OPERATING SYSTEM LINUX AS DEVELOPING AND RUNTIME PLATFORM FOR CONTROL SYSTEM OF PARTICLE ACCELERATOR

A.S. Chepurnov, Institute of Nuclear Physics, Moscow State University, 119899, Moscow, Russia
F.N. Nedeoglo, D.V. Komissarov, Department of Physics, Moscow State University, 119899, Moscow, Russia

Abstract

What operating system to choose is a common problem necessary to solve during development of control system for particle accelerator. It would be very attractive to use the same operating system as for development of software components of control system as for runtime both for real-time and non real-time applications. Remote boot-loading capabilities, mature software development tools, and other advantages of Linux make it very useful for development of software components of modern control system for particle accelerators. Our experience lets us to propose application of Linux operating system as for non real-time high levels as for real-time middle level of control system. Real-time extension of Linux - RTLinux is used for real-time applications. This choice allowed us to utilize effectively performance of x86 compatible computers, to have uniform hardware and software environment. Linux operating system have been used on different levels of control system of electron linacs as for gradual upgrading of the old control system as for development of new control system for newly designed compact accelerator. Software drivers for CAN-bus adapters together with DeviceNet library have been developed under Linux.

1 INTRODUCTION

Linux operating system (OS) has become more and more popular as dynamically developing modern Unix-like OS. Many of famous hardware and software manufacturers such as IBM, SGI, Oracle, National Instruments and others have support this OS.

Linux brings an attractive alternative to operating systems traditionally used in the field of accelerators control. The basic advantages of Linux OS against traditional operating systems are the open developing model and free of charge. There are thousands of developers and testers that take part in the developing of the OS. Open source availability allows anyone to implement new features and new applications not from scratch but based on the existing code.

Wide spectrum of compilers, high-level programming languages, cross-platform libraries, developing tools and real-time extensions allow to use Linux at all the levels of the control system as development platform as runtime environment.

To upgrade old fashion control system of a race track microtron (RTM) injector we have used Linux for the first time in 1996 [1,2]. Single PC under Linux replaced five old-styled LSI-11 minicomputers. We ensured that Linux is a stable and reliable OS and it could be used in the control systems more widely after two years of successful exploitation of the system. So, the Linux was chosen as uniform developing and runtime platform for our new control system of “industrial style” which is under final development now together with newly designed high beam current electron continuous wave linac [3].

2 LINUX BASED CONTROL SYSTEM

2.1 General layout of control system.

The control system for the linac consists of two classical levels (Figure. 1) - non-real-time top level and real-time front-end level. Both levels use Intel-compatible computers.

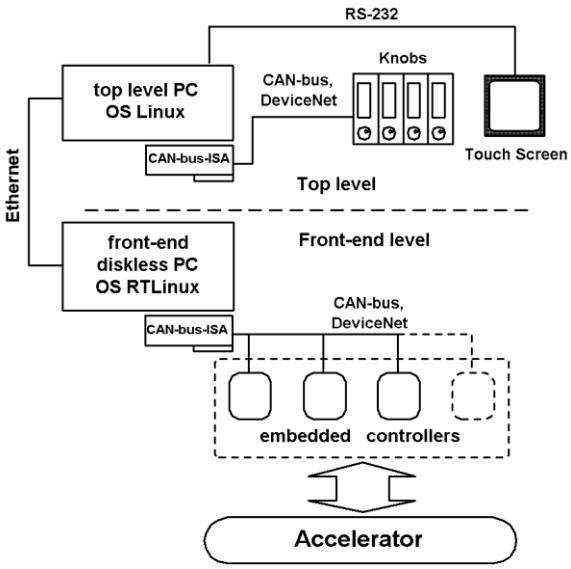


Figure 1. Layout of the control system.

Front-end level supports fast control algorithms, hardware locking, fast feedback loops, and signal conditioning hardware. The basic functions of top level are bootloading of front-end computers, supporting of man-machine interface and providing necessary database capabilities.

Top level and front-end PCs communicate via Ethernet fibre optic link, to provide galvanic isolation between levels of control system.

CAN-bus fieldbus is the second key component of our control system. DeviceNet was selected as well-defined and consistent version of CAN application layer (CAL). The CAN-bus-ISA adapter is installed into PC to provide computer access to CAN-bus network. Home designed CAN-bus-ISA adapter for PC is based on Philips SJA1000 CAN-bus controller and provides fast access to the CAN controller by direct memory mapping.

