National Centre for Radio Astrophysics

Internal Technical Report
GMRT/SERVO/PC104/002-Aug 2013

PC104 Servo Computer Software Installation Manual

THIYAGARAJAN BEEMAN
thiyagu@gmrt.ncra.tifr.res.in

Giant Metrewave Radio Telescope
Tata Institute of Fundamental Research
Khodad - 410504

Document Status History

S.No	Document Status	Description
1	Document Title	PC104 Servo Computer Software Installation Manual
2	Document Reference No.	GMRT/SERVO/PC104/002-Aug 2013
3	Revision	0.0
4	Date	29-July-2013
5	Prepared By	Thiyagarajan Beeman
6	Reviewed By	Servo Group Members
7	Approved By	SMEC Committee

Document Change Record

DCR No	First Issue	
Date	1-Aug-2013	
Document Title	PC104 Servo Computer Software Installation Manual	
Document Reference No	GMRT/SERVO/PC104/001-Aug 2013	
Document Revision No	0.0	
Page No	**Paragraph**	**Reason For Change**

ABSTRACT

The *PC104 Servo Computer Software Installation Manual* provides comprehensive information for installing Puppy Linux, development files, compilers, application packages and Real Time Application Interface (RTAI) on ICOP VDX6354D embedded boards. Preliminarily it discusses the hardware and software requirements, and installation steps to run Puppy Linux from the RAM. Later this document also lays down the ways to partition and install Puppy Linux on internal IDE compact flash. The RTAI and application package installation are discussed in two different methods. The first method follows the generic way of patching, configuring, compiling and installation of Linux Kernel and RTAI. The second method eases the work of users by running few readymade script files. Test for successful installation of Linux kernel and RTAI are given in the document making this document a complete manual for the software installation.

CONTENTS

List of Figures

List of Tables

Acronyms

USB	Universal Serial Bus
HDD	Hard Disk Drive
SATA	Serial ATA
IDE HDD	Integrated Drive Electronics Hard Disk Drive
GRUB	Grand Unified Boot Loader
GCC	GNU Compiler Collection
RTAI	Real Time Application Interface
initrd	Initial Ram Disk Image
BIOS	Basic Input Output System
CD-ROM	Compact Disk Read Only Memory
RAM	Random Access Memory

1. Introduction

This document explains the procedure to install the Linux operating system and Real time kernel patch on ICOP VDX6354D PC104 embedded boards. Present version of development based on Linux kernel 2.6.23 and Real Time Application Interface (RTAI) 3.8.1. We have chosen Puppy Linux-4.3.1 to provide the Graphical User interface to vanilla kernel 2.6.23. Puppy Linux-4.3.1 is chosen because of its following unique features,

- Puppy Linux-4.3.1 (Linux Kernel-2.6.30) matches with the RTAI patch version.
- Smaller in Size (~100MB),
- Bundled with collection of application suites,
- Less boot time (30 to 40 sec based on hardware),
- Graphical User Interface makes ease of use,
- Live booting from CDs, DVDs, USB flash drives and other portable media.

2. Puppy Linux Installation

The installation of Puppy Linux on HDD becomes easier with the program called *Puppy Universal Installer* that install puppy into many different media including HDD, USB, SATA or IDE Hard Drives, even in Compact Flash (CF) plugged into IDE. There are two installation methods are available to install Puppy Linux on HDD

1. Frugal Installation
2. Full Installation

In Frugal Installation, the files *vmlinuz*, *initrd.gz* and *pup_431.sfs* are copied to a partition. This partition may already have something installed on it and it will not be disturbed. This can be any type of partition Windows (FAT or NTFS) or Linux (EXT2, EXT3 or EXT4).

A Full Installation is the normal traditional Linux HDD installation, and requires the partition to have a Linux file system (EXT2, EXT3, EXT4 or REISERFS).

2.1 Requirements

- Puppy Linux-4.3.1 boot media
- ICOP VDX 6354D Embedded Board
- Compact Flash or HDD

2.2 Running from RAM

- Power off your computer system,
- Insert the boot media into USB PORT or CD-ROM and power it on.

You might need to press a specific key or combination of keys to boot from the media. On most systems, a message appears briefly on the screen very soon after you turn on the system. Typically, it is worded something like ***Press F11 to select boot device or press DEL to enter BIOS,*** *Once you have entered your BIOS setup program, find the section where you can alter your boot sequence. Change this sequence so that the USB is first in your boot device. This instructs the*

computer to first look at the USB drive for bootable media; if it does not find bootable media on the USB drive, it then checks your hard drive or diskette drive. Save your changes before exiting the BIOS.

Necessary Boot setting for ICOP VDX6354D embedded boards to make first boot device as USB as follows,

Boot -> Boot Device Priority -> First Boot Device -> USB Device

Boot -> Boot Device Priority -> Second Boot Device -> HDD:PM_InnoDisk Crop. – EDC4000

The following BIOS settings are required to install the Puppy Linux-4.3.1 on VDX 6354D Boards. These settings will help us to detect the IDE devices after booting the Puppy Linux.

Advanced -> IDE Configuration -> IDE Operate Mode -> From "Legacy Mode" to "Native Mode".
Advanced -> IDE Configuration -> Standard IDE Compatiable -> From "Enabled to Disabled".

Once the PC has booted with boot media, the following screen will appear,

Figure 2- Puppy Linux Boot Screen

Choose mode of installation either graphical or command line as well as we can pass the boot parameters to Puppy Linux. On screen message will guide you to pick required boot parameters by pressing function key F2. The different boot options which are passed to Puppy Linux are listed below.

acpi=off	Default on for PCs > 2001, may give boot/shutdown probs.
loglevel=<n>	Bootup verbosity. 7 is high verbosity for debugging.
pfix=ram	Run Puppy totally in RAM ignore saved sessions
pfix=<n>	number of saved sessions to ignore (multisession CD),

pfix=nox	command line only, do not start X,
pfix=noram	do not copy pup_xxx.sfs to ram (useful 256MB PCs only),
pfix=clean	file cleanup (simulate version upgrade),
pfix=purge	more radical file cleanup (to fix broken system),
pfix=rdsh	fro developers only (intiramfs shell).

Example:

Puppy acpi=off pfix=2 Ignore ACPI, blacklist last 2 saved sessions.

We have chosen graphical installation with default boot options. Wait for 5seconds as puppy will take you into installation processes.

Basically it copies the *drivers*, *puppy sfs files*, *kernel* and *initrd images* into RAM. The following screen confirms that.

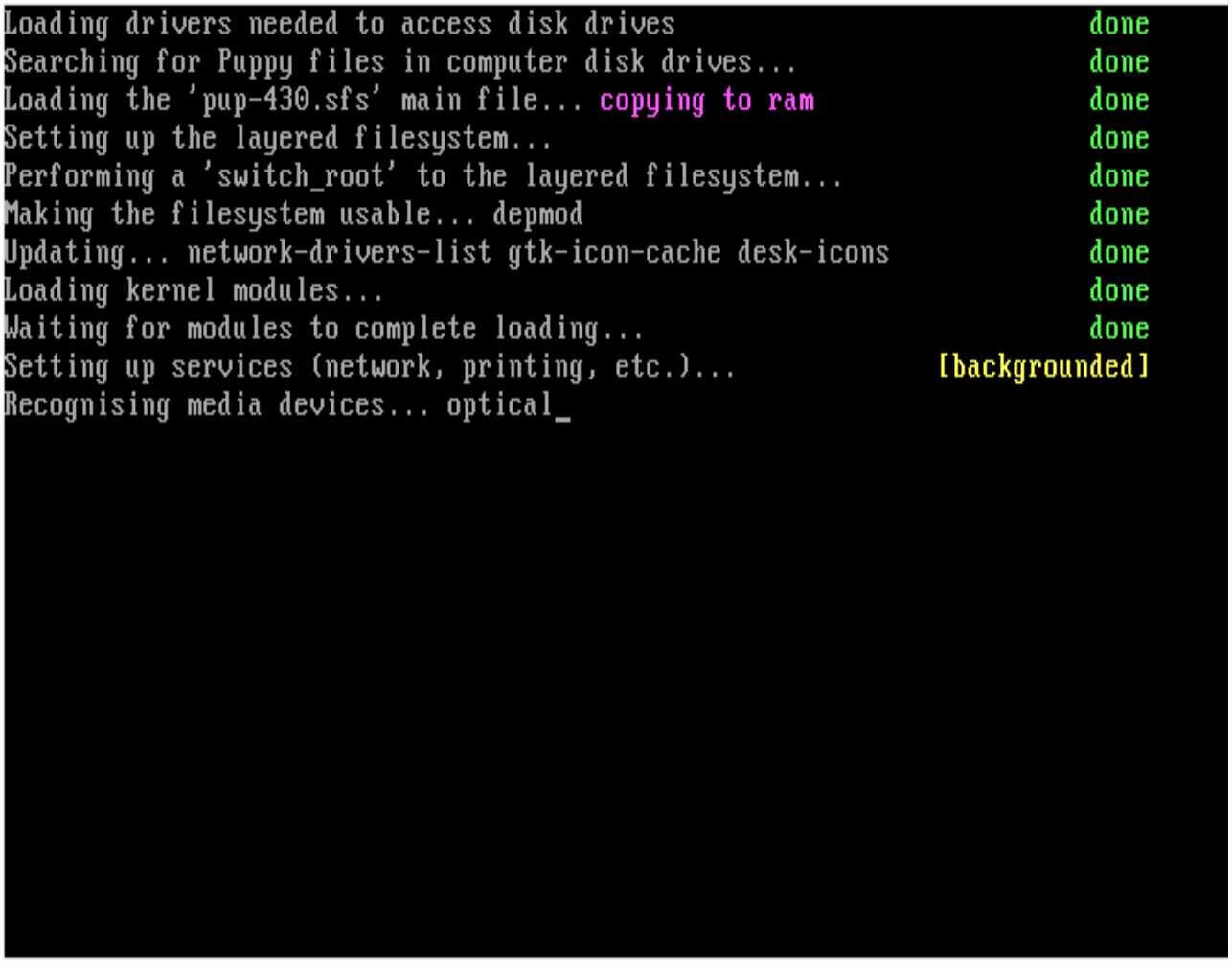


Figure 2- Kernel Module Loading

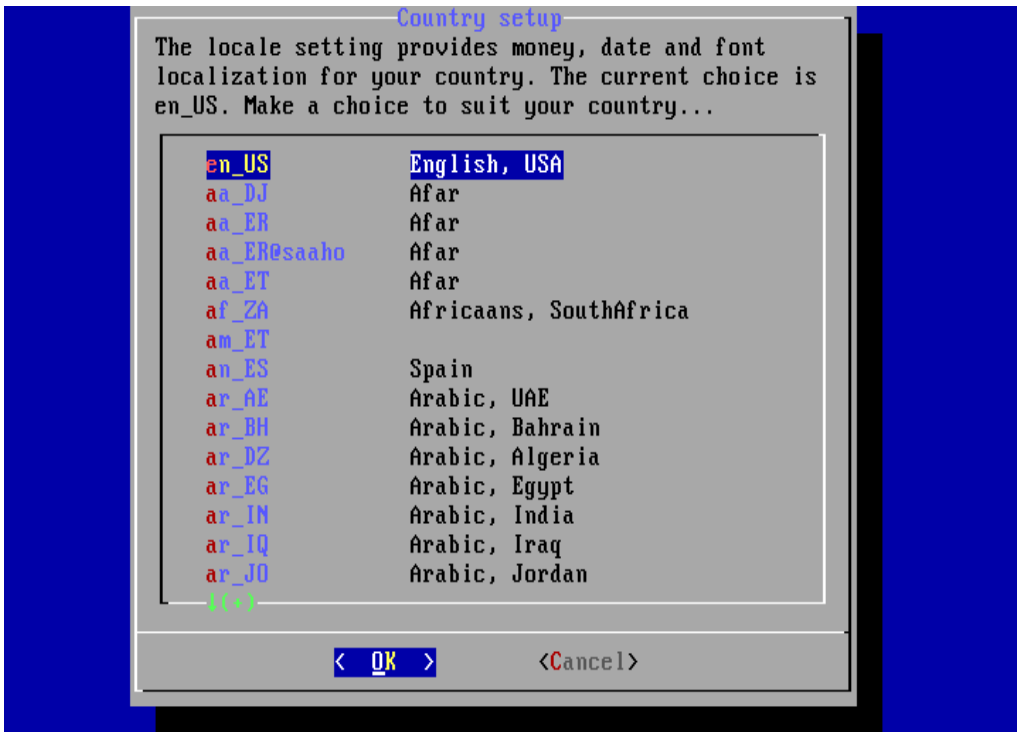
2.2.1 Keyboard Selection

Once the kernel image and drivers are copied into RAM, Puppy Linux will request you to select keyboard layout from known list of keyboard types,

Use Up/Down arrows on the keyboard to make your choice then press ENTER/RETURN for OK. Select the us qwerty (USA) option from list.

Figure 2- Keyboard Selection Wizard

2.2.2 Country Setup



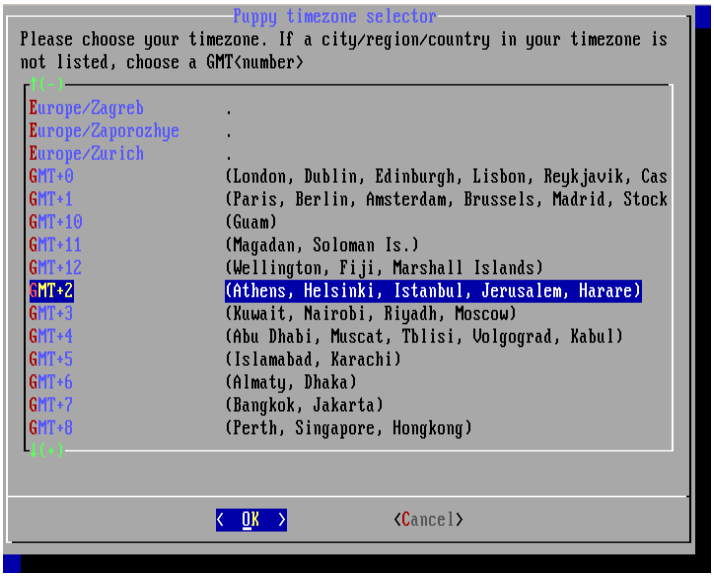
Select the country for localization of language, money, time and font from country setup wizard

Use Up/Down arrows on Keyboard to make your choice then press ENTER or RETURN for Ok.

Select the **en_US English,USA** option from list.

Figure 2- Country Setup Wizard

2.2.3 Time-zone setup



Select the time-zone from time-zone setup wizard, Use Up/Down arrows on keyboard to make your choice then press ENTER/RETURN for OK.
Select the **Asia/Calcutta** option from list.

Figure 2- Time-zone Setup Wizard

2.2.4 Puppy Video Wizard

Once keyboard, country and timezones are set puppy will ask to configure the Xserver to run puppy in graphic mode, Puppy has two Xserver namely XVESA and XORG. The following screen shots will guide to configure the XORG server.

Figure 2- Puppy Video Wizard - 1

Select <Xorg > from puppy video wizard. After choosing **Xorg** puppy will probe for monitor type and list the known types,

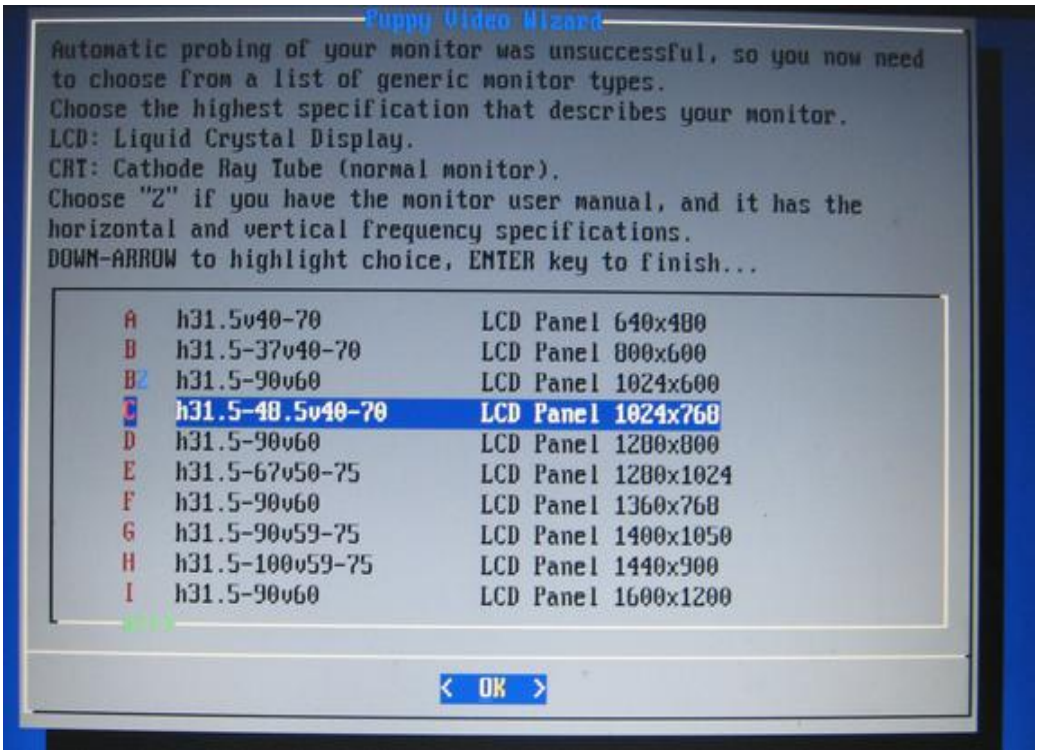


Figure 2- Puppy Video Wizard 2

Choose C h31.5-48.5v40-70 LCD Panel 1024x768 from listed options, Once monitor type is selected puppy will request you to select the video mode,

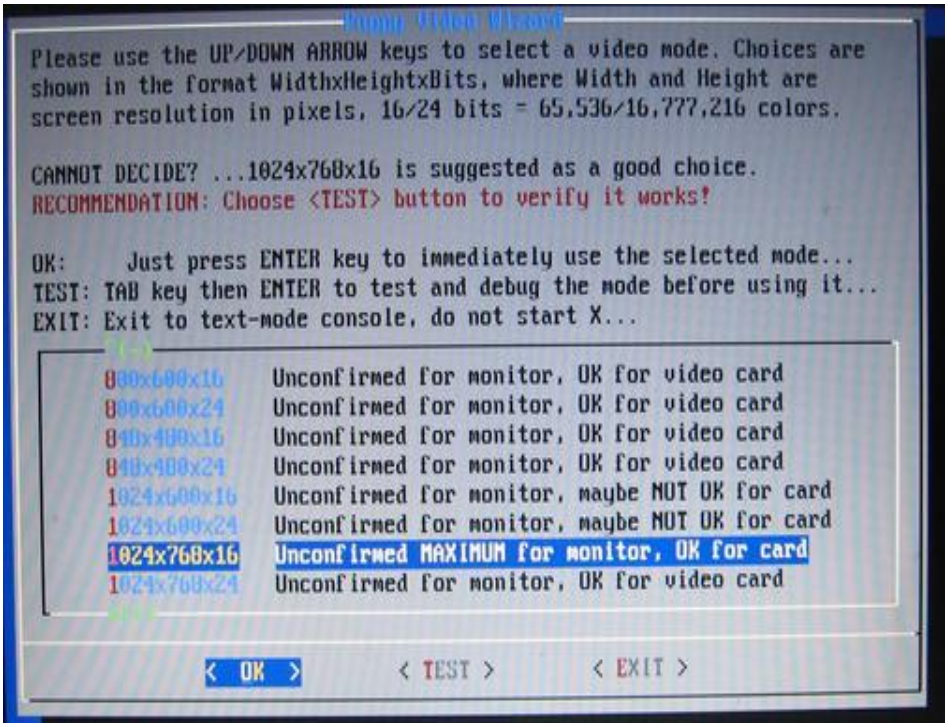


Figure 2- Puppy Video Wizard 3

Choose 1024x768x24 Unconfirmed for Monitor, OK for card from listed options,

At the end of video wizard puppy will take you into desktop. The desktop screen shot is shown below,

