ENSET Mohammedia

جامعة الحسن الثاني بالدار البيضاء
UNIVERSITÉ HASSAN II DE CASABLANCA

Project Operating System Theory The evolution of operating system with mini OS : HmadSO

Department : Math Informatique

Major : Génie du logiciel et Système informatiques distribués

Subject : Operating System theory

Realized By :

AIT EL CIAD HMAD

Under direction :

Mr Azougagh Driss

2016 - 2017

Contents

I. Introduction

Computer operating systems provide a set of functions needed and used by most application programs on a computer, and the links needed to control and synchronize computer hardware. On the first computers, with no operating system, every program needed the full hardware specification to run correctly and perform standard tasks, and its own drivers for peripheral devices like printers and punched paper card readers. The growing complexity of hardware and application programs eventually made operating systems a necessity for everyday use.

An operating system acts as an intermediary between the user of a computer and the computer hardware. The purpose of an operating system is to provide an environment in which a user can execute programs in a convenient and efficient manner.
An operating system is software that manages the computer hard- ware. The hardware must provide appropriate mechanisms to ensure the correct operation of the computer system and to prevent user programs from interfering with the proper operation of the system.

Internally, operating systems vary greatly in their makeup, since they are organized along many different lines. The design of a new operating system is a major task. It is important that the goals of the system be well defined before the design begins. These goals form the basis for choices among various algorithms and strategies.
Because an operating system is large and complex, it must be created piece by piece. Each of these pieces should be a well-delineated portion of the system, with carefully defined inputs, outputs, and functions.

So in this report I am going to present before operating system what is an operating system, First operating system also some example of operating systems and in the end I am going to present my own operating system which called HmadOS.

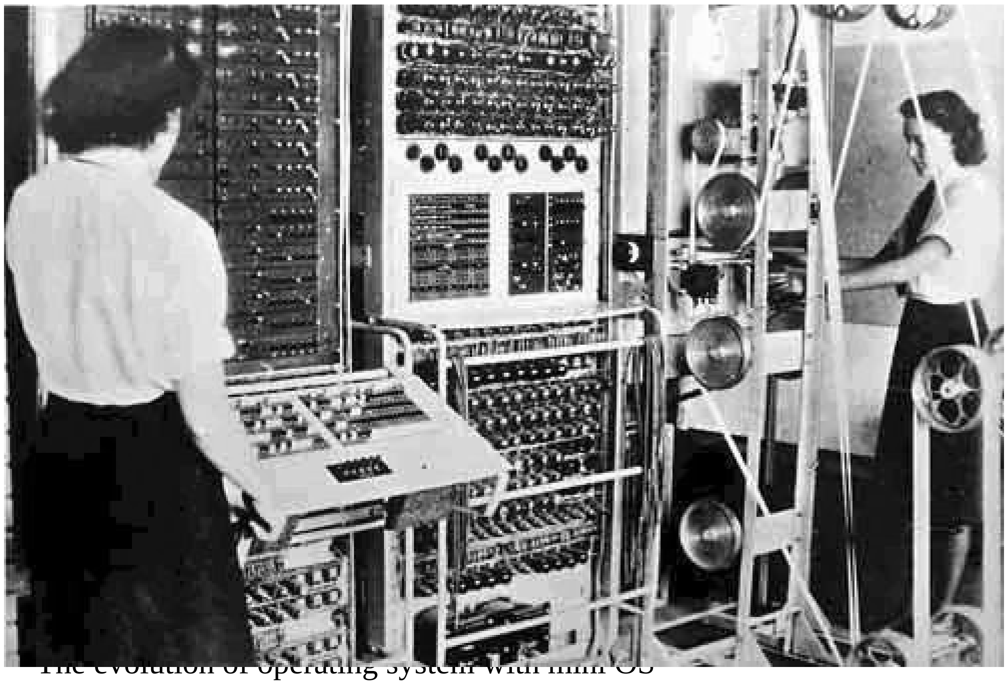
II. Before operating system

Early computers† ran one program at a time.

