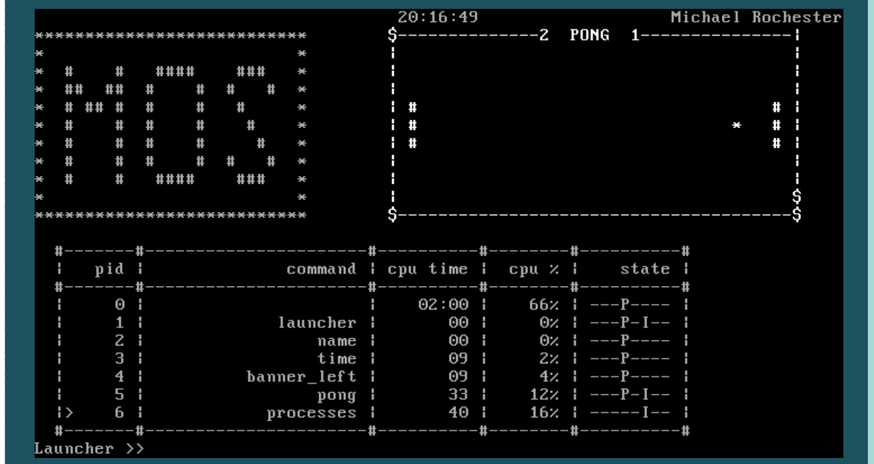
MOS: A Simple Multitasking Operating System

Michael Rochester

Developed in the powerful C language with embedded ASM for direct hardware control.

The Platform

MOS creates a platform for user-space applications to run. These applications can make use of hardware resources such as files, screens, and keyboards along with system resources such as time services, process management and, interprocess communications.



Targeting i386 Architecture with ELF binary format for compatibility with many modern platforms.

Mos is an i386-ELF Operating system designed with five key tenants in mind:

Fairness

MOS allows programs fair access of all hardware, maintaining cooperative sharing of CPU time, System Memory, Hard drives, Screen space, and Input devices.

Simplicity

With the vast number of tasks MOS must perform, simple system calls are provided, such that each can support reliable, understandable control of the kernel.

Versatility

MOS allows for a wide variety of complex behaviour by providing many common structures and tools such as environment variables, program parameters, process metrics, and interprocess communications.

Stability

MOS is designed to maintain stability regardless of how programs act and will selectively pause or kill offending programs to maintain system operation.

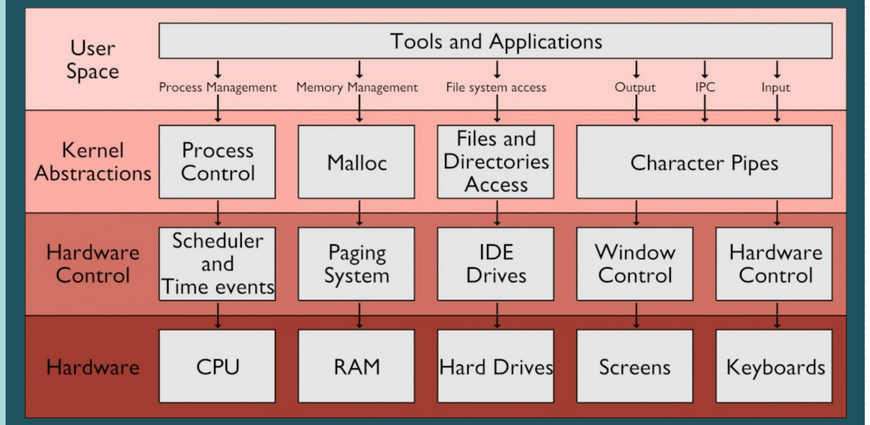
Utility

MOS provides a shell that supports calling programs with parameters and a history of commands, and also comes with a common set of commands such as file manipulations and process management.

Multi-boot compliance so MOS can run along side your other favourite operating system

Modularity

By maintaining strong modularisation, MOS has a clean design with well defined system boundaries. Each module can handle and recover from both internal exceptions and extra-modular failure.



Layered Design

The layered design allows for varying levels of abstractions between the user space applications and the hardware. These abstractions keep each module clean and simple, increasing maintainability and stability of the kernel.

Testing

A wide variety of testing techniques, including Self-Testing, Behavioural-Testing, Limit-Testing, Stress-Testing, and Acceptance testing, make MOS a provably stable system.

Department of Informatics
BSc (Hons) Computer Science
Academic year 2014/15