SE 577
Software Architecture

The Software Architecture of Linux

Reference

I.T. Bowman, R.C. Holt, and N.V. Brewster: “*Linux as a Case Study Its Extracted Software Architecture*”, Proc. ICSE ‘99, Los Angeles, CA., pp. 555-563.

About the Paper

- Describes work done to document the software architecture of the Linux kernel
  - Linux kernel (~800KLOC) is responsible for:
    - Process management
    - Memory management
    - Hardware device management

Why Study This Work?

- Look at an example of the architecture of a large, complex, software system.

Why Study This Work?

- Look at an example of the architecture of a large, complex, software system.
- Observe the approach used to describe/document the architecture of the software

Why Study This Work?

- Look at an example of the architecture of a large, complex, software system.
- Observe the approach used to describe/document the architecture of the software.
- Understand what needs to be done to figure out the architecture of such system [*extraction methodology*].

Why Study This Work?

- Look at an example of the architecture of a large, complex, software system.
- Observe the approach used to describe/document the architecture of such software system.
- Understand what needs to be done to figure out the architecture of such system [extraction methodology].
- Point out the differences and significance between the **conceptual** and **concrete** architectures?

Architectural Documentation

- Two types of architectural documentation:
  - Conceptual architecture
    (*Prescriptive* architecture)
    - How developers think about the system
    - Relationships between subsystems that are meaningful to developers
  - Concrete architecture
    (*Descriptive* architecture)
    - Relationships that exist in the implemented system.

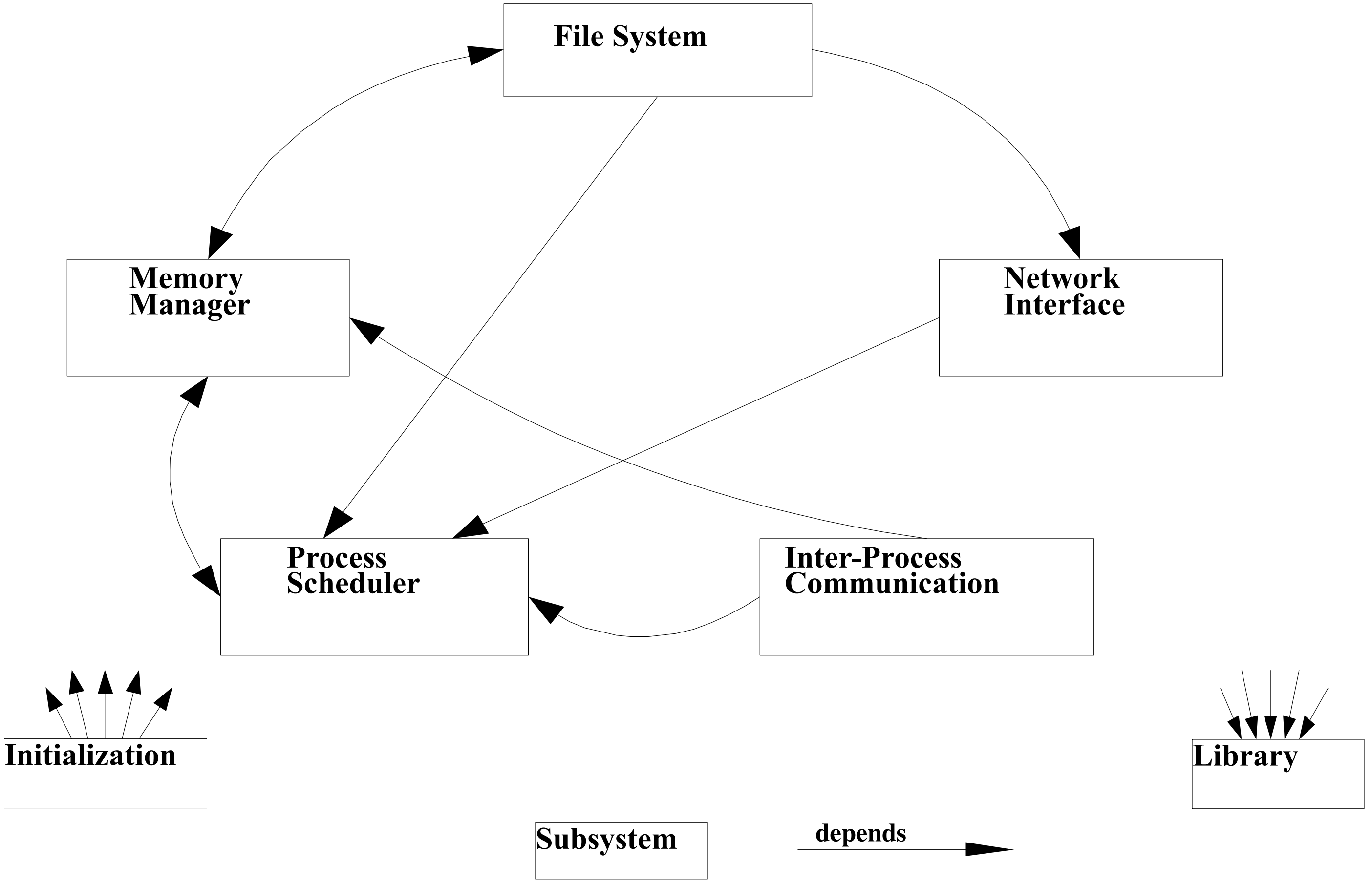
Architectural Documentation (cont'd)

