

ENTORNO DE PROGRAMACIÓN

TECNICATURA UNIVERSITARIA EN INTELIGENCIA ARTIFICIAL

22 de marzo de 2023.

ÍNDICE GENERAL

I	FUNDAMENTOS	3
1	HISTORIA	4
1.1	Orígenes de la computadora	4
1.2	Avances de la guerra	6
1.3	La informática moderna	8
1.4	Primeras computadoras personales	12
2	ARQUITECTURA DE LA COMPUTADORA	15
2.1	Gabinete	15
2.2	Placa madre	16
2.3	Fuente de alimentación	16
2.4	Microprocesador	17
2.5	Memoria RAM	18
2.6	Memoria secundaria	19
2.7	Placa de video	21
3	SISTEMAS OPERATIVOS	22
3.1	Proceso de arranque	22
3.2	Descripción	24
3.3	Núcleo	25
3.4	Terminal	26
3.5	Interfaz gráfica	28
3.6	Distribuciones	30
3.7	Proceso de apagado	31
4	CONCEPTOS DE PROGRAMACIÓN	34
4.1	¿Que es la programación?	34
4.2	Lenguaje de programación	35
4.3	Niveles de lenguajes	35
4.4	Compiladores e interpretes	37
4.5	Otros conceptos	37
5	SISTEMA DE ARCHIVOS	38
II	MANEJO DE TERMINAL	39
5.1	Comandos básicos	40
5.2	Shell scripting	40

III	CONCEPTOS ADICIONALES	41
5.3	Contenedores	42
5.4	Control de versiones	42

Parte I

FUNDAMENTOS

HISTORIA

1.1 ORÍGENES DE LA COMPUTADORA

👤 *Blaise Pascal* había desarrollado en 1642 una de las primeras calculadoras mecánicas de la historia, conocida como la «📺 Pascalina». Esta máquina, capaz de realizar operaciones aritméticas básicas mediante el uso de engranajes y ruedas dentadas, fue un importante avance en el campo de la mecánica y la automatización, y sentó las bases para el desarrollo de las máquinas calculadoras modernas. La Pascalina fue utilizada por matemáticos y científicos de la época, y su diseño influyó en el desarrollo de posteriores calculadoras mecánicas.



Figura 1.1: Pascalina

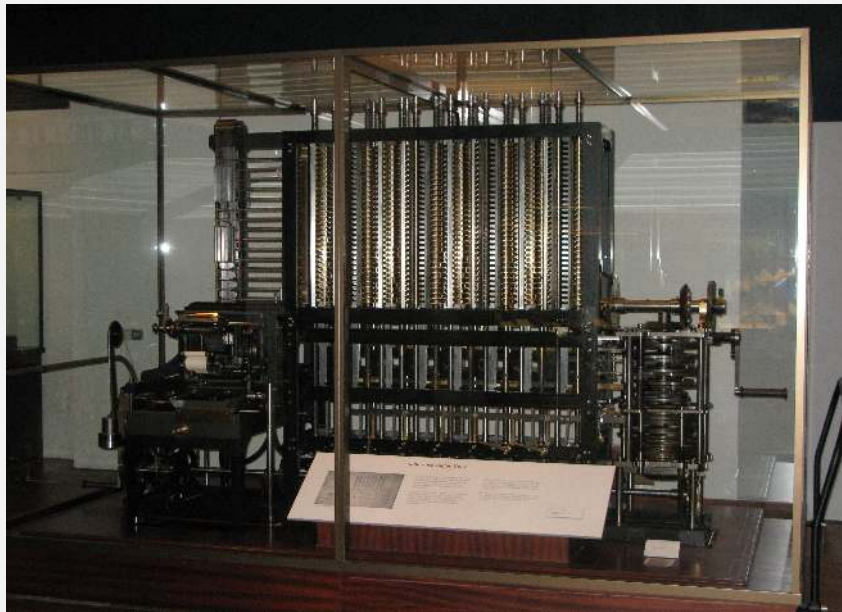


La «📺 máquina analítica» fue un proyecto desarrollado por 👤 *Charles Babbage* en 1837, considerado como el primer prototipo de una computadora moderna. Babbage, matemático, filósofo e inventor inglés, diseñó una máquina capaz de realizar cálculos automáticamente mediante el uso de engranajes



y ruedas dentadas, similar a la Pascalina de Blaise Pascal pero con un alcance mucho mayor. La máquina analítica tenía la capacidad de almacenar programas y realizar operaciones matemáticas complejas, y se considera el primer ejemplo de un sistema de procesamiento de datos automático. Aunque Babbage nunca logró construir la máquina completa debido a problemas financieros y técnicos, su proyecto sentó las bases para el desarrollo de las computadoras modernas y tuvo un gran impacto en el campo de la informática y la automatización.

Figura 1.2: Máquina diferencial



El 📺 teletipo fue una invención que permitió la transmisión automática de texto a través de una red telegráfica. Fue inventado a finales del siglo XIX, y su desarrollo permitió una comunicación más rápida y eficiente que el telégrafo de 📠 Samuel Morse. El teletipo utilizaba un sistema de impresión automática para recibir y transmitir texto, lo que permitió una comunicación más precisa y legible.

! Observación

Las terminales de teletipo (TTY) también se utilizaron como terminales de computadoras en las primeras décadas de la historia de las computadoras.

Figura 1.3: Teletipo Siemens t37h



1.2 AVANCES DE LA GUERRA

El comienzo de la Segunda Guerra Mundial en 1939 motivó un gran avance en la computación debido a la necesidad de procesar grandes cantidades de datos y realizar cálculos complejos para apoyar los esfuerzos de la guerra.



El ordenador  Z3 es una máquina de computación construida en 1941 por  Konrad Zuse, un ingeniero y matemático alemán. Era una máquina electromecánica basada en relés, que utilizaba un sistema binario para representar los datos y los cálculos. El Z3 contaba con una memoria programable, lo que permitía al usuario programar la máquina para llevar a cabo diferentes tareas. Fue destruido en 1943 durante un bombardeo en Berlín.



Figura 1.4: Réplica del Zuse Z3 exhibida en Múnich






La  Harvard Mark I es una computadora electromecánica construida en 1944 por IBM y patrocinada por el gobierno de los Estados Unidos. Fue diseñada por el matemático  *Howard Aiken* y se encuentra en el Museo de la historia de la computación en la Universidad de Harvard. La Harvard Mark I fue una máquina de gran tamaño, medía 8 metros de largo, 2 metros de alto y 2 metros de ancho, y utilizaba relés electromecánicos para llevar a cabo cálculos. La máquina era capaz de realizar aritmética básica y funcionaba con un sistema de tarjetas perforadas.



