Digital Assignment 1

Operating Systems

22BCE0682
Siddhant Bhagat

Q1) Explore the basics and architecture of XV6.

A→ Primary used for education purposes, XV6 is a simple Unix-like operating system, allowing students to understand operating system concepts through its compact and well structured codebase. Based on Unix Version 6, XV6 provides a minimal but fully functional system that emulates key Unix-like functionalities in a manageable format.

XV6 is a lightweight teaching OS developed by MIT to help students fundamental operating system principles. It implements key Unix concepts, such as processes, file systems, and memory management. But for simplicity, XV6 includes only essential features, making it accessible and allowing students to learn OS concepts by studying the source code.

* System Architecture

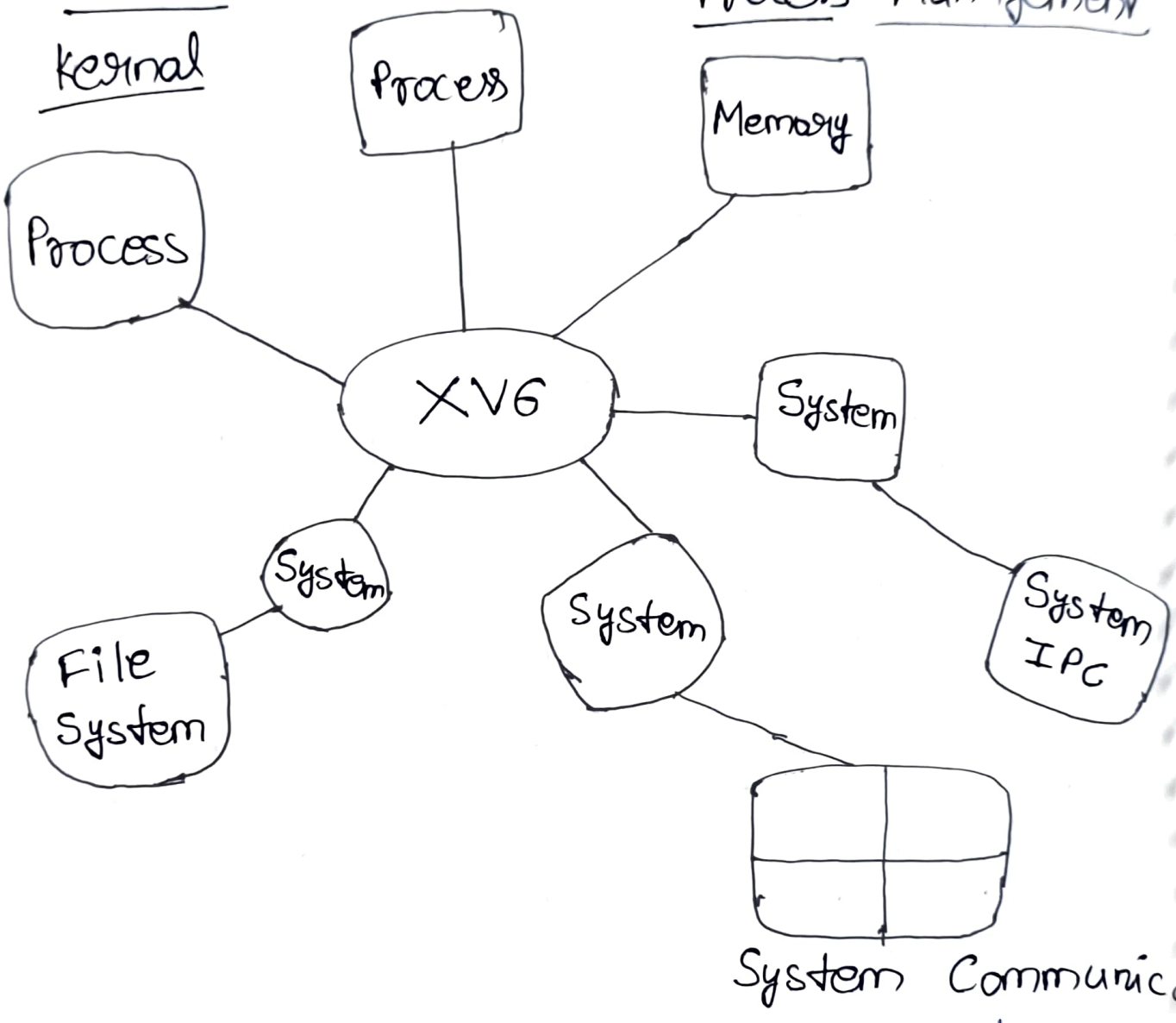
XV6 is designed to run on x86 hardware and operates within a single-CPU environment. It supports core OS functionalities like process management, memory management, file systems, and basic system calls.

