MP3 Player Box HOWTO

Table of Contents

Table of Contents

MP3 Player Box HOWTO

Konrad Rzeszutek `conradus@iname.com`

v1.1, 2001−04−19

This document describes how to build, configure, install, and use a custom MP3 player box. It lists the necessary hardware and answers a number of frequently asked questions.

---

8. References

---

1. Introduction

This is a build guide for a custom MP3 box, which has no harddrive, floppy, or video card and boots from the network. It also fetches the songs from the network. It is intended as template, quick reference, and a template to building a custom MP3 box. Frequently asked questions related to MP3 box building, its usage, problems, and references are given.

1.1 Acknowledgments

Much of this information came from authors private ideas and source files covered under GNU licenses. Huge thanks goes to Tom Jones (for helping me design the box), Ryan Defelic (for his MP3 box being a alpha−prototype), Steve Maroney (for asking me many questions), the LRP project, and of course **The Penguin** (which sits on his shrine next to my network closet :=))

1.2 Copyright

1.3 Feedback

I rely on you, the reader, to make this HOWTO useful. If you have any questions, corrections, or comments, please send them to me, conradus@iname.com, and I will try to incorporate them in the next revision.

---

2. MP3 box

The sole and only reason why you, the reader, is engulfing yourself in reading this material along with a nice warm container full with java, is so that you:

- Wish to make a self−supportive box with only four cables sticking out of it: an RJ−45 network cable, a power cable, and two RCA cables (or just one 1/8 mini stereo plug).
- Have all your mp3's on a big−server, while this MP3−box would fetch the files from the server.
- Have no floppy driver, no video card, nor any harddrive in the MP3−box. Just a power supply, sound card, network card, memory, cpu, and a serial infrared receiver.
- Be the envy of your friends :=)
- Play your mp3's using your remote without having to turn on your workstation.

2.1 Design

Its important to keep certain things in mind:

- Size of the computer component (I was tempted to use PC104 boards, but they were to expensive). Height of ISA/PCI cards (so that the box doesn't look like a baby−AT case).
- The appealing side of the box − does it look like a square rectangle, pyramid, triangle, or something even more weird?
- Paint. Do you want see−through box with flashing lights inside, inconspicuous black box, or chrome looking?
- Cost. The absolute minimum (taking into consideration that you, the reader, has no spare computer parts) is about $70 (excluding tax, shipping, etc)
- KISS (Keep It Simple and Stupid) − take everything that's unnecessary and get rid of it (parallel port, modem card, video card, etc).

The idea behind this box is to separate tasks − the MP3−box can only play the mp3's from a server, while you, the reader, can freely add more files to your big−end server. Thus, the MPEG−box wouldn't need a harddrive, nor floppy drive. It would boot the operating system from the network (from the same server where the mp3 files are located).

---

3. Necessary hardware

The software running the decoding process (mpg123) can happily work on a 486DX/66 or higher without any constraints. 16MB of RAM ought to do fine as well. Any sound card that is supported under Linux should work. The following list includes all the hardware I used:

- Pentium 120 downclocked to 90 Mhz (I needed to lower the temperature of the CPU so that I wouldn't have to use a fan − just a heatsink). Cost: $5.
- Unknown brand−name motherboard (AT−style, 4 PCI, 3 ISA, 4x DIMM, Award BIOS) − if you are going to get rid of the video card (after you have configured the machine) make sure you change the display from CGA/EGA (or VGA) to None. Cost: $10.
- Small powersupply (50 or less watts would do). I got one from a Compaq computer (the wiring on the powersupply didn't match the AT P8/P9 connector standards, thus I had to match the cables and rewire them to fit my AT motherboard). You could probably use any powersupply as long as it gives you 12VDC and ground. You won't need the 5VDC unless you have a keyboard, mouse, or floppy

connected. Cost: FREE (found the computer in a dumpster).

