Linux as a Case Study: Its Extracted Software Architecture

Paper by: Ivan T. Bowman, Richard C. Holt, Neil V. Brewster

Presentation by:

Abram Hindle

Department of Computer Science

University of Victoria

abez@uvic.ca

September 20, 2005

This Presentation

- What am I going to cover?
  - Introduction
  - Conceptual Architecture
  - Concrete Architecture
  - Previous Work
  - Summary

Introduction

- Want to make a conceptual model
- Want to extract a concrete model
- Compare the two.
- Use Linux 1998

Linux

- Open Source Unix-like Operating System
- Liberal licensing, free and open distribution of source.
- Able to share results with research community and they can verify
- 10 KLOC in 1991 to 1.5 MLOC in 1998
- Kernel was 800 KLOC in 1998
- Lack of documentation

Research Context

- Reverse Engineering Community
- Some Architecture Extraction
- This paper was based on work done for this class in 1998
- Design Patterns had been out 4 years
- Earliest citation in this paper was 1992
- Relatively new field (altho the class was already a 700 level)

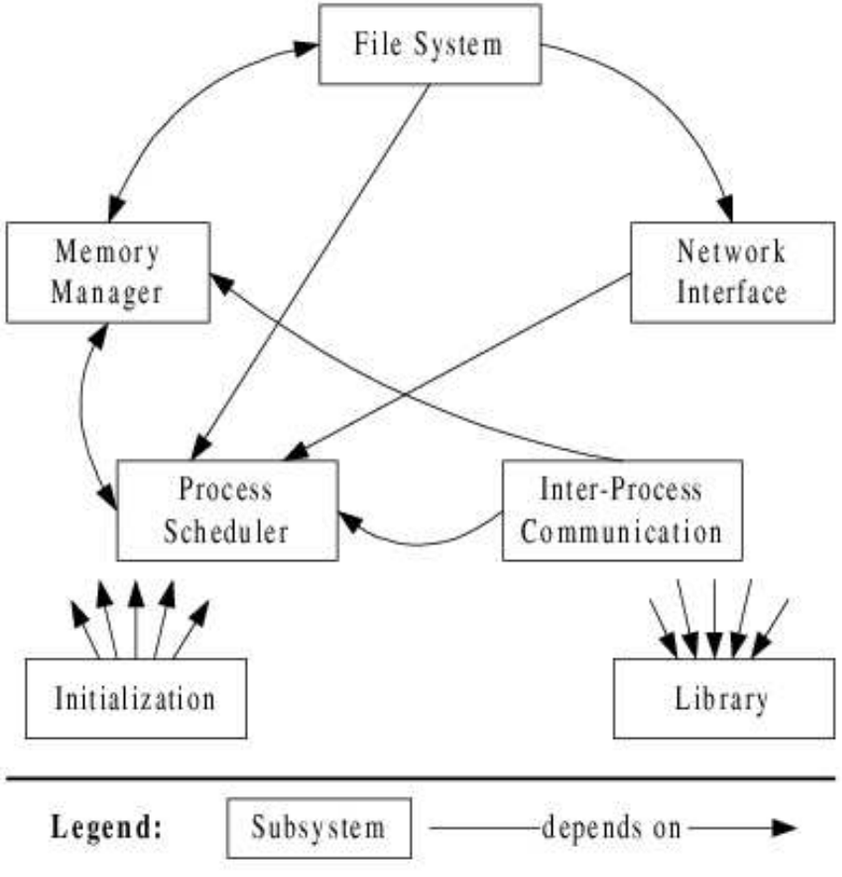

Figure 1: Linux Conceptual Architecture

Conceptual Architecture

- Built using the descriptions of the architecture from various sources.
- Used descriptions of UNIX and OS Architecture
- The documentation about Linux that was available.

