Create an Embedded Asterisk Server

JR Richardson
Engineering for the Masses

This project is really to have an Asterisk server "operate like an embedded system". A true embedded system is much smaller and specific with a minimal OS to run minimal hardware to accomplish a specific task for a "hardened" or repeatable event. Examples: Linksys Router, D-Link Wireless Access Point, VoIP hard phone, etc.

In embedded systems, there is usually a boot ROM that holds a compressed software image and upon boot, extracts into RAM and runs the system. Also common is a NVRAM (non-volatile) which stores any configuration files that need to be changed from time-to-time and these changes need to be maintained after a power cycle.

My version is really all about using off the shelf pieces and parts to keep the cost down. Create your own software to function as you need it and work specifically on the hardware you have. Do so in a repeatable, reliable and unconventional manner to gain the utmost "hardened" utility with this application.

The coolest thing is being able to walk up to the Asterisk server, cycle power and not worry if it will start back up, it will, in the same state as which it was last in. With a conventional setup, Asterisk installed on a hard drive, power cycling server without shutting down the OS properly is not be a good thing and can lead to file system corruption.

This project boots from a read-only memory (ROM) that stores a linux kernel (the one we compile for this hardware) and a compressed file system image. Also on the ROM is ldlinux.sys and syslinux.conf which loads the linux kernel and sets up the RAM drive.

This paper is not intended to be a detailed step-by-step howto but a guide that documents the process I used to create my system. I will go into detail in some areas that I had trouble with in effort to help others. I will lightly touch on some topics that I think is common knowledge or with minimal research can be worked out. This paper outlines a systematic, methodical and repeatable process of building stable Asterisk servers to produce an embedded type operation.

I used a compact flash memory module to act as the ROM.
http://www.logicsupply.com/product_info.php/cPath/47/products_id/241
I used this Flash to IDE adapter because it sits in the IDE slot on the motherboard, no need for a cable. The mobo BIOS sees the flash module as a regular hard drive, so no trick flash driver or special load process is needed to operate the ROM in this fashion. I've seen others use compact flash adapters that mount on the case where the flash module is accessible from the outside of the case when everything is buttoned up. This would be desirable to switch out images quickly without the need of opening the server case to access the flash module.

You can use any motherboard and case you want. I like the small form factor ones so I chose to use a VIA EPIA M10000 NEHEMIAH but in hind site, I would probably pick a different chipset, maybe a mini-ITX P-4 or ASUS micro-ATX. I don't care for the south bridge on the VIA board, getting the sound to work the way I want is a pain but it works ok with ALSA, not great but usable.

I use www.pricewatch.com to search for cheap Compact Flash Memory Modules. Originally, I started with 512Meg flash but the compressed image turned out to be only ~25 Meg so 512 was a waste of money. I have since settled on 64 Meg flash, this gives enough storage for voicemail and config files with plenty of room left over. If you have many VM accounts or run a database that is large, figure out how much non-volatile storage you need for the system and get a flash module that will accommodate the size storage you require plus 30 Meg for the compressed file system image.

For the development process we need a hard drive and CD drive. I used an old 10 Gig hard drive and a CD drive I had lying around. When the development process is complete, the hard drive and CD drive will go away.

The development process is a lot of trial and error but the end result will be awesome and work great. So let's get started.

1. **Choose Hardware**
   a. Motherboard, follow any recommendations from the Asterisk wiki, this is really inconsequential, most modern mobo's should do fine.
   b. zaptel cards FXO, FXS, T1, how many ports do you need? This will determine how many PCI slots you require and ultimately determine how small or large the case needs to be.
   c. CD drive
   d. hard drive, at least 10 gig
   e. memory, at least 256 Meg, the system will run 100% from RAM, the RAM drive will be ~85 Meg, so this leaves about 170 Meg of regular RAM, if more than 3 or 4 channels will be active at the same time, think about using 512 Meg of RAM.
   f. sound card, if you don't need console or announcing calls to a PA system then turn this off. But I enjoy using the Dial command at the Asterisk console to help in trouble shooting remotely.
   g. Video, minimal requirements, this system runs headless, doesn't have to but mine does.