Figura 1.5: Harvard Mark I

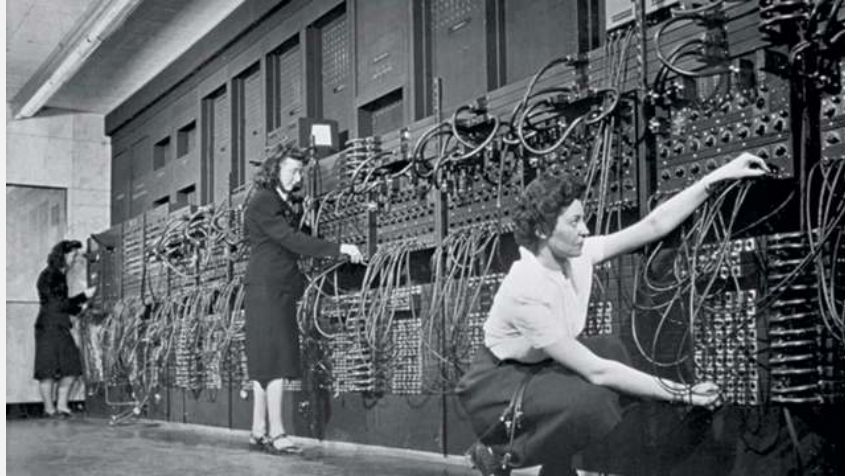


La  ENIAC (Electronic Numerical Integrator And Computer) fue una computadora electrónica programable construida por el gobierno de los Estados Unidos en 1946. Era una máquina enorme, medía 30 metros de largo, 2 metros de alto y 3 metros de ancho, y pesaba 27 toneladas. Utilizaba tecnología de válvulas electrónicas y era capaz de realizar cálculos numéricos complejos a alta velocidad. Fue utilizada para calcular los resultados de las explosiones nucleares y para resolver problemas científicos y técnicos en la industria. Fue inicialmente diseñada para calcular tablas de tiro de artillería destinadas al Laboratorio de Investigación Balística del Ejército de los Estados Unidos.

! Observación

Aunque la tecnología de válvulas era superior a la de relés, no fue hasta la invención del transistor en 1947 que las computadoras ganaron su verdadero poder de computo.

Figura 1.6: Electronic Numerical Integrator And Computer



1.3 LA INFORMÁTICA MODERNA

En la década de 1950, las computadoras eran enormes máquinas electro-mecánicas y electrónicas que ocupaban grandes espacios y requerían un equipo especializado para operarlas. Estas máquinas eran principalmente utilizadas para tareas de cálculo y procesamiento de datos, como la contabilidad y la investigación científica, y eran utilizadas principalmente por grandes empresas, organizaciones gubernamentales y universidades. A pesar de su tamaño y complejidad, estas computadoras marcaron el comienzo de la era informática moderna y sentaron las bases para el desarrollo de las computadoras personales y de escritorio de las décadas siguientes.

FORTTRAN (Formula Translation) es uno de los primeros lenguajes de programación de computadora. Fue desarrollado en 1957 por un equipo de ingenieros de IBM liderado por 🧑 John Backus. El objetivo de FORTRAN era proporcionar un lenguaje de programación que permitiese a los científicos y matemáticos escribir programas de manera eficiente y fácilmente para ser utilizado en las computadoras de ese entonces.





📺 PDP-1 (Programmed Data Processor-1) es un ordenador construido por la compañía Digital Equipment Corporation (DEC) en 1959. Fue el primer ordenador de la serie PDP y uno de los primeros de tipo minicomputadora. El PDP-1 fue un ordenador de tiempo compartido, lo que significa que varios usuarios podían acceder al sistema al mismo tiempo y compartir los recursos del ordenador. Esto fue un gran avance en comparación con los ordenadores anteriores, que solían ser utilizados por un solo usuario a la vez.

Figura 1.7: Programmed Data Processor-1



BASIC (Beginners All-Purpose Symbolic Instruction Code) es un lenguaje de programación creado en el verano de 1964, con el objetivo de desarrollar un lenguaje de programación fácil de aprender y usar para estudiantes no especialistas y principiantes en la programación. El lenguaje se basó en el lenguaje FORTRAN, pero con un enfoque en la simplicidad y la facilidad de uso.

El sistema  NLS (oN-Line System) fue un sistema de información desarrollado por  Douglas Engelbart y su equipo en 1968. Fue una de las primeras demostraciones de una interfaz gráfica de usuario (GUI), con un puntero del ratón sistema de hipertexto y ventanas. También introdujo varias características que se consideran fundamentales en la computación moderna, como el procesamiento de textos, el correo electrónico, la videoconferencia y la colaboración en tiempo real.



MULTICS (Multiplexed Information and Computing Service) fue un sistema operativo desarrollado en 1969 por un equipo liderado por el MIT y Bell Labs. El objetivo de MULTICS era crear un sistema operativo de tiempo compartido que pudiera ser utilizado por varios usuarios simultáneamente y ofrecer servicios avanzados, como el procesamiento de archivos, el manejo de bases de datos y el procesamiento de informes. A pesar de sus avances, MULTICS tuvo problemas financieros y técnicos que retrasaron su desarrollo y limitaron su adopción. Aun así, muchas de las características y conceptos de MULTICS se convirtieron en estándar en la industria de los sistemas ope-

Figura 1.8: NLS (oN-Line System)



rativos modernos, como el manejo de permisos, el sistema de archivos y la seguridad.

Tras haber participado en el desarrollo de MULTICS, 🧑 Ken Thompson y 🧑 Dennis Ritchie iniciaron en 1970 la creación de un nuevo sistema operativo para la computadora DEC PDP-7. El proyecto fue bautizado originalmente como UNICS e inicialmente no tuvo apoyo económico por parte de los laboratorios Bell. Al año siguiente, Dennis Ritchie desarrolla el lenguaje de programación C, un sistema diseñado para ser eficiente en términos de tiempo de ejecución y uso de recursos, y fácil de portar a diferentes plataformas de hardware.



Los sistemas operativos existentes hasta el momento eran propietarios y solo funcionaban en una plataforma específica, es por esto que en 1972, Ken Thompson y Dennis Ritchie decidieron reescribir el código de UNICS pero esta vez en lenguaje C, dando así origen a UNIX. Este cambio significaba que UNIX podría ser fácilmente modificado para funcionar en otras computadoras y así otras variaciones podían ser desarrolladas por otros programadores. Ahora, el código era más conciso y compacto, lo que se tradujo en un aumento en la velocidad de desarrollo de UNIX.


En 1973 en el Xerox PARC (Palo Alto Research Center), se desarrolló el  Xerox Alto: un ordenador de tipo personal, con una interfaz gráfica de usuario, soporte para ventanas y un mouse. Además, Alto también tenía un sistema de gestión de archivos, un procesador de texto, una herramienta de dibujo y soporte para redes de computadoras. Aunque el Xerox Alto nunca fue comercializado, muchas de sus características y conceptos se convirtieron en estándar en la industria de los ordenadores personales.

Figura 1.9: Xerox Alto



1.4 PRIMERAS COMPUTADORAS PERSONALES


La  Altair 8800 fue una de las primeras computadoras personales en ser comercializadas, lanzada en 1974. Fue un gran éxito de ventas y sentó las bases para el desarrollo de las computadoras personales que conocemos hoy en día. La Altair 8800 se vendía en kit y los usuarios debían armarla ellos mismos.

Figura 1.10: Altair 8800



En los primeros años del sistema Unix los Laboratorios Bell autorizaron a las universidades, a utilizar el código fuente y adaptarlo a sus necesidades. A partir de dicha iniciativa, en 1977 nace en la universidad de Berkley el sistema operativo BSD (Berkeley Software Distribution). Sus principales contribuciones fueron la implementación de mejoras significativas en el sistema de archivos y en la red.

