FreeBSD Concrete Architecture

pink fluffy unicorns

EECS 4314
Advanced Software Engineering

March 8th, 2023

OUR TEAM

Benedict Miguel
Jiachen Wang
Keshav Gainda
Ketan Bhandari
Mohaimen Hassan

Pegah Fallah
Shami Sharma
Sidharth Sudarsan
Suryam Thaker
Xiaochuan Wang

AGENDA

01 **Derivation Process**

Subsystem assignments and derivation of contain files

02 **Top Level Architecture**

Concrete architecture, style and functionality

03 **Interactions**

Between subsystems

04 **Reflexion Analysis**

Conceptual model
vs
Reflexion model

05 **LSEdit Demo**

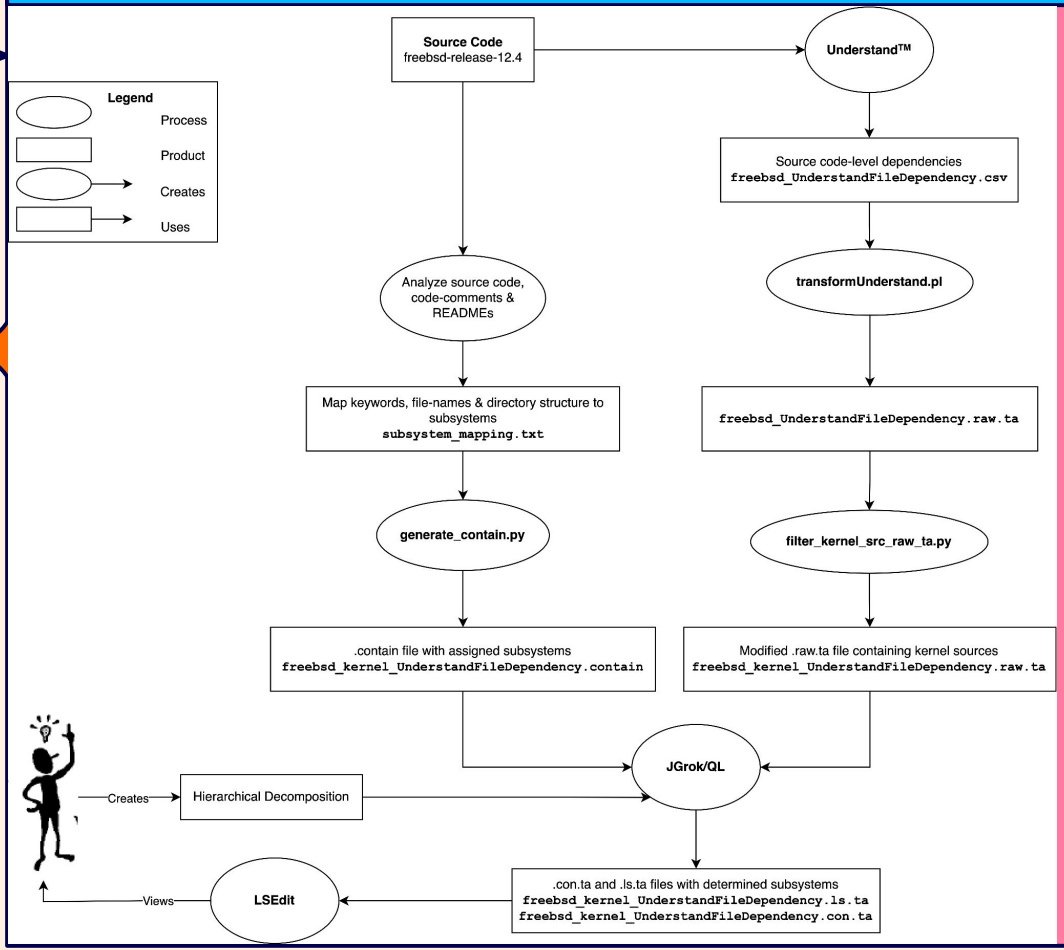
Extracted architecture and subsystems demo on LSEdit

06 **Lessons Learned**

Tools

- **Understand** by SciTools: Static Code Analysis Tool
  - Source code-level dependencies:
    - Extracts functions calls and variable access relations (e.g. function y calls function x)
    - Control flow and data flow dependencies
- **JGrok/QL**
  - Determine relations between files and subsystems
- **LSEdit**: The Graphical Landscape Editor
  - Visualize extracted system architecture
- **Python**
  - Scripting for processing, filtering and generating data

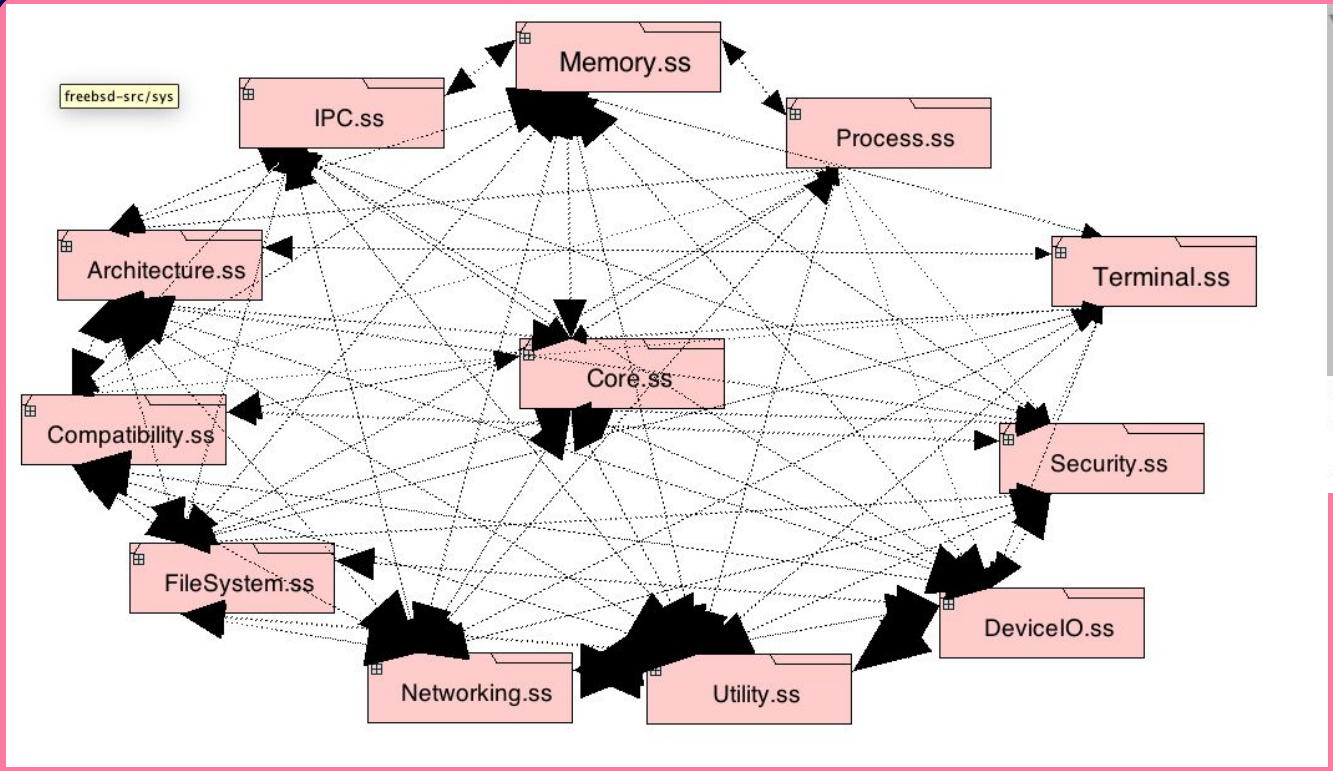
DERIVATION PROCESS



- /sys/ contains the source files of the kernel
- Assigned subsystems based on the subfolder structure of /sys/
- Determined the subfolders that may or may not be associated with a subsystem
- Analyzed the file-naming, code comments, READMEs and source-code function headers