- Architecture documentation
  - Individual functions and even modules are not described in detail.
  - Subsystems and relations between them are documented.
- Architectural documentation is valuable for understanding the system.
- Great help for evaluation or re-engineering efforts.

Architectural Documentation (cont'd)

- Unless architectural documentation is maintained it will become obsolete as the system undergoes further changes.
- Others have proposed work to keep architectural documentation up to date (e.g., *Software Bookshelf* by Finnigan, et al.)

Conceptual Architecture



Conceptual Architecture (cont'd)

- Diagram shows "depends-on" relationship
  - E.g., Memory Manager depends on the File System to swap memory to and from disk.
- Each subsystem has additional subsystems hierarchically nested within it (not shown).
- The Initialization subsystem depends on all other kernel subsystems (it calls initialization routines).
- All the kernel subsystems depend on the Library subsystem.

Conceptual Architecture (cont'd)

Seven major subsystems:

- Process Scheduler
  - Responsible for supporting multitasking by changing which user process executes.
- Memory Manager
  - Provides a separate memory space for each user process and uses swapping to support more processes than fit in physical memory.
- File System
  - Provides access to hardware devices. User processes can access keyboards, tape drives, hard drives, and modems using one interface that is implemented by the File System.

Conceptual Architecture (cont'd)

- Network Interface
  - Encapsulates access to network devices in a similar manner to the File System. User processes can communicate with other computers using several different types of network hardware and transmission protocols.
- InterProcess Communication
  - The IPC subsystem allows user processes to communicate with other processes on the same Computer. Synchronization memory sharing and interprocess messaging primitives are supported by the IPC subsystem.

Conceptual Architecture (cont’d)

