OPERATING SYSTEM

OS introduction including history and implementation, machine type

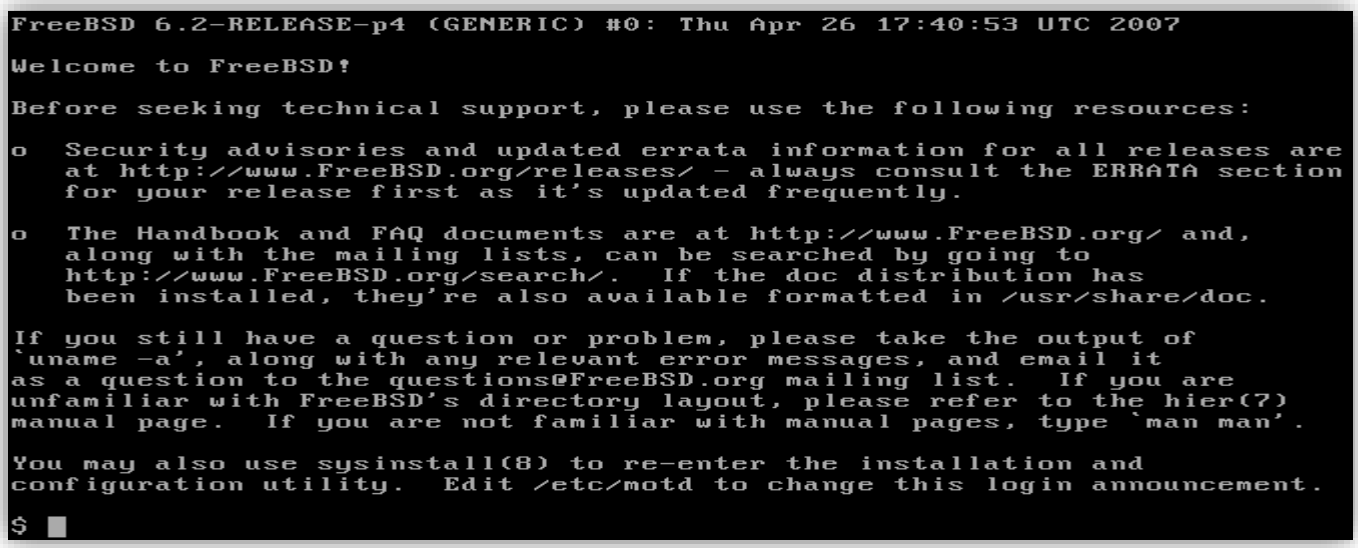
Student name	ID
Mahmoud Essam Fathy	20221460231
Abdelrahman Ashraf Ragab	20221374041
Abdullah Hussein Ibrahim	20221427861
Ziyad Ashraf Hafez	20221374025
Marwan Ali Abd-Elsatar	20221460240

Supervised by

Dr.Yasser Fouad

Assignment: OS introduction including history and implementation, machine type
Subject: Operating System
Dr. Yasser Fouad

Contents

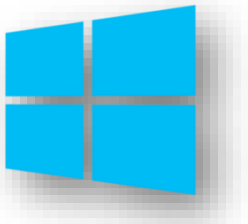
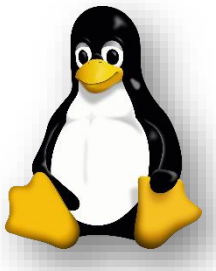


Assignment: OS introduction including history and implementation, machine type
Subject: Operating System
Dr. Yasser Fouad

What's an Operating System?

a modern computer consists of one or more processors, some main memory, disks, printers, a keyboard, a mouse, a display, network interfaces, and various other input/output devices, but also, how all this will work without something working as an access point between all of those hardware components and the application of software?

- Operating System Definition: acts as an intermediary between the user of a computer and the computer hardware.
- Purpose? providing an environment in which a user can execute programs in a convenient and efficient manner.
- The example OS readers are a lot, like Windows, Linux, FreeBSD, or even OS X and all of them have their own type of program to interact with it, sometimes called Shell or Bash in Linux, and Windows setting has its own Desktop environment to make it interactable with the user.
- Our main OS for the remaining tasks is FreeBSD, and we will discuss it in detail.

What's the History of it?

