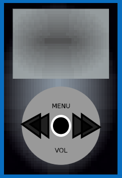
Embedded Linux and the mainline kernel

David Woodhouse

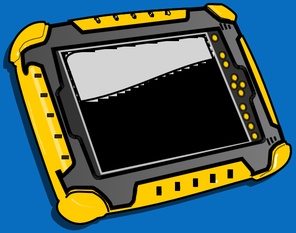
CELF Embedded Linux Conference

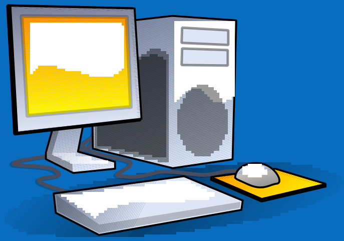
April 2009

Ubiquitous Linux



Embedded control device… phone… PDA… Internet tablet… router… media device… netbook… laptop… desktop… server… supercomputer…





“Embedded”...?

Portable Media Players

Phones

PDAs

“Internet Tablets”

Routers

Televisions

VCR / PVR / DVD / Media

Netbooks (?)

“Embedded”...?

Headless?

Handheld?

Power source?

Physical size?

Limited RAM?

Storage?

Other...

Embedded needs

Power management

Fast startup

Headless operation

Uncluttered user interfaces

Solid state storage

Embedded needs

Power management

Fast startup

Headless operation

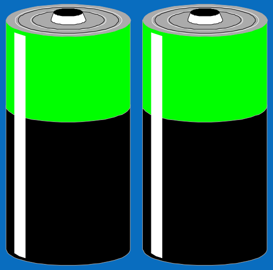
Uncluttered user interfaces

Solid state storage

Other users need these features too!