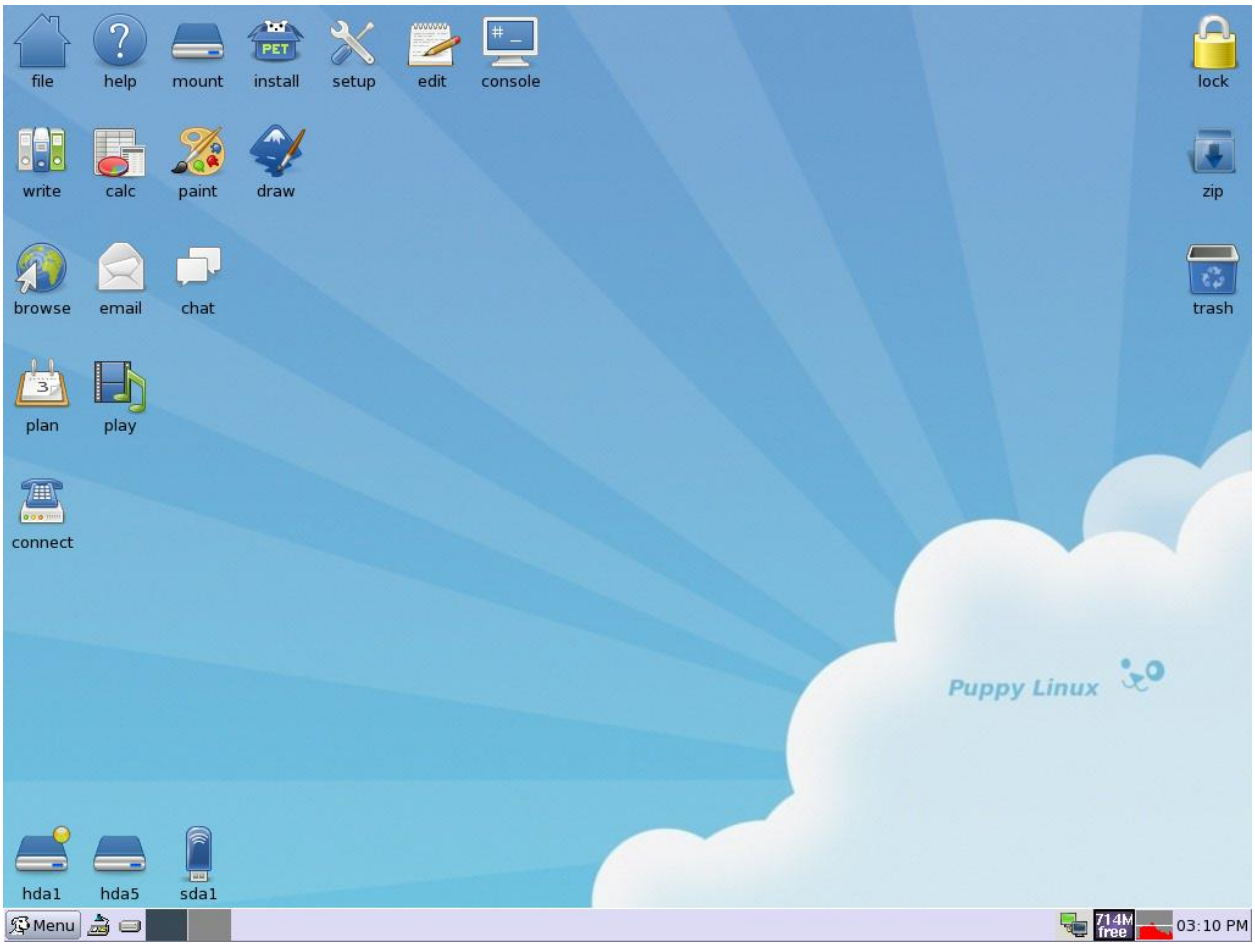


Figure 2- Puppy Linux Desktop

2.3 Hard Disk Partition

Disk partitioning basically divides the data storage space of a hard disk drive into separate areas referred to as partitions. Partitions are usually created when the hard disk is first being prepared for usage. Once a disk is divided into partitions, directories and files may be stored on them. Puppy Linux comes with GParted utility to partition HDD.

The following partition layout is to be created on installed HDD,

S.No	Partition	File System Type	Size
1	/dev/sda1	Ext3	1922MB
2	/dev/sda2	logical	2016MB
3	/dev/sda5	Ext3	2016Mb

2.3.1 Run GParted to Partition HDD

Goto **Menu -> System -> GParted**

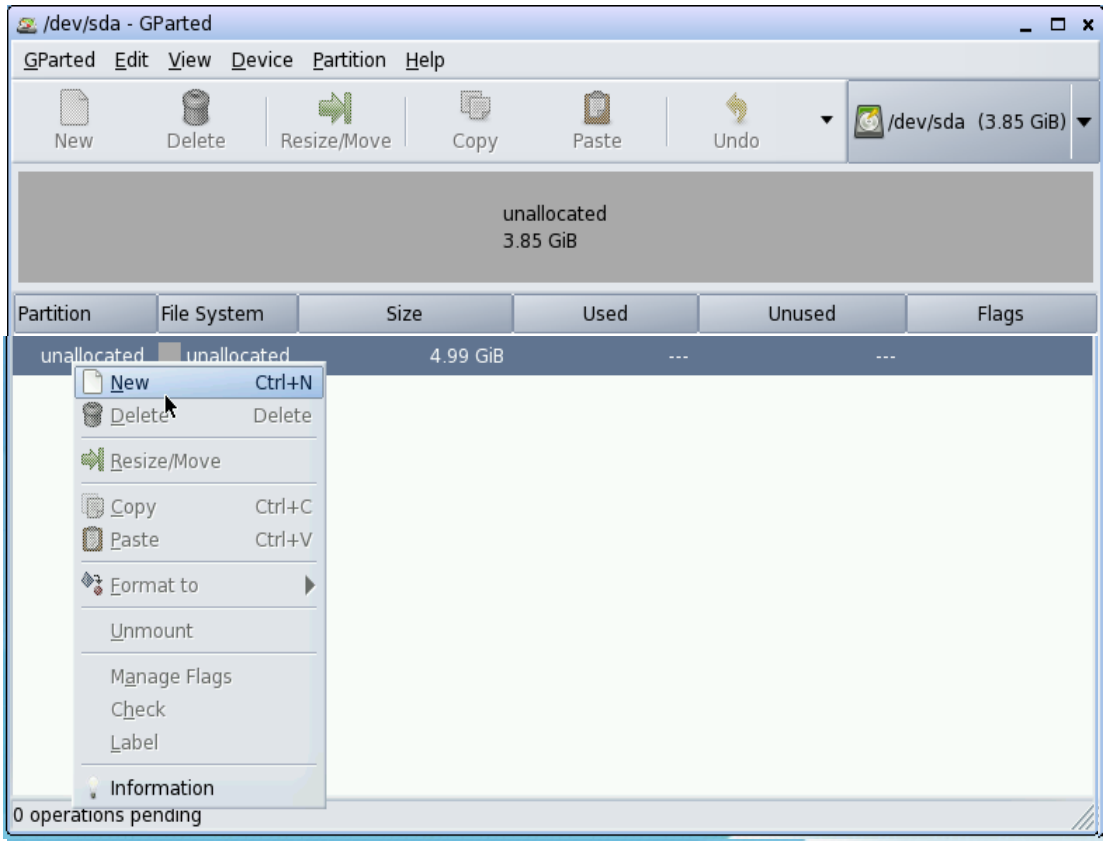


Figure 2- GParted HDD Partitioning Wizard 1

Select the unallocated partitions on the GParted window and right click on it. Popup window will open click on <**New**> option.

2.3.2 Creating Primary Partition

GParted will open “Create New Partition” window as shown in figure 2-11.

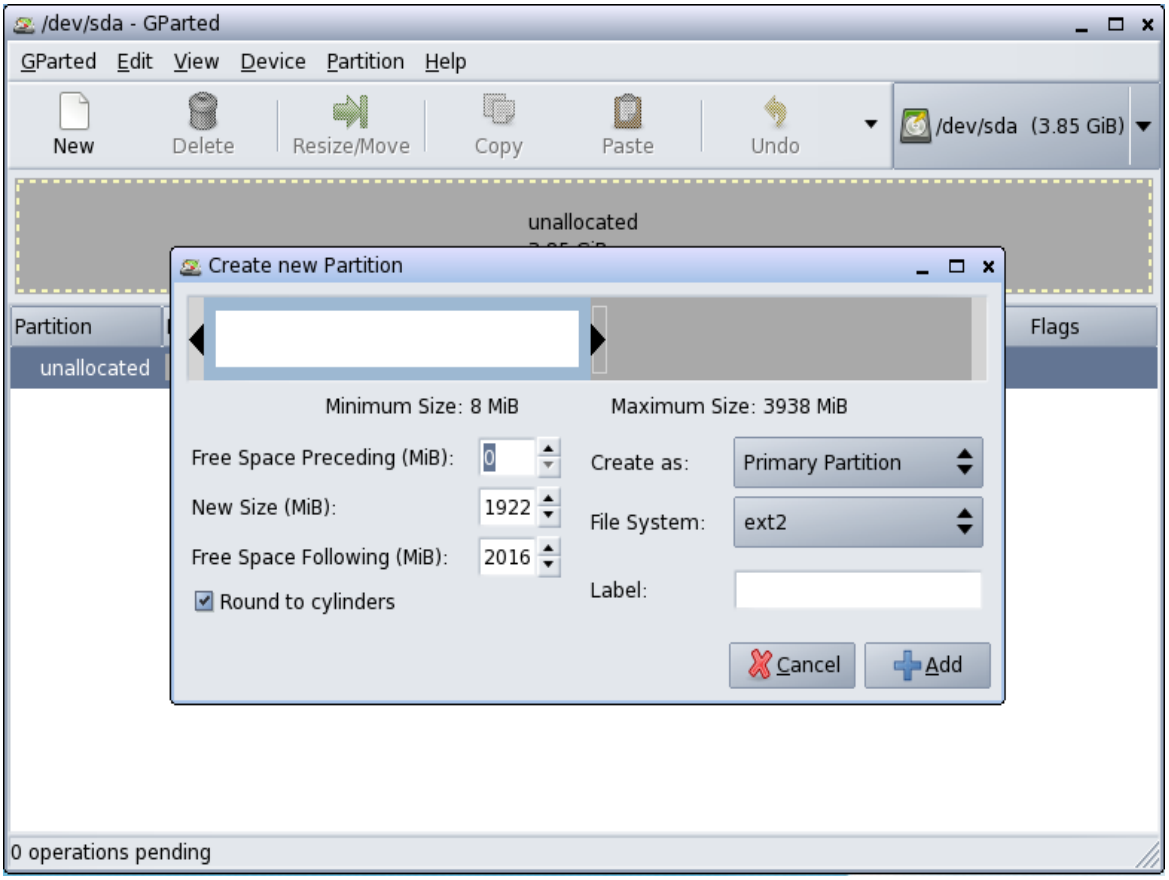


Figure 2- GParted HDD Partitioning Wizard 2

1. Adjust the slider to allocate required size of partition in our case it is close to 1922 MB.
2. Choose the partition type from “Create as SpinBox” (Primary Partition)
3. Choose the file system type from “File System Spin Box” (ext3)
4. Click **“<Add>”**
5. Confirm unallocated space divided into “New Partition #1 1922MiB” and unallocated space 2016MiB.

2.3.3 Creating Logical Partition

1. Select the unallocated partition from GParted window and right click on it, popup window will open as shown earlier and click on <**New**> option.

2. Gparted will open “Create new partition” window as shown in figure 2-12,

3. Adjust the slider to allocate maximum available size of HDD (2016 MB) ,

4. Choose the partition type from “Create as SpinBox” (Extended Partition),

   Note: File system SpinBox is disabled for Extended Partition,

5. Click “<**Add**>”.

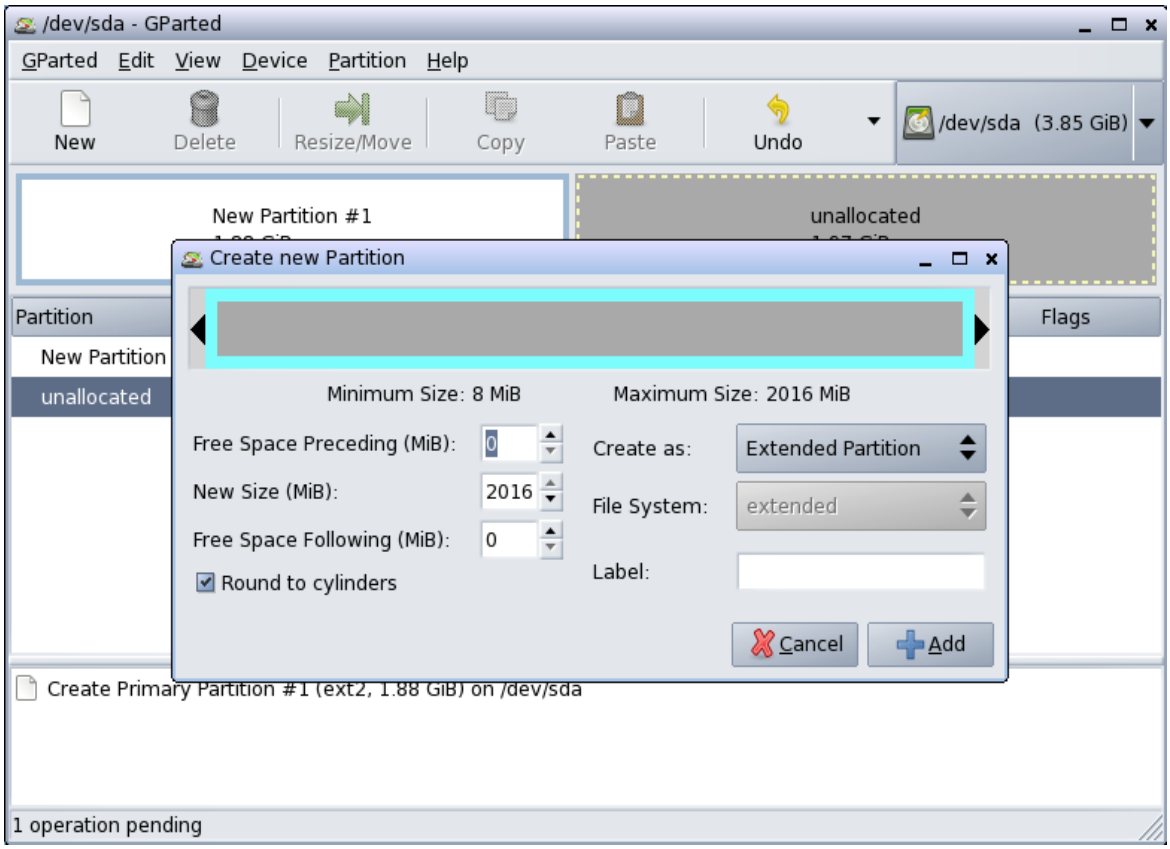


Figure 2- GParted HDD Partitioning Wizard 3

2.3.4 Creating Secondary Partition

1. Expand the newly created extended partition,

2. Select the unallocated partition, right click on it, popup window will open as shown earlier and click on <**New**> option.

3. Gparted will open “Create new partition” window as shown in figure 2-13,

4. Adjust the slider to allocate maximum available size of HDD, (2016 MB),

6. Choose the partition type from “Create as SpinBox” (Logical Partition),

7. Choose the file system type from “File System Spin Box” (ext2),

8. Click “<**Add**>”.

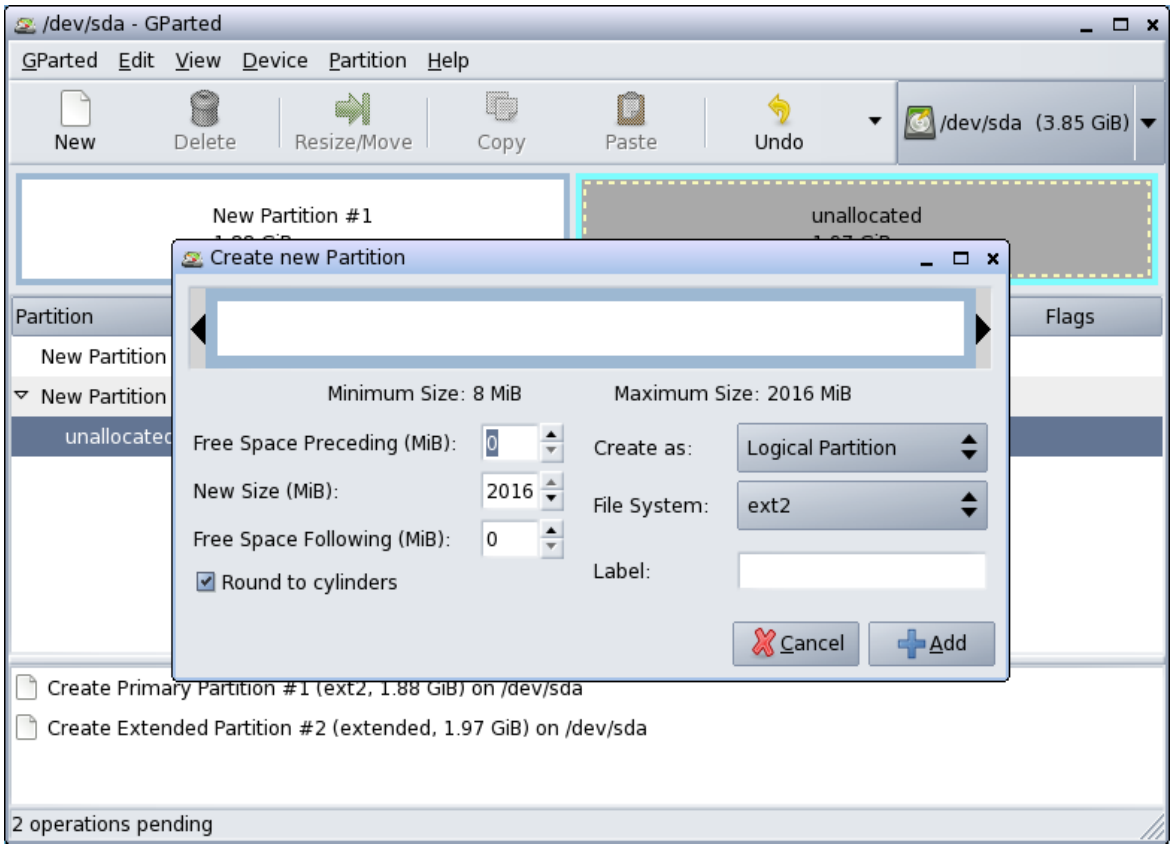


Figure 2- GParted HDD Partitioning Wizard 4

After creating partition layout on the selected HDD, it should be saved by clicking on

Edit → Apply All Operations

Accept the defaults on next screen (figure 2-14) and "<**Close**>"

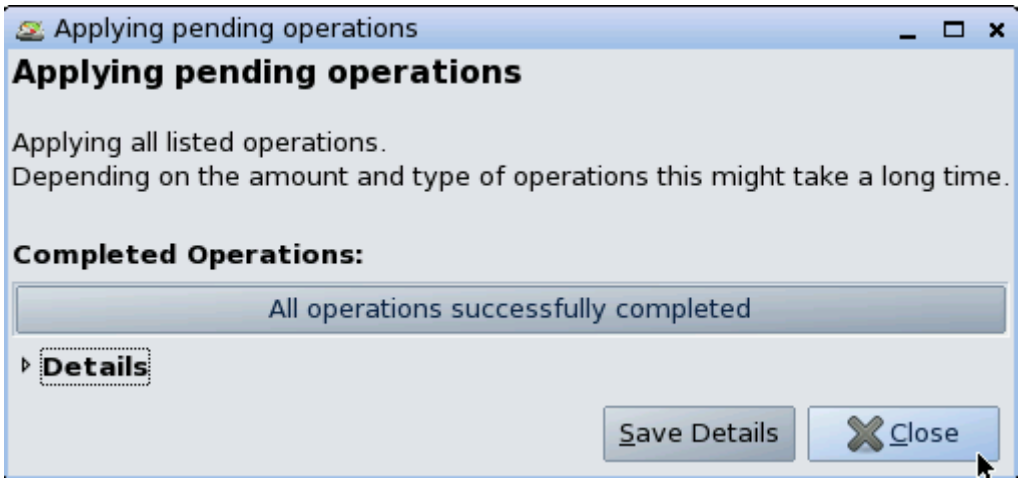


Figure 2- GParted HDD Partitioning Wizard 5

Finally we have prepared HDD for storing files and can exit the GParted menu.

