OPERATING SYSTEM PROJECT

Instructor: Zubaidah AlHazza

Students

Alya Alzaabi - 2022005560

Rabaa Alshbib 2022005690

Roodh Alblooshi 2022005691

Lana Zanneh 2022005620

Fatima Farooq 2023006109

executive summery

1- This case study will explore the implementation of zorin linux distribution & tools to manage processes
2- it uses 1-1 threading model, how it affects efficiency
3- zorin os has stable and powerful performance, drawbacks are overhead and resource consumption
4- it manage resources between threads and processes by mutexes and semaphores.
5- It handles deadlocks quick and unstoppable.
6- in memory management it uses paging, swapping and best fit allocation

Introduction

Operating systems are the backbone of modern computing, managing everything from memory to process scheduling. In this case study, we will discuss the subject of Zorin OS - a Linux distribution that is user-friendly and is based on Ubuntu, to find out how the OS really works with its various parts. Our motivation is to bridge theory with real-world implementation—diving into how Zorin OS manages processes, memory, synchronization, file systems, and deadlocks. Using built-in Linux tools and simulations, we aim to reveal how this system works behind the scenes, and what it means for developers, users, and the open-source community.

Process creation and management

- Efficient Process Handling: Zorin OS uses fork() and clone() system calls with a task_struct for process management and employs the Completely Fair Scheduler (CFS) using a red-black tree for fair CPU time distribution.
- Optimized Context Switching: Context switches are finely tuned to balance responsiveness and overhead, improving multitasking performance.
- Threading with NPTL (1:1 Model): Threads in Zorin OS map one user thread to one kernel thread, managed through pthread_create() and clone() with resource-sharing flags.
- Futex-Based Synchronization: Zorin uses fast userspace mutexes (futexes) for low-overhead thread synchronization, imp21roving multithreaded performance.
- Thread Pools & Async I/O: For better scalability in resource-heavy applications, Zorin supports thread pools and event-driven asynchronous I/O.

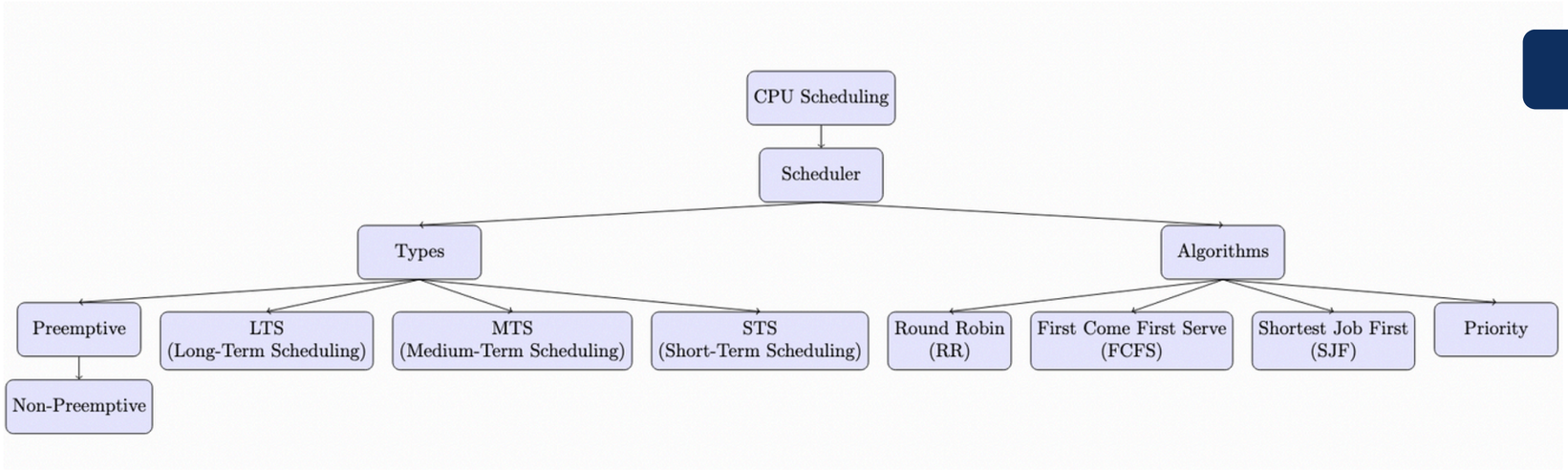
Strengths & Limitations of Zorin OS

1. Strength: Fair & Scalable Scheduling: CFS offers fair, efficient scheduling for desktops and servers, with cgroups enabling fine-grained resource control.
2. Strength: POSIX-Compliant Threading: NPTL ensures high performance and standard compliance for robust, multithreaded applications.
3. Limitation: High Resource Use: Each thread/process consumes kernel memory and stack, limiting scalability under constrained resources.
4. Limitation: Overhead in Context Switching: Frequent switches between threads or processes add kernel-level overhead, affecting performance in I/O-heavy apps.
5. Limitation: Complexity: Sophisticated scheduling and thread management may require tuning for specific workloads.

