EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH
ORGANISATION EUROPÉENNE POUR LA RECHERCHE NUCLÉAIRE

CERN - ST Division

CERN-ST-2001-012
30 January 2001

LINUX IN INDUSTRIAL CONTROL SYSTEMS

T. Riesco and E. Cennini

Abstract

Today the *Linux* operating system has become a real alternative for industrial control systems. *Linux* supports all layers in control systems starting with Real-Time or embedded systems for data acquisition, following with treatment, storage, communication and data adaptation, and finally, with supervision and user interfaces. In the last years the *Linux* development has grown being incorporated in several industrial systems demonstrating high performance, availability and stability for complex processes in chemical, automobile or petrol industries. In many of these industries *Linux* architectures have been tested and validated successfully. The new CERN policy supporting *Linux*, as well as the emergence of cheap and robust *Linux* solutions, motivates its implementation in our safety control and supervision systems in the near future.

Presented at the 4th ST Workshop
Chamonix, France, 30 January - 2 February 2001

1 HISTORY

Linux is an operating system that was initially created as a hobby by a young student, Linus Torvalds, at the University of Helsinki in Finland. Linus had an interest in Minix, a small UNIX system, and decided to develop a system that exceeded the Minix standards. He began his work in 1991 when he released version 0.02 and worked steadily until 1994 when version 1.0 of the Linux Kernel was released.

The current full-featured version is 2.2 (released January 25, 1999), and development continues. Linux is developed under the GNU General Public License and its source code is freely available to everyone. This however, doesn't mean that Linux and it's assorted distributions are free -- companies and developers may charge money for it as long as the source code remains available. Linux may be used for a wide variety of purposes including networking, software development, and as an end-user platform. Linux is often considered an excellent, low-cost alternative to other more expensive operating systems.

Due to the very nature of Linux's functionality and availability, it has become quite popular world-wide and a vast number of software programmers have taken Linux's source code and adapted it to meet their individual needs. At this time, there are dozens of ongoing projects for porting Linux to various hardware configurations and purposes.

2 LINUX AT CERN

The usual CERN UNIX environment (AFS, SUE, ASIS, compilers, CERNLIB, LHC++, LSF...) has been heavily tested in the past years and works very well with only very few exceptions.

Since the 15th of February 2000, the CERN customised version of Red Hat 6.1 is certified. The advantages of this version compared to the previously certified one (5.1) are multiple:

- kernel 2.2 with many enhancements: SMP, NFS...
- official AFS from Transarc
- better hardware support: network cards, laptops...
- better graphical installer (Anaconda)

IT Division currently provides a basic infrastructure to assist and advise the CERN Linux community and discussions are underway with the CERN Linux Users Group on how to adapt this more closely with the users' needs.

3 LINUX FEATURES

- Multitasking: Several programs running at the same time.
- Multiuser: Several users on the same machine at the same time (and no two-user licenses!).
- Multiplatform: Runs on many different CPUs, not just Intel.
- Multiprocessor: SMP support is available on the Intel and SPARC platforms (with work currently in progress on other platforms), and Linux is used in several loosely-coupled MP applications, including Beowulf systems and the Fujitsu AP1000+ SPARC-based supercomputer.
- Multithreading: Has native kernel support for multiple independent threads of control within a single process memory space.
- Runs in protected mode on the 386.
- Has memory protection between processes, so that one program can't bring the whole system down.

- Demand loads executables: Linux only reads from disk those parts of a program that are actually used.
- Shared copy-on-write pages among executables. This means that multiple process can use the same memory to run in. When one tries to write to that memory, that page (4KB piece of memory) is copied somewhere else. Copy-on-write has two benefits: increasing speed and decreasing memory use.
- Virtual memory using paging (not swapping whole processes) to disk: to a separate partition or a file in the filesystem, or both, with the possibility of adding more swapping areas during runtime. A total of 16 of these 128 MB (2GB in recent kernels) swapping areas can be used at the same time, for a theoretical total of 2 GB of useable swap space. It is simple to increase this if necessary, by changing a few lines of source code.
- An unified memory pool for user programs and disk cache, so that all free memory can be used for caching, and the cache can be reduced when running large programs.
- Dynamically linked shared libraries (DLL's), and static libraries.
- Does core dumps for post-mortem analysis, allowing the use of a debugger on a program not only while it is running but also after it has crashed.
- Mostly compatible with POSIX, System V, and BSD at the source level.
- Through an iBCS2-compliant emulation module, mostly compatible with SCO, SVR3, and SVR4 at the binary level.
- All source code is available, including the whole kernel and all drivers, the development tools and all user programs; also, all of it is freely distributable. Plenty of commercial programs are being provided for Linux without source, but everything that has been free, including the entire base operating system, is still free.
- POSIX job control.
- Pseudoterminals (pty's).
- 387-emulation in the kernel so that programs don't need to do their own math emulation. Every computer running Linux appears to have a math coprocessor. Of course, if your computer already contains an FPU, it will be used instead of the emulation, and you can even compile your own kernel with math emulation removed, for a small memory gain.
- Support for many national or customized keyboards, and it is fairly easy to add new ones dynamically.
- multiple virtual consoles: several independent login sessions through the console, you switch by pressing a hot-key combination (not dependent on video hardware). These are dynamically allocated; you can use up to 64.
- Supports several common filesystems, including minix, Xenix, and all the common system V filesystems, and has an advanced filesystem of its own, which offers filesystems of up to 4 TB, and names up to 255 characters long.
- Transparent access to MS-DOS partitions (or OS/2 FAT partitions), WNT, Windows 95 support and FAT-32 via a special filesystem. HFS (Macintosh) and CD-ROM filesystem.
- TCP/IP networking, including ftp, telnet, NFS, Appletalk server, Netware client and server, Lan Manager/Windows Native (SMB) client and server, and Many networking protocols: IPv4, IPv6, AX.25, X.25, IPX, DDP (Appletalk), Netrom, and others.