- Intel EtherExpress Pro 10/100 Management Network Card. Any network card will do as long as it has *PXE* or *BOOTP* functionality − those things are also known as network booting. Without this kind of network functionality, you would need to attach a harddrive/zip−drive/floppy−drive to boot the operating system from. Cost: $15 (Intel had a promotion and they were selling two for $30).
- Sound card. I got a Crystal CS424x card b/c its height was pretty small compared to other sound cards. Cost: $15.
- Remote control. This is the difficult choice − if you pick a Packard Bell remote, you will end up pulling your hair out. I got a Logitech AST Remote. Before you get a remote, you ought to check for software that will work with your remote. Cost: $5.
- Cabling − stereo RCA cables from RadioShack. Network cable and a jack from local distributor of electronic components. Cost: $9.
- Small 8mm screws to screw the custom box together. Picked up at Home Depot along with plexiglass. Cost: $2.
- Plexiglass − used to construct the custom box. Bending of the box is pretty easy, as long you have a torch and don't burn your own eyelashes with it. Picked it up at Home Depot. Cost: $10.
- Black paint so the box would look inconspicuous and color−match my stereo rack. Cost: $2.

Total cost: $75.

---

4. Server side software

Before you proceed with installing the server−software, it would be beneficial to review the following HOWTO's (in the listed order):

- Net−HOWTO
- Ethernet−HOWTO
- Diskless−HOWTO
- NFS−HOWTO

Particularly, the **Diskless−HOWTO** contains a wealth of information useful for this project.

The required software are mostly already installed on a stock Linux distribution, but some might be missing. The ones that are usually not found (dhcpd, pxe, etc) are included in the mpeg−box−project.tgz file.

- NFS server. Also called **nfs−server** or just **nfs**
- TFTP server. Called **tftpd**.
- DHCPd server. There are two versions. The latest one supports natively PXE extension (which my network card uses), but doesn't support menu based booting − something I needed during the development. I'm using v2.0 and for PXE extensions I'm using an DHCP proxy (all these strange terms will be explained later on).
- DHCP Proxy server − PXE. Not needed if you are using the latest version of DHCP. The package is called **pxe**. Make sure you pick the latest one, or just use the one that's included in my file.

4.1 NFS server installation.

I presume you have read the **NFS−HOWTO** . If you haven't, do yourself a favor and pick it up. Most of these information are straight from that HOWTO.

Before you start setting NFS, make sure you have **nfs−utils** or **nfs−server** package installed. If you don't have them, the dhcpd−2.0pl2 source code is included in the mpeg−box−project.tgz file.

- Extract the file −
  `tar −zxvf mpeg−box−project.tgz`
- go into *1st Step − configuring server/01 − dhcpd.*
  `cd "1st Step − configuring server/01 − dhcp"`
- From there run:
  ```
  ./configure
  make
  make install
  ```

Setting up the server will be done in two steps: Setting up the configuration files for NFS, and then starting the NFS services.

Setting up the Configuration Files

There are three main configuration files you will need to edit to set up an NFS server: */etc/exports*, */etc/hosts.allow*, and */etc/hosts.deny*. Strictly speaking, you only need to edit */etc/exports* to get NFS to work, but you would be left with an extremely insecure setup. You may also need to edit your startup scripts.

/etc/exports

This file contains a list of entries; each entry indicates a volume that is shared and how it is shared. Check the man pages (*man exports*) for a complete description of all the setup options for the file, although the description here will probably satisfy most people's needs.

An entry in */etc/exports* will typically look like this:

directory machine1(option11,option12) machine2(option21,option22)

where

directory

  the directory that you want to share. It may be an entire volume though it need not be. If you share a directory, then all directories under it within the same file system will be shared as well. For example it might be: */exports*

machine1 and machine2

  client machines that will have access to the directory. The machines may be listed by their IP address or their DNS address (e.g., machine.company.com or 192.168.0.8). Using IP addresses is more reliable and more secure.

optionxx

  the option listing for each machine will describe what kind of access that machine will have. Important options are:

- ro: The directory is shared read only; the client machine will not be able to write to it. This is the default.
- rw: The client machine will have read and write access to the directory.
- and many more ...

