Siddhant Bhagat

22BCE0682

DOMS Page No.

Date / /

# Digital Assessment

## B1+TB1 Operating Systems

Q1) Explore the basics and architecture of XV6.

A→ Focusing on modularity and simplicity XV6 operating system is a modern, educational reimplementation of Unix Version 6 designed for teaching purposes. It was developed at MIT for educational use, it provides simplified but functionally accurate representation of Unix. Its purpose is to give a clearer view of how an operating system is constructed without complexity of modern OS implementations.

✓ Architecture →

→ Kernal and User space -

* XV6 seperates user space from kernal space. Kernal space has complete access to hardware sources; which was programs execute in restricted environment.

* XV6 enforces this separation to protect the system from faults in user programs.

* It uses process concepts such as creation, scheduling, and termination.

→ Memory Management -

Siddhant Bhagat
22BCE0682

- * XV6 uses a simple virtual memory system with a linear page table mapping.
- * It has straightforward memory allocator based on bitmap technique, to allocate & free memory blocks efficiently.

→ Process and Thread Management

- * XV6 has a basic process management system with a simple round-robin scheduler. Each process has its own memory space and set of resources.
- * Although it does not support threads explicitly, XV6 allows for process creation (using fork) & management, which mimics multithreaded behavior more advanced Systems.

→ File Systems

- * Inode based [data structure] filesystem, similar to unix V6.
- * Supports basic file operation, such as reading writing, and file metadata management, and organises files in a hierarchial directly structure.

→ Interrupt Handling & System Calls

- * XV6 handles hardwork and software interrupts through an interrupt vector, enabling it to process asynchronous ~~excepts~~ events & perform operations requested by user programs.

Siddhant Bhagat

22BCE0682

* System calls are primary mechanism through which user-space programs request services from the kernal. XV6 has a basic set of system calls [open, read, write etc.]

*

→ Scheduling

* XV6 employs a simple round-robin schedules for managing process. There's no priority based or multi-level scheduling, which keeps the code straightforward & easy to understand.

* Data Structures used are

✓ Process table → To keep track of all process in the system.

✓ Inode table → Table keeps track of all active inodes, helping file system manage open files and directory entries.

✓ File Table → Stores open file descriptors, associating them with corresponding inodes & offset values.

✓ Page Table → XV6s memory is mapped using two-level paging system, with each process having its own page directory & page tables.

Siddhant Bhagat

22BCE0682

Q2) Summarize concepts behind process management in XV6.

A→ XV6 implements process concepts such as creation, scheduling, and termination. Each process is represented by a "proc" structure, with an integer ID, memory information and a state (eg: running, sleeping. It closely mimics the unix process model and is centered around creating, scheduling & managing process.

* Concepts of Process management in XV6.

* Process structure

Each process in XV6 is represented by a 'proc'.

* Fork and Exec:

fork() system call creates a new process, duplicating the parent process's memory and returning different values to parent and child to distinguish between them, exec() replaces the process memory with a new program allowing a process to execute a different program within the same address space.

* Scheduling: XV6 uses a basic round robin schedules that cycle

through each runnable process, giving each process a fair amount of CPU time. . Scheduler switches processes when current process either goes to sleep or is interrupted.

* Sleep & wakeup:
XV6 process can use sleep() & wakeup for inter-process communication & synchronisation.

* Process States

XV6 processes have states such as runnable, running, sleeping & zombie which help manage life cycle of each process creation to termination.

* Process Termination: exit() system call terminates a process, cleaning up allocated resources & making it as a zombie until the parent process calls wait() to remove it from system.