Operating System choices for IoT Devices

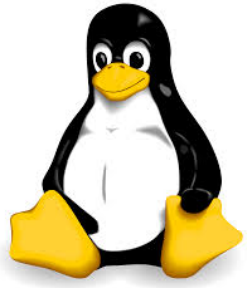
by Sven Behnsen

A little background: Sven Behnsen

- Hard & Software development since 1984
- Realtime Software distribution and development since 1989
- Consultant for Automotive industry since 2002
- Specialist for low level system startup/drivers, audio and storage media (HD, Flash, CD, DVD, etc.)

OS Choices

- Windows 10
- Linux
- Android
- Realtime OS
  - QNX
  - VxWorks
- Realtime Executives



VxWorks

MICROCHIP

TEXAS INSTRUMENTS

National Semiconductor

QNX

Things to consider

- Cost (Free, per unit royalties, fixed price)
- Available on desired hardware platform
- Suitable for intended use
- Intended number of units
- Time to market
- Development resources available
- Budget

- **TCO !!!**

Differences

- In order to understand where the differences are one MUST understand the OS Architecture

- Memory model
- Scheduler/Kernel
- Driver (I/O communication) concept
- Security & Certification

IEC 61508
ISO 26262
CC EAL 4+
IEC 62304 / 80001-1 etc

IoT vs Desktop

- IoT has limited CPU and Memory
- IoT is interacting with the Real World on two separate sides
- IoT user usually is not technical
- IoT must run forever
- IoT can not sacrifice main use for things like updates.

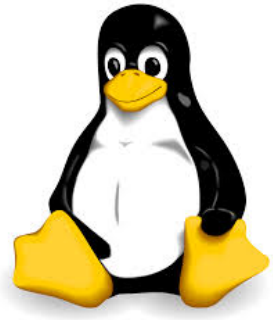
Technical Comparison

- Interrupt handling
- Scheduling (RR, FIFO, Adaptive)
- Latencies
- Real-time performance
- Memory protection
- Hardware supported
- Driver availability & architecture
- Development environment (including debugging & monitoring)

Windows 10 (Core)

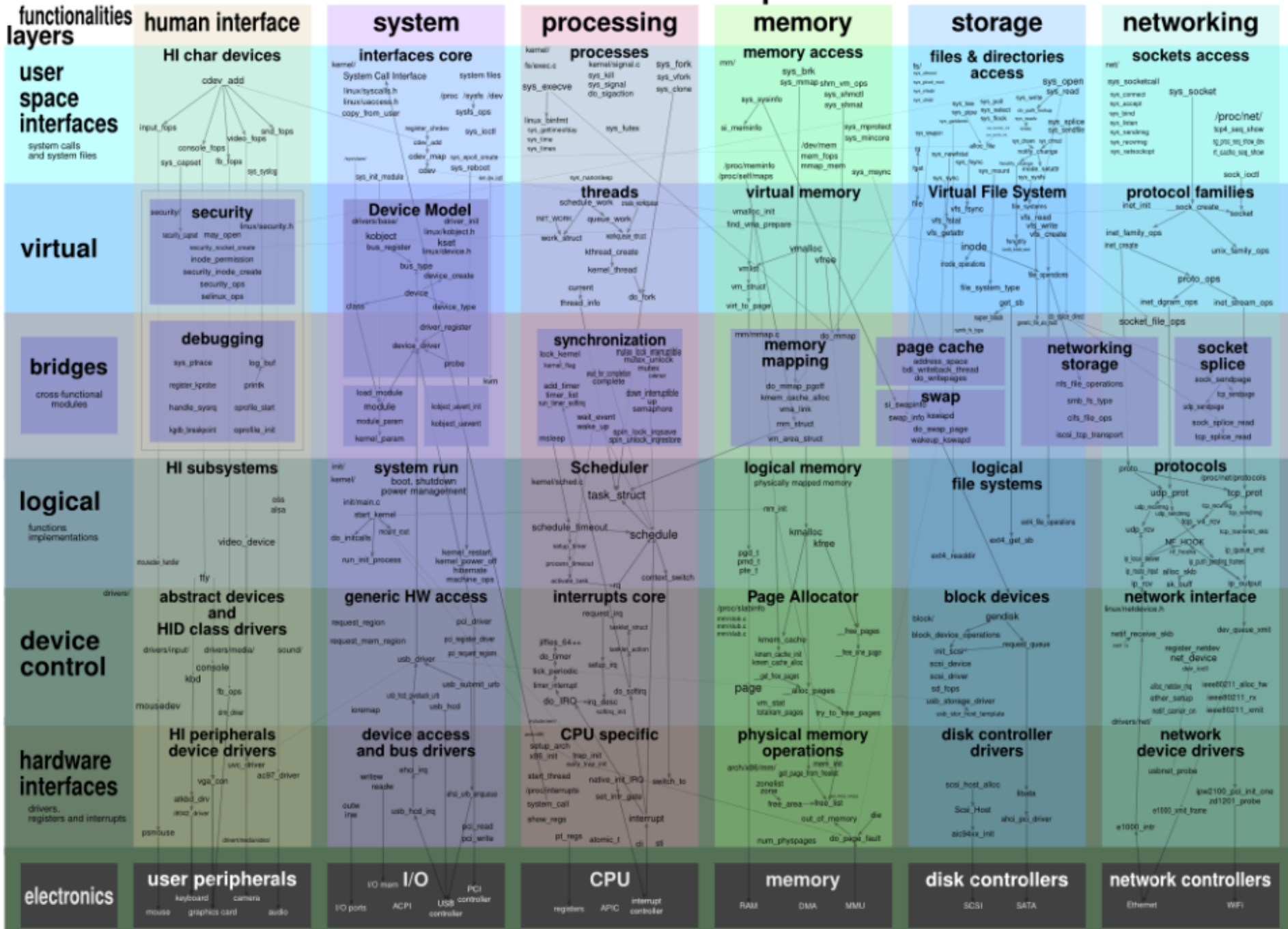
- Desktop Software Support (but who needs this in an IoT device?)
- Lots of development tools
- Large security risk because it is Windows
- Limited Realtime Performance
- Very Large
- Licensing is complex

