FreeBSD Conceptual Architecture

pink fluffy unicorns

EECS 4314
Advanced Software Engineering

February 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 **Architecture**

Reference OS Architecture &
Top-level Conceptual

02 **Subsystems**

The functionality of the
top level subsystems

03 **Interactions**

Between subsystems,
Use-case diagrams,
Dependency diagram

04 **Concurrency**

Multi-tasking &
Multi-threading

05 **Design Patterns**

Few observed patterns in
the kernel

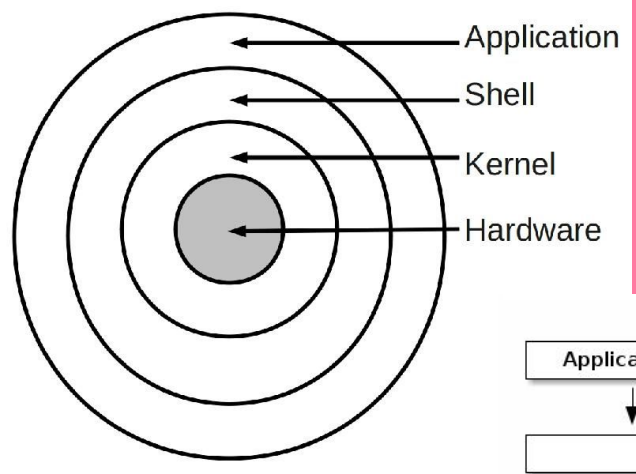
06 **External Interfaces**

Introduction and Overview

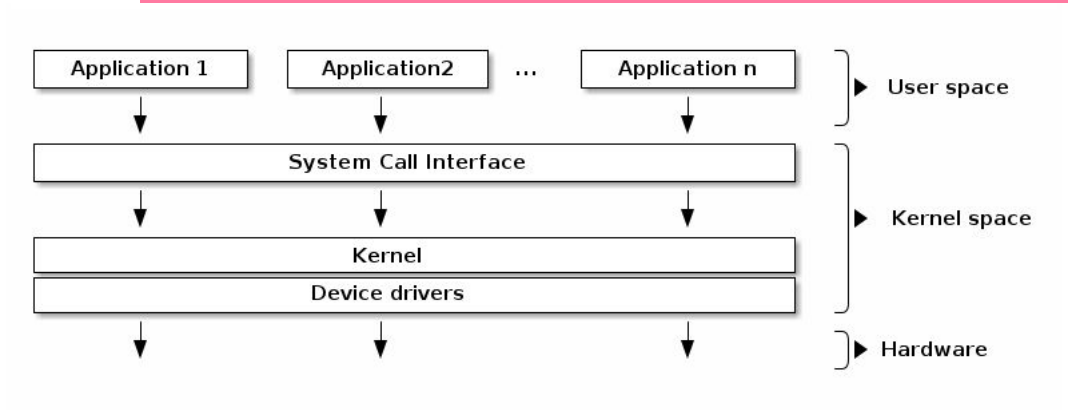
- FreeBSD is a free, open source Unix-like OS
- Widely used for servers, desktops, embedded systems, firewalls, network appliances
- Focuses on stability & security
- Strong software management system
- Advanced features: multi-architecture support, virtualization, jail mechanism.

