Ingeniería en Desarrollo de software

3er semestre

Programa de la Unidad didáctica:

Programación de sistemas operativos

Unidad 4. Diseño de sistemas operativos

Clave:

Ingeniería:	TSU:
15142317	**16142317**

Ciudad de México, enero del 2025

Universidad Abierta y a Distancia de México

Índice

Unidad 4. Diseño de Sistemas Operativos

Presentación de la unidad

En esta unidad hablaremos del desarrollo que debes seguir para diseñar un sistema operativo, el cual definiremos como el cerebro de la computadora que interactúa y administra los componentes físicos, estos componentes son los que conocemos como *hardware.* El sistema operativo se encarga de interactuar con el hardware y el software. Por lo tanto, el objetivo principal de un sistema operativo es ejecutar los programas que el usuario necesita y que dicha ejecución se haga de forma eficiente en interacción con los recursos con los que cuenta la máquina.

Existen diversas problemáticas y encargos que deben contemplarse antes del diseño e implementación de sistemas operativos, pues estos no sólo deben encargarse de la administración específica de los archivos (guardar, borrar, organizar, etc.), sino que también del diseño de interfaces, donde se establece la forma, color, tamaño en la que el usuario visualiza cómo manipular los archivos y los dispositivos.

Con respecto a la **funcionalidad**, un sistema operativo debe incluir una excelente **administración** de la **memoria**; recuerda que la memoria es la parte física de la máquina que guarda y organiza la información. La administración eficiente de la memoria permitirá que el sistema operativo optimice el uso de programas del usuario.

Para que el sistema operativo que se diseñe tenga un excelente manejo de los recursos de la máquina y así supere las demandas del mercado, es menester considerar los puntos previamente señalados en esta presentación de unidad.

Logros

- Deducir los principios básicos del diseño.
- Establecer la visión y cuestiones básicas del diseño.
- Definir el diseño de interfaces.
- Formular la arquitectura y características del diseño.
- Elaborar la estructura, mecanismo para su implementación.

Competencia específica

- Aplicar la visión básica sobre el diseño, para la implementación y desempeño del Sistema Operativo, mediante el principio de diseño, con el uso de las diferentes características de la arquitectura que lo conforman.

4.1. Base del diseño de sistemas operativos

Hablar de un proyecto, que sea el desarrollo de un sistema operativo, es pensar primero de forma global sobre el proceso que se llevará acabo para el desarrollo de los detalles como la manipulación de la memoria, el manejo de dispositivos de E/S y los procesos a seguir para que realice una buena administración de archivos, sin dejar de lado la seguridad y protección con la que debe contar el nuevo sistema a desarrollar. El desarrollo de un sistema operativo es complejo, sin embargo, la etapa crítica para la elaboración del sistema es determinar la arquitectura, establecer un plan de desarrollo y determinar los riesgos del proyecto.

Con la nueva era, el desarrollo de hardware se incrementa y evoluciona y cada vez son más las necesidades de procesamiento y lo que el sistema operativo debe controlar y gestionar de forma eficiente, todos los dispositivos y recursos que contiene un equipo de cómputo.

Hoy en día, el progreso en el desarrollo de los sistemas operativos ha incrementado de forma paralela con las nuevas tecnologías para mejorar los algoritmos que manejan los procesos dentro del sistema operativo.

Las bases principales para comenzar el diseño de sistemas operativos son:

- La **abstracción**: se considera como primera base para el diseño y se refiere a la pregunta ¿**qué** hace el sistema operativo?, sin tomar en cuenta el ¿c**ómo** lo hace?; por ejemplo: para el usuario de un programa de tipo procesador de textos, no es necesario que se dé cuenta cómo funcionan el proceso de captura de datos, ni de impresión de documentos, basta con que lo pueda usar y funcione.
- **Amplitud de funciones y administrador de errores:** Como se mencionaba con anterioridad sobre el constante avance tecnológico, es conveniente aumentar las posibilidades de multiprocesamiento de los sistemas operativos, para permitir la ejecución de varias aplicaciones sin que el sistema operativo colapse; al respecto es conveniente que maneje una serie de funciones que permitan administrar los posibles errores que el mismo sistema operativo no es capaz de corregir de forma automática.