2.4 Installing Puppy Linux into HDD

Puppy Linux comes with pre-build software called "*Puppy Universal Installer*". This utility makes ease of our installation processes.

Click on sdb1 

in desktop to mount boot media and close the opened window.

Goto

Menu → Setup → Puppy Universal Installer

Puppy Universal Installer window will open as shown below (figure 2-15), select the media to install puppy from listed options,

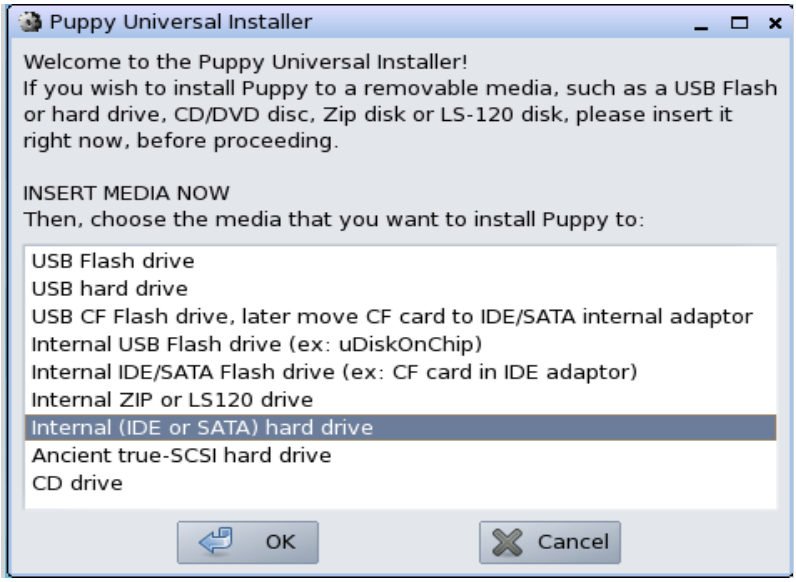


Figure 2- Puppy Installation Wizard 1

Choose Internal (IDE or SATA) hard drive. If you have multiple HDD select specific drive from the available options on next screen as shown in figure 2-16.

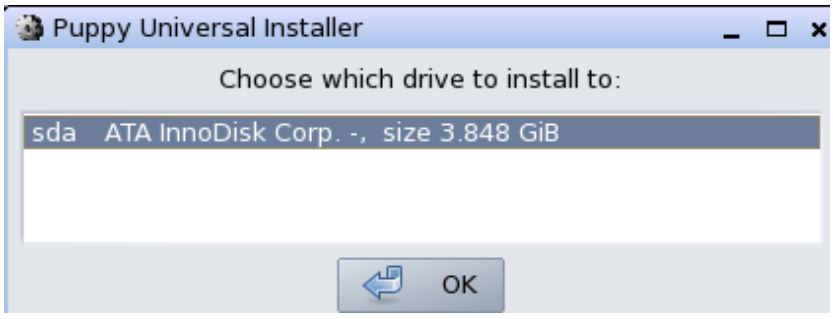


Figure 2- Puppy Installation Wizard 2

Select sda ATA InnoDisk Corp. -, size 3.848 GiB

From the chosen HDD, Puppy will list available partitions on that drive on next screen (figure 2-17), choose partition to install Puppy Linux

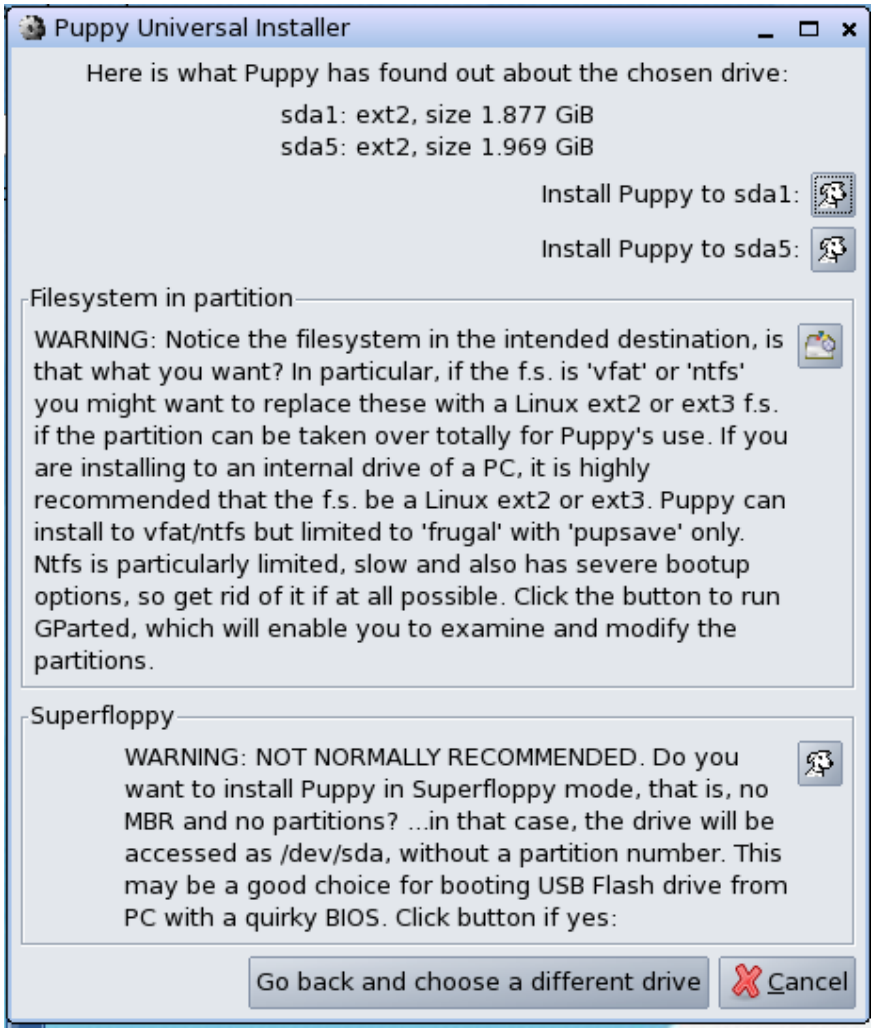
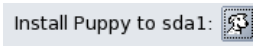


Figure 2- Puppy Installation Wizard 3

Click on "***Install Puppy to sda1***"  option.

Next screen (figure 2-18) will confirm the selected partition. (Accept default)

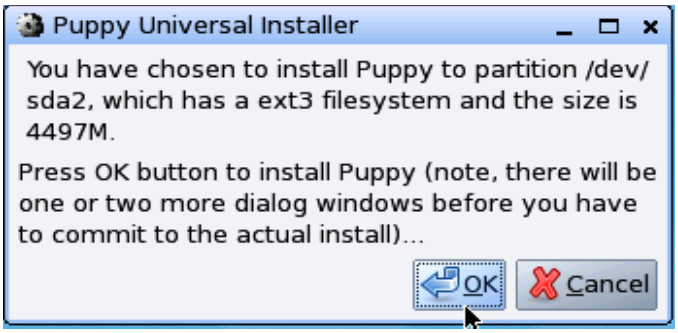


Figure 2- Puppy Installation Wizard 4

Next Puppy will request you to select type of installation as shown in figure 2-19,

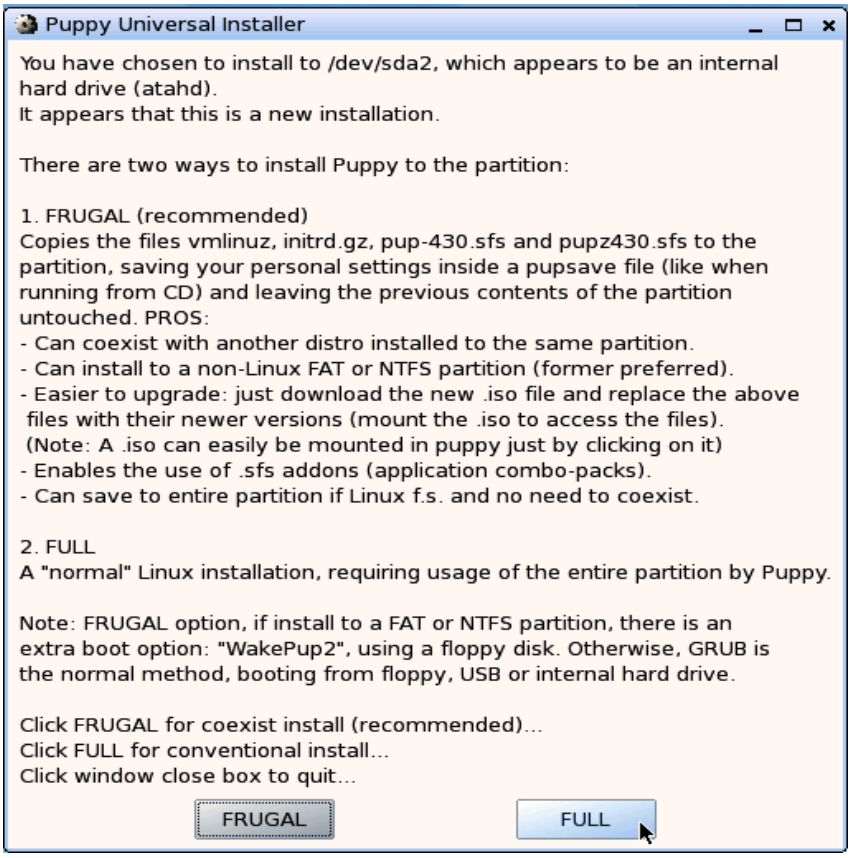


Figure 2- Puppy Installation Wizard 5

After selecting Full installation puppy will copy installation file from boot media into selected HDD partition as shown in figure 2-20.

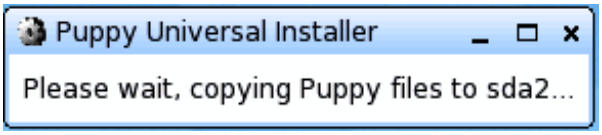


Figure 2- Puppy Installation Wizard 6

Finally puppy executes the configuration and run level scripts as shown in figure 2-21,

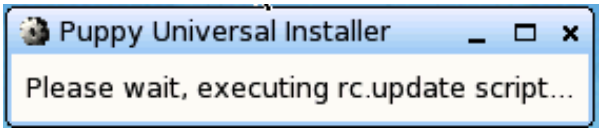


Figure 2- Puppy Installation Wizard 7

Between these steps sometimes you may have to wait a minute or two so be patient and all will continue as planned, finally puppy will ask you install the GRUB, next section will explains GRUB installation procedure.

2.5 Setting up GRUB

The installation of GRUB in Puppy Linux is easy. After successful installation of puppy on HDD, by default puppy will run GRUB installation scripts. The following screen shots (figure 2-22) will guide you to install GRUB on HDD,

- Select the "***Install/Upgrade***" option.

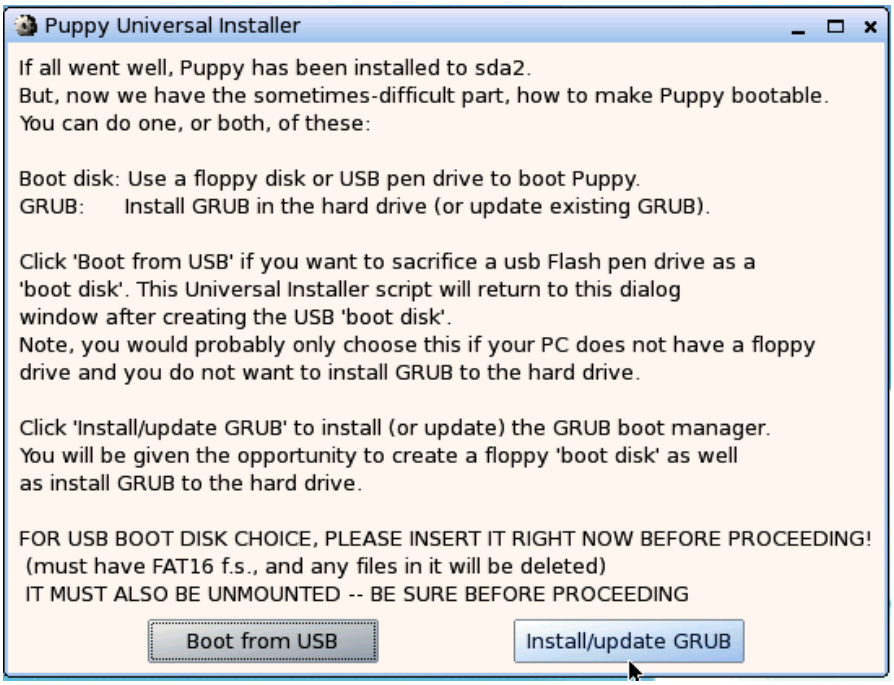


Figure 2- GRUB Installation Wizard 1

- Select the "***INSTALL***" option from the installer window shown in figure 2-23.

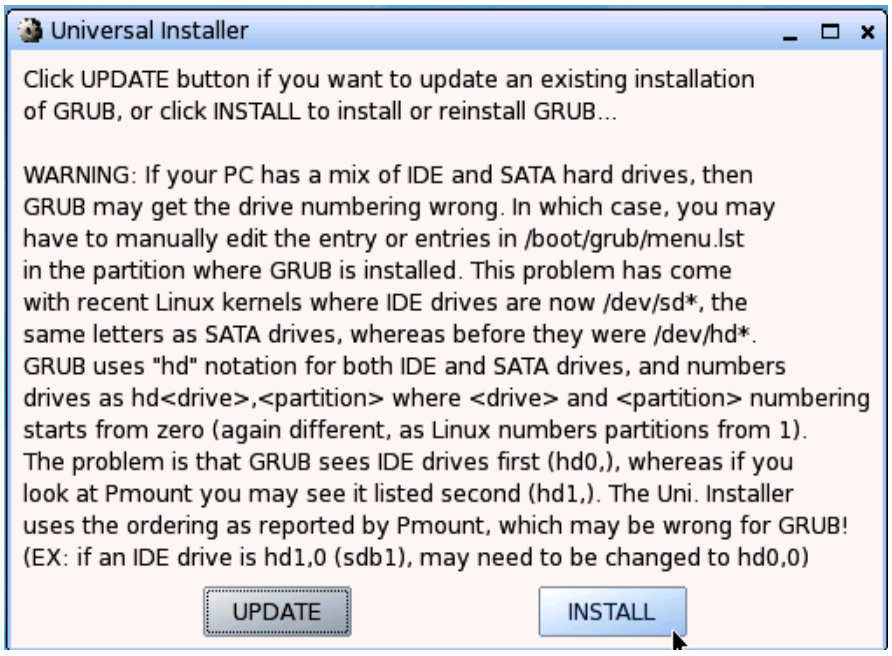


Figure 2- GRUB Installation Wizard 2

- Accept the defaults in the following screen (figure 2-24).

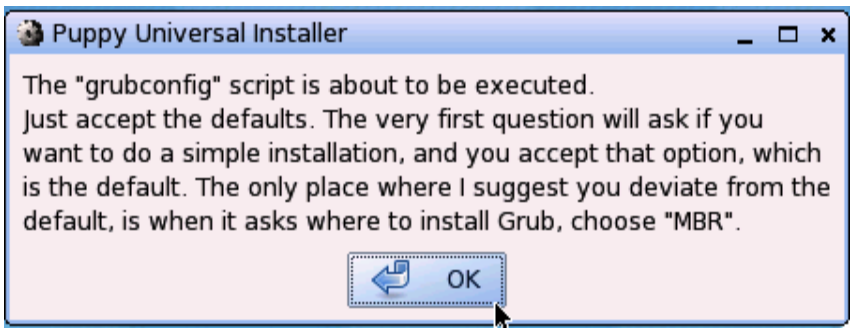


Figure 2- GRUB Installation Wizard 3

- Select "***SIMPLE***" option from next dialog (figure 2-25)

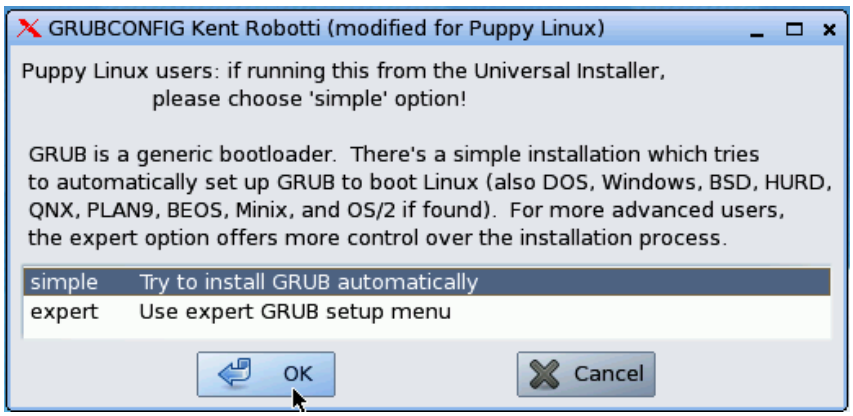


Figure 2- GRU B Installation Wizard 4

- Select frame buffer console from available options as shown in figure 2-26. In our case select "standard Use standard Linux Console (the safe choice)"

