Advices for Linux on PC

Software Documentation

Version 1.1
(2016-09-15)

Untere Waldplätze 23
D-70569 Stuttgart
Germany

Phone: +49(0)711-123722-0
Fax: +49(0)711-123722-99

About This Document

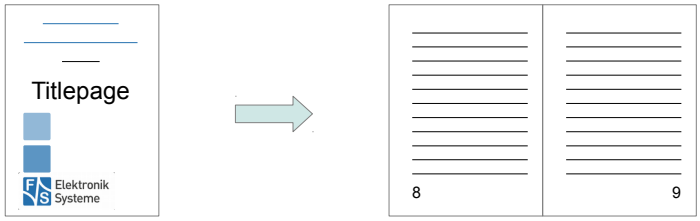
This document gives some advices and hints of how to install and configure the Linux distribution on your PC when working with F&S boards and modules under Linux. As this concerns the PC, it is valid for all types of boards, no matter if it is NetDCU, PicoMOD, PicoCOM, efus, armStone, QBliss or any other board type by F&S.

Remark

The version number on the title page of this document is the version of the document. It is not related to any Linux software release version. The latest version of this document can always be found at http://www.fs-net.de.

How To Print This Document

This document is designed to be printed double-sided (front and back) on A4 paper. If you want to read it with a PDF reader program, you should use a two-page layout where the title page is an extra single page. The settings are correct if the page numbers are at the outside of the pages, even pages on the left and odd pages on the right side. If it is reversed, then the title page is handled wrongly and is part of the first double-page instead of a single page.



Typographical Conventions

We use different fonts and highlighting to emphasize the context of special terms:

`File names`

Menu entries

`Board input/output`

`Program code`

`PC input/output`

`Listings`

`Variables`

History

Date	V	Platform	A,M,R	Chapter	Description	Au
2011-07-13	0.1	Linux-PC	A	-	First basic version	HK
2011-07-20	0.2		A, M	3.8	Add more ports for NFS and working with BuildRoot image	HK
2014-12-01	1.0		M	All	Completely rewritten; support for Fedora 20; new F&S layout	HK
2016-09-15	1.0		A,M	3, 4.1, 1.1	Add hint to editor, add needed host packages, add hints to fedora version and gcc	

V	Version
A,M,R	Added, Modified, Removed
Au	Author: CZ, DK, HF, HK, MK

Table of Contents

1 Introduction

When working with an F&S board like the NetDCU, PicoMOD, PicoCOM, armStone or efus under Linux, you usually need to install and/or configure the Linux on your PC, too. For example you need to activate TFTP to allow file downloads to the module. Or NFS to allow mounting of the root filesystem over the network during development. However this often depends on the type of Linux distribution you use. Therefore explaining these things in the context of the normal board documentation would distract too much. This is why we made this separate documentation where we collect some hints and explain how they can be applied depending on the Linux distribution.

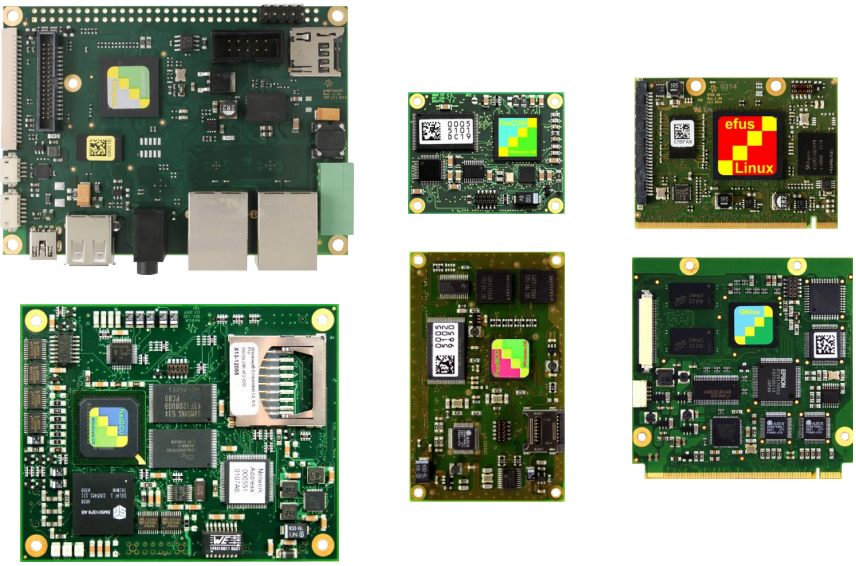


Figure 1: F&S boards and modules

In this documentation, we will show dialogs, commands and results in English, to be as international as possible. But for you it is perfectly OK to switch Linux to your local language. Of course some names may slightly differ then, for example of menu entries or dialog buttons. But usually it is easy to derive the correct name in your language from the context.

We also have made the experience that some localizations are not fully complete. So some programs may show a strange mix of English and local entries and some manual pages may only show the full set of options in the English version. Then try using

```
LC=C man <command>
```

to get the English man page.

One of the main concepts of Linux is to always have a command line program to do some task. This command line program does support all options and all features that are available. Then in addition there may exist a graphical version of this program. This graphical version usually just adds the graphical front-end. The actual work is still done by calling the command line program in the background. We try to give both variants, the command line variant and the graphical variant, wherever possible. However especially when configuring the system, this differs largely between the distributions and maybe even between different desktop versions, for example KDE or Gnome. This is why we have different subsections in each

chapter, explaining things for each variant separately. Just look for those subsections that match your personal installation and skip the others.

1.1 Choosing The Distribution

We at F&S are using the Fedora Linux distribution (V21). There is no deeper meaning behind this, Fedora is just a modern distribution, with regular updates and up-to-date software packages like many other distributions, too. We are simply used to it. And because we are using Fedora, we usually have a description for Fedora here in this text. For other distributions, we depend on reports from our users and customers. We would be very happy if you'd share your findings with us by sending us a short e-mail with the explanation how things work in your distribution. Then we can add it to this document here. At the moment there is not much more available than Fedora alone. If you want to use a newer version of Fedora it is important that the version of the gcc that comes with Fedora matches the gcc version that buildroot uses. Currently Buildroot uses gcc 5 so it is ok to use Fedora up to version 23. Fedora 24 comes with gcc 6 and should therefore not be used.

Of course if you prefer some other distribution like Ubuntu, OpenSuse, Debian, CentOS or similar, you are completely free to do this. But then we can not give similar good advice. Therefore if you are new to Linux, we recommend using the Fedora distribution, because there we can help best.

Figure 2: Different Linux distributions

Linux is developed continuously. This means new versions of the programs that build the Linux system get available on a daily basis. Some Linux distributions have decided only to make such new versions available with the next major release of the distribution after they have been tested thoroughly and are stable. Then the updates only consist of a few security fixes. Fedora on the other hand does make new program versions available at any time and very soon after they are introduced. Only very large development steps are reserved for the next distribution release. This has the advantage that you'll always have a rather up-to-date distribution if you apply all updates. But it also means that updates are available rather frequently.

At the moment it does not matter if you install a 32-bit or 64-bit version of the Linux distribution in question. Our current toolchains are still based on a 32-bit version, so you might need some additional packages for 32-bit support on your PC if you have installed the 64-bit version. But it should work nonetheless. 32Bit libraries end on .i686, 64 bit libraries end on .x86_64. If you want to install a 32bit library on a 64 bit system you have to type the ending .i686 (e.g. yum install zlib-static.i686). If you type the file without ending the 64 bit variant will be installed by default (yum install zlib-static). However we don't know if it makes sense to still provide a 32-bit based toolchain in the future, when nearly everybody uses 64-bit systems. So maybe using the 64-bit version might prove to be better suited for the future.

All Linux distributions have a large repository of free and open software. A full installation would occupy several gigabytes on your hard disk. This is why all distributions only install a small amount of this software on your PC by default. When installing, they usually ask for the purpose you want to use this computer for and then install the appropriate set of software packages. Here we recommend installing as a “Development Computer”, because then most of the development tools required to work with the F&S boards are already installed automatically, for example the compiler collection GCC and the development packages of the libraries.

However it may be that a few required packages are still missing from your system. Then you can install these packages with your distribution specific software installation program. For example the TFTP service is usually not installed, even when using the Development Computer variant. Then you have to install it manually.

1.2 Choosing The Desktop Version

When looking at the Graphical User Interface (GUI), there is no such thing as “the” Linux desktop. Linux supports many different GUIs: Gnome, KDE, Unity, MATE, Cinnamon, XFCE, LXDE, Enlightenment and even more. Some of them are full-featured graphical systems with lots of animations, 3D-effects and other gimmicks, some are slim and unspectacular. For people coming from a Microsoft Windows environment, where there is only one desktop environment, this is rather confusing. So in addition to choosing the right Linux distribution, there is also the task to choose the right desktop system.

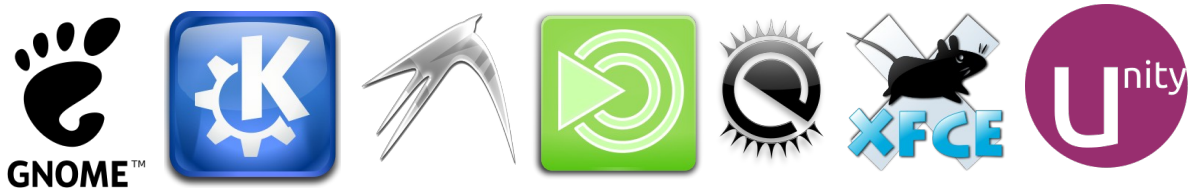


Figure 3: Different Linux desktop systems

You can not generally say that one desktop is better than the other, it is mainly a matter of personal taste. Some distributions also have a kind of standard desktop, i. e. the desktop that is installed by default. Here Fedora is traditionally a Gnome-based distribution.

In former times Fedora used Gnome 2 and right now it is equipped with Gnome 3 as standard desktop. Gnome 3 has a completely new and different control concept whereas Gnome 2 is more traditional. At F&S we wanted to stay with the traditional intuitive Gnome 2 concept so we switched to the actively developed Gnome 2 fork called MATE.

In our opinion MATE needs less time to get used to it than Gnome 3 especially for people that comes from windows. Therefore we recommend using a different desktop than the default one. Furthermore all our Fedora screenshots are taken with the MATE desktop.

If you want to start with an installation based on the MATE desktop right away, you have to download a different installation file that is available as a so-called Fedora "spin". Go to http://spins.fedoraproject.org and select the "Mate-Compiz" version of Fedora (see Figure 4). It does not differ much from the standard release, it just installs MATE instead of Gnome 3 as the default desktop. If you later decide that you don't like it, you can still install other desktop systems by using the package managing tool.

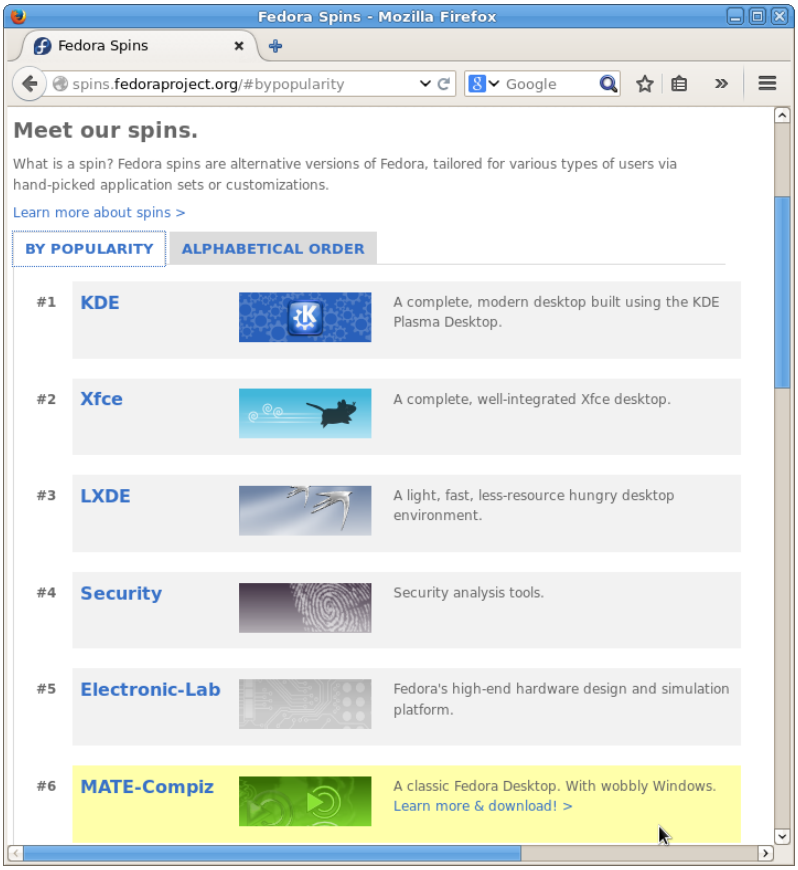


Figure 4: Download MATE spin

In the same spirit if you already have installed Fedora with a different desktop, then you can add the MATE desktop by hand if you want. Simply start the package managing tool, select the "MATE Desktop" group, activate and install it. Then log out or simply restart your system.