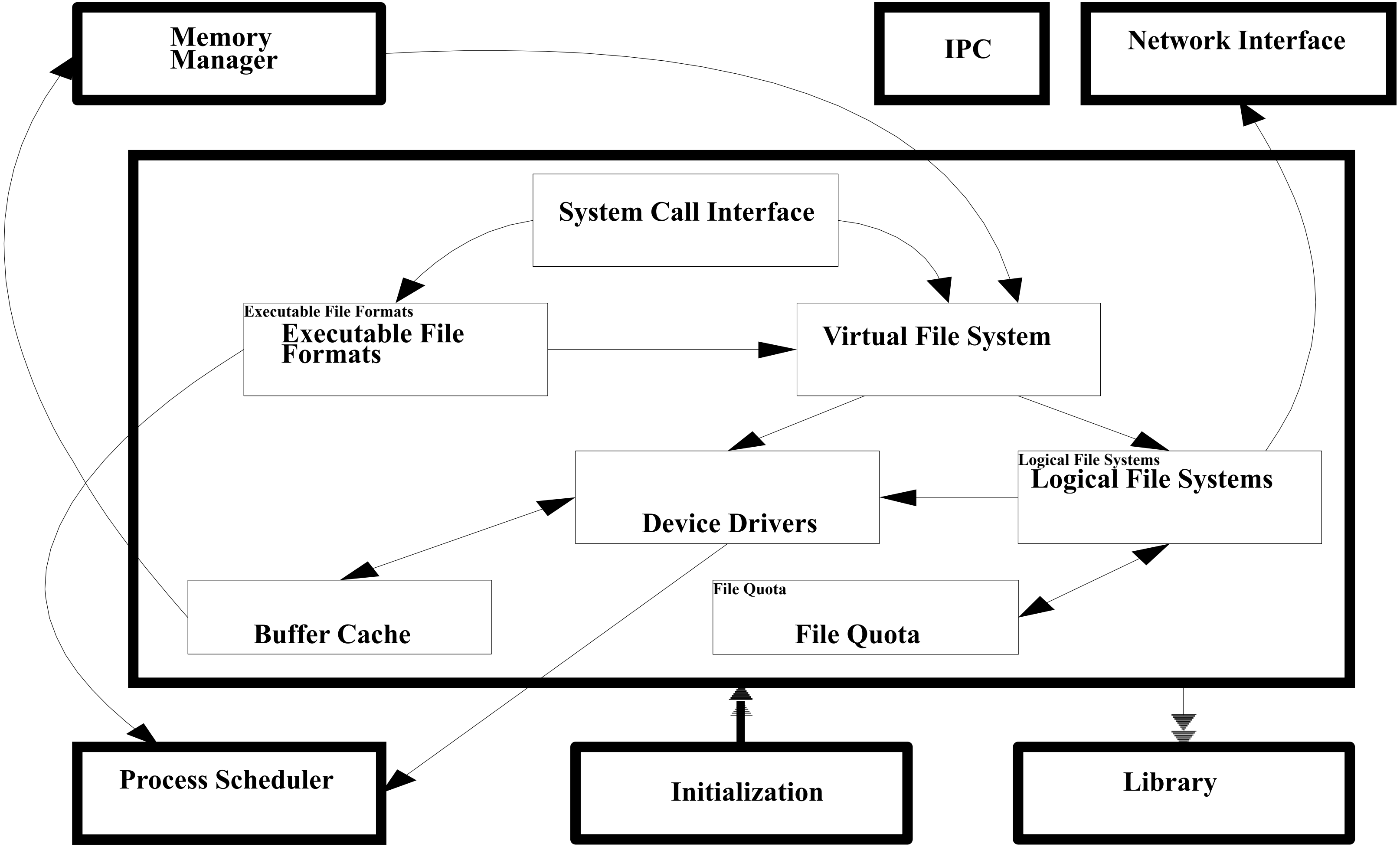
- Initialization
  - Responsible for initializing the rest of the Linux kernel with appropriate user configured settings.
- Library
  - Contains routines which are used throughout the kernel.

File System Conceptual Architecture

Three main roles:

- Provide access to a wide variety of hardware devices.
- Support several different logical file system formats that control how files are mapped to physical locations on hardware devices.
- Allow programs to be stored in several executable formats including interpreted scripts.

File System Conceptual Architecture (cont'd)



The File System

Five subsystems implement the main roles of the File System:

- Device Drivers
  - Perform all communication with hardware devices supported by Linux.
- Logical File Systems
  - Implement several logical file systems that can be placed on hardware devices; these different file systems allow interoperability with different operating systems and also allow specialized functionality such as encryption compression and high performance.

The File System (cont'd)

- Executable File Formats
  - Allow clients to execute programs from several different executable file formats including not only compiled programs but also interpreted scripts.
- File Quota
  - Allow system administrators to limit the amount of file storage that individual users may use.
- Buffer Cache
  - Provide memory buffers for input/output operations and reduces hardware accesses by caching data and eliminating redundant reads and writes.

The File System (cont'd)

Unexpected relationships from Conceptual architecture:

- Device Drivers ➔ Process Scheduler
  - A device driver might request to be suspended while waiting for i/o to complete.
- Logical File System ➔ Network Interface
  - Some logical files are stored on another computer and accessed using the network.
- Memory Manager ➔ Virtual File System
  - Swap memory to and from secondary storage.
  - [Not sure why this was unexpected…]

The File System (cont'd)

- Linux uses the Façade design pattern to allow user processes and other parts of the kernel to use elements of the File System through a single interface.
- The File System follows the “object-oriented” or “data abstraction” style (described by Garlan and Shaw).