FreeBSD

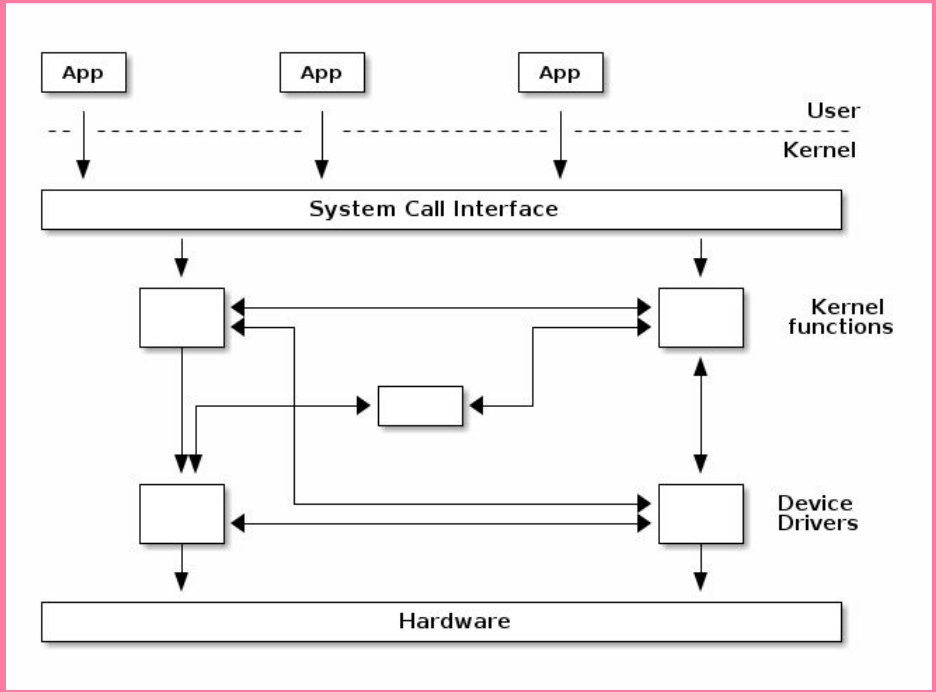
Reference Architecture for an OS



Layered Architecture

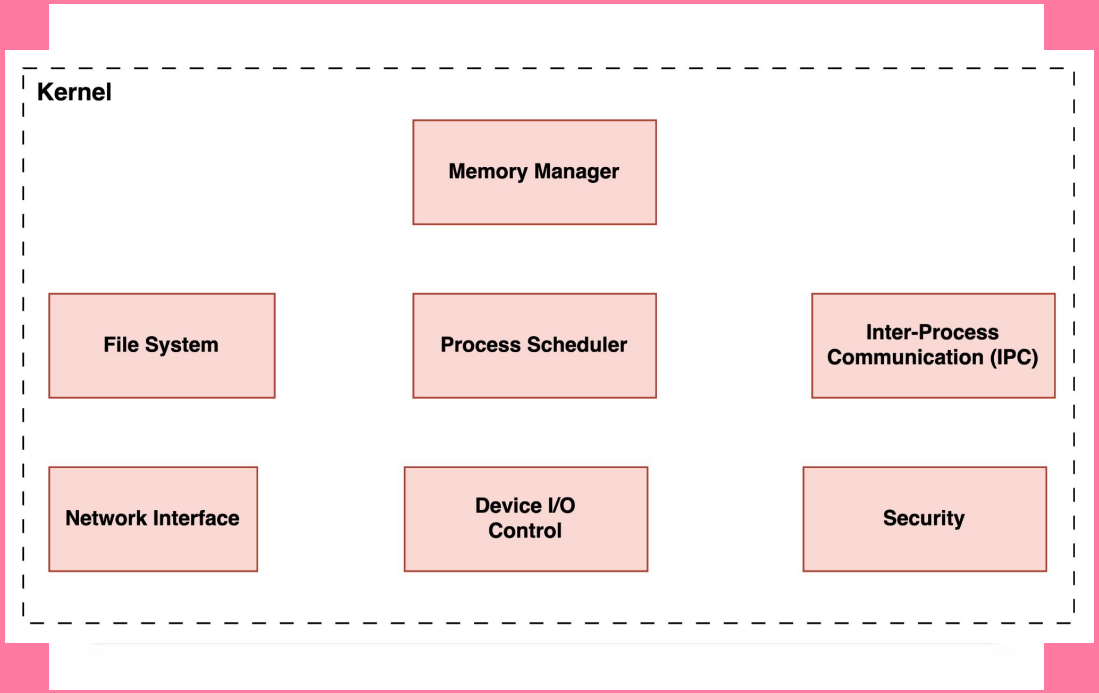


Reference Architecture for an OS



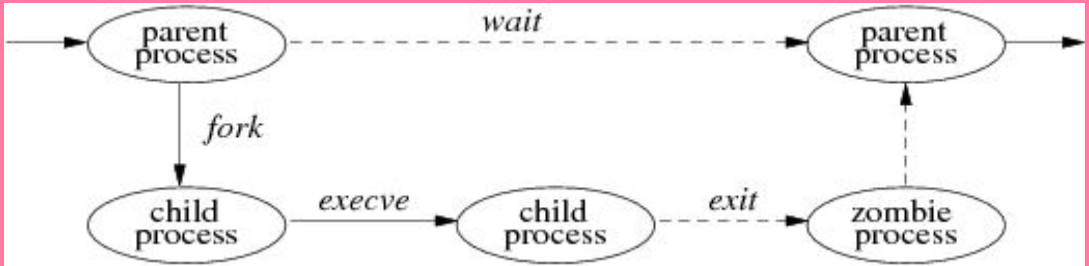
Monolithic Kernel

Overall FreeBSD Structure



Layered Structure With Monolithic Kernel

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

File Management

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

- Files are managed in a tree structure

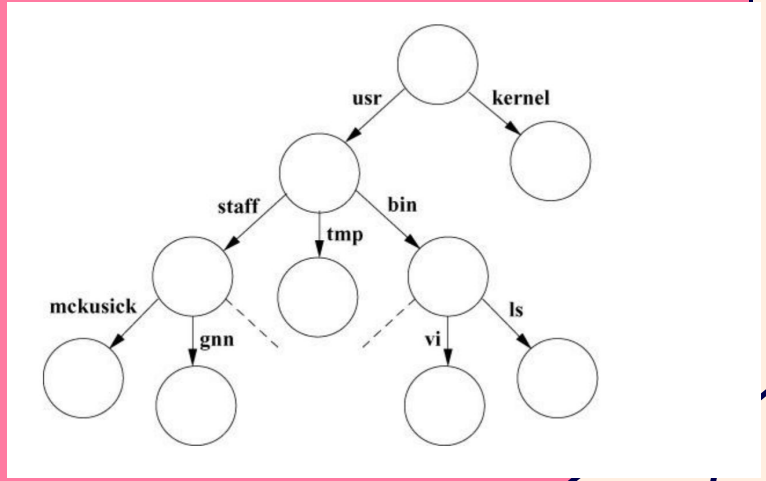
- Also provides a graphical method of file management

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.