The same is true if you are using Ubuntu. Ubuntu installs the Unity desktop by default. If you want to start with a different desktop, you have to download Kubuntu (for KDE), Xubuntu (for XFCE), Lubuntu (for LXDE), or Ubuntu Gnome (for standard Gnome-3 Shell). Of course any of these variants allows installing a different desktop later, too.

If you have several different desktops installed in parallel, you can select the one for your next session in the login screen at the stage where you enter your password. There is an icon or menu where you can click on and select one of the installed desktop variants.

1.3 Using Virtual Machines

The best way to develop for our Linux boards is to use a dedicated Linux PC. This offers the best compilation performance and accessing devices like network and USB is guaranteed to work.

But it is also possible to run the development PC as a virtual machine. For example our default environment at F&S company is Microsoft Windows, and we are running the development PCs for our Linux boards as Linux guests in virtual machines under our Windows hosts. We can even have several such virtual machines open at the same time. Each of these virtual PCs is running in an own window on our host and we can switch back and forth between the host and the guests at will without having to reboot the PC.

There is a free virtual machine software available from Oracle (previously Sun) that is called *VirtualBox*. This works without any problems, even passing USB traffic or serial ports to the guest works as expected.

Figure 5: VirtualBox

Installation of VirtualBox is rather straightforward. It uses a standard installer like most windows software. Then you have to create the virtual machine. Just give the type of Linux system that you want to install (e. g. Fedora 64-bit) and then create a virtual hard disk for it. Please define a hard disk with at least 25 GB of size, the bigger the better. If you can afford 50 GB or even 100 GB, you will not regret it. Different revisions of the root filesystems take quite a lot of space. And if you want to work with Yocto, you should have at least 50 GB of free space.

When asked for the memory to reserve for the virtual machine, you should say at least 2 GB of RAM and 2 GB of swap space. Linux uses RAM rather economically, but some compilation tasks may take quite a lot of memory, especially if they use many instances in parallel. And if you have more than one CPU core, you can also tell how many cores should be made available in the virtual machine. Again the more the better, this can speed up some compilation tasks considerably.

After you have created the virtual machine by saving all the settings, you can either insert the Linux installation CD into the CD drive, or you can mount the appropriate installation ISO image as a virtual CD drive. Then switch on the virtual PC. It will access the (virtual) CD drive and boot the Linux installer from there, and the normal installation procedure will take place, exactly as it would be done when installing for a regular PC. After everything is complete, you will have a regular Linux system on your virtual hard disk.

It is recommended to also install the so-called VirtualBox Guest Additions. These additions make the interchange between host and guest even more smooth. Please note that you may have to re-install these Guest Additions if you update the Linux kernel of your Linux guest.

There is one pitfall that can cause grey hair. If you want to access the virtual guest PC in parallel to your standard host PC, you have to set the network mode to “Bridge Mode” in the network configuration of the guest. Otherwise the guest can neither access the network directly nor can external hosts like our board access the Linux guest. Then services like TFTP

and NFS will not work and you don't know why. So please set “Bridge Mode” and everything will work.

Remark

Using a virtual machine is also an additional reason for not using the Gnome 3 desktop. Gnome 3 needs 3D graphics support which is often not fully available in a virtual machine.

2 Working With sudo

When working with Linux, you should always work as a standard user with restricted rights. This reduces the risk of damaging the system by accident. For example calling

```
rm *
```

as superuser “root” when in a system directory instead of a local project directory could cause a lot of data loss and can even result in a completely unusable Linux system.

However sometimes you have to be the root superuser to do some administrative task, for example when mounting or unmounting a filesystem, installing software or configuring the system. Usually this would require to switch to super user mode with command `su`, enter the root password and then execute the necessary commands. Then you would leave root mode again with `exit`. This is rather time-consuming, especially if only very few root commands have to be executed, but on a regular basis.

Having a second window with a root shell continuously open is also no good solution. For example you have to switch directories in both windows if you are doing work in different paths. And typing a command in the wrong window will lead us back to the potential danger of damaging the whole system by accident.

Here the command `sudo` is very handy. `sudo` allows to execute one command that is given on the command line with root privileges. When calling `sudo` the first time, you have to identify yourself with your own (!) password. Then for a configurable amount of time you don't need to enter the password again, even when issuing more `sudo` commands. If this time period passes without another `sudo` command, then you have to enter your password again at the next `sudo`. So to speak you gain root privileges for a small amount of time.

To make this work, you have to add your standard user name to the “sudoers” list. Just edit the file `/etc/sudoers` as root and add the two lines:

```
<user>     ALL=(ALL)   ALL
Defaults:<user> timestamp_timeout=15
```

The first line enables the user to use command `sudo` now. Here we could restrict the actions this user could do in privileged mode, but to keep it as simple as possible we just allow everything that root can do. The second line sets the default timeout to 15 minutes instead of five minutes which is the default. Five minutes is often a little bit too short. Of course you have to type your real user name instead of `<user>` in both lines.

In fact there is a second command that can be used after having been added to the sudoers list: `sudoedit`. This command allows editing global configuration files in a more safe way. This is done by editing a copy (!) of the original file, keeping the original content until the modification is fully done. When the file is saved and the editor is closed, the temporary file is used to overwrite the original file, making the replacement an atomic operation. By this procedure, there is never an inconsistent, half-edited configuration file active.

Remark

From now on we will use `sudo` and `sudoedit` in all our examples that need root access.

3 Some Useful Settings

Some settings, that are not set by default, make the work with Linux much more comfortable. Settings that you only want to have for yourself, you can edit in `~/.bashrc` or `~/.bash_profile`. Settings that you want to have for all your users (including root), you can add as a file `/etc/profile.d/myconfig.sh`.

Here is what the author has in these files:

```
# local aliases
alias rm='rm -i'
alias Rm='rm -f'
alias pd='pushd'
alias mv="mv -i"
alias cp="cp -i"

# Now the next one is tricky. Usually calling one of the above commands with
# sudo does not use the alias, but the standard command instead, which is
# unfortunately the unsafe version. This may be fatal. However we can use a
# special feature of aliases:
# If an alias value (i.e. the replacement command) ends in a blank, then the
# next word on the command line is also checked for aliases. Therefore if we
# make sudo to an alias itself with an ending blank, the next word (which is
# the command to execute) is alias-expanded too, which results in exactly
# what we want.
alias sudo="sudo "

# Copy history lines to be edited before executing them
shopt -s histverify

# Don't store duplicate lines in history
export HISTIGNORE="&"
```

Listing 1: Sample for /etc/profile.d/myconfig.sh

```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
       . /etc/bashrc
fi

# Readline bindings. Somehow ~/.inputrc is not read.
bind 'set completion-ignore-case on' 'set input-meta on' 'set output-meta on'

# User specific aliases and functions
alias ls='ls -F'
alias ll='ls -lF'
alias em="emacs"

# Editor to use in standard situations
export EDITOR=emacs
```

Listing 2: Sample for ~/.bashrc

Explanation

The command `rm` silently removes all given files. Especially when using wildcards, this may be very dangerous, as it does not use any trash can where files could be restored in case of error. Therefore in the authors opinion `rm` should ask for each file before actually removing it.

If a bunch of files should really be removed without asking, then use `Rm`, which must be typed deliberately, switching the case. But then please double-check if all file names are correct. In the same spirit, commands `cp` and `mv` should ask first before overwriting any existing files. The shortcut `pd` for `pushd` is also very convenient.

Command `sudo` should also obey to these aliases. Otherwise it would be even more dangerous if the standard user is used to being asked before removing/overwriting files, and when working as superuser he is not asked and all files are removed silently.

The last two commands help with the command history. The first command ensures that a command that is given with prefix '!' is copied from the history to the command line to be edited before being executed. And the last command only stores a command in the history if it is different from the previous command. This makes scrolling through the command history with arrow up and down easier because there are less entries.

The settings from `.bashrc` allow the file name completion on the command line to find also names that were given in the wrong case. And option `-F` adds a character to special file names when using `ls` and `ll` (`/` on directories, `*` on executables, `@` on links).

Because the author always uses the `emacs` editor when editing files, it can be called with the shortcut `em` and it is automatically invoked in standard situations as the result of the last two commands. Especially setting environment variable `EDITOR` is recommended, because the editor set here is also used when calling `sudoedit` in some of the instructions of the remaining document. If this is not set, a plain `vi` editor is used by default. Of course you can change this variable to your favorite editor. If you are not so convenient with linux maybe nano (a command line editor) or pluma (the default GUI editor on MATE desktop) will fit for you.

4 Software Installation And Updates

Every Linux distribution consists of a lot more software packages than are installed on your computer by default. The far bigger part is available to be installed manually. There are literally thousands of packages that can be installed for free, from small helper utilities up to large program systems like the development environment Eclipse or the 3D animation software Blender.

To make the software selection easier, Linux distributions usually divide their available software into groups, for example software related to a desktop system (Gnome, KDE, MATE, Cinammon, etc.), multimedia, office, games, graphics, programming, and so on. For example in Figure 6, you can see the available package groups of the Fedora distribution, where the MATE group is expanded to show the available packages. The software managing program from your Linux system allows you to browse these groups and select which packages you want to install.

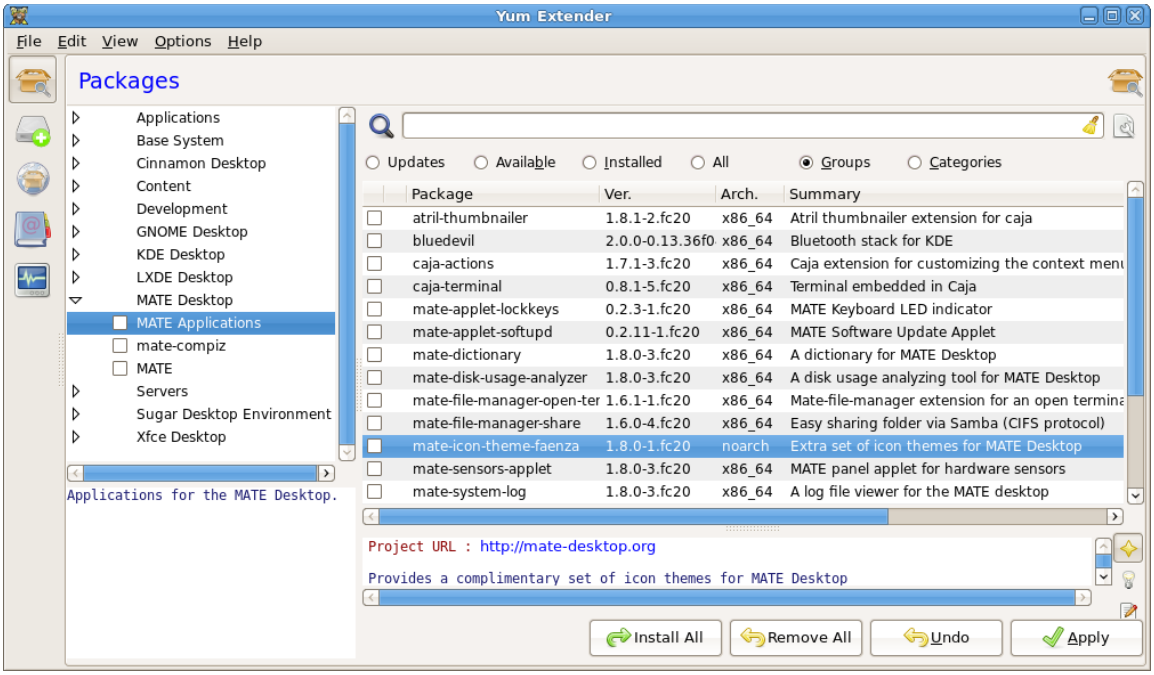


Figure 6: Package groups in Fedora

You can even include additional repositories that provide still more packages, for example from third parties or organizations with certain topics like educational software, embedded software or science applications.

There is one rather common case where it is useful to add an additional repository. Many Linux distributions have a very restricted view on Open-Source licenses. Therefore all programs that are not absolutely free of software patents or other restrictions, are not included with the regular distribution itself. For example there are often no MP3 audio and MPEG video decoders available in a Linux distribution because there are patents on the MP3 and MPEG algorithms. Or the proprietary graphics card drivers from Nvidia and AMD are usually also in a gray area and not directly available from the distribution.

However most distributions maintain a kind of shadow repository where such packages with not-so-clean license conditions can be found. These repositories can usually be added easily to the list of repositories and then all these additional packages become available. There is usually a website associated with the repository that tells exactly how this is done for a specific distribution. Two such examples would be RPM-Fusion for Fedora (http://www.rpmfusion.org) and Packman for OpenSuse (http://packman.links2linux.org).