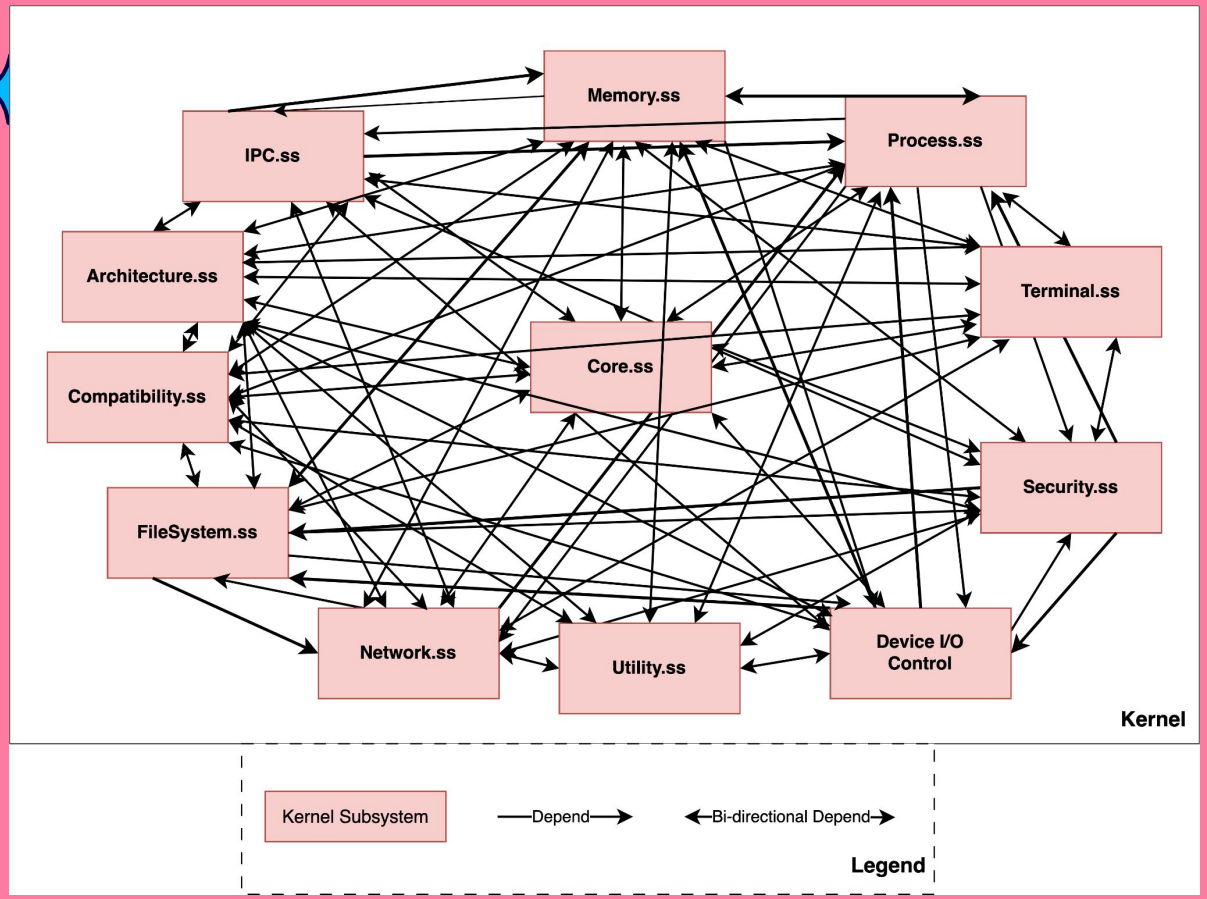
Extracted from LSEdit

Top-Level Concrete Architecture

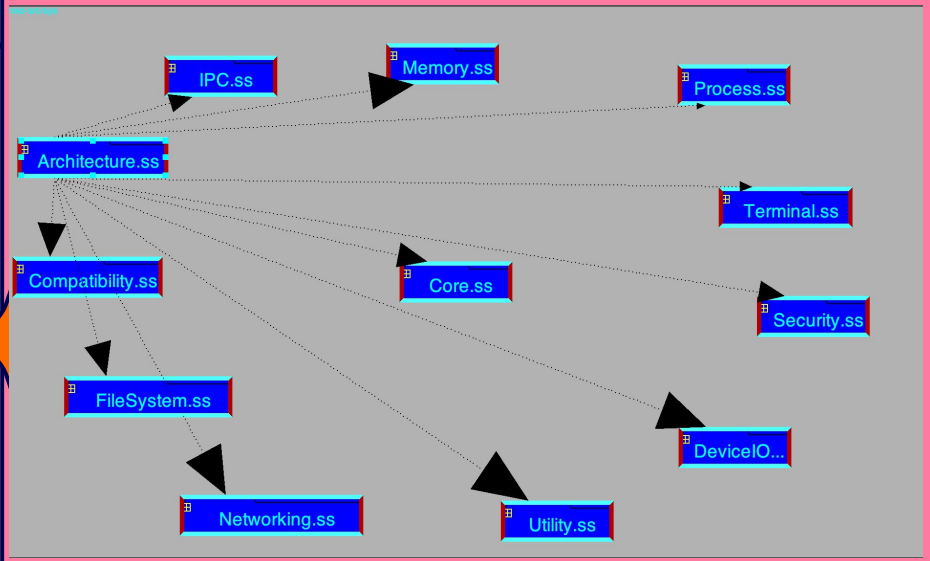


Top-Level Concrete Architecture

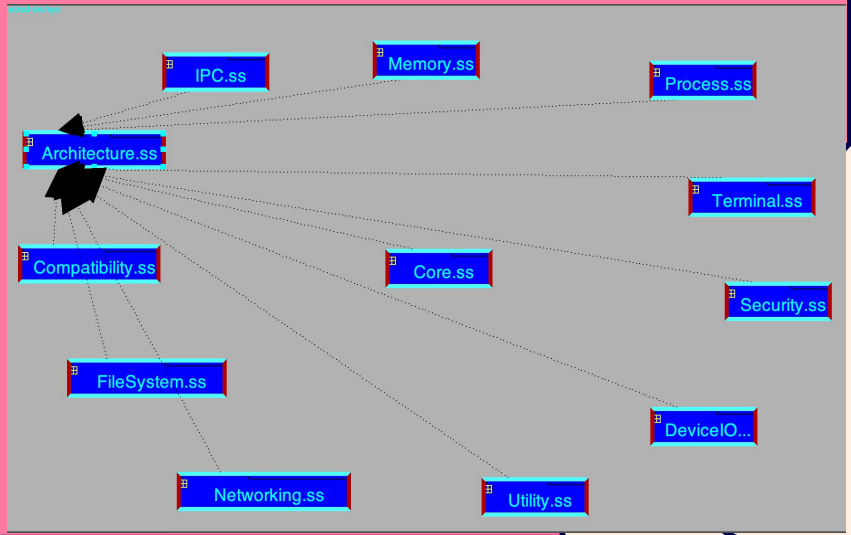
Monolithic Kernel



ARCHITECTURE

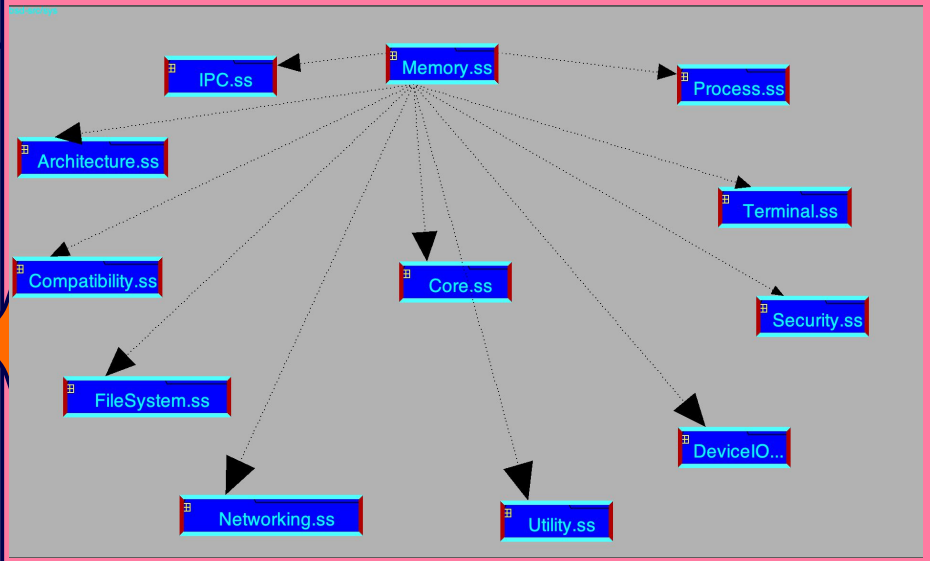


Architecture-dependent code

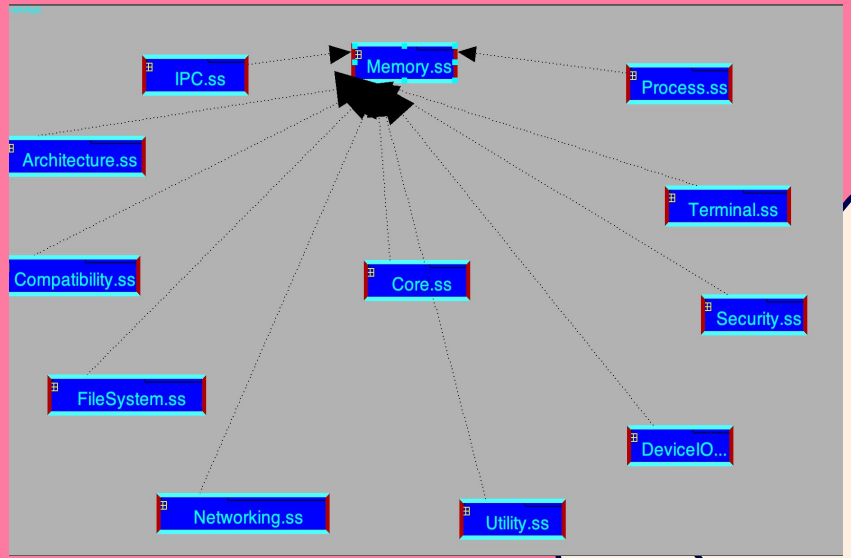


amd64	risc-v
arm64	x86
i386	mips
powerpc	sparc64

MEMORY

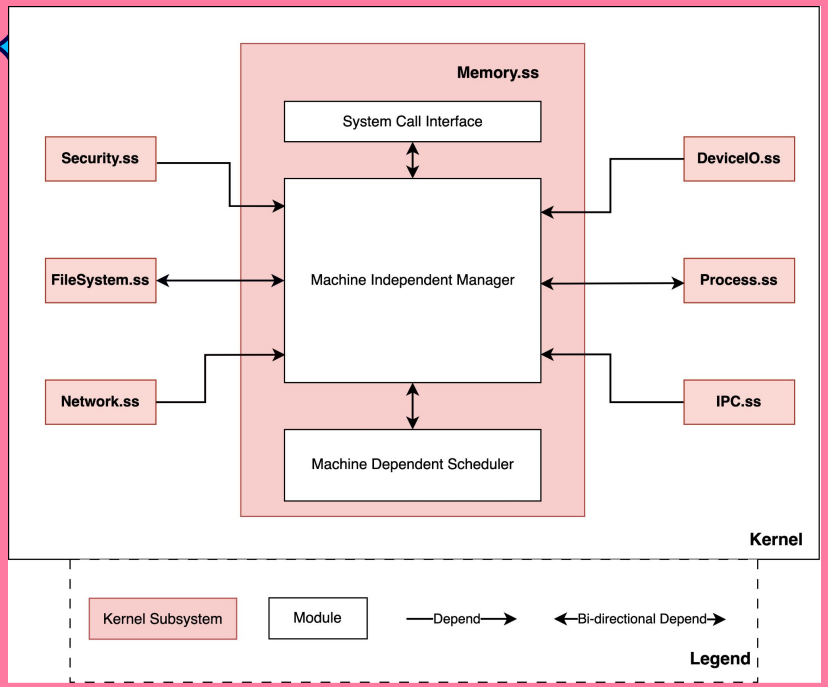


- Implementing the virtual memory abstraction
- Syscalls: *mlock, execve, read, write, etc.*
- Page-fault handling
- Kernel memory allocation
  - *malloc, free*