Estandarización: Actualmente existe una gran variedad de sistemas operativos y dispositivos de hardware que tienen una función específica, por lo que es necesaria una **estandarización** para tratar de establecer una comunicación entre las diferentes arquitecturas, por ejemplo: una tarjeta de video y la manera que el sistema operativo utiliza dicha tarjeta para que se visualicen las imágenes.

- **Protección al sistema operativo:** la **función principal** de un sistema operativo es establecer y definir las abstracciones correctas, es decir que funcione todo bien sin que al usuario le importe cómo lo hace. Cuando se protege **el sistema operativo**; los archivos y otras estructuras de datos también suelen protegerse y aislarse para evitar operaciones con datos no autorizados, así mismo, si los usuarios requieren compartir datos y recursos, el aislamiento debe ser bajo control del mismo. Por ejemplo: Cuando dos usuarios de sesiones diferentes en una misma máquina quieran pausar la impresora o dar de baja una tarjeta de video

que está siendo ocupada. O cuando las funciones de administración de procesos, archivos y subprocesos, como el que puedas utilizar Word, block de notas, en el caso del sistema operativo Windows, o dos archivos de Word para diferentes usuarios debido a que pueden existir múltiples sesiones al mismo tiempo en una computadora, el sistema operativo debe proporcionar mecanismos para mantenerlos separados y no deberá existir interferencia entre uno y otro, estos mecanismos son la protección al sistema operativo.

- **Control de fallos:** En lo que respecta al control de fallos, es necesario aislarlos, pues si alguna parte del sistema se cae, no deberá hacer que el resto del sistema se caiga. Por ello, durante el diseño del sistema operativo se debe tener cuidado de que las diversas partes estén lo suficientemente aisladas unas de otras, para evitar “el colapso total” del sistema operativo.

- **La administración del hardware** es una parte fundamental del sistema operativo, pues tiene que administrar todos los componentes de bajo nivel, así como las controladoras de interrupciones y las controladoras de bus. También se debe otorgar un marco en el que las controladoras de dispositivos puedan controlar los dispositivos de E/S más grandes, como discos, impresoras y otros dispositivos.

¿Por qué es tan difícil diseñar sistemas operativos?

Si bien hemos estado mencionando mucho que diseñar un sistema operativo es una labor complicada, debemos considerar que todos los sistemas operativos son fundamentalmente distintos de los pequeños programas de aplicación. Todo lo que ya se mencionó, hace que el diseño de un sistema operativo sea mucho más difícil que el de un programa de aplicación el cual solo contiene una función específica y no involucra tantas situaciones.

Tanenbaum (2003), resume en tres puntos básicos el diseño de Sistemas Operativos:

> En primer lugar, los sistemas operativos son programas extremadamente grandes. Ninguna persona por si sola puede sentarse frente a una computadora y producir un sistema operativo, todas las versiones actuales de UNIX rebasa el millón de líneas de código; Windows tiene más de 29 millones de líneas de código. Por ello, se convierte en una tarea bastante compleja para la producción de un sistema operativo que cumpla con los criterios óptimos de rendimiento y funcionalidad.
>
> Segundo lugar, manejar la concurrencia de varios usuarios y múltiples dispositivos de E/S, todos los cuales están activos a la vez. Administrar la concurrencia es, por su naturaleza, mucho más difícil que administrar una sola actividad secuencia.
> En tercer lugar, los sistemas operativos modernos suelen diseñarse de modo que sean portables, lo que significa que deben funcionar en múltiples plataformas de *hardware*. De igual manera, se debe reconocer cientos o incluso miles de dispositivos de E/S, los cuales se diseñan de forma independiente. (Pág.857)

Lo anterior da una clara idea acerca de que codificar un sistema operativo, es una tarea bastante complicada y que son muchos los puntos a considerar; tratar de inventar un nuevo proceso que forme parte del sistema puede ser una labor muy extensa. Es por ello que como futuro Ingeniero en Desarrollo de Software, puedes involucrarte en el diseño de sistemas operativos o puedes investigar sobre la existencia de procesos ya diseñados de uso libre, donde realices mejoras, sin que se tenga que replantear algo que ya se ha hecho, muchos de estos desarrollos son públicos y nos permiten ver el código, tales como los sistemas basados en Linux, que nos resulta un buen ejemplo de los puntos que mencionamos como básicos para el diseño un sistema operativo.