For our MP3−box, we just need to read−only access to the directory with mp3's. Suppose the MP3−box IP will be 10.0.0.8 and the directory with our music files is */exports/media*. The */etc/exports* would look like this:

```
/exports/media     10.0.0.8(ro)
```

/etc/hosts.allow and /etc/hosts.deny

These two files specify which computers on the network can use services on your machine. Each line of the file is an entry listing a service and a set of machines. When the server gets a request from a machine, it does the following:

- It first checks hosts.allow to see if the machine matches a description listed in there. If it does, then the machine is allowed access.
- f the machine does not match an entry in hosts.allow, the server then checks hosts.deny to see if the client matches a listing in there. If it does then the machine is denied access.
- If the client matches no listings in either file, then it is allowed access.

In general, it is a good idea with NFS (as with most internet services) to explicitly deny access to hosts that you don't need to allow access to.

The first step in doing this is to add the following entry to */etc/hosts.deny*:

```
portmap:ALL
```

Next, we need to add an entry to hosts.allow to give any hosts access that we want to have access. (If we just leave the above lines in *hosts.deny* then nobody will have access to NFS.) Entries in *hosts.allow* follow the format:

```
service: host [or network/netmask] , host [or network/netmask]
```

Here, host is IP address of a potential client; it may be possible in some versions to use the DNS name of the host, but it is strongly deprecated.

Suppose we have the setup above and we just want to allow access to huh.com and arakis.dune.com, and suppose that the IP addresses of these machines are 10.0.0.8 and.10.0.0.1, respectively. We could add the following entry to */etc/hosts.allow*:

```
portmap: 10.0.0.8, 10.0.0.1
```

For recent nfs−utils versions, we would also add the following (again, these entries are harmless even if they are not supported):

```
lockd:  10.0.0.8, 10.0.0.1
rquotad: 10.0.0.8, 10.0.0.1
mountd:  10.0.0.8, 10.0.0.1
statd:  10.0.0.8, 10.0.0.1
```

Starting the NFS server.

The shell scripts which start the NFS server are usually in */etc/init.d* or */etc/rc.d*. Depending on your Linux distribution, you either need to run: *netconfig*, *setup* (RedHat) ; *YaSt2*(SuSe); *turbo−config* (TurboLinux); or you are stuck editing */etc/init.d/nfs−server* files manually. If you run those programs, look for options that list **NFS** or **nfs−serveR** and start those services.

4.2 DHCPd server installation.

DHCP (Dynamic Host Configuration Protocol) is a protocol for computers to find out their IP address, gateway settings, netmask, and a lot more. Its a quite nice off−load for the system administrator. The reason why you want to use it is because when the MP3−box comes up, as said before, it doesn't have a harddrive, nor floppy drive − therefore it can't boot an operating system. But with a card that supports PXE, the following happens:

- The network card starts sending a DHCP request to the network
- The DHCP server offers a IP for the MP3−box.
- The MP3−box gets the IP address and then procceds to send PXE extension DHCP request − an extension to the DHCP, which is not covered in the RFC.
- The PXE extension daemon issues the NBP (Network Boot Program) to the card on the MP3−box.
- The NBP contacts the TFTP server and downloads the Linux kernel, and the initrd.gz file. After that it runs the Linux kernel.
- The Linux kernel does what it has been programmed to.

Without the IP address, the system wouldn't be able to get the NBP, neither the Linux kernel.

Setting up the DHCP daemon is a two stage process − installing the binaries, and setting up the configuration file for DHCP.

Installing DHCPd

The version of DHCP I'm using comes from Internet Software Consortium, the version is 2.0. You might already have the binaries installed, if that's the case − skip this section.

The source code is *1st Step − configuring server/01 − dhcpd/dhcp−2.0pl2*, go into that directory and issue the following commands:

```
./configure
make
make install
```

The shell scripts which start the DHCP server are usually in */etc/init.d* or */etc/rc.d*. Depending on your Linux distribution, you either need to run: *netconfig*, *setup* (RedHat) ; *YaSt2*(SuSe); *turbo−config* (TurboLinux); or you are stuck editing */etc/init.d/dhcpd* files manually.

