___The use of Linux for dedicated servers.

___Chel van Gennip, HISCOM BV Leiden, The Netherlands <chel@hiscom.nl>

1 Introduction.

This paper will focus on the use of Linux for dedicated systems. Dedicated systems have a well defined task and can be tuned to perform this task. Current technology, with low cost and fast networks and processors enable us to work in a distributed environment. Using a distributed system with dedicated servers for well defined tasks enables us to reduce the costs of system administration and maintenance. Normally there are a number of tasks that can be isolated very easily. In many cases the administration of such a task can be quite simple if the task is isolated. If we combine tasks with other tasks the total cost can be less, equal, or greater than the sum of the separated tasks. A typical example of isolating dedicated tasks are routers. Although many Unix systems have routing capabilities, dedicated routers are used by almost everyone. The reason for this choice is obvious, sharing this task with other tasks will cost extra money. Using an inexpensive OS on inexpensive hardware enables us to isolate tasks in a cost effective manner. Linux is a perfect example of a high quality, low cost OS that runs on low cost hardware.

2 Technical aspects Linux compared with other systems.

From a technical point of view a number of aspects are important. For dedicated tasks the requirements differ from task to task. If we consider the task of booting and serving X-terminals, we see that this task can easily be isolated. It does not need an administration of user accounts. The only task is to react to boot requests and to have static images of fonts, executables etc. to be accessed by NFS or TFTP. Combining this task with other tasks would imply we have to install NFS and TFTP on other systems. It also would imply we would have to give the administrator of this task extended privileges on other systems. Isolating this task will simplify system administration. On the other hand a large number of systems in a distributed environment increases the complexity of system administration, even if the systems are almost alike.

Before I will explain the specific advantages of Linux, I would like to emphasize the fact that from a software point of view Linux is a Unix system, with all the benefits of Unix as an open standard. Building on Unix leaves the freedom to choose for an implementation to fit the specific needs. The first benefit of Unix systems in general is the alikeness. Because many Unix system manufacturers emphasize the particular benefits of their implementation, one could think the systems differ a lot. In fact the systems are different, but mainly at the implementation level. The differences at the API level are quite minimal, one should notice that the differences at the API level between MS-Windows 3.11 and MS-Windows 95 (only two succeeding versions of a non Unix OS from the same vendor) are much larger!

Linux as an OS for the dedicated systems has a number of benefits.

The first benefit is the fact that the Linux system has an autodetection mechanism for most hardware. Also a module concept for device drivers is supported, so new hardware can be easily added even to existing kernels. This enables us to use the same software configuration for a number of systems, even if the hardware is not quite the same. In the market we see a lot of new products for e.g. ethernet, SCSI, disks etc. Some of these components are available only for a short time before they are replaced by newer better products, sometimes products are available less than a year. To deliver a system one has to use these new hardware components. If an OS has to be configured for specific hardware it is more difficult to manage.

A second benefit of Linux is the small footprint. Although standard distributions of Linux already are small, it is quite easy to strip the system to a minimal system. A minimal system has major benefits for support, security etc. In modern systems we see an explosive growth of software components. It many cases it is quite difficult to avoid the effects of this explosion. From my experience I know how drastic this can be. A small DB oriented application that would fit in 300kb in 1992 (only 25kb of this was my own code), needed 900kb when recompiled in 1994 and over 2Mb in 1996. A side effect of this growth were some incompatibilities, so we had to adjust the manuals. For dedicated systems with a well defined task one does not want to have to upgrade unless the task definition changes. As an example of how small Linux can be: the Linux system for our communication server can be installed from one single floppy disk.

A third benefit of Linux is the availability of the source code. This enables tailoring the OS to fit the dedicated task better. Although not a real requirement, it can be useful. Furthermore the availability of source code adds extra debugging facilities. In one case we noticed some problems between different TCP/IP protocol stacks. Everything worked fine, but in some cases there were 500ms latencies. After we added some debug statements in the kernel we were able to follow exactly the sequence of events: invocation of the API's, packets coming in and out, TCP/IP state changes.

3 Commercial aspects of Linux compared with other systems.

If dedicated systems are sold as a complete system, hardware, OS and application in one package, selling Linux for the OS is not so difficult. It is only one of the components of the complete product. From a customer point of view it is attractive to buy a complete product, performing a task without having to deal with individual components. In my opinion this part of the market is quite suitable to market this free software. As stated the complete product includes three major components: the hardware, the OS, and the application. For the hardware only industry standard components are used, these components or their replacements are well available and will be in the future. The OS is an intermediate layer between this standard hardware and standard API's. The important factor of Linux as the OS is the availability of the source code. With the source code one has the ability to guarantee the delivery and the support of the same product for a number of years. It is a good practice to freeze the production tools of a product at the moment of market introduction. With Linux one is able to freeze the OS, the compilers etc. at this moment. For a number of years bugs can be fixed and minor adjustments to support the product can be made. In general support on commercially available OS'es and tools is limited in time. If one seeks for support one has to upgrade to the current version. This upgrade can however be quite costly and also require extra costs for hardware and labor!