Every distribution has at least one software management tool (or package manager) that is capable of installing updates, new packages or uninstalling packages. For example if there are any updates to the installed software packages, you are usually informed automatically by a message box. Then you can instruct the software manager to install these updates.

In general the software manager is responsible to handle all intermediate steps required to do this. For example to use a specific software package, it may be necessary to install some additional libraries first. In this case the software manager will download all required packages and installs them in one go.

Unfortunately exactly this software management is one of the parts where all Linux distributions differ. The distributions compete in having the most packages, the most comfortable installer, the easiest handling, and so on. So software management is always a distribution-specific task.

In addition, two different package file formats have been established in the past: RPM packages (the name is an abbreviation for “Red-Hat Package Manager”) and Debian packages. Package files define what the packages provide, what prerequisites they need, the versions, and many more things. Unfortunately these two formats are not compatible, so your distribution is either using the one or the other format.

4.1 Essential Packages For Compilation

Some packages are not installed by default but we need them for a successful compilation of our source code. Otherwise compiling will break because come header files e.g. curses.h could not be found. Packages that can lead to trouble if they are missing are listet below. I686 refers to the 32bit variant of the package:

- glibc (i686)
- glibc-devel (i686)
- glibc-devel
- zlib (i686)
- zlib-static (i686)
- ncurses-devel
- bison
- bison-runtime
- bison-devel
- texinfo
- wget

- patch
- perl-ExtUtils-MakeMaker
- git
- automake
- autoconf

4.2 Install Packages On Fedora

Software under Fedora is packed with the RPM program. With the `rpm` command you can query the installed packages. For example

```
rpm -qa
```

lists all the packages that are installed on your computer. Be warned, this list is rather long. Use option `-i` to show additional information for a package. For example to show this information for package `nfs-utils`, you can give this command:

```
rpm -qi nfs-utils
```

To list all files that are provided by this package, type:

```
rpm -ql nfs-utils
```

You can also determine to which package a specific file on your PC belongs to. For example if you want to know where the file `/sbin/rpc.statd` comes from, you can type:

```
rpm -qf /sbin/rpc.statd
```

and you will see that it belongs to package `nfs-utils`.

Packages also have a version number and platform information attached to the name. For example the last command outputs `nfs-utils-1.3.0-2.1.fc20.x86_64` on the author's computer. This means it is version 1.3.0 of the `nfs-utils` program, in the package release 2.1 for Fedora (Core) 20 on x86_64 platforms. If you just want to reference the package in general, you can omit all this stuff. It is only needed if you want to reference a specific version, for example when you need to install exactly that specific version of a software.

However looking at the RPM packages is a rather low-level view of the things. The larger picture is to see what new packages are available, what additional packages can be installed and of course to actually download and install those software packages. Under Fedora, this is done with a tool called `yum`. `yum` knows what is installed on your computer (by maintaining a package database), knows what packages are available elsewhere and knows the dependencies between all the packages. You can list packages, install or update packages and you can remove packages. For example to list all installed packages, you can say:

```
yum list installed
```

This will result in a similar list as `rpm -qa` above, but it also shows from which repository the package was taken and installed. To see if there are updates available for any of your installed packages, you can give the command:

```
yum list updates
```

In this case, `yum` must download the newest repository data, which is why this command may take a while and also needs network access. Depending on how up-to-date your local installation is, this may result in a more or less comprehensive list. The command:

```
yum list available
```

lists all packages that are available on any repository. This list is even longer than the list of already installed packages.

4.2.1 Updating/Installing Software (Command Line)

To update a specific package on your system, just call `yum update` with the package name. For example to update the Firefox web browser to the newest version just call:

```
sudo yum update firefox
```

You have to be root to do this, therefore the `sudo` before the `yum` command. Now `yum` checks all the repository databases, determines which files have to be downloaded and then presents a list of actions it wants to do. Then it waits for an acknowledgement from your side. Pressing `n` will abort the update, which is quite handy if you see that the update will not do what you intended to do. Pressing `y` will start the update process. Now all the files of your package are downloaded, tested, installed and the previous version of the package is removed.

You can also give several packages on the command line in one go. Then all these packages are updated. The following command will update `firefox` and `yum` itself:

```
sudo yum update firefox yum
```

If you don't give any name, then all packages with available updates are updated. This can take a while.

```
sudo yum update
```

`yum` is quite intelligent when downloading packages. To reduce bandwidth, sometimes only so-called delta-packages are transferred and then combined with the previous version to get the full package.

Installing software is similar simple. For example to search for all packages that relate to “nfs”, just give the command:

```
yum search nfs
```

This not only finds packages where “nfs” is contained in the name, but also packages where “nfs” appears anywhere in the description text. By looking at the result, we find out that we need the packages `system-config-nfs` and `system-config-nfs-docs`. Then we can install these packages with:

```
sudo yum install system-config-nfs system-config-nfs-docs
```

Again you have to be root to be able to install new software. The installation itself executes completely similar to the updating process, including the list of planned actions and the possibility to abort everything before anything is actually modified.

4.2.2 Updating/Installing Software (GUI)

In previous versions, the Gnome and MATE desktops used two different programs for updating and installing software, showing up as “Software Update” and “Add/Remove Software” in the GUI. They were part of the PackageKit package.

In 2013, Gnome switched to a new program called `gnome-software`. This new version tries to make software management similar to an APP store on a tablet or smartphone. It concentrates on user applications and tries to hide libraries and system stuff from the user to make it more straightforward. Even though the idea by itself is not bad, the new version simply does not work as good anymore. On our system, `gnome-software` often reports an up-to-date system, when in fact there were dozens of updates available. Or for example it is much more difficult to install a new desktop version like MATE or KDE with this program. You simply can't find the programs that are required for this, because `gnome-software` does not show the package groups anymore. So we think that `gnome-software` needs some more time to mature and get more reliable.

In the meantime we recommend a different graphical front-end that is called *Yum Extender.* This is simply a one-to-one front-end for `yum`. If you know how `yum` works, you can immediately work with Yum Extenter, too. If Yum Extender is not automatically available in your Fedora installation anyway, you have to install it once, for example by calling:

```
sudo yum install yumex
```

After installation, Yum Extender is available as `yumex` on the command line or under menu entry *System → Administration → Yum Extender*.

When Yum Extender is started, you will see the Updates view (see Figure 7). If you have a different view, you can switch to this view by clicking on *Updates* at the top. Here you will see in a red font all the packages where updates are available. Of course if your system is up-to-date, this view is empty. Otherwise simply tick the packages you want to update (or untick the packages you want to keep) and then you can start the update process by clicking on *Apply*. You may have to legitimate yourself by entering your password to be allowed to do modifications to the software installation.

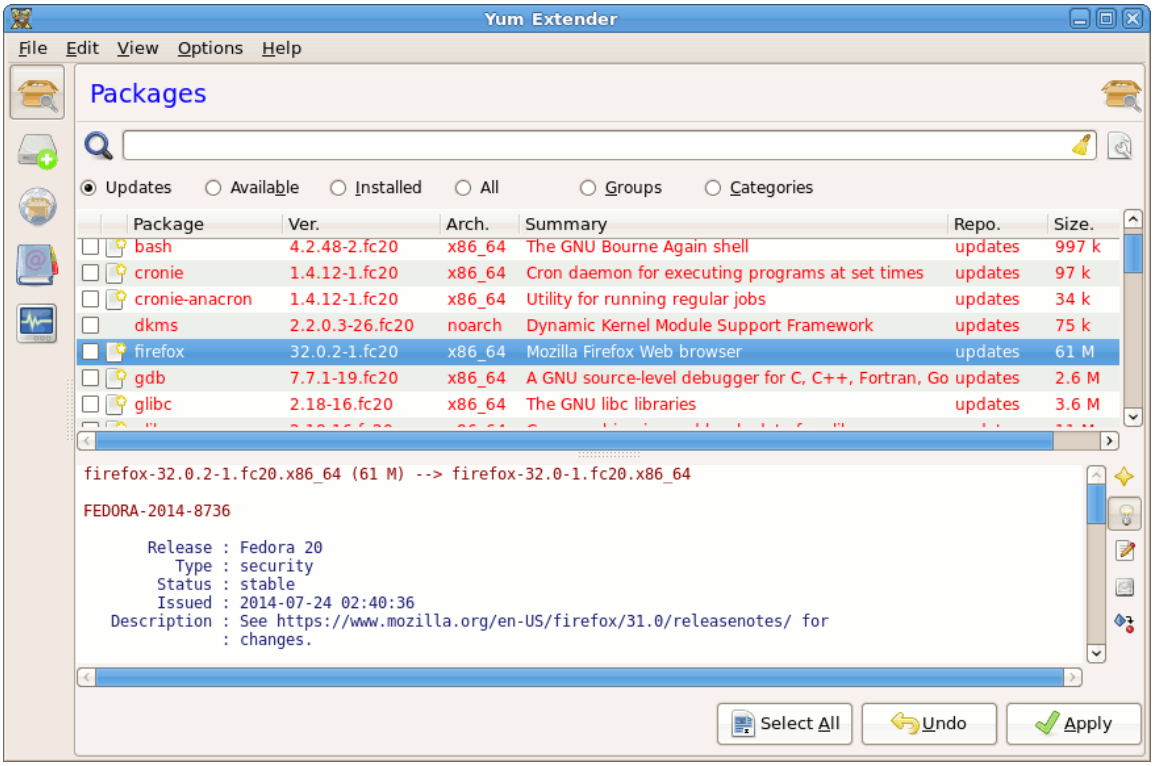


Figure 7: Updates view in Yum Extender

Yum Extender now simply calls `yum` in the background, i. e. it checks the data bases and dependencies. In a second dialog you will be informed about any additional packages that need to be installed, updated or removed. Here you can still abort the update. Typically you will accept the list and then Yum Extender downloads all packages and installs them.

Installing new packages with Yum Extender works completely similar. Just select *Available* or *Groups* at the top to list all available packages or all available groups. Here “available” means packages that are not yet installed. These packages are displayed using a black font (see Figure 8). To search for a specific package, just enter the name or part of it into the search bar. Then in the list of packages tick those package(s) that you want to install and click *Apply*. The same sequence as when updating packages will take place, including the possibility to abort installation after all dependencies have been checked. Again you have to legitimate yourself to be allowed to install software.

Three more package views are available in Yum Extender. If you select *Installed,* you will only see packages that are already installed. They use a green font. This is useful if you want to uninstall a package. If you select *All*, then you see all packages that can be found: updates (red), installed (green) and available packages (black). And finally if you select *Categories*, you can restrict the visible packages to a single repository or to a size range.

The icons on the right side of the lower pane allow switching the type of information that is shown in this lower section: the package description, update information about the package, the package history (change log), the list of files contained in this package (manifest) and the list of dependencies, i. e. which other packages need to be installed as prerequisties for this package.

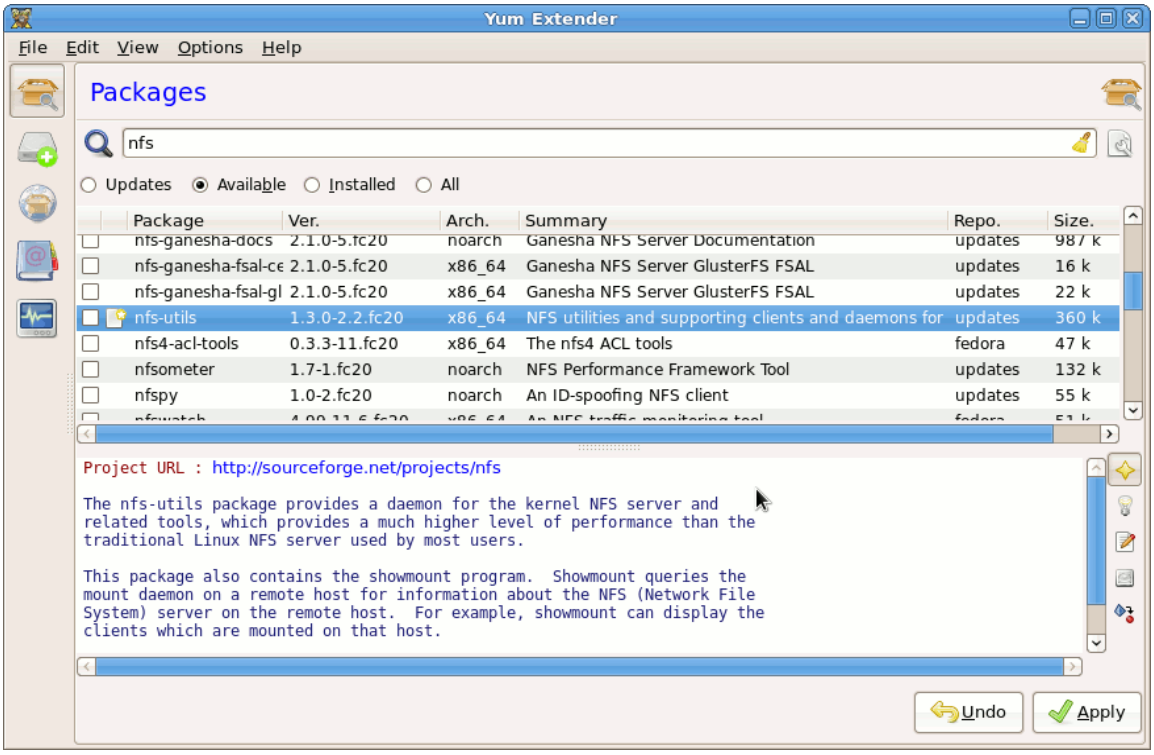


Figure 8: Available packages in Yum Extender

Besides all these different package views, you can also use the icons on the left side to show all pending actions (things that will be done if you click on *Apply*), all active repositories, a history log of all software management actions in the past and the output log of the current invocation.