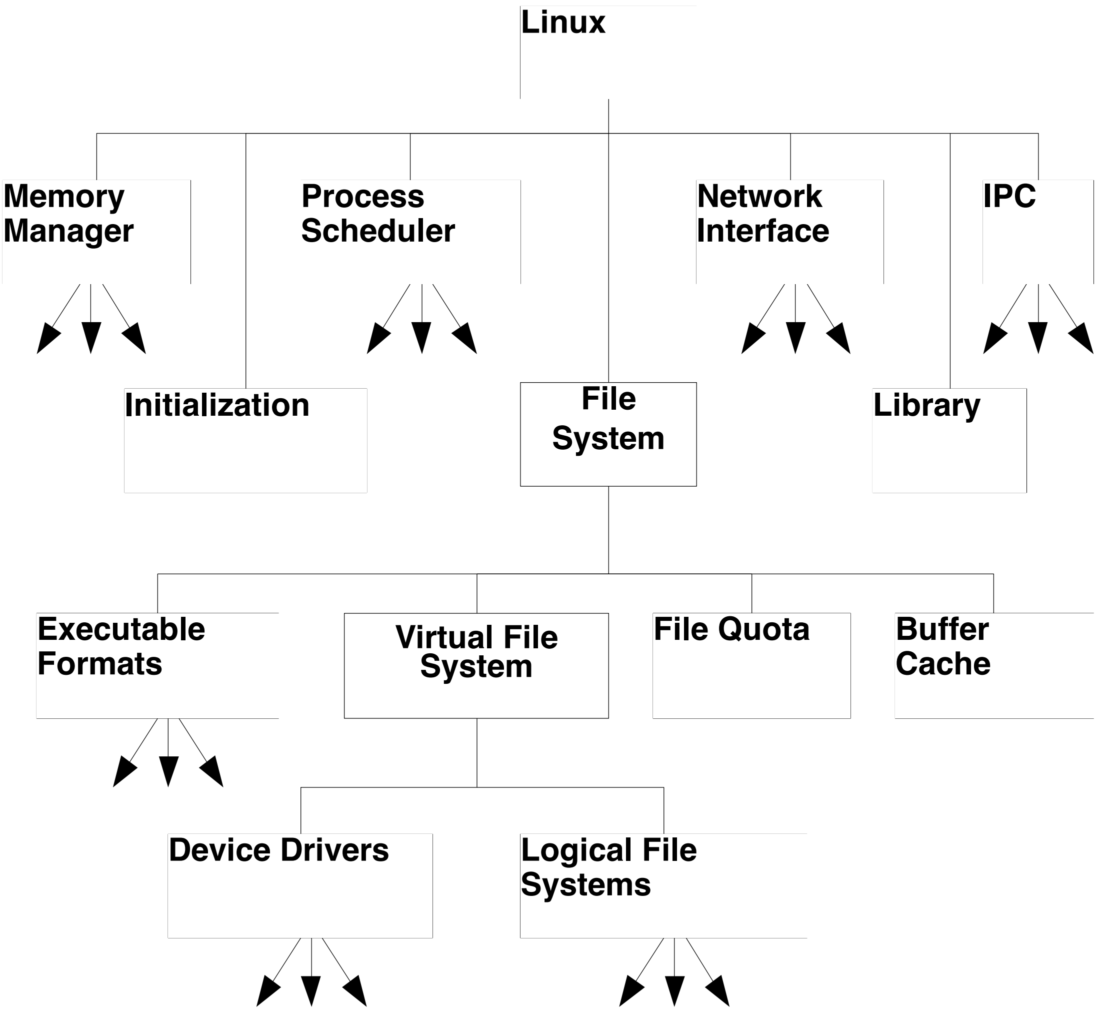
Extracting the Concrete Architecture

- Kernel contains 1,682 source files.
- Relations between source files are at too low a level for easy system understanding.
- Focus on relations between subsystems.
- Used tools (grok and lsedit) to determine what relations exist between subsystems, based on the relations between the source files that are contained in the subsystems.
- Combined tool support and human interpretation to extract the concrete architecture.