Process Scheduling in Zorin OS

Process Scheduling in Zorin OS
Overview:

- Zorin OS, built on Ubuntu and the Linux kernel, inherits the Linux process scheduling system, primarily using the Completely Fair Scheduler (CFS). CFS ensures that each process gets a fair share of CPU time by maintaining a virtual runtime for every task. Lower runtime tasks are prioritized, promoting responsiveness and efficiency. Zorin OS supports preemptive multitasking, meaning processes can be interrupted to serve more critical tasks. It also handles background, foreground, and daemon processes seamlessly. Users can adjust priorities with tools like nice and renice. For real-time needs, Zorin leverages SCHED_FIFO and SCHED_RR, ensuring high performance when necessary.
-



Fair CPU Allocation - Zorin OS uses the Completely Fair Scheduler (CFS) to ensure balanced CPU time across all processes based on priority.

Real-Time Scheduling - Supports FIFO, Round Robin, and Deadline policies for time-sensitive tasks like audio or robotics.

User and System-Level Control
Zorin OS allows both the system and users to influence scheduling through niceness values and priority settings, enabling flexible and responsive multitasking even on older hardware.

Synchronization in Zorin OS

How Synchronization Works in Zorin OS:

Inherits all Linux kernel synchronization methods (from Ubuntu base).
Used at both kernel level (drivers, modules) and user level (apps, services).
Key techniques include:
Mutexes - Only one thread/process can access a critical section at a time.
Semaphores - Manage access using counters for shared/exclusive access.

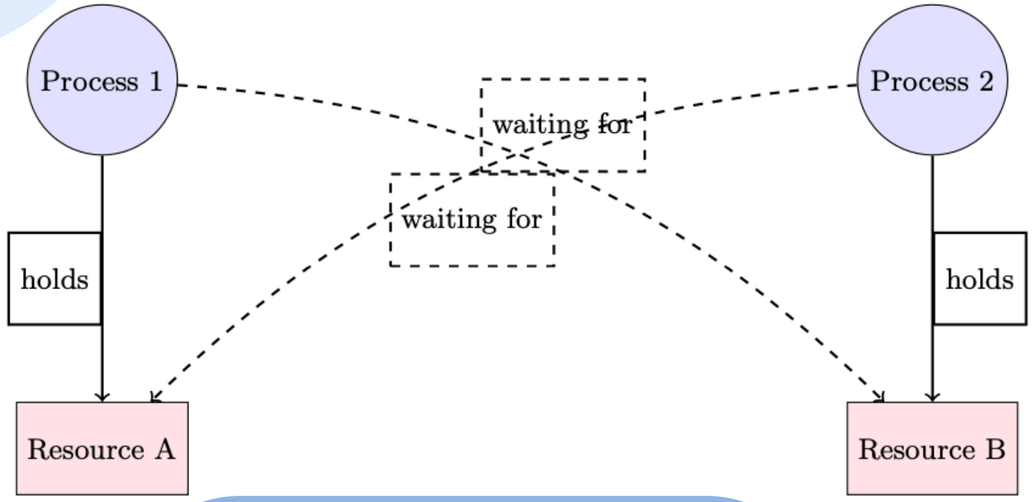
Types of Synchronization:

- Process Synchronization - Ensures processes execute in order (used in IPC, pipelines, shared memory).
- Thread Synchronization - Manages thread execution within a process; prevents race conditions.
- Data Synchronization - Maintains consistency across apps/devices (e.g., cloud sync like Dropbox).