2. **Setup Development Environment**
   a. Motherboard BIOS, turn off everything that is not needed
      1. serial ports
      2. parallel port
      3. diskette drives
      4. firewire ports
      5. USB ports, unless needed for timing if no Zaptel cards will be used.
      6. assign any Zaptel cards their own IRQ in the PCI settings
      7. turn off boot error halts, keyboard and such

   b. Attach hard drive and CD drive to mobo, use IDE channel 0. No need to attach flash module till the end of the system build. Partition the HD with at least four 1 to 2 gig partitions. I progressed through seven partitions. In Linux you can only

have 4 primary partitions, so when you partition the disk, 1, 2, 3, will be primary and 4 is skipped for 5, 6, 7 and 8 which are logical. You will end up with /dev/hda1,2,3,5,6,7,8. some use ~100Meg partition for /boot, this is not needed here because we are not running from the hard drive, I put lilo in the MBR.

c. Attach the CD drive, we will only use this to install the base system and to transfer the initial system build to another partition but it will come in handy, when you get into trouble, you can quickly boot up a KNOPPIX CD and repair any damage.
d. Setup a network connection, you'll need this to download packages and also SSH into the server from another machine.
e. Attach speakers, keyboard, and mouse, whatever else you need and start the coffee or pour some Crown (my preference).

3. Install Base System in HD Partition 1 (hda1)

a. I use Debian - Woody. I don't know why, I just like it I guess. Feel free to use whatever distribution you are most comfortable with.
b. I installed development packages and other packages needed to support Asterisk. There are a few How-To's on the web outlining how to install Asterisk on Debian and what packages are needed. Google "asterisk on debian" and many links will come up on the subject. DO NOT INSTALL ASTERISK JUST YET!
c. Once the base system is installed, and all dependencies are met, we now have to work on the kernel.

4. Compile New Kernel

a. There are 2 ways to compile a kernel, modular and monolithic. I used a modular approach. There are a bunch of kernel How-To's on the web also, search google accordingly and proceed.
b. Basically in Debian
    1. download new kernel in /usr/src, I used 2.4.26
    2. uncompress kernel source
    3. #ln –s linux-2.4.26 linux
    4. #cd linux
    5. #make menuconfig
    6. select kernel switches accordingly, I turn off everything I don't absolutely need to run the hardware. This is critical to reducing the size of the file system. If you have a lot of modules selected, then the /var/modules/kernel/files will be huge, to accommodate all the modules selected in the kernel. The standard kernel modules folder in a base system is around 35 Meg, after cutting out the fat in the kernel, my module folder went down to 5 Meg. I could probably reduce further if needed.
    7. save kernel file
    8. #make-kpkg clean
    9. #make-kpkg --append-to-version=.[some ID] kernel_image
    10. #cd ..
    11. #dpkg –i kernel-image-2.4.26.[some ID]_10.00.Custom_i386.deb
    12. run lilo? Yes
    13. reboot into new kernel, check that everything is still running as needed

14. Modify /etc/lilo.conf to include booting into hda2, hda3, hda5, hda6 and hda7.

5. Download Asterisk Files

a. You can use the cvs process to get all the Asterisk and Zaptel files.
b. Download whatever other software you need to compile and use with this server.
c. DO NOT COMPILE ANY SOFTWARE YET!

There is a global concept that may require clarification. When you go through this development process, you don't want to start from scratch when you screw up, and you will, many times, accept this now or do not proceed. You want a progressive process of tweaking, configuring hardware, setting up software, getting things harmonious between your OS, hardware, software and configurations. Once you're at a point where things are working, before you take another step forward which could ruin the whole distro or mess up the last few steps made, stop, copy what you have to another partition and re-start there. This is the reason for having many partitions on the hard drive. In this manner, you can always go back to a known working point instead of starting over completely. This also gives you an opportunity to experiment with different end products such as: hda5 has Asterisk setup with SIP extentions, HDA6 has Asterisk setup with FXS lines and hda7 has Asterisk setup as a PSTN Gateway with a T1 card.