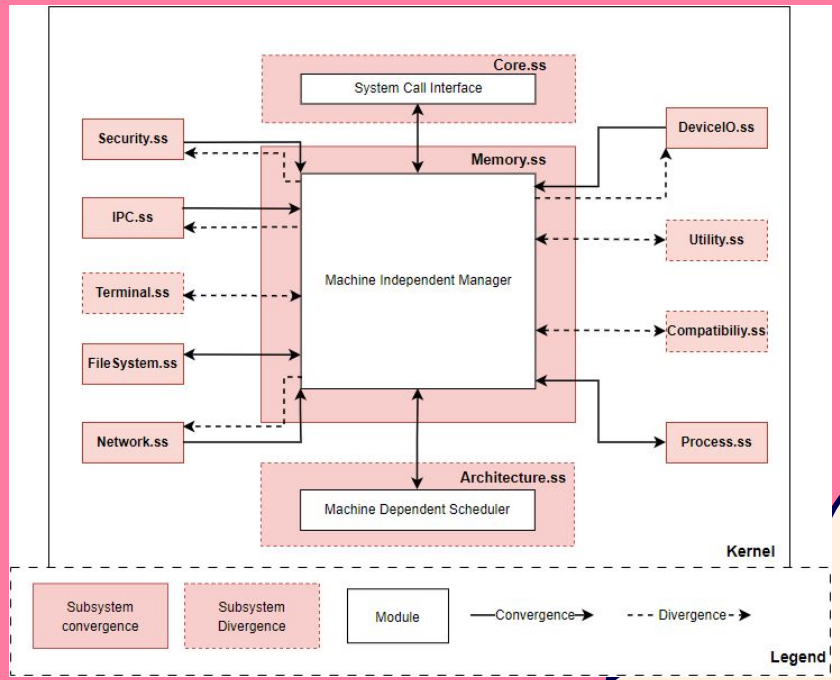


- Handling memory pressure
  - Reclaiming pages
  - Approximating LRUs
- Coupled with miscellaneous subsystems
  - *tmpfs, pmap*
  - Scalability w.r.t CPU & RAM

MEMORY SUBSYSTEM

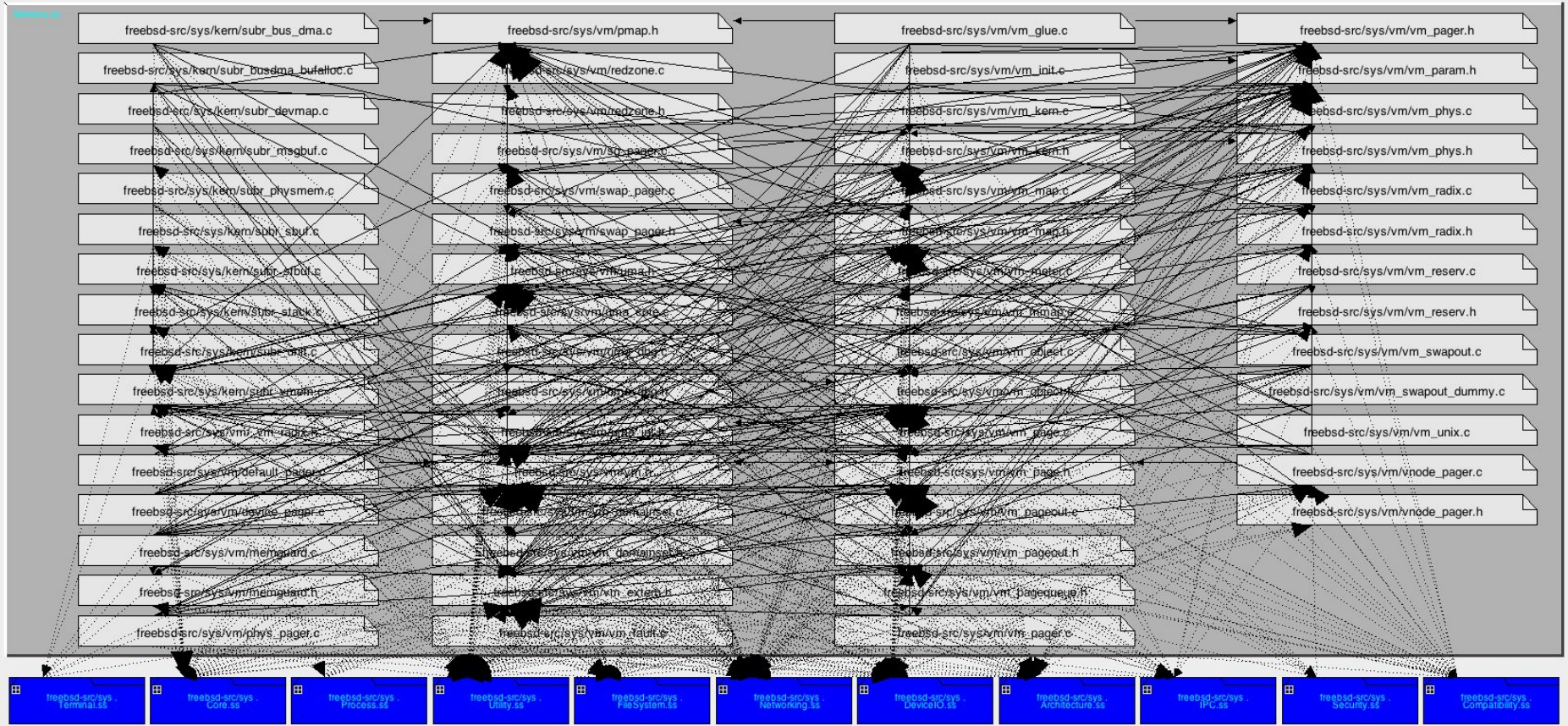


Conceptual Model

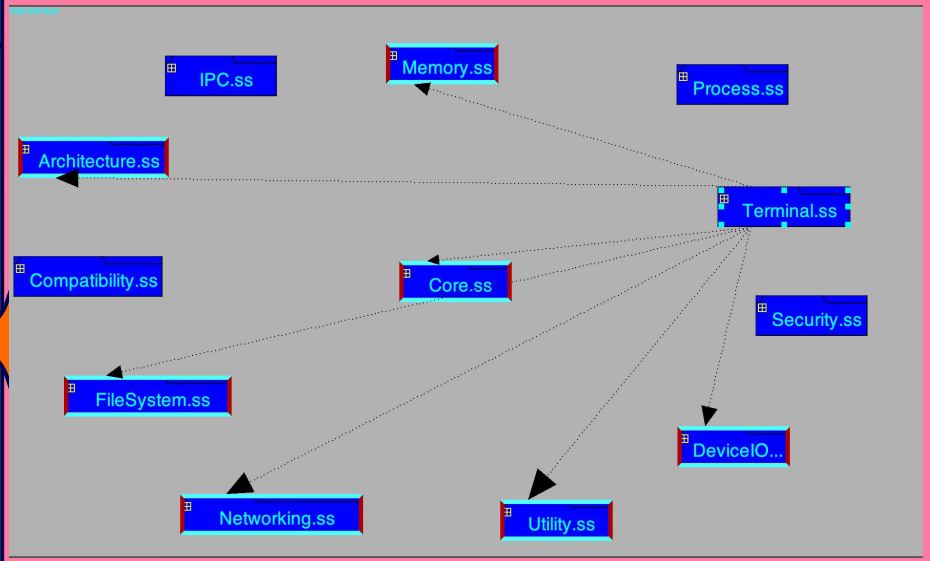


Reflexion Model

MEMORY: LSEDIT MATRIX LAYOUT



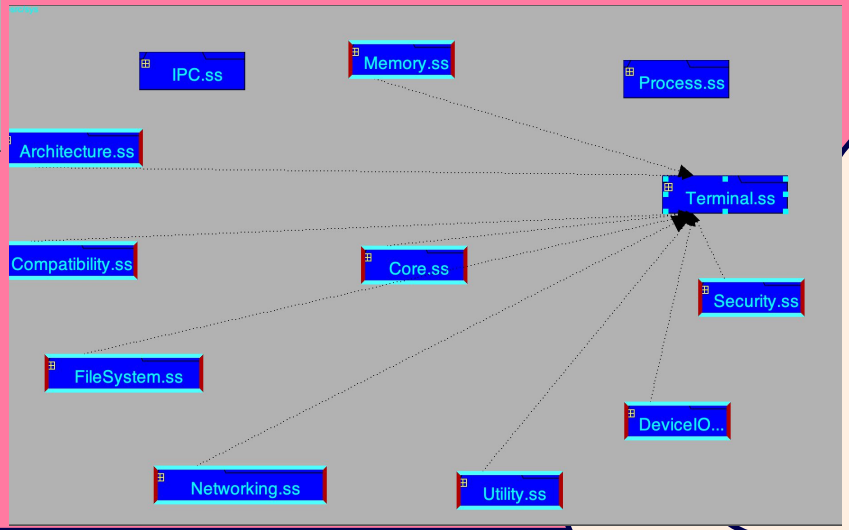
TERMINAL



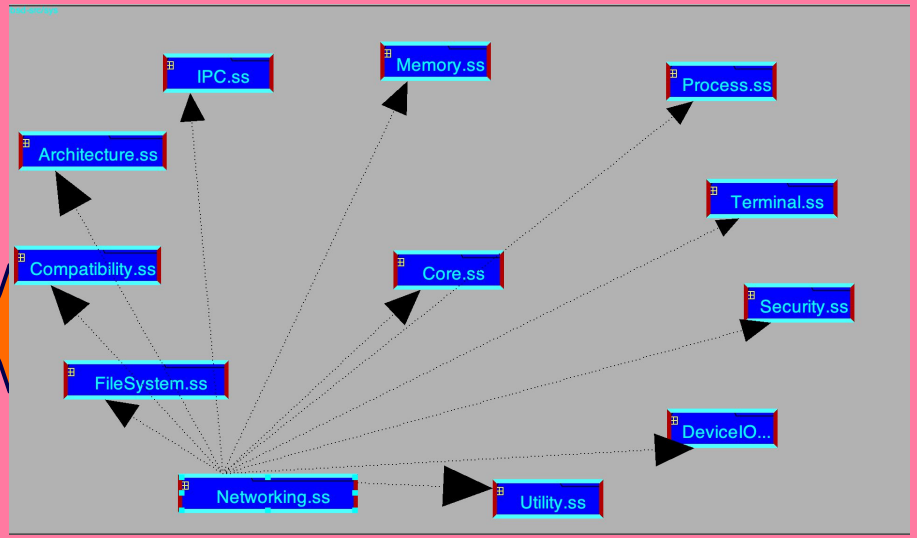
- Terminal driver
- Terminal emulator
- Virtual Terminal Manager

User Interactions

- Manage the interactions between the user and the system through an interface



NETWORKING

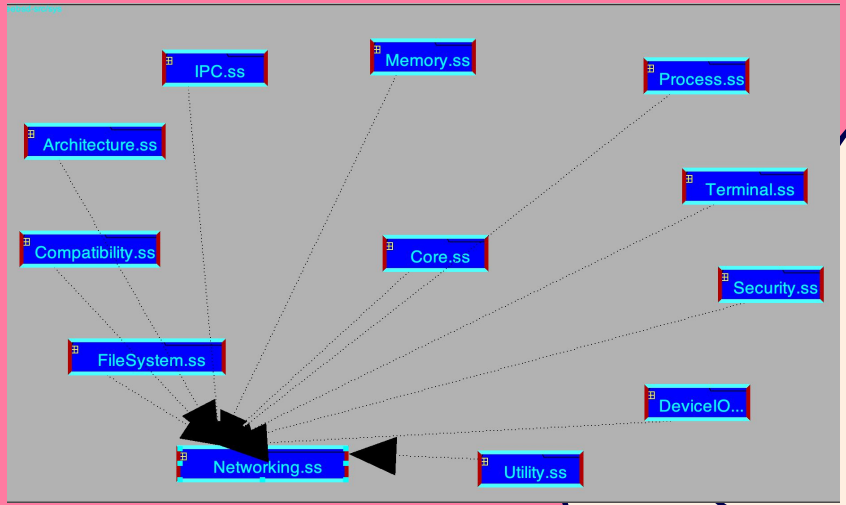


/sys/net - core networking code

/sys/net80211 - wireless networking (IEEE 802.11)