Setting up the configuration files for DHCPd

The most important part of the */etc/dhcpd.conf* file are the option fields and the host field. For the v2.0 of ISC−dhcpd its necessary to specify

```
option dhcp−class−identifier "PXEClient";
```

```
option vendor-encapsulated-options ff;
```

This will enable the DHCP server to recognize the network cards with PXE as valid. Make sure you also have the host section filled out. For example, for the MP3−box called *swallow.eoh* the host entry would look like:

```
host swallow {
        hardware ethernet 00:90:27:c1:dc:db;
        fixed-address swallow.eoh;
}
```

This is allow the DHCP server to give the box with that MAC address (which the network card displays during boot−up), the IP address associated with the host *swallow*. How is the host associated with 10.0.0.8 − that's the job of DNS server (which you hopefully have installed). If you don't have DNS, you can use flat−host resolution and add into */etc/hosts* the following line (replace the 00:90:27. .. with your MAC address):

```
10.0.0.8       swallow.eoh    swallow
```

Back to the program ...

Following is my */etc/dhcpd.conf* file:

```
option domain-name "eoh";
option dhcp-class-identifier "PXEClient";
option vendor-encapsulated-options ff;

option subnet-mask 255.255.255.0;
default-lease-time 600;
max-lease-time 7200;

subnet 10.0.0.0 netmask 255.255.255.0 {
        range 10.0.0.20 10.0.0.40;
}
host swallow {
        hardware ethernet 00:90:27:c1:dc:db;
        fixed-address swallow.eoh;
}
```

You can check your DHCP server by booting up a Windows workstation, and change its IP address to *Obtain an IP address automatically*.

4.3 PXE daemon

PXE is standard for remote booting. The following information is taken from *1st Step − configuring server/02 − pxe/pxe−README*:

PXE is an extension to DHCP and also method of remotely booting. The specs for PXE can be found at ftp://download.intel.com/ial/wfm/pxespec.pdf. The PXE daemon provides two capabilities: *proxyDHCP* and *PXE Bootserver*. The PXE daemon can be set up to provide either or obth of the capabilities. Both capabilities are required.

proxyDHCP works in parallel with DHCP and provides the booting client with a remote boot configuration options. ProxyDHCP provides the PXE client(s) with the following information: remote boot prompt with optional timeout, remote boot menu and PXE Bootserver discovery options.

The PXE Bootserver is a capability provided by the PXE daemon. The PXE Bootserver is the capability that provides the booting client with boot images for a particular boot environment.

PXE Bootserver serves up requested NBPs (Network Boot Programs) to PXE clients. PXE Clients locate PXE Bootservers using discovery information provided to the client by proxyDHCP. The discovery method used by the PXE client (multicast, broadcast or unicast) and the list of available bootserver types is controlled by proxyDHCP. PXE Bootservers always listen for all three types of discovery requests and will respond to all valid requests.

Installing PXE software.

If you have RedHat or SuSE, pxe is available as an RPM − download it and install it. Unfortunately, you are still going to need the sources − you will have to recompile the NBP (Network Boot Program).

All the paths mentioned below refer to the contents of the mpeg−box−project.tgz file, which has all the source files and configuration files listed below.

The source file for PXE server are located in

1st Step − configuring server/02 − pxe/pxe/pxe−linux/server.

If you don't have PXE installed, just run: *make* and *make install*. The files will be installed in */usr/local*. After that you should copy the *1st Step − configuring server/02 − pxe/pxe/pxe.init* file to */etc/rc.d* or */etc/init.d*. After that go into your */etc/rc2.d* and link */etc/init.d/pxe.init* file to *S99pxe*. The command is:

```
ln −s ../init.d/pxe.init S99pxe
```

If you have no clue what I'm talking about, just enter

```
pxe
```

that will start the PXE daemon (but do that after you have read the next section)

Setting up the PXE daemon.

To get the PXE daemon completely working, its necessary to setup a couple of configuration files/directories.