5 Services

Many features of Linux are provided by services that are running as so-called *daemons* in the background. These services are usually started at boot time and remain alive until the PC (or virtual machine) is shut down again. However sometimes it is necessary to restart an already active service or to start a new service while the system is running.

For example if you mount a filesystem image to some directory and then export this directory via NFS, you have to start NFS *after* mounting the image. Then if you want to exchange the filesystem image later with a newer version, you first have to stop the NFS service, unmount the old filesystem image, mount the new image and then start the NFS service again.

Or if you suspend a virtual machine and re-activate it a few days later, the Linux system running in it is probably not aware of this interruption and may not correctly reconnect the network. In this case it is sufficient to restart the network service to get the network up again.

5.1 Fedora

The Fedora Linux System used to work with a System-V based run-time environment. This was rather simple. Linux started the first process called `init`. The `init` process had a global configuration file `/etc/inittab` that defined what tasks (precesses, services, etc.) should be started. By defining a few different so-called *runlevels,* the system could be brought to different working modes. Runlevels were numbered 0 to 6 and there was a special runlevel named S (see Table 1).

Runlevel	Description
0	Halt the system
1	Go to single user mode
S	Single user mode
2	Multiuser mode, but without networking/NFS
3	Full multiuser mode
4	(unused)
5	Graphic mode (X11)
6	Reboot the system

Table 1: Runlevels in previous Fedora versions

Each level <n> consisted of specific shell scripts in `/etc/rc<n>.d` that told which service had to be started or stopped in this runlevel. This was decided by prefixing the script name with `S` for services that should be started and `K` for services that should be stopped (killed). In addition was a two-digit number defining the launch/kill sequence. In fact these `S` and `K` files were only symbolic links to the actual script files in `/etc/init.d`. This run-time environment was simple, but also slow and not very flexible.

Services

Starting with Fedora 15, Fedora switched to a `systemd` based run-time environment. The start scripts, the file `/etc/inittab` and the limited set of runlevels from previous versions are gone now. They are replaced by configuration entries called *units*. Each unit can either describe a single service, or a so-called *target.* Like a runlevel, each target has its own set of running and stopped services. But as there can be many different targets, it is possible to describe the system in a much finer granularity than it was possible with runlevels. And services can also be started in parallel, which results in a faster system start.

A unit (service or target) can tell which other units it requires and whether it must be executed before or after some other unit. These dependencies are evaluated by `systemd` and an optimal start sequence is chosen with as many services as possible started in parallel. All these configuration units are defined in directory `/usr/lib/systemd/system`. A few examples for targets can be found in Table 2.

Target	Description
`basic.target`	Start a basic system
`rescue.target`	Start a minimal system (like runlevel 1)
`multi-user.target`	Start a command line based system (like runlevels 2 to 4)
`graphical.target`	Start a graphical system (like runlevel 5)
`network.target`	Start the networking environment
`reboot.target`	Reboot the system (like runlevel 6)
`poweroff.target`	Shutdown the system and power off (like runlevel 0)

Table 2: Targets for systemd in current Fedora versions

For example when looking at the definition of `multi-user.target` in Listing 3, you can see that it “requires” the `basic.target`. This means `basic.target` must also be executed when `multi-user.target` is started. And it also tells that it needs to be run “after” `basic.target`. If this additional condition was not given, both targets would be started in parallel.

```
#  This file is part of systemd.
#
#  systemd is free software; you can redistribute it and/or modify it
#  under the terms of the GNU Lesser General Public License as published by
#  the Free Software Foundation; either version 2.1 of the License, or
#  (at your option) any later version.

[Unit]
Description=Multi-User System
Documentation=man:systemd.special(7)
Requires=basic.target
Conflicts=rescue.service rescue.target
After=basic.target rescue.service rescue.target
AllowIsolate=yes
```

Listing 3: Fedora's `multiuser.target` unit

5.1.1 Managing Services (Command Line)

The command line tool to manage services is the program `systemctl`. You must give a command and the appropriate unit name as parameters. Table 3 shows the commands that are available with `systemctl` to manage services.

Command	Action
`start`	Start the service
`stop`	Stop the service
`restart`	Restart the service (i. e. first stop, then start again)
`reload`	Tell the service to reload the configuration file
`status`	Report the state of the service (stopped ot running)
`enable`	Enable the service (i. e. start at system start)
`disable`	Disable the service (i. e. do not start at system start)

Table 3: `systemctl` commands for services

For example to restart the NetworkManager service you can use the following command:

```
sudo systemctl restart NetworkManager
```

The state of all services can be determined with:

```
systemctl status
```

Remark

The command `service` from previous Fedora versions is still available and will automatically be converted to a call of `systemctl`.

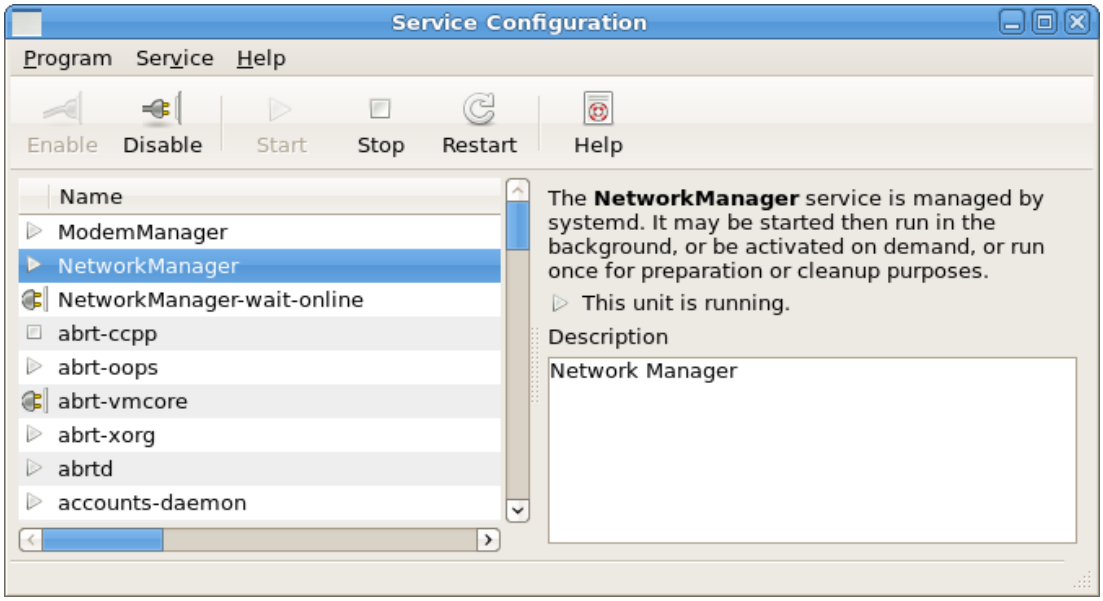


Figure 9: Managing services with `system-config-services`

5.1.2 Managing Services (GUI)

If you have installed the Fedora package `system-config-services`, then there is also a graphical tool with the same name `system-config-services` available under menu *System → Administration → Services*. This will start the dialog in Figure 9.

Here you simply click on the service name in the left pane and then on one of the action buttons *Start*, *Stop* or *Restart*. For example to restart the NetworkManager service, just select *NetworkManager* in the left pane like in the image and then click on *Restart*. You may have to validate yourself with your password to be able to change settings.

This dialog also allows configuring services, that is enable or disable services at system start by clicking on the *Enable* and *Disable* buttons.

6 Handling The Firewall

A current Linux system has an active firewall. That means network requests coming in and going out are filtered and only the ones that are allowed by the firewall rules are passed through. All others are rejected. When a new network protocol like TFTP or NFS is installed, it is also necessary to add appropriate rules to the firewall to let these network requests through.

Firewalls make use of the network routing of the Linux kernel that is called *iptables*. It is called iptables because the kernel holds several tables that define which network request will be routed which way, depending on network port, network protocol, network interface and direction (incoming, outgoing). `iptables` is also the name for a command line tool to manipulate these tables on a very low level.

There exist many different firewall front-ends that build on this iptables network routing in the kernel, for example the *iptables* service, the firewall daemon *firewalld*, *Firehol*, *Firetable*, *ferm*, *ufw/gufw*, *Firestarter*, and so on. All these front-ends try to make the life with all these tables and rules easier, usually by providing predefined sets of rules for common use cases and introducing more abstract concepts like NAT (network address translation), zones, port forwarding and similar. Again it depends on the Linux distribution which firewall front-end is actually used.

6.1 Fedora

Fedora used to have a static firewall based on the *iptables* service in the past. The disadvantage was that the whole set of routing tables had to be reloaded even if only one rule was changed. This interrupted existing communications and was not very flexible. Therefore current Fedora versions use a more dynamic firewall system that is now based on the firewall daemon *firewalld.* Here rules can be changed without interrupting any ongoing communications.

Firewalld defines different trust zones. In each zone you can tell if some networking service is allowed or not. In a trusted zone, many services may be enabled, while in an untrusted zone only a few or even no service at all is allowed. For each network interface you can define a different zone. For example you can set a rather restricted zone for the WLAN interface while at the same time the wired LAN interface works in a zone that has more freedom.

The following Table 4 shows the predefined zones in Fedora, sorted from untrusted to trusted.

Zone	Description
drop	Any incoming network packets are dropped, there is no reply. Only outgoing network connections are possible.
block	Any incoming network connections are rejected with an icmp-host-prohibited message for IPv4 and icmp6-adm-prohibited for IPv6. Only network connections initiated within this system are possible.
public	For use in public areas. You do not trust the other computers on networks to

Zone	Description
	not harm your computer. Only selected incoming connections are accepted.
external	For use on external networks with masquerading enabled especially for routers. You do not trust the other computers on networks to not harm your computer. Only selected incoming connections are accepted.
dmz	For computers in your demilitarized zone that are publicly-accessible with limited access to your internal network. Only selected incoming connections are accepted.
work	For use in work areas. You mostly trust the other computers on networks to not harm your computer. Only selected incoming connections are accepted.
home	For use in home areas. You mostly trust the other computers on networks to not harm your computer. Only selected incoming connections are accepted.
internal	For use on internal networks. You mostly trust the other computers on the networks to not harm your computer. Only selected incoming connections are accepted.
trusted	All network connections are accepted.

Table 4: Fedora firewall zones

You can add more zones if the list is not sufficient for your purposes.

A service from the point of view of firewalld is basically a set of port numbers combined with the network protocol TCP or UDP. So you don't have to list all ports that are used by the system service to enable it, it is sufficient to simply enable the appropriate firewalld service. Examples for firewalld services are *mountd*, *ftp*, *tftp*, *rpc-bind*, *ssh*, *nfs*, *telnet*, *http*, *https*, *smtp*, *ntp*, and so on. Fedora knows of about 60 predefined firewalld services.

If you want to enable a service that is not known to firewalld, you can either add a new named firewalld service by defining all the ports that this system service uses, or you can simply list all the ports without naming the service explicitly.

A firewalld service can either be enabled at run-time, probably only for a small period of time, or it can be enabled permanently. Therefore firewalld exposes two sets of zones, the run-time zones and the permanent zones. When the system starts, the run-time zones are taken from the permanent zones. From now on any modifications to the run-time zones take effect immediately, but they are only temporary. After the next restart of the system or at the next firewalld reload, the permanent settings are taken again. So to make a modification persist, you have to modify the permanent zone set. On the other hand if you modify the permanent zone set, these modifications are simply stored and do not take effect immediately. You have to reload the firewall to activate these settings by transferring them to the run-time set.

6.1.1 Modifying Firewall Rules (Command Line)

The command line program to modify firewalld is called `fireewall-cmd`. Usually you need superuser rights to modify anything, so `sudo` is your friend here. Only some minor queries can be done with normal user privileges. Here is a list of some useful commands.