/sys/netgraph - graph-based networking subsystem

/sys/netinet - IPv4 protocol implementation

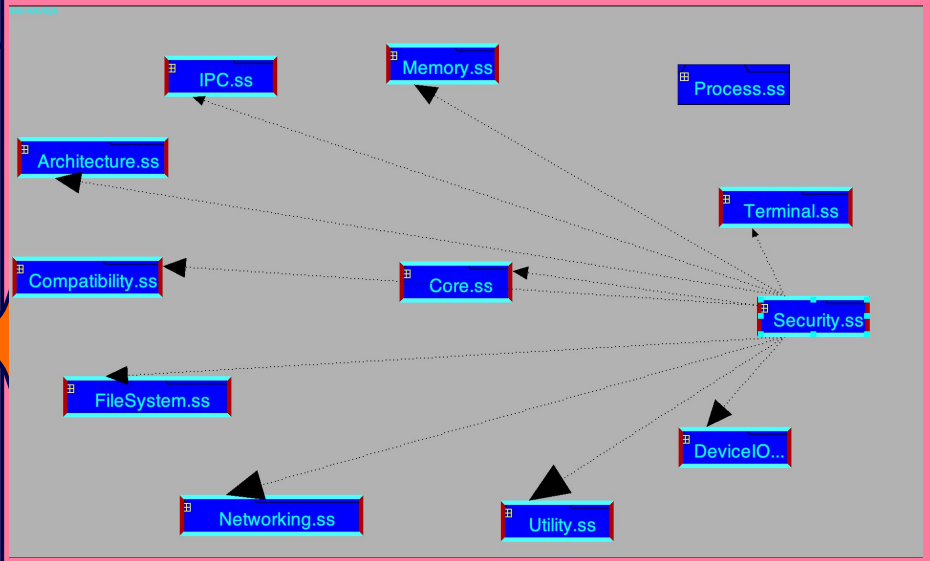


/sys/netinet6 - IPv6 protocol implementation

/sys/netipsec - IPsec protocol implementation

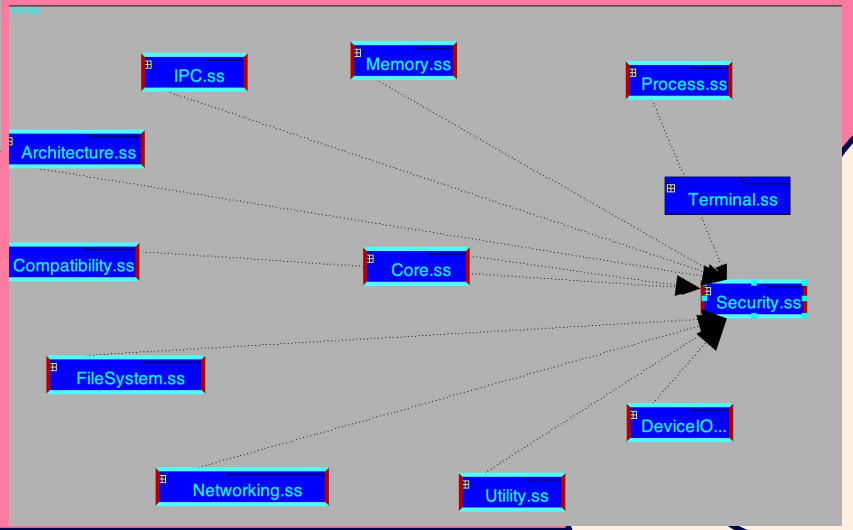
/sys/netpfil - packet filters

SECURITY



Implements Multilevel Security policies:

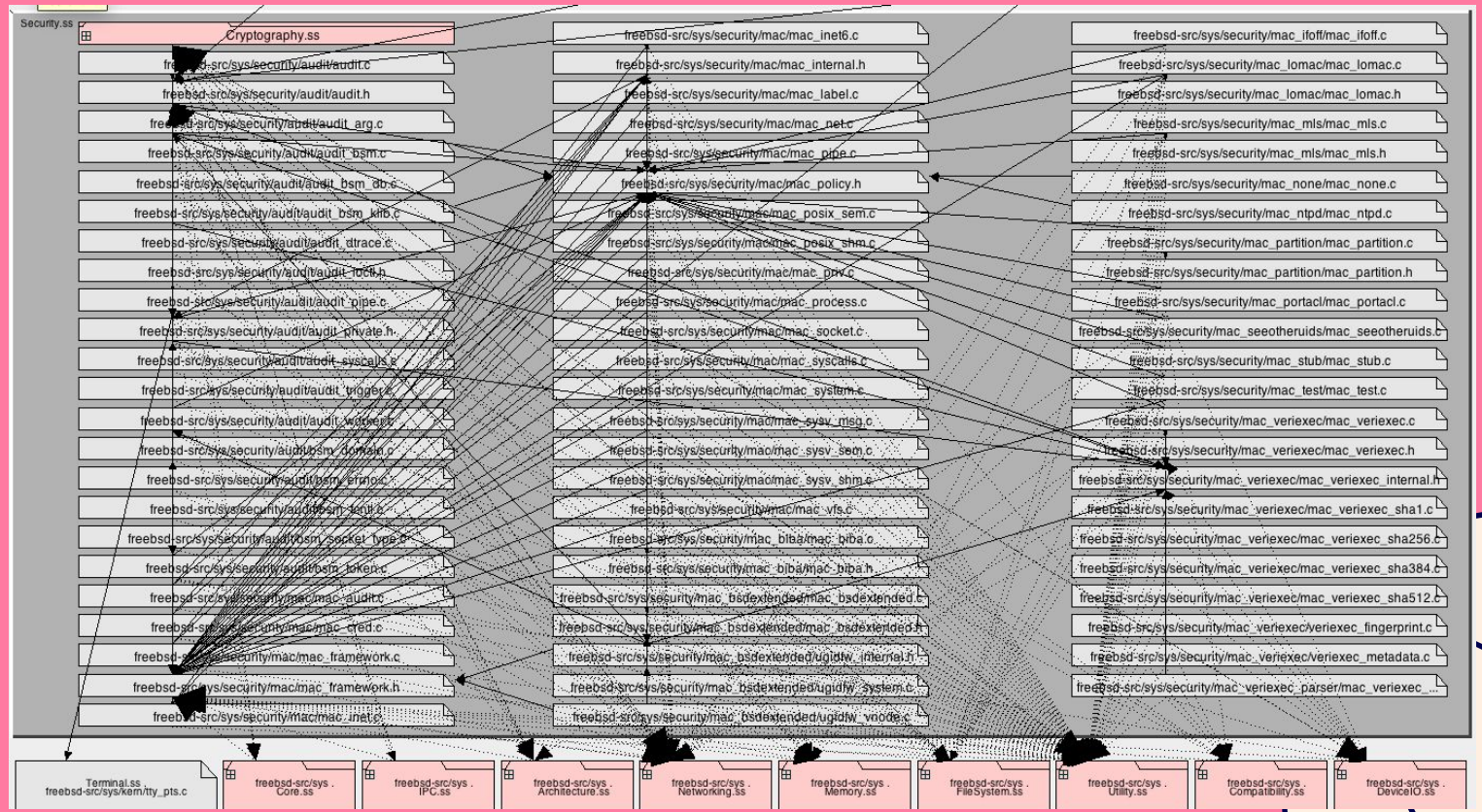
- Mandatory access control (MAC)
- Discretionary access control (DAC)



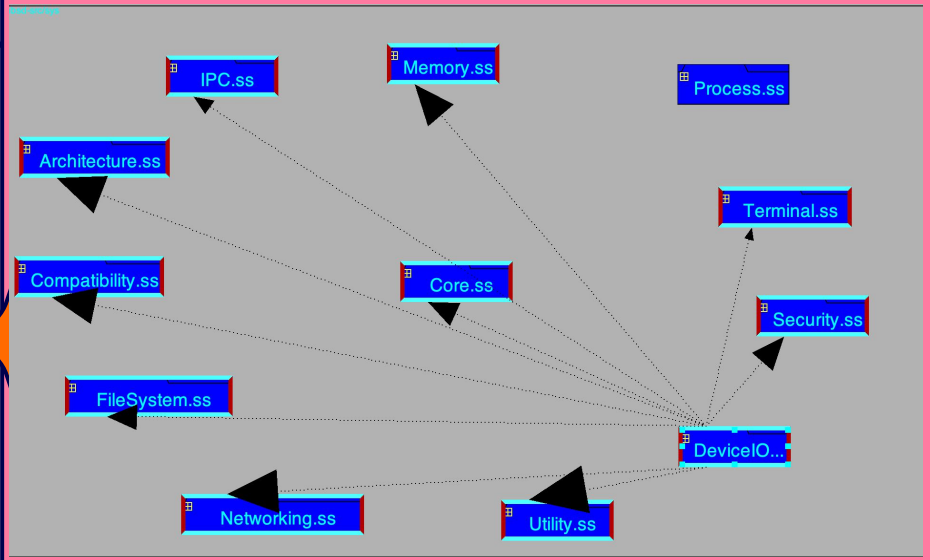

```
contain Security.ss Cryptography.ss
```

opencrypto is a framework for drivers of cryptographic hardware to register with the kernel so ``consumers'' (other kernel subsystems, and eventually users through an appropriate device) are able to make use of it.

