

Historie



- **1992: Gründung Sysgo GmbH**
- **„technischer“ Distributor für LynxOS**
Anpassung auf verschiedene VMEbus-Plattformen
- **Projekte mit anderen Betriebssystemen**
OS-9, pSOS+, vxWorks, AIX, NT, ...
- **1995: Portierung pSOS+ für MIPS R3/4000**
Auftraggeber: Integrated Systems Inc. (heute WindRiver)
- **1996: Mitarbeit bei Portierung LynxOS/MIPS**

LEO: OSEK OS Implementierung



- 1996: Entwicklung von LEO („Lynx Emulated OSEK“)
- OSEK-Kern und Applikation als Task unter LynxOS
- Anwendung: Rapid Prototyping (DaimlerChrysler)
- 1997: Portierung von LEO auf MIPS R3/4/5000
- Ansatz: Statt LEO neu zu schreiben, ersetze LynxOS durch μ Kern
- Geburtsstunde des μ Kerns „P4“
 - Wahl des „L4“ API (J. Liedtke, 1996) aufgrund TUD's L⁴Linux
 - Eigene Codebasis: Fokus auf Echtzeit & Portabilität
- 2002: Portierung P4 auf x86, PowerPC und ARM

LynxOS Zertifizierung



- **2000-2001: Zertifizierung von LynxOS nach DO-178B, Level A**
- **LynxOS: POSIX-konformes Echtzeitbetriebssystem („Echtzeit-UNIX“: Virtual Memory, Prozessmodell, etc.)**
- **Implementierung von „Ressource-Partitioning“**
- **Design Dokumentation (per „reverse Engineering“)**
- **Testsuite: 100% MC/DC Coverage**
- **Aufwand: ca. 40 Personen-Jahre, 16 von SYSGO**
- **Lektionen:**
 1. **„Es gibt Bedarf für sichere Betriebssysteme“**
 2. **„Mit einem μ Kernel wäre das besser gegangen“**

μKernel: ideal für safety-critical



- Kernel ist immer „trusted code“, hat den höchsten Sicherheitslevel
- Aufwand für Zertifizierung (Dokumentation, Testfälle, Überdeckungsanalyse) wächst mit dem Codeumfang
- => Kernel sollte so klein wie möglich sein
- => Kernel sollte nur das Sicherheitskonzept implementieren!
- Ein μKernel wie L4 implementiert die nötigen Abstraktionen:
 - Adressräume
 - kontrollierte Transaktion (IPC)
 - Scheduling