6. Copy File System on hda1 to hda2

a. Boot up in KNOPPIX using the CD drive
b. mount /dev/hda1 and /dev/hda2
c. #cp -ax /mnt/hda1/* /mnt/hda2
d. This copies the first partition file system to the second partition file system with an archiving switch -ax so it's really a clone file system.
e. Now boot up into the second hard drive partition.
f. Once you booted into hda2, we need to do some maintenance
g. Edit /etc/fstab and change the first line to reflect /dev/hda2 instead of /dev/hda1
h. Throughout this process we will be switching back and forth between partitions, booting in and out of partitions. To help keep track of which partition I'm currently working in, I created a file in /root/hda1 on hda1, /root/hda2 on hda2 and so on. A quick look on /root will tell me where I am. Also keep track of /etc/fstab and change when you move the file system to the next partition.

7. Install Asterisk

a. compile Zaptel, load modules, use the modconf utility, this edits /etc/modules.conf to load the zaptel modules at startup
b. compile Asterisk, configure PBX to your needs
c. setup auto loading Asterisk at startup
d. in Debian, edit /etc/init.d/skeleton
   DAEMON=/usr/src/asterisk/asterisk
   NAME=Asterisk
   DESC="Asterisk PBX"
e. save file as asterisk in /etc/init.d/
f. #chmod 755 /etc/init.d/asterisk this allows the file to be executed

g. #update-rc.d asterisk defaults 99 this updates the init scripts and sets Asterisk to start after all other processes have been started. #man update-rc.d for more info.
h. Reboot into hda2 and test that the OS and Asterisk are working as needed. Tweak and repeat this process as many times as needed to be confident things are setup correctly, rinse and repeat.
i. If something goes terribly awry and hda2 is beyond recovery, boot back into KNOPPIX and go back to step 6.

At this point we have Asterisk up and running on a full distribution of Debian (or whatever distro) and life is good. All the config files are happy, the phones are working, the RTP packets are flying, IAX is connected to the world and the PBX starts up on boot. Now let's start breaking things, yea! Remember, hda2 holds a working system and all packages intact to develop and compile new software. Use this file system in the future to compile in new versions of Asterisk, patch other software, upgrade kernel and so forth.

8. Copy File System on hda2 to hda3

a. There is no need to boot back into KNOPPIX at this point, you can boot into hda1
b. #mount /dev/hda2 and /dev/hda3
c. #cp –ax /mnt/hda2/* /mnt/hda3
d. boot into hda3
e. Do some maintenance as we did in step 6
f. #mv /root/hda2 /root/hda3
g. edit /etc/fstab change /dev/hda2 to /dev/hda3
h. reboot into hda3 and verify all is well

9. Strip Down the File System

a. At this point, out file system is about ~550 Meg. We're going to go on a diet and do some serious cardio to get the fat out and trim this system down to about ~75 Meg. This is also the point at which we will probably break something and have to go back to step 8, but that's ok, at least we don't have to start from scratch.
b. dselect is your friend. In dselect, deselect (no pun intended) all packages that are not required by the base system, almost. Scrutinize the dependencies when you deselect a package and make sure you don't accidentally break dependencies or uninstall a package needed for Asterisk like sox. I would give a list of packages safe to uninstall but it really depends upon your server. This is the worst part of the process and the most critical. Take your time, you can uninstall a little at a time, reboot and check to see that things are still working as needed.
c. Make sure to delete any deb files in dselect.
d. #df –h this gives you a tally in Meg of how much space is used on the partition. Man df for more info.
e. #du –sh */ this gives you the size of folder in the pwd
f. #du –sh /* this gives you the size of folders in the root directory
g. #du –sh /var/* this gives you the size of folders in the /var directory. Man du for more info.
h. Basically, I used the df and du commands to guide me in finding bloated directories and folders that I could safely

delete without having any system problems. When I came across folders and files that I wasn't sure if I could safely delete, I searched the web to find info on what programs they belonged to and deleted them or left them alone accordingly.

i. When I deleted folders or files from hda3, I didn't remove them totally, I moved them to another partition. I used hda7 as the repository and maintained the file system structure so I would know where the files came from. For instance I moved /var/cache/apt to /mnt/hda7/var/cache/apt. This particular one relieved 7.5 Meg from hda3.