Programs were directly loaded from (for example) paper tape with holes punched in it.

You'd program the earliest computers† by setting a large set of on-off switches.

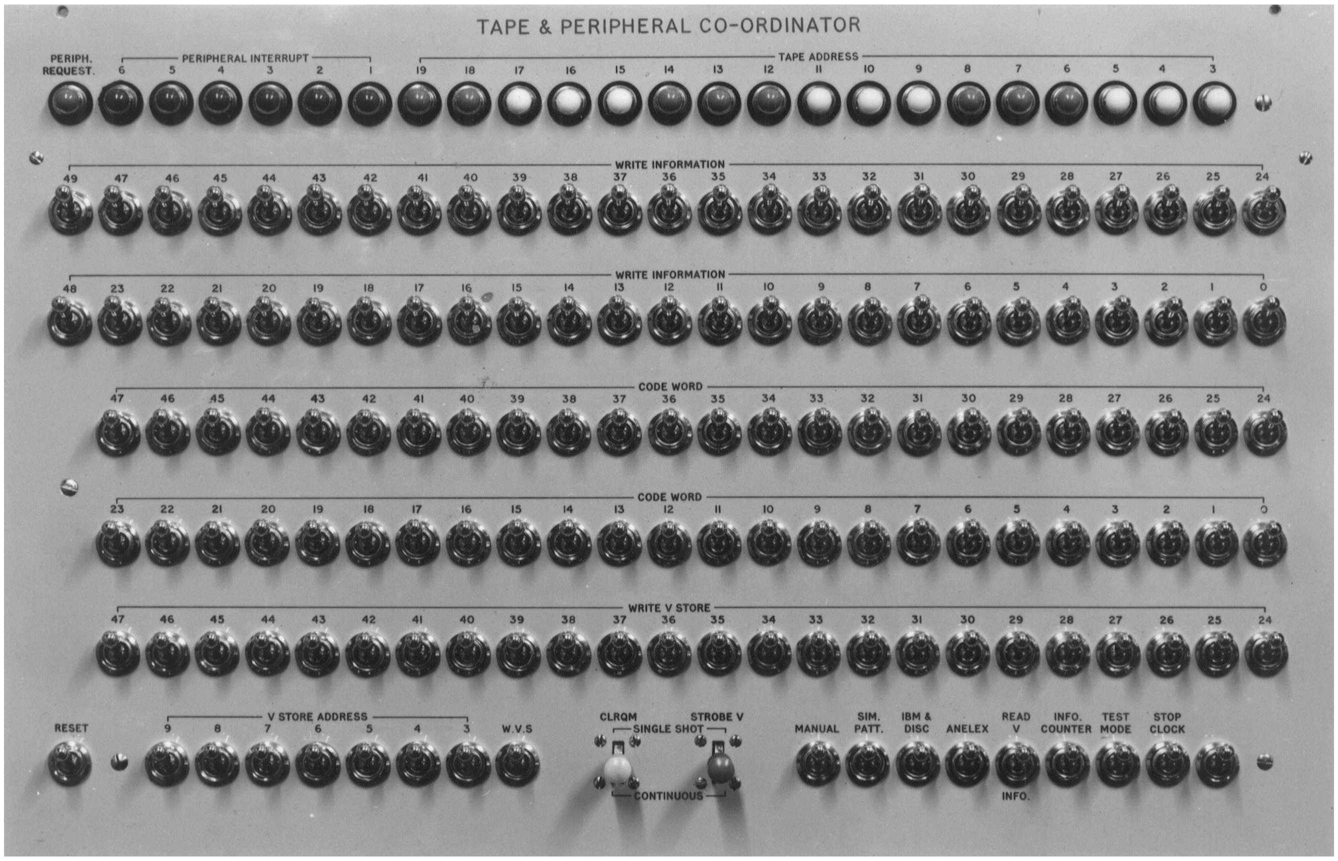
Colossus was a set of computers developed by British codebreakers in 1943–1945 to help in the cryptanalysis of the Lorenz cipher. Colossus used thermionic valves (vacuum tubes) to perform Boolean and counting operations. Colossus is thus regarded[2] as the world's first programmable, electronic, digital computer, although it was programmed by switches and plugs and not by a stored program.[3]