Partitionierung

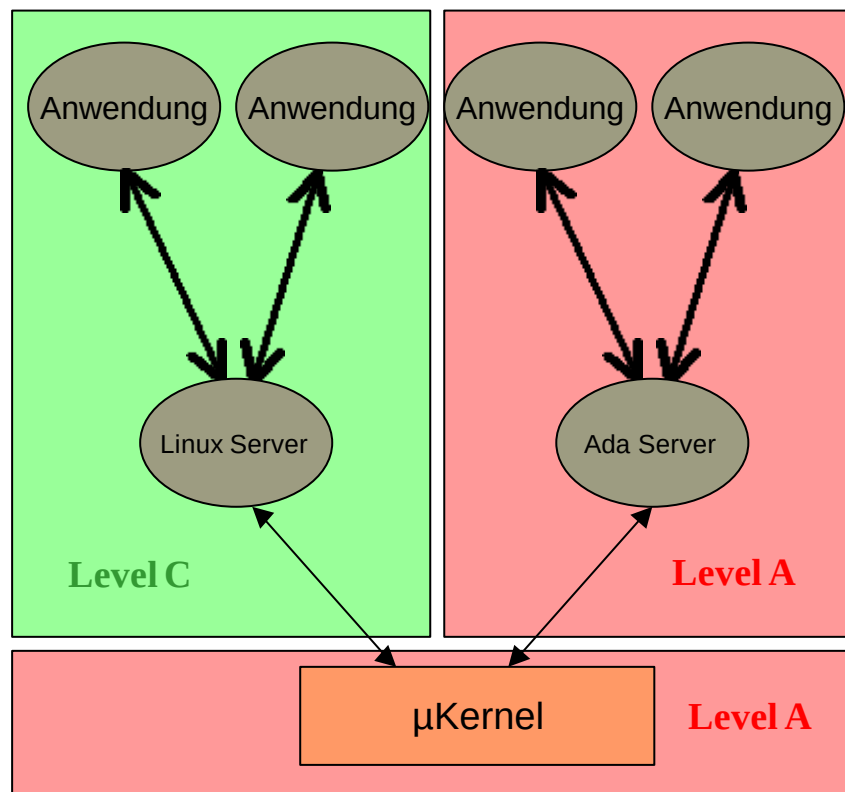


- **Aufteilen der Rechnerressourcen auf Subsysteme („virtuelle Maschinen“)**
- **Garantierte Verfügbarkeit der Ressourcen**
- **Ressourcen sind:**
 - Speicher
 - Rechenzeit
 - E/A-Geräte
 - Kernressourcen (Seitentabellen, TCBs, etc.)

Partitionierung (2)



- Partitionen sind entkoppelt -> können unterschiedliche Sicherheitsstufen haben



pikeOS: Ziele

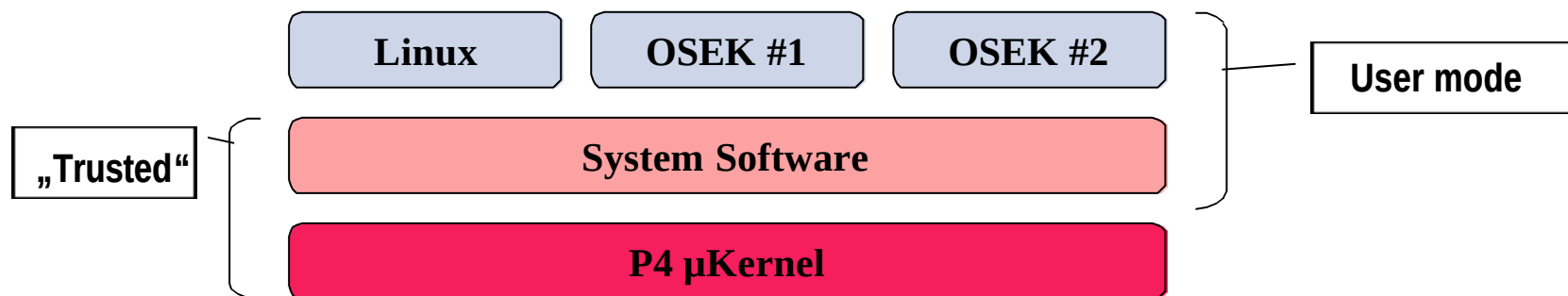


- **Neues Betriebssystem-Framework**
- **Basiert auf P4 μ Kern**
- **Partitionierung**
- **Sichere Koexistenz von nicht-vertrauenswürdigen Subsystemen**
- **Koexistenz von echtzeit- und nicht-echtzeit-Subsystemen**
- **BS Personalities für Standard-APIs/ABIs**
 - Linux, OSEK, POSIX, JVM, Ada Runtime,
- **Zertifizierbar (z.B. nach DO-178B)**
- **Marktfähiges Produkt für:**
 - Avionik, Kraftfahrzeug, Medizintechnik, ...

