**TP 1**

Los Sistemas operativos son programas para la administración eficiente de los recursos de la computadora.

Un programa que actúa como un intermediario entre un usuario de una computadora y el *hardware* de la misma.

Es un intermediario entre el usuario de una computadora y su hardware.

● Proporciona un entorno en el cual los usuarios puedan ejecutar programas de una manera práctica y eficiente.

● Es software que administra el hardware de una computadora.

● Provee las bases para los programas de aplicación.

Otra definición común es: un sistema operativo es aquel programa que se ejecuta en todo momento en la computadora (kernel). Todo lo demás son: ○ Programas del sistema (viene con el sistema operativo, pero no es parte del kernel), o ○ Programas de aplicación (todos los programas que no están asociados con el sistema operativo).

Objetivos del Sistema Operativo:

– Ejecutar los programas de usuario y permitir la

solución de problemas del usuario más

fácilmente.

– Conveniencia de uso del sistema de la computadora.

Uso del hardware de la computadora de manera eficiente.

1.3)

La internet de las cosas (IoT, por sus siglas en inglés) es un sistema de dispositivos de computación interrelacionados, máquinas mecánicas y digitales, objetos, animales o personas que tienen identificadores únicos [y la capacidad de transferir datos a través de una red](https://searchdatacenter.techtarget.com/es/opinion/La-IoT-esta-en-su-etapa-de-maduracion-su-red-ya-esta-lista), sin requerir de interacciones humano a humano o humano a computadora.

1.4)Es necesario en el caso de que el sistema operativo tenga soporte periódicamente, sino se puede volver lento.

1.5)

**Batch(lote):** Comando que se utiliza para poner en cola tareas para posterior ejecución.

Se conoce como **sistema por lotes** (en inglés **batch processing**), o modo *batch*, a la ejecución de un programa sin el control o supervisión directa del usuario (que se denomina [procesamiento interactivo](https://es.wikipedia.org/wiki/Procesamiento_interactivo)). Este tipo de programas se caracterizan porque su ejecución no precisa ningún tipo de interacción con el usuario.

**Interactivo**: Responde casi inmediatamente a los estímulos proporcionados por el usuario.

Tiempo compartido: Uso del sistema por dos o más personas al mismo tiempo

Tiempo real: Procesa las instrucciones recibidas al instante para poder mostrar un resultado.

**Red(network):**  Permite la interconexión de ordenadores para tener el poder de acceder a los [servicios](https://es.wikipedia.org/wiki/Servicio_de_red) y [recursos](https://es.wikipedia.org/wiki/Compartición_de_archivos), *hardware* y *software*, creando [redes de computadoras](https://es.wikipedia.org/wiki/Red_de_computadoras). Al igual que un equipo no puede trabajar sin un sistema operativo, una red de equipos no puede funcionar sin un sistema operativo de red. Consiste en un *software* que posibilita la comunicación de un [sistema informático](https://es.wikipedia.org/wiki/Sistema_informático) con otros equipos en el ámbito de una [red](https://es.wikipedia.org/wiki/Red_informática).

**Distribuido**: Es la unión lógica de sistemas operativos sobre una colección de nodos computacionales independientes. Conectados en red, comunicándose y físicamente separado**s.**

**Cluster:** Un cluster comparte almacenamiento y está vinculado a través de una red de área local (LAN).  Se aplica a los conjuntos o conglomerados de [ordenadores](https://es.wikipedia.org/wiki/Ordenador) unidos entre sí normalmente por una red de alta velocidad y que se comportan como si fuesen una única computadora.

● Se utiliza para brindar un servicio de alta disponibilidad, que continuará incluso si falla uno o más sistemas del cluster. ○ En general, se obtiene alta disponibilidad al agregar un nivel de redundancia en el sistema. ○ La alta disponibilidad proporciona mayor confiabilidad, crucial en muchas aplicaciones.

● Algunos sistemas son tolerantes a fallas, ya que pueden sufrir una falla de cualquier componente individual y seguir funcionando.

● La tolerancia a fallas requiere un mecanismo para permitir que la falla sea detectada, diagnosticada y, si es posible, corregida

**Movil**

los sistemas operativos móviles son mucho más simples y están más orientados a la conectividad inalámbrica, los formatos multimedia para móviles y las diferentes maneras de introducir información en ellos. Algunos de los sistemas operativos utilizados en los dispositivos móviles están basados en el modelo de capas.Las empresas desarrolladoras de teléfonos celulares utilizan distintos sistemas operativos que, si bien en el fondo cumplen las mismas funciones, poseen importantes diferencias.

