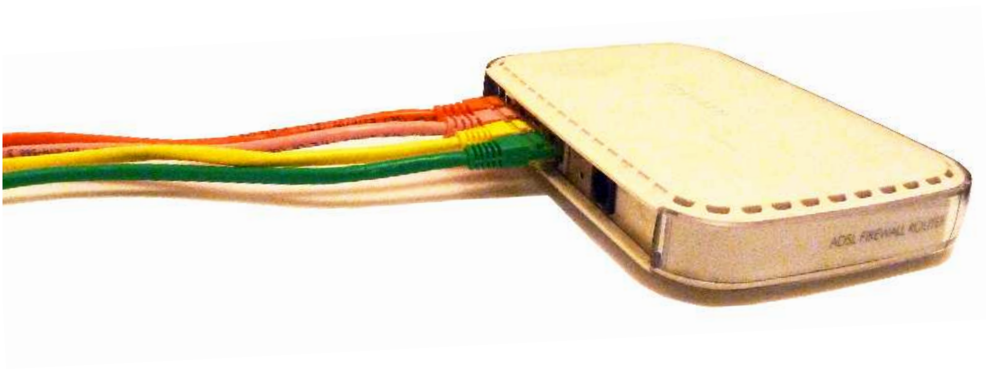
The WROCC Guide to Networking

Part 2 **£3.00**

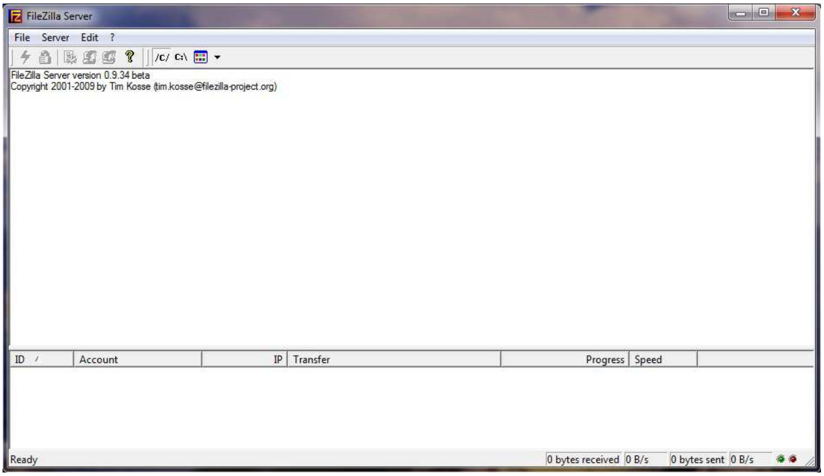
Contents



File Transfer Protocol with Windows 7

September 2010

By Chris Bass – chris.bass@havengym.org.uk



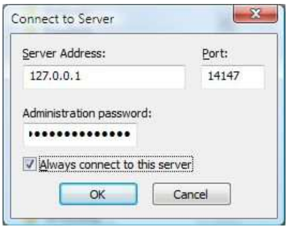
FileZilla Server version 0.9.34 beta
Copyright 2001-2009 by Tim Kosse (tim.kosse@filezilla-project.org)

Figure 2: The main FileZilla window

In the past I have been successfully using LanMan98 on both a RiscPC and an Iyonix to access Windows XP files, but have been unable to do the same with Windows 7. I came across an article in *Archive* 21.1 on how to transfer files between a RiscPC and Vista using FTP by Philip Ingram, so I decided to try this on Windows 7.

FTP is an open protocol widely used to distribute files, although it lacks good security (others can easily break in) and solid integrity (you can't be sure that what you receive is what was sent). This security failing shouldn't be a problem on a private network, and the integrity failing should also not cause trouble on a sensible domestic network.

FTP requires a server and a client. The data normally lives on the machine where the server software runs and is accessed using the client software, which normally runs on a different machine. In the rest of this article we shall see how to set up a server on a Windows 7 PC and access the data using a client running under RISC OS.



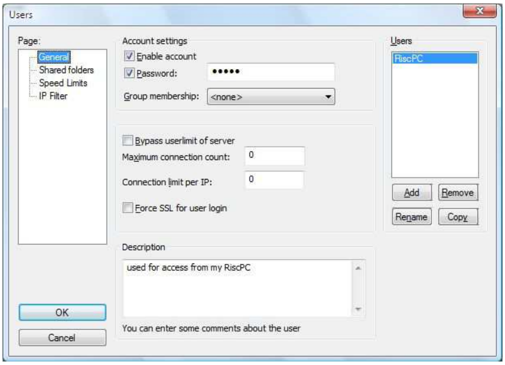
Server Address: 127.0.0.1
Port: 14147
Administration password: ••••••••••••••
☑ Always connect to this server

Figure 1: Connecting to the FileZilla server

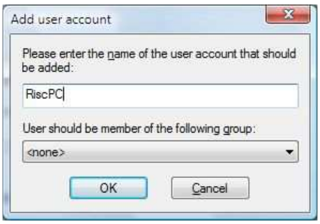
Installing an FTP server on Windows 7

Many implementations of FTP servers are available for Windows and FileZilla is an open-source server that seems to work acceptably. It is a 1.7MB download from `sourceforge.net/projects/filezilla` – follow the 'View All Files' link, and scroll down the page to select the latest FileZilla Server for Windows XP/Vista. At the time of writing, the server release is 0.9.36.

Download it and put it somewhere where you can find it, such as ***Public\Public Downloads***, then double-click on the downloaded file to start the installation. Windows 7's security system will then tell you that "An unidentified program wants access to your computer". Click *Allow*, then click *I agree* to the GNU General Public Licence (assuming you do agree!). Accept the default components, the default location for the program, the default for how the server should be started, the default for how



Account settings
[✓] Enable account
[✓] Password: •••••
Group membership: <none>
[ ] Bypass userlimit of server
Maximum connection count: 0
Connection limit per IP: 0
[ ] Force SSL for user login
Description
used for access from my RiscPC
You can enter some comments about the user
Users: RiscPC



Please enter the name of the user account that should be added:
RiscPC
User should be member of the following group:
<none>

Figures 3 & 4: Creating and configuring the RiscPC's account

the interface should be started and you should see "Setup was completed successfully".

Before moving on, it is worth noting that the defaults set up the server to run when Windows is running, which is probably how you would use a system of sharing files. If you don't think that is appropriate, alter the options to suit yourself. You might also prefer not to run the server interface, as it starts up very visibly every time you restart the PC.

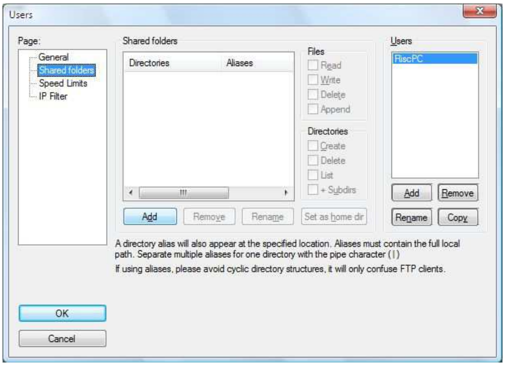
Configuring FTP on Windows 7

By default, the installation process will end and immediately start the server so that it can be configured. A "Connect to Server" window will be displayed as in figure 1. Accept the default Server Address and Port, then enter the password for an administrator account on your Windows 7 PC and select "Always connect to this server". Click *OK*. (The password is only required if the PC administrator account has been set up with one.)

The server then starts and displays a window as in figure 2 with a menu containing *File*, *Server* and *Edit*. Select *Edit – Users* to see a window as in figure 3: the entry *General* should be highlighted. Click *Add* under *Users*, which will open an Add user account window as in figure 4.

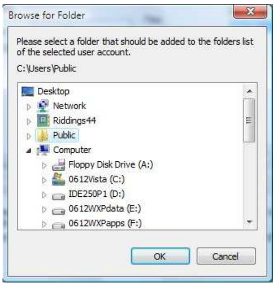
I added a user called "RiscPC" with no group membership. Click *OK* and return to the Users screen where you can check a tickbox to add a password and enter a comment about the user as in figure 3. This will also work if you have an Iyonix – just add another user and call it Iyonix.

After setting up a user you need to give that user access to some data. I suggest granting access to the ***Public*** folder, as this is the way Windows 7 expects to be used. Click on *Shared folders* and the *Add* under *Shared folders* as in figure 5.



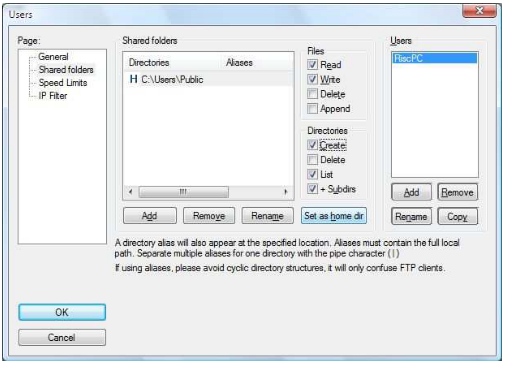
A directory alias will also appear at the specified location. Aliases must contain the full local path. Separate multiple aliases for one directory with the pipe character (|)
If using aliases, please avoid cyclic directory structures, it will only confuse FTP clients.

