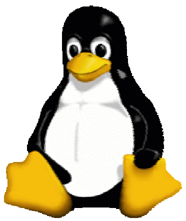
Embedded Real-Time Linux for Cable Robot Control

Frederick M. Proctor
Group Leader, Control Systems Group
National Institute of Standards and Technology, USA

National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

Intelligent Systems Division
Manufacturing Engineering Laboratory

Linux



- Linux is a Unix clone, written by Linus Torvalds at the University of Helsinki
  - begun in 1991, version 1.0 released in 1994
  - full-featured Unix: protected mode, multiprocessing, multitasking, virtual memory, shared libraries, networking
  - available for 386 and higher processors, Compaq Alpha, Sun SPARC, Motorola 68K and PowerPC, ARM, MIPS, more
- Linux source code is freely available as Open Source under the Gnu General Public License
- Many companies sell pre-configured distributions: Red Hat, Mandrake, Caldera, SuSE

Embedded Linux

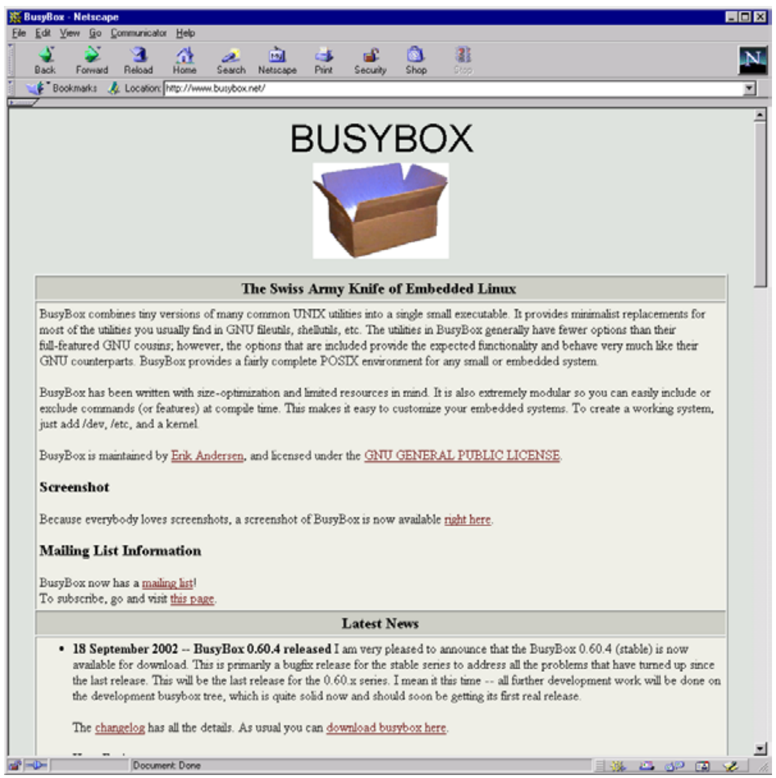
- Free and portable, Linux is popular for embedded systems
  - highly customizable for minimal use of computing and power resources
  - ability to run from ROM, Flash with no rotating media
- Linux supports soft real-time execution
  - tasks that can tolerate some variation in execution time
  - no requirement for completion before a deadline
- Linux doesn't support hard real-time execution
  - optimized for best average response time
  - can't guarantee task execution by a deadline, even for interrupt-based device drivers

Real-Time Linux

- Changes to Linux scheduler for real-time operation are available, and free
  - RTL from New Mexico Tech: X86, PowerPC, Alpha
  - RTAI from Milan Polytech: X86, PowerQUICC
- RTL and RTAI provide similar mechanism
  - RT scheduler runs RT tasks first
  - Linux is run as the last task, and is preempted for RT tasks
  - RT layer captures and defers interrupts to Linux device drivers
  - RT layer dispatches interrupts to RT device drivers as usual
- Your RT software is effectively a real-time device driver, with shared memory or FIFO communication to non-real-time Linux processes