4 How to support Linux.

Especially in relation with free software, the question is raised: "How do I support this system?". To support a system one should consider the following questions:

- Is the system well documented?
- Is the system easy to manage?
- Can I hire personnel with expert knowledge of the system?
- Can I get drivers for specific hardware?
- If an error is found, especially an error only showing under specific conditions, what can I do to get the problem fixed within a certain time?

We found out that the answer on all of these questions is positive.

Documentation.

First of all Linux complies with standards like Posix and X-Open, so all of the interfaces normally used are very well documented. A lot of books with information about Linux are available from various publishers. Information in electronic form is available from the LDP. Furthermore FAQ's and HOWTO's are available, even the source of the system and all of the tools is available. The fact al lot of information is also available in a electronic form improves the access, either by HTML browsers or even access with the old Unix tools like grep and vi. No information is hidden.

System management.

Normally two versions of Linux are available, a stable version and a development (alpha and beta test) version. Of the stable versions an number of distributions exist. These distributions deliver normally a stable and complete Unix system that is easy to install and to manage. Currently the Red Hat distribution is good, but other distributions deliver good products as well. For dedicated systems the Linux system can easily be stripped down to reduce the effort of maintenance.

Experts.

Experts normally need more information about the internals of a system than normal users. Often the knowledge of experts is based on experience with the external behaviour and reverse engineering. With Linux there is no hidden information, and the behaviour of the system is discussed openly in newsgroups and mailinglists. Unlike other systems, there are no books about the undocumented features of Linux. A good starting point to become a Linux guru is a normal education in Unix and C. From this point all documentation needed is freely available. As a result of this experts can be easily trained and therefore are widely available.

Drivers.

Getting drivers for specific hardware can be a problem with any Unix system. Most hardware is supplied with drivers for MS-windows. A lot of hardware vendors do not even consider supplying unix drivers. For some hardware no specification of the hardware interface is available and therefore it is almost impossible to create drivers for this hardware. The Linux system supports a lot of hardware. By now the number of Linux systems is high enough to convince hardware manufacturers to supply drivers for Linux. As with any unix system one should verify the support of specific hardware.

Fixing bugs.

Every system has bugs. If I want to deliver a dedicated system I want to be sure none of the bugs show up during normal use. Sometimes the definition of what is a bug is ambiguous. The TCP/IP protocol for example leaves some aspects open to the implementor. As a result of this sometimes some combinations of network components show some degradation in performance. With commercial systems often it is difficult to get bugs fixed in general. Getting such vague bugs fixed is almost impossible. With Linux my experience on this topic is limited, because no real bugs were detected. We encountered however one case of a vague TCP/IP bug. After trying to locate the bug with normal network monitoring tools without success, we finally found the bug by adding some debug statements in the TCP/IP stack. Finding and fixing the bug was simple.

5 Case 1, a communication frontend.

Our first use of Linux for a product was our communication frontend processor. Within the HISCOM Hospital Information System, it has always been a strategy to offer acces at the working location for anyone. Therefore a huge number of terminals and workstations are connected to the HIS. To handle the load of all these stations communication frontends were introduced in the late 70's. Over time more and more handling of the I/O of the stations was implemented in these communication frontends. For the 5th redesign of the hardware platform for the communication frontend a number of choices were made:

- The use of industry standard components.
- A Unix system to support the standard software functions.
- Plug compatible with previous versions.
- No specific maintenance or administration of these systems.

To get a system free of maintenance and administration is was decided to boot the system from the network. On the system itself there is only temporary storage of data. The first problem was that some of the hosts only supported the MOP protocol. This protocol is not available for a lot of hardware, so we had to implement it ourselves. To avoid problems with different types of hardware it was decided to first boot a unix system, and to start the MOP boot afterwards.

As a result of this design a Unix system had to be installed on the communication servers, here we had a conflict with the requirement "No Systemadministration". A compromise was found in giving all stations the same Unix system (binary), and to install new versions automatical. By this time the Linux was chosen because of the Linux special functionalities:

- Customisable boot.
- Autodetection of hardware.
- No serialization or keys for the Unix system.