4.1.1. La visión profesional del diseño

Existe un entorno que puede afectar el diseño de un sistema por ello, no sólo debemos contemplar los puntos básicos del diseño sino es menester abrir la visión profesional entorno a la de demanda del mercado.

Actualmente hay en el mercado sistemas que satisfacen ciertas necesidades y es vital considerarlas en el diseño; por ejemplo:

Conectividad del sistema operativo: Es decir, debes pensar la forma en que va a reconocer todo lo que salga al mercado para que pueda ser conectado al sistema.
Orientación a objetos: Este se refiere a la forma en que vas a aplicar las nuevas técnicas de programación, si se va a aplicar orientación a objetos estructurados.
Aplicaciones cliente/servidor: Preguntarnos ¿el sistema que estoy diseñando va a tener aplicaciones cliente servidor como muchos que hay actualmente en el mercado? o sólo trabajará de manera local.
Y, por último, que **servicios** va a ofrecer este sistema, que logren hacer la diferencia con los servicios que ya nos ofrecen los sistemas tradicionales.

Recuerda que, mediante la formación de esta visión con respecto a lo que ofrece y utiliza el mercado actual, el desarrollo del sistema se puede aprovechar para crear aplicaciones con funciones específicas para los usuarios, basadas en el rendimiento del sistema y aprovechamiento de cada dispositivo conectado al mismo.

4.1.2. Cuestiones básicas sobre el diseño

Una vez que conoces las bases principales que debes considerar antes de diseñar un sistema operativo (abstracción, amplitud de funciones, etc., vistas en el tema 4.1 Base del diseño de sistemas operativos), y que en el tema anterior 4.1.1 Visión profesional del diseño se manifestó un panorama general en lo que debes pensar con respecto a lo que ya existe o, lo que el mercado demande, seguimos con las principales características en las que hay que hacer hincapié **al momento de iniciar** el proceso del diseño de un sistema operativo, es decir, los temas anteriores son preguntas que debes hacerte antes de comenzar a determinar la idea conceptual y enfocado al diseño del sistema, damos el siguiente paso: cuestiones básicas para **comenzar el diseño del sistema** operativo:

- **Transparencia:** La transparencia se determina sobre el diseño de interfaces de llamadas al sistema, de modo que es visible si la existencia de dos o más procesadores, aparte de saberlo por velocidad de procesamiento, lo podemos ver físicamente en alguna ventana. También se dice que un sistema no es transparente si el acceso a los archivos remotos (de una a otra computadora) se realiza mediante el establecimiento explícito de una conexión en la red con un servidor remoto, es decir, tenemos que establecer comandos de conexión entre computadoras para luego ejecutar ciertas acciones. Lo que de acuerdo con su función se tienen diferentes tipos de transparencia: los de localización, migración, réplica, concurrencia y paralelismo, según lo que se utilice en su momento.

- **Flexibilidad.** La flexibilidad es una de las cuestiones básicas sobre el diseño de un sistema operativo, ya que éste debe ser adaptable a aspectos tales como los siguientes:

    - **Monolítico:** se refiere a que la máquina deberá ejecutar un núcleo (procesador) tradicional que proporcione la mayoría de los servicios.

    - **Micronúcleo:** comúnmente nombrado *microkernel,* este deberá proporcionar la **menor** cantidad de servicios posibles, y por consecuencia

los demás servicios que sean requeridos se deben obtener a partir de servidores al nivel de usuario. De forma que muchas de las llamadas al sistema se realizan mediante señalamiento a la ubicación del núcleo (procesador) que va a proporcionar el servicio requerido.