I/O Device Management

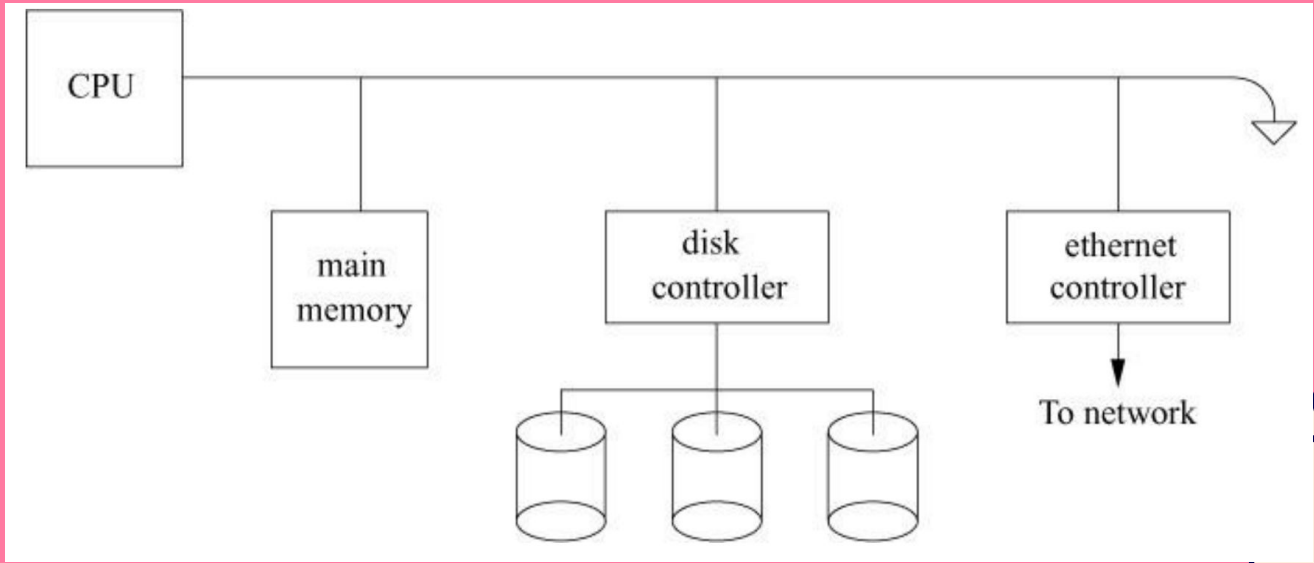
- Kernel acts as an intermediary between the hardware and the operating system
- Responsible for providing the abstract view of the hardware to the operating system and handling all I/O requests.
- FreeBSD provides more advanced I/O device management systems

  -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

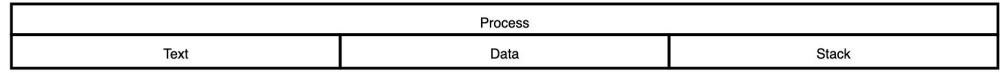
Memory Management

FreeBSD uses the page table for handling physical to virtual memory spaces

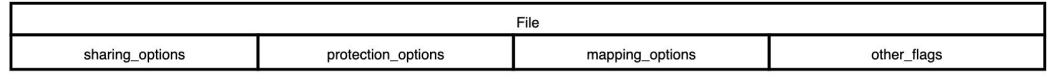


Memory Management

Process stored in memory is divided into three parts

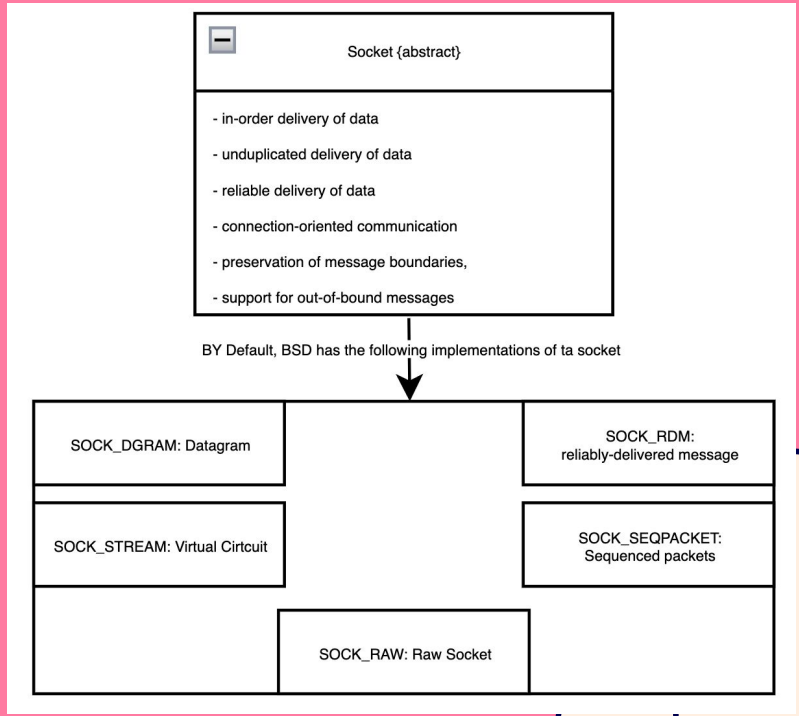


File stored in memory



Interprocess Communications

- **Communication Domains**
  - AF_UNIX: Used for internal communications
  - AF_INET: Used for communications to and from the internet
  - AF_NS: Used for the Xerox Network Systems protocols
- **Sockets**
  - Abstract data-type used for communicating within the domains
- **Protocols**
  - Sockets must respect the protocols used by the domain
  - For Example, the internet uses TCP and UDP protocols.



Security

- **Process Credentials**

  variables for identification. UNIX user identifiers (UID), group identifiers (GID)

- **Privilege Model**

  modify system-level data.

