Linux in Embedded Systems for Engineers

- Southern California Linux Expo 2005

This talk is aimed at engineers.

It contrasts the advantages of software development using a full Debian Linux distribution on the desktop with the more restrictive and different challenges of an embedded target where many of the goodies go away.

- Alexander Perry
  - PAMurray
  - IEEE Consultants, San Diego
  - UCSD Extension, Engineering

alex@pamurray.com
http://www.pamurray.com/

Boxed Linux - Contents

- Lots of packaging
  - Outer shrinkwrap wrapper, cardboard interior box
  - Printed thin card outer sleeve, mostly advertising
- An instruction manual and/or booklet
- A CD-ROM with a printed image on the front
  - Raw disk content is an aggregation, separate license

- The files on the CDROM form a "distribution"
  - A consistent common runtime environment
  - A collection of Packages to choose from

- A "Package" is a specially formatted file
  - Any programs, data files, install scripts, etc
  - Associated documentation, examples, licenses ...
  - Carefully configured to run in that environment
  - "Dependencies" specify one package needs another

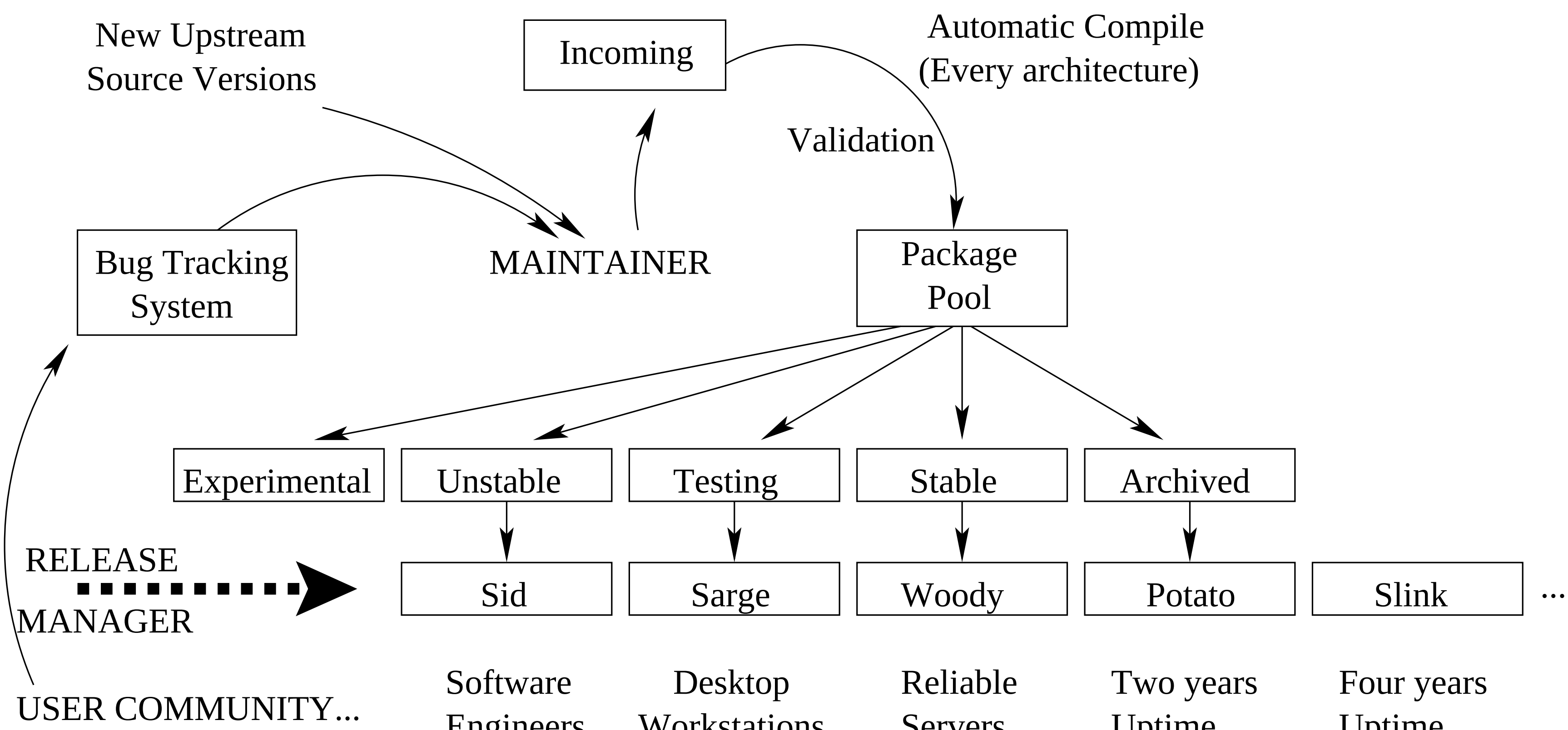
The Debian GNU/Linux distribution

- 13000 packages of software
  - The Linux kernel and associated administrative programs
  - Various GNU tools, utilities and applications
  - Thousands of other applications and alternatives
  - Apache, MySQL, Perl, OpenOffice, KDE, Mozilla, LTSP, ...

- An automatic tool "lintian" validates packages
  - Nonconformant submissions are automatically rejected
  - Searchable public bug tracking, http://bugs.debian.org/
  - Program "reportbug" helps all users submit useful reports

- Validated dependency data between packages
  - Security and version upgrades are reliable and fast
  - Upgrades rarely need any reboots
  - Active users are not disturbed

Release Process offers Integration Quality



- Extensive documentation ensures consistency
  - Software vendor suggestions - 7 pages
  - Repository recommendations - 7 pages
  - Policy manuals (nine parts) - 143 pages
  - Maintainer guide, Developers reference - 103 pages
  - Menu, Internationalization support - 150 pages

Most Distributions offer Similar Benefits

- Several hundred other distributions to choose
  - http://www.lwn.net/Distributions/
- There is a price for that environment
