JOS operating system implementation

This project involves the development of the Jos operating system. The project has progressed significantly, with five of its laboratories successfully completed. Each laboratory has focused on different aspects of the operating system, ranging from the basic kernel functionalities to advanced features such as process management, memory allocation, file systems, and user interface design. The details of each part can be explored in the link provided below. The Jos operating system project is an educational endeavor that allows students and developers to gain hands-on experience in OS development. By working through the laboratories, participants learn critical concepts and skills necessary for building a functional operating system. These skills include low-level programming, debugging, system calls, interrupt handling, and concurrency. The completed laboratories provide a comprehensive understanding of the operating system's inner workings and offer a solid foundation for further development and enhancement. Future laboratories and tasks will continue to build on this foundation, incorporating more complex features and optimizations to improve the overall performance and usability of Jos. For a detailed breakdown of each laboratory and the specific objectives and outcomes achieved, please refer to the link below. This link will guide you through the various stages of the project, showcasing the progress made and the technical challenges overcome in developing the Jos operating system.

In summary, to set up the Jos operating system on Ubuntu 14.04, we installed the necessary tools (binutils, gcc, gdb), navigated to the jos directory, enabled virtualization, and started QEMU using make qemu-nox. The initial commands help and kerninfo were tested successfully, indicating that the basic system setup is operational.

In this project, I developed a 64-bit UNIX-like operating system from the ground up. The key concepts I implemented include:

- Virtual memory management utilizing a 4-level page table, with features such as page fault handling and copy-on-write (CoW).
- Concurrency and multi-programming support through a round-robin scheduler and timer interrupts.
- Inter-process communication mechanisms.
- Filesystem implementation.