EMBEDDED IN THIN SLICES

Linux System Configuration (Part 2)

Linux Application Design

The Linux system can be configured in many ways. This article examines some options, including BusyBox, web servers and server-side support, mail servers, time server interfaces, secure networking, graphics libraries, and browsers.

By Bob Japenga (US)

Let's start with some definitions. In Part 1 of this article series I talked about configuring the kernel. The kernel is the nerve center of a Linux system (i.e., the OS). But the OS needs applications to do real work. You need web servers, user interfaces, I/O processing, and other logic encapsulated in applications to do the actual work.

After you configure and build the kernel, applications need to be selected and then built for your particular kernel and processor. Linux provides many open-source applications to help fulfill your design. As a system designer, you need to choose which applications you want to go into your embedded system. Of course, if your processor is infinitely fast and has an unlimited amount of RAM and flash memory, you can just skip this article. Others may need to learn more.

Applications are put together in a "package." Packages are assembled in what is called a "distribution." You may have heard of Ubuntu or Fedora. These are collections of packages (one or more applications) and a kernel put together as one unit called a Linux distribution. For example, my company recently started a project for a customer and wanted to know what they were using for Linux (i.e., tools being used, kernel version, and applications running). When the customer told us they were using an Ångström distribution, we knew what they had (more on Ångström next time). In this article, I want to discuss the kinds of packages you may want to include in your system.

WHAT ARE YOUR OPTIONS?

One of the reasons my company uses Linux is because of the vast array of packages available. Several times our customers have asked: "Can you …?" You can fill in the blank. And many times, because of the packages available, we can say yes.

One example involved creating a virtual display. One of our customer's customers wanted to control our embedded device from a PC as if they were in front of the device's touchscreen on the device. Voila. An open-source package has already been designed to do just that.

Let's look at some of the options you have as a system designer in open-source packages for embedded Linux.

BUSYBOX: THE SWISS ARMY KNIFE OF PACKAGES

BusyBox is one package that is used in a lot of embedded systems that are concerned about resources. BusyBox puts a lot of (more than 250) tiny UNIX/Linux applications into one small executable. For example, typical networking applications such as File Transfer Protocol (FTP) and Telnet are included in BusyBox. Common file manipulation applications such as "cp" (copy a file), "cmp" (compare), and "more" are included.

BusyBox can be configured to delete some of these applications if space or simplicity is appropriate. We reduced BusyBox to just a handful of commands on one tiny Linux system. With some distributions you can configure BusyBox to build some "real" UNIX/Linux applications and delete them from BusyBox.

There is one thing you should be aware concerning BusyBox. BusyBox commands are not always exactly the same as the standard UNIX/Linux documentation. Generally they eliminate certain features rather than changing the core operation. The first time I used BusyBox on a project, I found that many things I wanted to do were not present. Some just required recompiling and rebuilding BusyBox. Sometimes they were not present at all. So beware.

WEBSERVERS AND SERVER-SIDE SUPPORT

Since you are using Linux, you most likely have a networked system. If you want to be able to browse to the device (we usually use this for configuring the system) or you want to respond to HTTP queries from a host, you will need a web server. You system uses a web server to deliver web content via the Internet. As a system designer you have a lot of options.

Apache—Apache is the grandfather of all Linux web servers. Approximately 60% of all web content on the Internet is handled by Apache servers. Apache is a completely full-featured web server with more options than I could describe in several articles. With this functionality comes significant resource usage. Some estimates recommend 400 MB of RAM and hundreds of megabytes of disk space. It is so configurable, hard statistics are not available about how much memory Apache uses. We used Apache once on an embedded system that had a gargantuan amount of memory.

Lighttpd—We have used Lighttpd on many of our embedded designs. Requiring less than a tenth of the resources of Apache, it supports almost all of the features we have ever needed to design web-based embedded systems.

Server-side languages—You need some kind of server-side support language to serve most modern webpages. If your device is going to serve anything but a simple webpage, you should include one or more of the following languages: PHP, Perl, Python, Ruby, Java, or Tcl. These languages are resource hogs on very small embedded systems.

On one project, using PHP would have taken 8 MB of flash memory. On this system, which had 16 MB of RAM and 32 MB of flash memory, we served webpages with a C language program. When we started the project I was warned that serving webpages in C was going to be difficult. But we just didn't have the flash memory resources. Now I can truly say that the server-side software was ugly. Most of you won't be limited like we were. However, if you are somewhat limited you should choose one language you will use to serve webpages.

MAIL SERVERS

We did a Linux project replacing an embedded 8088 with a fax modem on an industrial control. Whenever the system needed to sound an alert, it would fax the details to everyone on the list. We convinced our customer that e-mailing would be a much better way to go. Adding e-mail support to Linux is trivial. As with setting up a web server, with mail servers you can go with the standard full-featured mail server (called the SMTP server) or there are various small-footprint mail servers.

