory and, when a program writes to a file in that particular directory, it actually writes to the memory. It prevents error messages to flood the terminal about files being written on a read only disk.

8 Preparing the CF

It is often desirable to format your CF with a non-journaling file system (e.g., **Ext2**), reducing disk writes. Lets assume your CF is recognized as /**dev**/**sda** and you want to install the OS at its first partition.

- Exit **chroot** environment

  ```
  # exit
  ```

- Guarantee your CF is unmounted and create as many partitions as you want on the disk. Do not forget to make the first partition bootable

  ```
  # umount /dev/sda1
  # cfdisk /dev/sda
  ```

- Format the first partition as **Ext2** and disable disk checking

  ```
  # mkfs.ext2 /dev/sda1
  # tune2fs -c 0 -i 0 /dev/sda1
  ```

- Mount your CF card at a directory of your choice

  ```
  # mkdir /mnt/cf
  # mount /dev/sda1 /mnt/cf
  ```

- Copy your OS's files to the CF, except for the /**proc** directory. You can copy the files by hand (**Midnight Commander** package is recommended) or you can move the /**proc** directory to a temporary location, copy everything and then move it back

  ```
  # mv /embedded/proc /tmp/tmpproc
  # cp -a /embedded/* /mnt/cf/
  # mv /tmp/tmpproc /embedded/proc
  ```

- Create a /**proc** directory on the CF

  ```
  # mkdir /mnt/cf/proc
  ```

9 Installing GRUB bootloader

- Set a basic GRUB installation on your CF

```
# grub-install --recheck --root-directory=/mnt/cf /dev/sda
```

- Enter GRUB's command-line and setup GRUB on the CF's first partition

```
# grub
grub> root (hd0,0)
grub> setup (hd0)
```

- Create a **menu.lst** file on the CF's GRUB directory

```
# vim /mnt/cf/boot/grub/menu.lst
```

containing

```
# Example config file for GRUB
# The GNU GRand Unified Bootloader
# /boot/grub/menu.lst

# general configuration:
timeout  10
default   0
fallback  1
fallback  9
#color     blue/cyan  yellow/magenta

# boot sections follow
# each is implicitly numbered from 0
# in the order of appearance below

# Embedded Debian
title   Debian 4.0 Etch with Xenomai Extension
root    (hd0,0)
kernel  /boot/vmlinuz root=/dev/hda1
initrd  /boot/initrd

# end file menu.lst
```

- You can add other OS entries in the previous **menu.lst** file

10 Testing Xenomai

- Reboot into your new real-time Debian distribution and run **xeno-test** script

  ```
  $ su
  # /usr/xenomai/bin/xeno-test
  ```

  This test will create a 100% load for your processor and measure the Kernel's latency.

- If you get the error

  ```
  Incompatible feature set userland requires "tsc", kernel provides "set fastsynch smp" missing "tsc",
  ```

  go back to Section 6 and reconfigure Xenomai

- If, during the script execution, the message "something died a..." keeps being displayed, go back to Section 6 and see the topic about changing **dash** to **bash**.

- If everything goes without errors, you are done!

References

[1] Amarpreet Singh Ugal. Hard Real time Linux* Using Xenomai* on Intel Multi-Core Processors, 2009. Available at `http://edc.intel.com/Link.aspx?id=2656`.

[2] Kristof Van Hertum. Debian on compact flash, 2008. Available at `http://kristof.vanhertum.be/?p=3`.

[3] Windel Bouwman. Xenomai quick build guide, 2008. Available at `http://www.xenomai.org/index.php/Xenomai_quick_build_quide`.

[4] nixCraft. How to: Compile Linux kernel 2.6, 2006. Available at `http://www.cyberciti.biz/tips/compiling-linux-kernel-26.html`.

[5] GNU GRUB Manual 0.97. Installing GRUB using grub-install. Available at `http://www.gnu.org/software/grub/manual/html_node/Installing-GRUB-using-grub_002dinstall.html`.

[6] GNU GRUB Manual 0.97. Installing GRUB natively. Available at `https://mail.gna.org/public/xenomai-help/2009-04/msg00047.html`.

[7] Cristóvão Souza. How-to Install RTAI in Ubuntu Hardy, 2008. Available at `https://woc.uc.pt/deec/getFile.do?tipo=2&id=5690`.

[8] EMC Documentation Wiki. Debian Etch Compile RTAI, 2008. Available at `http://wiki.linuxcnc.org/cgi-bin/emcinfo.pl?Debian_Etch_Compile_RTAI`.

[9] Martin Shepherd. [Xenomai-help] Xeno-test bug. Available at `http://www.gnu.org/software/grub/manual/html_node/Installing-GRUB-natively.html`.

[10] Wikipedia, the free encyclopedia. Debian Almquist shell. Available at `http://en.wikipedia.org/wiki/Debian_Almquist_shell`.