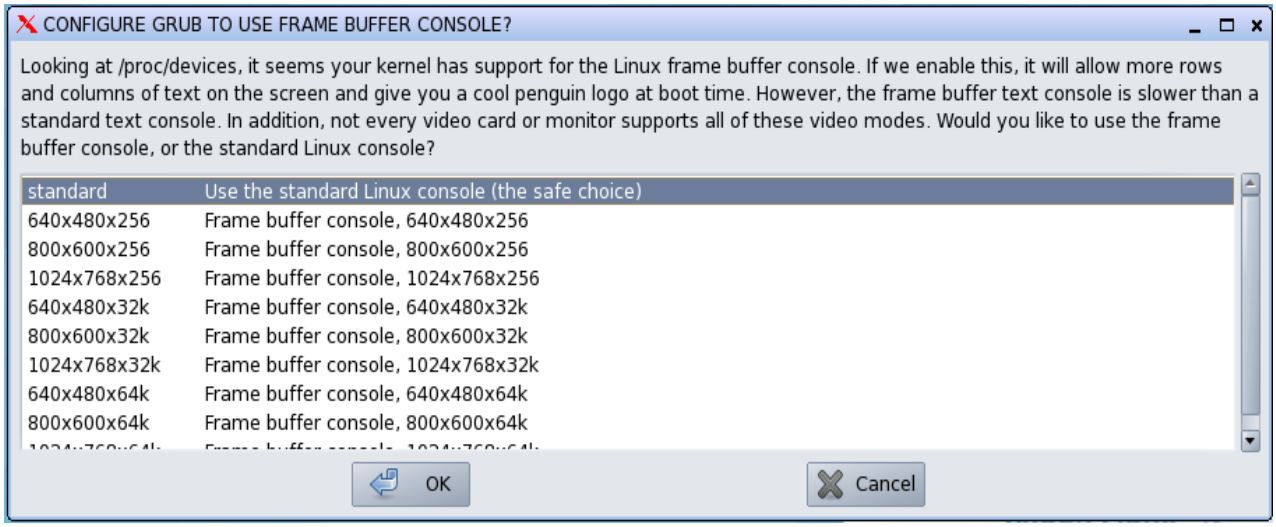


Figure 2- GRUB Installation Wizard 5

- Give the partition details to store GRUB files, (If you are running GRUB from Universal Installer Accept the default entry) on the following screen (figure 2-27),

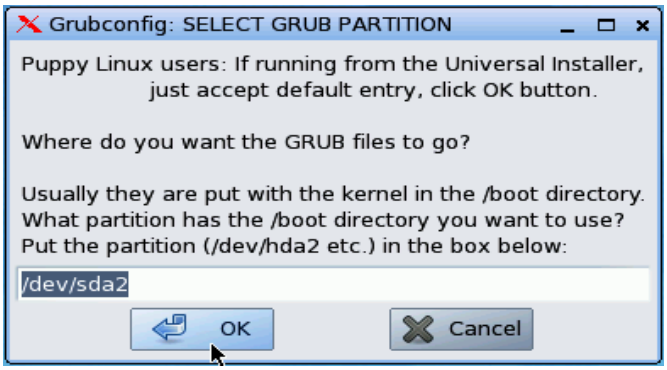


Figure 2- GRUB Installation Wizard 6

- Select the GRUB Destination from the available options in the next screen,

  Select MBR Install to Master Boot Record (possibaly unsafe) option and click <**OK**>

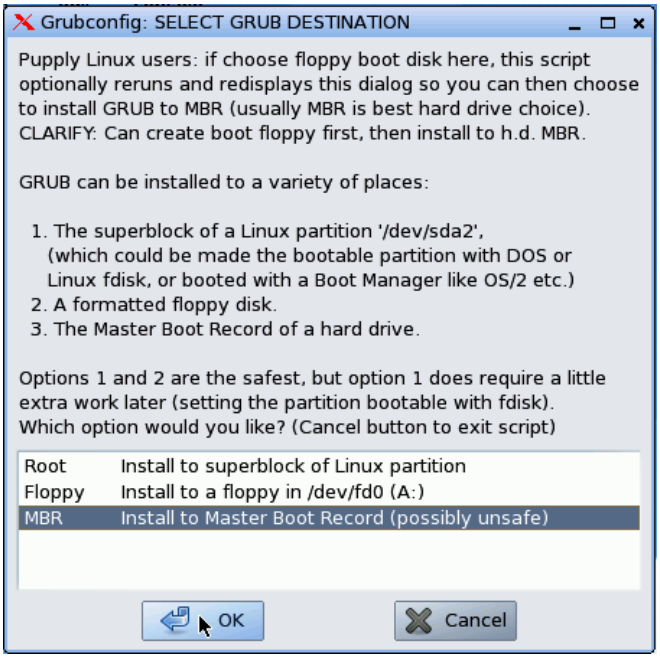


Figure 2- GRUB Installation Wizard 7

Finally GRUB installer reports the success message as shown in figure 2-29,

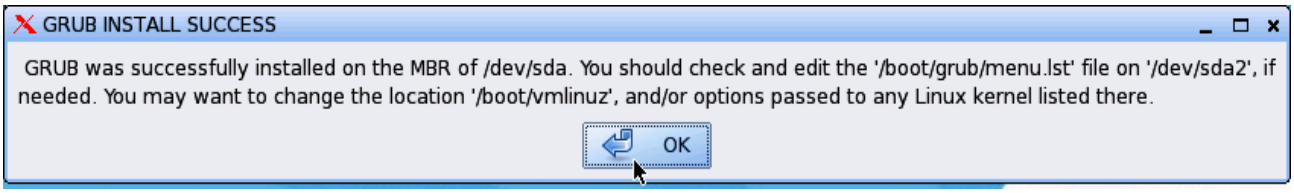


Figure 2- GRUB Installation Wizard 8

At the end of GRUB configuration installer will opt you to exit the installer or want to re-run the grub as shown in figure 2-30. Select the “No” and exit installer.

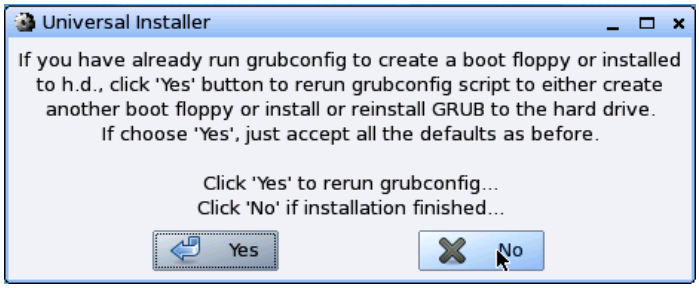


Figure 2- GRUB Installation Wizard 10

After installing GRUB the installation is almost finish. All you have to do is reboot into your newly installed Puppy Linux system.

Reboot

Menu → Shutdown → Reboot

While rebooting puppy will offer you to save your personal settings and files as shown figure 2-31,

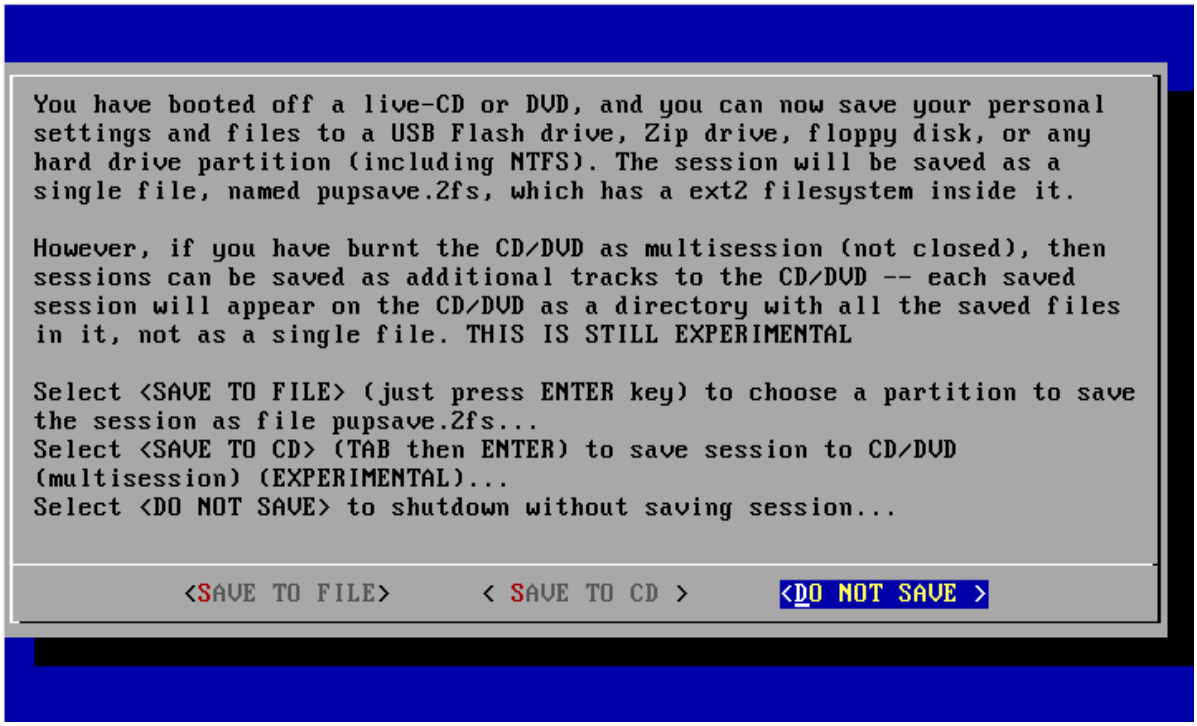


Figure 2- Puppy Conformation Wizard

Select **<DO NOT SAVE >** and Enter. Finally remove boot media.

2.6 Rebooting into Puppy

Once system has rebooted, GRUB screen will opt you to select the version of the Puppy Linux select Linux (on /dev/sda1) option and press, <Enter>. To get into puppy desktop you may need to select the keyboard, country, time-zone and Xserver setup that we done in section 2.2.1 to 2.2.4 after successful configuration puppy will take you into desktop as shown in figure 2-32.

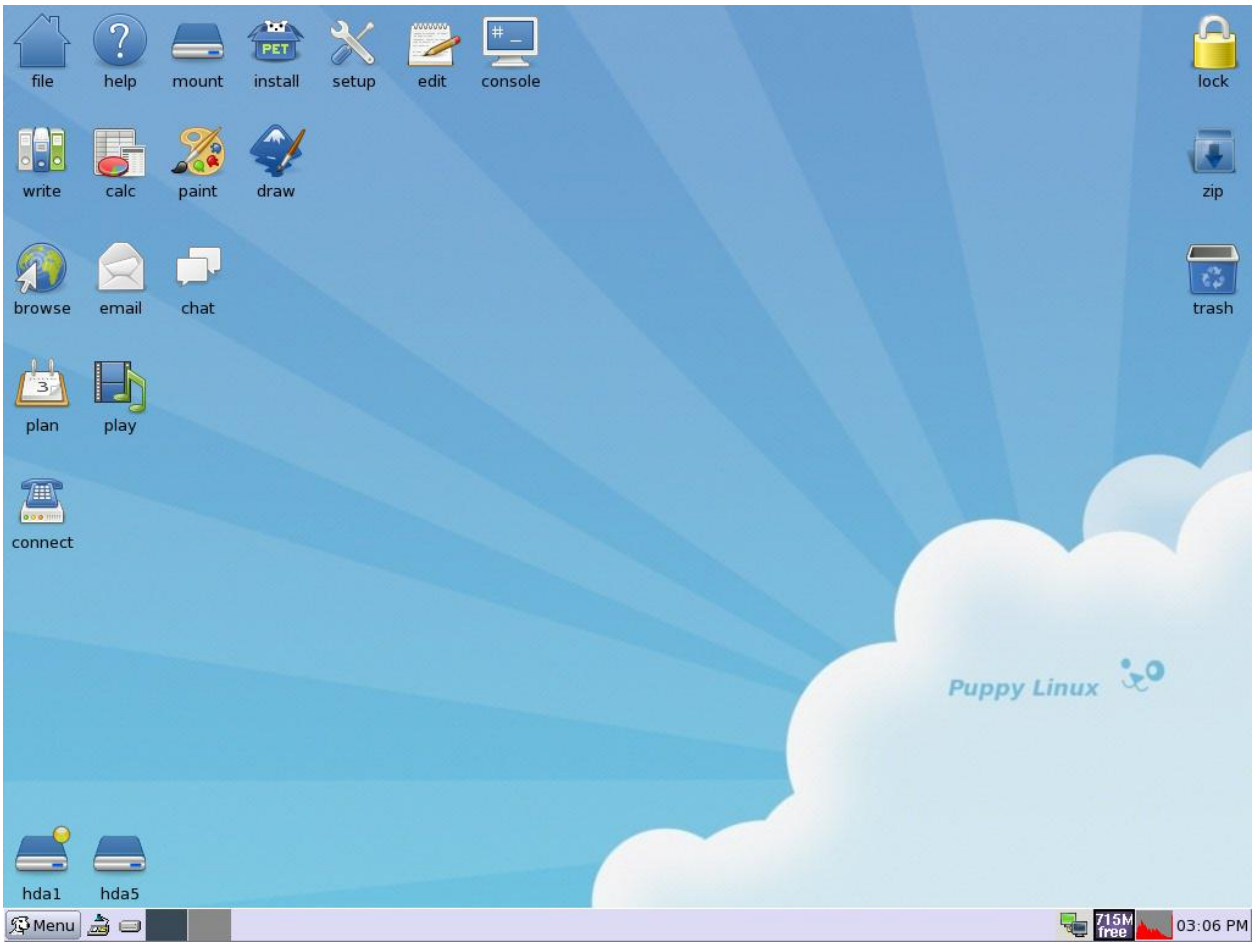


Figure 2- Puppy Linux Desktop

3. Compiler Installation

To make puppy into complete compiling environment we need to install add-on file called ***devx_xxx.sfs*** (xxx refers version of installed Puppy Linux our case 4.3.1). It contains all of the tools for compiling C/C++ applications.

The devx_431.sfs file installation is explained below:

Insert the media having devx_431.sfs file (In our case insert USB) wait for a second till USB is detected by Puppy Linux and displayed on desktop as shown in figure 3-1.

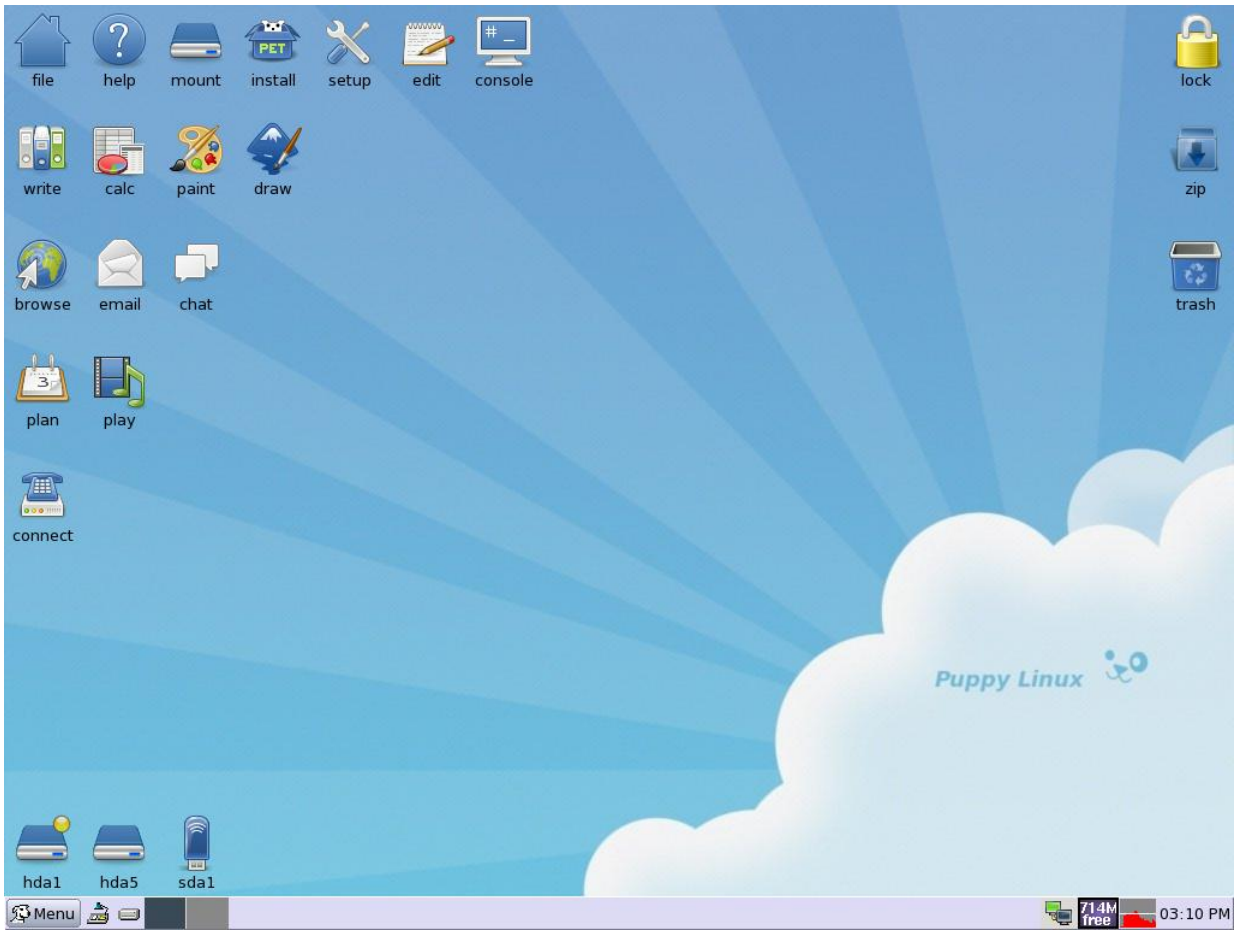


Figure 3- Desktop with Detected USB

To mount USB, click on it. It will be automatically mounted and rox file manager displays it content as shown in fig 3-2.