Test if the firewall is active or not:

```
firewall-cmd --state
```

Get a list of all supported firewalld services:

```
firewall-cmd --get-services
```

Get a list of all supported zones:

```
firewall-cmd --get-zones
```

Get the default zone

```
firewall-cmd --get-default-zone
```

List the network interfaces, active firewalld services and extra ports for a given zone:

```
firewall-cmd --zone=<zone> --list-all
```

Enable a firewalld service in a given zone:

```
sudo firewall-cmd --zone=<zone> --add-service=<service>
```

Disable a firewalld service in a given zone:

```
sudo firewall-cmd --zone=<zone> --remove-service=<service>
```

Check if a firewalld service is enabled in a given zone:

```
sudo firewall-cmd --zone=<zone> -query-service=<service>
```

Enable a port/protocol combination in a given zone. `<protocol>` is either `tcp` or `udp`.

```
sudo firewall-cmd --zone=<zone> --add-port=<port>/<protocol>
```

Disable a port/protocol combination in a given zone.

```
sudo firewall-cmd --zone=<zone> --remove-port=<port>/<protocol>
```

Check if a port/protocol combination is enabled in a given zone. Please note that ports that are defined as part of a firewalld service are not found, only ports that were added as extra ports with `--add-port`.

```
sudo firewall-cmd --zone=<zone> --query-port=<port>/<protocol>
```

If you omit `--zone=<zone>` from these commands, then the default zone is used. All these commands manipulate the run-time zone set. If you want to change the permanent zone set, you have to add `--permanent` to each command.

Reload the run-time zone set from the permanent zone set:

```
sudo firewall-cmd -reload
```

There are quite a few more commands available, for example to add a new zone, to add a network interface to a zone, to provide port forwarding, masquerading or IP filters. Just call

```
firewall-cmd --help
```

to get a comprehensive list of options.

There is also a good documentation available at https://fedoraproject.org/wiki/firewalld.

6.1.2 Modifying Firewall Rules (GUI)

The graphical configuration tool is called `firewall-config` and can be found under menu *Services* → *Administration* → *Firewall*. You have to authenticate yourself to be allowed to view and change the firewall settings. It presents a dialog where you can select whether to modify the run-time zone set or the permanent zone set by selecting *Runtime* or *Permanent* in the drop-down box at the top (see Figure 10).

Figure 10: Run-time and permanent zone set

In the left pane you can select whether to change zone or firewalld service settings. If you select *Zones* on the left, then you can enable or disable firewalld services in the *Services* tab on the right side by checking or unchecking the firewalld service name (see Figure 11).

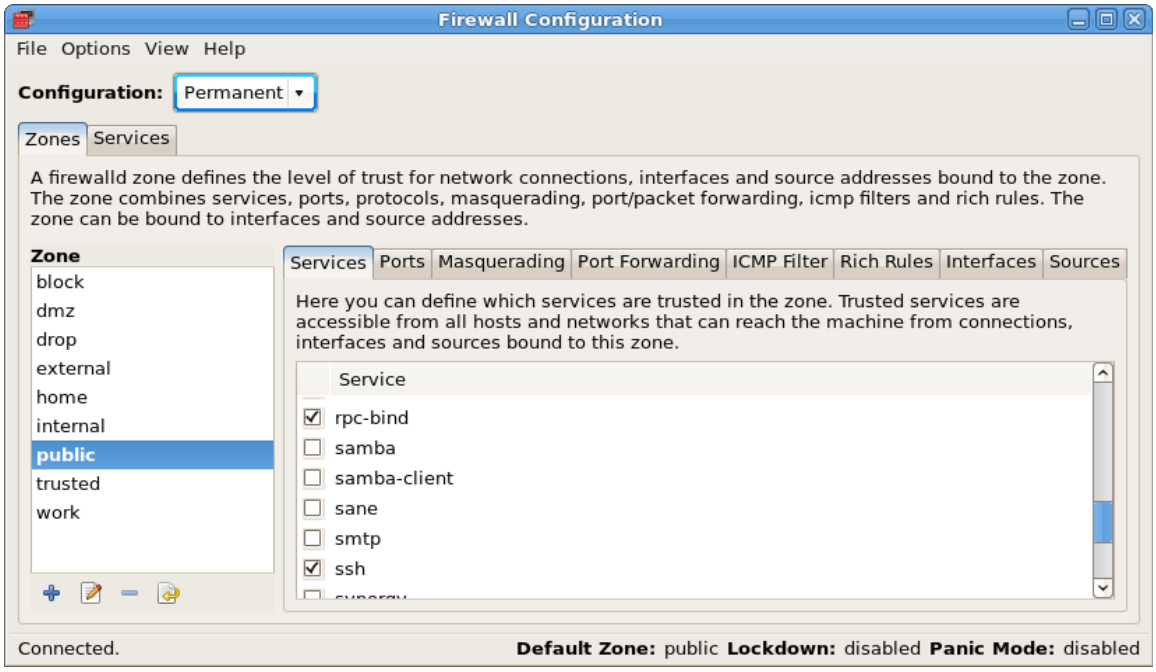


Figure 11: Enable or disable firewalld services in a zone

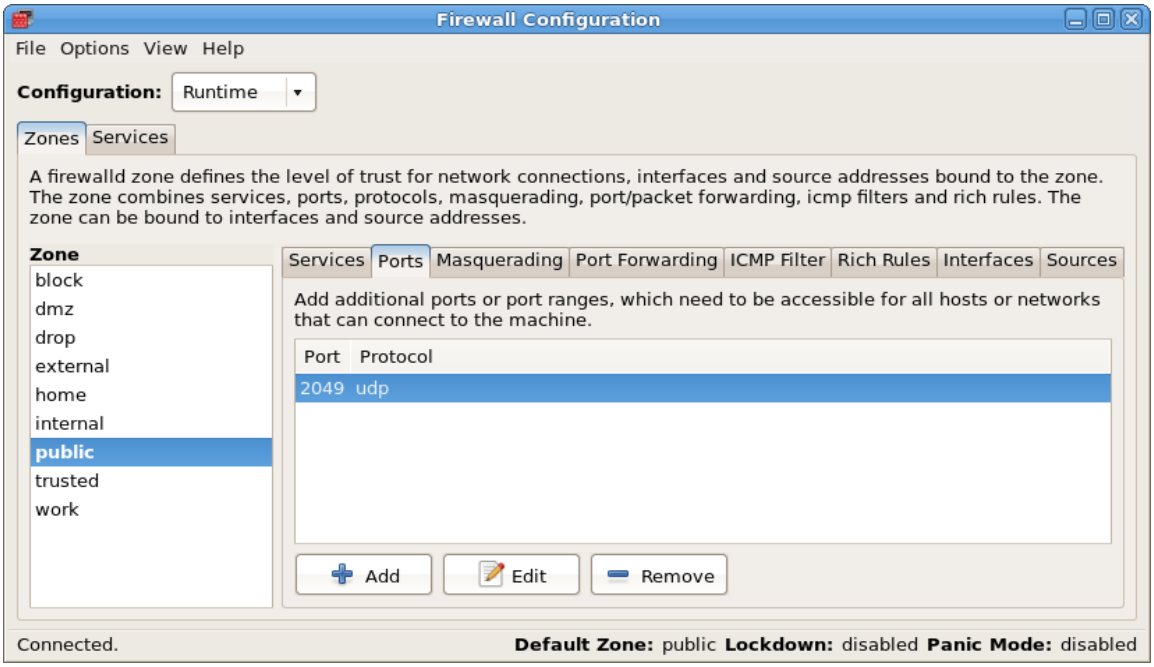


Figure 12: Enable or disable extra ports

Or you can configure any extra ports in this zone by selecting the *Ports* tab on the right (see Figure 12). You can also see that if you are in the *Permanent* zone set, you can also modify the list of available zones, which is not possible if you are in the *Runtime* zone set.

If you select *Services* in the left pane, then you can define which ports the firewalld service is using (see Figure 13).

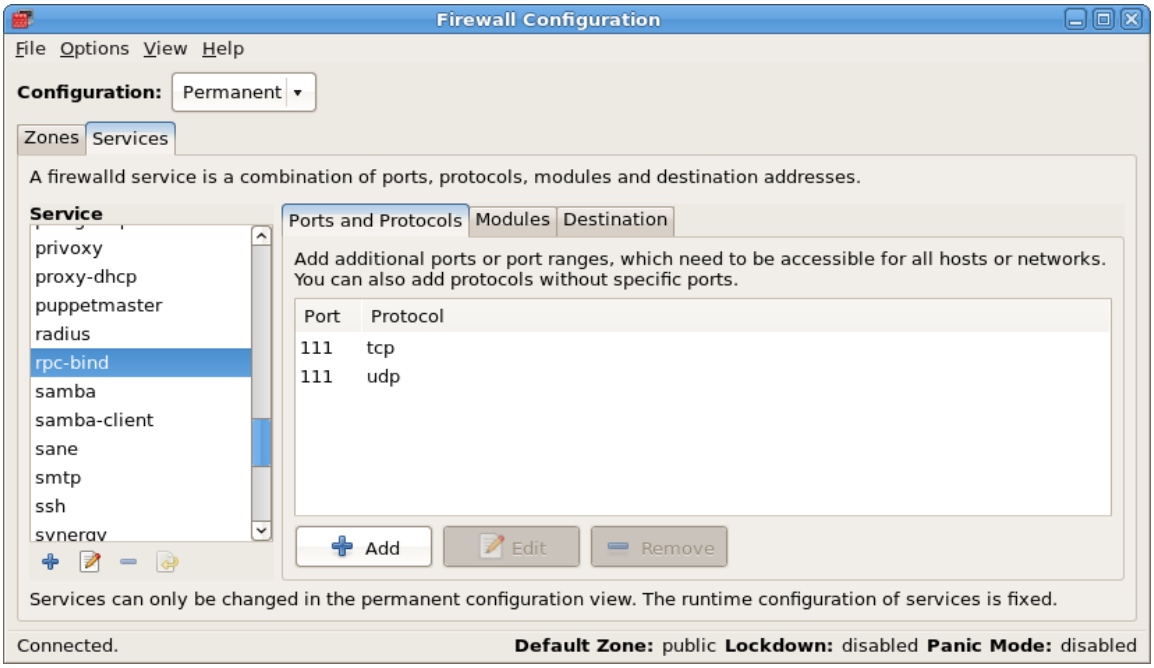


Figure 13: Define ports of a firewalld service

Again if you are in the *Permanent* zone set, you can modify the list of firewalld services, and if you are in the *Runtime* zone set, you can not.

Please note that modifications of the *Runtime* zone set take effect immediately while modifications of the *Permanent* zone set are simply stored. If you want to activate the permanent settings, for example when you are done with a bunch of modifications, simply call menu *Options* → *Reload Firewalld* to activate your new settings by transferring them to the *Runtime* zone set.

7 Installing TFTP Support

For downloading files to the board, mainly in U-Boot, the TFTP protocol is used. TFTP is a very simple version of FTP and can only send and receive single files in binary mode. When working with F&S boards and modules, we strongly recommend installing TFTP support on your PC or else most of our download instructions will fail.

Often TFTP is not a service on its own, but it is part of the larger `inetd` service. `inetd` is a framework for several small network services. If all these small services would start their own listener threads on the network, there would be quite a lot of threads, wasting a lot of memory and most of them would do nothing at all for most of the time. Therefore the meta-service `inetd` was invented. For all these small services, only `inetd` actually starts a few listener threads. If a service is requested, it detects the type of request and only then starts the appropriate daemon process. So if a TFTP request comes in, `inetd` will be triggered, detects that this is a TFTP request and then starts the TFTP daemon that does the real transfer. After the file transfer is complete, the TFTP daemon is stopped again, freeing all resources.

This means in addition to the TFTP server, we also have to think about the `inetd` service. This results in the following tasks.

- Install TFTP server packages
- Configure the TFTP server, e. g. the directory where to load/store files
- Configure `in etd` service if appropriate
- Enable the TFTP port in the firewall
- Enable the TFTP service and/or `inetd` service
- Test TFT server by doing a TFTP transfer

The following sections will configure the server to use directory `/tftpboot` for loading and storing files. This is the directory that is used in all F&S examples and instructions.

Remark

The PC will always be the TFTP server, you won't need a TFTP client there.

7.1 Fedora

Fedora uses exactly the variant as explained above, where TFTP is not a service on its own. Therefore we can not simply switch it on or off with the regular `systemctl` command line tool or the `system-config-services` GUI. Instead it is part of `xinetd`, which is Fedora's implementation of `inetd`. This is why we also have to tell `xinetd` that TFTP is available and `xinetd` should handle it.

7.1.1 Install TFTP Server Packages

To install the TFTP server on Fedora, you have to install the package called `tftp-server`, for example by calling:

```
sudo yum install tftp-server
```

This will automatically install package `xinetd`, too, if it was not available yet. Please see chapter 4.2.2 on page 14 for how to install software packages graphically.