**1.6) FALTA**

**1.7)Kernel:** es un [*software*](https://es.wikipedia.org/wiki/Software) que constituye una parte fundamental del [sistema operativo](https://es.wikipedia.org/wiki/Sistema_operativo), y se define como la parte que se ejecuta en [modo privilegiado](https://es.wikipedia.org/wiki/Modo_privilegiado) (conocido también como modo núcleo).[1](https://es.wikipedia.org/wiki/Núcleo_(informática)" \l "cite_note-what_is_gnu_en-1)​ Es el principal responsable de facilitar a los distintos programas [acceso seguro](https://es.wikipedia.org/wiki/Seguridad_informática) al [*hardware*](https://es.wikipedia.org/wiki/Hardware) de la [computadora](https://es.wikipedia.org/wiki/Computadora_electrónica) o en forma básica, es el encargado de gestionar recursos, a través de servicios de llamada al sistema. Nexo entre hw y sw.

Este reside en la memoria RAM cuando se enciende el ordenador y permanece en funcionamiento hasta que este se apaga.

Tiene principalmente dos responsabilidades:

1. Servir a los requerimientos de programación a bajo nivel, por ejemplo tratando las interrupciones hardware (teclados, discos duros, tarjetas de video, etc...).

2. Proveer un entorno a los procesos, que son las instancias en ejecución de los programas o threads.

1.8)FALTA

1.9) FALTA

1.10)

La multiprogramación aumenta la utilización de la CPU organizando los programas para que la CPU siempre tenga uno para ejecutar. En un sistema multiprogramado, un programa en ejecución se denomina proceso.

El SO mantiene varios procesos en la memoria, selecciona y comienza a

ejecutar a uno.

Es posible que el proceso deba esperar a que se complete alguna tarea.

En un sistema no multiprogramado la CPU quedaría inactiva, mientras

que en uno multiprogramado el SO simplemente cambia a otro proceso

y lo ejecuta.

Cuando ese proceso necesita esperar, la CPU cambia a otro proceso, y

así sucesivamente; finalmente el primer proceso termina de esperar y

recupera la CPU.

Mientras haya un proceso por ejecutarse, la CPU nunca estará inactiva.

1.11) FALTA

1.12)

una **interrupción** (del inglés *interrupt request*, en español «petición de interrupción») es una señal recibida por el [procesador](https://es.wikipedia.org/wiki/Unidad_central_de_procesamiento) de una [computadora](https://es.wikipedia.org/wiki/Computadora), para indicarle que debe «interrumpir» el curso de ejecución actual y pasar a ejecutar código específico para tratar esta situación.

Otra forma de interrupción es un trap (o una excepción), que es una interrupción generada por software causada por un error (por ejemplo, división por cero o acceso de memoria no válido) o por una solicitud específica de un programa de usuario de un servicio del sistema.

Esta solicitud se realiza ejecutando una operación especial denominada llamada del sistema

(system call).

1.13)FALTA

1.14)

La arquitectura SMP (también llamada **UMA**, del inglés ***U****niform* ***M****emory* ***A****ccess*, en español "acceso uniforme a memoria") se caracteriza por el hecho de que varias unidades de procesamiento comparten el acceso a la memoria, compitiendo en igualdad de condiciones por dicho acceso, de ahí la denominación "simétrico".

Los sistemas SMP permiten que cualquier procesador trabaje en cualquier tarea sin importar su localización en memoria; con un propicio soporte del [sistema operativo](https://es.wikipedia.org/wiki/Sistema_operativo), estos sistemas pueden mover fácilmente tareas entre los procesadores para garantizar eficientemente el trabajo.

Una [computadora](https://es.wikipedia.org/wiki/Computadora) SMP se compone de microprocesadores independientes que se comunican con la memoria a través de un [bus](https://es.wikipedia.org/wiki/Bus_(Informática)) compartido. Dicho *bus* es un recurso de uso común. Por tanto, debe ser arbitrado para que solamente un microprocesador lo use en cada instante de tiempo.

MULTIPROCESAMIENTO ASIMÉTRICO

1. A cada procesador se le asigna una tarea específica; el procesador maestro planifica y asigna el trabajo a los procesadores esclavos.
2. El más común en sistemas más grandes.
3. El que manda es el de mayor jerarquía.
4. Una desventaja es que se recarga todo el trabajo sobre el procesador que manda o el de mayor jerarquía