The **Atlas Computer** was a joint development between the University of Manchester, Ferranti, and Plessey. The first Atlas, installed at Manchester University and officially commissioned in 1962, was one of the world's first supercomputers, considered to be the most powerful computer in the world at that time.[1] It was said that whenever Atlas went offline half of the United Kingdom's computer capacity was lost.[2] It was a second-generation machine, using discrete germanium transistors. Two other Atlas machines were built: one for British Petroleum and the University of London, and one for the Atlas Computer Laboratory at Chilton near Oxford.

A derivative system was built by Ferranti for Cambridge University. Called the Titan, or Atlas 2, it had a different memory organisation and ran a time-sharing operating system developed by Cambridge University Computer Laboratory. Two further Atlas 2s were delivered: one to the CAD Centre in Cambridge (later called CADCentre, then AVEVA), and the other to the Atomic Weapons Research Establishment (AWRE), Aldermaston.

The University of Manchester's Atlas was decommissioned in 1971,[3] but the last was in service until 1974.[4] Parts of the Chilton Atlas are preserved by National Museums Scotland in Edinburgh; the main console itself was rediscovered in July 2014 and is at Rutherford Appleton Laboratory in Chilton, near Oxford. CADCentre's Atlas 2 was decommissioned in late 1976.


TAPE & PERIPHERAL CO-ORDINATOR
PERIPH. REQUEST.
PERIPHERAL INTERRUPT
TAPE ADDRESS
WRITE INFORMATION
CODE WORD
WRITE V STORE
RESET
V STORE ADDRESS
W.V.S
CLRQM
STROBE V
SINGLE SHOT
CONTINUOUS
MANUAL
SIM. PATT.
IBM & DISC
ANELEX
READ V
INFO.
INFO. COUNTER
TEST MODE
STOP CLOCK

The **Manchester Mark 1** was one of the earliest stored-program computers, developed at the Victoria University of Manchester from the Small-Scale Experimental Machine (SSEM) or "Baby" (operational in June 1948). It was also called the **Manchester Automatic Digital Machine**, or **MADM**.[1] Work began in August 1948, and the first version was operational by April 1949; a program written to search for Mersenne primes ran error-free for nine hours on the night of 16/17 June 1949.

[†]I am using the word "Computer" to mean the sort of device that exists nowadays in the billions. Of this vast number of computers, all but an insignificantly tiny number are digital electronic programmable computers with stored programs. I'm sure the original question is not about how people with the job title "Computer" spent their working day. In between those two types of computer, there is a progression of interesting devices not covered in this answer.

III. What is operating system

An operating system acts as an intermediary between the user of a computer and the computer hardware. The purpose of an operating system is to provide an environment in which a user can execute programs in a convenient and efficient manner.

An operating system is software that manages the computer hardware. The hardware must provide appropriate mechanisms to ensure the correct operation of the computer system and to prevent user programs from interfering with the proper operation of the system.

Internally, operating systems vary greatly in their makeup, since they are organized along many different lines. The design of a new operating system is a major task. It is important that the goals of the system be well defined before the design begins. These goals form the basis for choices among various algorithms and strategies.

Because an operating system is large and complex, it must be created piece by piece. Each of these pieces should be a well-delineated portion of the system, with carefully defined inputs, outputs, and functions.

IV. First operating system

The first operating system used for real work was GM-NAA I/O, produced in 1956 by General Motors' Research division[2] for its IBM 704.[3] Most other early operating systems for IBM mainframes were also produced by customers.[4]

Early operating systems were very diverse, with each vendor or customer producing one or more operating systems specific to their particular mainframe computer. Every operating system, even from the same vendor, could have radically different models of commands, operating procedures, and such facilities as debugging aids. Typically, each time the manufacturer brought out a new machine, there would be a new operating system, and most applications would have to be manually adjusted, recompiled, and retested.