7.1.2 Configure TFTP Server

By default the Fedora TFTP server looks into directory `/var/lib/tftpboot` for the files to be transferred. However we want `/tftpboot`. Therefore simply create a symbolic link from `/tftpboot` to `/var/lib/tftpboot`. In addition allow everybody to write to this directory, because then you can copy files there without having to use `sudo` all the time. And this also allows uploading files via TFTP from the board to this directory on the PC. This results in the following two commands.

```
sudo ln -s /var/lib/tftpboot /
sudo chmod a+w /tftpboot
```

7.1.3 Configure `xinetd` For TFTP

The directory `/etc/xinetd.d` contains all the configuration files for the services that are handled by `xinetd`. When the TFTP server package was installed, this also installed an appropriate configuration file for TFTP there. However the default configuration has TFTP switched off. Change this by editing the file `/etc/xinetd.d/tftpd`:

```
sudoedit /etc/xinetd.d/tftpd
```

Here change the line

```
    disable = yes
```

to

```
    disable = no
```

7.1.4 Open Firewall For TFTP (Command Line)

TFTP is a known service to the Fedora firewalld. Therefore you simply have to add the firewalld service *tftp* to the default zone of the permanent zone set and reload the firewall

```
sudo firewall-cmd --permanent --add-service=tftp
sudo firewall-cmd --reload
```

7.1.5 Open Firewall For TFTP (GUI)

Start `firewall-config` under menu *Services* → *Administration* → *Firewall*. Then select the *Permanent* zone set in the drop-down box at the top, *Zones* in the left pane and the default zone (printed in bold font, usually “public”). Now on the right side click on the tab *Services* and check the *tftp* firewalld service (see Figure 14). Then you can reload the firewall with menu *Options* → *Reload Firewalld*.

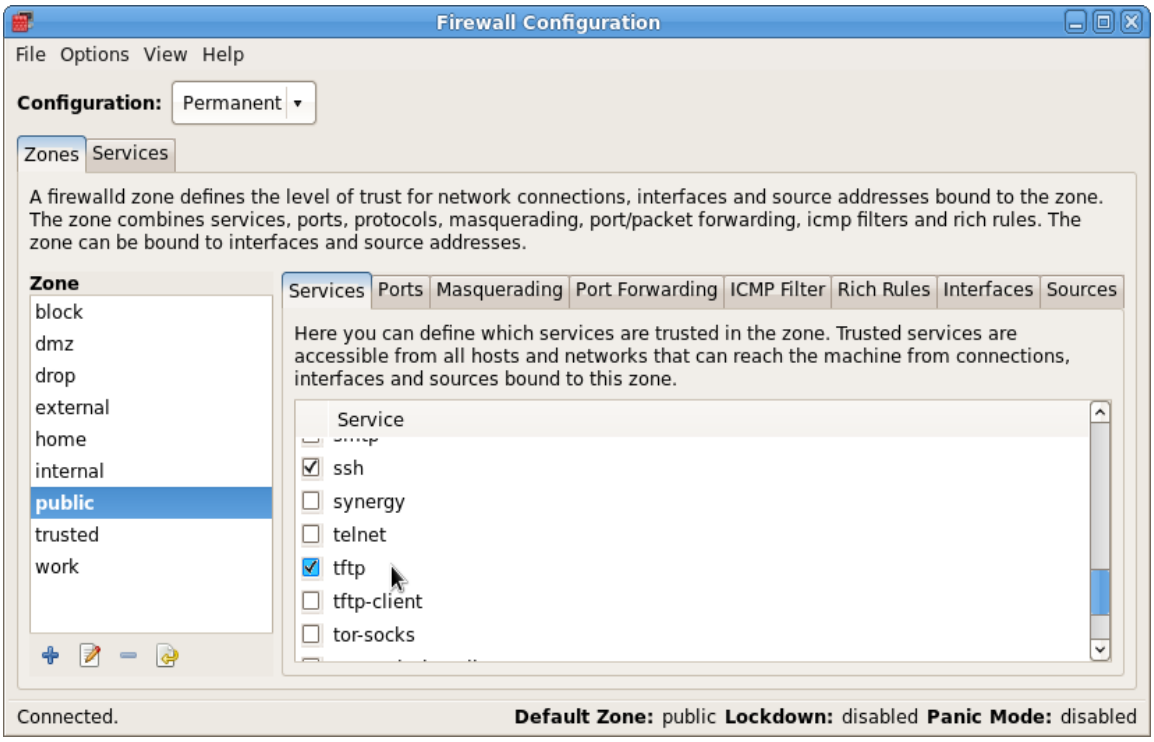


Figure 14: Activate tftp service in public zone

7.1.6 Activate `xinetd` Service

Now you can enable the `xinetd` service to be started automatically when the system boots up. And also start the service right away now.

```
sudo systemctl enable xinetd.service
sudo systemctl start xinetd.service
```

See chapter 5.1.2 on page 20 for how to enable and start services graphically.

7.2 Test TFTP Service

Copy a file to the `/tftpboot` directory, for example a U-Boot image `uboot.nb0`. The image itself does not matter, we just want to test whether it works at all.

```
cp uboot.nb0 /tftpboot
```

Then start your board. Stop in U-Boot by pressing a key while the autoboot time counts down. If not done yet, set the following environment variables (see Table 5).

Variable	Description
ethaddr	MAC address of your board
ipaddr	IP address of your board
serverip	IP address of your server (=PC)
netmask	Mask of your network

Table 5: U-Boot environment variables for TFTP

Save these settings with

```
saveenv
```

Now try the TFTP download:

```
tftp uboot.nb0
```

If the file is transferred, everything is OK. If there are any timeouts, something is still wrong.

8 Installing NFS support

While developing application software with the board, new versions will be compiled rather frequently. If it would be necessary to add each new version to the root filesystem first, build a new root filesystem image, reboot the board to U-Boot, transfer the root filesystem image to the board using TFTP, erase the previous image, store the new image to NAND flash memory and then reboot the board to Linux before being able to run the application, then the turnaround cycle times would be tremendous. Here it is far more convenient to have the PC export a directory with the current root filesystem via network and have the board mount this directory. Then no data needs to be moved to the board. Just copy the application to this exported directory on the PC, and it is immediately visible on the board and can be used.

Linux uses the Network File System (NFS) to do this. NFS is based on Remote Procedure Calls (RPC) and is available in three major versions: V2, V3 and V4. The first two are rather old, from the 1980, a time when firewalls were not invented yet. They arbitrarily use package based UDP or connection based TCP protocol, and they use random RPC ports to avoid man-in-the-middle attacks. However random ports are fatal when using a firewall. A firewall needs as few ports as possible and also constant (static) port numbers. Fortunately most NFS servers can be configured for static ports nowadays.

V4 is a recent re-implementation of NFS and removes many of these disadvantages. For example it only uses one RPC on TCP port 2049 now. Therefore V4 is considered safer than the other two versions. This is why today many Linux distributions only support NFS V4 by default. However when using the NFS protocol in U-Boot and when mounting the root filesystem via NFS in Linux, some NFS requests are still done using V2 and/or V3. This is why these versions must be enabled, too.

This results in the following tasks.

- Install NFS server packages
- Configure NFS server to support V2/V3, too; use static RPC port numbers
- Enable these static RPC ports in the firewall
- Define what directories are exported via NFS
- Start NFS service
- Test NFS service by doing a file transfer

8.1 Fedora

Fedora's NFS is mainly targeted to NFS V4. This means the NFS server does not support V2 and V3 by default and also the firewall only knows about TCP port 2049, which is the only port required for NFS V4. So when using earlier NFS versions, we need to make sure that the RPCs use static port numbers and we also have to add these ports to the firewall.

Table 6 shows which RPC daemons are implicitly launched when the NFS server is started. Some of these get their port numbers from the file `/etc/services` that lists all well-known

port numbers. Then they are already static. For all other ports we will use port numbers that are either prepared by Fedora in some way or that are commonly used by NFS nowadays.

RPC	Daemon	TCP port	UDP port	Remark
portmapper	`rpcbind`	111	111	Ports taken from `/etc/services`, handled by firewall service *rpc-bind*
idmap	`rpc.idmap`	-	-	No port required (NFSv4 only)
mountd	`rpc.mountd`	20048	20048	Ports taken from `/etc/services`, handled by firewall service *mountd*
rquotad	`rpc.rquotad`	875	875	Ports taken from `/etc/services`
nlockmgr	(kernel module)	32803	32769	Set in `/etc/sysconfig/nfs`
status	`rpc.statd`	662	662	Set in `/etc/sysconfig/nfs`
nfs	`rpc.nfsd`	2049	2049	Ports taken from `/etc/services`, The TCP port is already handled by firewall service *nfs*

Table 6: Fedora RPCs for NFS

For the firewall, we will add a new firewalld service *nfs23* that handles the additional ports for NFS V2 and V3.

8.1.1 Install NFS Packages

NFS requires Fedora packages `nfs-utils` and `nfs-utils-lib`. They can be installed with

```
sudo yum install nfs-utils nfs-utils-lib
```

If you also want to define the exported directories with a graphical tool, then you also need the package `system-config-nfs` and if you want to have documentation and help for this then you also need package `system-config-nfs-docs`. You can install them with

```
sudo yum install system-config-nfs system-config-nfs-docs
```

See chapter 4.2.2 on page 14 for how to install packages graphically.

8.1.2 Configure NFS V2/V3 And Use Static Ports

The NFS configuration is done in file `/etc/sysconfig/nfs`. There is no graphical front-end available for doing this, you just have to edit the file as superuser.

```
sudoedit /etc/sysconfig/nfs
```

Now look for the following lines. These lines are not in sequence, they are spread all over the file.

```
#LOCKD_TCPPORT=32803
#LOCKD_UDPPORT=32769
RPCNFSDARGS=""
STATDARG=""
```

Now modify these lines until they look like this, i. e. remove the comment hash mark from the first two lines and insert the text between the quotation marks of the last two lines.

```
LOCKD_TCPPORT=32803
LOCKD_UDPPORT=32769
RPCNFSDARGS="-V 2 -V 3"
STATDARG="-p 662"
```

This will enable NFS V2 and V3, and also sets the ports of the two RPCs “nlockmgr” and “status” to their appropriate static numbers.

8.1.3 Open Firewall for NFS (Command Line)

Unfortunately the command line front-end to firewalld has no feature to define a new firewalld service. However all firewalld services are held in XML configuration files. The built-in services are stored in directory `/usr/lib/firewalld/services` and the services that are defined or modified by the user are stored in directory `/etc/firewalld/services`. The file name is simply built by adding extension `.xml` to the service name.

Create the service *nfs23* by creating the file `/etc/firewalld/services/nfs23.xml`:

```
sudoedit /etc/firewalld/services/nfs23.xml
```

The file should have the content of Listing 4.

```
<?xml version="1.0" encoding="utf-8"?>
<service>
  <short>NFS2+3</short>
  <description>The NFS V2 and V3 protocol.</description>
  <port protocol="tcp" port="32803"/>
  <port protocol="udp" port="662"/>
  <port protocol="udp" port="2049"/>
  <port protocol="tcp" port="662"/>
  <port protocol="udp" port="875"/>
  <port protocol="tcp" port="875"/>
  <port protocol="udp" port="32769"/>
</service>
```

Listing 4: Fedora's `multiuser.target` unit

After you have stored the file, reload the firewall. This is necessary to make the new service *nfs23* known to firewalld.

```
sudo firewall-cmd --reload
```

Now enable the firewalld services *rpc-bind*, *mountd*, *nfs* and our new *nfs23* in the default zone. Of course you have to do this in the permanent zone set or it will be lost when the firewall is reloaded, e. g. after at the next reboot. Finally also reload the firewall right now to move the permanent zone set to the run-time zone set, which activates the new settings immediately.

```
sudo firewall-cmd --permanent --add-service=rpc-bind
sudo firewall-cmd --permanent --add-service=mountd
sudo firewall-cmd --permanent --add-service=nfs
sudo firewall-cmd --permanent --add-service=nfs23
sudo firewall-cmd -reload
```

8.1.4 Open Firewall For NFS (GUI)

Start `firewall-config` under menu *Services* → *Administration* → *Firewall*. The first step is to define a new firewalld service *nfs23*. Defining new services can only be done in the permanent zone set, therefore select *Permanent* in the drop-down box at the top and *Services* in the left pane (see Figure 15). Now click on the + button below the services.