Linux



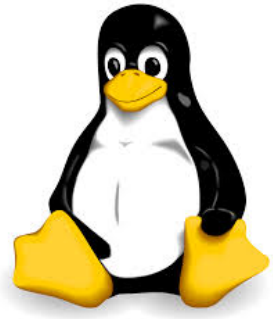
- Many flavors available
- “Free” means you have to do everything yourself
- Drivers are linked to the kernel
- Kernel is very complex (grows 1000 lines of code per hour)

Linux kernel map



© 2007, 2010 Constantine Shulyupin www.MakeLinux.net/kernel_map

Linux



- Many flavors available
- “Free” means you have to do everything yourself
- Drivers are linked to the kernel
- Kernel is very complex (grows 1000 lines of code per hour)
- Tool chains are not unified
- Licensing issues due to GPL, GPL2, etc.

Android

- Focus on UI and mobile applications
- Java development required (C through NDK)
- Slow build cycle (got better with AS 2.0)
- Not real time at all
- Licensing and branding issues with Google
- No real support available

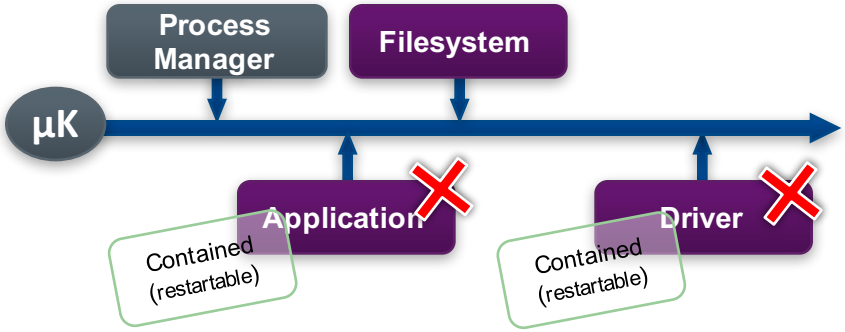
Realtime Operating Systems

- QNX
  - Full POSIX API
  - Excellent Tool chain
  - Memory Protection
  - Real Microkernel design
- VxWorks (Windriver)
  - Only partial memory protection
  - Significant learning curve
  - Only for large volume applications

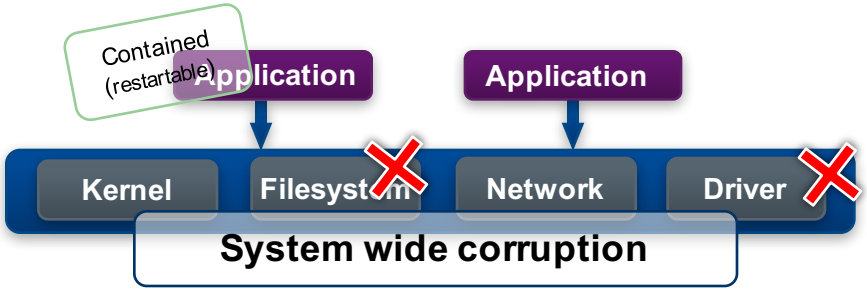
Realtime Executives

- Very limited functionality
- No memory protection
- Very complex development
- Usually high up front cost
- Usually bound to one specific hardware

TRUE **Microkernel** (QNX Neutrino)
- MMU with full protection
- Applications, drivers, and protocols are protected



Monolithic Kernel (Win, VxWorks, Linux)
- MMU with partial protection
- Applications are protected



Real Time Executive
- No MMU and no protection
- Applications, drivers, and protocols are all in Kernel space

Conclusion

- Android is not really a choice at all
- Linux is an option within limits depending on app and budget
- Windows cost/risk is large.
- Specialized stacks are only an option for very high volume
- Realtime OSs (particularly QNX) will get you the best and quickest result but don‘t come for free. IoT is really their domain.

Thank you for listening!

Questions?

sven@behnsen.eu