- **Capability Model**

  runs those untrustworthy code to be run in a safe way

- **Mandatory Access Control**

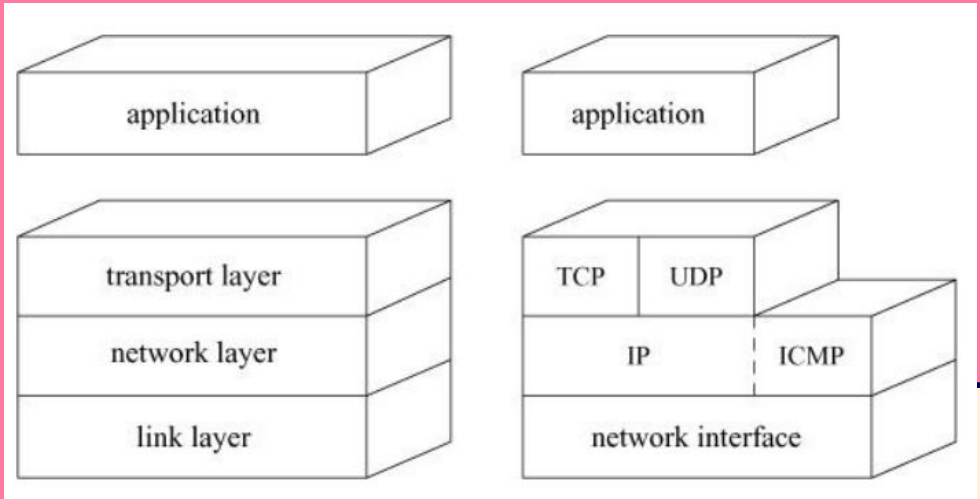
  a set of security policies

- **Cryptography**

  Keep user's information safe

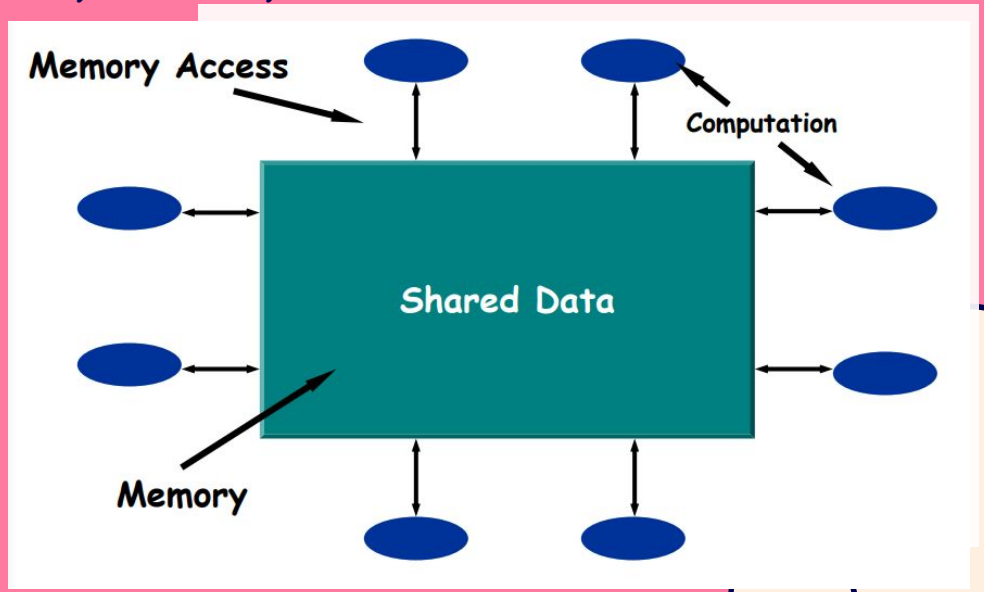
Network Management

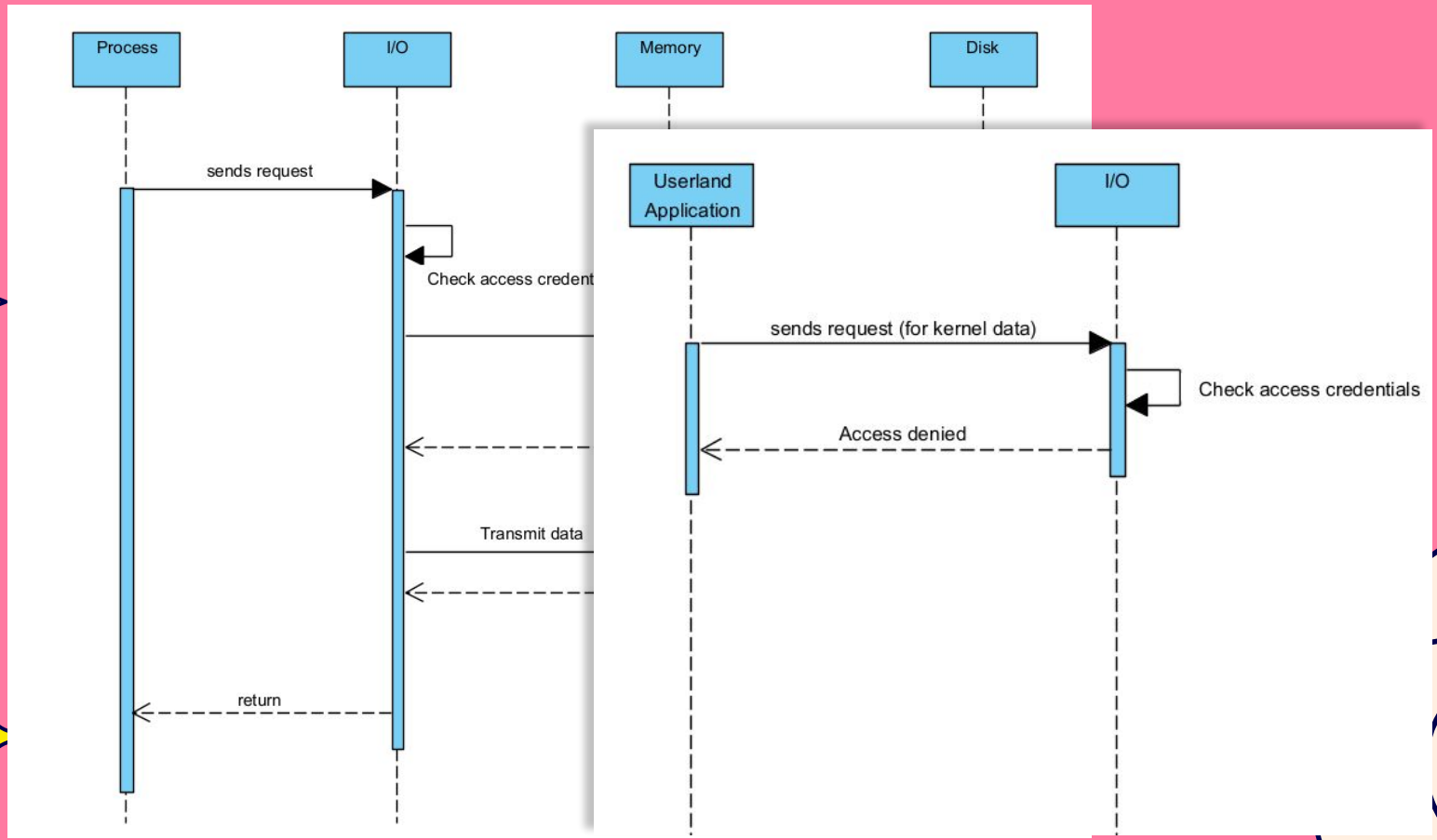
- Use TCP to manage the network