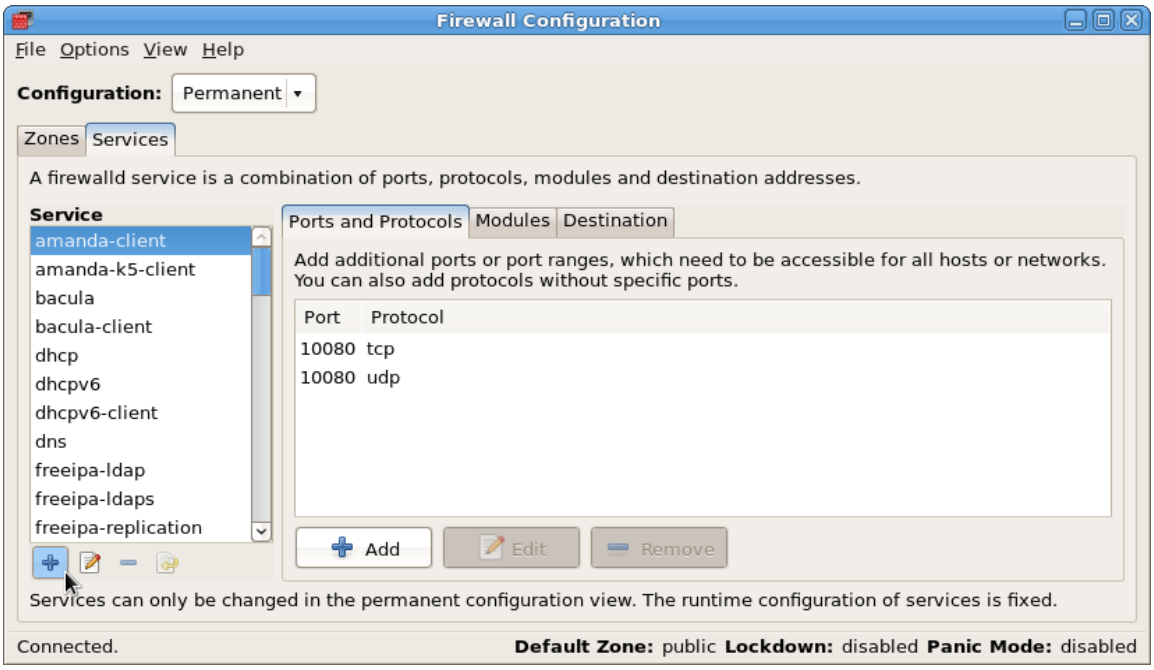


Figure 15: Add a new firewalld service

This will open a small dialog where you can enter the name and description of the new service *nfs23*. Fill in the entries as shown in Figure 16.

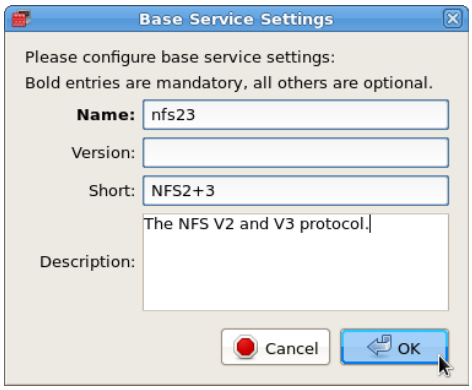


Figure 16: Add name and description for nfs23

When you click on *OK*, the dialog closes and you will find the new firewalld service *nfs23* in the list of services in the left pane now. Click on it. Check that the right side shows the tab *Ports and Protocols*. It should be empty (see Figure 17).

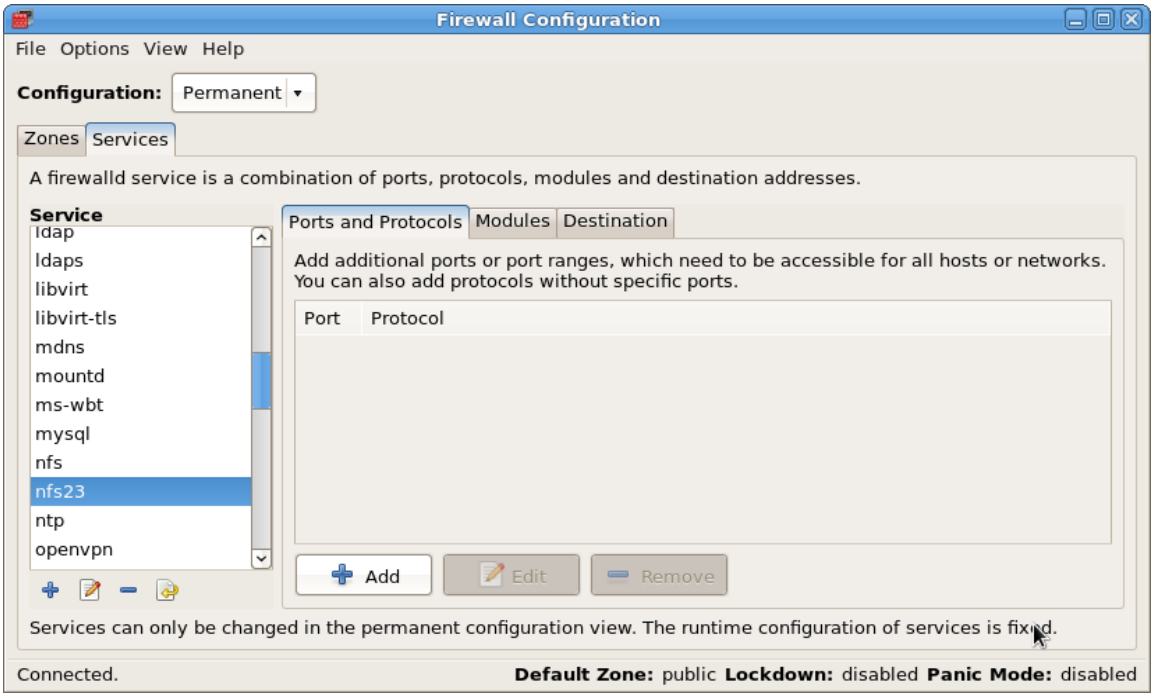


Figure 17: New service nfs23, no ports yet

Click on the *Add* button in the right pane. This opens another small dialog where you can define the port specifications. Enter number *2049* in the *Port* field and select *udp* from the drop-down box in the *Protocol* field (see Figure 18). This will add the UDP port 2049 for NFS. Then clock on *OK*.

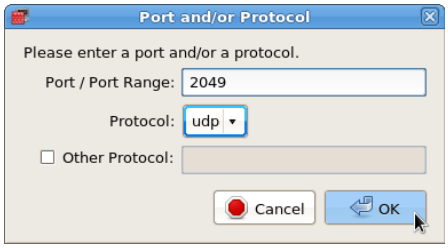


Figure 18: Add port 2049/udp to nfs23

Repeat this step to add the following ports one after the other: 662/tpc, 662/udp, 875/tcp, 875/udp, 32803/tcp, 32769/udp. The result should look like Figure 19.

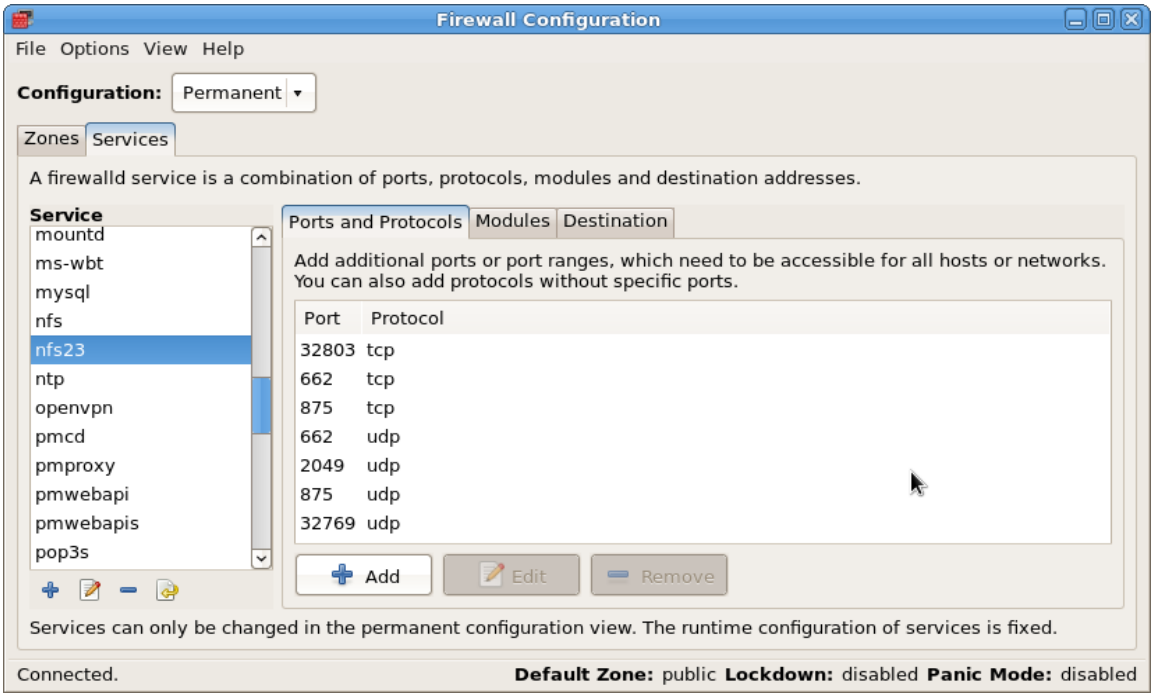


Figure 19: Service nfs23 with ports added

Now reload the firewall with menu *Options* → *Reload FirewallD* to make service *nfs23* known to the zones. Otherwise the new service would not appear in the zones lists.

The next step is to enable the NFS firewalld services. Click on *Zones* in the left pane and select the default zone (shown in bold font, usually “public”). Verify that the right side shows tab *Services*. Now check the services *mountd*, *nfs*, *nfs23* and *rpc-bind* (see Figure 20, for rpc-bind you have to scroll down further).

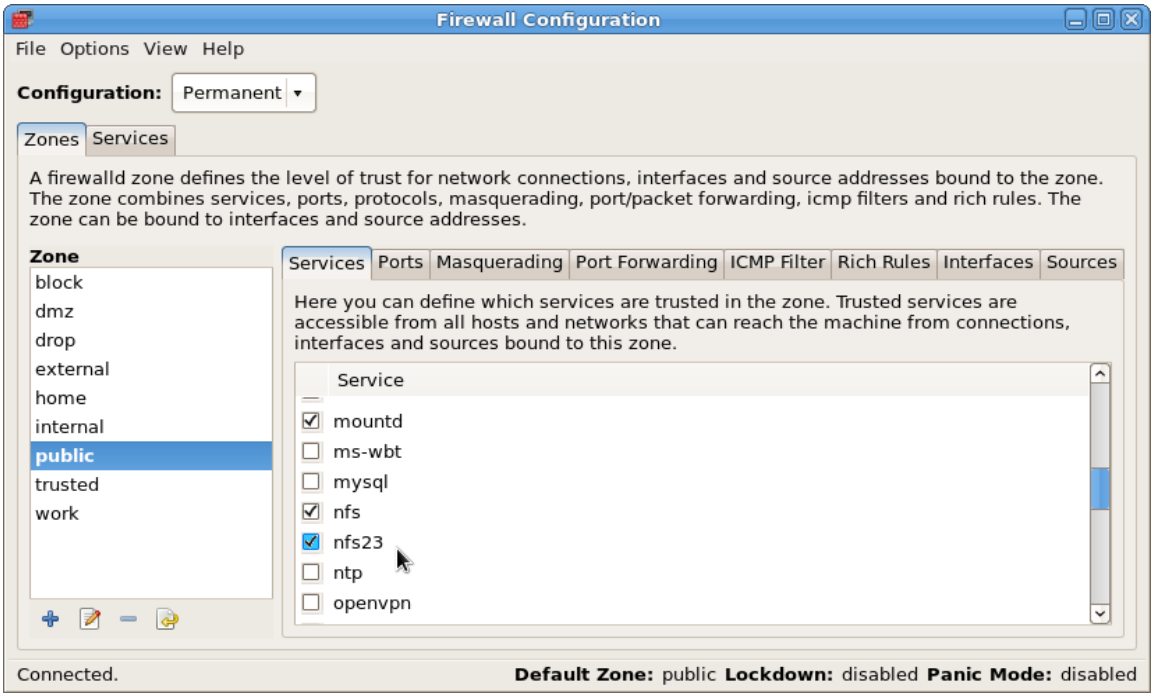


Figure 20: Add NFS services to public zone

That's it. You just need to activate these settings again by reloading the firewall rules with menu *Options → Reload FirewallD*. Then you can close `firewall-config`.

8.1.5 Define Exported Directories (Command Line)

The following commands will export the two directories `/rootfs` and `/download` via NFS because these are used in all F&S examples and documentations. `/rootfs` is meant for exporting a root filesystem and `/download` is meant for some general downloads. If these directories do not exist yet, create them now with:

```
sudo mkdir /rootfs /download
```

The exported directories are configured in file `/etc/exports`. You simply have to edit this file with:

```
sudoedit /etc/exports
```

Then add these two lines to the file:

```
/rootfs     *(rw,sync,no_root_squash)
/download   *(rw,sync,no_root_squash)
```

Both directories will accept connections from all clients, are mounted read/write and the root user is accepted as root. Remember that when you are using a remote NFS root filesystem that all data stored in the root filesystem of your board is actually transferred to the PC and stored in this directory here. This is why we also need write access, otherwise no data could be stored on the board.