Managing Shared Resources:

- Mutexes & Semaphores - Control exclusive/limited access to shared data; prevent deadlocks.
- Spinlocks - Ideal for real-time, low-level tasks; avoids sleeping during waits.
- Read/Write Locks - Allow multiple readers or one writer; used in logging, caching, config management.

deadlock in Zorin OS



This diagram shows a deadlock where Process 1 holds Resource A and waits for Resource B, while Process 2 holds Resource B and waits for Resource A. Since neither can proceed, both are stuck, creating a circular wait — a key cause of deadlock.

What is Deadlock?

A condition where two or more processes wait indefinitely for each other's resources. It occurs when these four conditions hold:

- **Mutual Exclusion:** One process at a time can use a resource
- **Hold and Wait**:A process holds one resource and waits for another
- **No Preemption :** You can't force a process to give up a resource
- **Circular Wait :** Each process is waiting for the next in a loop

Detection in Zorin OS:

Zorin OS doesn't show deadlocks by itself, but we can use tools like:

- **ps aux:** Shows all programs; if it says "D" state, it might be stuck
- **strace:** Shows what a program is trying to do and where it's stuck
- **top:** shows which programs are using a lot of memory or not responding

How to Fix It:

- **Stop one of the programs using kill -9 <PID>**
- **Take the resource away (not common for normal programs)**
- **Set a time limit so a program doesn't wait forever**

How to Prevent Deadlock:

- **Always lock resources in the same order**
- **Ask for all needed resources at once**
- **Add timeouts so a program doesn't wait forever**

Legal & Ethical Aspects of Zorin OS

1

Built-in Security Measures: Features like AppArmor (application sandboxing), UFW firewall, and LUKS full-disk encryption secure the system from unauthorized access.

2

Privacy-First Design: User data collection is opt-in, and Zorin OS includes controls for location, file history, and app permissions (especially for Snap and Flatpak).

3

Open-Source Licensing (GPL v2.0): Zorin OS follows GNU GPL v2.0, ensuring software remains free, shareable, and any modifications must also be open-source.

4 - Transparency Practices: Zorin maintains a warrant canary, signaling that no secret government data access requests have been made.

5 - Compatibility Security: Windows app compatibility via WINE includes extra layers of protection against malware and system compromise.

6 - Ethical Compliance: Emphasizes freedom, transparency, and user control—making it ethically sound for users valuing privacy and open development.

Memory

Virtual Memory

Despite the tiny size of the real RAM, the operating system (OS) may make every program appear to have access to a big, continuous block of memory by using virtual memory. Imagine that your workstation is little, and you are reading a massive book. Thus, you just keep a small number of pages on your desk; the others remain on the shelf. You can switch it out whenever you require a different page. The OS does just that, using the hard drive as an extension of the RAM. Programs that are not in use are moved to the swap area, a location on the disk, when the system's physical memory runs low.

Memory Allocation

How memory is allocated to processes by the OS: Contiguous Allocation: Easy, yet fragmenting. Each process is given a single block. Paging: no external fragmentation. There are fixed-size blocks of memory. Division: creates logical divisions in memory (code, data, stack). Adaptive Allocation Techniques: The initial fit designates the first sufficiently large hole. Finding the smallest appropriate.
opening (reducing waste) is the best fit. Worst Fit: Locates the biggest hole; may leave significant residue. Choice of Zorin OS: Our operating system will minimize fragmentation while preserving performance by utilizing paging in conjunction with a Best Fit dynamic allocation technique.

File Management

One of an operating system's primary duties is file management. It manages all aspects of generating, preserving, classifying, gaining access to, and safeguarding files and folders (also known as directories).

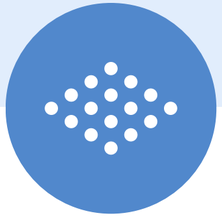
Types of Files: Ordinary Files: Videos, executables, text files, etc. Directories: Hold additional directories and files. Similar to shortcuts, symbolic links direct users to another file or folder. Hardware devices are represented by special files, such as /dev/sda for disk. For instance, Zorin OS makes use of the ext4 file system, which is a dependable and quick Linux file system. It is compatible with: Journaling (assists in crash recovery) Ownership and Permissions Large partitions and files Metadata based on inodes

Zorin OS Complete Implementation

https://drive.google.com/file/d/1iv_1_I3X3xi3AcG0P3xdV-26vHa2r6KO/view?usp=sharing

Conclusion

Zorin OS shows how modern Linux systems manage multitasking, memory, synchronization, and security in a real, user-friendly way. It meets technical goals and works well for students, developers, and everyday users.



THANK YOU!