  - Utilities are compiled for general purpose usage
  - Scripts automatically run to adjust settings
  - Databases keep track of files, programs, versions
  - Scripts add/remove packages, with error checking
  - Often a hundred megabytes of overhead disk space
- Embedded linux has to be different
  - The processors are often slower, with less memory
  - Filesystem space is usually thousands of times smaller
  - 16 MB of flash in a chip, instead of a 60 GB drive

Who does Embedded Package Management

- In many cases, it's the engineer
  - Manually configure, adapt and build source code
  - Find dependencies and select compatible versions
  - Yields a small, fast product - but a lot of effort
- Sometimes a simple makefile
  - Embedded distributions find compatible versions
  - Often, an included makefile can build everything
  - Engineer just has to make adaptations as needed
  - Can only make limited changes before makefile breaks
- There are lightweight tools
  - Embedded Debian (cross tools), Familiar (ipkg etc)
  - Provide the install and removal management benefits
- Clearly, the robustness of desktop packaging is lost

Where do Embedded Distros come from?

- Making a distribution is hard work
  - Why are companies releasing them ?
- To gather customers in other product
  - Hopes to migrate you to fee product
  - So review the lock-in features
  - eg. Lynux and LynxOs with BlueCat
- To get assistance in supporting them
  - They built distro for inhouse use
  - Hoping to share support effort with you
  - Compare their work quality against yours
  - eg. Lightning Linux (Switzerland)
- To sell their consulting services
  - Sample of the quality of their work
  - Small, clean code, easy to extend
  - eg. ucLinux original release

Licensing - Part of the Business Strategy

- Licenses define what can and cannot happen
  - They constrain the business models associated with them
  - Both for the software author and for the recipient
  - It’s bad to accidentally destroy your profit opportunity

- Projects have associated business plans
  - Therefore, only certain licenses can be incorporated
  - Similarly, not all licenses will be offered to users
  - Whoever is responsible for such planning must decide

- License selection is not an engineering activity
  - It is a management decision role
    - Advised by legal support if necessary

Where are Embedded Distros going ?