Figures 5 & 6: Setting the shared folder



Please select a folder that should be added to the folders list of the selected user account.
C:\Users\Public

This button will lead to a window where you can browse for a folder. Figure 6 shows the ***Public*** folder selected. Click *OK*, which returns you to a window that lets you set the access control for the shared folder. I opted to allow reading and writing of files and creation and listing of directories and their subdirectories as in figure 7 (over the page). Click on *Set as home dir* before clicking *OK*.


Users

Page:
- General
- Shared folders
- Speed Limits
- IP Filter

Shared folders

Directories	Aliases
H C:\Users\Public	

Files
- [✓] Read
- [✓] Write
- [ ] Delete
- [ ] Append

Directories
- [✓] Create
- [ ] Delete
- [✓] List
- [✓] + Subdirs

Users
- RiscPC

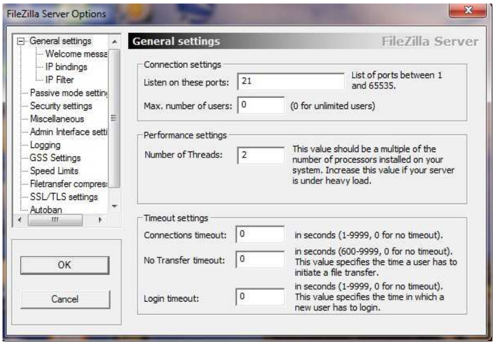
A directory alias will also appear at the specified location. Aliases must contain the full local path. Separate multiple aliases for one directory with the pipe character (|)

If using aliases, please avoid cyclic directory structures, it will only confuse FTP clients.

Figure 7: The shared folder set

The "H" to the left of ***C:\Users\Public*** in figure 7 indicates that it is the home directory.

At this point the server is usable but is configured in a way more suitable for a public fileserver than for a home network: connections will time out if not used for 120 seconds. To change this, follow *Edit – Settings* and select *General Settings* if this is not displayed. Change the three timeout settings in the lower right of the panel to zero, which will stop the server from timing out any connection (see figure 8 below).


FileZilla Server Options

- General settings
  - Welcome messa
  - IP bindings
  - IP Filter
- Passive mode settin
- Security settings
- Miscellaneous
- Admin Interface setti
- Logging
- GSS Settings
- Speed Limits
- Filetransfer compres
- SSL/TLS settings
- Autoban

General settings — FileZilla Server

Connection settings
- Listen on these ports: 21 — List of ports between 1 and 65535.
- Max. number of users: 0 — (0 for unlimited users)

Performance settings
- Number of Threads: 2 — This value should be a multiple of the number of processors installed on your system. Increase this value if your server is under heavy load.

Timeout settings
- Connections timeout: 0 — in seconds (1-9999, 0 for no timeout).
- No Transfer timeout: 0 — in seconds (600-9999, 0 for no timeout). This value specifies the time a user has to initiate a file transfer.
- Login timeout: 0 — in seconds (1-9999, 0 for no timeout). This value specifies the time in which a new user has to login.

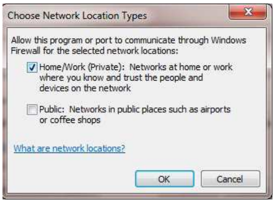
Figure 8: Timeout settings for a home network

The server configuration is now complete, but you won't be able to access it without telling your firewall how to treat access to the server.

Firewall settings

The following settings are for Windows 7's built-in firewall: if you have a different firewall you probably know enough to set it up yourself. Follow *Start* (the round, unlabelled button on the taskbar at the extreme bottom left of the Windows 7 screen) – *Control Panel – Windows Firewall* and then click on *Allow a program or feature through Windows firewall* near the upper left corner of the window. A window opens: see figure 9 opposite. Click on the *Change Settings* button top right of this window, then on *Allow another Program*. This opens another window, in figure 10.

Click on *Browse* to tell the Firewall about the FileZilla Server program. For most people, it will be in ***C:\Program Files\FileZilla Server*** as in figure 11. If your system is different, you will probably know where you installed it; if not, use the search in Windows Explorer to find it. Once located, click on *Open* and FileZilla Server should now appear in the list of programs. Click on it so that it is highlighted, then click on *Network Location Types* to open the dialogue shown here.

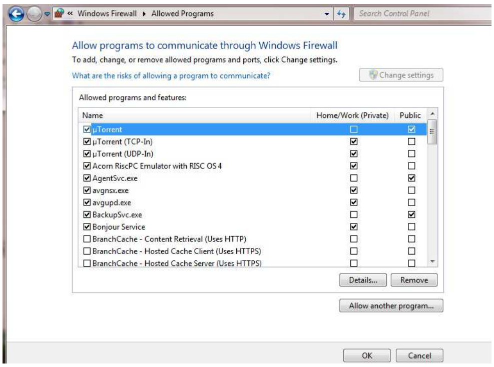

Choose Network Location Types

Allow this program or port to communicate through Windows Firewall for the selected network locations:

- [✓] Home/Work (Private): Networks at home or work where you know and trust the people and devices on the network
- [ ] Public: Networks in public places such as airports or coffee shops

Select *Home/Work (Private)* and click *OK* and then exit from the firewall setup by clicking *OK*. This should ensure that no-one outside your private network can access the FTP server; if you have a different requirement, that is beyond the scope of this article. The server is now configured and ready for use.

Before leaving the Windows 7 machine, you need to find the network name and the IP address of your PC. Follow *Control panel – Network and Sharing Centre*. You should see the name of your computer just above "(This computer)". Then click on *Change Adapter settings* situated on the left. Click on appropriate network card to highlight it, right-click and select *Properties*. Click on *Internet version 4 (TCP/IPv4)* to find your IPv4 address. FTPc should be quite happy with



« Windows Firewall ▸ Allowed Programs

Search Control Panel

Allow programs to communicate through Windows Firewall

To add, change, or remove allowed programs and ports, click Change settings.

What are the risks of allowing a program to communicate?

Change settings

Allowed programs and features:

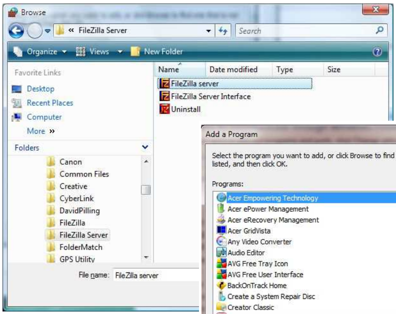
Name	Home/Work (Private)	Public
☑ µTorrent	☐	☑
☑ µTorrent (TCP-In)	☑	☐
☑ µTorrent (UDP-In)	☑	☐
☑ Acorn RiscPC Emulator with RISC OS 4	☑	☐
☑ AgentSvc.exe	☐	☑
☑ avgnsx.exe	☑	☐
☑ avgupd.exe	☑	☐
☑ BackupSvc.exe	☐	☑
☑ Bonjour Service	☑	☐
☐ BranchCache - Content Retrieval (Uses HTTP)	☐	☐
☐ BranchCache - Hosted Cache Client (Uses HTTPS)	☐	☐
☐ BranchCache - Hosted Cache Server (Uses HTTPS)	☐	☐

Details... Remove

Allow another program...

OK Cancel

Figure 9: The programs allowed through the Windows 7 Firewall



Browse

« FileZilla Server

Search

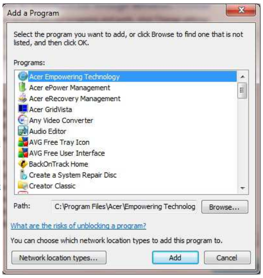
Organize ▾ Views ▾ New Folder

Favorite Links: Desktop, Recent Places, Computer, More »

Folders: Canon, Common Files, Creative, CyberLink, DavidPilling, FileZilla, FileZilla Server, FolderMatch, GPS Utility

Name	Date modified	Type	Size
FileZilla server			
FileZilla Server Interface			
Uninstall			

File name: FileZilla server



Add a Program

Select the program you want to add, or click Browse to find one that is not listed, and then click OK.

Programs:

- Acer Empowering Technology
- Acer ePower Management
- Acer eRecovery Management
- Acer GridVista
- Any Video Converter
- Audio Editor
- AVG Free Tray Icon
- AVG Free User Interface
- BackOnTrack Home
- Create a System Repair Disc
- Creator Classic

Path: C:\Program Files\Acer\Empowering Technolog Browse...

What are the risks of unblocking a program?

You can choose which network location types to add this program to.

Network location types... Add Cancel

Figures 10 & 11: Adding FileZilla to the list of allowed programs

either the name or the IP address, but I found the IP address to be more reliable! Depending on your setup, you may need to add the name and IP address of the Windows 7 machine to your RISC OS Hosts file, as described in issue 27.9 of *The WROCC*.