Conceptual Architecture

- Subsystems
  - Process Scheduler
  - IPC
  - Memory Manager
  - File System
  - Network Interface
  - Init
  - Library

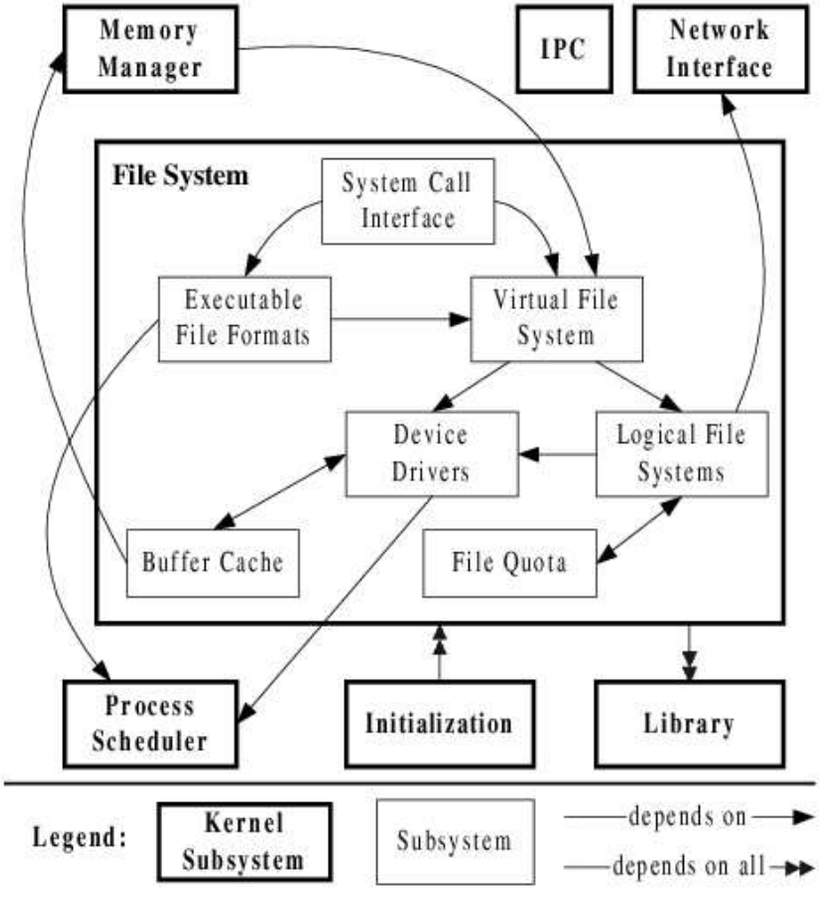

Figure 2: File System Conceptual Architecture

File System Conceptual Arch.

- Subsystems
  - Device Drivers
  - Logical Filesystem
  - Executable File Format
  - File Quota
  - Buffer Cache
  - Facades: SysCall Interface and Virtual File System

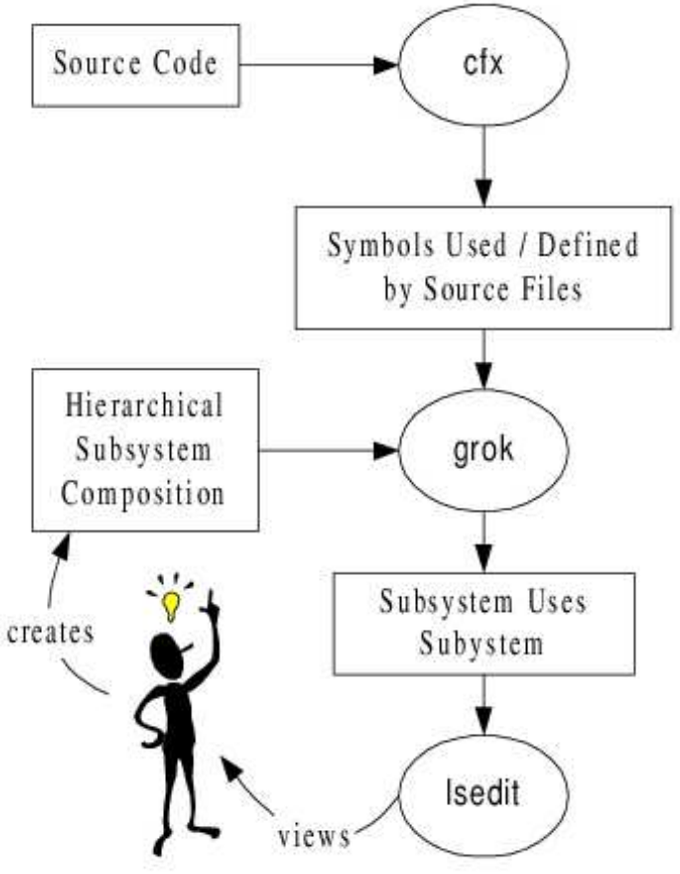

Figure 3: Extraction Process

Extraction

- Linux Kernel (800KLOC) (1682 source files)
- Symbols extracted through cfx
- Symbols related via grok
- Subsystems grouped via lsedit and grok
- Used Hierarchical grouping to group subsystems into bigger systems (e.g. the filesystem)