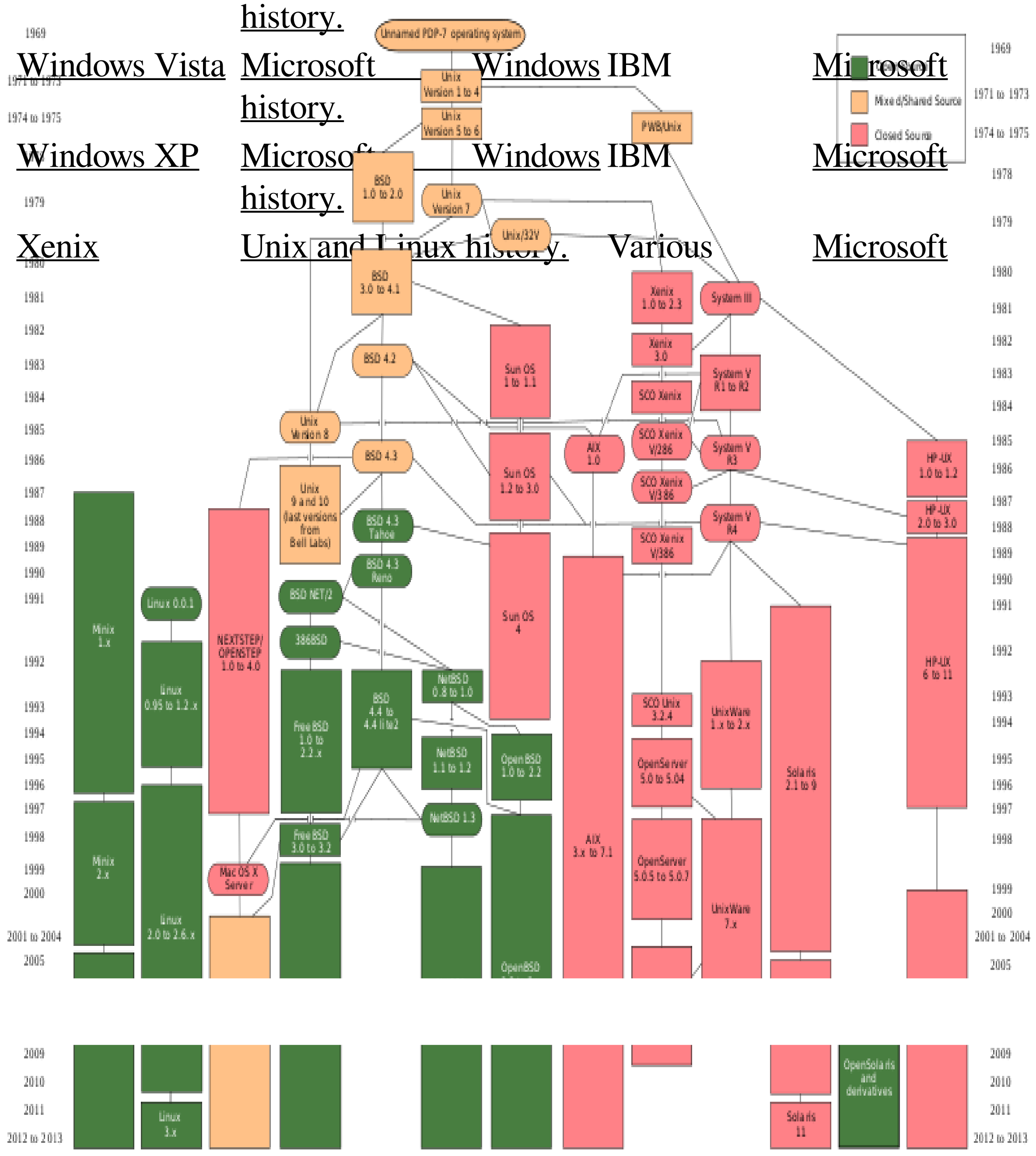
V. liste of operating system

Operating system	Date first released	Platform	Developer
AIX and AIXL	Unix and Linux history.	Various	IBM
AmigaOS	Currently no AmigaOS history.	Amiga	Commodore
Android	November 5, 2007	Mobile	Google
BSD	Unix and Linux history.	Various	BSD
Caldera Linux	Unix and Linux history.	Various	SCO
Corel Linux	Unix and Linux history.	Various	Corel
CP/M	Currently no CP/M history.	IBM	CP/M
Debian Linux	Unix and Linux history.	Various	GNU
DUnix	Unix and Linux history.	Various	Digital
DYNIX/ptx	Unix and Linux history.	Various	IBM
HP-UX	Unix and Linux history.	Various	Hewlett

			Packard
iOS	2010	Mobile	Apple
IRIX	Unix and Linux history.	Various	SGI
Kondara Linux	Unix and Linux history.	Various	Kondara
Linux	Unix and Linux history.	Various	Linus Torvalds
MAC OS 8	Apple operating system history.	Apple Macintosh	Apple
MAC OS 9	Apple operating system history.	Apple Macintosh	Apple
MAC OS 10	Apple operating system history.	Apple Macintosh	Apple
MAC OS X	Apple operating system history.	Apple Macintosh	Apple
Mandrake Linux	Unix and Linux history.	Various	Mandrake
MINIX	Unix and Linux history.	Various	MINIX
MS-DOS 1.x	MS-DOS history.	IBM	Microsoft
MS-DOS 2.x	MS-DOS history.	IBM	Microsoft
MS-DOS 3.x	MS-DOS history.	IBM	Microsoft
MS-DOS 4.x	MS-DOS history.	IBM	Microsoft
MS-DOS 5.x	MS-DOS history.	IBM	Microsoft
MS-DOS 6.x	MS-DOS history.	IBM	Microsoft
NEXTSTEP	Apple operating system history.	Various	Apple
OS/2	1987	IBM	IBM
OSF/1	Unix and Linux history.	Various	OSF
QNX	Unix and Linux history.	Various	QNX
Red Hat Linux	Unix and Linux history.	Various	Red Hat
SCO	Unix and Linux history.	Various	SCO