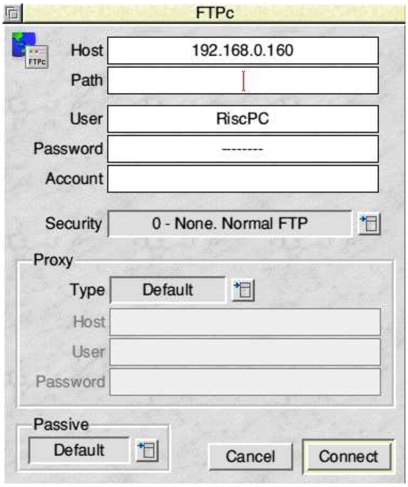
Setting up the FTP client

Armed with the IP address from above and the user name and password from figure 3, we can configure the FTP client. The example uses FTPc 1.44 – the current version – which is available from `www.ftpc.iconbar.com`. Unzip the archive and copy ***!FTPc*** to a convenient place (rather a contrast from the server install!). Double-click on the application to load it to the iconbar, then click on the FTPc icon on the iconbar to see a window such as in figure 12.

Enter your Windows 7 system's name or IP address in the Host box and the username and password as set in figure 5. Click *Connect* and you should see something like figure 13. If you are using the ***Public*** folder as in this example, you might find that the folders seen by FTPc have names slightly different from those seen by Windows 7 – for the ***Public*** folder is smoke and mirrors between you and the folders that really hold the files.

That's it! You should now be able to transfer files both ways between RISC OS and Windows 7.

You will probably want to set up the Windows 7 link in the user menu of FTPc. This

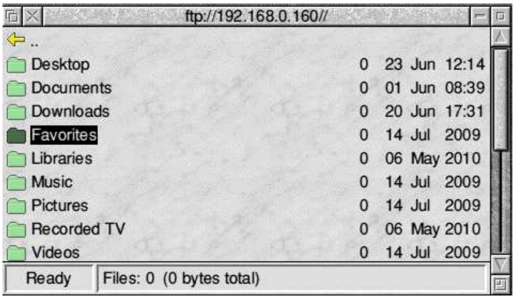


FTPc

Host: 192.168.0.160
Path:
User: RiscPC
Password: --------
Account:
Security: 0 - None. Normal FTP
Proxy
Type: Default
Host:
User:
Password:
Passive: Default

Cancel Connect

Figure 12: Setting up FTPc under RISC OS



ftp://192.168.0.160//

```
..
Desktop        0  23 Jun  12:14
Documents      0  01 Jun  08:39
Downloads      0  20 Jun  17:31
Favorites      0  14 Jul  2009
Libraries      0  06 May 2010
Music          0  14 Jul  2009
Pictures       0  14 Jul  2009
Recorded TV    0  06 May 2010
Videos         0  14 Jul  2009
Ready   Files: 0  (0 bytes total)
```

Figure 13: The Windows 7 files as seen in FTPc

is most easily done by clicking Menu in the FTPc window seen in figure 13 on the previous page and selecting *Add to user menu* to open the window shown in figure 14; after inserting a suitable name, click the *Add* button.

If you are using the Windows 7 machine to back up your RISC OS system, you may wish to tick the *Add/remove ,hex filetype* option as this will preserve RISC OS filetypes. For example, a ***!Run*** obeyfile will become ***!Run,feb*** in the Windows 7 folder. When copied back, it will lose the ***,feb*** suffix and become an obeyfile again.

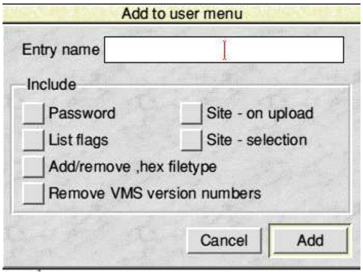


Figure 14: FTPc allows entries to be added to its User Menu, to remember their settings

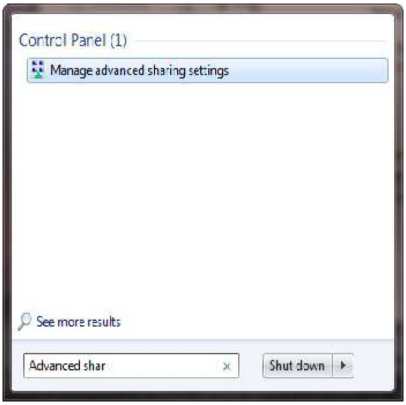
Using LanMan98 with Windows 7

December 2010

By Chris Hughes – chris@wrocc.org.uk

In his article in the September 2010 issue of *The WROCC*, Chris Bass indicated that he had so far been unable to access shared folders on Windows 7 using LanMan. In fact you can, and the process is actually fairly painless.

I am currently using LanMan98 version 2.01 with Windows 7 Home Premium, and it works when you have made the necessary changes on the Windows 7 PC – there are one or two things that you need to remember. One is that by default, Windows 7 has higher security than XP did: by default it uses 128-bit encryption, whereas we can only currently use 56-bit at best on RISC OS. The second issue is that unlike on XP, you can't normally share an entire drive any more – for fairly obvious security reasons, when you think about it (there are ways around this, which are not in this article).



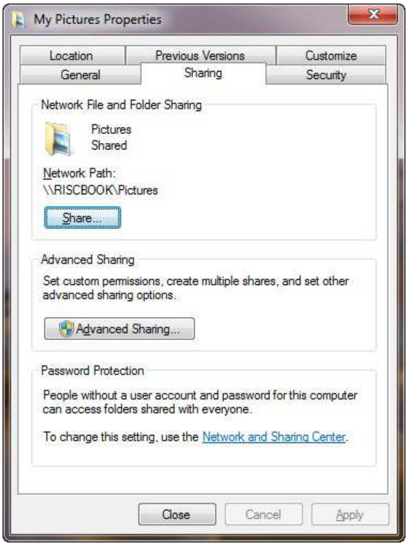
Searching for the Advanced Sharing Settings options from the start menu

Setting the encryption level

The first thing to do is to change the encryption level on the PC side of things. To do this, go to *Start* and in the search box, type 'Advanced Sharing' – the first entry under 'Control Panel' in the list that is displayed as part of the instant search is 'Manage Advanced Sharing Settings'. Select this option and a new window opens: this is the main configuration window for sharing across a network.

Expand the *Home or Work (current profile)* option, and ensure that the following changes have been made:

- *Turn on network discovery*
- *Turn on file and printer sharing*
- *Turn on sharing so anyone with network access can read and write files in the Public folders*
- *Enable file sharing for devices that use 40- or 56-bit encryption*
- *Turn off password protected sharing* (this can be on if you want to use passwords – it's up to you).


My Pictures Properties

Location | Previous Versions | Customize
General | Sharing | Security

Network File and Folder Sharing

Pictures
Shared

Network Path:
\\RISCBOOK\Pictures

Share...

Advanced Sharing

Set custom permissions, create multiple shares, and set other advanced sharing options.

Advanced Sharing...

Password Protection

People without a user account and password for this computer can access folders shared with everyone.

To change this setting, use the Network and Sharing Center.

Close | Cancel | Apply

Setting the share options for My Pictures

Then click on *Save Changes* and this will save the settings.

Share some files

The next step it to turn on file sharing for your folders, so go to the folder you want to share. Bear in mind that if it's one of the new Windows 7 'Libraries' these are just a 'dummy folder' view on to a group of 'real folders' which make up the library: therefore you can't share the library, but only the individual folders within it. For reference, the default folders for ***Documents***, ***Music***, ***Pictures*** and so on are in fact located in ***C:\Users\username***\ where 'username' is your username on the machine.

I will use the ***My Pictures*** folder as the example: navigate to the ***My Pictures*** folder within the ***C:\Users\username***\ structure. Right-click the ***My Pictures*** folder icon and go to *Properties*; select the *Sharing* tab as in the screenshot on the right. If the folder has not yet been turned on for sharing then you need to click on the *Share...* button, then click on *Share* and on *Done*: the folder is made available as a shared folder.

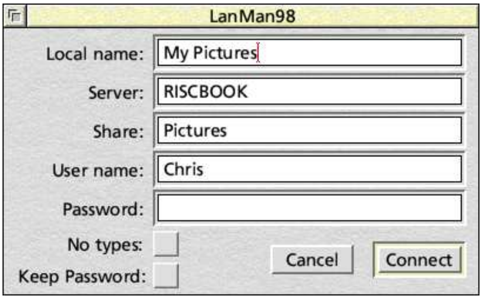
However, you still need to ensure the actual permissions are turned on. Click on the *Share...* button again, but this time click on the little down arrow that is in the window and a list of the available groups appear to allow sharing rights. I suggest selecting 'Everyone': click on *Add* at this point, and it now appears in the main list in the window – probably with a permission of 'Read'. Click on the little down arrow beside the word 'Read' and select 'Read/Write' instead, then click on the *Share* button again; you should now be able to access this folder across the network with LanMan98 as normal.