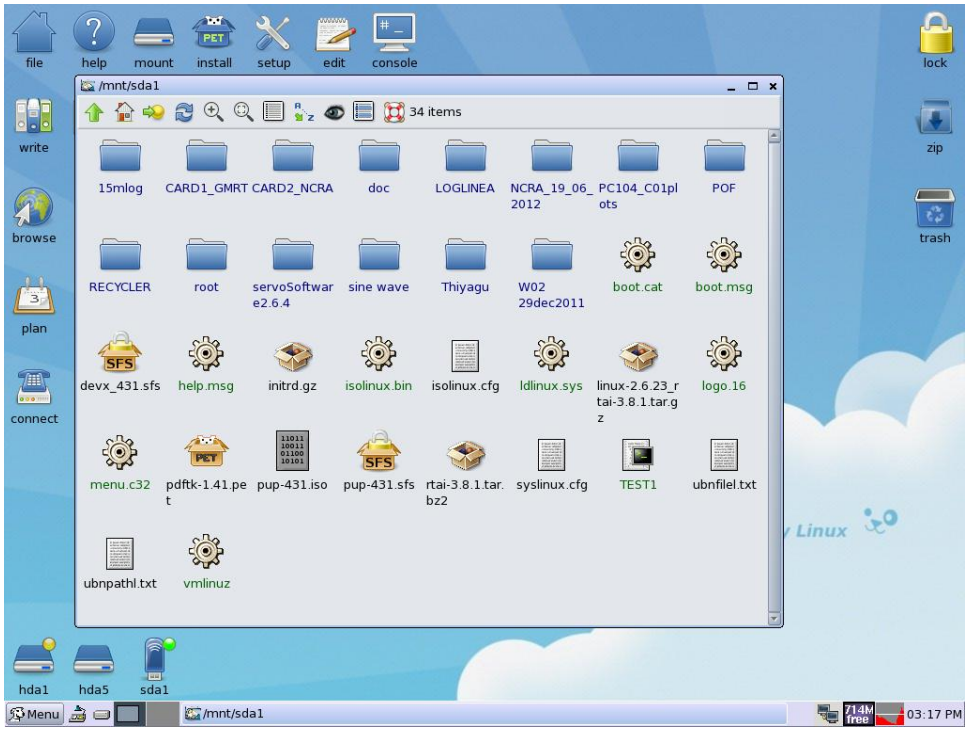


Figure 3- ROX File Manager with Mounted USB

Click on devx_431.sfs to mount. The following figure 3-3 displays status of mounting and shows content of devx_431.sfs

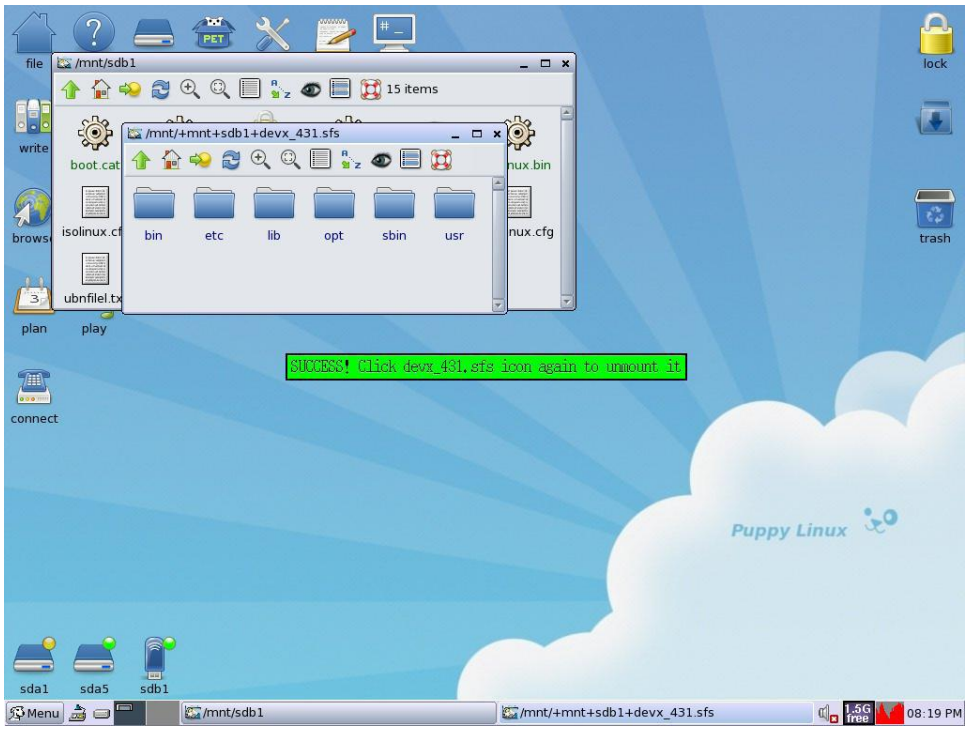


Figure 3- Directory Listing of devx_431.sfs

Goto the terminal

Menu → Utility → Rxvt terminal emulator

Copy the files in mounted devx_431.sfs files into /mnt/home/ by giving following command as shown below fig 3-4,

```
# cp -a -remove-destination /mnt/home/mnt/+mnt+sdb1+devx_431.sfs/*
/mnt/home/
# sync
```

Figure 3- Command to copy devx files

Goto /mnt/sdb1 file manager window by clicking it in task bar,

Click on the devx_431.sfs to unmount. Puppy will confirm with the message "**Unmounting dexx_431.sfs**".

Unmount the USB by issuing following command in terminal as shown.

```
# cd
# umount /dev/sdb1
```

This will completes installation of devx_431.sfs file. To check installed version of *gcc* issue following command in terminal

```
# gcc -v
```

The above command will displays *gcc* version and path to binaries, flags that are set during compilation.

4. RTAI Installation

This section covers all the steps needed to install RTAI on a Puppy Linux environment, including kernel configuration. The main objective is to provide all the necessary information in a detailed manner, even for those who possess little or no knowledge about the kernel compilation process.

The overview of whole process, together with a complete set of steps for building and installing RTAI from scratch are listed below,

- Selection of RTAI version
- Installing the Puppy Linux,
- RTAI supported Linux Kernel version,
- Downloading the Linux Kernel and RTAI tar files,
- Unpack and uncompressing the files,
- Using the patch file in the RTAI release that corresponds to the kernel source you downloaded, patching the kernel source to incorporate the RTAI modifications,
- Generating a configuration file that suites the machine you're going to run on.
- Building the Linux kernel
- Installing the Linux kernel,
- Configuring RTAI,
- Building RTAI,
- Installing RTAI,
- Rebooting and Testing.

4.1 Selection of RTAI and Linux Kernel Source

We need to select version of RTAI, Linux kernel and a Linux distribution to use. These have to be compatible, which is the tricky bit. Each version of RTAI supports only a limited subset of Linux kernel versions, and we have to use the exact version of, one of those RTAI patch file with Linux kernel version. With the available system resources in GMRT we selected Puppy Linux for Linux distribution. It uses Linux kernel version 2.6.30.5. The latest release of RTAI-3.8.1 has been used as it has patch for Linux kernel 2.6.23.

Summary of selected software versions,

Linux Distribution : Puppy Linux – 4.3.1

Linux Kernel : 2.6.23

RTAI Version : RTAI-3.8.1

4.2 Installing the Linux Distribution (Puppy Linux -4.3.1)

Refer Section 2.0 *Puppy Linux Installation* of this document, which explains the detailed steps to install Puppy Linux distribution on ICOP VDX6354D embedded boards and section 3.0 *Compiler Installation* will turns on the puppy into complete compiler for compilation of Linux kernel source and various high level language programs.
This document describes two method of RTAI installation.

- Generic method of Linux Kernel and RTAI Installation
- GMRT method of Linux Kernel and RTAI Installation

4.3 Generic method of Linux Kernel and RTAI installation

In this method of installation, installer needs a good back ground in Linux kernel building and kernel configuration files. If installer not selected the proper driver for hardware, CPU type, file system type etc in the main *.config* file it may lead into problem, that time installer may able to recover the system back. It is highly encouraged to follow the GMRT method of installation.

4.3.1 Downloading the Linux Source

The source for the Linux kernel can be found at www.kernel.org. The home page will lead you directly to the source for the latest stable release. And also it has a repository of all the source of previous releases, with HTTP and FTP links to them.

For example www.kernel.org/pub/linux/kernel/v2.6, this has all the various 2.6 releases. We can easily download 2.6.23 by following link, http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.23.tar.bz2, put the tar file into */usr/src* directory if exist otherwise create a *src* directory under */usr*.

Command line way to download the Linux source from kernel.org

```
$ wget -c -P /usr/src
http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.23.tar.bz2
```

4.3.2 Downloading the RTAI files

The RTAI homepage is at www.rtai.org. Download the copy of latest RTAI release from homepage and put it in */usr/src* directory. As like Linux kernel RTAI also maintains archive of previous versions, pick the required version from following link https://www.rtai.org/RTAI/rtai-3.8.1.tar.bz2. Alternately you can download by using wget command

```
$ wget -c -P /usr/src http://www.rtai.org/RTAI/rtai-3.8.1.tar.bz2
```

4.3.3 Unpacking the files

Extract the Linux kernel and RTAI files in */usr/src* directory,

```
$ tar -xf linux-2.6.23.tar.bz2

$ tar -xf rtai-3.8.1.tar.bz2
```

Now we have two directories called rtai-3.8.1 and linux-2.6.23 in */usr/src*.

4.3.4 Applying the RTAI patch

The RTAI patch is applied just like any Linux kernel patch, it adds additional configuration options to main kernel source. The following command applies RTAI patch to Linux kernel 2.6.23.

```
$ cd /usr/src/linux-2.6.23/
$ patch –p1 < /usr/src/rtai-3.8.1/base/arch/i386/patches/hal-
linux-2.6.23-i386-1.12-03.patch
```

4.3.5 Configuring the Linux Kernel

In Linux build processes; what modules get included in the kernel is controlled by a configuration file at the top level of the Linux source tree (i.e. /usr/src/linux-2.6.23). This is the directory where almost all of the work involved in building the kernel takes place. Each configuration option in the Linux kernel has a C pre-processor variable associated with it. Most options need to be set to “yes”, “no” or “module”, where “yes” means compile it into the basic kernel, “no” means leave it out completely and “module” means include it in a loadable module.

Once you have a *.config* file, there are some useful tools that can be used to modify it and fine-tune it for your system, all invoked as variants of the ‘make’ command,

make oldconfig:

This takes an existing .config file, usually one used for an earlier release of Linux on your system, and asks you about all the 'new' configuration settings – ones for which the old file doesn't have a setting, and so which presumably represent new options added in later releases. All it does is prompt you for the various undefined options one by one, with a bit of online help. You can't go back to a previous question without starting over again, although you can fix it up later using 'xconfig'.

make xconfig:

This is the rather nice GUI configuration tool provided in Linux 2.6. It lets you move around the options, which are grouped fairly conveniently, and provides on-line help when you click on them. Selecting some options brings up some others – for example, if you select one of the ethernet options, say support for 100Mbit ethernet, you find yourself presented with a bewildering array of options for drivers. This GUI is built using Qt, so you need that to be installed on your system.

make menuconfig:

Is the older, rather clunkier GUI provided in Linux 2.4. Similar to xconfig, but not so easy on the eye.

4.3.6 Compiling the kernel

After configuring kernel we have .config file. Now we need to make the main kernel file, which is kept in */boot* in a compressed form and is loaded as part of the boot process. This file is called *bzImage* when it is built. We also need to build all the kernel modules. For building *bzImage* and kernel modules it takes the following commands,

```
$ make or make bzImage
$ make modules
```

Both of the above commands are issued without any privileges. We should do them logged in as a normal user. Compilation time will vary depends on the options/modules are selected, as well as processor capability. At the end it builds bzImage and kernel modules.

4.3.7 Installing the Kernel

Installing the kernel is split into following parts,

a. Installing the kernel modules,

b. Installing the kernel itself,

c. Creating initial ram disk image (initrd.img)

a. Installing Kernel Modules:

Kernel modules are normally to be place in */lib/modules/<version>*. Kernel modules are installed by issuing the following command,

```
# make modules_install
```

b. Installing Kernel:

Before we install the kernel, we want to make sure we do not overrun our current kernel, or previously existing kernel. So we will install the kernel itself manually. Runnable kernel are expected to be in */boot* directory. Simple way to install the kernel is copy the *bzImage* into */boot.*

```
# cp arch/i386/boot/bzImage /boot/bzImage-2.6.23
```

make sure the name is unique, and especially different than the current kernel.

c. Creating Initial Ramdisk image:

The initial RAM disk (initrd option in the GRUB menu, or, the file "initramfs-YourKernelName.img") is an initial root file system that is mounted prior to when the real root file system is available. The initrd is bound to the kernel and loaded as part of the kernel boot procedure. The kernel then mounts this initrd as part of the two-stage boot process to load the modules to make the real file systems available and get at the real root file system. The *initrd* contains a minimal set of directories and executables to achieve this, such as the *insmod* tool to install kernel modules into the kernel.

```
# mkinitrd /boot/initrd-2.6.xx.img 2.6.xx
```

The second argument, the version string allows *mkinitrd* to locate the files it needs to build the disk file whose name is given in the first argument. Finally copy the new system map file into */boot* directory and set up the correct symbolic link for it.

```
# cp System.map /boot/System.map-2.6.xx.map
# ln –s /boot/System.map-2.6.xx /boot/System.map
```

Now we need to fix the configuration file for our boot loader. Most 2.6 systems use GRUB boot loader. GRUB is controlled by a configuration file called "*/boot/grub/menu.lst*". Use the traditional "vi" editors to edit menu.lst file to add the new grub entry as looks below,

```
title linux-rtai
kernel (hd0,0) /boot/bzImage-2.6.xx ro root=/dev/hda1
initrd /boot/initrd-2.6.xx.img
```

Save the changes and exit. The disks and partitions are numbered by bus numbers, rather than by letters: hd0,0 means /dev/hda1. Having done all these, reboot the system and boot with new installed kernel. Check the version of running kernel by giving following command,

```
# uname -a
```

This will outputs the kernel name, host name, kernel release, kernel version and compiled date, hardware platform and operating system.

4.3.8 Configuring RTAI

RTAI is configured in a similar way as Linux kernel, using a variation on "*make menuconfig*". The RTAI installation instruction recommends building RTAI in a separate build directory, not the actual source directory. So create a directory called '*rtai-build*', set this directory as default directory for build and run "*make menuconfig*" in that, using the *–f* option to make to point it to the *makefile* provided in the RTAI source directory.

```
$ cd /usr/src/rtai-3.8.1/
$ mkdir rtai-build
$ cd rtai-build
$ make –f /usr/src/rtai-3.8.1/makefile menuconfig
```

In the absence of a .config file for RTAI itself, defaults are taken from the *rtai-core/arch/i386/defconfig* file. RTAI configuration feels very much like Linux kernel configuration, and this is obviously that is intended. The only exception was that the default location for the Linux kernel source which the RTAI build needs to access is set to the */usr/src/linux* directory structure, which needs to be changed if you put the kernel source in a personal location. So the only option to be changed in rtai .config file was,

General → Linux source tree (path to Linux Source tree) /usr/realtime

General → Installation Directory (Leave the default) /usr/src/linux-2.6.xx

Then save and quit. The RTAI system configures itself and generates all the makefiles it will need. Finally we are ready to build RTAI.

4.3.9 Building and Installing RTAI

Building the RTAI is much simpler, just issue the *make* command from the build directory (*/usr/src/rtai-3.8.1/rtai-build*)

```
$ make
```

The easiest way to clean things up if we want to repeat the RTAI build, is just issue the '*rm –rf rtai-build/*'*.

To install the compiled RTAI modules we need to be root, following command will install the compiled modules,

```
$ su
# make install
```

4.3.10 Rebooting & Testing

Now you need to reboot your new system and boot with rtai patched kernel. If every thing goes fine we are ready to test the RTAI. Basically RTAI comes with a number of test programs. One of these runs a latency test, rescheduling itself at a 10 KHz rate and measuring the difference for each reschedule between the actual and excepted rescheduling times. Most of RTAI modules and test programs installed in */usr/realtime* to run the latency test you need to become root and execute the '*run'* command for the test,

```
$ su
# cd /usr/realtime/terstsuite/kern/latency
# ./run
```

RTAI programs run as kernel modules. If there is any indication of a problem, it may be that there was a problem with the module version – if the kernel suspects that the module was built for a different version of the kernel, it will refuse to run it. This can happen easily when reconfiguring kernels, because it is important to build the module with the same include files as were used for the kernel actually in use, including the .config file. If you change the configuration file and rebuild the kernel without rebuilding the RTAI modules, you can get this sort of problem.

Problems loading modules are usually logged in /var/log/messages, so 'tail' this file if you have any problems to see if you can see anything relevant there, If all goes well, you should see some latency figures that don't look very impressive until you realize they're in nanoseconds, not microseconds. Then you realise this really is a hard real-time kernel you're running.

4.3.11 Summary of Commands

Getting the Source Packages for Linux kernel and RTAI and unpacking

```
$ cd /usr/src
$ wget -c -P /usr/src
http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.xx.tar.bz2
$ wget -c -P /usr/src http://www.rtai.org/RTAI/rtai-3.8.1.tar.bz2
$ tar -xf linux-2.6.xx.tar.bz2
$ tar -xf rtai-3.8.1.tar.bz2
```

Applying RTAI patch to Linux Source

```
$ cd /usr/src/linux-2.6.xx/
$ patch –p1 < /usr/src/rtai-3.8.1/base/arch/i386/patches/hal-
linux-2.6.xx-i386-1.12-03.patch
```

Configuring the Linux Kernel

```
$ make menuconfig or
$ make xconfig or
$ make gconfig
```

Building the Linux kernel

```
$ make clean
$ make or make bzImage
$ make modules
```

Installing the Linux kernel

```
$ su
# make modules_install
# cp arch/i386/boot/bzImage /boot/bzImage-2.6.xx
# mkinitrd /boot/initrd-2.6.xx.img 2.6.xx
# cp System.map /boot/System.map-2.6.xx.map
# ln –s /boot/System.map-2.6.xx /boot/System.map
# vi /boot/grub/menu.lst
# exit
```

Configuring RTAI

```
$ cd /usr/src/rtai-3.8.1/
$ mkdir rtai-build
$ cd rtai-build
$ make –f /usr/src/rtai-3.8.1/makefile menuconfig
```

Building and Installing RTAI

```
$ make
$ su
# make install
```

Reboot

```
# reboot
```

Testing RTAI

```
$ su
# cd /usr/realtime/testsuite/kern/latency
# ./run
```

4.4 GMRT method of Linux Kernel and RTAI Installation

There is no much difference between the previous method of installation and present one. In this method all the commands are written in three scripts files. These files will take care of all the jobs done by us in the previous method. The pre-requisite for this method of installation is target system must be connected to either ***gmrt or ncra network.*** All the files required for installation can be downloaded from tech1.gmrt.ncra.tifr.res.in machine. Refer appendix A to configure network.
The following section describes the function of each script files job. Create the */root/install/* directory,
Goto

Menu ⟶ Utility ⟶ Rxvt Terminal Emulator

Change to */root/install* directory,

```
# mkdir /root/install
# cd /root/install
```

Download the script files from tech1.gmrt.ncra.tifr.res.in. script1.sh, script2.sh and script3.sh and change the file permissions to executable,

```
# wget http://tech1.gmrt.ncra.tifr.res.in/thiyagu/PC104/rtai-install/script1.sh
 <ENTER>
# wget http://tech1.gmrt.ncra.tifr.res.in/thiyagu/PC104/rtai-install/script2.sh
<ENTER>
# wget http://tech1.gmrt.ncra.tifr.res.in/thiyagu/PC104/rtai-install/script3.sh
<ENTER>
```

```
#chmod a+x script1.sh
#chmod a+x script2.sh
#chmod a+x script3.sh
```

script1.sh

Run the `script1.sh` file from terminal,

```
#./script1.sh
```

- Creates temporary directory in */root called install/* and downloads necessary files into it.
- Creates */usr/src* directory and downloads Linux kernel and RTAI source files into it.
- Uncompress the tar files into */usr/src* and instructs to run ***script2.sh*** file.

script2.sh

```
#./script2.sh
```

- Configures the Linux kernel with the default .config file in source tree, Opens text based console as shown in figure 4-1 to configure the kernel, all the required configuration are already done. Simply select < Exit > or press <ESC> <ESC> to exit.

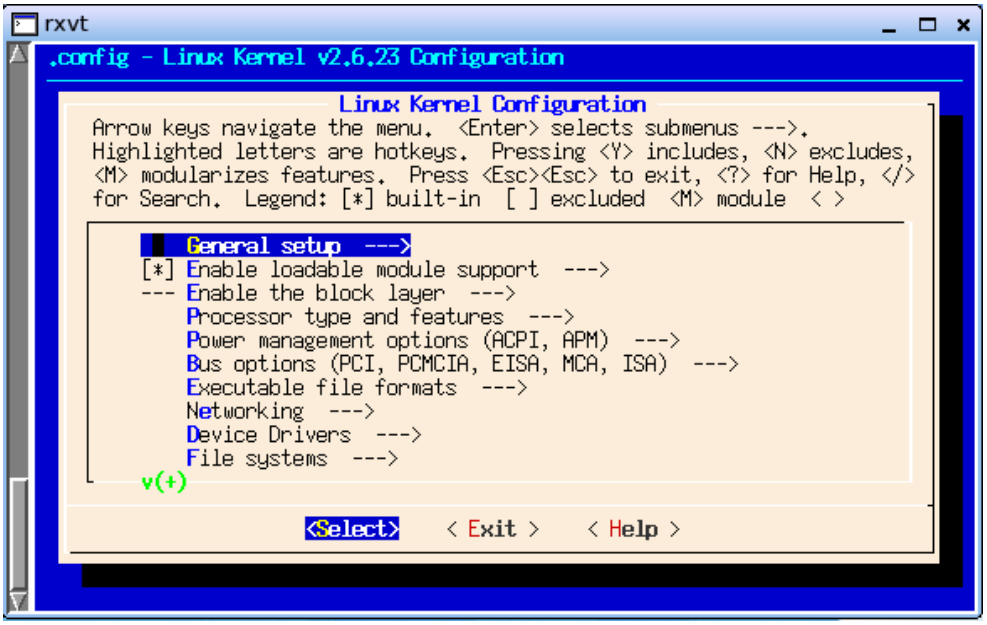


Figure 4- Linux Configuration Wizard

- Building the Linux kernel and installing it.
- Updating menu.lst
- Instructs to run the script3.sh

script3.sh

```
#./script3.sh
```

- Configures the RTAI
- Building and installing RTAI kernel modules.
- Reboots the System and instructs to run test suite programs.