22BCE0582
Siddhant Bhagat

The architecture includes components similar to more complex Unix-based systems, but it excludes advanced functionalities for simplicity.

Kernal structure

Process Management



XV6 follows a modular architecture

* Process Management: Manages process creation, switching and termination. Each process has a unique PID and can perform system calls to interact with the kernal.

22BCE0682
Siddhant Bhagat

File Systems:

XV6 implements a Unix-like file system with a hierarchical directory structure and simple block-based storage. Files are organised as inodes, and the file system uses a log to track changes making it somewhat resilient to crashes.

Memory Management

Supports virtual memory using paging, providing each process its own address space. XV6 uses a single-level page for virtual to physical address translation.

System Calls

Kernal provides a minimal set of system calls for basic operations [eg: read, write, fork, exit]. These calls allow user-level programs to interact with the hardware through the kernal.

Process Lifecycle and Scheduling

Process in XV6 go through a lifecycle similar to traditional Unix systems:

* Creation: Processes are created using fork, which clones the calling process.

22BCE0682
Siddhant Bhagat

* Execution:

Processes execute user-space code until they yield, sleep, or complete execution.

* Scheduling:

XV6 uses a round-robin scheduling algorithm. This simplicity is essential for understand fundamental scheduling techniques without the complexity of multi-level for priority based scheduling

* Exit:

Processes terminate with exit, allowing the kernel to reclaim resources.

File System Architecture

* XV6 file system is designed is represented by an inode, which stores metadata and pointers to data blocks.

* Directories:

file system is tree-structured with directories, containing file & subdirectory references.

* Logging:

XV6 includes a simple journaling mechanism for log-based transactions, ensuring some resilience against unexped crashes.

22BCE0682
Siddhant Bhagat

* Memory Management

XV6 provides a simple paging mechanism for virtual memory. Each process has its own page table, translating virtual addresses to physical ones. Memory allocation in XV6 is done using a basic allocator that merges free memory and assigns it to processes. The System's design avoids complexities like multi-paging or Segmentation, making it easier to understand the fundamentals of memory managemat.

Inter-process Communication.

IPC in XV6 is limited but includes basic mechanisms for process communication, such as pipes. Pipes enable data exchange between processes, with Kernal managing buffers for read and write operations. This functionality introduces students to basic of inter-process communication in an OS without the added complexity of sockets or message queues.

22BCE0682
Siddhant Bhagat

System Calls

XV6 offers a small, well-defined set of system calls for interacting with kernel.
These calls handle essential operations such as:

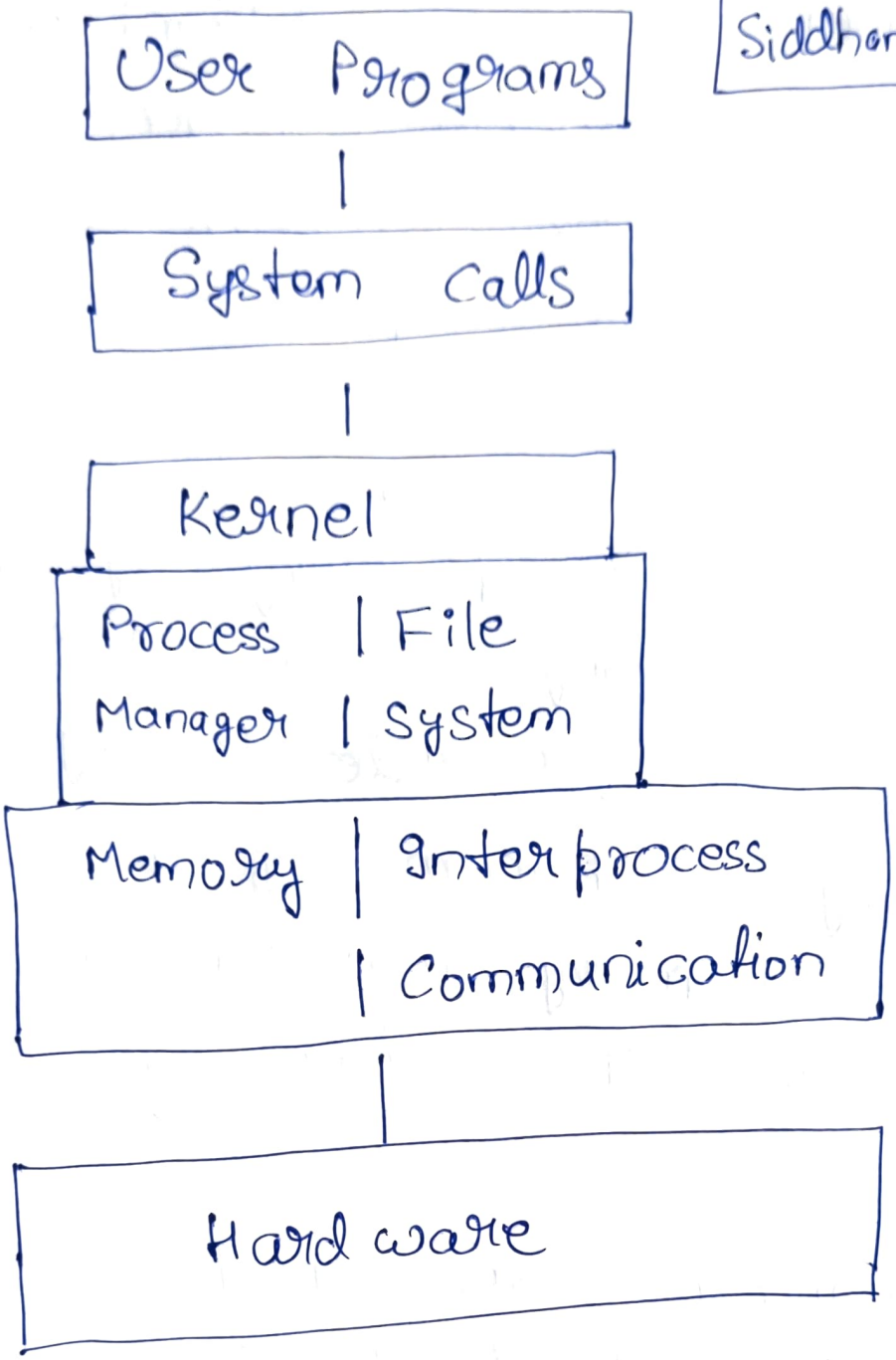
* File manipulation: open, read, write and close.
* Process control: fork, exe, exit & wait.
* Memory Management: sbrk for adjusting the heap size.