8.1.6 Define Exported Directories (GUI)

The following commands will export the two directories `/rootfs` and `/download` via NFS because these are used in all F&S examples and documentations. `/rootfs` is meant for exporting a root filesystem and `/download` is meant for some general downloads. If these directories do not exist yet, create them now with:

```
sudo mkdir /rootfs /download
```

Now start `system-config-nfs` from menu *System* → *Administration* → *System-Config-NFS*. You have to validate yourself with your password to be able to modify any export settings. Then the dialog from Figure 21 will show up.

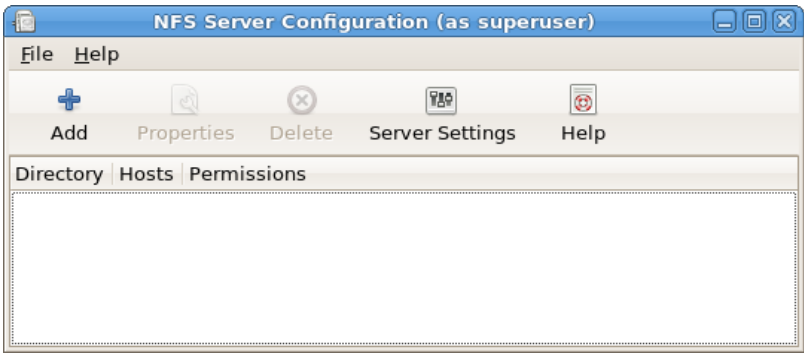


Figure 21: NFS configuration tool

Click on button *Add*. This shows another dialog with three tabs where you can enter the name and parameters of a directory that should be exported via NFS. In the *Basic* tab enter `/rootfs` as directory and `*` to allow all hosts to access this share (see Figure 22). This directory will hold the root filesystem later. You also should activate read/write access or you can not store data on your board. Remember that when you are using a remote NFS root filesystem that all data stored in the root filesystem of the board is actually transferred to the PC and stored in this directory here.

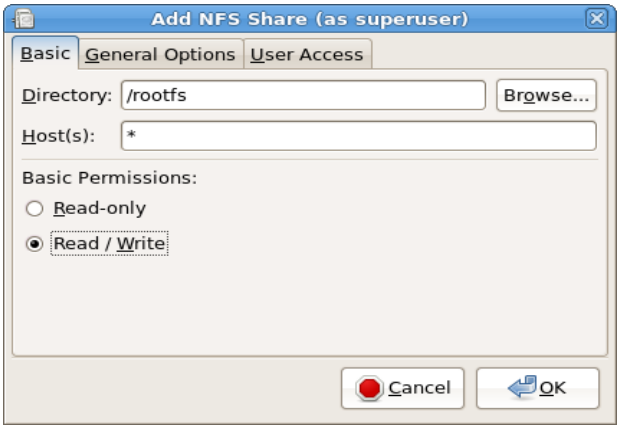


Figure 22: Add NFS share, basic options

In the *General Options* (see Figure 23) tab you can leave all settings as they are.

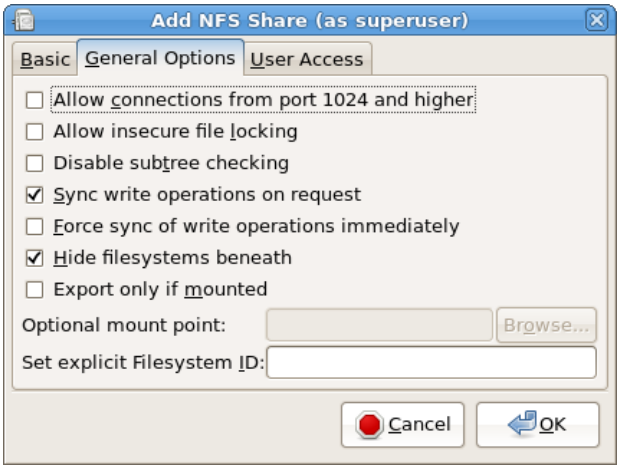


Figure 23: Add NFS share, general options

In the *User Access* tab (see Figure 24) you have to tick “Treat remote root user as local root”. The root filesystem will definitely need root access for booting and running, so root must have access to these files that are owned by root. Now you can close this dialog with *OK*.

Figure 24: Add NFS share, user access options

Back in the main dialog click again on button *Add* and repeat the above steps for directory `/download`.

The final result should look like Figure 25. You are done now and can close the program `system-config-nfs`.

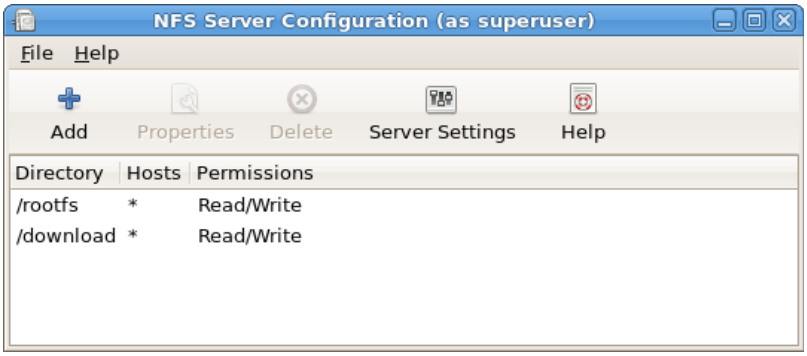


Figure 25: NFS configuration with all shares added

8.1.7 Start NFS Server

Finally enable NFS automatically when booting the system by enabling the `nfs.target` and also start the NFS server right now by starting `nfs-server.service`:

```
sudo systemctl enable nfs.target
sudo systemctl start nfs-server.service
```

The following commands can then be used to start, stop or restart NFS.

Start NFS:

```
sudo systemctl start nfs-server.service
```

Stop NFS:

```
sudo systemctl stop nfs-server.service
```

Restart NFS:

```
sudo systemctl restart nfs-server.service
```

Remark

Because the `nfs-server.service` unit is also available under the name `nfs.service`, you can usually abbreviate `nfs-server.service` with simply `nfs` in the commands above.

You can check if the NFS server is configured correctly by issuing the following command:

```
rpcinfo -p
```

The command lists all available Remote Procedure Calls. The result should look similar to Listing 5. There may be additional entries listed, but you should see all NFS calls using all the given versions.

```
   program vers proto   port  service
    100000    4   tcp    111  portmapper
    100000    3   tcp    111  portmapper
    100000    2   tcp    111  portmapper
    100000    4   udp    111  portmapper
    100000    3   udp    111  portmapper
    100000    2   udp    111  portmapper
    100024    1   udp    662  status
    100024    1   tcp    662  status
    100003    2   tcp   2049  nfs
    100003    3   tcp   2049  nfs
    100003    4   tcp   2049  nfs
    100227    2   tcp   2049  nfs_acl
    100227    3   tcp   2049  nfs_acl
    100003    2   udp   2049  nfs
    100003    3   udp   2049  nfs
    100003    4   udp   2049  nfs
    100227    2   udp   2049  nfs_acl
    100227    3   udp   2049  nfs_acl
    100021    1   udp  32769  nlockmgr
    100021    3   udp  32769  nlockmgr
    100021    4   udp  32769  nlockmgr
    100021    1   tcp  32803  nlockmgr
    100021    3   tcp  32803  nlockmgr
    100021    4   tcp  32803  nlockmgr
    100011    1   udp    875  rquotad
    100011    2   udp    875  rquotad
    100011    1   tcp    875  rquotad
    100011    2   tcp    875  rquotad
    100005    1   udp  20048  mountd
    100005    1   tcp  20048  mountd
    100005    2   udp  20048  mountd
    100005    2   tcp  20048  mountd
    100005    3   udp  20048  mountd
    100005    3   tcp  20048  mountd
```

Listing 5: Output of `rpcinfo -p`

Here you can see the portmapper (`rpcbind`) in versions 2, 3 and 4 on UDP and TCP port 111, status (`rpc.statd`) on UDP and TCP port 662, nfs (`rpc.nfsd`) in versions 2, 3, and 4 on UDP and TCP port 2049, nlockmgr (`rpc.lockd`) in versions 1, 3 and 4 on UDP port 32769 and TCP port 32803, rquotad (`rpc.rquotad`) in versions 1 and 2 on UDP and TCP port 875 and mountd (`rpc.mountd`) in versions 1, 2 and 3 on UDP and TCP port 20048. If you compare this to Table 6 from page 33, then this is exactly what we wanted. In addition we also have nfs_acl in versions 2 and 3 also on UDP and TCP port 2049. This is an additional NFS service to handle Access Control Lists.

8.2 Test NFS Server

Copy a file to the `/download` directory, for example a U-Boot image `uboot.nb0`. The image itself does not matter, we just want to test whether it works at all.

```
sudo cp uboot.nb0 /download
```

Then start your board. Stop in U-Boot by pressing a key while the autoboot time counts down. If not done yet, set the following environment variables (see Table 7).

Variable	Description
ethaddr	MAC address of your board
ipaddr	IP address of your board
serverip	IP address of your server (=PC)
netmask	Mask of your network

Table 7: U-Boot environment variables for NFS

Save these settings with

```
saveenv
```

Now try the NFS download:

```
nfs ${serverip}:/download/uboot.nb0
```

If the file is transferred, everything is OK. If there are any timeouts, something is still wrong.

8.3 Working With Exported Filesystem Images

Often you will have to export filesystems that change on a regular basis, for example because you have built a new version with Buildroot or Yocto. The following section shows how to do this.

Let's assume that you have some file `rootfs.ext3` that holds the filesystem image. First you have to mount this image under directory `/rootfs`.

```
sudo mount -o loop rootfs.ext3 /rootfs
```

Here `-o loop` will use the so-called *loop device*. This means that a file that is already part of a mounted filesystem (your active filesystem of the PC) can in turn be used as a filesystem on its own and be mounted on a new mount point (here `/rootfs`). Or put in other words: if you want to mount a filesystem image from a file instead of from a block device as usual, you have to use option `-o loop`.

After having mounted the filesystem image, you can start the NFS service. Please see the above sections of how to do this. For example on Fedora this is

```
sudo systemctl start nfs
```

After that you can work with the exported filesystem.

Now let's assume you are done with this version and you want to build a new version of the filesystem. You have to unmount the filesystem image first before you can build the new image, because otherwise the file is locked and the build process can not write to it. But when you try to unmount the image, the system will complain that the filesystem is still in use. This is because NFS still may hold open file handles on it. So the first step is to stop the NFS server. Again see the above sections on how to do that. For example on Fedora this is

```
sudo systemctl stop nfs
```

When this is successful, you can unmount the filesystem

```
sudo umount /rootfs
```

Now the filesystem image is free again and you can start building your new version. When this is finished, you can again mount it with the loop device and start the NFS server again.

Remark

You can not mount UBI or UBIFS images via the loop device. UBI images are meant to be stored directly on flash memory and do not use a block device for this. But the loop device can only mount filesystem images that are targeted to block devices. That is EXT2, EXT3, BTRFS, FAT, NTFS and similar.

This is why we usually build at least two different filesystem images in Buildroot and Yocto: A UBIFS image meant to be stored in NAND flash memory on the board and an EXT2 or EXT3 image meant to be exported as NFS root filesystem. Additional images, for example a FAT filesystem to be stored on an SD card, can be enabled optionally.

9 Appendix

List of Figures

List of Tables

Listings

Important Notice

The information in this publication has been carefully checked and is believed to be entirely accurate at the time of publication. F&S Elektronik Systeme assumes no responsibility, however, for possible errors or omissions, or for any consequences resulting from the use of the information contained in this documentation.

F&S Elektronik Systeme reserves the right to make changes in its products or product specifications or product documentation with the intent to improve function or design at any time and without notice and is not required to update this documentation to reflect such changes.

F&S Elektronik Systeme makes no warranty or guarantee regarding the suitability of its products for any particular purpose, nor does F&S Elektronik Systeme assume any liability arising out of the documentation or use of any product and specifically disclaims any and all liability, including without limitation any consequential or incidental damages.

Products are not designed, intended, or authorised for use as components in systems intended for applications intended to support or sustain life, or for any other application in which the failure of the product from F&S Elektronik Systeme could create a situation where personal injury or death may occur. Should the Buyer purchase or use a F&S Elektronik Systeme product for any such unintended or unauthorised application, the Buyer shall indemnify and hold F&S Elektronik Systeme and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, expenses, and reasonable attorney fees arising out of, either directly or indirectly, any claim of personal injury or death that may be associated with such unintended or unauthorised use, even if such claim alleges that F&S Elektronik Systeme was negligent regarding the design or manufacture of said product.