Setting up LanMan98

To set up the connection at the RISC OS end, load LanMan98, then click Menu over its icon and select *New*. In the window that opens, you need to fill the first four fields (as shown in the example on the right opposite).

- The *Local name* is whatever you want to call the connection, and appears under the RISC OS iconbar icon when connected.
- *Server* is the PC's *Computer Name*, and is set in the *System* section of the Windows control panel: it is 'RISCBOOK' in my example.
- *Share* is the name of the share on the PC: unless you set it otherwise (from *Advanced Sharing...* in the folder properties dialogue on the left) it is the name of the shared folder.
- *User name* is your username on the PC.

Now click on *Connect* and it should connect without errors: if it does, then click on the new icon that has appeared on the iconbar, and hopefully you will now have access to your files on the Windows 7 PC.


LanMan98

Local name: My Pictures
Server: RISCBOOK
Share: Pictures
User name: Chris
Password:
No types:
Keep Password:

Cancel | Connect

Setting up LanMan98 at the RISC OS end

Firewalling RISC OS

Peter Richmond – peter@wrocc.org.uk

Steve Potts was presenting the October meeting, and in order to more easily understand the way network traffic travels between client and server, had arranged for me to bring my video/data projector to enable him to show the two computer screens side by side. One screen was showing the client and the other was showing the server. Steve had cunningly made backdrops for these two computer screens with the words 'client' and 'server' on them just to help the audience out.

Steve was using two A9home computers to show the networking firewall in operation, and mentioned that the firewall configuration tool he was using was from the latest RISC OS 6 build, which he had copied over to run on the A9homes as they cannot currently run a full version of RISC OS 6.

Firstly, Steve enquired how many people had some sort of home network – about half of the attendees did, so Steve carried on, assuming some sort of baseline for understanding the terminology of network messages. Steve then mentioned that although there was a newer IP address standard (IPv6) slowly being adopted in the wider computing world, that he would be staying with the more widely used version IPv4, because RISC OS and many consumer networking devices don't yet support IPv6 and many people are unfamiliar with its very different notation.

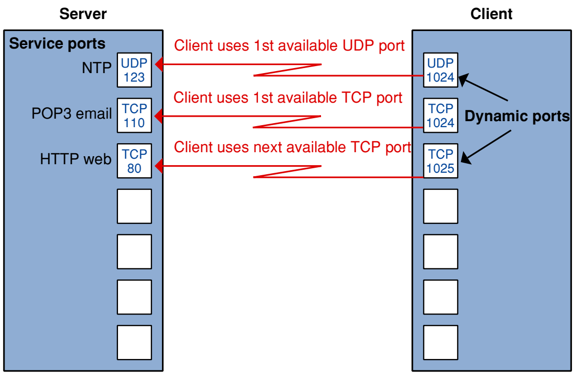
Different protocols

In TCP/IP networking there are three main types of protocol, TCP, UDP and ICMP. Steve pointed this out because firewalls treat them all differently when processing rules.

With the use of a simple diagram (shown above), Steve illustrated that TCP/IP servers use 'well-known' port numbers for common services so that clients know how to connect. Steve's example showed that email (POP3) servers use TCP port 110, web servers use TCP port 80. The same diagram showed that clients usually use dynamic outbound ports, and so POP3 and HTTP clients will be connected via the next free port in the range allowed for such use, for example TCP 1024, 1025 respectively in Steve's example. The diagram also illustrated that, although both TCP and UDP make use of port numbers, they are separate ranges, so TCP 80 is not the same as UDP 80 for example.

After the introductory slides, Steve moved to practical demonstrations. First he tested the link between the two machines using a task window and the Ping tool. Additionally, WebJames was used as the Web Server. Igor was used as a remote command console on the server and Steve's own VNCServer front end was also running the RISC OS VNCServer module. NetSurf, Nettle (Telnet), and Avalanche were used respectively on the client to connect to those services.

Configuration of the firewall on RISC OS 6 is much more easily understandable than on previous versions of RISC OS (it was first introduced with RISC OS 4.29). This is mainly because there is now a Configuration Plugin with standard settings and menu options, rather than the previous 'editing a text file' approach.


Server
Client
Service ports
NTP
UDP 123
POP3 email
TCP 110
HTTP web
TCP 80
Client uses 1st available UDP port
Client uses 1st available TCP port
Client uses next available TCP port
UDP 1024
TCP 1024
TCP 1025
Dynamic ports

Typical network connections between client and server

Rules of entry

The way the Firewall works is that all traffic is denied when the firewall is active, if no rules are applied. The plugin provides a comprehensive list of standard rules, which can be used as-is or adapted as required. Steve suggested that one approach to building rules was to use the 'Allow all outbound' rule, and then look at setting up some 'inbound' rules.

The rule editing window gives hints about some inappropriate combinations of settings, and reminds you of standard subnet masks which are in common use. Interactive help is available on all of these menus and values, to further assist in constructing rules.

Having demonstrated creation of simple rules, Steve then went on to remind us that it is easy to have firewalls block traffic, but that you can make some complicated, and if you're not careful, contradictory rules.

Next came actual examples of the firewall blocking and allowing particular traffic into and out of the computers. To demonstrate web browsing, Steve set the 'allow all outbound' rule on the client computer; he then only needed to allow the web server's response to arrive by opening the firewall to traffic originating from port 80 (the web server).

To help people get started using the firewall, the configuration program supplied in RISC OS 6 provides a *Default* button, that allows things to happen with regard to DNS, web browsing and local traffic. Additions to this standard list may be POP3 for email.

Although a router gives a firewall, adding the RISC OS firewall will often help further protect your computer. Having at least one of the two is important: as an example, Steve said that if you are using Samba (the Windows disc sharing protocol), you'd need to have a firewall of some sort because otherwise people could look at your shared drive from the internet.

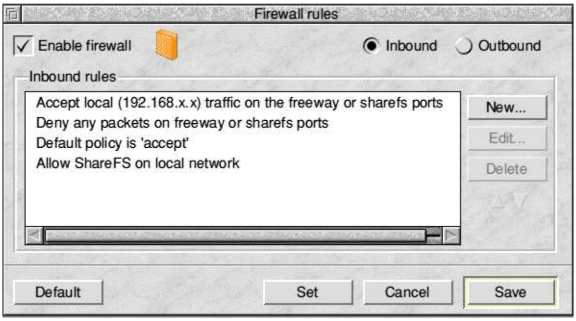
Steve then pointed out (and demonstrated) that the order in which the rules are applied to the firewall is important, and that these rules can be moved up and down the list with regard to their order of being carried out. Steve also pointed out that there may be some overlap on some of these rules with regard to filtering criteria. Normally you would apply very specific rules first, followed by the more general rules. Whenever the firewall is able to match a rule against packets of data, it stops looking for other rules to match, so the most important rules must be nearer the top of the list.

Wireless security

There were then some questions about wireless interfaces, where Steve was clear that you would be more likely need to have a software firewall enabled on wireless networks, because if hackers manage to get into your wireless network, then a router firewall is useless because they would already be on the inside of that one. Some wireless encryption can apparently be hacked within around ten minutes using the right software tools.

If you have more than one network interface in your machine, the RISC OS firewall allows you to make different rules for each interface. This is probably more complex than most users need, but the facility is there if required.

The surprising point about Steve's talk was that, apart from RISC OS 6 itself, all the other programs were public domain!


Firewall rules

✓ Enable firewall

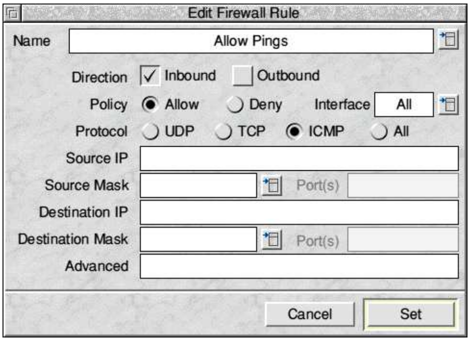
● Inbound ○ Outbound

Inbound rules

Accept local (192.168.x.x) traffic on the freeway or sharefs ports
Deny any packets on freeway or sharefs ports
Default policy is 'accept'
Allow ShareFS on local network

Firewall rules are set in Configure, using the firewall dialogue to set inbound and outbound restrictions

At the end of the evening, everyone had a much clearer idea of the usefulness of firewalls – this being greatly aided by the dual screen method of presentation that Steve used.


Edit Firewall Rule

Name	Allow Pings
Direction	[✓ Inbound] Outbound
Policy	[✓ Allow] Deny
Interface	All
Protocol	UDP TCP [✓ ICMP] All
Source IP	
Source Mask	
Port(s)	
Destination IP	
Destination Mask	
Port(s)	
Advanced	

Setting a rule to allow inbound pings

Using the Firewall

Ian Macfarlane – ian@maceng.karoo.co.uk

Steve Potts began his talk by asking how many of those present had a home network with more than one machine. There were a few members, including your writer. He explained about the network protocol standard, TCP/IP, which had two versions, IPv4 and IPv6. Steve said he would only cover aspects of IPv4 in his talk, since IPv6 was not available on RISC OS. TCP/IP allowed communication between machines and was the basis of the internet. Data was transferred in packets, which could be of three basic types: TCP, UDP and ICMP. Applications on the internet used port numbers of the correct packet type to signify that they owned the data. A computer's firewall would only allow transfers between known port numbers. If an application is running on two computers it will use its own allocated port number to transfer the data between the machines. Each machine will have its own unique address, but use the port number associated with the type of transfer.