- *pxe.conf* which is located in *1st Step − configuring server/03 − etc files* should be copied to */etc*. The differences between this file and the one that comes with PXE software is small − the order of boot programs is changed. The original *pxe.conf* is located in *1st Step − configuring server/02 − pxe/pxe/pxe−linux/server/services*.
- *mtftpd.conf* which lists multicast addresses and the files associated with it. Its better to leave the file alone. Copy it from *1st Step − configuring server/03 − etc files* into */etc* directory.
- The file */etc/inetd.conf* needs to be modified so that it will include support for TFTP (in case MTFTP doesn't work) and MTFTP. The lines that should be added are:
```
mtftp   dgram   udp     wait    root    /usr/sbin/tcpd in.mtftpd /tftpboot
tftp    dgram   udp     wait    nobody  /usr/sbin/tcpd in.tftpd
```
- Add into */etc/services*:
```
mtftp           1759/udp
```

```
pxe             67/udp
pxe             4011/udp
```

This will your life a little simpler when you are going to debug network problems.

* The last thing will be to create the */tftpboot* directory with all the files in it. Do the following command:

```
mkdir -p /tftpboot/X86PC/UNDI/BStrap
mkdir -p /tftpboot/X86PC/UNDI/linux-install
```

The copy the *bstrap.0* file (which is located in *1st Step − configuring server/02 − pxe/pxe/pxe−linux/server/services* into */tftpboot/X86PC/UNDI/BStrap*. Do the same thing for *linux.0*, but copy it into */tftpboot/X86PC/UNDI/linux−install*. If you don't like any of these patchs, modify the */etc/pxe.conf*, */etc/mtftpd.conf* accordingly.

After all these steps, make sure you have a DHCP daemon running and working (look in the section about DHCP to find out how). Start the pxe daemon by typing

```
pxe
```

The debug information for PXE is piped into */PxeServiceLog.txt*. To turn this debug−feature off, edit the */etc/pxe.conf* − search for *DebugOutToFile*.

Turn on your system with a network card that supports PXE. You should see the card getting an IP from the DHCP server, and then showing up a menu. If that's the case, you have successfully configured the server system, and its time to get configure the Linux kernel and initrd file for the MP3−box in the next section.

---

5. Client side software

The name of this section is rather misleading − since we are booting the system remotly, all the software is not located on the client, but rather on the server. The software, which includes the Linux kernel and a small Linux distribution is under 2MB of space. It could probably be squished into 1.44MB (so that it would fit onto a floppy) but its more fun to remotly boot the MP3−box, and not use floppies. The Linux kernel used was 2.2.16. Any versions can be used. The mini−distribution used is LRP − it normally fits on a 1.44 floppy compressed. The only thing that was left from the original mini−distribution is the binaries − the rest was changed by me.

5.1 Linux kernel

Making a new kernel for the MP3−box is a necessity. Your sound card will inevitable different than mine, and you might have a different network card.

The Linux kernel tree that I used is supplied. Look in:

2nd Step − configuring client/01 − linux kernel/linux−2.2.16

and type:

```
cd "2nd Step - configuring client/01 - linux kernel/linux-2.2.16"
make menuconfig
```

and choose the right drivers. If you want to use my startup scripts for the mini−linux distribution, make all the drivers modularized (network, filesystem, sound card).

After successfully choosing your drivers, make the kernel. Copy your new kernel *2nd Step − configuring client/01 − linux kernel/linux−2.2.16/arch/i386/boot/bzImage* into your tftpboot directory as *linux.1* (**not as linux.0**!)

```
cd "2nd Step - configuring client/01 - linux kernel/linux-2.2.16"
make bzImage
cp arch/i386/boot/bzImage /tftpboot/X86PC/UNDI/linux-install
```

Presumably, you configured your network, sound drivers as modules. It will be necessary to compile them and put them in temporary directory (until you get the the mini−distribution setup). To compile and install in temporary directory do:

```
export INSTALL_MOD_PATH=/tmp
make modules;make modules_install
```

The modules for the kernel will be located in */tmp/lib/modules/2.2.16*. Remember that path.