## Diferencias clave entre multiprocesamiento simétrico y asimétrico

1. El punto más distinguible entre el multiprocesamiento simétrico y asimétrico es que las tareas en el sistema operativo son manejadas solamente por el procesador maestro en el multiprocesamiento asimétrico. Por otro lado, todos los procesadores en multiprocesamiento simétrico ejecutan las tareas en el sistema operativo.
2. En el multiprocesamiento simétrico, cada procesador puede tener su propia cola privada de procesos listos, o pueden tomar procesos de una cola lista común. Pero, en el multiprocesamiento asimétrico, el procesador maestro asigna procesos a los procesadores esclavos.
3. Todo el procesador en Multiprocesamiento Simétrico tiene la misma arquitectura. Pero la estructura de los procesadores en multiprocesador asimétrico puede diferir.
4. Los procesadores en multiprocesamiento simétrico se comunican entre sí mediante la memoria compartida. Sin embargo, los procesadores en multiprocesamiento asimétrico no necesitan comunicarse entre sí, ya que están controlados por el procesador maestro.
5. En caso de que el procesador maestro falle, un procesador esclavo se convierte en procesador maestro para continuar la ejecución. Pero, si falla un procesador en multiprocesamiento simétrico, se reduce la capacidad de cálculo del sistema.
6. El multiprocesador asimétrico es simple, ya que solo el procesador maestro accede a la estructura de datos, mientras que el multiprocesador simétrico es complejo, ya que todos los procesadores necesitan trabajar en sincronización.

1.15)

Para garantizar la correcta ejecución del sistema, debe ser posible distinguir entre la ejecución del código del SO y el código definido por el usuario.

Se necesitan como mínimo dos modos de operación separados: modo de usuario y modo kernel.

Se agrega un bit, llamado bit de modo, al hardware de la computadora para indicar el modo actual:

kernel (0) o usuario (1).

El bit de modo permite distinguir entre una tarea del SO y una del usuario.

Cuando una aplicación de usuario solicita un servicio del sistema operativo a través de una

llamada del sistema, este debe pasar del modo usuario al modo kernel.

El hardware permite que se ejecuten instrucciones privilegiadas sólo en modo kernel. Si se intenta

ejecutar una instrucción privilegiada en modo usuario, el hardware no la ejecuta sino que la

considera ilegal y se produce un trap al SO.

1.16)

Definimos instrucciones privilegiadas a aquellas que pueden potencialmente dañar a otros procesos. P. ej. operaciones de E/S, actualizar el reloj, desactivar interrupciones, manipular la MMU o el bit de protección, etc.

Protegemos a los procesos y al SO si controlamos la ejecución de instrucciones privilegiadas.

1.17)

La principal diferencia entre la seguridad y la protección radica en el hecho de que la seguridad maneja las amenazas de información externa en los sistemas informáticos, mientras que la protección trata con las amenazas internas.

1.18)

Una llamada al sistema es un método o función que puede invocar un proceso para solicitar un cierto servicio al sistema operativo. Dado que el acceso a ciertos recursos del sistema requiere la ejecución de código en modo privilegiado, el sistema operativo ofrece un conjunto de métodos o funciones que el programa puede emplear para acceder a dichos recursos. En otras palabras, el sistema operativo actúa como intermediario, ofreciendo una interfaz de programación (API) que el programa puede usar en cualquier momento para solicitar recursos gestionados por el sistema operativo.

1.19)

la diferencia principal entre API y llamada al sistema es que API es un conjunto de protocolos, rutinas y funciones que permiten el intercambio de datos entre varias aplicaciones y dispositivos, mientras que una llamada al sistema es un método que permite que un programa solicite servicios al núcleo.

1.20)

Mediante las instrucciones que aporta el intérprete, el usuario puede comunicarse con el [núcleo](https://es.wikipedia.org/wiki/Núcleo_(informática)) y por extensión, ejecutar dichas órdenes, así como [herramientas](https://es.wikipedia.org/wiki/Software) que le permiten controlar el funcionamiento de la computadora.

por un lado la necesidad de simplificar las cosas al máximo (filosofía KISS: Keep It Simple, Stupid). En Unix, la máxima siempre ha sido hacer procesos o programas pequeños que funcionen muy bien.

Por otro, porque el shell como tal es algo subjetivo. No hay un único shell, sino varias implementaciones, y cada usuario escoge el que más le guste: bash, sh, ksh, fish,…

2.21)

El SO se divide en varias capas (niveles). La capa inferior (capa 0) es el HW; la más alta (capa N) es la interfaz de usuario.

La capa M consiste en estructuras de datos y un conjunto de funciones que pueden ser invocadas por capas de nivel superior. La capa M, a su vez, puede invocar operaciones en capas de nivel inferior.

**Ventajas :**

La principal ventaja del enfoque en capas es la simplicidad de construcción y depuración.

**Desventaja:**

* Es difícil definir adecuadamente la funcionalidad de cada capa.
* El rendimiento general es deficiente debido a la sobrecarga al

requerir que un programa de usuario atraviese varias capas para

obtener un servicio del SO.

2.22)

Un sistema operativo que no dispone de módulos cargables en el núcleo debe tener toda aquella funcionalidad que pueda llegar a ser necesitada precompilada dentro del núcleo base. El problema de este enfoque consiste en que en general la imagen del núcleo sería mucho mayor, ocupando un gran espacio memoria. Así mismo, resultaría necesario que los usuarios recompilaran y reiniciaran el núcleo base cada vez que se necesite agregar nueva funcionalidad al mismo.