- General purpose distributions change fast
  - Backward compatibility not always considered
  - You may need to port code every year
  - At risk of being left behind and abandoned

- Specialist distributions tend to bog down
  - When existing developer team is happy
  - It does what they want from it
  - You may be the only active developer

- Somewhere there is a happy medium
  - Active development and improvement
  - But slow and methodical, stable
  - Hard to judge at short notice

The Four Sections of an Embedded System

- A bootloader to run at power on
  - Needs to read flash storage (and write new images)
  - Often constrains how Linux can share that flash
  - Partition table restrictions, kernel size, etc

- Custom configured Linux kernel
  - Support for integrated features and peripherals
  - All the generic drivers you need, none of the rest

- Peripherals needed by the application
  - Usually unlike the equivalents on desktop computers
  - May be directly connected (not PCI), or custom logic
  - These drivers not needed to start the Linux kernel

- A filesystem with all software
  - This is what that package management is building

Bootloader - Thin Embedded System

- The bootloader is like BIOS and GRUB in one
  - It loads the kernel and initial ramdisk
  - Some of them can load these from the network
  - The x86's have PXE and/or EtherBoot for example
  - On desktop computers, this is called DISKLESS boot

- Embedded systems use flash, not disk
  - Can't call it FLASHLESS - bootloader is in flash
  - This is fast; avoids flash write and flash read
  - Reboots are as fast as sending 1MB over Ethernet

- Recommended as a quick way of iterating
  - First to get a kernel version that starts cleanly
  - Second to get a ramdisk that starts all peripherals

Linux kernel overview

- The only program with absolute control
  - Manages all the memory and disk paging
  - Operates all device and peripheral interfaces
  - Enforces security and access limiting rules
  - Manages network connections and protocols

- Memory is virtualized
  - Programs reuse the addresses transparently
- Disk drives use memory too
  - Store pending data that’s about to be written
  - All reads, and completed writes, kept for a while

- Unused memory is moved out
  - Backing storage is usually on a disk drive partition
  - If short of disk space, can use network storage
  - May have several prioritized swap areas available
- May swap out inactive programs for more disk cache

Non-Network devices and peripherals

- Device drivers mostly portable
  - eg, PCI boards work on x86, PowerPC, ARM, IA64, etc
- No special new APIs
  - Each peripheral becomes a special kind of file
  - Normal access uses read and write as usual
  - Special features all use the ioctl() calls
- These files have permissions
  - Hardware access is treated like regular files
  - Simplifies deciding which users can use what
  - Read and/or write, match by user and/or group
  - The "root" user bypasses these file checks

Network connectivity

- Protocols are integrated
  - Enables secure and fast implementation of many protocols
  - Firewall routing consistently enforced on all traffic
  - Security rules are user independent - unless explicit
- Network interfaces are equivalent
  - Simplifies configuration, testing, debugging
  - Type independent routing and traffic switching
  - Virtualized, loopback and userspace capabilities
- No restriction on number of interfaces
  - Simultaneously use multiple ISPs, VPNs and LANs
  - Start and stop links, change settings, anytime
  - Wireless includes WiFi, Bluetooth, Ham, GSM, etc
- Network sockets are key to distributed computing
  - Allows computing effort to be offloaded elsewhere

Adding modules to the kernel

- Modules add/remove any time
  - Separately compiled additions to a kernel
  - Do not reside in memory unless loaded (for use)
- Useful for temporary hardware
  - PCMCIA / PC card, PCI hot swap chassis, SCSI,
  - USB and Firewire devices, SCSI bridge, etc
- Their licensing need not be GPL
  - Linus has made the statement and decision
  - Thus, closed source device drivers available
  - Provides support for hardware without documentation
  - Consequently rarely portable to embedded targets