- **Confiabilidad.** Este punto puede llegar a ser tan crítico dependiendo de la objetividad del sistema, pues si falla algún proceso, un proceso adicional al sistema operativo deberá encargarse del trabajo. La confiabilidad puede verse disminuida, ya que en muchas de las ocasiones se requiere que ciertos servicios simultáneos estén procesando información en funcionamiento. Esto se puede mejorar mediante un diseño que no requiera dentro de su arquitectura un funcionamiento simultáneo de componentes críticos y establecer un punto de redundancia; cuya característica es la **duplicidad de componentes claves del sistema operativo**. Así, no habrá procesos simultáneos que interfieran con la confiabilidad del sistema.

  Si no olvidamos la parte de la seguridad, que se basa en la protección de recursos y la tolerancia a fallos, en donde por lo general se recomienda que las fallas del sistema operativo sean ocultas con una recuperación de **forma transparente** para el usuario y nunca dude de la confiabilidad.

- **Desempeño.** El desempeño juega un papel muy importante, ya que cuando se ejecuta una aplicación se realiza en forma ágil con capacidad de poder realizar tares de multiprocesos.

- **Escalabilidad.** Indica que se tiene que ir mejorando, tal como lo requiere la nueva era de tecnología, en la administración de dispositivos de E/S en el rendimiento, la velocidad, la gestión de procesos y recursos.

- **Portabilidad.** Además de los aspectos abordados hasta el momento, la portabilidad se ha vuelto una necesidad básica en sistemas operativos modernos, pues suelen diseñarse de tal forma que, puedan utilizarse en diferentes arquitecturas de *hardware*, y deban reconocer la mayoría de los dispositivos de E/S sin ningún problema.

Para saber más, consulta:
Diseño de sistemas operativos, en: Tanenbaum, A. (2003) *Sistemas Operativos Modernos*. México: Pearson Educación.

Por lo general, cada sistema operativo por naturaleza propia suele ser distinto de otro en su nivel de complejidad y uso, de acuerdo con el objetivo de cada uno. Un sistema operativo suele ser un programa extremadamente grande, por lo cual, se puede pensar que el diseño es imposible de desarrollar, pero ningún punto debe ser excluido o minimizado.

Se debe pensar la manera en que vas a controlar tu sistema operativo y así evitar a usuarios inconformes u hostiles que desean intervenir en el funcionamiento del mismo; como llevar a extremo cada uno de los puntos anteriores para encontrar la vulnerabilidad del sistema. Por ello, se debe tener cuidado de no caer en situaciones donde no se tenga la idea clara de cómo se va a utilizar el sistema y hasta dónde se va a permitir el acceso a un usuario.

4.1.3. Conceptos generales de una arquitectura

La **arquitectura de un sistema** es la estructura que lo conforma de acuerdo con sus componentes de software, propiedades extremamente visibles y la relación entre ellos. El nivel de diseño en el que se basa la arquitectura del software es complejo, pues depende de la estructura, los componentes del sistema, sus relaciones, los principios de funcionalidad, los componentes, los conectores, la configuración y las restricciones.

La arquitectura radica en ofrecer la comunicación entre las personas involucradas, la documentación, restricciones de implementación y cualidades del sistema.
La arquitectura de los sistemas es un conjunto de conceptos y decisiones de diseño relativos a la estructura del sistema operativo, que deben hacerse antes de la ingeniería, con el fin de asegurar la satisfacción del usuario.

La arquitectura de un sistema se compone de:

- **La visión arquitectónica**: estilo, principios, mecanismos y claves de comunicación se refieren a la meta-arquitectura como que corra sobre múltiples arquitecturas de hardware y plataformas, que sea compatible con aplicaciones anteriores.
- **Las vistas arquitectónicas**, soportan la documentación y comunicación de la arquitectura en términos de los componentes y sus relaciones, las vistas son útiles para interactuar con los componentes, para desempeñar sus responsabilidades y para evaluar el impacto.
- **Los patrones arquitectónicos**, tales como capas, cliente servidor, mecanismos y puentes, son utilizados para ser tomados como base del diseño, es decir, la forma en que se va a comunicar con otros dispositivos.
- **Los principios de diseño arquitectónico claves**: abstracción, separación de responsabilidades, simplicidad y técnicas de ocultamiento de interfaces, son otros de los conceptos que son utilizados para el diseño y son técnicas de programación que adquieres al momento de estudiar programación sobre todo si es orientada a objetos.