4.5 Testing RTAI Installation

RTAI comes with a number of test programs. One of these runs a latency test, rescheduling itself at 10 KHz rate and measuring the difference for each reschedule between the actual and expected rescheduling times. If you didn't change the default in the RTAI configuration, most of RTAI is installed in */usr/realtime*, and to run the latency test you need to become root (puppy runs as root all time) and execute the 'run' command for the test

4.5.1. Latency Test

```
# cd /usr/realtime/testsuite/kern/latency
# ./run
```

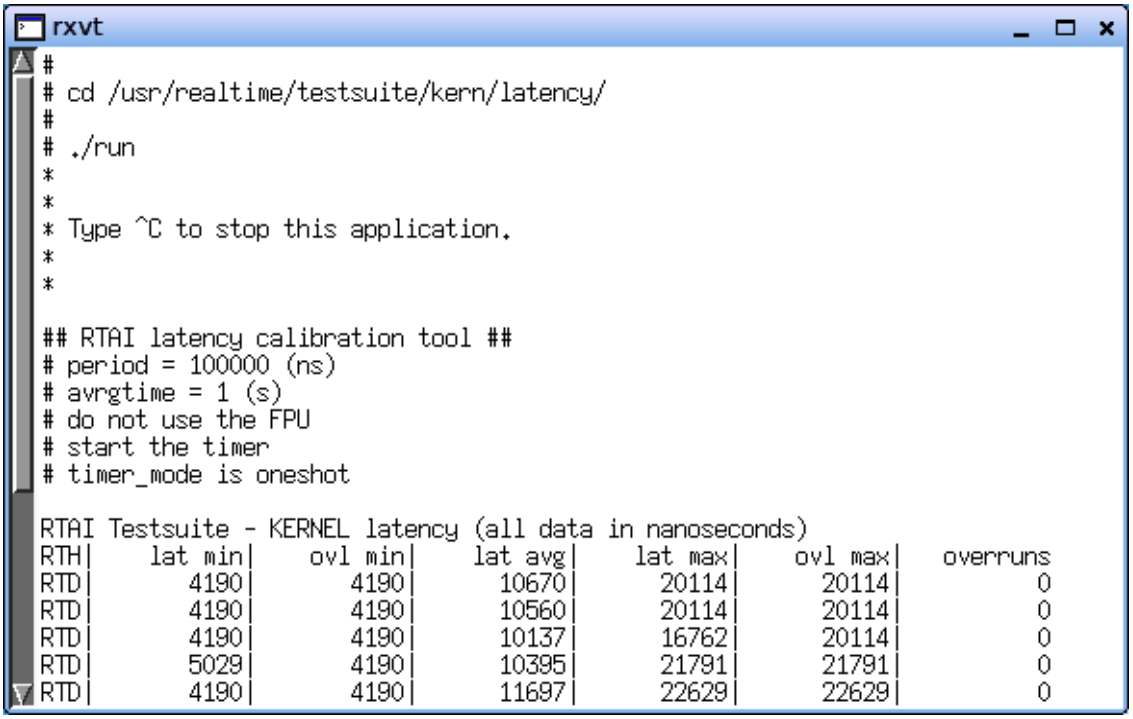


Figure 4- RTAI Latency Test

4.5.2. Preemption Test

```
# cd /usr/realtime/testsuite/kern/preempt
# ./run
```

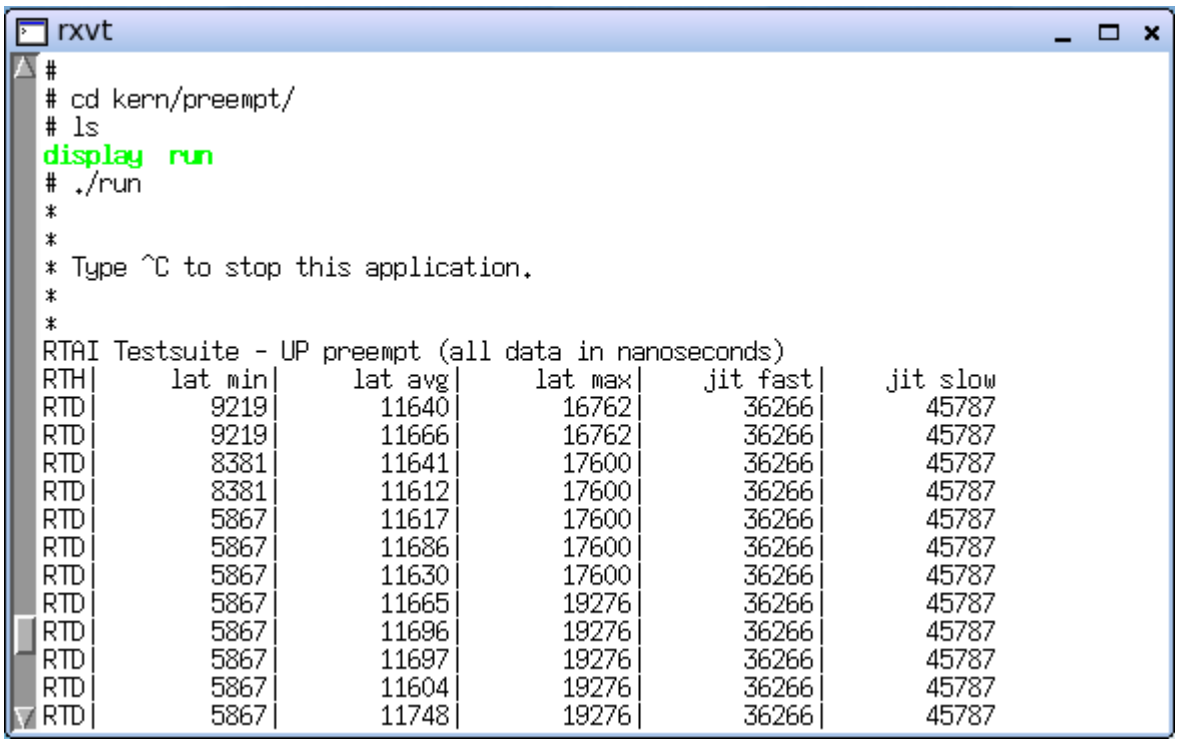


Figure 4- RTAI Preemption Test

4.5.3. Switches Test

```
# cd /usr/realtime/testsuite/kern/switches
# ./run
```

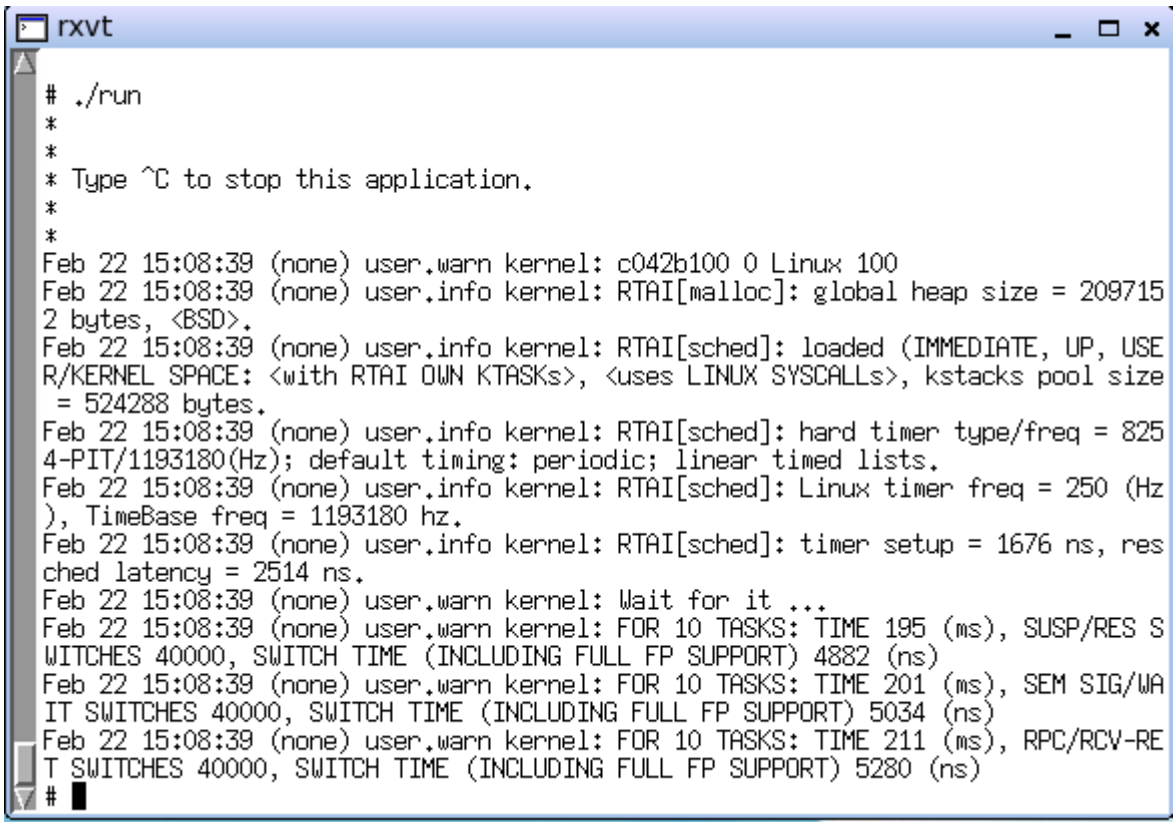


Figure 4- RTAI Switches Test

RTAI programs run as kernel modules. If there is any indication of a problem, it may be that there was a problem with the module version – if the kernel suspects that the module was built for a different version of the kernel, it will refuse to run it. This can happen easily when reconfiguring kernels, because it is important to build the module with the same include files as were used for the kernel actually in use, including the .config file. If you change the configuration file and rebuild the kernel without rebuilding the RTAI modules, you can get this sort of problem. Problems loading modules are usually logged in /var/log/messages, so ‘tail’ this file if you have any problems to see if you can see anything relevant there.If all goes well, you should see some latency figures that don’t look very impressive until you realize they’re in nanoseconds, not microseconds. Fig shows the latency response in VDX6354D embedded boards.

5. PET Package Installations

PET Packages are additional software applications that can run on Puppy Linux to add extra facilities to it. These packages are not part of standard puppy release, in order to suit our applications we need to install some packages, the list of additional packages to be installed on puppy Linux are listed in table 5-1,

Sr.No	PET Package Name	Description
1	gnuplot-4.2.5-i486.pet	Plotting utility
2	openssh-client-5.1p1.pet	Secure client component for remote communication
3	openssh-server-5.1p1.pet	Secure server component for remote communication
4	qt-3.3.8.pet	Provides the qt libraries for gui
5	tightvnc_viewer-1.3.9.pet	Client component for remotely view & interact with X displays
6	x11vnc_server-0.9.4B.pet	Server component for remotely view & interact with X displays
7	enscript-1.6.6-i486.pet	ASCII to PS conversion
8	pdftk-1.41.pet	PDF file merge utility

Table 5- PET Package List

The installation steps are given below,

Open the terminal,

Menu → Utility → Rxvt terminal emulator

Download the installation scripts from *tech1.gmrt.ncra.tifr.res.in*.

```
# wget http://tech1.gmrt.ncra.tifr.res.in/thiyagu/PC104/rtai-install/pet-install.sh <ENTER>
```

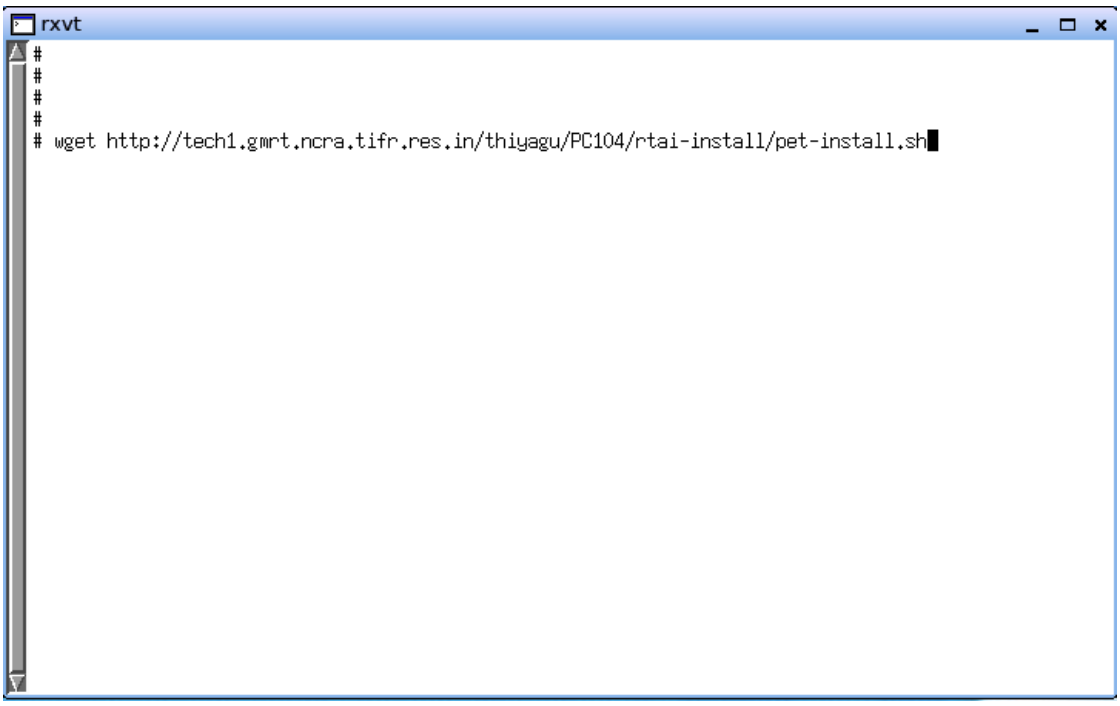


Figure 5- PET Download terminal

Change file-permission of `pet-install.sh`

```
#chmod a+x pet-install.sh
```

Run the pet-install.sh script file from terminal. It downloads all PET files from tech1.gmrt.ncra.tifr.res.in and asks conformation to install packages.

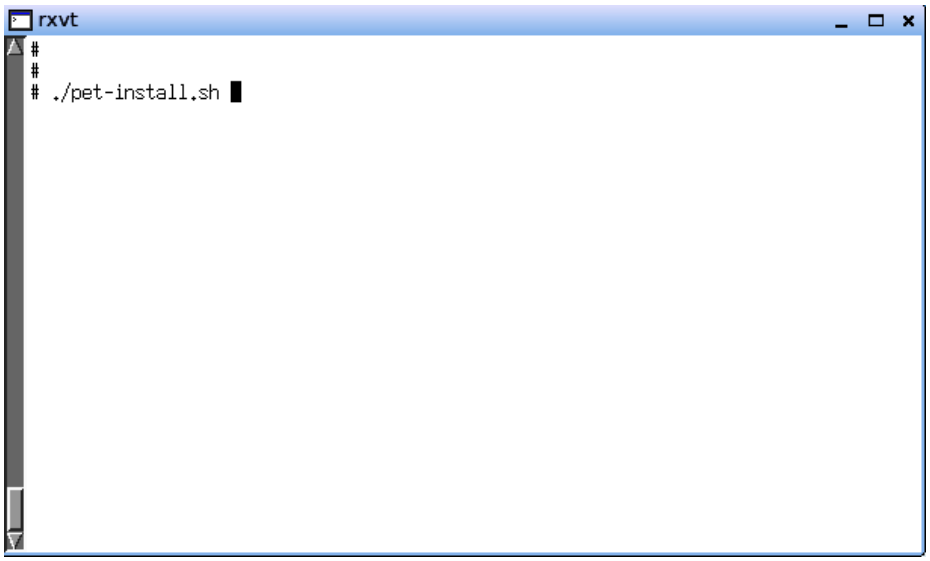


Figure 5- PET Install Script

GNU Plot:

GNU Plot installation conformation message as shown in fig 5-3,

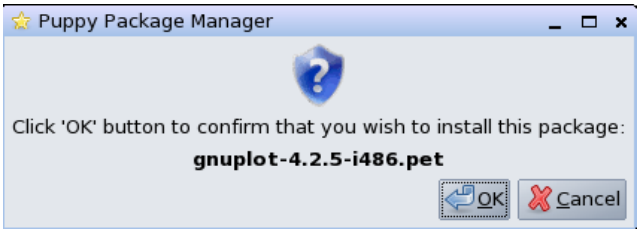


Figure 5- GNUPlot Install Conformation

Click < ok > to install gnuplot application. Finally puppy reports successes message as shown in fig 5-4. and updates help page.

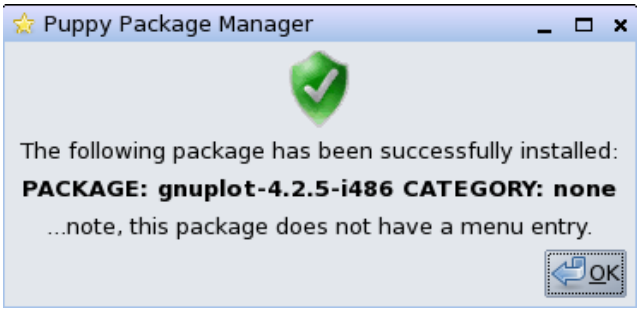


Figure 5- GNUPlot Installtion Successes

x11vnc_server:

Conformation message to install x11vnc_server as shown in fig 5-5,

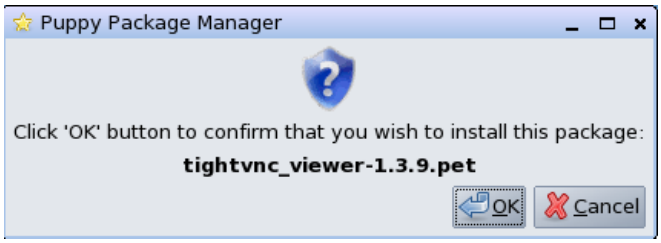


Figure 5- x11vnc server Installation Conformation

Click < ok > to install x11vnc_server application. Puppy reports success message, update its entry into “Network” and finally updates help page as shown in fig 5-6.