The Universal Serial Bus (USB)

- Popular for Consumer Electronics
  - Quickly and easily attach your mobile peripherals
  - Lets you avoid opening the case to use PCI slots

- USB 1.1 is the standard service
  - Driver is UHCI-HCD or OHCI-HCD depending on chipset
  - Latency for I/O is one millisecond (can be more)
  - Less than 1MB/sec bandwidth - shared among all devices

- USB 2.0 is on newer computers/chipsets
  - Driver is EHCI-HCD ... if not present, falls back to 1.1
  - Latency for I/O can be reduced as low as 125 microsecs
  - Available bandwidth is comparable to fast ethernet

Embedded target may not have spare PCI

- Difficult to install peripherals for diagnostics

- So hang them all off one external USB hub
  - Hard drive (extra storage, swap, logfiles)
  - Printer port (syslog hardcopy, hardware control)
  - VGA adaptor (graphics display, video monitoring)
  - Network interface (dedicated GDB, syslog, NFS)
  - Serial port (flash programmer, external watchdog)

- If your chipset does not have integrated USB
  - Plug-in boards for PCI, mini-PCI and PC-Card available

- USB uses memory mapped, bus mastered I/O
  - Reduced processor impact compared to other options
  - One interrupt triggered, even for many active devices
  - This is comparable to the more expensive ethernet cards

Kernel availability and customization

- All releases made available for download
  - http://www.kernel.org/
- The whole kernel is GPL licensed
  - Would you like to read seven million lines of code ?

- Interactive menu-driven configuration
  - Select only the hardware you really have available
  - Remove unused code for a smaller and faster kernel
  - Choose features, optimize for a specific purpose

- Distributions make this automatic
  - Compiling the source, installing as an alternative
  - You can try it and, if it doesn’t work, stop using it

Linux runs on many different platforms

- Targets many fast processor families
  - More than any other operating system ...
  - Intel/AMD/Sun/HP's 64-bit processors
  - IBM's 370 mainframe family
  - PowerPC, ARM, Sparc, MIPS, etc
- Also targets small, cheap, low power ones
  - The Dragonball (aka Palm pilots)
  - ColdFire, i960, 68k, 8086
- For clean code, simply recompile it
  - Even for 3D graphics card drivers

Platform mobility is a big benefit

- Your project currently only targets one
  - Remember it is likely to migrate with hardware pricing
  - So try to write clean code now so you just recompile
- Many bugs hide when only one target
  - Therefore, build for several, even if you only ship one
  - If targeting a PDA, make it run on the desktop too
- If there is a risk of processor change
  - Make a single build environment switchable
  - A global parameter to specify computer platform
  - Most package build engines support it - built in
  - Need to review command line switches carefully

Test the File System Contents

- Put stable kernel and ramdisk in flash
  - If network boot is faster than flash boot, keep using it
  - The rest of the file system (after ramdisk) however ...
- Share the filesystem between target and host
  - The whole thing can be NFSROOT mounted by the target
  - Attach a SCSI disk drive with dual host adaptors
  - Hand over USB flash drives using a device sharing hub
  - Install a USB device adapter card directly in the host
- If this is a partition and not compressed
  - Use RSYNC to update only the changed blocks

uClinux, the microcontroller version

- uC ... as in Microcontroller
  - For systems without a Memory Management Unit (MMU)
  - Therefore no memory or hardware protection
  - Do not use floating point - software "float" only
  - Must throttle user load, and network listen()
- Real Time extensions invaluable
  - Tenfold improvement - now comparable to ordinary PC
  - Interrupt response is measured in processor cycles
- Multitasking support limitations
  - Works fine, runs init and inetd by default
  - Static linked binaries can use lots of RAM
  - fork() impossible since it implies a MMU
  - vfork() works, fine for spawning processes
  - Stack is statically sized, but malloc() works