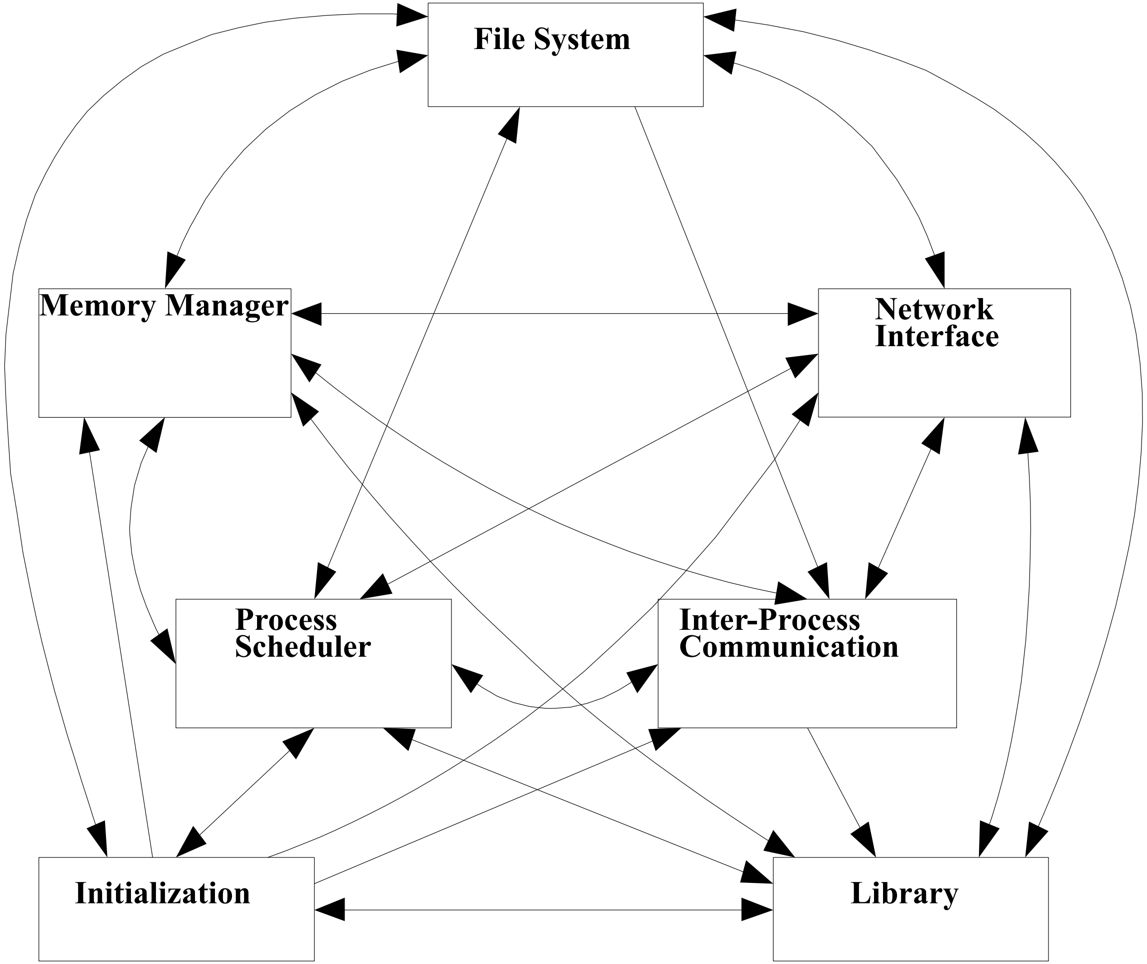
Extraction Methodology

- Manually created a tree structure of subsystems.
- Assigned each subsystem to a single containing subsystem (used the subsystems from conceptual architecture.)
- Manually assigned source files to subsystems based on:
  - directory structure
  - file naming conventions
  - source code comments
  - documentation
  - examination of the source code
- If a set of source files seemed logically related, created a new subsystem to contain them.