Network topology

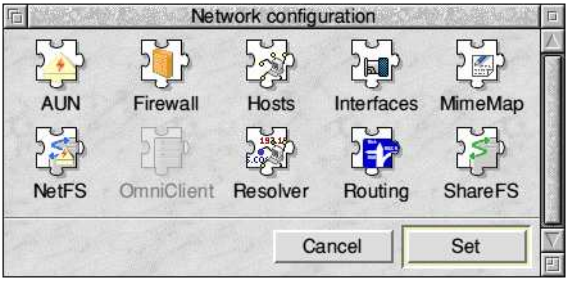
Steve described a typical network. There was a server, which may have several ports, one for each application. There may be an e-mail server with a specific port number and a web server with a different port number. The software on the client, which was requesting the service from the server, would allocate the next free TCP or UDP port to make an out-going connection of the same type; this was called dynamic selection. For example if the client wanted to make a web connection it would make sure that it was talking to port 80 on the server as this is the allocated number for web service, but would use its next available port for the out-going request.

The terms '*client*' and '*server*' apply not to a machine, but to the process. Whichever machine is asking for data, it is acting in the role of a *client* for that data transmission and the machine supplying the information is acting as the *server*. Since there are generally many of these processes going on at the same time, a computer could be acting as a client for one data transmission and a server for another.

Steve then showed a simple firewall working on his computer, which was connected to another computer, not the internet, in order to keep it simple. He used the Ping facility to demonstrate that they were communicating with each other using the ICMP protocol, which doesn't use ports; ICMP is used for sending error messages round the network. The TCP protocol is used for conversational data transfers and the UDP protocol for sending data one way. He went on to demonstrate how the web-server (WebJames) worked on one of the computers and then the Telnet-server (Igor); with this he was able to open a filter window on the other machine.

Breaking the connection

He had thus established applications working across a link and then demonstrated how the



The firewall can be accessed from the Network configuration on RISC OS 6

firewall in RISC OS could be used to prevent these data transfers. In RISC OS 4.29 the firewall could be manipulated by command lines, but there is now a front end application in RISC OS 6.20 which makes it easy to configure the rules of the firewall: Steve altered a rule to show what effect it had on the transmission, using the Ping facility. He explained why the rules for both inbound and outbound transfers in the firewall existed and was able to demonstrate these rules working. There exist predefined rules for RISC OS, with interactive help; its firewall type is known as a packet filter; it is not a full firewall. Steve said that in order to set up and test these rules, it was easiest to permit all outbound transfers and work on the rules for inbound transfers: you could more easily see what was happening; the rules sometimes can contradict each other.

Steve then examined the default rules for the RISC OS firewall. He made the point about the order that the rules occurred in the list, explaining that the first rule took precedence and so on down the list. In general, specific rules should be placed at the top of the list to be effective. In answer to a question from the floor, Steve said he was not aware that RISC OS 5 had a firewall [*it doesn't*]. Another questioner asked where the firewall lived, and Steve was able to show the slot in the RISC OS boot sequence where the firewall was started up, if invoked. He also showed us the port numbers and what they were allocated for – in a file that should never be amended!

When to use it?

I asked Steve on what occasion would a RISC OS user want to alter these rules. Steve said that it would be necessary to stop certain traffic, if that traffic was becoming a problem. He didn't think that there was much, if any, malpractice on RISC OS. He thought that the firewall was disabled by default. When Steve visits clubs and uses his computer on their networks, he does switch his firewall on. He said that in most situations the router's firewall protected the local area network from external attacks and the computer's firewall was not required. However Steve did make the point that it was possible for someone to be parked outside your home and hack into your network via the wireless link that many routers possess and so get behind your router's firewall. The type of wireless key was very important. The WEP type key was not secure enough and he recommended using the latest WPA2 (Wireless Protection Access) encryption.

Peter Richmond asked about WebJames and the VNC server and Steve said that they were freely available. The Flying Pig's website was also worth a look at. To help with the formation of the firewall rules, Steve said that there was a command available, called *netstat* on Windows and *inetstat* on RISC OS. This details what the TCP/IP stack is doing at the present time and he talked around these interesting commands. Steve felt that he was becoming too technical and so stopped his talk there.

Useful links

Avalanche: `effarig.co.uk/riscos`
Igor: `www.flypig.co.uk/?page=riscos&dnload=risc%20os`
VNC Server: `http://www.steve-potts.me.uk/software.html`
Nettle: `nettle.sourceforge.net`
NetSurf: `www.netsurf-browser.org`

Connecting With NFS

by Steve Fryatt – stevef@wrocc.org.uk

So far, Chris Hughes and Chris Bass have looked at the use of Samba (LanMan98) and FTP for sharing files with Windows computers. For those with Linux or Mac systems, or who are just after an alternative to ShareFS for inter-RISC OS transfers, there's an alternative. The Network Filing System (or NFS) is straightforward to set up, comes as standard on Linux-based machines and – best of all – the RISC OS software is free!

RISC OS support comes courtesy of Alex Waugh, in the form of Sunfish and Moonfish. Sunfish is an NFS client, and will read files stored on another machine (assuming it has an NFS server present); Moonfish is an NFS server, and will make files from the machine it's installed on available over the network to other computers. If you have two RISC OS boxes, Sunfish will happily talk to Moonfish.

In the family

For simplicity, we'll start by looking at how to set up both Sunfish and Moonfish to talk to each other. This can actually be useful: it makes a viable alternative to ShareFS if you're experiencing the 'large file' problems that can plague modern, fast networks. Once we've understood what the various options mean, we can then move on to making a RISC OS box talk to Linux another month.

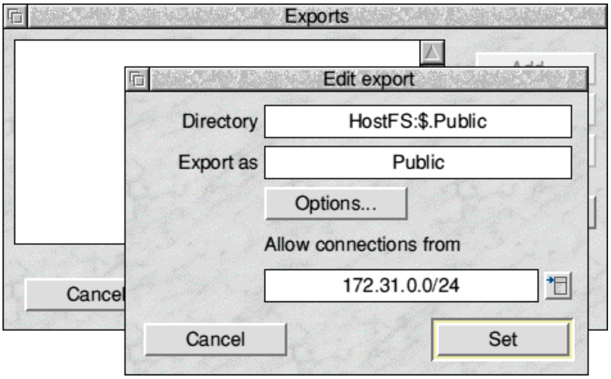
As we might expect, installing Moonfish is simply a case of dragging the application out of the download archive into a suitable place on your disc: it can be started with a double-click and sits on the right-hand side of the iconbar. If you would prefer the server to be available to other systems all the time, or would just like to re-claim some iconbar space, then Moonfish can integrate with Configure: simply place the ***!Moonfish*** application folder into ***!Boot.Resources.Configure*** and it will appear as another icon inside the standard Configure window.

Clicking on the Moonfish icon – either from Configure or on the iconbar, depending upon how it has been installed – opens up an Exports window which allows shared directories to be set up. Clicking on *Choices...* will open the Moonfish Choices window; if installed into Configure, setting *Autoload module at boot* will ensure that Moonfish runs whenever the machine starts up (this option isn't available in the 'stand-alone' setup).

Sharing folders

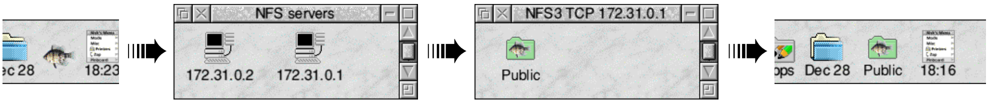
Just as with the other file-sharing systems we've looked at, Moonfish needs to be told which directories it should make available, and how. Open the Exports window – if it isn't already – and click on *Add...* to start creating a new 'export'. The easiest way to specify the folder is to drag it from a filer window: in this case, we'll share the standard ***$.Public*** that Acorn provided as a generic store for a user's files.

Dropping the Public folder into the Edit export window sets *Directory* to the full pathname (***HostFS:$.Public*** in our case) and gives it a default



Directory: HostFS:$.Public
Export as: Public
Allow connections from: 172.31.0.0/24

Adding a new folder for export in Moonfish



Selecting a server from the Sunfish server browse window shows all of the available mounts on that machine; selecting one of these opens that mount and places an icon on the iconbar

name of 'Public' in the *Export as* field. The latter is the name that will appear over the network when looking for folders to connect to, and should be unique in this copy of Moonfish. It would be OK to have a 'Public' export on two different machines: they could be told apart by the machines' IP addresses.