2.2 Top level.

PC compatible computer runs under Linux 2.2.x at the top level of control system. To improve interaction between operator and accelerator we use knobs-type modules and plan to use touch screen. The module constructed around single-chip micro-controller consists of encoder, two lines of high brightness LCD, and four keys with corresponding LEDs. The modules (up to four in our case) could be assigned dynamically with any adjustable or controllable parameter of accelerator. The modules communicate with top level PC via CAN-bus.

2.3 Front-end level.

At the front-end level the diskless PC runs under Linux 2.2.x with RTLinux 2.2a (real-time extension of Linux). Diskless PC boots operating system via BOOTP protocol from the top-level PC. Then root file system is mounted with the help of NFS protocol.

CAN-bus-ISA adapter installed in front-end PC controls embedded controllers that belonged to a family of "Smart Devices"--intelligent controllers which support functions of real-time digital feedback control, data acquisition and processing [4].

3 SOFTWARE DESIGN ISSUES

Control system software was developed with the GNU development tools only under the Linux OS. The GNU C compiler (GCC) with standard GNU C library (GNU LIBC), GNU Debugger (GDB) and Concurrent Version Control system (CVS) were used.

3.1 Software support of CAN-bus.

CAN-bus is used at both levels of control system. To ensure high level of compatibility and ability to use as home made as "off the shelf" components DeviceNet high level protocol for CAN-bus was used [5]. We did not find neither appropriate support of CAN-bus nor DeviceNet software stack under Linux. So, DeviceNet compliant protocol stack and different CAN-bus device drivers were developed [6].

To support CAN-bus-ISA adapters under the Linux kernel a Linux kernel mode driver has been developed. Application software interfaces with the driver by means of writing and reading messages to or from special character device file. The driver can handle up to four CAN-bus-ISA adapters simultaneously.

The DeviceNet compliant protocol stack provides Slave capabilities for various types of front-end controllers and Master capabilities for host personal computer (PC) running under Linux OS.

To support the protocol stack for different hardware platform it was developed in the form of software library (Figure2) and consists of the following components:

- the library kernel,
- the module with system dependent functions,
- the module with interface to CAN-bus.

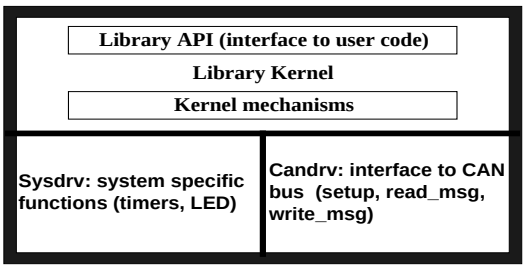


Figure 2. The structure of the DeviceNet library.

This partitioning scheme ensures portability of our software to platforms with poor resources (micro-controllers) as well as platforms with rich resources such as PCs. The library kernel contains the protocol stack only and interface to the application program. All dependencies to particular environments are located in the two other parts of the library. So if ones kernel module has been debugged and tested under Linux, it can be used in DeviceNet compliant devices, developed for other platforms. To port the library to a new platform it is necessary to modify system dependent functions (module called *Sysdrv*) and interface to CAN-bus (called *Candrv*).

Various versions of DeviceNet library have been tested including Intel-compatible PC under Linux, single-chip micro-controllers from Microchip (PIC16C7x/87x) and DSP from TI (TMS320C2xx).

3.2 Software support of real-time.

Real-time extension of Linux OS - RTLinux allows developing software components, which have got hard real-time capability [7]. The real-time processes are implemented in RTLinux as lightweight threads and run in the kernel memory space. RTLinux coexists with Linux OS and Linux kernel operates as separate real-time process with the lowest priority using a virtual machine layer in RTLinux.