Dichos conceptos surgen a partir del estándar de la Arquitectura - IEEE 1471-2000, misma que a continuación se menciona:

> La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los

principios que orientan su diseño y evolución. Adoptada por Microsoft en estrategia arquitectónica / MSF.

A modo de resumen, recuerda que la arquitectura es el pilar para llegar al diseño de un sistema, es la forma en que vas a aplicar las técnicas y mecanismos para lograr la comunicación entre el sistema – usuario – de manera que, determina cómo se transmite la información, quién la va a recibir y de qué manera la va a trabajar para dar paso a la siguiente etapa conocida como la implementación, en donde se pone en funcionamiento los mecanismos que se desarrollaron.

4.1.4. Características diferenciales del diseño

El definir la arquitectura a usar en el momento del desarrollo de un sistema da origen a los requerimientos, los cuales son los puntos a seguir y establecen el funcionamiento o comportamiento que el sistema va a tener.

Una vez que se tienen los requerimientos hay que establecer cuáles son las características que van a ser la diferencia con otros sistemas diseñados.

Se puede decir que en las características diferenciales que hacen un diseño de un sistema operativo, se pueden considerar las siguientes:

- Considerar un **nivel de abstracción** (mencionado en temas anteriores) donde los desarrolladores pueden establecer el comportamiento global del sistema, en el que se toma en cuenta el funcionamiento, rendimiento, confiabilidad, etc., así, una excelente arquitectura hace del diseño un sistema fácil de establecer, ya que el usuario solo visualiza lo que el sistema hace y no se preocupa por el cómo lo hace.
- Considerar la **evolución del diseño** del sistema: el diseño de la arquitectura sirve como memoria para el sistema, lo que va evolucionando en tiempo de desarrollo, este sirve para el proceso de mejorar el sistema, ya que se conocen los

antecedentes y el camino hacia lo que se quiere lograr, cambiar o mejorar del diseño que ya se tenía.

Considerar los puntos mencionados es menester en el diseño de un sistema.

El documentar el proceso de la arquitectura del sistema ayuda bastante al diseño y de igual manera a todo el ciclo de vida de este. Se pueden considerar otras de las características diferenciales del diseño, tales como los servicios disponibles, especialización de servicios, recursos compartidos, transparencia de localización, independencia de *hardware*, diálogo basado en mensajes, escalabilidad y reusabilidad de componentes.

4.2. Diseño de interfaces

Según la manera en que el usuario y las computadoras interactúan entre sí, para el manejo de un sistema y todas sus aplicaciones, las interfaces son esas ventanas que aparecen para dar indicaciones a la computadora o la forma en que el usuario se comunica con los dispositivos de la computadora convirtiéndose en uno de los aspectos más fascinantes en el proceso de desarrollo de sistemas pues, mucho del trabajo se puede plasmar en la presentación de cómo está planteado el sistema para su manejo.