La IBM PC fue una de las computadoras más importantes en la historia de la informática, ya que sentó las bases para el estándar de computadora personal que se utiliza en la actualidad. Fue introducida por IBM en 1981 y se convirtió rápidamente en el estándar de la industria para las computadoras personales. Una de las características más importantes de la IBM PC fue su arquitectura abierta. A diferencia de otras computadoras personales de la época, la IBM PC tenía un diseño abierto que permitía a los usuarios y terceros desarrollar sus propios productos y programas para ella. Esto ayudó a impulsar un gran ecosistema de desarrolladores y fabricantes de periféricos que crearon una gran variedad de software y hardware para la computadora.

Figura 1.11: IBM PC



System V es una versión del sistema operativo UNIX desarrollado por Bell Labs en 1983. Fue una de las primeras versiones de UNIX en ser comercializada y distribuida ampliamente, y tuvo un gran impacto en el desarrollo de los sistemas operativos tipo UNIX. Aunque UNIX se convirtió en un estándar en la industria de la computación, no se consideraba «libre» debido a las restricciones en su uso y distribución impuestas por AT&T.


Ese mismo año fue fundado el movimiento GNU por 🧑 *Richard Stallman*, un programador y defensor de la libertad de software. El objetivo principal del movimiento GNU es desarrollar un sistema operativo completo y gratuito basado en el estándar UNIX, de manera que cualquier persona pueda usar, estudiar, compartir y modificar el software sin restricciones.




! Observación

El movimiento de software privativo había sido fundado con anterioridad en 1976 por Bill Gates a través de la denominada «**W** carta abierta a los aficionados».

X Windows es un sistema de ventanas para sistemas tipo UNIX. Fue desarrollado en el Massachusetts Institute of Technology (MIT) en 1984. El objetivo principal de X Windows era proporcionar un sistema de ventanas que pudiera ser utilizado en una variedad de sistemas, permitiendo una interfaz gráfica de usuario (GUI).

Minix fue un sistema operativo educativo desarrollado en el año 1987 por  *Andrew S. Tanenbaum*, para enseñar principios de diseño y funcionamiento de sistemas operativos a estudiantes universitarios. Originalmente diseñado para ser utilizado en computadoras IBM PC y compatibles, Minix tenía un diseño similar al de UNIX, pero con un conjunto reducido de herramientas y utilidades.



Mientras estudiaba informática en la Universidad de Helsinki en 1991,  *Linus Torvalds* desarrolla un núcleo de sistema operativo como proyecto personal, basándose en el diseño de Minix. Hasta el momento el proyecto GNU había desarrollado una amplia gama de software, incluyendo un compilador, un intérprete de línea de comandos y diversas herramientas de programación. Sin embargo, faltaba un kernel, que es la parte del sistema operativo que administra los recursos del sistema, como la memoria y los procesos.



En resumen, Linux surgió como un proyecto personal de Linus Torvalds, pero con el tiempo se convirtió en una parte fundamental del proyecto GNU y en uno de los sistemas operativos de código abierto más utilizados y respetados de la industria tecnológica.

ARQUITECTURA DE LA COMPUTADORA

2.1 GABINETE

El gabinete de la PC es una carcasa que cubre y protege los componentes de una computadora. Sus principales funciones son:

PROTECCIÓN El gabinete protege los componentes de la computadora de daños físicos, polvo y otros factores ambientales que pueden dañarlos.

ORGANIZACIÓN Los gabinetes de PC están diseñados para mantener todos los componentes en su lugar y organizados de manera eficiente. Esto hace que sea más fácil para el usuario trabajar en la computadora y realizar mejoras o reparaciones.

REFRIGERACIÓN: Los gabinetes de PC también ayudan a mantener los componentes frescos mediante la circulación de aire a través de la carcasa. Muchos gabinetes tienen ventiladores y otros sistemas de enfriamiento integrados para evitar el sobrecalentamiento de la computadora.

Figura 2.1: Gabinete Phobos Tg Xtech



2.2 PLACA MADRE

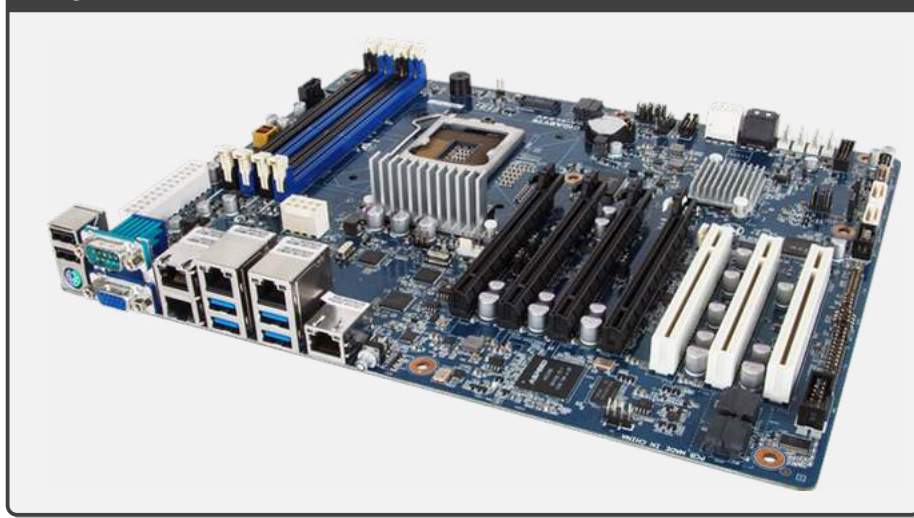
La placa madre es la pieza central de una computadora, encargada de conectar y comunicar todos los componentes esenciales del sistema. A través de sus conectores, la placa madre une la CPU, la memoria RAM, las unidades de almacenamiento, la tarjeta gráfica y otros dispositivos.

Además, la placa madre distribuye la energía eléctrica necesaria a todos los componentes a través de los conectores de alimentación, y controla los puertos de entrada/salida que permiten la comunicación de la computadora con dispositivos externos, como los puertos USB, de audio y de red.

La placa madre también incluye un chip de memoria ROM donde se almacena la BIOS, un programa que se encarga de configurar la computadora al encenderla y realizar pruebas iniciales del hardware.

La mayoría de las placas madre tienen un chip de audio integrado que proporciona capacidades de audio.

Figura 2.2: Placa madre



2.3 FUENTE DE ALIMENTACIÓN

La PSU (Power Supply Unit) o fuente de alimentación se encarga de convertir la corriente eléctrica de la toma de corriente en la energía eléctrica necesaria para alimentar los componentes internos de la computadora.

Esta recibe la corriente eléctrica de la toma de corriente a través del cable de alimentación y la convierte en diferentes voltajes que son suministrados a los componentes de la computadora.

La PSU también protege los componentes de la computadora de sobretensiones, cortocircuitos y otros problemas eléctricos que pueden ocurrir. En caso de que se detecte una sobrecarga o falla, la PSU puede cortar el suministro de energía para proteger los componentes de la computadora.