Security is important for NFS shares: there's no password protection on offer as standard. *Allow connections from* contains details of what IP addresses can connect to the new shared folder: by default this is '*' meaning 'anyone', which is far from secure.

The easiest way to use this field is to specify a network address through the 'IP address and netmask' notation that we looked at back in October 2009 (in part 1 of this guide). If your home network uses IP addresses in the range 192.168.0.*x*, then entering '192.168.0.0/24' would only allow machines whose addresses were 192.168.0.*x* to connect. If you recall that this range of addresses is reserved for local network use, and will be hidden from the outside world by NAT (again, see October 2009), you can see this ensures that only local machines can access the server – adjust the IP address to suit your local network settings.

In the example above, the system is running on RPCemu and so the local 'network' is using the obscure 172.31.0.*x* range of local addresses – the same idea still applies, though.

Clicking on *Options...* leads to more advanced settings that control how the export works; we can leave those until we need to talk to a non-RISC OS system. Instead, click on *Set* and then *Save* to make the new export active: the ***$.Public*** folder should be visible to other machines on the local network.

Into the light

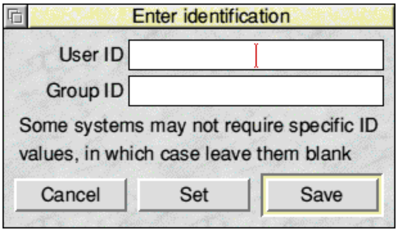
The client end of the NFS connection is made by Sunfish: if Moonfish is analogous to Samba on RISC OS, then Sunfish would be LanMan98. Just as with Moonfish, installation can be as simple as copying ***!Sunfish*** from the archive to a space on your hard disc; if you would prefer to have it start when the machine boots, then add it to the list of applications under *Boot – Run* in Configure.

When it starts, Sunfish installs itself as a filing system on the left of the iconbar: clicking Select over the 'fish' icon (or Adjust over any NFS mount) will open a browser showing any machines on the network that are offering NFS exports, listed by IP address. Remember that the example above is running on RPCemu, so the local 'network' is using the 172.31.0.*x* range of local IP addresses.

Double-clicking on one of these machines' icons will show the specific exports on offer from that machine: in the case of 172.31.0.1, there is the one export called 'Public' which we set up in Moonfish. Double-clicking on this will place it on the iconbar (the first mount will replace the 'fish' icon) and open a filer window to show the root directory.

Who am I?

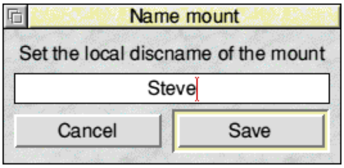
Depending on the server, selecting a mount for the first time might result in an Enter identification window opening. This is needed


Enter identification

User ID
Group ID
Some systems may not require specific ID values, in which case leave them blank

for connecting to 'real' NFS servers on Linux machines, but for now it can be left blank when connecting to a Moonfish server on RISC OS.

Clicking Menu over the 'Public' entry in the exports window above leads to an *Edit* menu which allows various settings to be changed. *Name export...* leads to the Name mount dialogue, from where a name can be given which will show under the iconbar icon and in the filenames of files on that mount. This could be used to make it clear which machine the mount was from, or to shorten some of the longer names that other servers use by default.



User id choices... allows the settings for User ID and Group ID – which we left blank during the connection process – to be changed if required. We'll come back to what these are for in a later column.

Finally, the *Filename and filetype choices...* item leads to a dialogue controlling the way that filenames and RISC OS filetypes are handled. Again, with Moonfish this isn't important so we'll return to it later.

Saving our setup

Sunfish will save any changes made to a mount whenever a *Save* button is clicked in one of the dialogues. To save the current set of mounts, so that they are automatically opened on the iconbar the next time Sunfish starts, select *Save mounts* from the iconbar menu.

Filesharing to Linux over NFS

April 2011

By Steve Fryatt – stevef@wrocc.org.uk

In the previous article, I looked at the use of Sunfish and Moonfish to transfer files between two RISC OS machines. This isn't quite as daft as it sounds, as it provides an alternative for when ShareFS fails, but perhaps a more likely use for them is to transfer files to and from a machine running Linux using the NFS protocol.

The main problem with writing this article is defining what we mean by 'Linux': there are many different versions of the operating system (usually known as 'distros'), and sometimes they are set up in slightly different ways. What follows is written for the popular Ubuntu (which the Club has on the RisCube): as this is a Debian-based distro, the details should also work on other similarly-derived variants.

If you use another flavour of Linux, then NFS is fairly standard and the key details should still be the same. The main differences are likely to be in how the filesystem is installed, and where the configuration files live. Hopefully, even then, this article should give enough clues to enable Google to be used successfully.

As the process is very similar to the one on RISC OS (albeit without the graphical front-end), it would be well worth going back and reading pages 12 to 14 before continuing.

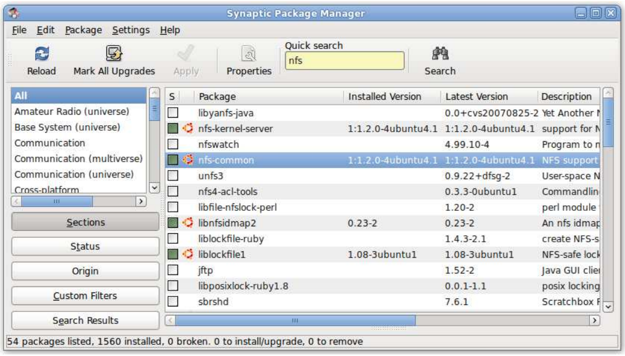
Don't share!

If you're using a recent version of Ubuntu (I use 10.04 – or 'Lucid Lynx'), then you will probably have spotted a *Sharing Options* entry in the context menu when you right-click over a folder in a directory. Although this might seem an obvious thing to investigate, it sets up a Windows-compatible Samba share which can be accessed from RISC OS via LanMan98.

Having never had any happy experiences with Samba on Linux (back in the days of SuSE 7, it must be said), I opted to install the NFS server on Ubuntu and connect to it via Sunfish. The process is very similar to the one for setting up

How to get connected

Sunfish and Moonfish can be downloaded for free from `www.alexwaugh.com/networking`



S	Package	Installed Version	Latest Version	Description
☐	libyanfs-java		0.0+cvs20070825-2	Yet Another N
🟩	nfs-kernel-server	1:1.2.0-4ubuntu4.1	1:1.2.0-4ubuntu4.1	support for N
☐	nfswatch		4.99.10-4	Program to m
🟩	nfs-common	1:1.2.0-4ubuntu4.1	1:1.2.0-4ubuntu4.1	NFS support
☐	unfs3		0.9.22+dfsg-2	User-space N
☐	nfs4-acl-tools		0.3.3-0ubuntu1	Commandlin
☐	libfile-nfslock-perl		1.20-2	perl module
🟩	libnfsidmap2	0.23-2	0.23-2	An nfs idmap
☐	liblockfile-ruby		1.4.3-2.1	create NFS-s
🟩	liblockfile1	1.08-3ubuntu1	1.08-3ubuntu1	NFS-safe lock
☐	jftp		1.52-2	Java GUI clie
☐	libposixlock-ruby1.8		0.0.1-1.1	posix locking
☐	sbrshd		7.6.1	Scratchbox F

54 packages listed, 1560 installed, 0 broken. 0 to install/upgrade, 0 to remove

Ubuntu's Synaptic Package Manager showing that NFS is installed

shares in Moonfish on RISC OS, although the configuration is done via text files. As NFS isn't installed by default, the first step is to make sure that it is on the machine.

Software in Ubuntu is usually installed via the package manager, and NFS is no exception. From the menu bar at the top of the desktop, select *System – Administration – Synaptic Package Manager* and enter your user's password when requested: you'll need to be logged in as a user with administrator rights to do this (if you're using the default account in Ubuntu, then this will be the case). You will also need an active internet connection.

The window that opens contains a long list of available packages: in the *Quick search* field, enter 'nfs' to narrow it down. In the list that remains, it should be easy to find **nfs-kernel-server** and **nfs-common**: right-click each and choose *Mark for installation*. Now enter 'portmap' into *Quick search*, and repeat the process for **portmap**. With all three marked, click on *Apply* (the tick) in the toolbar above, and Ubuntu's answer to Moonfish will be installed. When complete, close Synaptic down.

What to share?

The next step is to tell NFS which folders to share – the equivalent of configuring Moonfish on RISC OS. The list of exports is stored in the */etc/exports* file: this must be edited by the root user, as security prevents normal users from altering it. Open a terminal and enter

```
sudo gedit /etc/exports
```

supplying your password again when requested, and a new copy of GEdit should open on the desktop. On Ubuntu it will initially be empty, apart from some comments (starting with '#') at the top.

The file consists of a series of lines, each of which defines one folder to export. Assuming that I wanted to export my user's home folder, I could add the line '/home/steve' to the end of the file. Unlike under Moonfish, it isn't possible to give exports names.

