MOS: A Simple Multitasking Operating System

Michael Rochester

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

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

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

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

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.

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.

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

```
                                  20:16:49              Michael Rochester
**********************      $-------------2 PONG 1--------------
*     *  ****  ###  *      |                                   |
**  ##       #      *      | #                                 |
##  ##  ####   ###  *      | #                                 |
 # ###       #      *      | #                            #    |
*   *  ****  ###  *      |                           * # |
**********************     |                                 $ |
                           $--------------------------------$
                           
#------#------------#--------#-------#--------#
| pid  |    command | cpu time | cpu % |  state  |
#------#------------#--------#-------#--------#
|  0   |            |  02:00  | 66%  | ---P----  |
|  1   |   launcher |   00    | 0%   | ---P-I--  |
|  2   |      name  |   00    | 0%   | ---P----  |
|  3   |      time  |   09    | 2%   | ---P----  |
|  4   | banner_left|   09    | 4%   | ---P----  |
|  5   |      pong  |   33    | 12%  | ---P-I--  |
|> 6   |  processes |   40    | 16%  | -----I--  |
#------#------------#--------#-------#--------#
Launcher >>
```

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.

	Tools and Applications					
User Space	Process Management	Memory Management	File system access	Output	IPC	Input
Kernel Abstractions	Process Control	Malloc	Files and Directorys Access	Character Pipes		
Hardware Control	Scheduler and Time events	Paging System	IDE Drives	Window Control	Hardware Control	
Hardware	CPU	RAM	Hard Drives	Screens	Keyboards	

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.

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

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.

Targeting i386 Architecture with ELF binary format for compatibility with many modern platforms.
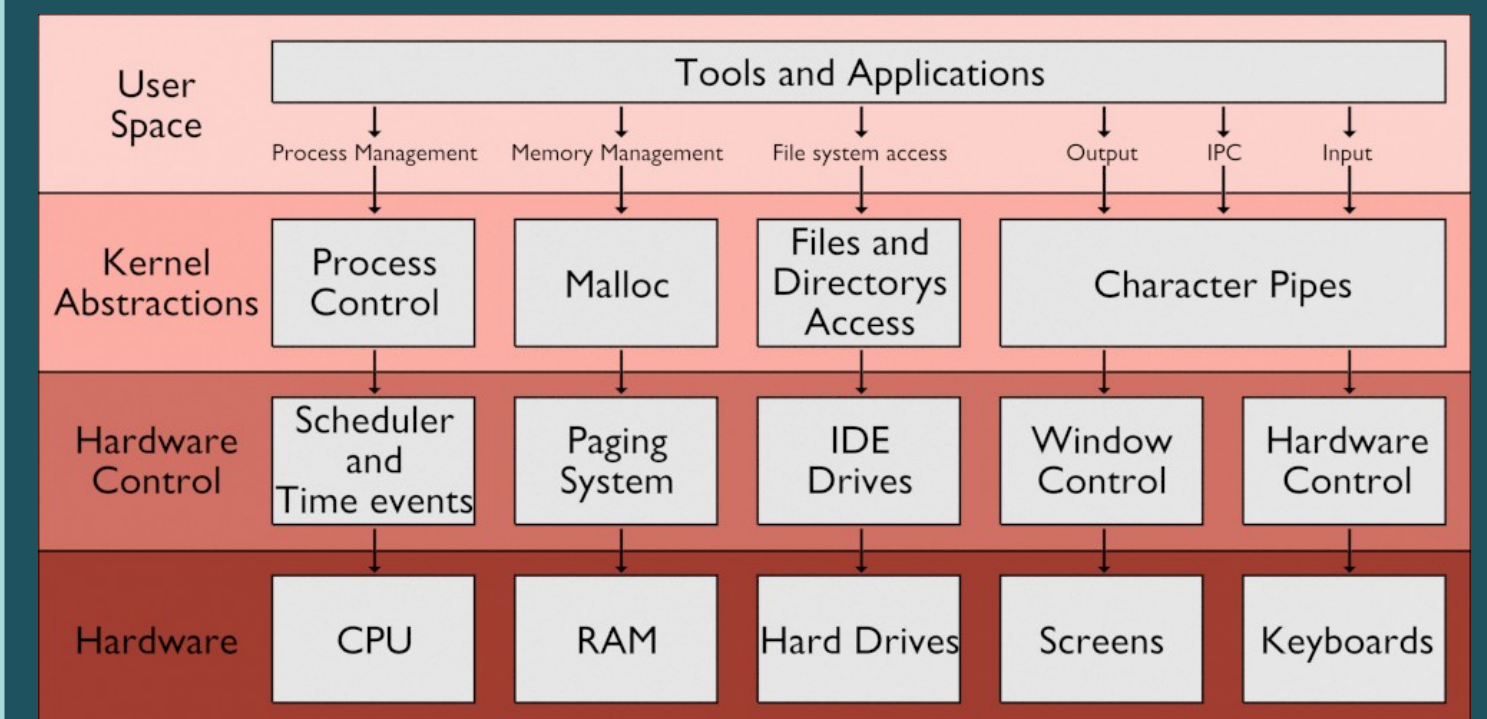
Utility
MOS provides a shell that supports calling programs with parameters and a history of commands, and also comes with a common set of commands such as file manipulations process management.

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