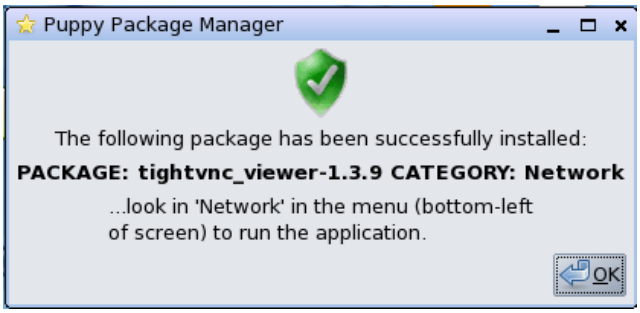


Figure 5- x11vnc server Installation Successes

tightvnc_viewer:

Conformation message to install tightvnc_viewer as shown in fig 5-7,

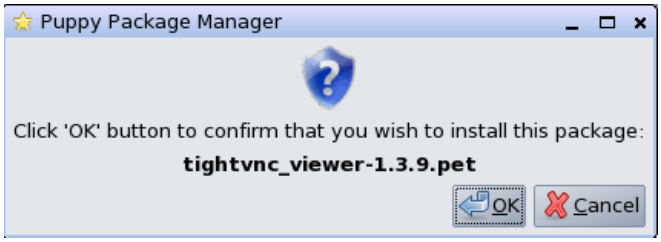


Figure 5- TightVNC viewer Installation Conformation

Click < ok > to install tightvnc_viewer application. Puppy reports success message, update its entry into “Network” and finally updates help page as shown in fig 5-8.

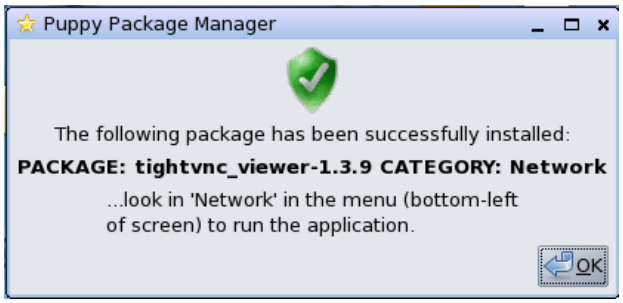


Figure 5- TightVNC Installation Successes

openssh_server:

Conformation message to install openssh_server as shown in fig 5-9,

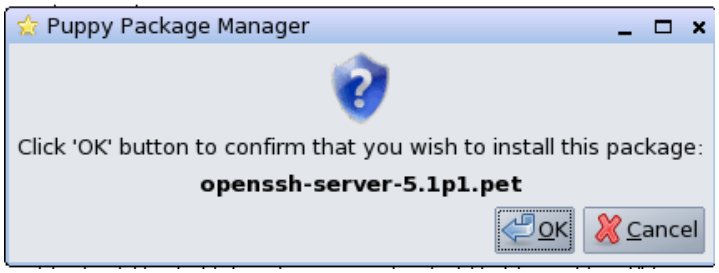


Figure 5- open-ssh server Installation Wizard

Click < ok > to install openssh_server application. Puppy reports success message and finally updates help page as shown in fig 5-9.

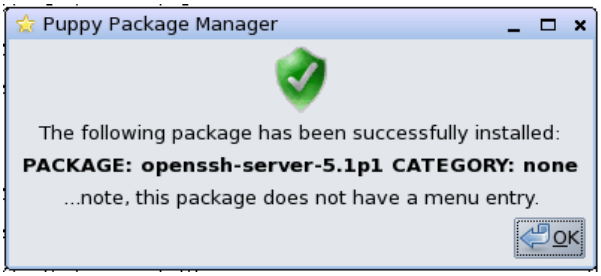


Figure 5- open-ssh server Installation Successes

openssh-client:

Conformation message to install openssh-client as shown in fig 5-10,

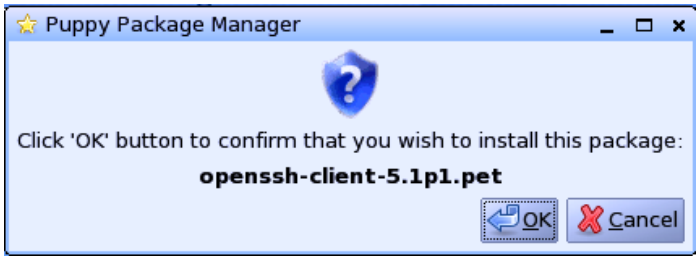


Figure 5- open-ssh client Installation Conformation

Click < ok > to install openssh-client application. Puppy reports success message and finally updates help page as shown in fig 5-12.

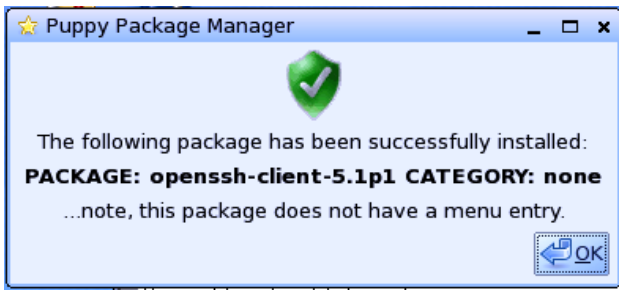


Figure 5- open-ssh client Installation Successes

Qt:

Conformation message to install qt as shown in fig 5-13,

Figure 5- Qt Installation Conformation

Click < ok > to install qt application. Puppy reports success message and finally updates help page as shown in fig 5-14.

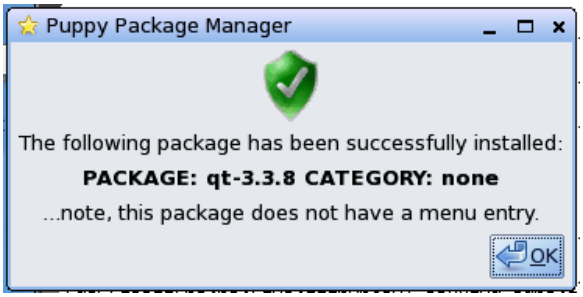


Figure 5- Qt Installation Successes

enscript:

Conformation message to install enscript as shown in fig 5-15,

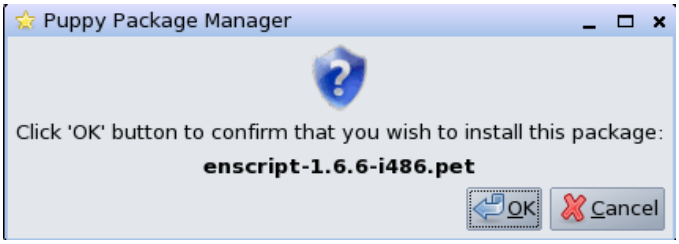


Figure 5- enscript Installation Conformation

Click < ok > to install enscript application. Puppy reports success message and finally updates help page as shown in fig 5-16.

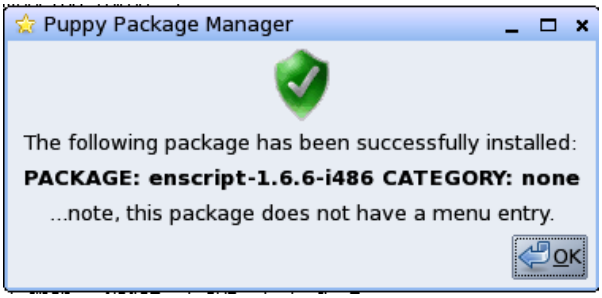


Figure 5- enscript Installation Successes

pdftk:

Conformation message to install enscript as shown in fig 5-17,

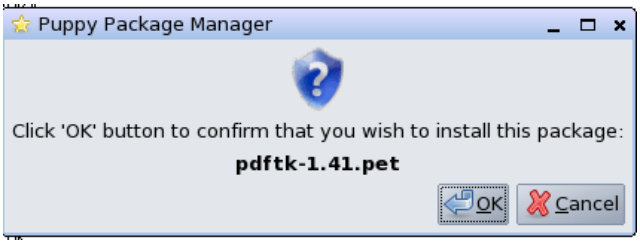


Figure 5- Pdftk Installation Conformation

Click < ok > to install enscript application. Puppy reports success message and finally updates help page as shown in fig 5-18.

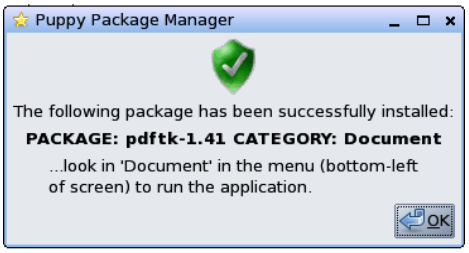


Figure 5- Pdftk Installation Successes

Starting x11vnc-server:

PET installation scripts installs all pet packages listed above, at end it configures x11vnc-server to listen remote request. The fig 5-19 will ask you to select the type of service

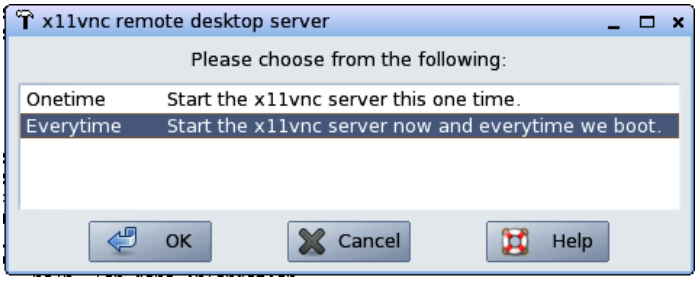


Figure 5- x11vnc server Configuration wizard 1

Select "Everytime Start the x11vnc server now and everytime we boot" and click <OK>.

Next puppy will request you to give the password to accept the remote request as shown fig 5-20,

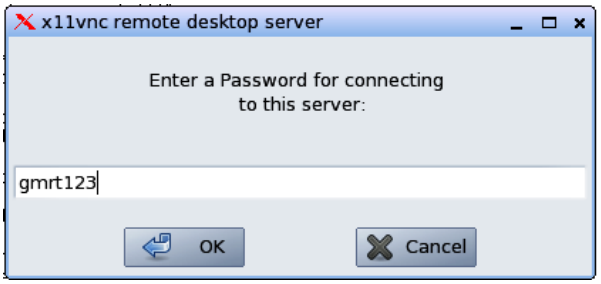


Figure 5- x11vnc server Configuration wizard 2

Finally puppy starts x11vncserver with it server address as shown in fig 5-21,

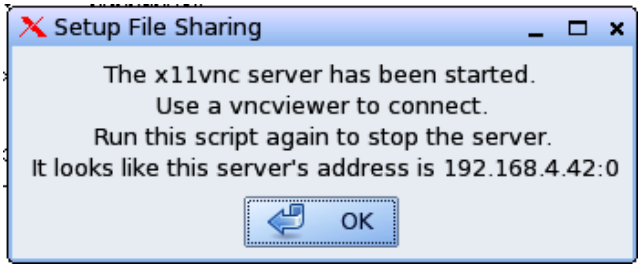


Figure 5- x11vnc server Configuration wizard 3

Appendix.A Network Configuration in Puppy Linux

1. Ensure PC104 System is connected to Network.

2. Open terminal

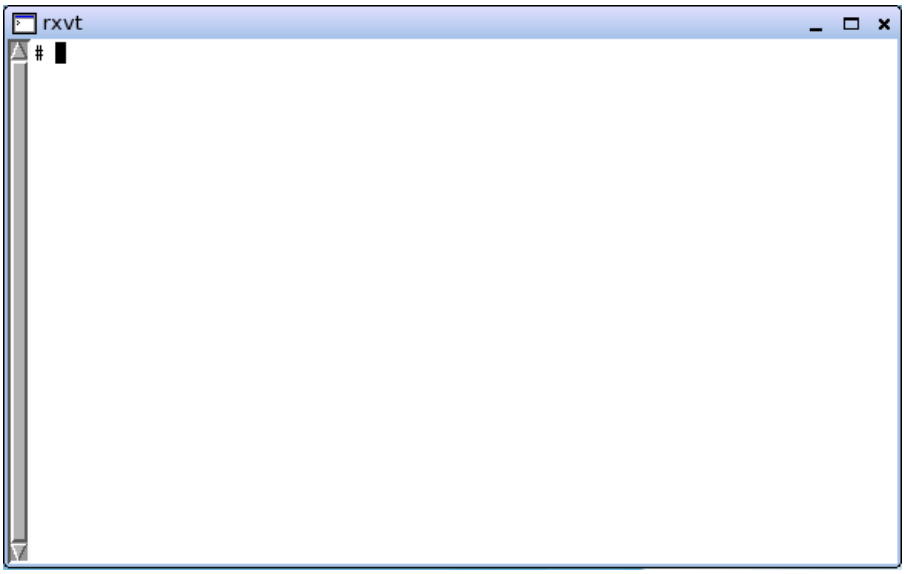


Fig A-1 rxvt terminal

3. Check eth0 interface whether it is up or down?

```
# ifconfig
```

The above command list only following shown in fig A-2 then ethernet interface is DOWN.

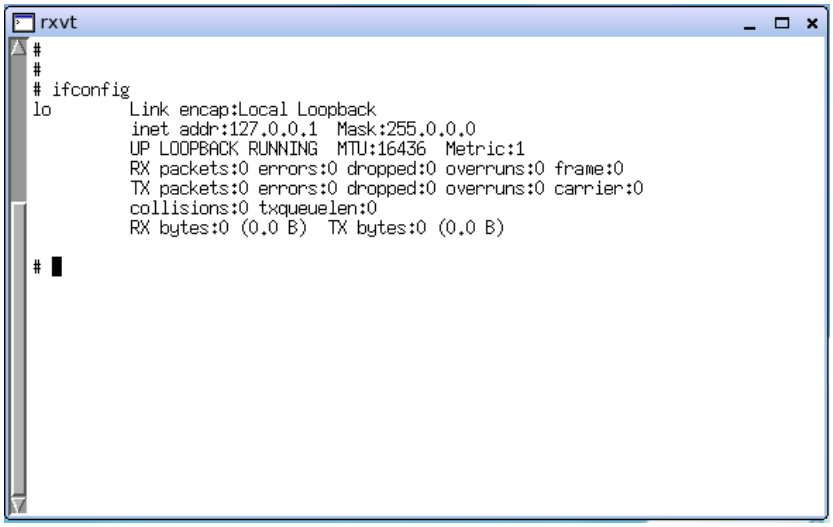


Fig A-2 Loop Back interface only Enabled

On contrary it shows eth0 interface then eth0 is UP as shown in fig A-3

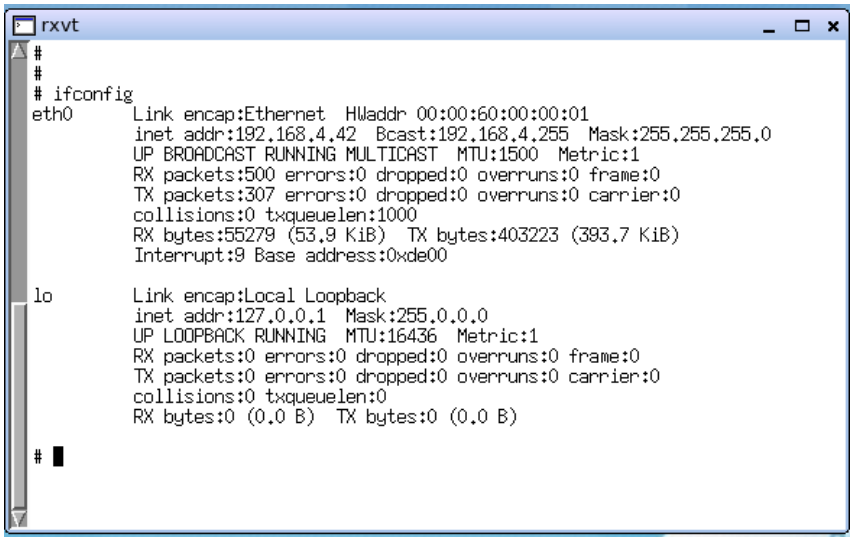


Fig A-3 eth0 interface and loopback both enabled

4. If it is “UP” goto setp 6 else goto next setp

5. Enable eth0 interface

   Issue command `# ifconfig eth0 up` to enable eth0 interface, and issue `# ifconfig` to verify it.

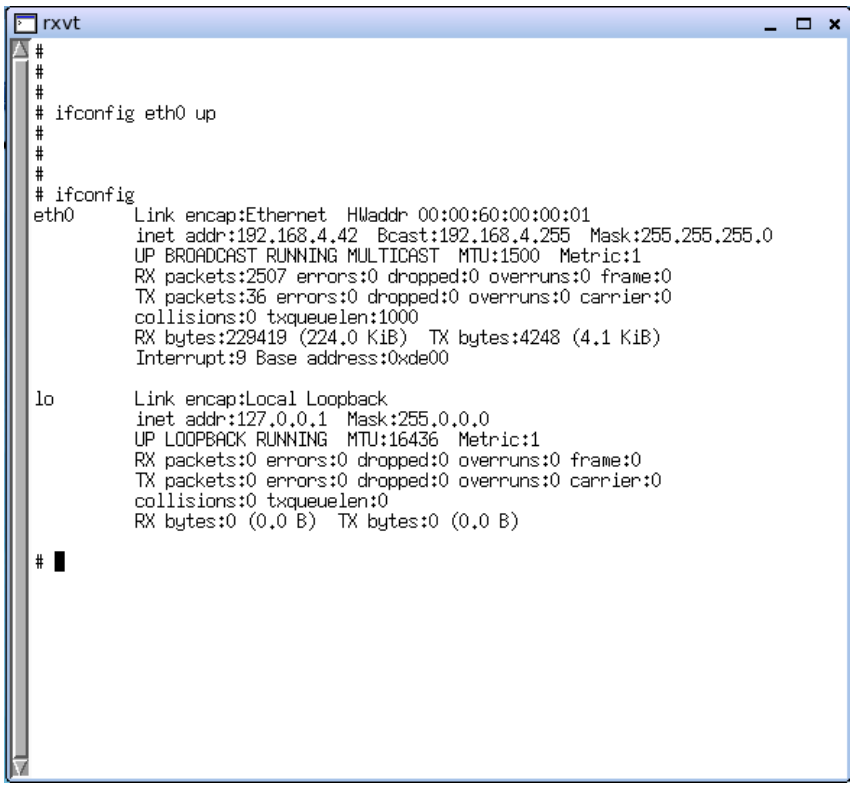


Fig A-4 Enabling eth0 interface

6. Type following command in terminal

   `# net-setup.sh <Enter>`

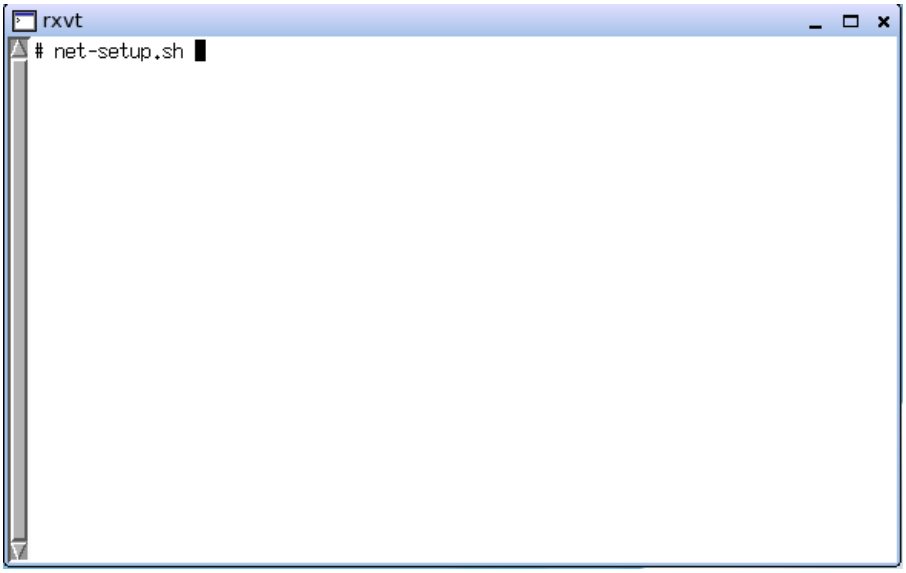


Fig A-5 Configuring network

The above command will open “Puppy Network Wizard” as shown in fig A-6 Puppy Network Wizard,

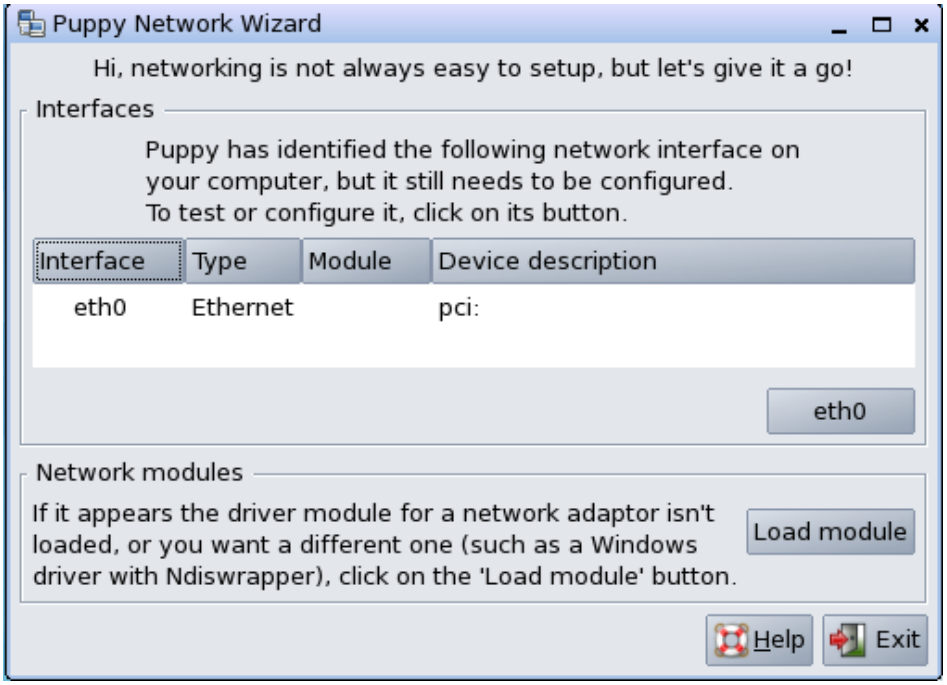
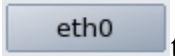


Fig A-6 Puppy Network Wizard

Select the eth0 and Click 

the button. “Configure network interface eth0” window will open as shown fig A-7.