Figura 2.3: Fuente de Alimentación



2.4 MICROPROCESADOR

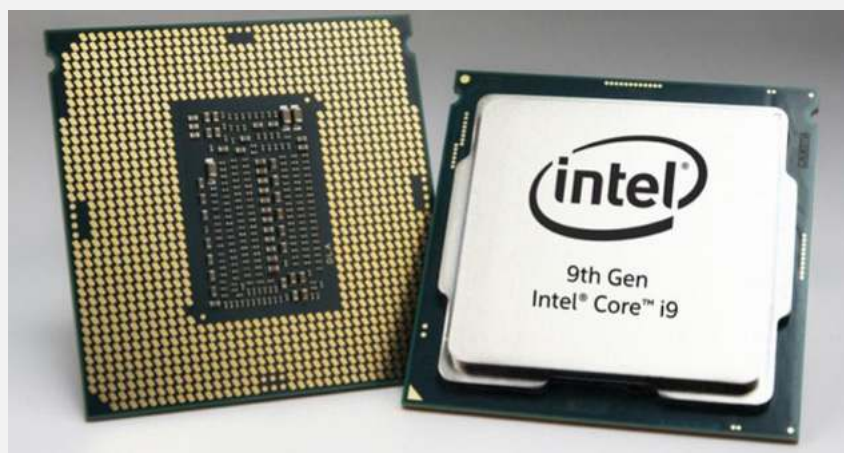
La CPU (Unidad Central de Procesamiento) es el componente principal de una computadora que realiza la mayoría de las operaciones de procesamiento de datos. Es un chip integrado que se coloca en el zócalo de la placa madre y está compuesto por varios núcleos (o cores) que trabajan en conjunto para ejecutar instrucciones y procesar datos.

La CPU es responsable de procesar y ejecutar los programas de software, manejar la entrada y salida de datos, y controlar los componentes del sistema, como la memoria RAM, el disco duro y las tarjetas de expansión. La velocidad y la capacidad de la CPU son factores clave que determinan el rendimiento

general de una computadora.

Algunos procesadores modernos tienen gráficos integrados en su diseño. Estos gráficos integrados se denominan iGPU (unidad de procesamiento de gráficos integrados) y están diseñados para proporcionar capacidades gráficas básicas para aplicaciones informáticas y de juegos de baja exigencia.

Figura 2.4: Intel Core i9



2.5 MEMORIA RAM

La memoria RAM (Random Access Memory o Memoria de Acceso Aleatorio) es un tipo de memoria que se utiliza en las computadoras para almacenar temporalmente los datos y programas que están en uso. La RAM es un componente clave en el rendimiento general de una computadora, ya que proporciona un acceso rápido y aleatorio a los datos y programas que el procesador necesita para operar.

Cuando una aplicación o un programa se ejecuta en la computadora, los datos y las instrucciones necesarios se cargan en la memoria RAM desde el disco duro. La RAM permite que el procesador acceda rápidamente a estos datos y programas, lo que acelera el tiempo de ejecución y la velocidad de la computadora en general.

Además, la memoria RAM es una memoria *volátil*, lo que significa que pierde todos los datos almacenados en ella cuando se apaga la computadora. Por lo tanto, es importante guardar los archivos y datos importantes en el disco duro o en otro dispositivo de almacenamiento persistente.


! Observación

Aunque el microprocesador también tiene una memoria volátil llamada *registros*, estos son mucho mas caros y en consecuencia pequeños.

Figura 2.5: Memoria RAM Corsair Vengeance DDR4



2.6 MEMORIA SECUNDARIA

Un  disco duro es un dispositivo de almacenamiento de datos magnético que se utiliza en las computadoras para almacenar permanentemente archivos y programas. A diferencia de la memoria RAM, que es una memoria volátil y pierde todos los datos almacenados en ella cuando la computadora se apaga, el disco duro mantiene los datos almacenados incluso después del apagado de la computadora.

Una de las principales diferencias entre el disco duro y la memoria RAM es la velocidad. La memoria RAM proporciona un acceso rápido y aleatorio a los datos y programas, lo que permite al procesador acceder a ellos rápidamente. En comparación, los discos duros son mucho más lentos en términos de velocidad de acceso, ya que el brazo de lectura/escritura necesita moverse físicamente para acceder a los datos en los platos.

Figura 2.6: Disco rígido Seagate Barracuda 1TB



Una SSD (Solid State Drive) es un dispositivo de almacenamiento de datos que utiliza memoria flash para almacenar permanentemente archivos y programas en la computadora. A diferencia de un disco duro tradicional, que utiliza platos magnéticos giratorios y cabezas de lectura/escritura para acceder a los datos, una SSD no tiene partes móviles y utiliza chips de memoria flash para almacenar y acceder a los datos.

La tecnología SSD es más rápida que la de un disco duro porque no hay partes mecánicas que necesiten moverse para acceder a los datos. En lugar de eso, los datos se almacenan en chips de memoria flash, que son mucho más rápidos para acceder y leer que los discos duros. Como resultado, las SSD proporcionan un mejor rendimiento en términos de velocidad de lectura/escritura y tiempo de acceso.

Figura 2.7: SSD Kingston



2.7 PLACA DE VIDEO

Una placa de video, también conocida como tarjeta gráfica, es un componente de hardware de la computadora que tiene como objetivo procesar y generar imágenes en la pantalla. Su función es liberar a la CPU (unidad central de procesamiento) de la computadora de la tarea de procesamiento gráfico, lo que permite que la CPU se concentre en otras tareas.

! Observación

Además de su uso en gráficos, en el campo de la inteligencia artificial son comúnmente utilizadas para entrenar y ejecutar redes neuronales profundas. Esto se debe a que las placas de video tienen una arquitectura altamente paralela que les permite procesar grandes cantidades de datos de manera eficiente. Como resultado, las placas de video son ideales para el procesamiento masivo de datos que se requiere en la inteligencia artificial.

Figura 2.8: Tarjeta Gráfica NVIDIA RTX 2080 Ti



SISTEMAS OPERATIVOS

3.1 PROCESO DE ARRANQUE

Durante el arranque de una PC, ocurren varias cosas importantes que permiten que el sistema operativo se inicie correctamente y la computadora esté lista para su uso.

Al presionarse el botón de arranque se activa la fuente de alimentación de la computadora, la cual suministra la energía necesaria para que la placa madre comience a funcionar. A partir de aquí, comienza un proceso que consiste en varias etapas:

POST (POWER ON SELF TEST) La placa madre realiza un autodiagnóstico para verificar que todos los componentes de hardware de la computadora estén funcionando correctamente. Si detecta algún problema, emitirá un mensaje de error y detendrá el proceso de arranque.

Figura 3.1: Etapa de Autodiagnóstico





BOOT LOADER A continuación la placa madre debe cargar un programa llamado «*cargador de arranque*». El cargador de arranque es un programa cuyo objetivo principal es cargar el núcleo del sistema operativo. El cargador de arranque mas utilizado en linux es « GRUB»; y « *bootmgr*» es el proporcionado por los sistemas modernos de Windows.