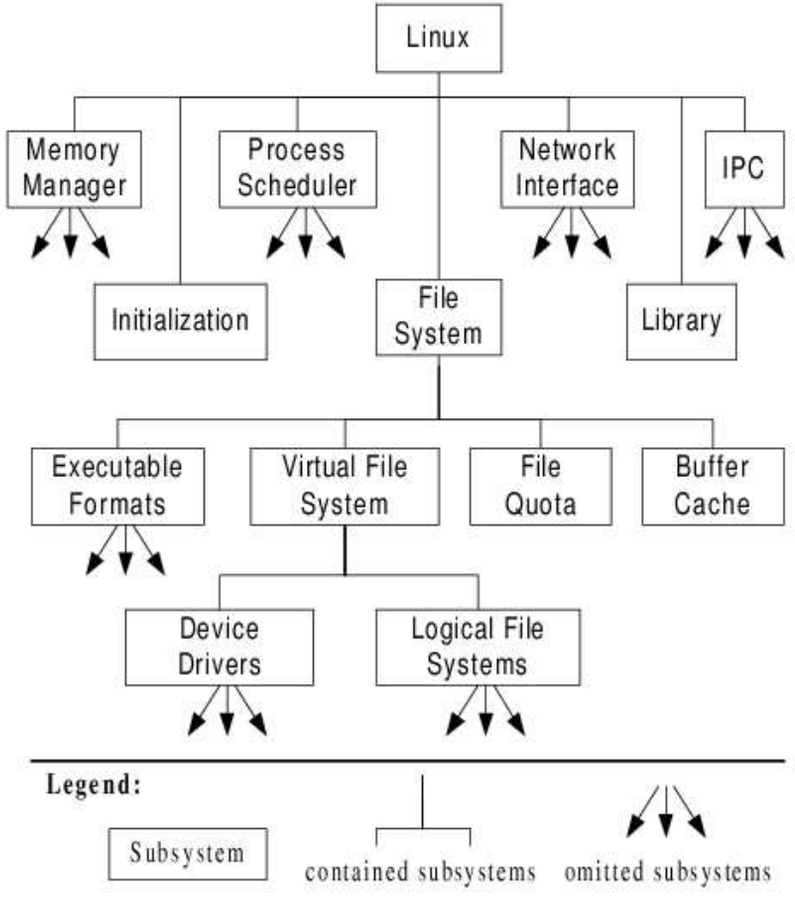

Figure 4: Partial Subsystem Hierarchy

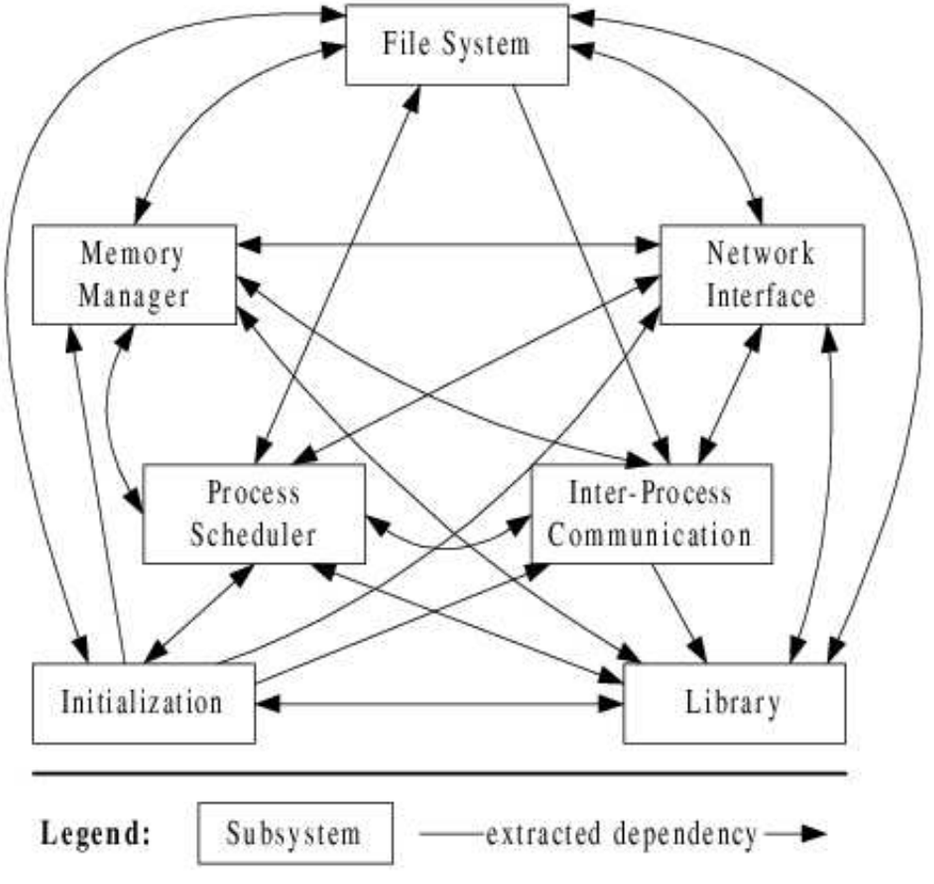

Figure 5: Linux Concrete Architecture

Concrete Architecture

- Similar to conceptual
- More interconnections (some surprising and somewhat spurious dependencies (printk))
- 19/42 Inter subsystem dependencies in the conceptual model
- 37/42 Inter subsystem dependencies in the concrete model