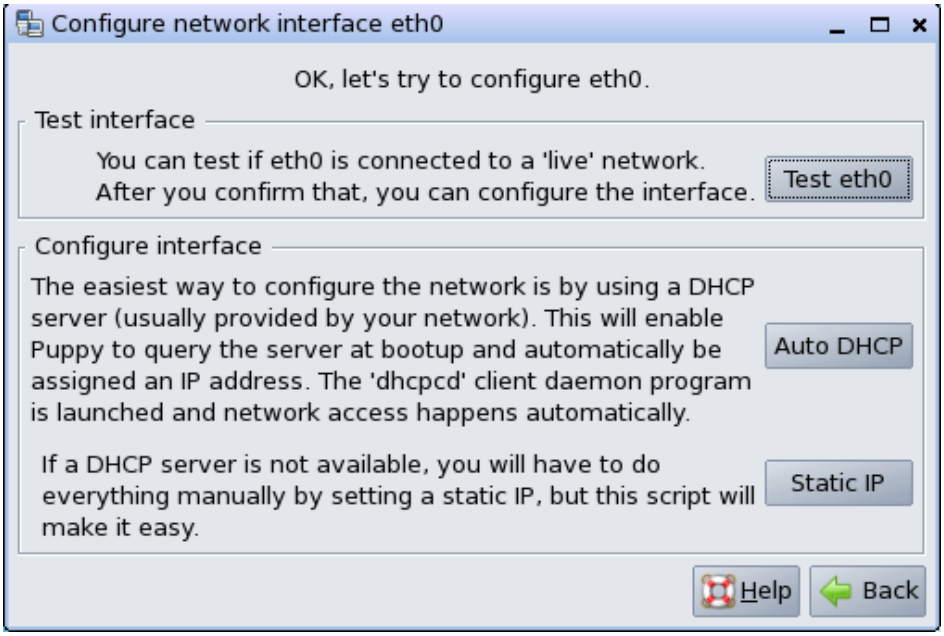


Fig A-7 Configure network interface eth0 1

Click the  button. “Set Static IP” window will open as shown in fig A-8.

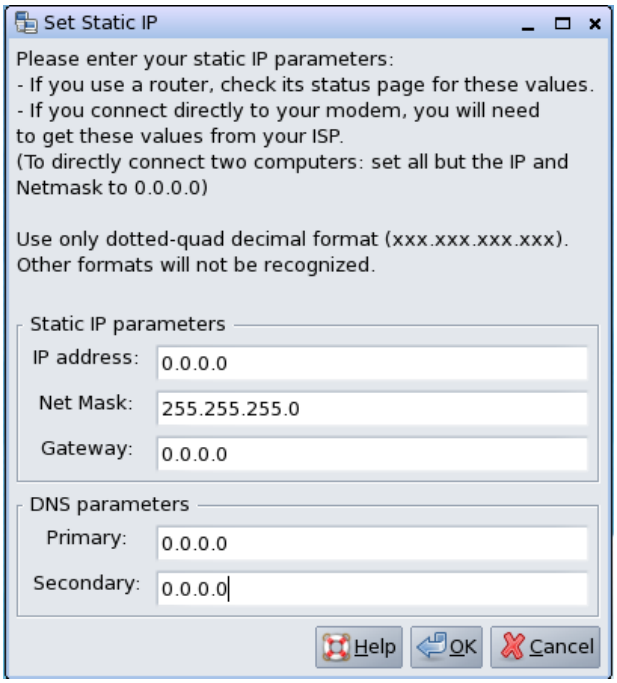


Fig A-7 Set Static IP

Input the following fields as per below in the “Set Static IP window”.

Static IP parameters

IP Address: 192.168.8.42

Net Mask: 255.255.255.0

Gate Way: 192.168.8.1

DNS parameters

Primary: 158.144.18.14

Secondary: 158.144.18.17

Click OK Button.

7. Puppy Network Wizard Static IP conformation window will open. Accept the default as shown in fig A-8

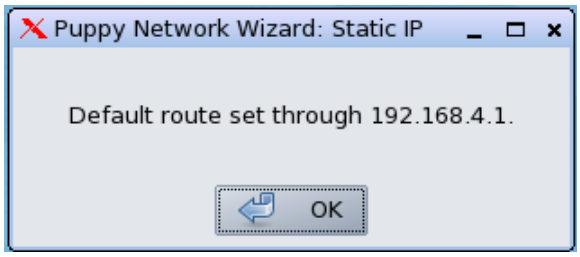


Fig A-8 Static IP Conformation

8. Puppy will ask you to save the present configuration for next boot. Click YES in the Xdialog window as shown in fig A-9.

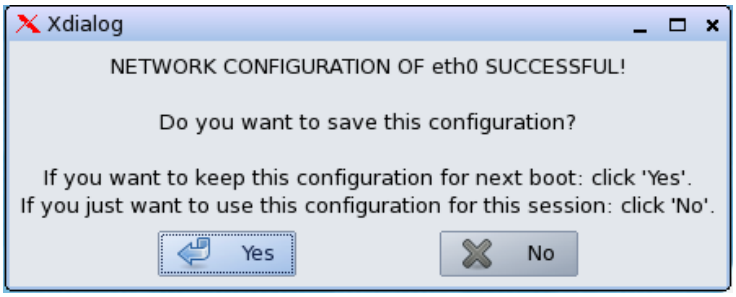


Fig A-9 Network Configuration xdialog

9. Puppy “Configure network interface eth0” window will open with “NETWORK CONFIGURATION OF eth0 SUCESSFUL!” as shown on fig A-10.

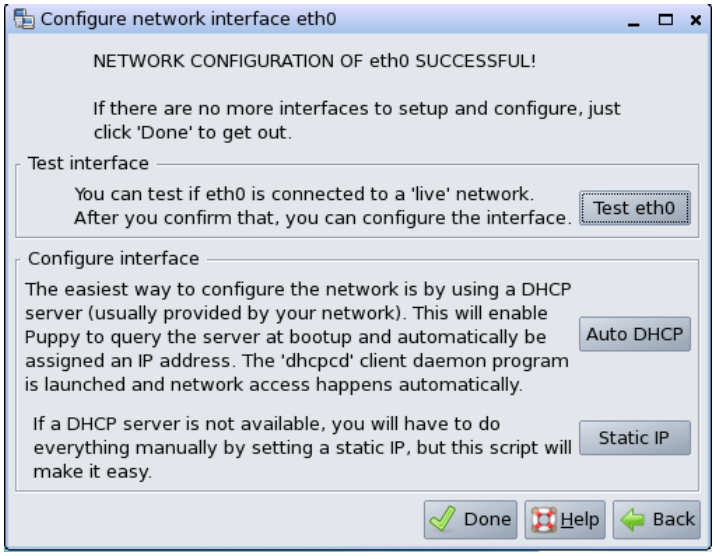


Fig A-10 Configure network interface eth0 2

10. Click the 

push button in Fig A-10. Puppy will check the connection with live network. If live network exist (Puppy was able to find a live network) message was reported in “Configure network interface eth0” as shown in Fig A-11.

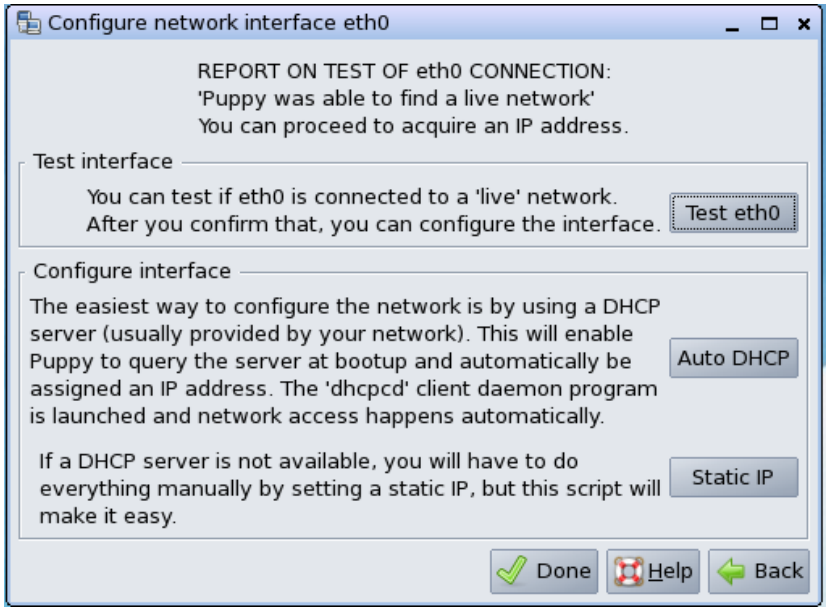


Fig A-11 Configure network interface eth0 3

11. Click the 

Push button to exit the wizard. In the Desktop task bar you can see the blinky which shows the connection of local computer with network.

Appendix.B Copying files from PC104 to USB Pendrive

1. Connect USB Pendrive to PC104.

2. Wait for a second it detected and displayed on Desktop

3. Open terminal (Menu -> Utility -> Rxvt terminal emulator) and issue following command

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

4. /dev/sda1 may vary depends on system, ideal case most PC104 system will detect first external drive as sda1. If it is detected in another name create directory under /mnt in the respective name and mount it.

Example:

Locate the file location in our case is (/root/eventlog) to copy this file into pendrive, issue following command,

```
# cp /root/eventlog /mnt/sda1
```

To unmount the device give following command,

```
# cd
# umount /dev/sda1
```

Appendix.C Script Files

Script File 1

```
#!/bin/bash
#
#  Version 1.0
#  Downloading Linux Kernal, RTAI and necessary files
#


echo "Getting the Linux kernel source  and RTAI Source files"
mkdir /usr/src/

URLS="http://tech1.gmrt.ncra.tifr.res.in/thiyagu/PC104/rtai-
install/linux-2.6.23_rtai-3.8.1.tar.gz \
http://tech1.gmrt.ncra.tifr.res.in/thiyagu/PC104/rtai-
install/rtai-3.8.1.tar.bz2 \
http://tech1.gmrt.ncra.tifr.res.in/thiyagu/PC104/rtai-
install/Config-2.6.23_rtai-3.8.1-ver1.1 \
http://tech1.gmrt.ncra.tifr.res.in/thiyagu/PC104/rtai-
install/menu.lst"

for u in $URLS
do
 wget -P /usr/src $u
 rc=$?
 if (($rc==0)); then
  echo -en "\033[33;32m!! $u Download Successes !!"
  else
   echo -en "\033[33;31m!! $u Download Failed !!"
 fi
 echo -en "\033[0m"
done

cd /usr/src
echo -en "Extracting linux-2.6.23_rtai-3.8.1.tar.gz"
tar -xf linux-2.6.23_rtai-3.8.1.tar.gz
echo -en "Extracting rtai-3.8.1.tar.bz2"
tar -xf rtai-3.8.1.tar.bz2

rm -rf linux-2.6.23_rtai-3.8.1.tar.gz
rm -rf rtai-3.8.1.tar.bz2
```

```
#!/bin/bash
#
#  Version 1.0
#
#  Linux ver 2.6.23 with r6040 and rtai-3.8.1 patch applied.
#  Configuring and Installing Linux Kernal
#

echo "---------------------------------"
echo " Configuring Linux Kernel... "
echo "---------------------------------"$'\n'

cd /usr/src/linux-2.6.23_rtai-3.8.1
make clean
mv /usr/src/Config-2.6.23_rtai-3.8.1-ver1.1 .config
make menuconfig
echo "-----------------------------"
echo "Building the Linux Kernel..."
echo "-----------------------------"$'\n'
make && make modules && make modules_install
echo "----------------------------"
echo "Installing the Linux Kernel"
echo "----------------------------"$'\n'
cp arch/i386/boot/bzImage /boot/bzImage-2.6.23_rtai-3.8.1
chmod a+x /boot/bzImage-2.6.23_rtai-3.8.1
cp System.map /boot/System.map-2.6.23_rtai-3.8.1
ln -s /boot/System.map-2.6.23_rta-3.8.1 /boot/System.map
echo "----------------------------"
echo "Updating the GRUB menu.lst"
echo "----------------------------"$'\n'
mv /usr/src/menu.lst /boot/grub/menu.lst

echo -en "\033[32m !!! Linux kernel ready to use !!!"$'\n'
echo -en "!!! Reboot and boot with Compiled Kernel !!!"$'\n\n'
echo -en "\033[33;31m Don't FORGET TO CHANGE BIOS SETTINGS"$'\n\n'
echo -en "\033[0m"$'\n'
```

```
#!/bin/bash
#
#  Version 1.0
#  Configuring, Building and Installing RTAI
#

echo "-------------------------------"
echo "       Configuring RTAI        "
echo "-------------------------------"
cd /usr/src/rtai-3.8.1/rtai-build
echo "-------------------------------"
echo "     Building RTAI Modules     "
echo "-------------------------------"
make

echo "-------------------------------"
echo "   Installing RTAI Modules     "
echo "-------------------------------"

make install

echo  " "
echo -en "\033[32m !!! RTAI INSTALLATION SUCESSFUL !!! "$'\n'
echo -en " REBOOT AND RUN RTAI TESTSUITE PROGRAM "$'\n'
echo -en "\033[0m"$'\n'
```

```
#!/bin/bash
#
# Pet Download and Installation Scripts
# Version 1.0
#

SHELL=/bin/bash
PATH=/bin:/usr/bin:/sbin:/usr/sbin:/usr/local/bin:/usr/X11R7/bin:/root/m
y-
applications/bin:/opt/qt4/bin:/opt/mozilla.org/bin:/opt/samba/bin:/usr/l
ib/qt/bin:

wget http://tech1.gmrt.ncra.tifr.res.in/thiyagu/PC104/rtai-install/xwin
mv /root/xwin /usr/bin/xwin
chmod 755 /usr/bin/xwin
ln -s /usr/bin/xwin /usr/bin/startx
wget http://tech1.gmrt.ncra.tifr.res.in/thiyagu/PC104/rtai-
install/servoboot.sh
chmod a+x /root/servoboot.sh
wget http://tech1.gmrt.ncra.tifr.res.in/thiyagu/PC104/rtai-
install/rc.local
mv /root/rc.local /etc/rc.d/rc.local
chmod a+x /etc/rc.d/rc.local
wget http://tech1.gmrt.ncra.tifr.res.in/thiyagu/PC104/rtai-
install/startup.sh
mv /root/startup.sh /root/Startup/
chmod a+x /root/Startup/startup.sh
wget http://tech1.gmrt.ncra.tifr.res.in/thiyagu/PC104/rtai-
install/desktop_shortcut
wget http://tech1.gmrt.ncra.tifr.res.in/thiyagu/PC104/rtai-install/plot-
2.0.tar.bz2
wget http://tech1.gmrt.ncra.tifr.res.in/thiyagu/PC104/rtai-
install/epstopdf
wget http://tech1.gmrt.ncra.tifr.res.in/thiyagu/PC104/rtai-install/open-
rtc
tar -xf /root/plot-2.0.tar.bz2
wget http://tech1.gmrt.ncra.tifr.res.in/thiyagu/PC104/Source-
Code/servoProgram/Stable/servoSoftware2.6.5.tar.bz2
tar -xf servoSoftware2.6.5.tar.bz2

mv /root/epstopdf /usr/bin/
chmod a+x /usr/bin/epstopdf
mv /root/plot-2.0/ /root/plot/
chmod a+x /root/plot/*
rm -rf plot-2.0.tar.bz2 servoSoftware2.6.5.tar.bz2
mkdir /mnt/servo_disk
mkdir /mnt/rtai
mkdir /root/log
cd /tmp/

echo "------------------------------------------------------------------"
echo "Getting pet-download list from tech1.gmrt.ncra.tifr.res.in..."
echo "------------------------------------------------------------------
"$'\n'

echo "Changing to /tmp/ directory..."$'\n'
```

```
cd /tmp
echo $PWD
wget -c -b http://tech1.gmrt.ncra.tifr.res.in/thiyagu/PC104/rtai-
install/pet-download.txt
chmod a+x /tmp/pet-download.txt

#if [ ! -f /tmp/pet-download.txt ]; then
#     echo -en "\033[33;32m /tmp/pet-download.txt does not EXIST..."$'\n'
#     echo -en "\033[0m"
#     exit

#else
      echo -en "\033[33;32m Downloading pet packages..."$'\n'
      echo -en "\033[0m"
      wget -c -P /tmp/install -i /tmp/pet-download.txt
      echo -n "Installing pet packagese..."
      echo " "
      petget +/tmp/install/gnuplot-4.2.5-i486.pet
      petget +/tmp/install/x11vnc_server-0.9.4B.pet
      petget +/tmp/install/tightvnc_viewer-1.3.9.pet
      petget +/tmp/install/openssh-server-5.1p1.pet
      petget +/tmp/install/openssh-client-5.1p1.pet
      petget +/tmp/install/qt-3.3.8.pet
      petget +/tmp/install/enscript-1.6.6-i486.pet
      petget +/tmp/install/pdftk-1.41.pet

      echo -n "Creating symbolic links..."
      echo " "
      ln -s /usr/lib/qt-3.3.8/lib/libqt-mt.so.3.3.8 /usr/lib/libqt-
mt.so.3

      echo -n "Configuring x11 VNC Server..."
      echo " "
      x11vnc-gui

      rm -rf /tmp/install

#         cd /root/servoSoftware2.6.5/
#         make
       rm /tmp/pet-download.txt
fi
```