4 LINUX IN INDUSTRIAL CONTROL SYSTEM

Today we needed software to control variety of devices connected via industrial buses or lab cards; to process, display and load/save/print acquired data. This software has to be really universal to enable to connect devices with very different control algorithms, data formats and data processing. It is necessary to control more devices simultaneously using one computer and have a remote control over TCP/IP.

All this features described are performed by Linux. Most Linux systems for industrial control run on PC platforms; however, Linux can also be a reliable workhorse for embedded systems.

Linux is generally considered to be very reliable and stable when running on PC hardware, particularly when compared to a popular alternative. Linux kernel is quite stable is the embedded systems. There are boards including some specific peripherals as well as the CPU.

The driver interfaces are well-defined and today we can found easily cards and drivers for CAN, Modbus, Profibus etc. Most drivers of a like kind are fairly similar, so migrating a disk, network or serial port driver from one device to another is usually not difficult.

It is common to make tables between Linux and other operating systems, I will not do because you can found many on the net. In my experience, Linux is at more stable as the big-name commercial operating systems with which I have worked. Plenty of war stories abound for any operating system and need not be repeated here.

The advantage to Linux is that the source code is available, well-commented and very well-documented. As a result, you are in control of dealing with any problems that come up.

If the industrial control system has a hard disk, the reliability of the file system comes into question. These systems are almost never shut down properly. Power just gets disconnected at random times. The experience has been very good, using the standard (EXT2) file system.

The standard Linux initialisation scripts run the fsck program, which does an excellent job of checking and cleaning up any dangling inodes. One change that may be wise is to run the update program at a 5 or 10-second interval instead of the default 30 seconds. This shortens the time window that data sits in the local memory cache before being flushed to disk, thus lowering the probability of losing data.

5 OUR EXPERIENCE IN ACCESS CONTROL

In 1999 we decided to change some parts of the access control system because them did not perform all the capabilities that we were expecting.

After considerate 2 operating systems: (WNT and Linux), we decided to use Linux. We considerate first the connection capabilities and the remote control of applications running in both systems: the answer was Linux. To implement some new drivers and specific controls, we needed to make code for this drivers, the clear answer was Linux since we had free of charge the compiler Gcc and a lot of utilities that in NT we had to pay.

Also we evaluated the connection to the ORACLE database at CERN, the answer was Linux. We had all the libraries and many applications made with. For NT we had, but we had to pay for.

Since we had 15 machines to install, we calculate the total price for licenses, patches etc and the answer once again was Linux. All the software for Linux was free of charge included a power database, MySQL.

6 FUTURE AND INVESTIGATION

6.1 SoftPLC

Inside the Access Control Section we are working in a Software PLC project. A Software PLC is a deterministic PC-based control that combines the discrete & analog I/O control found in "hard-PLC's" with the powerful data handling, networking, and open architecture of PC's.

The SoftPLC project is an attempt to produce a working PLC-like program running on Linux, with a GUI for configuring and viewing it's current state.

We intend to take advantage of the fact that we have an underlying Operating System (OS) and therefore we will use its features to try and make the modular, allowing for parallel code development.

A running Software PLC consists of modules running in an infinite loop, and accessing the common PLC memory/state. One module could be a PID loop, using two points for its inout, and a third point for its output. Another possible module could be a Siemens S7 emulator, that executes S7 code. Yet another possible module would be an I/O board driver, that would copy the state of some PLC points to its outputs, and copy the state of its inputs to other PLC points.

6.2 Real Time

A real time system can be defined as a **"system capable of guaranteeing timing requirements of the processes under its control"**.

Inside the Access Control Section we are investigating about Real Time, specifically the Real Time Application Interface from the Dipartimento di Ingegneria Aerospaziale - Politecnico di Milano.

We have made already a prototype and it is working well.

A real time system must be fast and predictable. Fast means that it has a low latency, i.e. it responds to external, asynchronous events in a short time. The lower the latency, the better the system will respond to events which require immediate attention. Predictable means that it is able to determine task's completion time with certainty.

Typically a real time system represents the computer controlling system that manages and coordinates the activities of a controlled system, that can be viewed as the environment with which the computer interacts. The interaction is bidirectional, say through various sensors (environment -> computer) and actuators (computer -> environment), and is characterised by timing correctness constraints.

It is desirable that time-critical and non time-critical activities coexist in a real time system. Both are called tasks and a task with a timeliness requirement is called a real time task.

Typically real time tasks have the following types of requirements and/or constraints:

1. Timing constraints. The most common are either periodic or aperiodic.
2. An aperiodic task has a deadline by which it must finish or start, or it may have a constraint on both start and finish times. A periodic task has to be repeated once per period. Most sensory processing is periodic, while aperiodic requirements can arise from dynamic events.
3. Resource requirements. A real time task may require access to certain resources such as I/O devices, data structures, files and databases.
4. Communication requirements. Tasks should be allowed to communicate with messages.

Concurrency constraints. Tasks should be allowed concurrent access to common resources providing the consistency of the resource is not violated.