Slackware Linux	Unix and Linux history.	Various	Slackware

Sun Solaris	Unix and Linux history.	Various	Sun
SuSE Linux	Unix and Linux history.	Various	SuSE
Symbian	1997	Mobile	Nokia
System 1	Apple operating system history.	Apple Macintosh	Apple
System 2	Apple operating system history.	Apple Macintosh	Apple
System 3	Apple operating system history.	Apple Macintosh	Apple
System 4	Apple operating system history.	Apple Macintosh	Apple
System 6	Apple operating system history.	Apple Macintosh	Apple
System 7	Apple operating system history.	Apple Macintosh	Apple
System V	Unix and Linux history.	Various	System V
Tru64 Unix	Unix and Linux history.	Various	Digital
Turbolinux	Unix and Linux history.	Various	Turbolinux
Ultrix	Unix and Linux history.	Various	Ultrix
Unisys	Unix and Linux history.	Various	Unisys
Unix	Unix and Linux history.	Various	Bell labs
UnixWare	Unix and Linux history.	Various	UnixWare
VectorLinux	Unix and Linux history.	Various	VectorLinux
Windows 2000	Microsoft Windows history.	IBM	Microsoft
Windows 2003	Microsoft Windows history.	IBM	Microsoft
Windows 3.X	Microsoft Windows history.	IBM	Microsoft
Windows 7	Microsoft Windows history.	IBM	Microsoft
Windows 8	Microsoft Windows	IBM	Microsoft

	history.		
Windows 95	Microsoft Windows history.	IBM	Microsoft
Windows 98	Microsoft Windows history.	IBM	Microsoft
Windows 10	Microsoft Windows history.	IBM	Microsoft
Windows CE	Microsoft Windows history.	PDA	Microsoft
Windows ME	Microsoft Windows history.	IBM	Microsoft
Windows NT	Microsoft Windows history.	IBM	Microsoft
Windows Vista	Microsoft Windows history.	IBM	Microsoft
Windows XP	Microsoft Windows history.	IBM	Microsoft
Xenix	Unix and Linux history.	Various	Microsoft