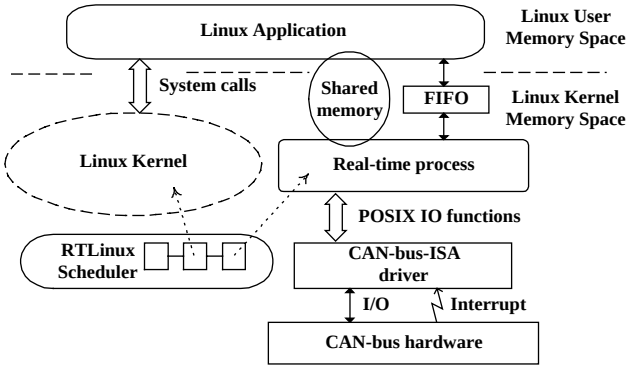
RTLinux version 2 consists of core component and several optional components. The core component is distributed as Linux kernel patch. The core component allows registering low interrupt handlers that cannot be

preempted by Linux itself. The optional components provide:

- pure priority based scheduler,
- set of functions to work with system clock and timers,
- support for POSIX IO interface (read/write/open/close) for real-time device drivers,
- real-time FIFOs, that connect a real-time process and Linux user space process through a special character device file so the Linux process can read/write to real-time component,
- shared memory between real-time components and Linux processes.

RTLinux uses the loadable kernel module mechanism implemented in Linux OS to load as real-time processes as optional components into the memory [8].

Figure 3. Interaction of RTLinux components in control system software.



RTLinux version 2 supports the real-time POSIX.1b threads as well as API of RTLinux version 1 for backward compatibility. There is also support for POSIX mutex locks in the latest minor versions of RTLinux 2.x. The support of POSIX IO interface in RTLinux provides a filesystem like interface to real time drivers.

RTLinux provides only basic real-time capabilities whereas Linux OS provides all other general services. An application that requires real-time capabilities consists of two parts: real-time kernel module implementing real-time functionality and Linux process communicating via FIFO or shared memory with the real-time module.

To support access of real-time software running on front-end PC to CAN-bus the RTLinux driver for CAN-bus ISA adapter was developed. The driver provides POSIX IO interface for real-time control and acquisition processes.

If real-time process wants to access CAN-bus it opens special character device file (/dev/canX), and just read from or write to this file the CAN frames (Figure 3).

3.3 Control system application software

The application software of control system is based on architecture with Distributed Shared Memory (DSM) [3]. Modules of application software watch and control an accelerator through a segment of DSM. Mirroring mechanism of DSM segments lied hidden from application software. Software components accessing the segment of DSM, and not responsible for mirroring, might know nothing about inter-level communication construction and were not concerned with the appearance of data. This approach ensured rather clear application program interface, which simplified work of programmers and made possible, the independent development of parts of application software as mirroring algorithm for different types of hardware.

4 CONCLUSION

The newest features of last RTLinux versions such as POSIX compatible interfaces appeared since our last publication [3] and our last experience has strengthened the belief that RTLinux is usable for accelerator control.

During the last three years the new control system has been developed as a result of our efforts in the fields of Linux OS and CAN-bus applications. We propose Linux OS as attractive alternative to other operating systems as developing as runtime platform for control systems of particle accelerators.

REFERENCES

[1] A.S. Chepurnov, I.V. Gribov, et. al., " Moscow University Racetrack Microtron Control System: Ideas and Developments", Proc. of ICALEPCS (Tsukuba, Japan, KEK, 11-15 Nov., 1991) Tsukuba, KEK, 1993, pp. 140-142.

[2] F. Nedeoglo, A. Chepurnov, D. Komissarov, Linux and RT-Linux for accelerator control - pros and cons, application and positive experience. // Proc. of ICALEPCS'99.

[3] A. Chepurnov, A. Alimov, et. al., "Control System for New Compact Electron Linac.", // Proc. of ICALEPCS'99.

[4] A.S.Chepurnov, A.A.Dorochin, K.A.Gudkov, V.E.Mnuskin, A.V.Shumakov, "Family of Smart Devices on the base of DSP for Accelerator Control.", Proc. of ICALEPCS, W2B-d (Chicago, Illinois USA, 1995).

[5] DeviceNet Specifications, Volume 1, Release 2.0, Volume 2, Release 2.0.

[6] A. Chepurnov, D. Komissarov, F. Nedeoglo, A. Nikolaev, "DeviceNet Implementations under Linux for Use in Control System of a Particle Accelerator." // Proc. of ICALEPCS'99.

[7] V. Yodaiken, M. Barabanov, "RTLinux Version Two Design", // VJY Associates LLC, 1999, http://www.rtlinux.com/archive/design.pdf

[8] David A. Rusling, "The Linux Kernel", // Linux Documentation Project, 1997.