The filename is followed by one or more spaces, then a series of 'permissions' that limit who can access the share: given that NFS has no other security, this is important. Just as with Moonfish, the machines that can connect are specified as an IP address and netmask combination, so to limit access to the machines on a local network using the address range 192.168.0.*x* we can enter '192.168.0.0/24' – see the previous article for an explanation.

The final part is a list options to apply: for simple read-only access you probably want '(ro,async)', for read-write '(rw,async)'. These options must follow the IP address with no space in between: you can have multiple groups of addresses, each with their own options! The full line would read

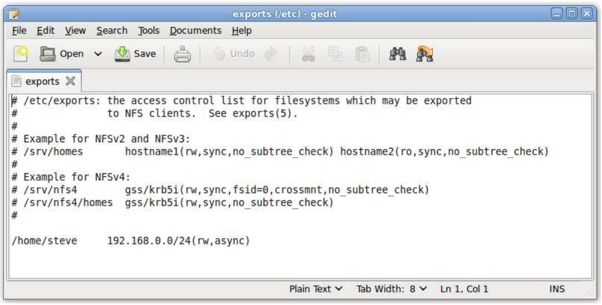
```
/home/steve 192.168.0.0/24(rw,async)
```

if I wanted read-write access, as shown above.

Save the file, and close GEdit. To get NFS to reload the file without a reboot, enter

```
sudo exportfs -a
```

in the terminal to complete the process. The home folder should now be available to other machines on the local network.


exports (/etc) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
exports
Plain Text Tab Width: 8 Ln 1, Col 1 INS

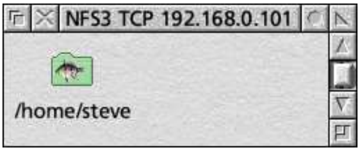
```
# /etc/exports: the access control list for filesystems which may be exported
#               to NFS clients.  See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes       hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
#
# Example for NFSv4:
# /srv/nfs4        gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes  gss/krb5i(rw,sync,no_subtree_check)
#

/home/steve     192.168.0.0/24(rw,async)
```

The /etc/exports file, showing a single share being set up

Access the share

Back on RISC OS, start Sunfish and open the network browser (again, see the previous article). The Linux machine should now appear, and double-clicking on it should show the share that we created. Double-clicking in this will open the Enter identification dialogue.



Unlike RISC OS, Linux has strict rules about file ownership and access permissions: this is important for security, and not just on multi-user machines. If you're accessing your home directory via NFS, then care must be taken to ensure that the files will 'belong' to you when you try and access them natively from Linux.

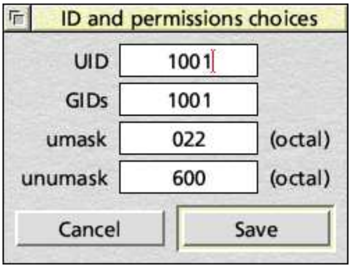
When we connected to a Moonfish share in the previous article, the *User ID* and *Group ID* fields were just left blank – this time, we need to enter values that correspond to the user that we want to own the stored files on the Linux system. If you're just sharing your home directory, then this is likely to be your 'UID' and 'GID' values; if you're accessing a share somewhere else, then you'll need to pick some more appropriate values (it's hard to be more specific, unfortunately).

To find a user's UID and GID, then make sure that you're logged in as the corresponding user on the Linux machine and enter the `id` command in the terminal. This will report details of the account, including two numbers for UID and GID; note them down, and back in Sunfish enter the UID for *User ID* and the GID for *Group ID* (on Ubuntu, the numbers usually start at 1000 for the first user, and count up from there). Click *Save* to store the values permanently, or *Set* to remember them for this session only.

As described in the previous article on page 13, it's now possible to give the share a meaningful name on the iconbar and to configure it as required.

Getting permissions

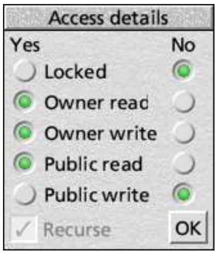
The details of the User and Group IDs can be altered by clicking Menu over the share in the browser and selecting *Edit – User id choices...* to open the ID and permissions choices dialogue. This has fields to change the *UID* and *GIDs* associated with the share, if the ones entered at the time of the first connection need updating for any reason.



UID	1001	
GIDs	1001	
umask	022	(octal)
unumask	600	(octal)

This dialogue also allows file access permissions to be set in the *umask* and *unumask* fields. RISC OS has a limited understanding of permissions: files can be set to have 'owner read' and 'owner write' access, as well as 'public read' and 'public write' using the Filer's *File – Access* menu. Linux (and hence NFS) has much more control, but the default values of 022 and 600 generally work well. For more details as to what they do, and how to change them, see the box on the opposite page.

Linux file permissions



Access details

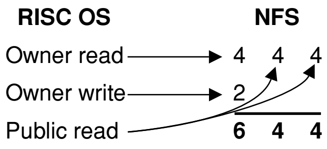
Yes		No
	Locked	●
●	Owner read	
●	Owner write	
●	Public read	
	Public write	●

✓ Recurse OK

Unlike RISC OS, which has simple owner/public read/write access permissions for files, Linux (and NFS) has much more flexibility. Alongside read and write there's an 'execute' flag indicating that a file can be run as a piece of software; in addition to owner and public, there are also settings for the file's 'Group' (the *Group ID* setting that Sunfish requires).

Sunfish maps the RISC OS owner read and write settings directly across to User read and write on NFS, and applies the public read and write to both Group and All; the execute flags are all ignored (as are other special flags beyond the scope of this article).

There's a problem, however: RISC OS doesn't really enforce access permissions in most circumstances, so it isn't uncommon for them to be set wrongly (such as files having no owner read or write access). To handle this – and the missing bits – Sunfish allows us to modify the flags it has calculated before it sends them over to NFS.



Although they're usually given a more friendly face, access permissions on Linux can be represented as a three digit octal (base 8) number in the form 'UGA' (User, Group and All). Each digit is formed by adding 4 for read access, 2 for write and 1 for execute. Given this, we can see that the RISC OS permissions shown above would map to the NFS permission '644': read/write access for the owner, and read access for everyone else.

Sunfish then takes this calculated value, and sets any bits that are set in the unumask: the default value of 600 ensures that the user always has read/write access to their own files. Finally, it *clears* any bits that are set in the umask: 022 ensures that write access is never given to *anyone but* the User (the umask is a Linux concept, so Sunfish uses the same behaviour).

Filename and Filetype Conversion

New

By Steve Fryatt – stevef@wrocc.org.uk

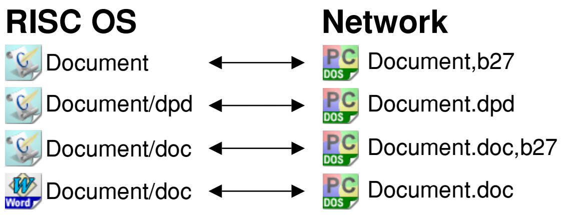
Few platforms have the concept of filetypes in the way that RISC OS does. Although some identify files based on 'fingerprints' within their data, the most common system used to link data to applications is simply to use a suffix to the filename: often separated from the name itself by a full-stop. Thus a file that we would know of as ***text*** might be represented by the filename ***file.txt*** (or ***file/txt*** if we were to view it on RISC OS, where a dot is used as the directory separator).

Whilst the concept of filetypes is useful when we're storing data on native ADFS discs, it causes a problem when we need to put files on to storage media that originates on other systems as we need to find somewhere to store it. Sometimes RISC OS borrows unused parts of the disc to keep its own information (such as Win95FS does with FAT), but with networked storage even this isn't possible.

Hiding filetypes

As a result, the standard convention for storing RISC OS filetypes on non-native discs is to encode them into the filename in a similar way to the ***.txt*** extensions belonging to other systems. The idea originated in AcornNFS, and the RISC OS filetype number is added to the end of the name as stored on the remote machine, written in hexadecimal and separated by a comma. For example, a text file (whose RISC OS type is 'Text' or '&FFF' numerically) might actually be stored with a filename of ***file,fff*** or even ***file.txt,fff***.

The RISC OS filing system automatically hides the *,fff* extension from the user, so at the RISC OS end the method used is completely transparent. However, things aren't as tidy at the other end of the network connection: if you're saving files to a Windows or Linux machine with the intention of using them there, then they can very easily all acquire invalid extensions and the filetype mappings no longer work (a text file named ***file.txt,fff*** won't be recognised by Windows applications as being text, for example).



Combining filenames and filenames

Mime mappings

Fortunately, if you're using a reasonably modern piece of software at the RISC OS end, there's a simple solution. RISC OS maintains a central repository of the mappings between its own numerical filetypes and the three (or more) letter file extensions used on other platforms. It's known as the MimeMap: the name comes from the fact that it also holds details of MIME types used in email and on the web.