5.2 Linux filesystem

We need to make a filesystem that the Network Boot Program can download from the server, and pass to the kernel. The maximum size of such filesystem is limited to the amount of memory in the machine, but generally it shouldn't be bigger than 4MB.

Since we want to boot the MP3−box from the network, and have no media storage device on the MP3−box, the filesystem cannot be on a harddrive, floppy drive, nor ZIP drive. Linux provides a facility for that called Initial RAM Disk (initrd) – its a file, which actually is a ext2 filesystem embedded in a file.

Creating initrd filesystem.

To understand why we need to create a INITial Ram Disk filesystem, its necessary to understand how Linux boots up.

- After the kernel is finished loading itself, it starts executing itself.
- The kernel converts initrd (which the boot loader loads into memory as linux.2) into a "normal" RAM disk.
- When its done, it mounts the device specified in *root_dev* (which is changed by the Network Boot Program). This usually points to */dev/ram0* or */dev/ram1*.
- */linuxrc* is executed.
- When linuxrc terminates, the "real" root file system is mounted (which can be the RAM disk)
- */sbin/init* is invoked.

For more info, *man initrd*.

Creating initrd fs.

The initrd – is a initial ram disk embedded inside a file. Creation of such a file is quite straightforward (you have to be root to use *losetup*).

```
dd if=/dev/zero of=/root/initrd count=4096 bs=1024
losetup /dev/loop0 /root/initrd
mke2fs /dev/loop0
losetup -d /dev/loop0
file /root/initrd
```

This creates a *initrd* file 4MB big. The last command should say: *Linux/i386 ext2 filesystem*. This is the file that we eventually use to boot the Linux OS on the MP3−box.

Mounting initrd fs.

To use the initrd for something its necessary to make a mount−point a mount the initrd file:

```
mkdir /data
mount /root/initrd /data -o loop
```

Now you freely copy files back and forth onto the 4MB initrd file, which is mounted under */data*.

Copying LRP filesystem to initrd fs.

Its time to put something meaningful on the 4MB initrd. We need to copy a small linux distribution onto the initrd, otherwise we can't use it boot the MP3−box.

```
cd "2nd Step - configuring client/02 - the filesystem"
tar -cf - * | (cd /data; tar -xvf - )
```

Copying custom modules,

Do you remember the modules that we compiled in the previous section?

```
cp -Rf /tmp/lib/modules/2.2.16/* ./data/lib/modules/2.2.16
sync
df -h
```

df −h serves only to show you how much space you have left on the filesystem.

Preparing the initrd fs for NBP.

Using the initrd file generally means we have to unmount the file, gzip it and copy it into the tftpboot directory.

```
umount /data
sync
gzip -c9 /root/initrd  > /tftpboot/X86PC/UNDI/linux-install/linux.2
```

This will unmount the initrd, and compress it into *linux.2* file (which is what the NBP will look for and download).

Conclusion

That's basicly the method of designing and using an initrd filesystem. One thing to keep in mind − the initrd should always be compressed when deployed.

5.3 Configuring NBP (Network Boot Program).

After you have copied your initrd file and tried to boot up the workstation, you found out that the kernel complains about not being able to mount the root partition. The problem lies in the NBP that is supplied with PXE − it was compiled for RedHat type initrd images, and therefore does some changes to the kernel.

- When the NBP is finished downloading the kernel, it sets the *root_dev* environment to whatever was compiled inside the NBP. By default, that is 0101, which is */dev/ram1* (look at its major, minor).
- NBP passes the environment to kernel (in and in turn overwriting whatever settings the user supplied to the kernel using *rdev*) and starts it.
- The kernel happily executes and tries to mount a filesystem from */dev/ram1*>, while the initrd image sits in */dev/ram0*.
- The kernel hangs.

The solution is to recompile the NBP with the right root_flags. Look in

1st Step − configuring server/02 − pxe/pxe/pxe−linux/nbp.linux/prepare.c. Line 212 lists the hexadecimal address. Change it to *0x0100* − that way the kernel will mount */dev/ram0* and actually find a initrd filesystem.