- There are at least two layers involved in a host-to-host connection
- Transport layer
- Network layer
- Link layer



Interactions

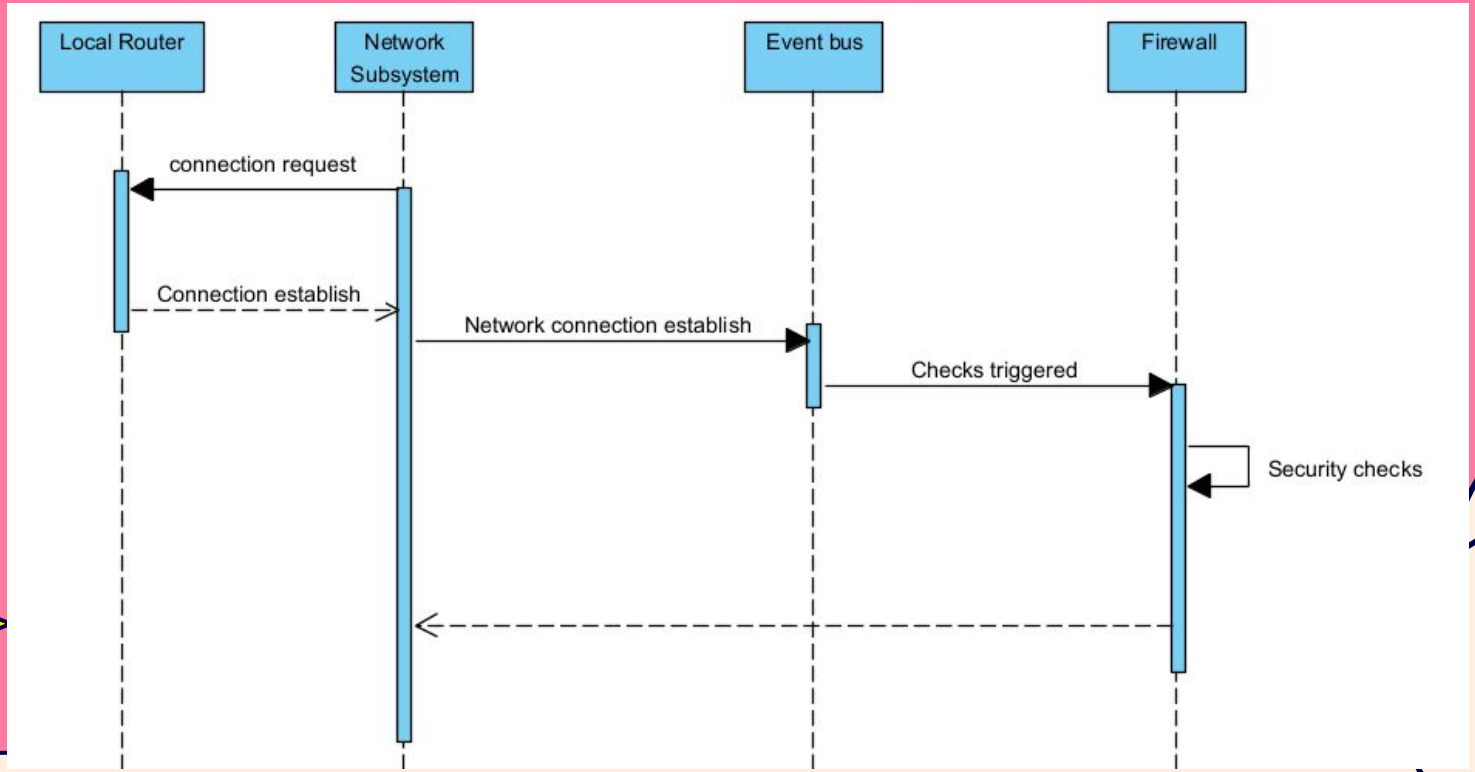
- Layered & Monolithic architecture
  - Efficiency & independency for subsystems
- Memory & I/O & Disk
  - Repository & Pipes & Layers
- Security
  - Event-based