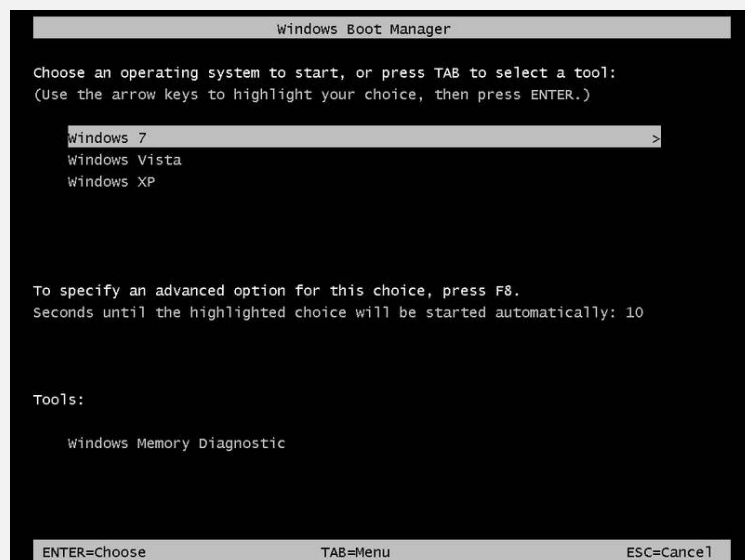
Figura 3.2:  GRUB



Figura 3.3:  bootmgr



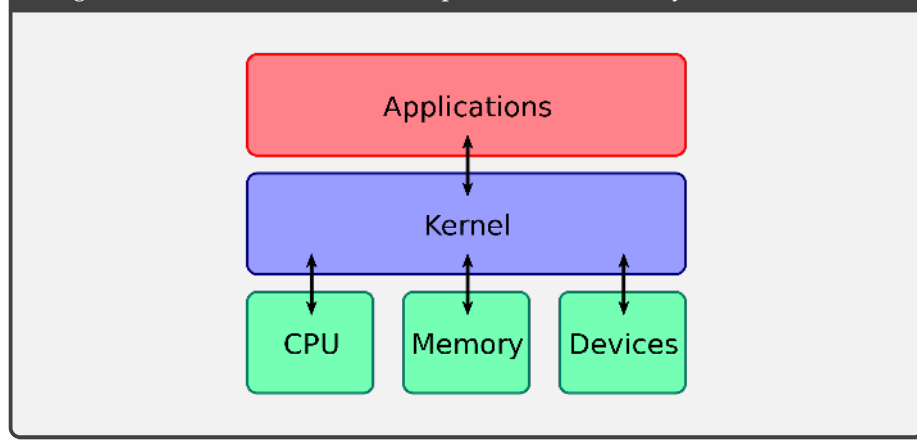
NÚCLEO A continuación, el núcleo del sistema operativo toma el control de la computadora. Durante sus tareas iniciales se encargará de identificar el hardware disponible, cargar los controladores necesarios y montar el sistema de archivos del sistema. Finalmente dará comienzo al primer programa de usuario, a partir del cual se ejecutaran todos los demás programas.

INIT En los sistemas Linux, el programa inicial del sistema operativo se llama «*init*». Init se encargará de cargar los scripts de arranque del sistema, así como también ejecutar los servicios esenciales para el funcionamiento del mismo, y proveer al usuario de un entorno gráfico o de línea de comandos.

3.2 DESCRIPCIÓN

Un sistema operativo es un conjunto de programas y herramientas que controlan y coordinan las actividades de una computadora o dispositivo electrónico, y permiten a los usuarios interactuar con el hardware y el software de manera sencilla y eficiente. Está compuesto por un núcleo (o *kernel*) que tiene control completo sobre el hardware en el que corre, y una serie de programas utilitarios que se comunican con el.

Figura 3.4: Comunicación entre aplicaciones, núcleo y hardware



Las funciones principales de un sistema operativo incluyen:

- Gestionar el hardware: El sistema operativo es responsable de gestionar el hardware de la computadora, como el procesador, la memoria, el disco duro, la tarjeta gráfica, entre otros. Controla cómo se utilizan estos recursos y asigna la cantidad adecuada de memoria y procesador a cada aplicación.
- Proporcionar una interfaz de usuario: El sistema operativo proporciona una interfaz de usuario que permite a los usuarios interactuar con el ordenador y ejecutar aplicaciones y programas.
- Gestionar los archivos y directorios: El sistema operativo se encarga de gestionar los archivos y directorios del ordenador, lo que permite a los usuarios crear, modificar, copiar y eliminar archivos y carpetas.

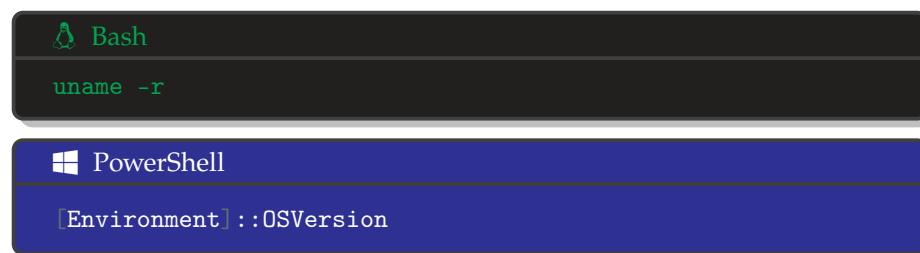
Entre las principales características de un sistema operativo se encuentran:

- Multitarea: Un sistema operativo permite que varias aplicaciones se ejecuten al mismo tiempo y asigna los recursos necesarios para que funcionen correctamente.
- Multiusuario: Un sistema operativo puede ser utilizado por varios usuarios al mismo tiempo y garantiza que cada usuario tenga sus propios archivos y configuraciones.
- Portabilidad: Los sistemas operativos pueden ser instalados en diferentes tipos de hardware, lo que los hace altamente portables.

3.3 NÚCLEO

El *kernel* (o núcleo) de un sistema operativo es la parte central y más fundamental del mismo. Es responsable de controlar el acceso a los recursos del hardware, gestionar los procesos, la memoria y la entrada/salida, y proporcionar una interfaz para que las aplicaciones interactúen con el hardware del sistema.

Podemos consultar cual es el núcleo que se esta ejecutando con el comando:



El núcleo se ejecuta en modo privilegiado, lo que significa que tiene acceso directo al hardware y puede ejecutar instrucciones que otros programas no pueden.

Para casi cualquier tarea las aplicaciones de usuario necesitan pedirle permiso al kernel, a través de una instrucción denominada «**W** llamada a sistema». Cuando se produce una llamada a sistema el CPU deja de ejecutar el programa, y comienza a ejecutar la funcionalidad del núcleo requerida, luego de la cual se continua con la ejecución del programa.

3.4 TERMINAL

A menudo se utilizan términos como «terminal», «consola virtual», «emulador de terminal» o «intérprete de línea de comandos» de forma indistinta, lo que puede llevar a cierta confusión. A continuación, se explican las diferencias entre estos términos:

TERMINAL Se refiere a el o los dispositivos físicos que se utilizan para interactuar con un ordenador mediante la entrada y salida de texto. En la actualidad está compuesta principalmente por el teclado y el monitor.

CONSOLA VIRTUAL Es una aplicación implementada dentro del núcleo que provee acceso al sistema simulando una terminal de teletipo. En los sistemas tipo Unix se puede acceder a ellas presionando Ctrl + Alt + F1, Ctrl + Alt + F2, etc.