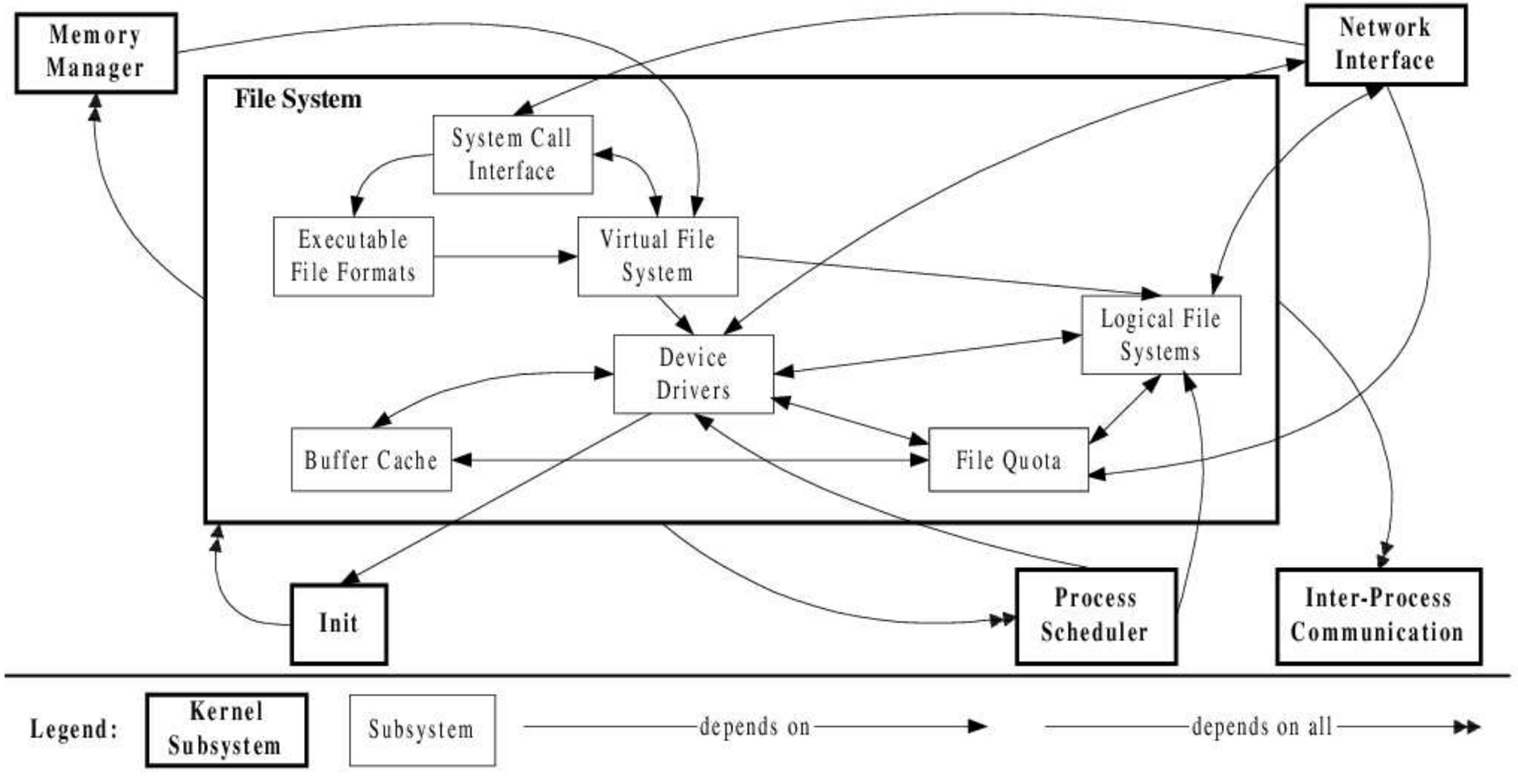

Figure 6: File System Concrete Architecture

Filesystem Concrete Architecture

- Same as the conceptual + more dependencies
- Surprises included networking dependencies (NCPFS, SMBFS)
- Depended on the IPC system for kernel level synchronization
- Some shortcuts

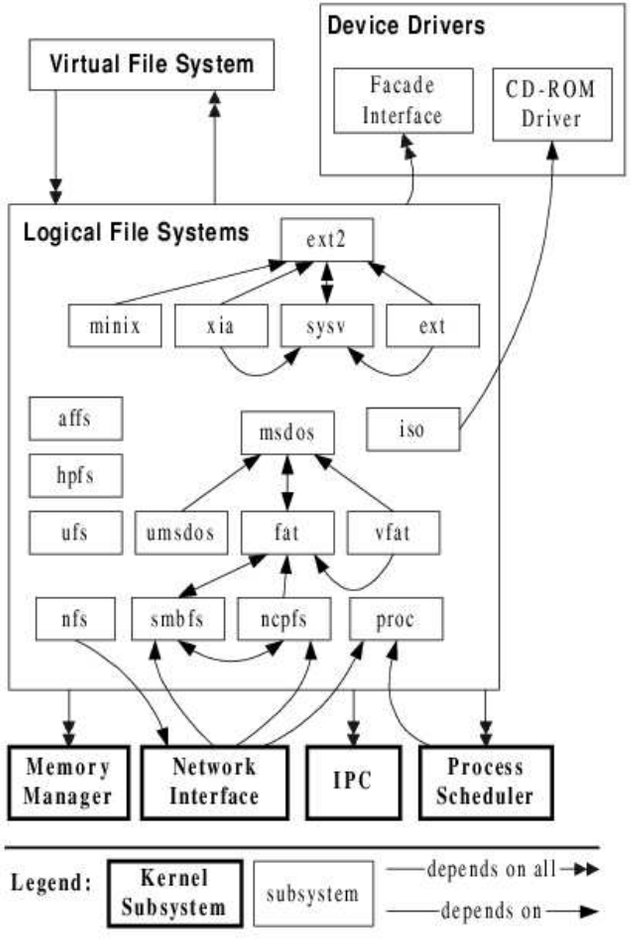

Figure 7: Logical File System Concrete Architecture

Logical Filesystem Concrete Arch

- PROC Filesystem had crosscutting dependencies
- ISO 9660 Filesystem to CDROM Device Drivers
- Win32 related filesystems related to themselves
- UNIX filesystems (Ext2, SYSV) related to each other.

Their Conclusions

- Linux might have cut some of the dependencies in the future
- Still needed humans to come up with the hierarchical decomposition
- Concrete Architecture had more dependencies than expected.
- Efficiency in Linux was sometimes improved via bypassing dependencies.
- Headers files could be moved to more local locations

Issues

- Was the concrete architecture simply interpretted via the conceptual architecture?
- We’re shown how the concrete model is almost a fully connected graph, it would be nice to compare this against other software systems.

Summary:

- Researched Linux and UNIX OSes to produce a conceptual model
- Analyzed Linux to produce a concrete model
- Discovered some unexpected interdependencies
- Refined conceptual model using the concrete model

Questions

- What about alternatives views of architecture (4+1, etc.)
- Did the conceptual architecture match mostly because that's what they were looking for?
- What other diagrams or comparisons could be shown to make this study more valuable?
- Are these dependencies bad?