Both Sunfish and LanMan98 version 2 (version 1 was different, and we'll come back to that later) can use the MimeMap to try and ensure that RISC OS filetypes are preserved with the least changes to the filename at the other end. Since a picture is worth a thousand words, take a look at the example above of an Ovation Pro document called 'Document': RISC OS type 'OvnPro', or &B27. The three-letter extension for Ovation Pro (for Windows) is ***.dpd*** ('David Pilling Document'), and in this example it has been registered in MimeMap so that the system knows about all of these relationships.

In the first case, where there is no RISC OS extension, SunFish or LanMan98 adds ***,b27*** to the end of the filename on the remote system to preserve the RISC OS type. In the second example, the RISC OS name already contains ***/dpd***: as this is enough to identify the RISC OS type through MimeMap, the filename isn't changed. The third example is perhaps a little surprising: the filetype is still 'OvnPro', but the extension is ***/doc***. MimeMap ties ***/doc*** to the 'MSWord' filetype (&AE6), so the remote file will have ***,b27*** added to its name – after the ***.doc*** – to preserve both the full filename *and* the RISC OS type. The final example changes the RISC OS type to 'MSWord' to match the ***/doc*** extension, so the filename once more travels to the other side of the network unchanged.

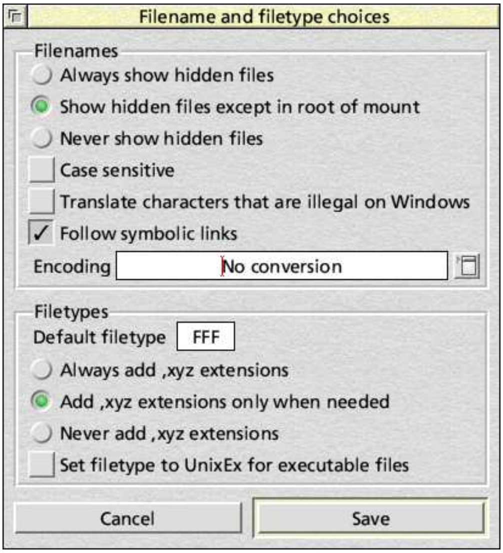
This 'intelligent' renaming, where file names only gain the ***,xxx*** extension when the filename and RISC OS filetype don't match, offers us the best of both worlds. RISC OS types are always preserved, but when we need to use the files on 'the other side' their names are left in the required form so long as we save them with the correct ***/xxx*** extension at the RISC OS end. Equally, files created on Windows or Linux with ***.xxx*** extensions that are known to MimeMap will automatically pick up the correct RISC OS types when viewed through Sunfish or LanMan98.

Configuring Sunfish

To set the filetype translation options in Sunfish, find the mount in question in the browser, click Menu over it and select *Edit – Filename and filetype choices...* to open the relevant dialogue box.

The *Filetypes* section handles the options described above, and *Add ,xyz extensions only when needed* gives the optimum behaviour. Of the other options, *Always...* will include extensions even when unnecessary, while *Never...* will mean that filetypes are only retained when there's an ***/xxx*** extension on the name.

If a suitable filetype can't be found in the filename, then Sunfish will apply the default type: this is text (&FFF) in the example above.


Filename and filetype choices

Filenames
- Always show hidden files
- [✓ Show hidden files except in root of mount]
- Never show hidden files
- Case sensitive
- Translate characters that are illegal on Windows
- [✓ Follow symbolic links]

Encoding: No conversion

Filetypes
Default filetype: FFF
- Always add ,xyz extensions
- [✓ Add ,xyz extensions only when needed]
- Never add ,xyz extensions
- Set filetype to UnixEx for executable files

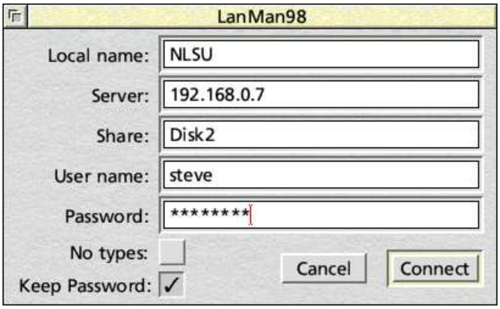
The filename and filetype choices in Sunfish

Finally, it's possible to make files that have the 'execute' permission set on the server take on the RISC OS type of UnixEx (&FE6).

Configuring LanMan98

In contrast, LanMan98 has a smaller range of options: configuration is limited to the *No types* switch when a new share is opened. When off, LanMan98 operates in the way described above: adding ***,xxx*** extensions only when required. Files with no known type default to the type given by the *LanMan98$DefaultType* variable in ***!LanMan98.!Run*** – this is normally text (&FFF).

If the *No types* option is turned on, then all files appear with the default type, and no filename munging takes place: any existing ***,xxx*** extensions will be shown on the RISC OS side.


LanMan98

Local name:	NLSU
Server:	192.168.0.7
Share:	Disk2
User name:	steve
Password:	********
No types:	
Keep Password:	✓

LanMan98's No types option

Usefully, the No types option in LanMan98 is held in the pathname used to access the share: you can see it as '#notypes' on the end of the name in the titlebar of affected filer windows. As such, it's possible to set the option on demand – anywhere that a filename is given (such as applications like StrongMen, Director and so on).

The final difference from Sunfish is that, before consulting MimeMap, LanMan98 will first check its own private list of mappings which are stored in ***!LanMan98.Mappings***. While these are essential in version 1 (which didn't support MimeMap), in version 2 this file should ideally be empty or have all of the entries commented out (with a '|' at the start each line). Adding new mappings to MimeMap is preferable, as they are then available to the whole system.

Illegal characters and hidden files

There are two more wrinkles to be aware of with LanMan98 and Sunfish. First, RISC OS disagrees with Windows systems as to what characters can be used in filenames. LanMan98 will map ?, < and > to #, $ and ^ respectively; Sunfish will, too, as long as *Translate characters that are illegal on Windows* is on.

Second, both Windows and Linux have a concept of 'hidden files' which are not shown to the user by default. LanMan98 doesn't show these as standard, unless ***!LanMan98.!Run*** is edited to set *LanMan98$ShowHidden.*

In Sunfish, the way that hidden files are handled is controlled by options in the Filenames section of the dialogue above: *Always show...* and *Never show...* should be self -explanatory. *Show hidden files except in the root of mount* makes use of the fact that on Linux a user's root home directory contains many hidden files storing settings. Hiding these is often useful, while in other directories further down the tree hidden files are more likely to be ones that the user will want to see and edit.

The WROCC Guide to Networking

This time last year, WROCC launched Part 1 of our Networking Guide: a compilation of articles from the Club newsletter – *The WROCC* – which had followed on from an evening that we spent 'demystifying networking' in September 2009.

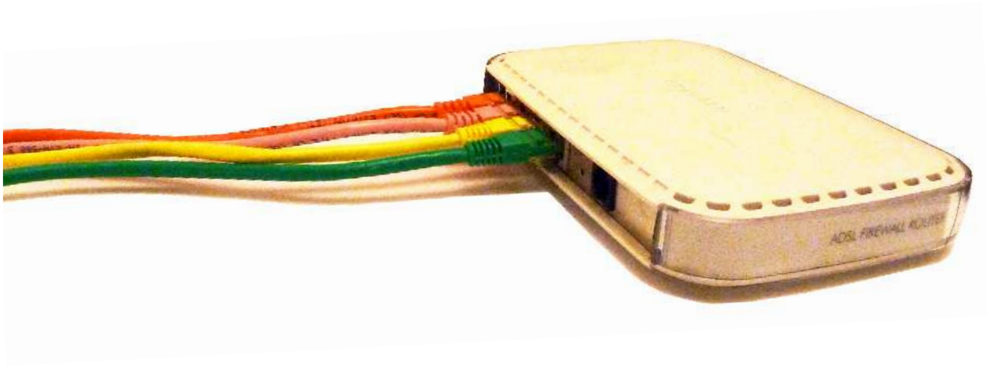
The themed articles didn't stop, and so another twelve months on we're back again with Part 2 of the guide. While the first instalment was mainly about the mechanics of connecting RISC OS systems to the wider world (with a look at ShareFS and Samba thrown in), this time we further investigate the thorny subject of how to share files with other systems and also take a look at the RISC OS 6 firewall.

Taken together, the two parts of *The WROCC Guide to Networking* form a useful reference for anyone who needs RISC OS to work effectively with other systems. These articles only form part of our newsletter, however: each year, we cover many other aspects of our favourite platform in our monthly issues.

If you would like to receive *The WROCC*, you can join us today for the introductory price of £7 for the first year. This isn't only for those who live within travelling distance of Yorkshire – our many 'postal members' receive a PDF copy electronically or in printed form via the Royal Mail for a small extra fee to cover printing and postal costs.

If you would like to know more about joining us, just ask at the Club stand for more information or get in touch after the show.

Steve Fryatt – Editor, *The WROCC*
editor@wrocc.org.uk



Wakefield RISC OS Computer Club
`www.wrocc.org.uk`
3 Riverdale Avenue, Stanley, Wakefield, WF3 4LF