j. Here are some folders to look at and eliminate:
   /var/lib/ remove kernel folder you're not using
   /var/lib/apt
   /var/lib/asterisk remove any mp3's you're not using
   /var/lib/dpkg
   /usr/share/doc
   /usr/share/locale
   /usr/share/man
   /usr/share/zoneinfo
   /boot/ remove old kernel files
   /usr/src/* remove all

k. Remove files and folders a little at a time and reboot into hda3 periodically to test file system for problems. You can always put back files from hda7 if there is a major issue.

l. If something goes terribly awry and hda3 is beyond recovery, boot back into hda1 and go back to step 8.

m. After you get the file system down to about ~75Meg, you can surely go further and all is still working, then we can proceed with preparing the file system to be compressed and boot from the compact flash.

10. Compress the hda3 File System

a. I found an excellent How-To at http://silent.gumph.org/content/4/1/011-linux-on-cf.html It goes into great detail on how to boot a Linux distribution from a compact flash. You can really follow step by step and forget about the rest of this paper, I just added some notes that helped me through the rest of the process. I changed the procedure a little bit.

b. I didn't make my partitions large enough to create the compressed file system on the same partition as the file system I wanted to compress so I moved that part of the procedure to a new partition, hda5. I didn't have any real problems with the "Silent PC" procedure except I had to keep track of where the temp file system actually was in relation to hda3

c. Before you compress the file system, edit /etc/fstab and change /dev/hda3 to /dev/fd2 so the kernel doesn't try to mount the compact flash as /. Remove any lines for swap disk, CD drive and any other drive paths that may have been configured in during setup. To keep Debian from trying to fsck the disk, change pass column to 0 instead of 1. Put in a drive path for the flash disk [/dev/hda0 /mnt/hda0 vfat defaults 0 0]

d. Copy any config, log files and voicemail folders from hda3 to a temp folder on hda5, maintain file system structure. Later we will copy these files to the compact flash and this will

serve as our NVRAM. So review what is going on with your setup and collect all the files you want to write to during normal operation of the system and copy them over to the temp directory on hda6.

e. Now go back into the temp file system on hda5 and sim-link all the files and folders you copied to the temp directory on hda5 (your NVRMA files) to /dev/hda0/path. Example using /ect/asterisk/extensions.conf:
#ln –s /etc/asterisk/extensions.conf /dev/hda0/etc/asterisk/extensions.conf (on one line)
Now when the system is running in RAM and you want to make a change to extentions.conf, proceed normally and the file will actually be written and read from the flash (acting as NVRAM) so when the server is rebooted, the changes will still be there.

f. Continue with the "Silent PC" procedure and compress the file system on hda5, now you should have your compressed file system ~23 Meg, lite.gz. Now you can proceed and prepare the compact flash.

11. Prepare the Compact Flash

a. Shut down the system and install the IDE adapter and compact flash memory module.
b. Boot into hda1 or hda2
c. Again, this is outlines in the "Silent PC" procedure. Syslinux the compact flash.
d. Copy the compressed file system lite.gz from hda5 to the compact flash root partition.
e. Copy all the NVRAM files from the temp folder from hda5 to the compact flash, maintain file system paths (so the files match the sim links you created).
f. Create a syslinux.conf file on the compact flash, outlined in the "Silent PC" procedure. Make sure your RAM Disk size is a little greater than the file system size.
g. Copy the kernel image from /mnt/hda3/boot/vmlinuz-2.4.26.[some id].whatever to the compact flash root partition. Change the name of the kernel image to just vmlinuz

12. Boot Your New Embedded Asterisk Server

a. I would leave the hard disk and CD drive connected for a bit longer and just change the boot sequence in the mobo BIOS to test the compact flash. If your hard disk is in IDE0 and your compact flash is in IDE1, change the BIOS to boot IDE1 first.
b. Turn on the server and you should see syslinux uncompress the file system into a RAMDISK and start booting the kernel like normal. The only noticeable difference is it goes really fast. If all is working, your home free. If not, go back a step or two and figure out what went wrong and correct.
c. Shut down the server and remove the hard drive and CD drive
d. Move the compact flash IDE adapter to IDE0
e. Reboot into the new world. Cycle power a few times and be amazed that the server boots up every time and still works.