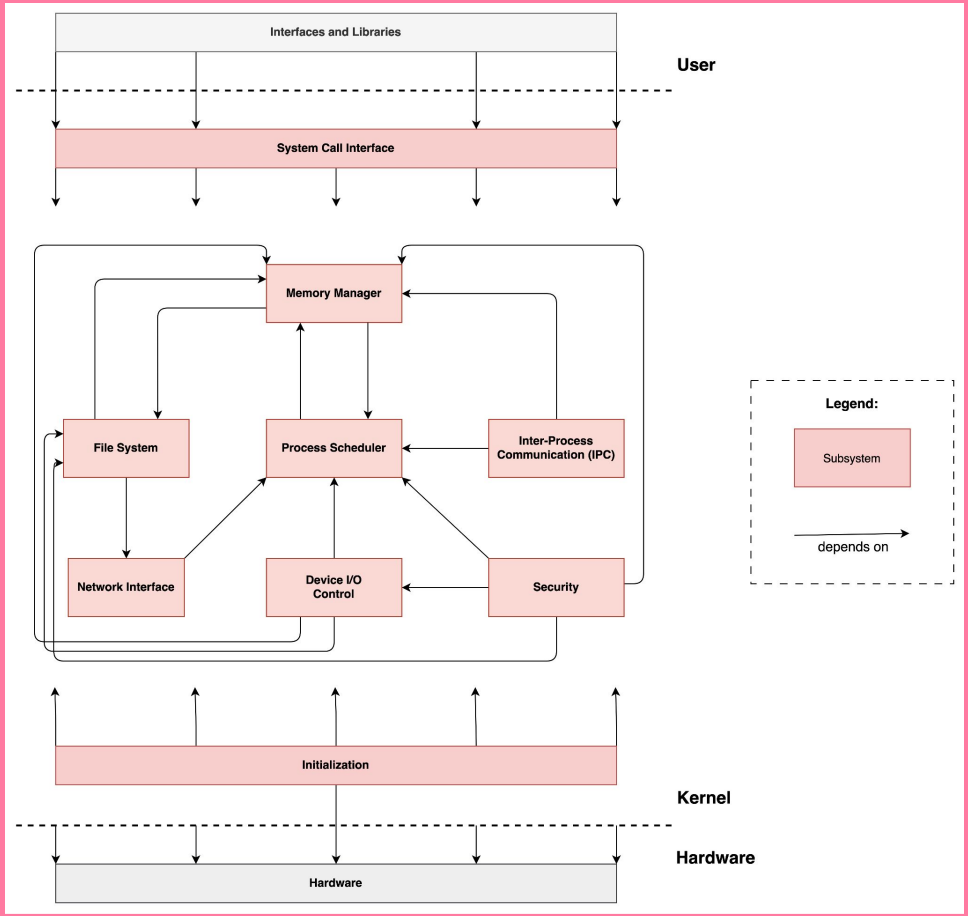

Process
I/O
Memory
Disk
sends request
Check access credent
Transmit data
return
Userland Application
I/O
sends request (for kernel data)
Check access credentials
Access denied

Use Cases

- **Security trigger – Firewall**

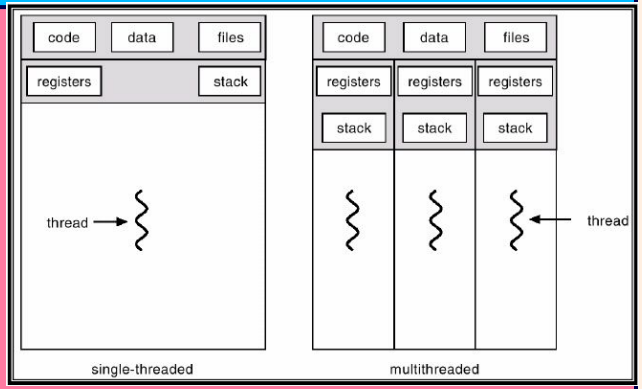


Dependency Diagram



Concurrency

Processes — Threads



- *Multi-tasking*: allowing multiple processes to run concurrently on a single CPU
- *Multi-threading*: within each process, allowing multiple threads of execution to run concurrently within a single process