SECURITY: LSEDIT MATRIX LAYOUT



DEVICE IO



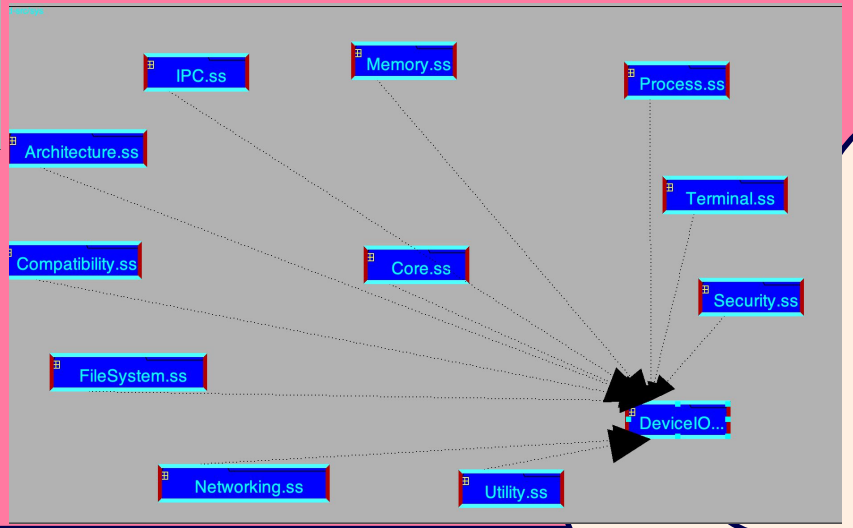
Device driver interface

- Hardware to OS

Bus management

- Build connections to the bus

- Device allocation and management
- I/O scheduling
- Buffer management



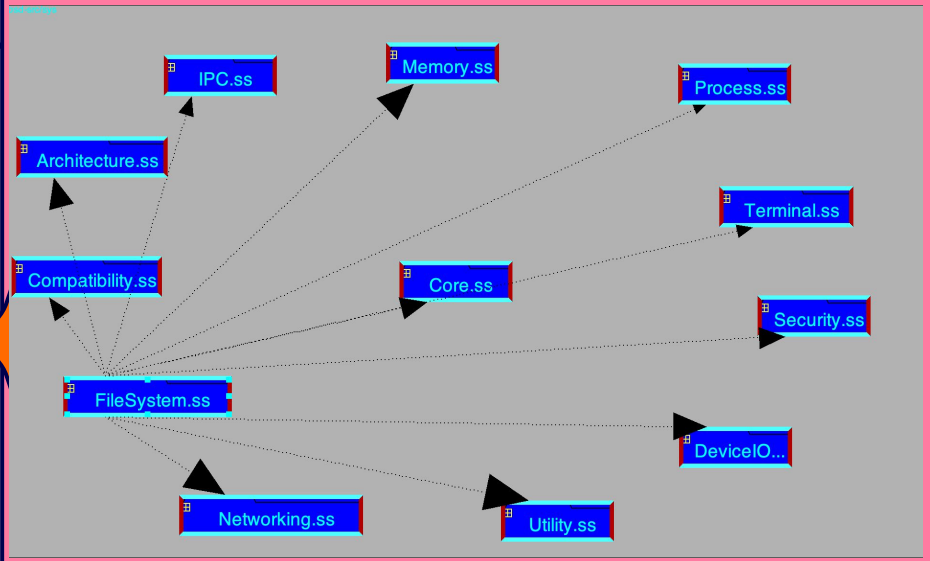
I/O Device Management

- Kernel acts as an intermediary between the hardware and the operating system
- Responsible for providing the abstract view of the hardware to the operating system and handling all I/O requests.
- FreeBSD provides more advanced I/O device management systems

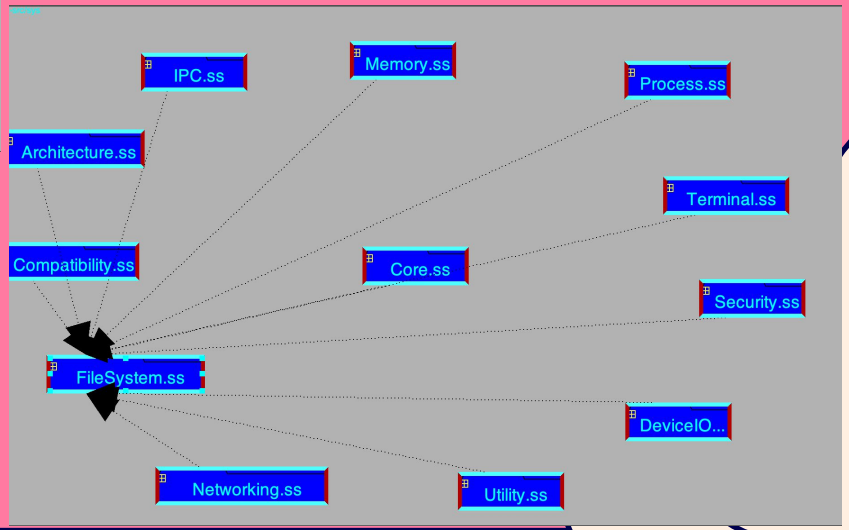
  -Complete Fair Queueing (CFQ)

  -Deadline Scheduling
- Able to handle very large amounts of I/O
- Ideal in enterprise environments where the I/O requirements are very demanding

FILE SYSTEM



- UFS
- ZFS
- NFS



File Management

- Uses Unix File System (UFS) to store files.

- Files are managed in a tree structure

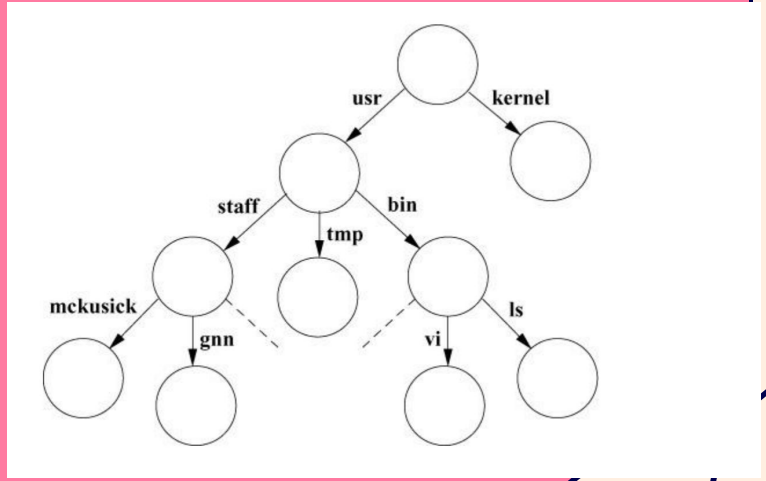
- Also provides a graphical method of file management

Zettabyte File System

- Data Compression, Check Summing, Snapshots

NetWork File System

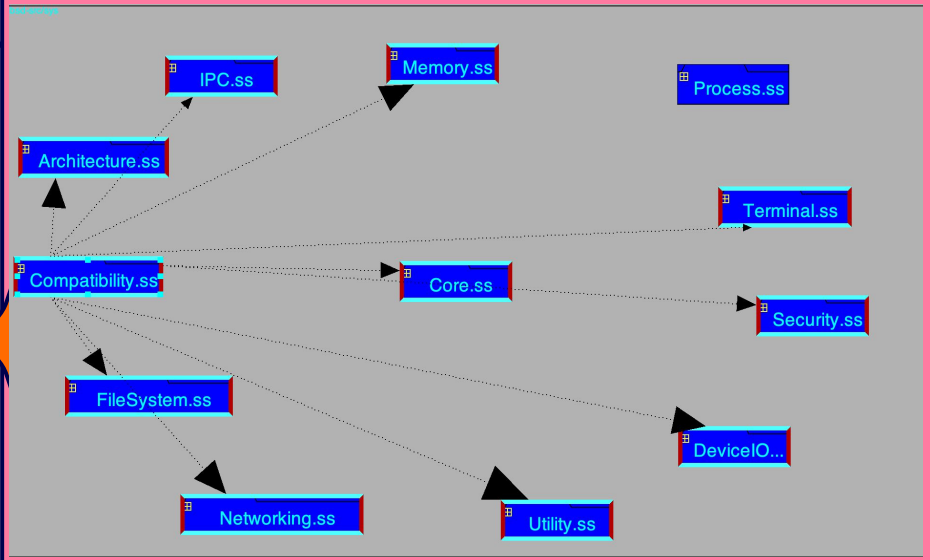
- Access to files over the network



- Been known to support various other file-management systems.

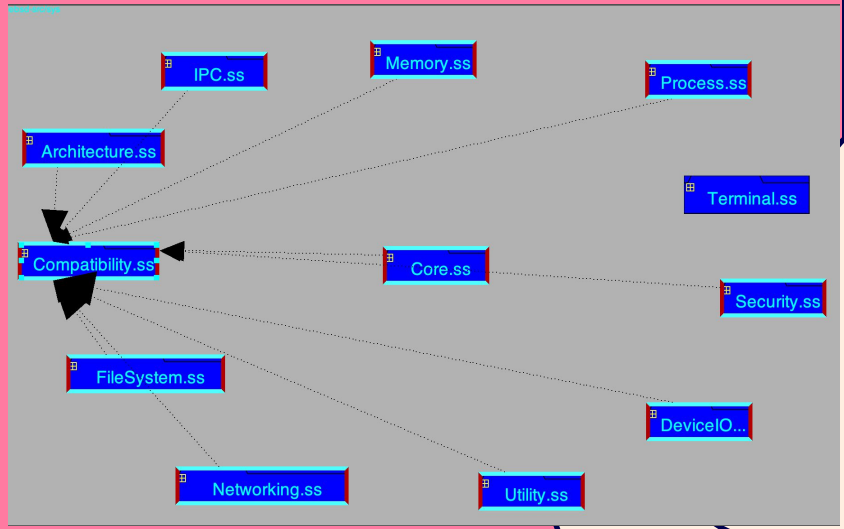
- Very ideal in multi-device environments.