Embedded Linux Distributions

Dozens of embedded Linux distributions are available



We selected BusyBox, distributed free as open source

Diskless Operation

- For applications that experience shock or vibration, solid-state read-write media is a must. Some alternatives:
  - Compact Flash, with built-in IDE interface for direct disk replacement
  - DiskOnChip, which requires newer Linux 2.4 kernel Memory Technology Devices (MTD) subsystem, device drivers
- Write operations wear out Flash media
  - "wear leveling" spreads out write operations transparently, lengthening lifetime to hundreds of years for typical use
  - achieved through either file system layer (e.g., Journaling Flash File System (JFFS)) or on the chip itself (e.g., DiskOnChip TrueFFS)

Booting

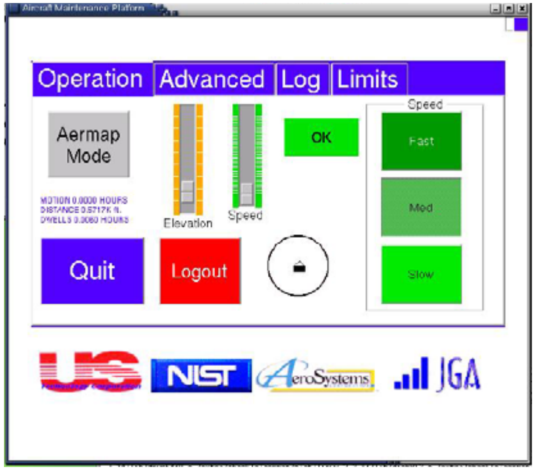
- Booting from IDE-emulating Flash is automatic
  - IDE interface makes Flash look like a normal disk
- Non-IDE flash requires additional software
  - for DiskOnChip, doc-lilo is needed
  - RAM disk image holds compressed Linux kernel and some boot files; you create this off-line and load into Flash
  - doc-lilo reads from Flash, loads RAM disk image, and booting continues as usual
- Linux supports RAM disks for files that do not need to persist between reboots, e.g., log files

Booting

- For instant-on applications, PC BIOS self-test and generic device initialization can be replaced with LinuxBIOS
  - project originated at Los Alamos National Lab
  - Linux boots from cold start to prompt in a few seconds
  - requires a specific port of LinuxBIOS to your PC board
- For networked applications, Linux can be configured to use BOOTP
  - commonly used for rack-mounted clusters
  - saves media cost, simplifies kernel upgrades

Graphics support

- Linux typically uses the X Windows graphics system
  - takes tens of megabytes of disk, megabytes of RAM
- Stripped-down alternatives exist that still support mouse input and multiple windows
  - GGI, DinX
  - MicroWindows/NanoX
  - Qt/Embedded ⇒



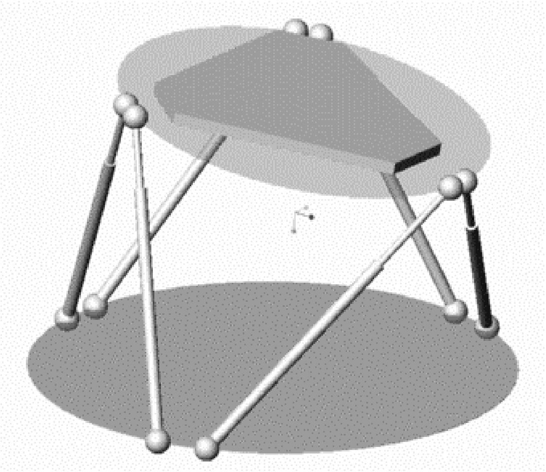
- These use either video board-specific libraries, the SVGA standard, or the newer Linux Frame Buffer abstraction
  - the Frame Buffer has been ported to many modern boards, and supports higher resolution, more colors