VI. the objectif a operating system

VII. the compenent of an operating system

VIII. how does it work

IX. comparaison between the oprating systems

X. how to create your own operating system

http://www.osdever.net/bkerndev/Docs/whatsleft.htm

I will tell you about my personal experience of writing a x86 64 bit OS from scratch. Like many others who have answered, my first experience of building an OS comes from an OS class which I took during masters. You can find source code of my OS here: mysterious4 / SBUnix - Bitbucket

Programming OS is one of the most interesting and satisfying experience in terms of learning.

Get yourself familiar with qemu (x86 emulator), gdb (debugging), ctags (Navigating code), git (maintaining repo).

I hope you have sufficient knowledge of C, ASM, x-86 architecture, calling convention for x86 .

Firstly, I think you should really ask yourself what level of complexity you wish to have in your OS. Sketch out the rough design, components you wish to program etc.

If learning OS concepts is your main aim, I would suggest start with an existing implementation; something like MIT - JOS. 6.828 / Fall 2011 / Overview
It will provide you with skeleton code for memory management, page table setup stuff and then you would incrementally add support of various syscalls, interrupt handlers etc.

If you wish to go the hard way and program everything yourself, roll up your sleeves, its going to be exhaustive. I will enumerate steps for you:

Step 1. Writing your own bootloader: Use BOCHS which is a 32bit emulator. Read some theory about where BIOS is loaded, real modes, protected mode and then start implementing. This link might help you : Category:Babystep - OSDev Wiki

Step 2. Write code for defining your kernel entry point where before calling main, you will setup your stack, GDT tables etc.

Step 3. Now you have task of writing ISR (interrupt service routines). Start from basic stuff. Set up your own GDT, IDT tables. Write your own printf by writing to video memory at 0xB8000. Refer this Printing To Screen

Step 4. Writing code for PIC (programmable interrupt microcontroller). PIC tells processor what IRQ to call within our IDT (interrupt descriptor table) when interrupt is recieved from device. Refer this : Operating Systems Development Series. PIT (programmable interrupt timer) is your system clock. Code up a simple timer handler.

Step 5. Have a basic device driver for your keyboard by mapping its scan codes. Write this into your keyboard interrupt handler.

Step 6. Now comes part of handling physical memory. Your BIOS e801 map will give you total main memory available. Further when you will come down to paging, it is going to request you physical pages of size 4 Kb. Divide memory into page size and have a bit-map/free list type physical memory manager which will keep track of occupied and free pages.

Step 7. Now arrives the mammoth task of programming virtual page tables. Start by reading first some theory that you won't usually find in OS books like one's which explain when to load cr3 register with physical addr of pml4 in case of 64-bit to enable paging, self referencing page tables etc. Map your kernel to higher memory address.
Write code for heap memory management, kernel malloc etc.

Step 8. Now comes part of process management. Define simple datastructres for your managing your processes - PCB, MM_struct, task_struct (required later on) etc. (refer linux 1.0). Now write procedure for loading simple kernel level process. Once you have this running, go on do define ring permissions and then user level processes.
Write a basic round robin scheduler by keeping simple runqueue.

Step 9. Once you have processes running, now you have job of loading processes from memory. Define very naive tarfs kind of in memory file system, then procedure for reading elf binary and loading it in your memory.

Step 10. If you want to proceed more, you have task of writing your own file system and later on you can go on to add support for network stack, multi threading etc.

Wish you all the best and definitely I would love to hear back about your experience building a mini os.

XI. how to craete your own distribution of gnu/linux

XII. how you can be a part of the community of developing kernel of linux

Annex
https://en.wikipedia.org/wiki/Atlas_(computer)
https://en.wikipedia.org/wiki/Colossus_computer#Operation
http://www.computerhope.com/os.htm