5.4 Configuring startup scripts.

When the kernel completes loading the initrd into memory and mounts it as a filesystem, then */sbin/init* is executed, which handles the rest of starting the operating system.

- init reads in */etc/inittab* − which lists which scripts must be executed depending on the computer's state.
- It starts executing the boot−time configuration script */etc/rc.d/rc.boot*.
- *rc.boot* setups BusyBox and POSIX links, configures all the modules − sound, network, and filesystem. If you are using your own modules, you **must** edit this file and change the settings.
- Then *init* changes the runlevel to its default − 2, which result in running */etc/rc.d/rc.multi*.
- *rc.multi* setups the network card (configures its IP), mounts the NFS server and starts the audio−subsystem (*/etc/rc.d/rc.audio*). It gets all the custom configuration (where is the NFS server. its IP, etc) from */etc/rc.conf*.
- *rc.audio* configures the sound, creates playlists, and starts the ARCamp program (which listens to the remote and runs/kills mpg123).

The scripts were made as simple as possible. The files you ought to look into and alter are:

- */etc/rc.conf* − has the NFS IP, the mountpoint, and the client's IP
- */etc/rc.d/rc.boot* − loads the network module, and the filesystem module.
- */etc/rc.d/rc.audio* − loads the sound card module.

5.5 Configuring the infrared receiver program.

The Logitech Remote I obtained was quite easy to get working with Linux. The software used in Linux was **ARCaMP** (AST Remote control MP3 Player). The software basicly listens at the serial port for some predefined characters, and based on them, launches *mpg123*.

Getting scan−codes.

The source code is included in *2nd Step − configuring client/99 − the sources/ARCaMP*. It differs from the original code − the key mappings are different and the default playlist has been altered. If you find out that the program works with your remote, but the buttons −> action mappings are screwed up, then edit the defs.h to match your scan−codes with your remote. Getting the scan−codes is quite easy:

```
cat /dev/ttyS1 | od −t x2
0000000 2000 7f10 7f10 7f10 7f10 7f10 7f10 7f10
0000020 7f10 7f10 202a 7f10 7f10 7f10 7f10 7f10
0000040 7f10 7f10 7f10 7f10 7f10 7f10 202a 7f10
0000060 7f10 7f10 7f10 7f10 7f10 7f10 7f10 202a
0000100 0f0f 0f0f 0f0f 0f0f 0f0f 2a0f 0f20 0f0f
0000120 0f0f 0f0f 0f0f 2a0f 0f20 0f0f 0f0f 0f0f
0000140 0f0f 0f0f 202a 1212 1212 1212 1212 2a12
0000160 1220 1212 1212 1212 1212 2a12 1220 1212
0000200 1212 1212 1212 1212 1212 202a 1212 1212
```

/dev/ttyS1 is the serial port for my infrared receiver, yours might be different. *od −t x2* converts the output to hexadecimal.

The first eight bytes are just offset locations − ignore them. If you look closely at the strings it becomes clear that whenever you hit a the same button on the remote, you get similar looking strings of numbers. The first three rows are actually the fastforward button being hit three times. The repetition of *ox20, ox2A, 0x7F,* and *0x10* suggest that *0x20* and/or *9x2A* are remote−id number, while *0x7F* or *0x10* are codes for fastforward.

The next three rows are for the play button (the scancode is *0x0F*). And as you can see, the repetition of *0x20, 0x2A* suggest remote−id again.

The last three rows are for the stop button − *0x12*.

Compiling ARCaMP.

Just run *make* and copy the *arcamp* file into */data/bin* directory − assuming you still have the initrd filesystem mounted under */data*.

Different remote.

If you have different remote, and the scancode−match procedure doesn't work, then you won't be able to use ARCaMP. Instead you will have to search for some software under Linux that will support your remote. Look at the Remote control programs section at the end of the document for some URLs.

Just keep in mind, that you will have to modify */etc/rc.d/rc.audio* (on the initrd fs) so that it will spawn a different program then ARCaMP.

mpg123

The program is staticly compiled and put in */bin* (on the initrd) directory. If you find it necessary to get a more recent version , recompile it with *−static* flag and put in */bin*.