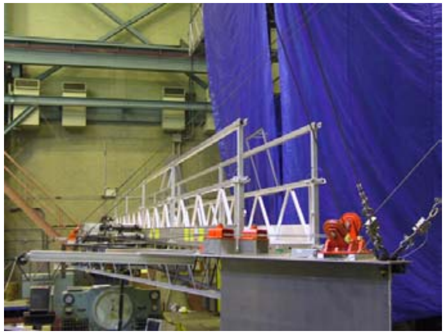
Configuring Embedded RT Linux

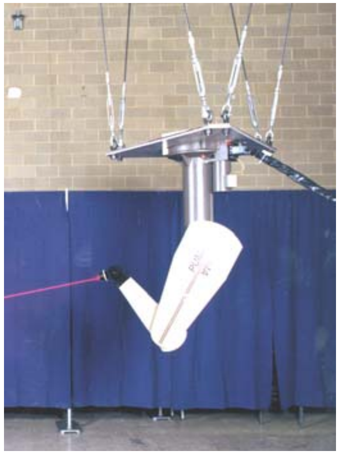
- Set up a conventional development system with hard disk, floppy, CD-ROM, etc. and RT Linux source code
  - follow instructions provided with plain vanilla Linux, RT Linux distributions
  - build a bootable RT Linux kernel, including Memory Technology Devices subsystem, Flash disk drivers
  - build a bootable floppy with this kernel, additional floppies with useful utilities
- Boot embedded system off the floppy
  - use utility floppy to format flash disk, copy kernel and boot loader
  - Copy your application code to Flash as it evolves
- Other options: development system = embedded system; networked embedded system

Cable Robots

- Stewart Platform parallel kinematic mechanism turned upside-down
- Cables instead of linear actuators
- Quite stiff, and improves with loading
- Dual of serial kinematic mechanism: inverse kinematics are closed form (easy), forward kinematics are iterative (hard)