Time	Action
1940 : 1950s vacuum tubes	it was created without any operating system; users have full access to the computer machine and write a program for each task in absolute machine language
1955 – 1965 Transistors and Batch Systems	First OS ever in 1950 called GMOS by IBM The second-generation operating system was based on a single stream batch processing system

1965 – 1980 Multiprogramming: performing different tasks on a computer at the same time. Do you know that AMD kind better than intel in multiprocessing ?	In late 1960s, designers were very capable of developing a new operating system that could simultaneously perform multiple tasks in a single computer program called multiprogramming and in this generation has existed minicomputer which was its name is DEC PDP-1
1980 – Present Day Personal Computers	Generally, the development of Personal Computers is very similar to the minicomputers that were developed in the third generation, was very high cost this time The major factor this time was birth of Microsoft with their Windows by 1975, as they first MS-DOS was introduced at 1981 with dificullty sure to make simple human understand it Then it upgraded to windows 95, then 98, and XP which was so popular this time, with 10 versions and last one is 11 by 2021

What about FreeBSD?

FreeBSD introduced first early 1993, it was the brainchild of the unofficial OS 386BSD it was Bill Jolitz's OS, which had been up to that point suffering rather severely from almost a year's worth of neglect.

The first distribution of was FreeBSD 1.0 in 1993

The Goal of this FreeBSD that to provide software that may be used for any purpose and without strings attached.

Add to that the very open and flexible process of its development, being literally built from the contributions of thousands of people around the world as a great Open-source OS.

FreeBSD offers a ported software collection with thousands of commonly sought-after programs.

There are about 36000 ports; the entire Ports Collection requires approximately 3 GB.

Almost every port is also provided as a pre-compiled "package", which can be installed with a simple command (pkg install) by those who do not wish to compile their own ports from the source.

Implementation

What's implementation from an Operating system view?

OS implementation is the process of writing and testing the source code for an operating system. It is a complex and challenging task that requires a deep understanding of computer hardware and software architecture.

OS Implementation Steps:

1. Design: The first step is to design the OS architecture. This includes defining the different components of the OS and how they will interact with each other.
2. Implementation: Once the architecture is designed, the different components of the OS are implemented in a programming language such as C or C++. The OS is typically implemented as a kernel, which is the core of the OS and provides basic services such as memory management, process management, and device management.
3. Testing: Once the OS is implemented, it needs to be thoroughly tested to ensure that it is stable and reliable. This involves testing the OS under different conditions and with different workloads.
4. Deployment: Once the OS is tested and ready for use, it can be deployed on computers. This may involve installing the OS on a hard drive or running it from a live USB drive.

Key components of kernels include:

Process management: The kernel is responsible for creating, scheduling, and destroying processes. It also manages the memory and other resources that are allocated to each process.

Memory management: The kernel is responsible for managing the computer's memory. It allocates memory to processes and ensures that they do not interfere with each other's memory.

Device management: The kernel is responsible for managing the computer's hardware devices. It provides a unified interface to the devices so that processes can access them without having to know the specific details of the hardware.

File system: The kernel provides a file system that allows processes to store and retrieve data.

Networking: The kernel provides a networking stack that allows processes to communicate with each other over a network.

Examples of OS implementation

The Windows kernel:

The Windows kernel is implemented in C and C++. It is a hybrid kernel, which means that some of the kernel components are compiled into a single binary file, while others are implemented as separate device drivers.

The core of the Windows kernel is implemented in C. This includes components such as the process manager, memory manager, and device manager. Device drivers are typically implemented in C or C++, depending on the specific device.

The Windows kernel is also modular, which means that it is made up of a number of separate components that can be loaded and unloaded as needed. This makes the kernel more flexible and adaptable.

The Unix kernel:

Unix kernels are typically implemented in C and assembly language. The C code is used to implement the kernel's core functionality, such as process management, memory management, and device management. The assembly language code is used to implement low-level hardware-specific functionality, such as interrupt handling and context switching.

The Unix kernel is typically implemented as a monolithic kernel, which means that all of the kernel components are compiled into a single binary file. This makes the kernel very efficient, but it also makes it more difficult to maintain and update.

The Linux kernel:

Linux kernel is implemented in C. Since it's UNIX-like it is also a monolithic kernel, which means that all the kernel components are compiled into a single binary file.

Each of these OS kernels has its own unique design and implementation. However, they all share some common features, such as memory management, process management, and device management.

FreeBSD

The implementation of FreeBSD is divided into two main parts: the kernel and the userland.

The kernel is the core of the operating system. It is responsible for managing the system's resources, such as the CPU, memory, and disk. The kernel is written in C and is compiled into a single executable file.

The userland is the part of the operating system that is visible to users. It includes the shell, command-line utilities, and graphical user interface (GUI). The userland is written in a variety of languages, including C, C++, Python, and Perl.

The FreeBSD kernel is implemented as a monolithic kernel. This means that all of the kernel's components are compiled into a single executable file. This makes the kernel more efficient, but it also makes it more difficult to modify and debug.

The FreeBSD kernel is divided into a number of different modules.