6. Plexiglass box

The box can be made of wood, metal or anything else. But the most simplest way of building ones box is to use plexiglass. Plexiglass under heat − from a torch, kitchen stove − will warm up and is susceptible to bending. The disadvantage of plexiglass is that its can carry electric charge − thus its unadvisable to put electronic components on the plastic − use those small plastic pegs for motherboards − glue them on the plexiglass.

The design of the box depends on your parts. If you have a small motherboard, low−profile sound card and network card, you can make the box 4"−5" tall. The placement of motherboard, power supply, and the infrared serial receiver all depends on your imagination.

In the case of my box, the minimum I could get was 6" height (the motherboard + plastic pegs + ISA Sound card). Therefore I decided on building a wide box to compensate for the height. The shape of the box is semi−pyramid.

During the design, heat and air circulation has to be taken into account. If you wish to make your system noise−less (without fans), make the box big, and drill holes on sides, and on the top so that hot air can raise and escape. You also might consider downclocking the CPU − that way you can use a heatsink instead of CPU fan. Another alternative is to drill holes one side of the box, and put a fan on other side, and have the CPU in the middle of the air−flow. Take also in consideration your power supply. The small power supplies (83W, 40W) don't exhaust too much heat and can be safely left without a fan. Unfortunately, the bigger they get the more heat they exhaust − and the heat has to be drawn out.

In my case I decided to use one fan to cool down the power supply and CPU. The fan is mounted on top of the power supply. The power supply has exhaust vents on top, from which heat escapes. The CPU is located 20cm away from the fan, under 45" angle − also not that far away from the CPU I drilled some holes to suck in fresh air. What essentially happens is that outside air is sucked in thru the holes, passes over the CPU and then combined with the air from the power supply is sucked out by a fan. For pictures: check mpeg−box pictures.

---

7. Answers to Frequently Asked Questions

7.1 Can't compile the kernel. It says: "2nd: No such file or directory"

Move the whole linux−2.2.16 directory somewhere else where the directory−name doesn't have spaces in itself.

7.2 My remote doesn't work with ARCaMP

Tough! :P

You will have to look for some other software that will support your remote control. Look in the Remote programs URL section.

7.3 The Linux kernel can't mount the filesystem.

Look in Configuring – NBP Network Boot Program

7.4 How do I bend plexiglass?

Get a torch and heat the opposite side of the plexiglass that you are going to bend. You should move the torch back and forth, not to fast, and not too slow. One minute – two minutes should be fine, if you warm up it too much, bubbles will appear. After you have heated the material, put in on table and slowly bend it. Keep it under the desired angle up to a minute.

7.5 The system doesn't boot up after I take the video card out.

Go into the BIOS settings and change the **Video Mode** (it might be called something else on your system) and change from **CGA/EGA** or **VGA** to **None**.

---

8. References

8.1 Remote control programs

- IRC is a package that supports receiving and sending IR signals of the most common IR remote controls. It contains a daemon that decodes and sends IR signals, a mouse daemon that translates IR signals to mouse movements, and a couple of user programs that allow to control your computer with a remote control. www.lirc.org
- Cajun is a program that allows you to turn any computer (>75mhz) into a massive audio jukebox for your car or home. It uses the matrix–orbital serial display and supports the IRman infra–red remote control interface. Soundcard output is delivered to your (car or home) stereo for amplification. The software supports a hotlist and shuffle mode. cajun.sourceforge.net
- IRMP3 is a multimedia audio jukebox for Linux. IRMP3's strength is that it easily integrates into home, car, and mobile environments (where a keyboard/screen interface may not always be available). It supports several input and display devices such as infrared remote controls, LCD displays, network connections, and serial keypads. It includes a robust set of tools like alarms, sound environments, mixers, sleep functions, and more. It supports MP3 players and network–ready control & status monitoring, and includes a flexible, modularized source which allows functionality to be easily extended with user–developed code. www.dpotter.com/irmp3