COMPATIBILITY

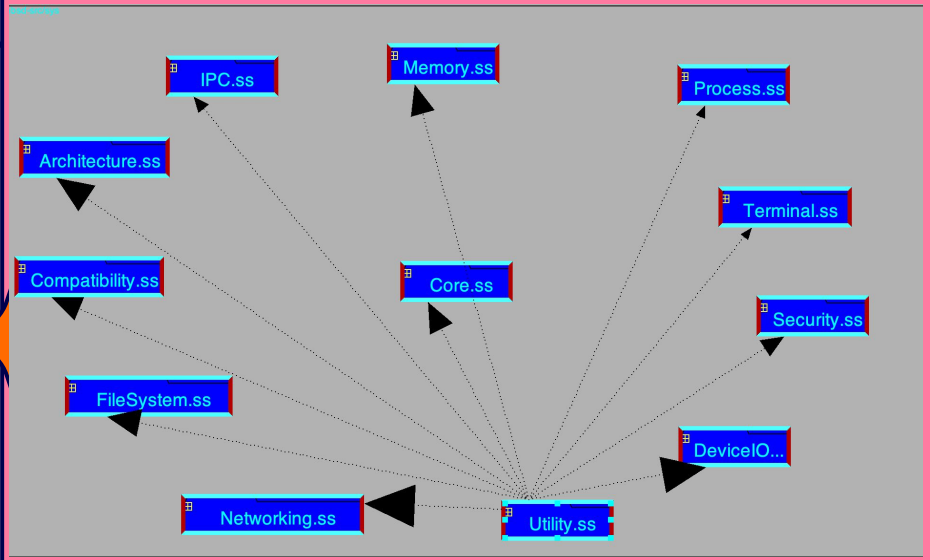


Supporting binaries built for Linux

Separation of concern to support other binaries, such as ones built for embedded systems, in the future as well

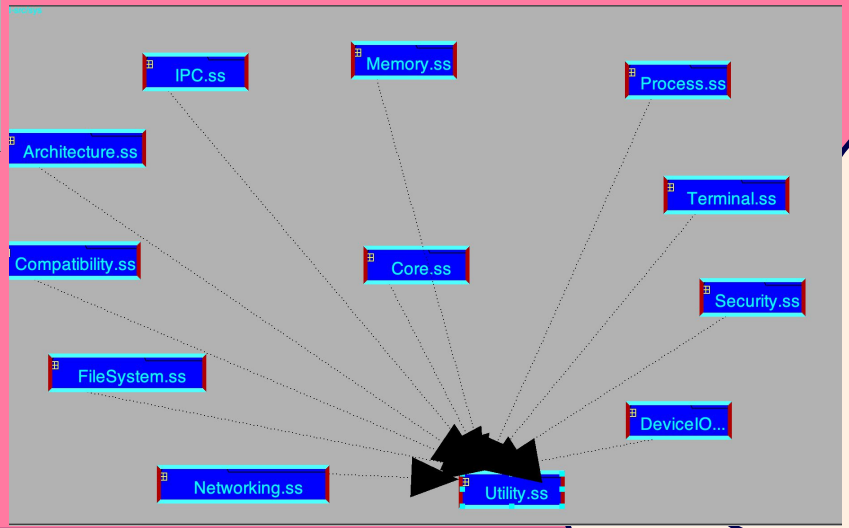


UTILITY

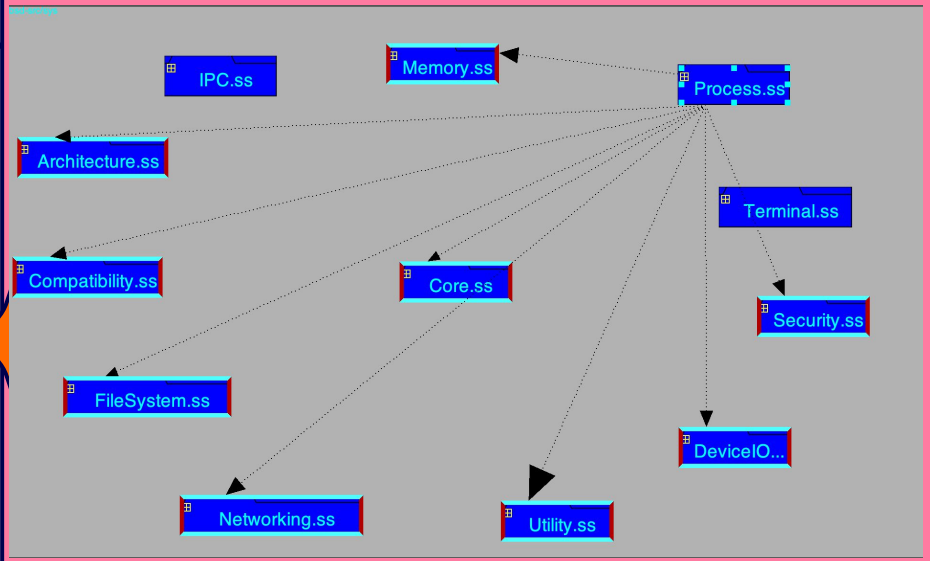


contain Utility.ss Debug.ss

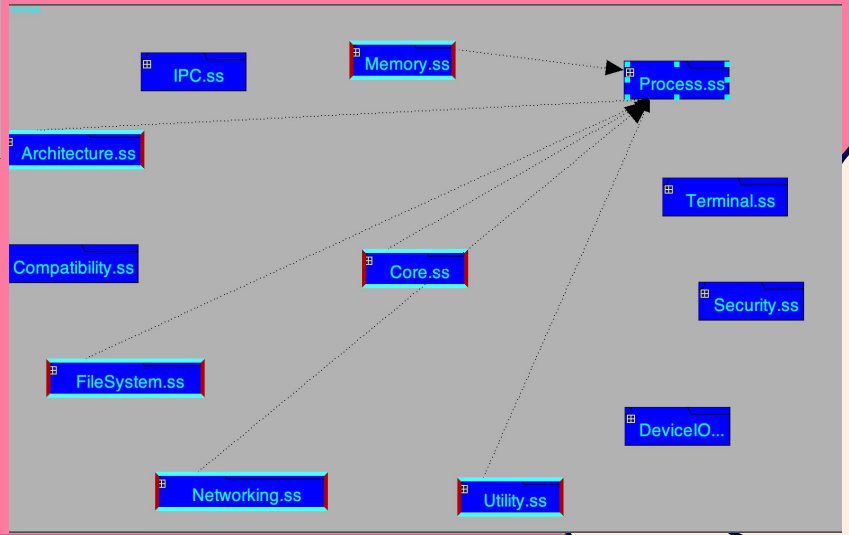
contain Utility.ss External.ss



PROCESS

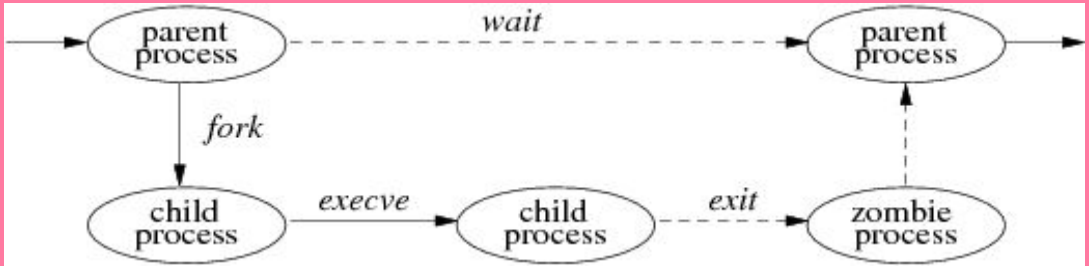


Dependencies of Processes



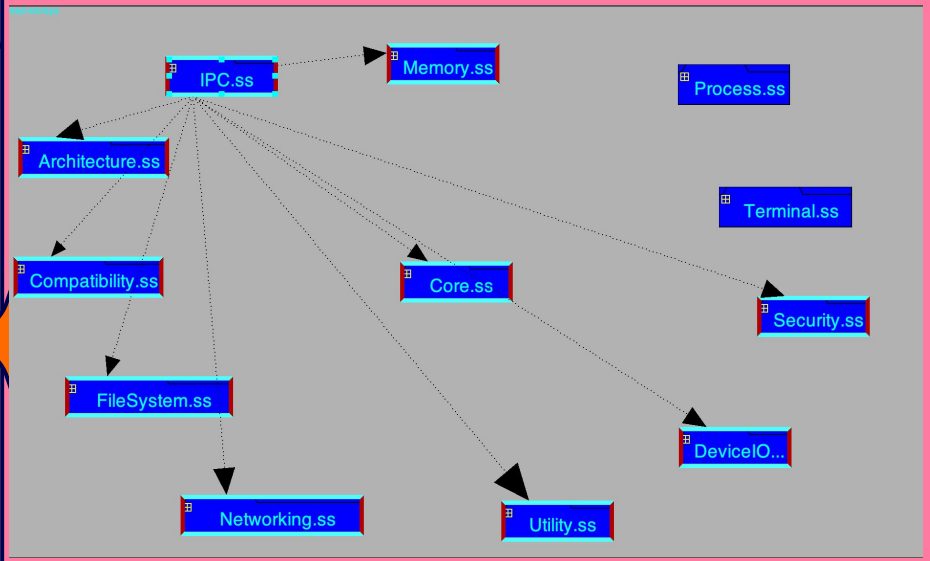
Dependencies on Processes

Process Management



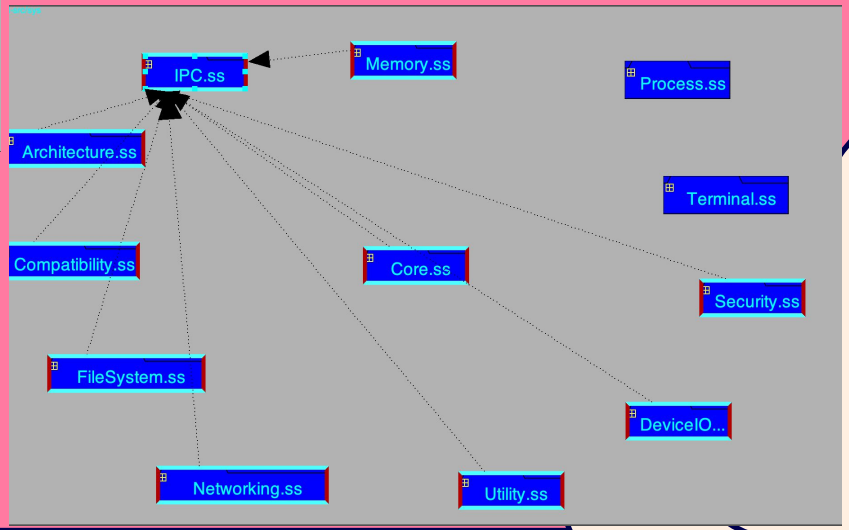
- Processes managed by the kernel
  - Using a priority-based algorithm
- *Jail* system for isolation and accessibility control for different processes
- The image shown depicts the life-cycle of a process
- Use of multiple fork calls for process creation
- Parent-child hierarchical structure for effective running