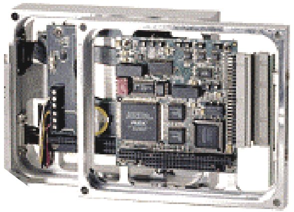




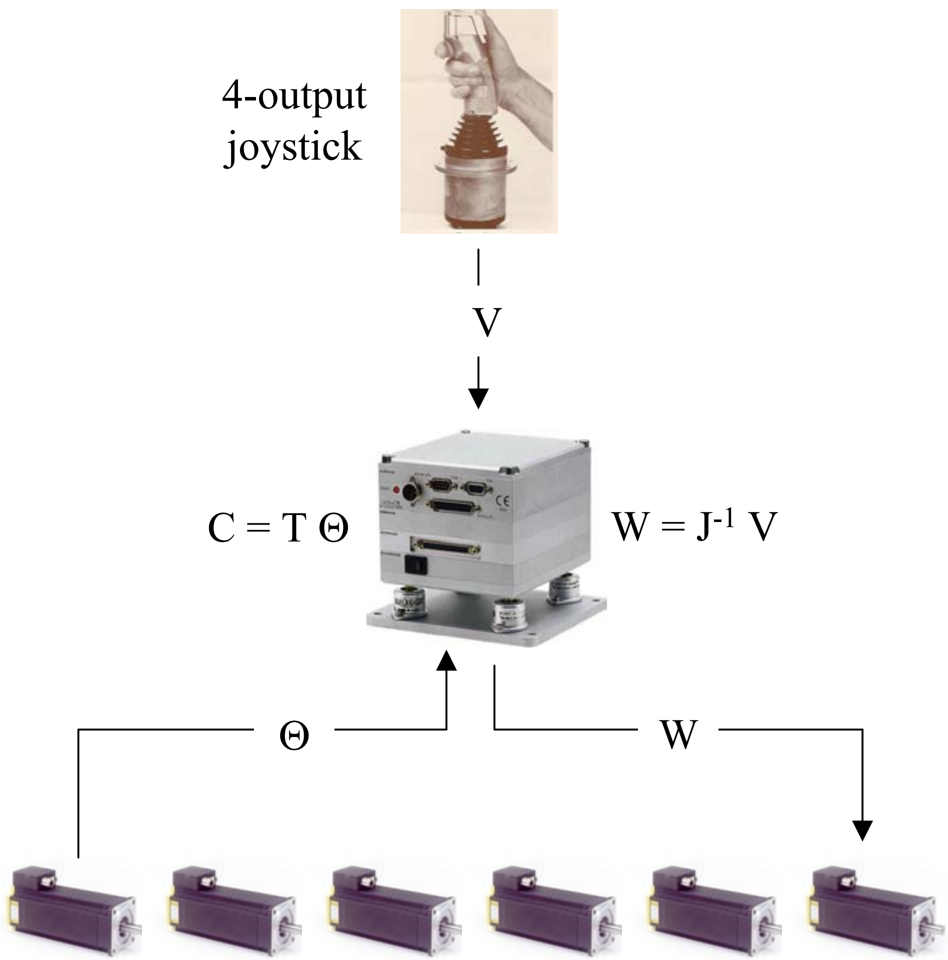


Our Computing Needs

- Solid-state media for shock- and vibration resistance
- Real-time control for Cartesian velocity teleoperation
- Bi-directional serial I/O to digital motor controllers
- Analog input, digital I/O to sensors and relays
- Our system:
  - PC-104 with Pentium Geode processor
  - BusyBox Linux, New Mexico Tech RTL
  - DiskOnChip 96 Mb Flash
  - Qt/Embedded, Touch screen w/ custom driver
  - RS-232/422 serial; analog input, digital I/O
  - Ethernet for development



Cartesian Teleoperation



- Joystick outputs XYZ-Yaw velocities; Roll and Pitch set to zero
- Controller transforms to cable velocities using inverse Jacobian J^{-1}
- Cable velocities sent to motor controllers via RS-232,422 serial links
- Motor controllers reply with rotational positions ($\Rightarrow$ cable lengths)
- Controller transforms to Cartesian position using forward kinematics T
- Repeat next Δt ...
- Note: J^{-1} is an instantaneous relationship; for finite duration between commands some roll and pitch velocities will creep in

Automatic Leveling

$$\begin{bmatrix} \\ \\ J^{-1} \\ \\ \\ \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ V_z \\ V_r \\ V_p \\ V_w \end{bmatrix} = \begin{bmatrix} \omega_0 \\ \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \\ \omega_5 \end{bmatrix}$$

$\Rightarrow$ . . . $\Rightarrow$

$$\begin{bmatrix} \\ \\ T \\ \\ \\ \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \\ R \\ P \\ W \end{bmatrix}$$

Synthetic leveling: roll and pitch are computed from T;

$$V_r = -k\, R_{computed}\,,\quad V_p = -k\, P_{computed}$$

Sensor leveling: sensor produces outputs proportional to roll, pitch;

$$V_r = -k\, R_{meas}\,,\quad V_p = -k\, P_{meas}$$

Automatic Leveling

- Synthetic leveling:
  - no need for separate sensor
  - can't compensate for cable sag, uncalibrated kinematics
- Sensor leveling:
  - a true measure; compensates for cable sag, uncalibrated kinematics
  - requires a separate sensor and associated computer inputs
- These can be combined to detect sag outside some allowable range, or cable interference
- Both methods are closed-loop, and require tuning of gains
  - simple proportional (P) control worked fine
  - PID can clean up steady-state error, damp response

Calibration and Homing

- Calibration of cable robots is difficult
  - large structures present accessibility problems
  - pulleys spread as the platform rises
  - effective cable drum diameter changes as cable wraps up
  - net result: accuracy on the order of centimeters over 10 meters
  - during teleoperation, people will accommodate for this
- A homing procedure is necessary
  - since the forward kinematics are iterative, we need a good estimate of the initial Cartesian position for measured cable lengths
  - from scratch, we define a Cartesian home position with respect to world coordinates; run inverse kinematics to get cable lengths
  - if the robot is not homed the cables must be jogged to their home lengths, which should be marked for convenience
  - during routine operation, we save Cartesian position to Flash at shutdown and restore at startup, allowing power-down anywhere

Summary

- Linux is a free operating system with embedded- and real-time distributions, useful for research and commercial applications
- Solid-state replacements for rotating disk storage protect against shock and vibration, making robust systems
- Sophisticated graphical user interfaces can be built with modest storage and memory requirements
- We built a cable robot controller using the PC-104 form factor, DiskOnChip Flash media, and free software
- Cartesian teleoperation using non-trivial kinematics was accomplished successfully