The main modules are:

- Process management: This module is responsible for managing the execution of processes, which are programs that are running on the system.
- Memory management: This module is responsible for managing the system's memory.
- File system: This module is responsible for providing a way for users to store and organize their files.
- Device drivers: These modules provide an interface between the kernel and the system's hardware devices.

The FreeBSD userland is implemented as a collection of separate programs. The main programs are:

- Shell: The shell is a command-line interpreter that allows users to interact with the operating system.
- Command-line utilities: These utilities provide a variety of functions, such as creating and deleting files, managing directories, and running programs.
- Graphical user interface (GUI): The GUI provides a graphical way for users to interact with the operating system.

The FreeBSD kernel and userland are developed and maintained by a large community of volunteers. The FreeBSD source code is freely available and can be downloaded from the FreeBSD website.

Here are some of the key features of the FreeBSD kernel:

- SMP and SMT support: FreeBSD supports symmetric multiprocessing (SMP) and simultaneous multithreading (SMT). This allows multiple processors and cores to be used to run processes simultaneously.
- Virtual memory: FreeBSD supports virtual memory, which allows the operating system to use more memory than is physically installed on the system.
- Journaling file system: FreeBSD uses a journaling file system called UFS2 by default. This file system is more reliable and less susceptible to data loss than traditional file systems.
- Networking support: FreeBSD includes built-in support for a wide variety of networking protocols, including TCP/IP, UDP, and IPv6.
- Security features: FreeBSD includes a number of security features, such as Pluggable Authentication Modules (PAMs) and Mandatory Access Control (MAC).

FreeBSD is a powerful and reliable operating system that is used in a wide variety of environments, from web servers to embedded systems. It is a good choice for users who need a stable and secure operating system with a wide range of features.

Machine Type

Machine type is a configuration specification that defines the hardware characteristics and resources of a computing system or virtual machine.

Embedded: FreeBSD makes an excellent platform to build embedded systems upon. With support for the x86 (both 32 and 64 bit), ARM, AArch64, RISC-V, POWER, and PowerPC computers, coupled with a robust network stack, cutting edge features, and the permissive BSD license, FreeBSD makes an excellent foundation for building embedded routers, firewalls, and other.

x86 or x86-64: The x86 and x64 architectures refer to the two most widely-used types of instruction set architectures (ISA) created by Intel and AMD. An ISA specifies the behavior of machine code and defines how the software controls the CPU

ARM: Processors are used extensively in consumer electronic devices such as smartphones, tablets, wearables and other mobile devices.

AArch64: Is a machine type that refers to the 64-bit ARM architecture. It is an evolution of the ARM architecture designed by ARM Holdings. AArch64 provides a 64-bit instruction set architecture (ISA) and is commonly used in a variety of computing devices, including mobile devices, servers, embedded systems, and IoT devices.

RISC-V: Open-source instruction set architecture used to develop custom processors for a variety of applications, from embedded designs to supercomputers.

PowerPC: a RISC (Reduced Instruction Set Computer) architecture which are very powerful and low-cost microprocessors

Architecture	Support level[56]	Notes
x86-64	Tier 1	referred to as "amd64"
x86 (IA-32)	Tier 1 (tier 2 in 13.x)	referred to as "i386"
64-bit ARM	Tier 1	
32-bit ARM	Tier 2	
MIPS	Tier 2	
32-bit and 64-bit PowerPC	Tier 2	
64-bit SPARC	Tier 2 (tier 4 in 13.x)	only 64-bit (V9) architecture
64-bit RISC-V	Tier 3 (tier 2 in 13.x)	as of 12-RELEASE
NEC PC-9801	Tier 4 (tier 2 in 11.x)	referred to as "pc98", support removed in 12-CURRENT[57]
IA-64	(was tier 3)	unsupported as of 11.0
DEC Alpha	(was tier 4)	support discontinued in 7.0

References

1. Andrew – Tanenbaum – Modern Operating Systems Book.
2. Operating System Concetps (Nnth Version)
3. https://www.javatpoint.com/history-of-operating-system
4. https://en.wikipedia.org/wiki/FreeBSD
5. https://docs.freebsd.org/en/
6. https://www.freebsd.org/
7. https://phoenixnap.com/kb/x64-vs-x86
8. https://www.techtarget.com/whatis/definition/ARM-processor#:~:text=Arm%20processors%20are%20used%20extensively,and%20internet%20of%20things%20devices.
9. https://en.wikipedia.org/wiki/AArch64#:~:text=AArch64%20provides%20user%2Dspace%20compatibility,has%20no%2064%2Dbit%20counterpart.
10. https://www.synopsys.com/glossary/what-is-risc-v.html#:~:text=RISC%2DV%20is%20an%20open,from%20embedded%20designs%20to%20supercomputers.
11. https://www.geeksforgeeks.org/powerpc-architecture/