Figura 3.5:  Consola virtual en Ubuntu

```
Ubuntu 18.04 ubuntu tty1
ubuntu login: Ubuntu
Password:
Welcome to Ubuntu 18.04 (GNU/Linux 4.15.0-23-generic)

 * Documentation:  https://help.ubuntu.com/

278 packages can be updated.
71 updates are security updates.

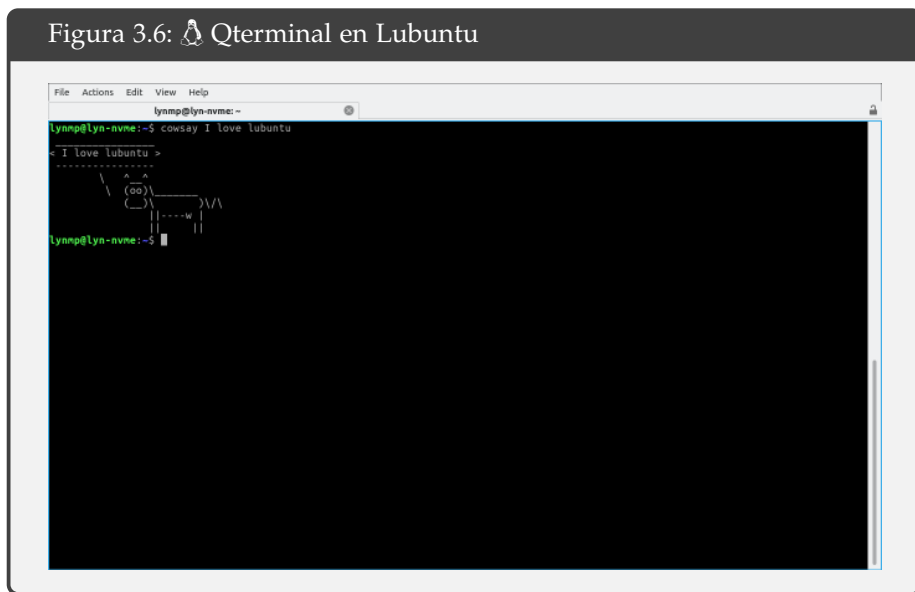
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

Ubuntu@ubuntu:~$
```

EMULADOR DE TERMINAL Es un programa de usuario que permite interactuar con un sistema operativo a través de una ventana en un entorno gráfico. Los emuladores de terminal son comúnmente utilizados para acceder a sistemas remotos o para ejecutar aplicaciones de línea de comandos en sistemas operativos.

Figura 3.6:  Qterminal en Lubuntu



Bash

Para saber que emulador de terminal se está utilizando se puede escribir el comando:

```
echo $TERM
```

SHELL También llamado «interprete de linea de comandos», es un programa que permite a un usuario interactuar con el sistema operativo mediante la ejecución de comandos a través de una interfaz de línea de comandos.

Bash

Para saber que interprete de linea de comandos se está utilizando se puede escribir el comando:

```
echo $SHELL
```

3.5 INTERFAZ GRÁFICA

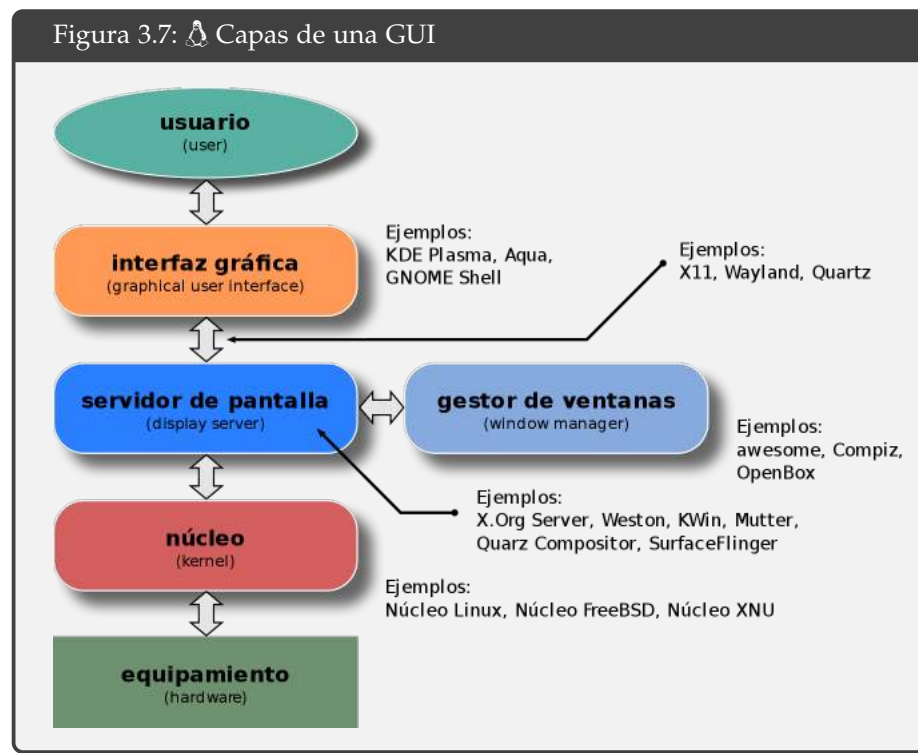
Una interfaz gráfica de usuario es una forma de interactuar con un programa o sistema operativo mediante el uso de elementos gráficos, como ventanas, iconos, botones y menús, en lugar de usar comandos de texto en una línea de comandos.

! Observación

A las interfaces de usuario gráficas las llamamos «GUI» por sus siglas en inglés «*Graphical User Interface*»; en cambio a las interfaces de texto las llamamos «CLI» por las siglas «*Command Line Interface*».

El entorno de escritorio, el sistema de ventanas, el servidor de pantalla y el gestor de ventanas son componentes importantes de un sistema operativo gráfico que trabajan juntos para proporcionar una interfaz de usuario intuitiva y fácil de usar.

Figura 3.7: Capas de una GUI



A continuación, se describen las funciones de cada uno de ellos:

ENTORNO DE ESCRITORIO Es un conjunto de aplicaciones, herramientas y utilidades que proporcionan una interfaz de usuario gráfica para un sistema operativo. El entorno de escritorio incluye menús, barras de herramientas, iconos, fondos de pantalla, gestores de archivos y otras herramientas que hacen que el uso del sistema operativo sea más fácil e intuitivo para el usuario. Algunos ejemplos de entornos de escritorio son GNOME, KDE, XFCE y LXDE.