System calls in XV6 are directly mapped to assembly code, which allows it to interface closely with hardware.

Diagram

XV6 architecture is a high-level diagram illustrating the architecture of XV6, showing the primary components and their interactions:

22BCE0682
Siddhant Bhagat



The diagram outlines how user programs interact with the hardware through system calls and kernel modules such as process management, file systems, memory management and IPC.

Conclusions

22BCE0682
Siddhant Bhagat

XV6 offers a valuable experience by simplifying complex OS features into a manageable and comprehensive System. Its Unix-like design provides a structured and accesible frame work to study core operating system principles, making it an ideal learning tool for understanding the underlying mechanisms of process control., memory management, file systems, and IPC within operating systems.

Q2) Summarize the concepts behind process management in XV6.

A→ Process management in XV6 centers around handling process creation, execution and termination with a straight forward architecture that illustrates fundamental OS principles.

Here's a deeper look at each concept:

* Process Lifestyle

22BCE0682
Siddhant Bhagat

* Creation with fork:

Processes in XV6 are created using the fork system call, which duplicates the parent process. The new process or "child" is assigned a unique Process ID [PID] and inherits the parent memory space, file descriptors, and other states.

* States:

A process in XV6 can be in one of several states: unused, embryo (while its being initialized), sleeping, runnable, running or zombie. The zombie state indicates that the process has terminated but its resources haven't been fully reclaimed by the system.

* Termination with exit

When a process finishes execution, it calls exit, which releases memory, closes files, and places it in a zombie state. The parent process can call wait to collect the exit status and reclaim the remaining resources.

22BCE0682
Siddhant Bhagat

* Scheduling

Round Robin Algorithm:

XV6 uses a basic round-robin scheduling algorithm, where each process in the runnable state receives a fixed time slice of CPU time. This method cycles through processes without priority or multitier scheduling, offering a fair and straightforward approach.

Runnable Queues:

Processes ready to execute are placed in a runnable queue. scheduler traverses thin queues, running each process in turn, demonstrates concepts of fair access and time-sharing.

* Context Switching

Saving and Restoring Registers:

In context switching, kernal saves the CPU registers, stack pointer,

and program counter of current process in its Process control Block [PCB]. when switching to a new process, kernal loads that process's saved state, allowing it to resume from where it left off.

* System Calls and Process Control

✓ System Calls: fork, exec, exit, wait, kill, sleep.

✓ waiting and Reaping Processes: wait call

✓ Signals and kill: simple kill

IPC

Pipes: XV6 provides simple implementation of pipes for communication between processes. A pipe acts as a buffer that allows data enabling unidirectional communication.
This feature, while limited, introduces students to the basic of inter-process data exchange.