pikeOS: Stand heute



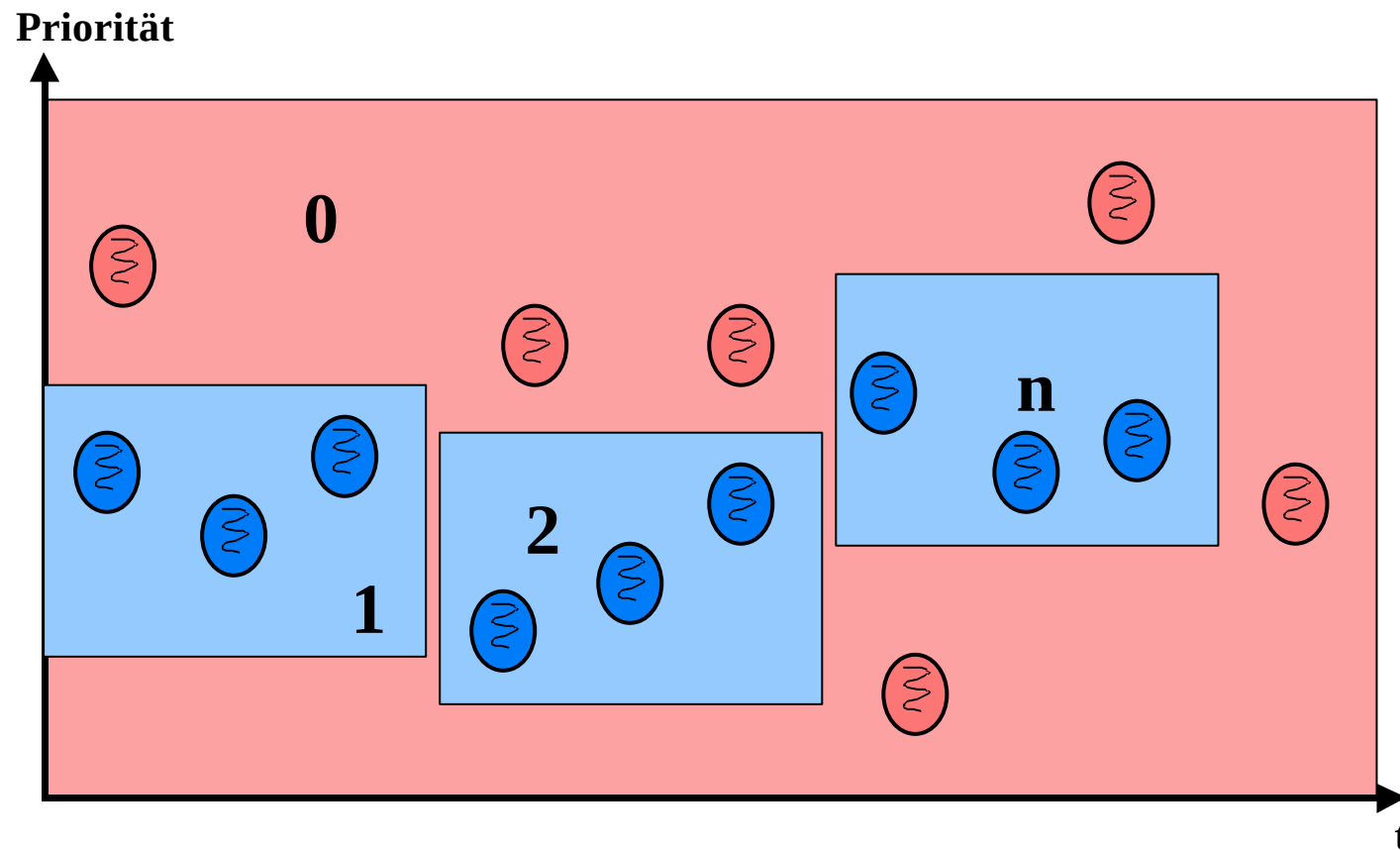
- **P4 Mikrokern (diverse Architekturen)**
- **„System Software“**
Speicherpartitionierung, Konfiguration per XML-Datei
- **Linux Server**
Basiert auf User-Mode-Linux, derzeit Version 2.4.20
- **OSEK OS Server**
mehrere Instanzen



pikeOS: nächste Schritte



- Zeitpartitionierung: ARINC 653 Erweiterung



pikeOS: nächste Schritte



- **Kernel Ressource-Partitionierung**

 - Freispeicher-Pool pro Partition

- **Weitere Personalities:**

 - **POSIX PSE53**
 - **legacy RTOS (z.B. vxWorks)**
 - **Java Virtual Machine**
 - **.....**