Figura 3.8:  Unity en Ubuntu 22.10

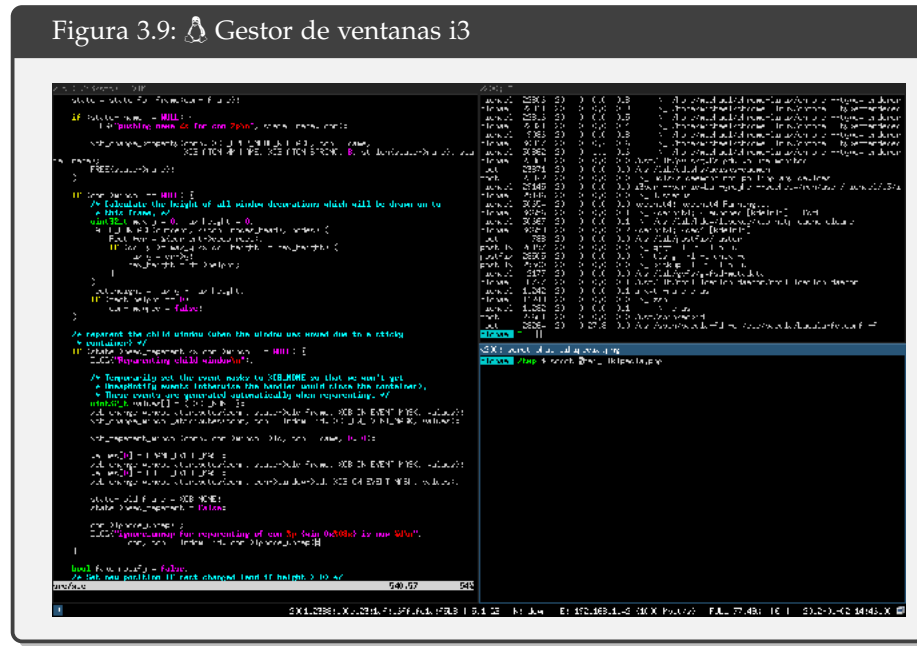


SISTEMA DE VENTANAS Es un sistema que permite la creación y manipulación de ventanas de aplicaciones en la pantalla. El sistema de ventanas se encarga de administrar la posición, tamaño, apariencia y eventos de las ventanas en la pantalla. También se encarga de la gestión de los recursos gráficos, como el uso de la memoria, la gestión de la entrada y salida de datos, y el manejo de la interacción entre aplicaciones. Algunos ejemplos de sistemas de ventanas son X11 y Wayland.

SERVIDOR DE PANTALLA Es un programa que se ejecuta en el sistema operativo y se encarga de controlar la pantalla, el teclado y el ratón del sistema. El servidor de pantalla recibe la entrada de teclado y ratón y la envía a las aplicaciones en ejecución en el sistema. También se encarga de mostrar la salida gráfica de las aplicaciones en la pantalla. El servidor de pantalla más utilizado en Linux es Xorg.

GESTOR DE VENTANAS Es un programa que se ejecuta en el entorno de escritorio y que se encarga de administrar la apariencia y el comportamiento de las ventanas de las aplicaciones. El gestor de ventanas proporciona una variedad de características, como la decoración de ventanas, la administración de escritorios virtuales, la configuración de atajos de teclado, y la gestión de la colocación de ventanas en la pantalla. Algunos ejemplos de gestores de ventanas son Metacity, KWin, Compiz, Openbox y i3.

Figura 3.9:  Gestor de ventanas i3



3.6 DISTRIBUCIONES

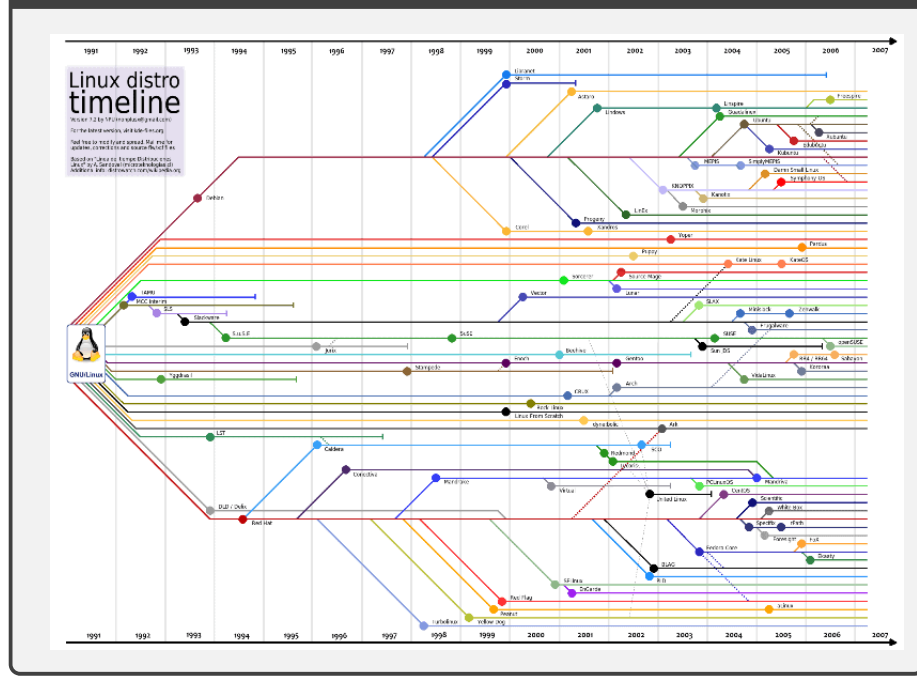
Las distribuciones de Linux son sistemas operativos basados en el kernel de Linux, que están compuestos por una combinación de software libre y de código abierto, como aplicaciones, controladores, herramientas de gestión de paquetes, etc.

Existen muchas distribuciones de Linux diferentes, como Debian, Ubuntu, Fedora, CentOS, Arch Linux, entre otras. Cada distribución tiene sus propias características, objetivos y filosofía, y están diseñadas para satisfacer las necesidades de diferentes usuarios y aplicaciones.

Las distribuciones de Linux existen porque el software de código abierto permite a los usuarios y desarrolladores acceder, modificar y distribuir el código fuente del software. Esto ha permitido que muchas personas y comunidades puedan desarrollar y distribuir sus propias versiones personalizadas

de Linux. Además, al ser un sistema operativo altamente personalizable y adaptable, cada distribución puede estar diseñada para satisfacer las necesidades específicas de diferentes usuarios, como por ejemplo para usuarios de servidores, programadores, usuarios de escritorio, entre otros.

Figura 3.10: 🐧 Línea de tiempo de distribuciones de Linux



3.7 PROCESO DE APAGADO

Antes de finalizar la ejecución del sistema operativo, se inicia un proceso que cierra todos los programas y servicios que se están ejecutando en la computadora. Luego, se guardan todos los datos pendientes y se asegura que todos los dispositivos de almacenamiento, como los discos duros o las unidades flash USB, estén en un estado seguro antes de apagar la alimentación. Finalmente, se envía una señal al hardware para que se apague por completo y se desconecte la alimentación en caso de ser necesario.

Hay varias opciones disponibles para «apagar» una computadora, cada una con diferentes efectos.

APAGADO Cuando se selecciona la opción de «apagar», la computadora cierra todos los programas y procesos en ejecución y se apaga completamente. La próxima vez que se encienda la computadora, se iniciará el proceso de arranque completo.

```
Bash
shutdown

PowerShell
shutdown /s
```

REINICIO Al seleccionar la opción «reiniciar», la computadora cierra todos los programas y procesos en ejecución, se apaga brevemente y luego se reinicia automáticamente. Esta opción es útil para solucionar problemas de hardware o software y para actualizar el sistema operativo.

```
Bash
reboot

PowerShell
shutdown /r
```

HIBERNACIÓN La opción de «hibernar» guarda todos los datos y configuraciones del sistema en el disco duro y luego apaga la computadora. Cuando se vuelve a encender la computadora, el sistema restaura automáticamente los datos y la configuración de la sesión anterior.

```
Bash
systemctl hibernate

PowerShell
shutdown /h
```

SUSPENSIÓN La opción «suspender» pone la computadora en un estado de bajo consumo de energía, dejando alimentada solamente la memoria RAM. De esta manera los programas y procesos en ejecución se conservan y la computadora puede volver a su estado anterior cuando se reanuda la actividad. Esta opción es útil para ahorrar energía y reanudar rápidamente el trabajo en curso.