IPC



Sockets

- Abstract data-type used for communicating within the domains



Protocols

- Sockets must respect the protocols used by the domain

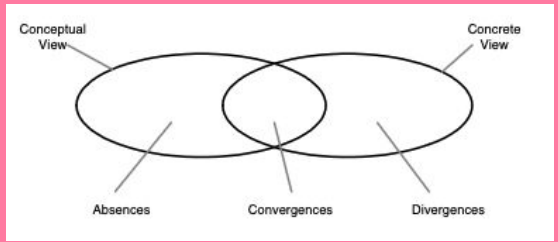
REFLEXION ANALYSIS

Conceptual Model

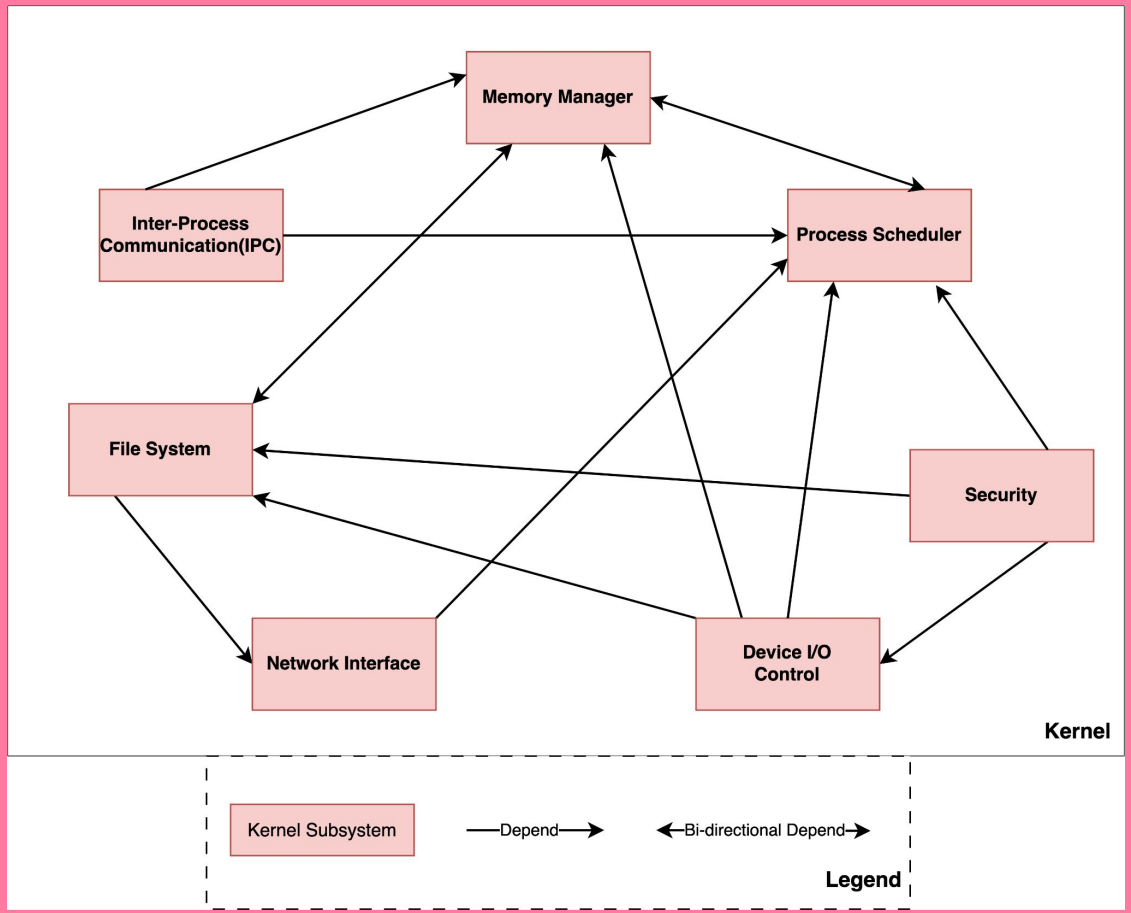
vs

Reflexion Model

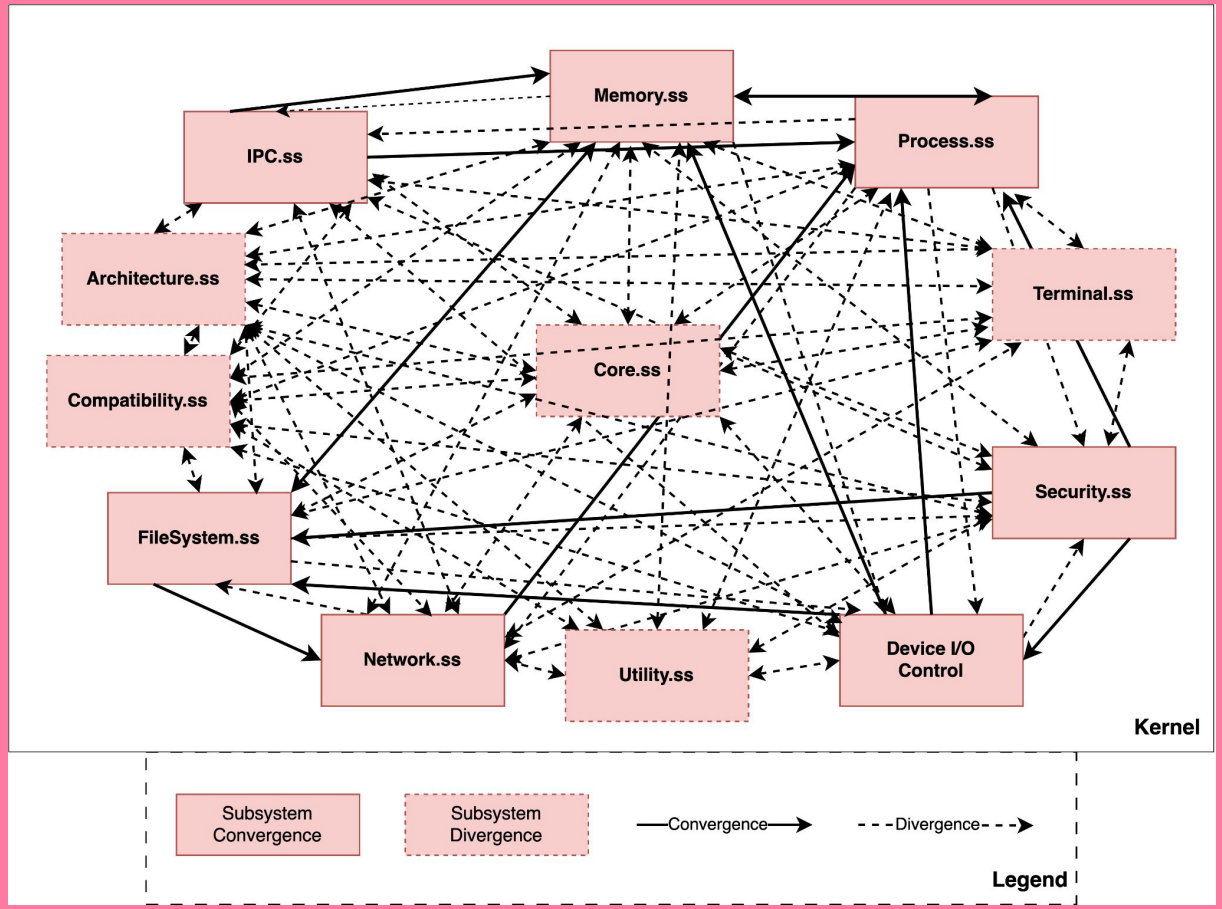
→ Convergences & Divergences

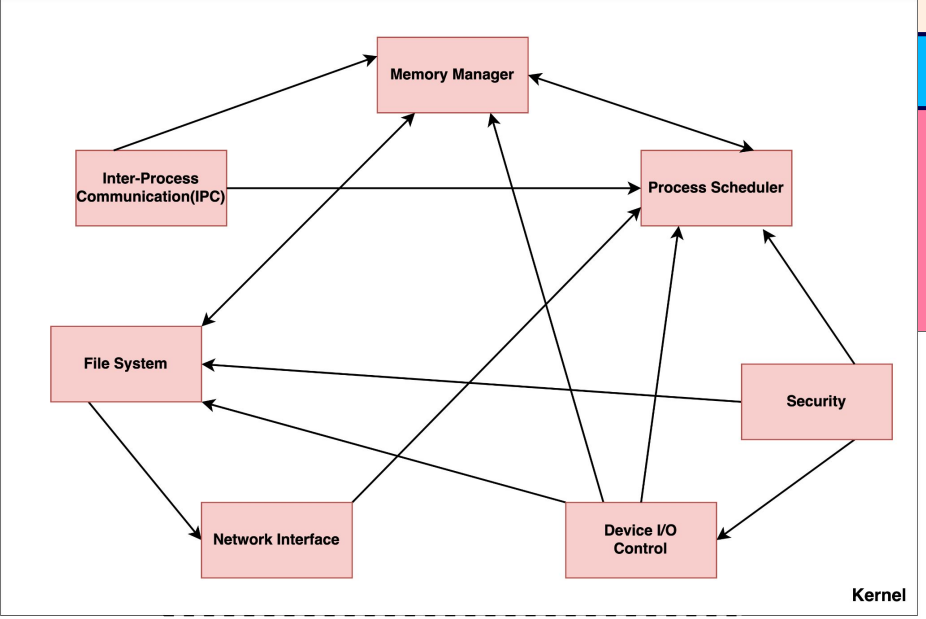


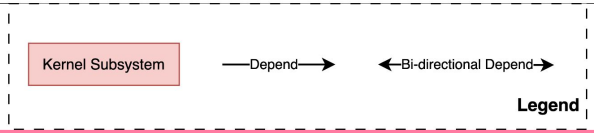
CONCEPTUAL VIEW



REFLEXION DIAGRAM

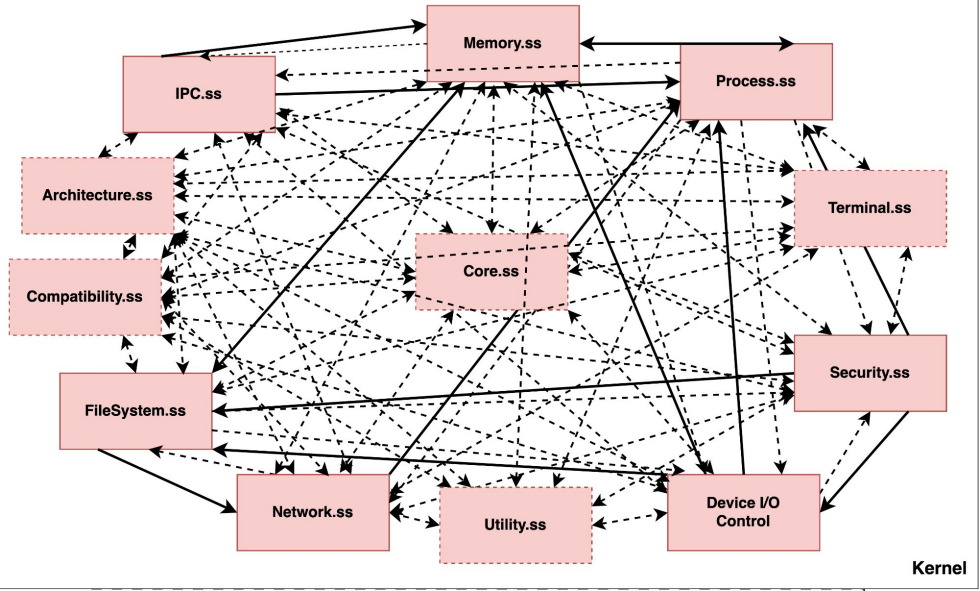


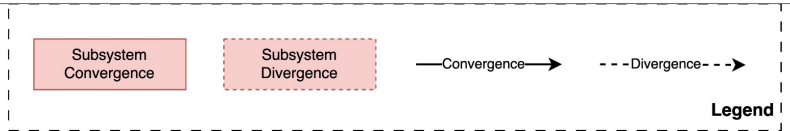




- Use of different mechanisms and bypass interfaces: Expediency and Efficiency
- Low-level implementation difficulties: Deviation from conceptual relationship