Finally the Linux boot was installed using a read-only boot partition. In this boot partition two files are located at fixed places: the compressed Linux kernel and a compressed image of the root (ramdisk) filesystem. The combination Linux/Lilo supports booting Linux this way, however one small modification was needed to allow the root filesystem to be loaded from the harddisk into the ramdisk. After startup, Linux initializes another partition with mke2fs and mounts this filesystem for the storage of scratch files. At this point the MOP program is started to locate the host, download the communication frontend binaries and the configuration information. After this the boot is completed with the initialization of the TCP/IP stack and the start of the communication frontend. The Linux root filesystem is stripped to a minimum to let the system work and to support easy upgrades. The root filesystem is in total 2MB, one 3rd of this are diagnostic programs for the hardware etc.

Upgrade is simple, only a simple scriptfile containing a shell archive is sent to the communication server. The script replaces the two files in the boot partition and reboots the system.

The first installation is also very simple. A Linux system is booted from a floppy. The floppy contains the same Linux system and the same root filesystem. Besides that the floppy contains a small filesystem with installation scripts. If this filesystem is available this filesystem is mounted and the scripts in this filesystem are executed. To ease the job of installation a small patch in the kernel was applied to detect the keyboard. If a keyboard is present the installation should be started manually, if no keyboard is present the whole installation starts without human intervention. Normal installation of the Linux system is done in the factory. After the assembly of the system, the install floppy is put in the floppy drive and the system is switched on. Within 2.5 minutes the installation is completed.

Problems encountered.

Although the Linux system did not impose any problems, we did encounter some problems that are worth mentioning.

Although we could use industry standard components almost everywhere, we were not able to find a proper housing to meet our needs. We decided to design a special housing.

Some motherboards are not designed to work without a keyboard: "No keyboard detected, press F1 to continue".

Some motherboards startup in sleep mode, disabling the powersupply.

Status.

After thorough testing on or in-house development systems, the communication servers are installed in the hospitals now for some time. These systems function properly.

6 Case 2, a tape server.

When porting our HIS from our proprietary operating system BOS running on VAX computers to Unix systems (on e.g. Inter processors) we encountered a problem with the handling of tapes. In our HIS tapes are used as a log device to log all transactions in the system. After a crash, due to hardware or software inconsistency the system is reset. The first actions on the tape last used as log device are: skip forward to end of tape and count, go one block back and read. Inspection of Posix and several unix implementations showed there was no standard way to do so, in many cases there was no way at all to do so.

Here we had the choice between writing our own devicedriver for a number of unix systems or to find a creative solution for the problem.

The creative solution was to install a tape server on a dedicated unix system, and to access this tapeserver via the network.

Of all the unix systems inspected Linux turned out to have the most complete interface for tape devices. As we had previous experience with Linux for the communication servers, we decided to use Linux.

The tape server not only solved the interface problem, but also gave a lot of extra functionality:

- A number of systems now could share the tape devices.

- The network link between host and tapeserver enabled the remote placement of the tape unit. This is a major advantage for a log device.

- With the distribution software of the tapeserver it was quite simple to implement the same functionality on files on disk: "virtual tapes". With these virtual tapes many tapes can be mounted at the same time without requiring many physical tape-units.

From a operating point of view it sometimes is sufficient to store the data in a safe place outside the computercentre for a period of 14 day's. With the tapeserver in a safe place on a distance of the computercentre, there is no need to copy the virtual tapes to real tapes and to store them in a safe place. So in many cases the tapeserver even works without tapes!

Setup.

Base of the tapeserver is the same hardware and software as the communication server. At boot time, not only a scratch filesystem is mounted, but also a filesystem for the storage of the virtual tapes. For a more secure storage of these virtual tapes, this filesystem is mounted with the sync flag, so all data is stored on disk immediately.

Status.

The tapeserver now is in alpha test and functioning well. Because of the extra functionality many candidates for betatesting are available.

7 Conclusion.

After one year of experience with Linux in a commercial environment our conclusions are:

- Linux is a stable Unix system.

- Linux supports all the standard interfaces needed. No porting problems were encountered.

- Linux also implements some useful non-standard functions (raw-ethernet i/o, tape i/o).

- Most customers know Linux, there is no hesitation to use Linux for these dedicated systems!

- Linux is a fast Unix system, execution times for system call's and rescheduling are good, also the use of memory is very efficient.

At least for dedicated systems Linux is a good choice!

Author information

Chel van Gennip has 20 jears of experience with information processing. Currently is the leader of a development team at HISCOM BV, the market leader for hospital information systems in The Netherlands. Furthermore he is an active promoter of open systems as treasurer of the NLUUG, boardmember of Europen, one of the founders of the NLLGG (Dutch Linux User Group) etc.