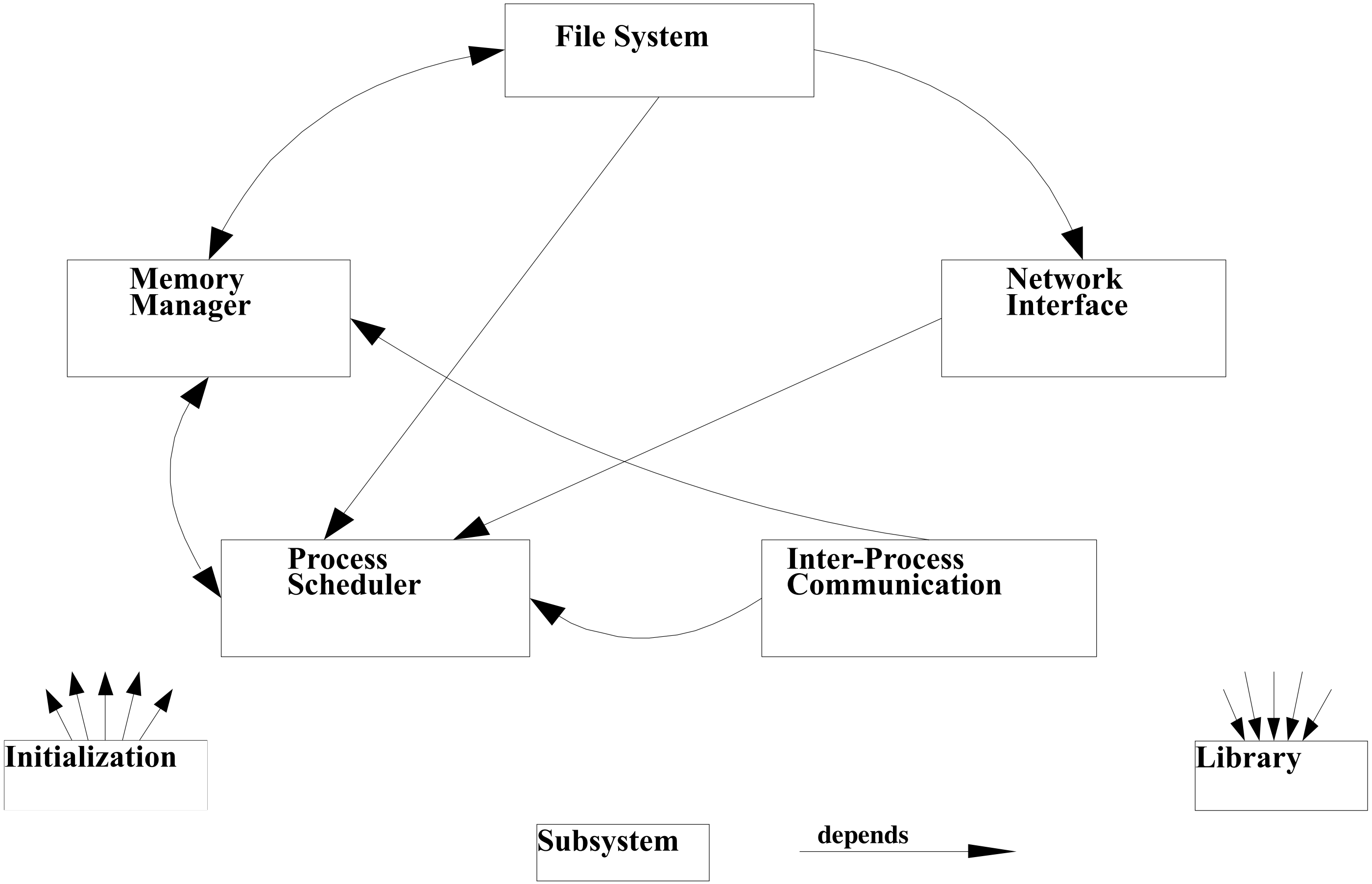
Hierarchical Decomposition (Partial)