 Bash

```
systemctl suspend
```

! Observación

Si se llega a producir un corte en el suministro eléctrico, el estado de la computadora se pierde pues la RAM será incapaz de retener su información.

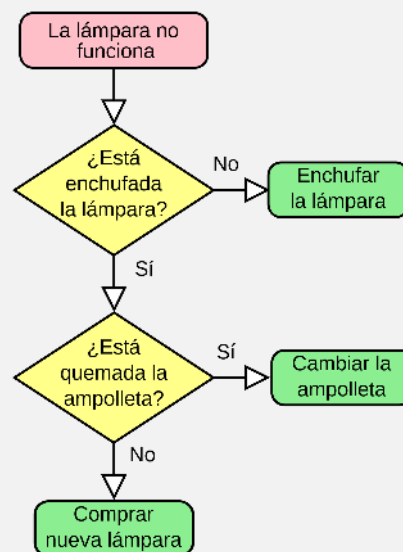
CONCEPTOS DE PROGRAMACIÓN

4.1 ¿QUE ES LA PROGRAMACIÓN?

La programación es el proceso de diseñar, escribir, probar y mantener el código informático. En términos más simples, la programación es la forma en que los desarrolladores crean software, aplicaciones y sistemas.

El proceso de programación implica varios pasos, que incluyen la definición del problema que se está tratando de resolver, la identificación de los requisitos y especificaciones necesarios para crear la solución, la escritura del código en un lenguaje de programación, la realización de pruebas para asegurarse de que el software funciona correctamente y la documentación del código para que otros desarrolladores puedan entender y trabajar en él.

Figura 4.1: Diagrama del flujo de un programa



4.2 LENGUAJE DE PROGRAMACIÓN

Un lenguaje de programación es un conjunto de reglas, símbolos y palabras clave que se utilizan para comunicar instrucciones a una computadora. Estos lenguajes permiten a los programadores crear software, aplicaciones y sistemas, al proporcionar un medio para escribir código en un formato que la máquina pueda entender y ejecutar.

Los lenguajes de programación se utilizan para describir las acciones que una computadora debe realizar, como realizar cálculos matemáticos, interactuar con dispositivos periféricos, almacenar y recuperar datos, y responder a las entradas del usuario. Los lenguajes de programación pueden variar en complejidad y enfoque, y algunos se centran en tareas específicas, mientras que otros son más generales y versátiles.

En esencia, un lenguaje de programación es una herramienta para crear software y sistemas informáticos, permitiendo a los programadores transformar sus ideas en código ejecutable.

! Observación

En esta materia aprenderemos el lenguaje de programación de «bash». A lo largo de la carrera también aprenderán el lenguaje «python».

4.3 NIVELES DE LENGUAJES

Los lenguajes de programación se pueden clasificar en tres niveles: alto, bajo y binario.

ALTO Estos lenguajes son más cercanos al lenguaje natural humano y se utilizan para escribir programas complejos de manera más fácil y rápida. Ejemplos de lenguajes de programación de alto nivel incluyen Python, Java, C++, Ruby, PHP, entre otros. Estos lenguajes tienen una sintaxis más amigable para el programador y se encargan de muchos de los detalles de bajo nivel, como la administración de memoria y el manejo de errores.

</> Código

```
print("Hola mundo!")
```

BAJO Estos lenguajes están más cerca del lenguaje de la máquina y se utilizan para escribir programas que interactúan directamente con el hardware del ordenador. Ejemplos de lenguajes de programación de bajo nivel

incluyen el lenguaje ensamblador y el lenguaje C. Estos lenguajes requieren que el programador tenga un conocimiento más detallado del hardware y de la forma en que se maneja la memoria.

</> Código

```
.global _start          # gcc -nostdlib -no-pie hola.s
.text

mensaje: .ascii "Hola mundo!\n"
_start:
    mov $1, %rax        # La llamada a sistema 1 es write
    mov $1, %rdi        # El descriptor 1 es la salida estandar
    mov $mensaje, %rsi  # Direccion de memoria del mensaje
    mov $12, %rdx       # Cantidad de bytes
    syscall             # write(1, mensaje, 12)

    mov $60, %rax       # La llamada a sistema 60 es exit
    mov $0, %rdi        # Queremos devolver el numero 0
    syscall             # exit(0)
```

BINARIO El lenguaje binario es el lenguaje de máquina utilizado por los ordenadores para ejecutar programas. Este lenguaje está compuesto de ceros y unos, que representan instrucciones que el procesador del ordenador puede entender y ejecutar directamente. Es muy difícil y tedioso para los programadores escribir directamente en lenguaje binario, por lo que se utilizan los lenguajes de programación de nivel alto y bajo para crear programas que luego se traducen a lenguaje binario.

</> Código

```
401000: 48 6f          # H o
401002: 6c            # l
401003: 61            # a
401004: 20 6d 75      # m u
401007: 6e            # n
401008: 64 6f         # d o
40100a: 21 0a         # ! \n
40100c: 48 c7 c0 01 00 00 00 # mov $1, %rax
401013: 48 c7 c7 01 00 00 00 # mov $1, %rdi
40101a: 48 c7 c6 00 10 40 00 # mov $mensaje, %rsi
401021: 48 c7 c2 0c 00 00 00 # mov $12, %rdx
401028: 0f 05         # syscall
40102a: 48 c7 c0 3c 00 00 00 # mov $60, %rax
401031: 48 c7 c7 00 00 00 00 # mov $0, %rdi
401038: 0f 05         # syscall
```

4.4 COMPILADORES E INTERPRETES

Un intérprete y un compilador son dos tipos de programas que se utilizan para convertir el código fuente escrito por un programador en instrucciones ejecutables por una computadora.

Un intérprete es un programa que lee el código fuente de un programa y lo traduce en instrucciones ejecutables en tiempo real. El intérprete lee una línea de código fuente, la traduce a lenguaje de máquina y la ejecuta antes de pasar a la siguiente línea. Debido a que el intérprete realiza la traducción y la ejecución de cada línea a medida que se lee el código, puede ser más lento que un compilador. Sin embargo, el intérprete tiene la ventaja de que el programador puede ejecutar el código directamente sin necesidad de compilarlo previamente.

Un compilador, por otro lado, es un programa que traduce todo el código fuente a lenguaje de máquina en un solo paso antes de su ejecución. El compilador lee todo el código fuente del programa y lo traduce en un archivo ejecutable que puede ser utilizado por la computadora sin necesidad de leer el código fuente original nuevamente. Debido a que el código fuente se traduce y compila solo una vez, el código compilado se ejecuta generalmente más rápido que el código interpretado. Sin embargo, el proceso de compilación puede llevar más tiempo que el proceso de interpretación.

4.5 OTROS CONCEPTOS

5

SISTEMA DE ARCHIVOS

Parte II

MANEJO DE TERMINAL

5.1 COMANDOS BÁSICOS

5.2 SHELL SCRIPTING

Parte III

CONCEPTOS ADICIONALES

5.3 CONTENEDORES

5.4 CONTROL DE VERSIONES