TIME SERVER INTERFACE

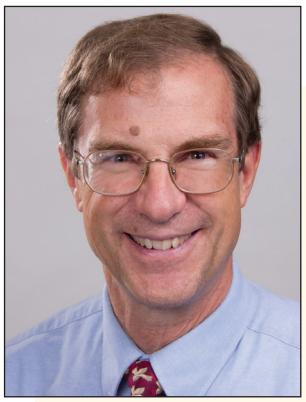
One of the common problems we face when we create "headless" designs (i.e., when there is no user interface) is how to set the time and date. There is a Linux package that solves your problem. The Network Time Protocol (NTP) enables your device to continuously connect to a time server and keep your time and date current. It works much like what you have on a Windows PC.

SECURE NETWORKING

When we started designing networked systems, security was not at the top of our priority list. Now, customers no longer want Telnet (an unsecure way of connecting to your device). They demand Secure Shell (SSH), which is just Telnet over a secure network. By incorporating the OpenSSL package (which we are all familiar with because of the HeartBleed virus), you can add a secure layer to all of your network communications. Generally, adding OpenSSL is not enough since you will have to provide some mechanism to handle the encryption keys (encapsulated in certificates). Again, the Linux open-source community will provide that for you.

GRAPHICS LIBRARIES

Many Linux systems that we build have some sort of GUI. With any graphical interface, you need the graphical libraries to easily design a snazzy user interface. The forerunner of Linux graphics libraries is the X.Org Foundation's X Window System. The first UNIX system I ever worked with included this library. It is large and sometimes



ABOUT THE AUTHOR

Bob Japenga has been designing embedded systems since 1973. In 1988, along with his best friend, he started MicroTools, which specializes in creating a variety of real-time embedded systems. With a combined embedded systems experience base of more than 200 years, they love to tackle impossible problems together. Bob has been awarded 11 patents in many areas of embedded systems and motion control. You can reach him at rjapenga@microtoolsinc.com.

painfully slow. GTK+ is a faster and smaller-footprint option. We have used it on several systems and were well satisfied. QT is a smaller and faster library, but my company does not have any experience using it.

We have done a few designs where the user interface is a browser interface. This requires the system to have a browser and a web server. Instead of designing widgets and gadgets from scratch, draw upon the vast number of webpage designs in the world and point your browser to the device's link local address.

BROWSERS

Whether you browse to your own web server (as described earlier) or enable your user to browse externally (we have only done this on one embedded system), you need to choose at least one browser package. The common options we have used include Chromium (why it is not called Chrome is an interesting story for another day), Firefox, and Midori. Chromium is uses a lot of resources, but it is well known. We found Firefox to be unstable on the one platform on which we ran it. Midori is a fast and lightweight option. It is part of the Raspberry Pi distribution. We have limited experience using it.

There is a commercial browser specifically designed for small embedded platforms: NetFront and NetFront NX. It is said to be "extremely" optimized for embedded platforms. Nintendo 3DS and Playstation 3 Internet browsers use this platform. Unfortunately, I do not have licensing information or any experience using it.

MISCELLANEOUS PACKAGES

There are a wide range of other packages to suit many different needs. cURL is a powerful command-line tool for sending all kinds of data over the Internet over a variety of protocols. We have used cURL in a several settings where we wanted to communicate over the Internet via a shell script rather than programmatically.

How can I discuss packages without talking about package managers? A mechanism for managing the versions of your packages is built into the fabric of Linux. It seems as if each distribution has its own package manager with its own nuances. Simply stated, it is a powerful way to safely and automatically update your applications and packages in the field. That said, we have never used a package manager to control the updating of our proprietary code on an embedded system. It seems as if our customer's requirements for security, the website, version control, or something else prevents us from using a package manager in our embedded system.

Perhaps you have successfully used a package manager on an embedded system. E-mail me what you did and how you did it. We have tried a few times to use them to prevent us from reinventing the wheel for software updates, but we have always been stymied.

Another package we use on some of our systems is a database programming language called SQL. If your design requires a distributed and relational database, you may want to look into SQL. Several different SQL packages are available. We have used MySQL, which is a full-featured implementation and SQLite, which is a somewhat "lite" version of MySQL. By adding this package to your toolkit you add all of the power of a managed relational database.

KNOW YOUR OPTIONS

Configuring Linux involves configuring and building the kernel, selecting the packages you want installed in your system, and then building them into a file system image. This article provided you with information about some of your options (in thin slices of course). Part 3 of this article series will discuss the options you have for building this vast array into a workable Linux system.

circuitcellar.com/ccmaterials

RESOURCES

The Apache Software Foundation, www.apache.org.

The Ångström Distribution, www.angstrom-distribution.org.

BusyBox, www.busybox.net.

cURL, http://curl.haxx.se.

The Fedora Project, https://fedoraproject.org.

The GTK+ Project, www.gtk.org.

Lighttpd, www.lighttpd.net.

Midori, www.midori-browser.org.

MySQL, www.mysql.com.

The Network Time Protocol (NTP), www.ntp.org.

The OpenSSL Project, www.openssl.org.

QT Project, http://qt-project.org.

SQLite, www.sqlite.org.

Ubuntu, www.ubuntu.com.

Wikipedia, "Heartbleed."

X.Org Foundation, www.x.org/wiki.

SOURCE

NetFront
ACCESS Co., Ltd. | www.access-company.com