Concrete Architecture



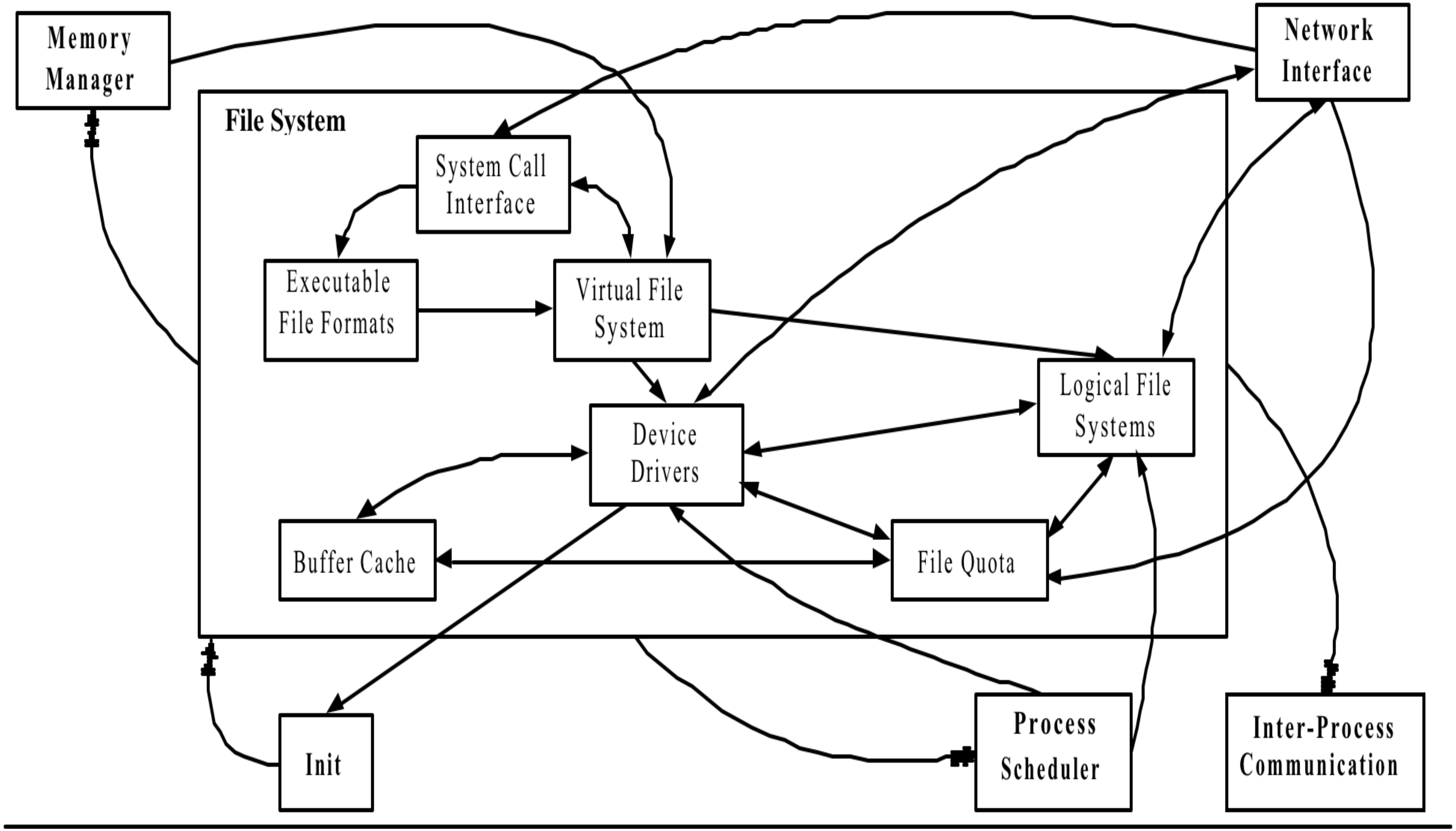
Conceptual Architecture



Concrete vs Conceptual Architecture

- Same subsystems
- Dependency relations appear to be quite different
  - Concrete architecture almost fully connected.
- Lesson learned: a concrete implementation is likely to have more dependencies. Why?
  - Developers might avoid existing interfaces for better efficiency.
  - Unclear if the dependencies are required.

File System Concrete Architecture



File System Concrete Architecture (cont'd)

- Same subsystems.
- Also has substantially more dependency relations. Why?
  - Missed dependencies in the documentation.
  - Functionality that was implemented in multiple subsystems.
  - Unexpected control flow implementations.

File System Concrete Architecture (cont'd)

- Some unexpected relations:
  - Network Interface ➔ File System
    - The Network Interface directly calls functions in the implementation of two logical file systems (NCPFS, SMBFS).
  - Process Scheduler ➔ Device Drivers
    - A Process Scheduler routine calls a routine implemented within the Device Drivers.
  - File System subsystems ➔ IPC
    - Synchronization primitives are also used by the kernel.

Logical File System Concrete Architecture

- Contains 17 different logical file systems.
- They map logical files to physical locations on storage devices.
- Has more dependencies than predicted in the conceptual architecture.

Takeaways

- Automated tools are very important, but tools alone are not enough.
- Human participation still required.
  - Source code examination is part of the deal
- Remember: the architecture of a system is not "visible".
- Substantial differences between conceptual vs. concrete architecture
  - Missed the use of subsystems (e.g., IPC implements mechanisms used throughout the kernel).
  - Existing interfaces are bypassed for efficiency reasons.

Takeaways (cont'd)

- The concrete architecture should be used to refine the conceptual architecture.
- May not be desirable to add all relations
  - Many are not essential and hinder system understanding.
- Concrete architecture identifies potential improvements.