Mini System Monitor with File Access Logger in C using System Calls

Team - 7 Batch - C

G Prajwal Priyadarshan – cb.sc.u4aie24214

Kishore B – cb.sc.u4iae24227

Kabilan K – cb.sc.u4aie24224

Rahul L S - cb.sc.u4iae24248

Introduction

- The main goal of this project is to create a small tool, like a **mini task manager**, that can help us monitor what is happening inside the system and also control some activities.

- Normally, tools like **Task Manager in Windows** or **System Monitor in Linux** are used to check which programs are running, how much CPU they are using, and to stop programs that are not responding. In our project, we aim to build a similar system but in a simpler way.

- We are going to write this in **C language**, and instead of using ready-made libraries, we will use **system calls**.

- A system call is basically a direct way for a program to ask the operating system to do something, like read a file, kill a process, or get process details.

Why do we choose it

- This project has been chosen because system monitoring plays a vital role in the efficient functioning of an operating system.
- Users often need to identify running processes, analyze their CPU usage, and terminate processes that are consuming excessive resources.
- In addition, monitoring file access times and detecting permission changes are important for ensuring system security and reliability.
- By implementing this project, we aim to gain a deeper understanding of how the operating system manages **processes, files, and system resources**.
- Furthermore, developing the tool in **C using system calls** provides valuable hands-on experience in system-level programming and direct interaction with the operating system kernel, rather than relying on high-level abstractions.

Objectives

❑ Resource Monitoring : To track CPU and memory usage in real-time to optimize system performance and efficiency.

❑ File Access Logging : To monitor and log file access activity to ensure security and compliance for sensitive data.

❑ Process Management : To manage, display and control active processes to prevent system overload and improve resource allocation.

❑ To detect and alert on File Permission Changes

Core Functionalities

List Active Processes

- `opendir()`
- `readdir()`
- `/proc/[pid]/comm`

Show running process names & PIDs

Display CPU Usage per Process

- `fopen(),`
- `/proc/[pid]/stat`
- `/proc/stat`

Show how much CPU time each process uses

Kill a Process

- `kill()`

Terminate a process by PID

Core Functionalities

Log File Access Times

- `stat()`
- `ctime()`

Log last access/modified time of a file

Alert on Permission Change

- `stat()`,

monitor file mode bits
Alert when file permissions change

Monitor Disk Space Usage

- `statvfs()`

Show available vs used disk space

Core Functionalities

Monitor Memory Usage per Process

- `fopen()`
- `/proc/[pid]/status`

Display memory usage (VmRSS, etc.)

Log All Executed Processes

- Read `/proc`, check `/proc/[pid]/exe`

Maintain a log of all running/executed binaries

System Calls Involved

System Call	Kernel Role	Used For
`opendir()` / `readdir()`	Reads directory entries	Listing processes in `/proc`
`fopen()` / `fread()`	Reads `/proc` virtual files	CPU, memory stats
kill()	Sends signals to processes	Terminate process
stat()	Gets file inode metadata	Access time, permissions
ctime()	Converts `time_t`	Readable access time
statvfs()	Retrieves filesystem statistics	Disk usage
readlink()	Resolves symbolic links	Find executed binary by a process.

Applications

- Lightweight **system monitoring tools** for embedded systems or IoT.
- **Security auditing** – track access to sensitive files.
- Extendable to a GUI-based resource manager.

Expected Output

- A terminal-based interactive program with menu options:
- View list of processes with PIDs.
- See per-process CPU usage.
- Enter PID to kill a process.
- Display file access and permission change logs.
- Show available disk space.
- Clear, user-friendly output, updated dynamically or on request.
- Logs for tracking file access and permission changes stored in a local file.

Thank You !