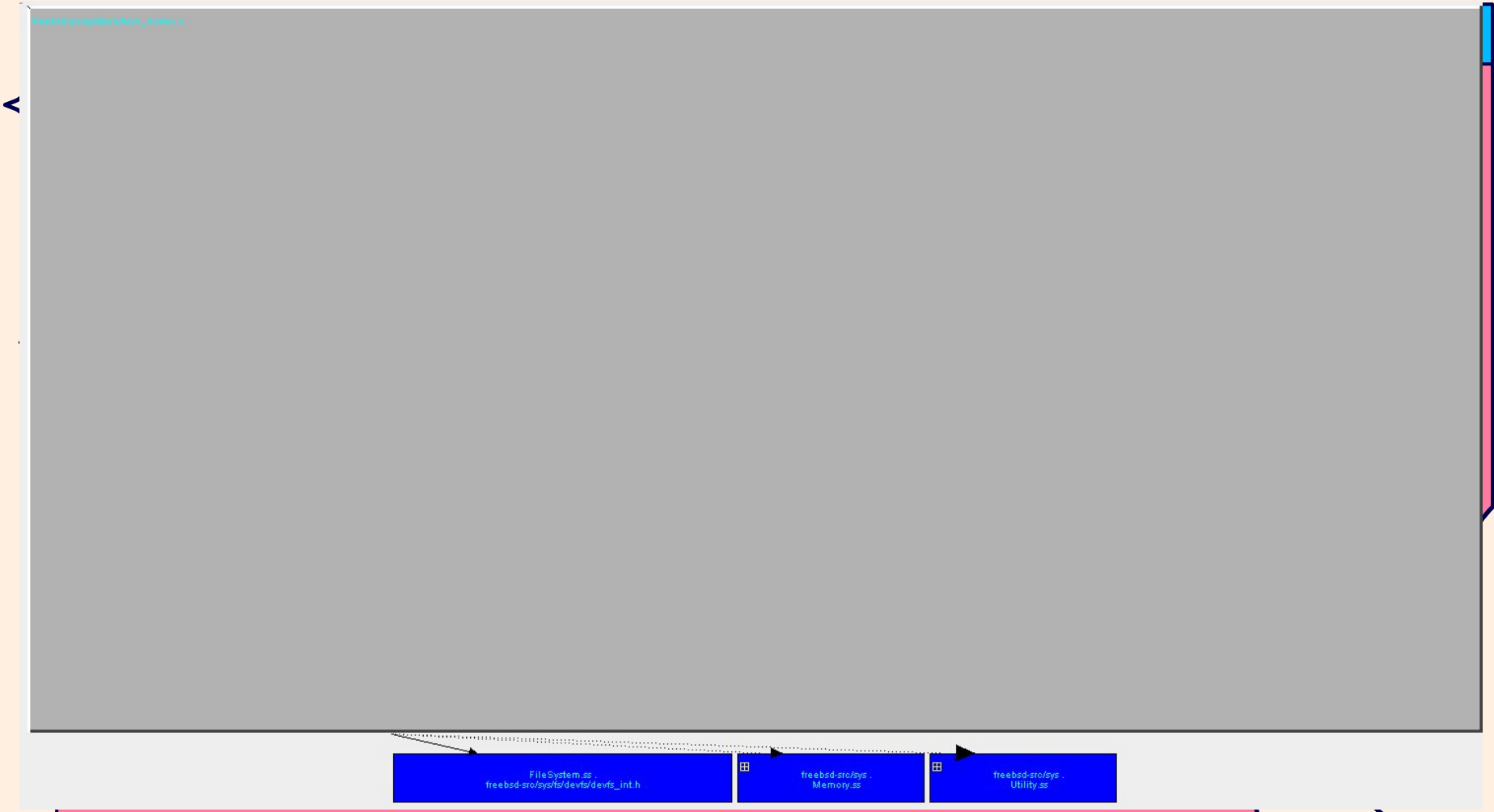
- Always expect more dependencies in concrete architecture
- More functionalities

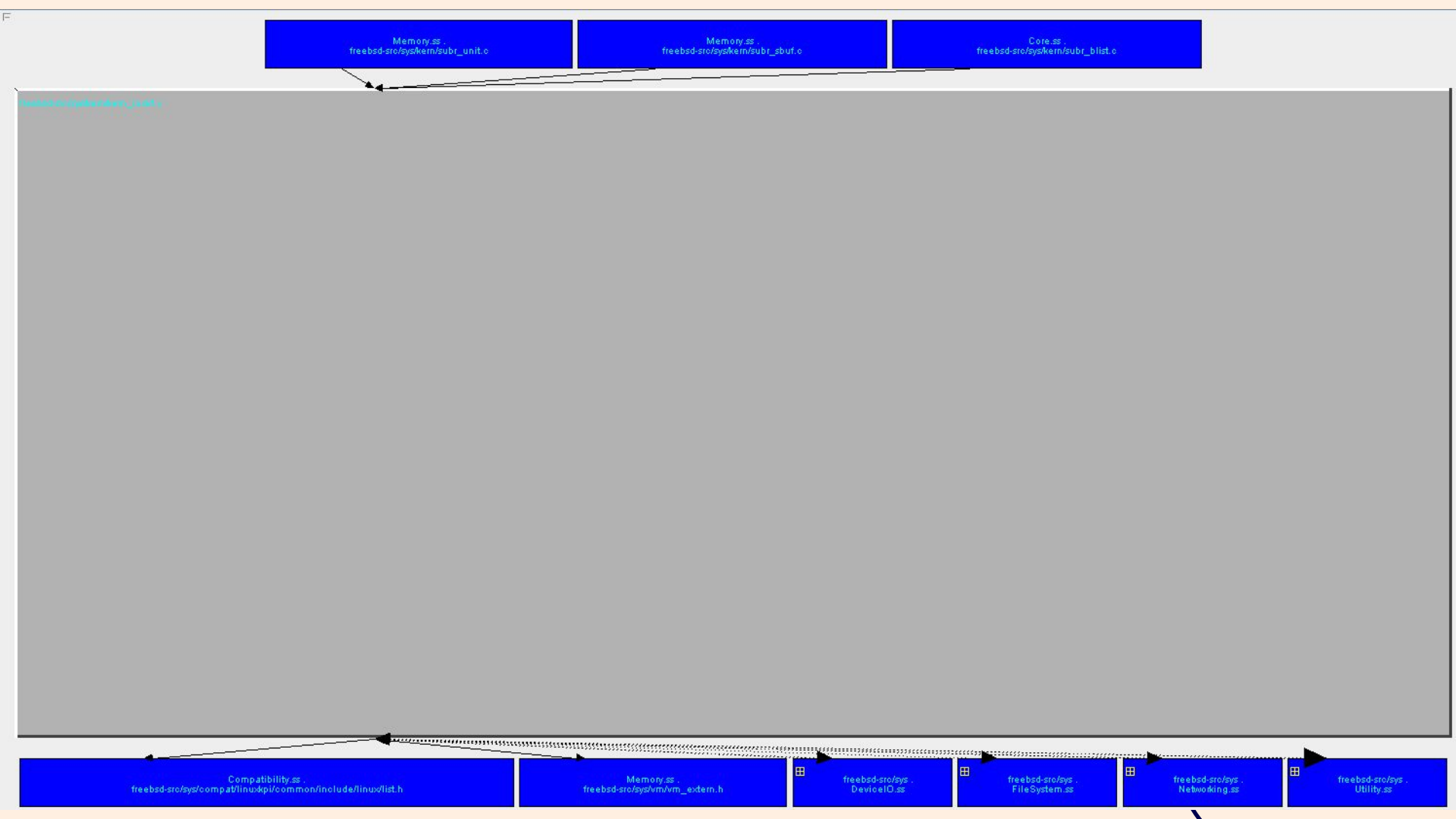




Concurrency

- In order to make sure that errors don't occur when multiple threads are using the same memory area, or different users are accessing the same file.
- The *mutex* and the *lockf* can be used for this situation.
- *Mutex* can arrange CPU to process the thread without conflict and synchronization
- *lockf* can ensure that when executing a command while holding a file lock


FileSystem.ss .
freebsd-src/sys/fs/devfs/devfs_int.h
freebsd-src/sys .
Memory.ss
freebsd-src/sys .
Utility.ss


Memory.ss .
freebsd-src/sys/kern/subr_unit.c
Memory.ss .
freebsd-src/sys/kern/subr_sbuf.c
Core.ss .
freebsd-src/sys/kern/subr_blist.c
Compatibility.ss .
freebsd-src/sys/compat/linuxkpi/common/include/linux/list.h
Memory.ss .
freebsd-src/sys/vm/vm_extern.h
freebsd-src/sys .
DeviceIO.ss
freebsd-src/sys .
FileSystem.ss
freebsd-src/sys .
Networking.ss
freebsd-src/sys .
Utility.ss

LSEdit Demo

Lessons Learned

- Determining appropriate hierarchical decomposition is an **iterative** process with manual human involvement.
- Concrete understanding of implementation of FreeBSD components and how they interact with each other.
- Concrete implementation almost always uses a different mechanism from conceptual data/control flow
- Understandings from concrete should be used to refine conceptual
- Using the *Software Reflexion Framework* to identify convergence, divergence and absences
- Familiarity with concrete architectural styles focusing on implementation
- Gaining experience in communication, documentation, and teamwork

Conclusion

- Architecture of the Memory Subsystem and top level subsystems
- Derivation Process
- Reflexion Model
- LSEdit

References

1. https://cgit.freebsd.org/src/tree
2. https://docs.freebsd.org/en/books/arch-handbook/book/
3. *Marshall Kirk McKusick, George V. Neville-Neil, Robert N.M. Watson (2015)* The Design and Implementation of the FreeBSD® Operating System, 2 e.d.
4. *William Joy, Robert Fabry, Samuel Leffler, M. Kirk McKusick, Michael Karels,* Berkeley Software Architecture Manual 4.4BSD Edition https://docs.freebsd.org/44doc/psd/05.sysman/paper.html
5. *David Garlan, Mary Shaw* (1994) An Introduction to Software Architecture