Concurrency

Global Control Flow via

System Calls → **Interrupts** → **Context Switching** →

Ensures the state of a process can be restored and execution can be resumed

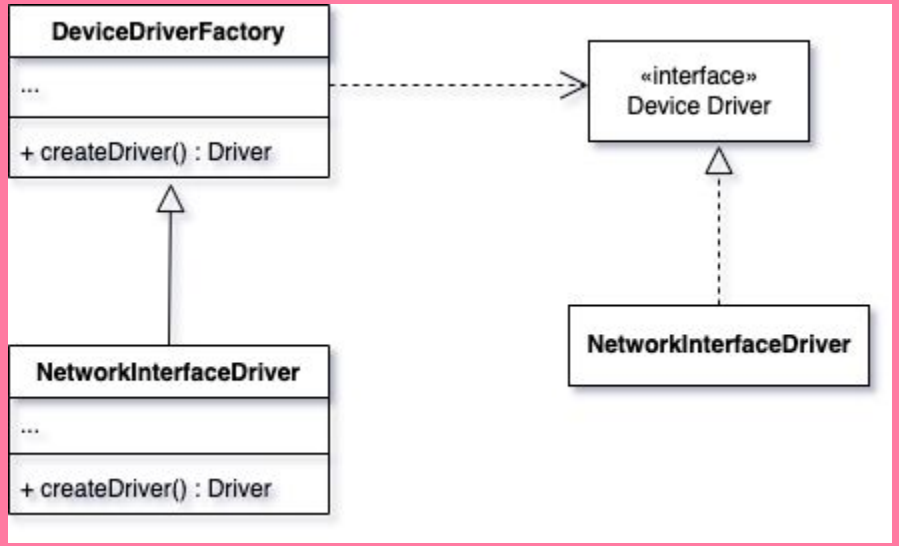
WhatsApp:
5M+ concurrent TCP connections on a single BSD server

← Kernel is extremely concurrent and supports 128+ threads

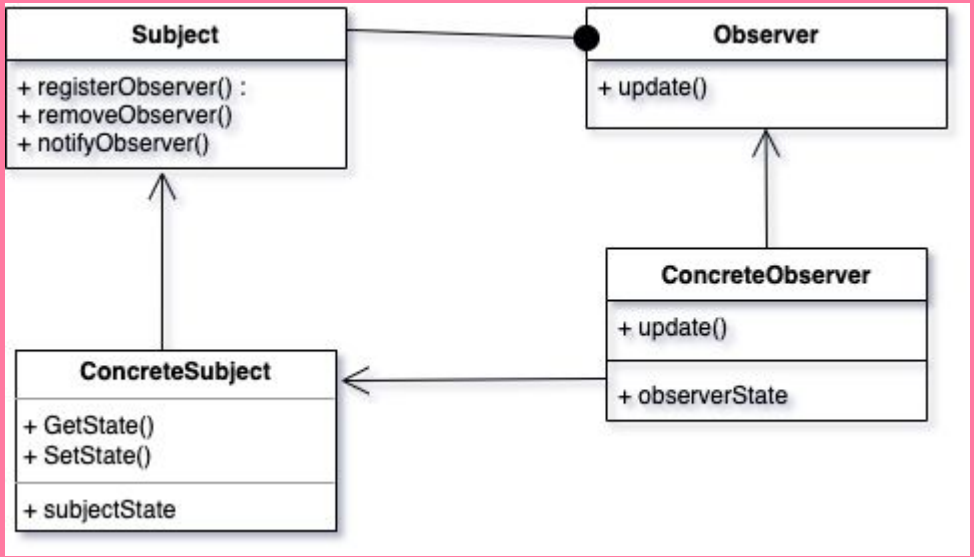
Design Patterns

- Factory Pattern
- Observer Pattern
- Abstract Factory Pattern
- Singleton Pattern