Power Management

Battery life

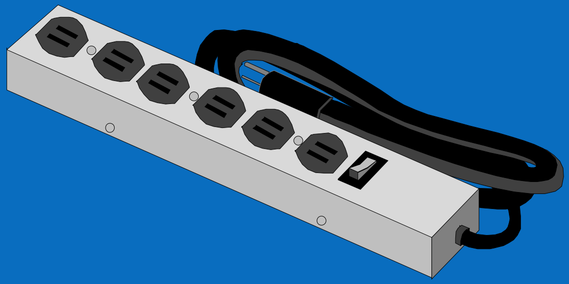


Power Management

Battery life

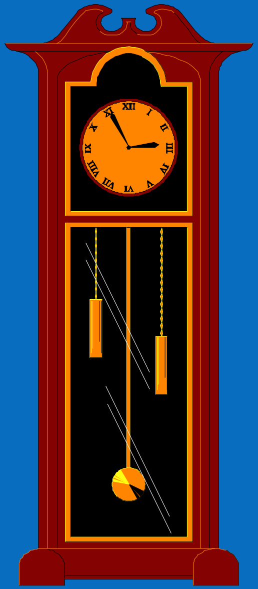
Cost of power consumption

Heat output



Tickless operation

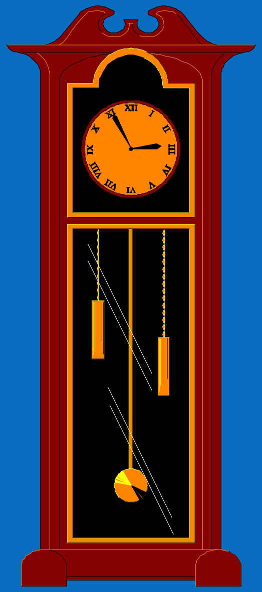
Power savings



Tickless operation

Power savings

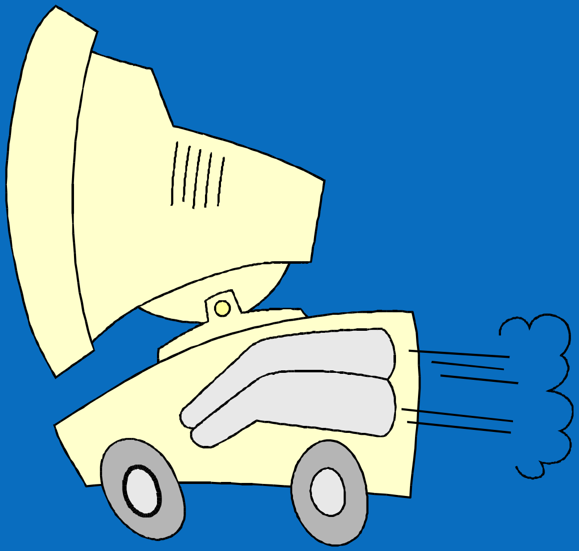
Scalability for virtualisation



Fast boot

Hard limits for mobile telephones

User experience for consumer electronics

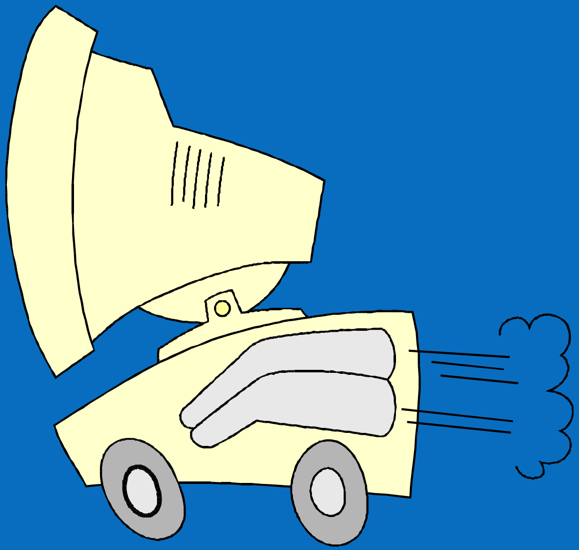


Fast boot

Hard limits for mobile telephones

User experience for consumer electronics

Server availability



User interfaces

Ease of use for consumer equipment

User interfaces

Ease of use for consumer equipment

... and for everyone else:

- OLPC / Sugar
- Netbooks
- Simple desktop environments

Solid state storage

FLASH storage in “embedded” devices

Solid state storage

FLASH storage in “embedded” devices

Solid State Disk

Others...

Execute in place (XIP)

- From FLASH for embedded systems
- Shared file system data under virtualisation

DMA API usage

- For cache coherency on embedded systems (ARM, some PPC)
- For IOMMU on larger systems

We are not so special!

Community impressions

“Enterprise” Linux

“Embedded” Linux

Community impressions

- “Enterprise” Linux
- “Embedded” Linux
  - Working with old code
  - Not working with upstream
  - Inclined towards “special” one-off hacks
  - Irrelevant to the general case

Community impressions

- “Enterprise” Linux
- “Embedded” Linux
  - Working with old code
  - Not working with upstream
  - Inclined towards “special” one-off hacks
  - Irrelevant to the general case

We must prove them wrong!

“Embedded” success stories

Tickless

Preemptive kernel

Power management

Suspend to RAM

Solid state storage

Squashfs

Working with the community

Find generic points of interest

Publish early and often

- In git trees
- Separate trees for separate development efforts
- Also send patches for review

Solicit and respond to feedback

Work with upstream maintainers

BE PART OF THE COMMUNITY!

Staying close to upstream

Advantages

- Easier for product updates and new products
- Easy to use fixes and new features
- External contributions
- Code review and testing

Costs

- Writing acceptable code can be hard and takes time
- Upstream kernel is a fast-moving target
- Releasing information may be difficult

Tips on contributing code

Find parallel requirements

Avoid “hacking around” problems

Avoid overengineering

Care about locking

Coding Style

Submit patches carefully

What's next for “Embedded” Linux?

- Solid state storage
  - More work on SSDs
  - Flash file system development (UBI, logfs, btrfs)
- Better power management
- More real time development
- What do ***you*** need?

Questions?

intel®

Leap ahead™