uClinux architectures

- Motorola
  - MC68EZ328 DragonBall, M68328 - ucsimm kit
  - M68EN302, M68EN322, MC68360 QUICC
  - M68020 (Atari and Prisma projects)
  - MCF5272 etc - ucdimm kit
  - MC68EC030 - Cisco 2500,3000,4000 routers
  - 5206 ColdFire, 5307 - ADOMO set top box
- ESA SPARC - Leon open source
- ARM
  - Atmel AT91 - with eval board
  - ARM7TDMI - Aplio VoiceOverIP telephone
  - StrongARM, the Intel XScale family
- Intel - i960
- Axis
  - ETRAX 100 - AXIS 2100 Network Camera
- Hitachi - SuperH

Modular Application Capability

- Scalable software is often client-server
  - Or more layers, with interfaces and abstraction
  - Data centers can segregate and consolidate layers
  - This offers more performance and also lower cost
- Embedded versions are often monolithic
  - That's good if your device is always independent
  - Reasonable if the processing layers are not reusable
  - But what about multifunction and/or connected devices?
- The device doesn't have a managed network
  - No data center admins to specify service locations
  - You need to install one of the discovery protocols

Modular Application - Example breakdown

- This is not a special Operating System or kernel feature

- Just a collection of co-operating programs
  - They can all be on different computers
  - There are many choices for each category
  - Delivering a highly customizable environment

- Here are ten categories to consider ...

- 1. Your application(s), eg OpenOffice
  - The many programs you wanted to run
  - Some may be across the internet somewhere
  - Power users may have dozens at one time

Provision of a graphical environment

- 2. The X windowing environment, eg xfree86
  - Multiple programs can simultaneously use it
  - Needs access to mouse, keyboard and display
- 3. Window manager, eg blackbox
  - Keeps track of windows and menu bars
  - Decides which window receives keyboard input
- 4. Desktop manager, eg kde
  - Maps documents and files to screen icons
  - Provides consistency between logins
- 5. Device drivers for user peripherals
  - Audio, Video, Input, removable storage
  - This (and Linux) may be the only local software

Other associated invisible services

- 6. Network related infrastructure
  - Name, storage, outgoing mail, time, authentication, ...
  - These can be outsourced, need a local fallback solution
  - A stub service tries to discover the local server
- 7. Printing (and other peripherals)
  - Conversion of documents into postscript or PDF
  - Rendering of queued job to printer binary file
  - Delivery of binary page images through kernel driver
  - ... these can be serialized if not offloadable
- 8. Additional storage (memory/disk)
  - Most interactive apps need per-user storage areas
  - Nothing to stop you putting some swap space there
  - Add swap while app holds a file open, then close
  - Also enlarges VFS space for any temporary files

Stability, Reliability, Scalability, Security

- 9. Multiprocessor support, of course
  - SMP motherboards, processors sharing memory, hardware
  - Clusters of separate computers, networked together
  - Installations of hundreds of Linux systems is routine
  - OpenMosix and NUMA are applicable for small systems

- Embedded market has already gone multiprocessor
  - Use those capabilities - don't ignore or disable them
  - More performance for customers with multiple devices

- 10. Virtual Private Networking (VPN) support
  - Needed by the users, accessing their remote services
  - Useful for the device, to secure its cluster traffic

- Smart cards and public key infrastructure (PKI)
  - Protecting data and any migrated process images

Thank you for your interest

- Any questions ?

Revision Control is Crucial

- Most open source projects use CVS
  - There are better alternatives available
  - But, unless you want all engineers to have to learn two
    - Use CVS for the in-house code archive
- CVS is structured and has many features
  - Spend several days learning to do branch control well
  - History is a project's lifeblood - don't be scared to commit
- CVS is concurrent, no locking mechanism
  - Better to use the branching and merging features
  - Enables parallel development, regresion and bug fixing
- Weekly developer team meeting (or more often)
  - Review branch status, goals and any major checkins
  - Discuss tricks, mistakes and anything wrongly committed