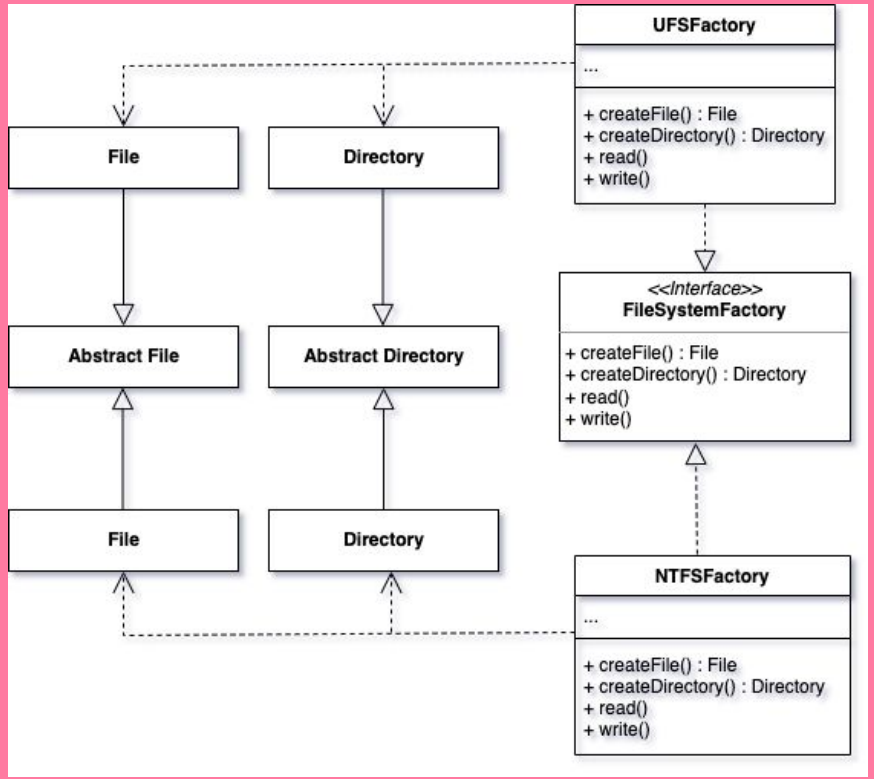
Factory Pattern



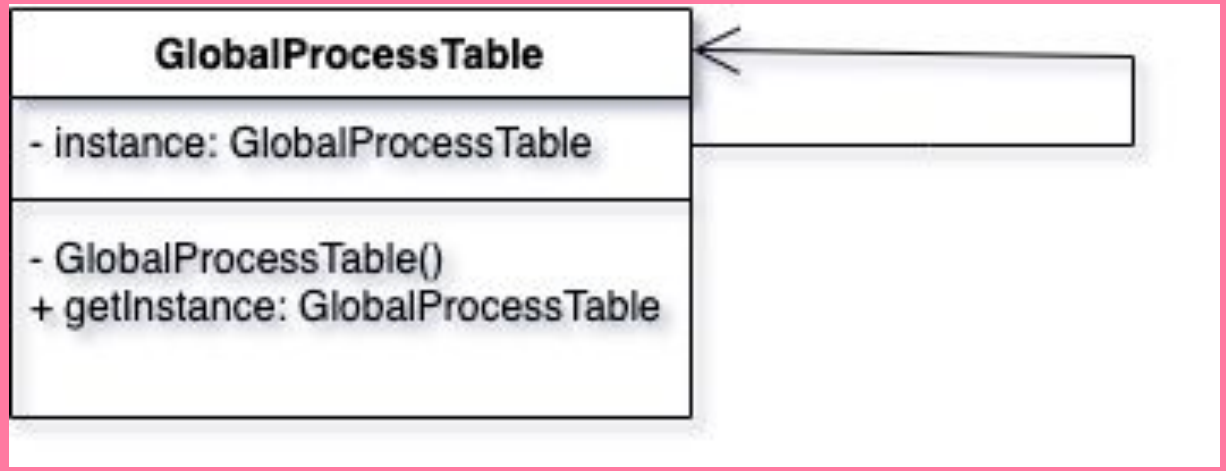
Observer Pattern



Abstract Factory Pattern



Singleton Pattern



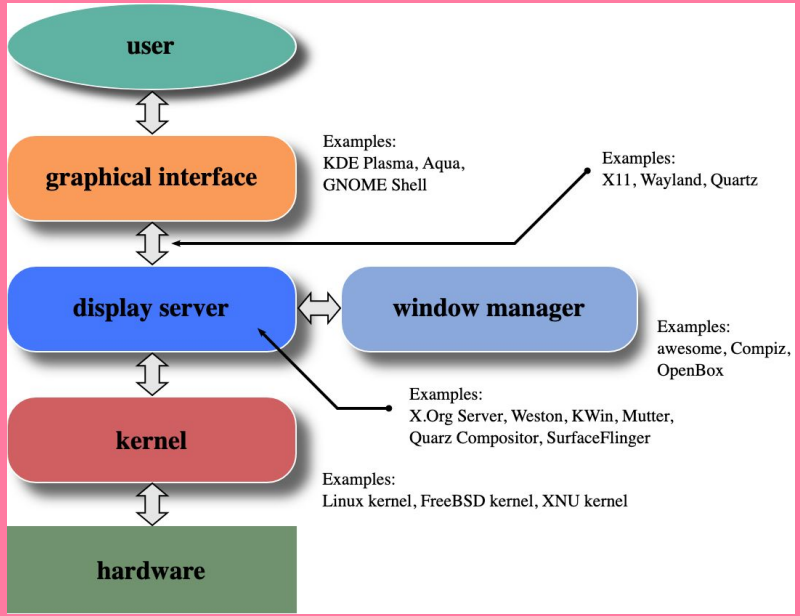
External Interfaces

- FreeBSD has external interfaces which allows the operating system to communicate with other entities, such as users, devices, and other systems.
- This could range from controlling other devices, to receiving data from other systems.
- Without a wide range of external interfaces, the operating system would not be useful as we would not be able to control many aspects of our machine, if any at all.

Graphical User Interface

- Often referred to as GUI.
- Offers users the ability to visually control and issue commands to the system.
- Arguably the easiest form of interaction for the user as there is a very low learning curve.
- To create a GUI in a Unix-based system like FreeBSD, we need several components.

Graphical User Interface

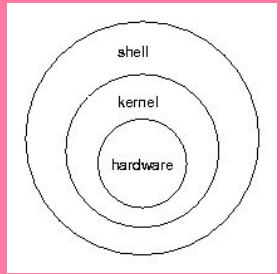


Command Line Interface

- Often referred to as CLI.
- Offers users the ability to issue commands to the system through keyboard inputs.
- A bit on the advanced side as there is a certain learning curve.
- Primary mode of interaction for FreeBSD.

Command Line Interface

- The command line interface is known as a shell.
- Shells have built in functions which help users do routine tasks, and manage their system
- Users can invoke commands through this shell.
- The shell works by making system calls, which is provided through an API exposed by the kernel. This API is the external interface which the CLI calls upon to execute actions given by the user.



Derivation Process

1. Documentation analysis
2. Identification of system's components and subsystems
3. Identification of dependencies
4. Diagram creation

Lessons Learned

1. Understanding of FreeBSD components and how they interact with each other.
2. Understanding of operating systems
3. Best practices for designing and building systems
4. Familiarity with architectural styles
5. Gaining experience in communication, documentation, and teamwork

References

1. https://docs.freebsd.org/en/books/arch-handbook/book/
2. *Marshall Kirk McKusick, George V. Neville-Neil, Robert N.M. Watson (2015)* The Design and Implementation of the FreeBSD® Operating System, 2 e.d.
3. *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
4. *David Garlan, Mary Shaw* (1994) An Introduction to Software Architecture