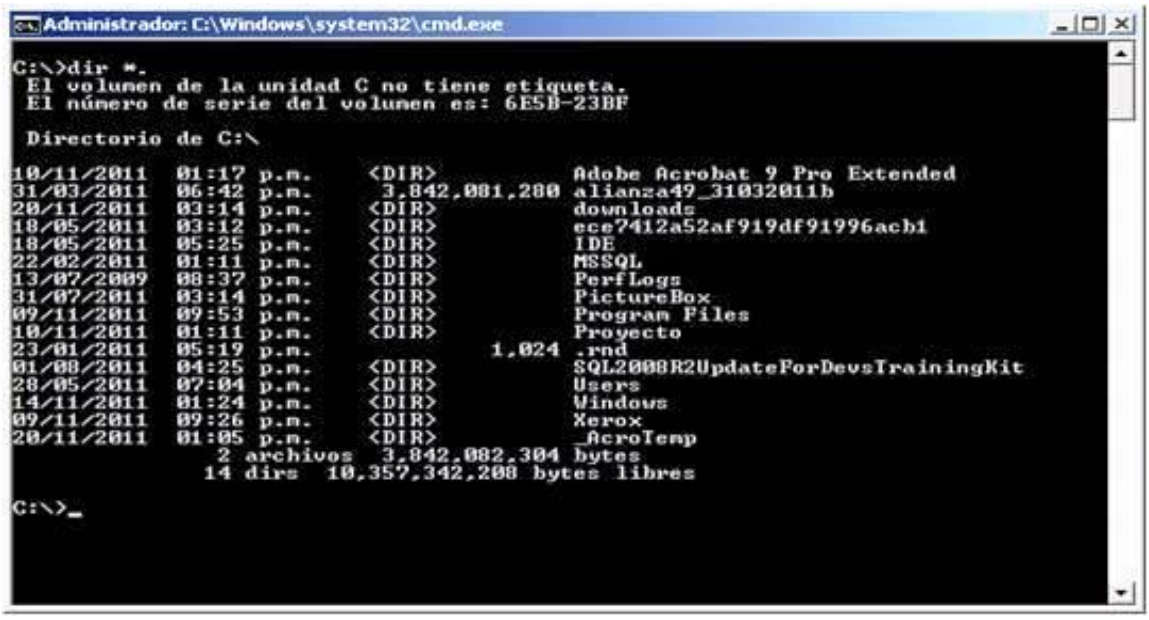
Aunque ya se ha mencionado que, durante la etapa de desarrollo de un sistema operativo, la administración de los recursos de hardware es el objetivo principal del diseño, no se podrá dejar de lado la forma en que los usuarios interactúan con la computadora.

Los diferentes tipos de interfaces que deben considerar para el desarrollo de un sistema son:

- **Interacción humano-computadora.** Está relacionado con la práctica del diseño, construcción e implementación de sistemas de cómputo interactivos centrados en el usuario. Una de sus principales actividades en esta área, es el desarrollo de nuevos sistemas de interface para los usuarios, lo que implica desarrollo de nuevas técnicas de graficación que sean visualmente atractivas para quien usa el sistema, incorporando animaciones colores, etcétera.

Dispositivos de interface.

- **Interface basada en texto.** Los desarrollos de sistemas orientados en texto o caracteres sólo tenían la capacidad de desplegar códigos *ASCII* en pantalla. En contraste a los orientados a gráficos, la interface de texto consiste en un conjunto de comandos que el usuario debe memorizar y a los cuales se dan instrucciones al sistema para el manejo de los recursos.



Interface de usuario de *MS-DOS*

- **Interface gráfica de usuario.** Este es un programa que aprovecha las capacidades de despliegue gráfico de la computadora, lo cual permite hacer más sencillo el manejo del sistema; este tipo de interface utiliza iconos y menús para realizar comandos, abrir archivos y ejecutar alguna acción dentro de la aplicación. Algunos de los componentes gráficos comunes en este tipo de interfaces son: el apuntador, dispositivo apuntador, iconos, menús, ventanas, y escritorio. Cabe señalar que cada sistema operativo tiene su propia arquitectura gráfica.

Ejemplo de İnterface gráfica

- **Interfaces alternativas.** Este tipo de interfaces son poco utilizadas, pero de gran utilidad para pensar sobre alguna alternativa de seguridad; las interfaces alternativas pueden ser las que se basan en el reconocimiento de voz, la cual soporta un diálogo interactivo entre el usuario y una aplicación de software. Los dispositivos de interface neuronal permiten a los usuarios aprovechar las señales eléctricas generadas por sus cuerpos para controlar una computadora o dispositivos eléctricos conectados entre sí.

- **Interface de usuario.** Su principal objetivo es implementar interfaces que sean eficientes y efectivas al ser utilizadas por el usuario; el desarrollo se inicia sobre el planteamiento de prototipos de interfaces en donde se determinan que eventos ocurrirán y los procesos lógicos a dichos eventos.

Para saber más, consulta:
Niño, J. (2011). *Sistemas Operativos Monoexpuesto.* España: Editex.

Existe una metodología para la etapa de pruebas conocida como **pruebas de usabilidad**, que puede validar el diseño de la interface y revelar áreas que requieran refinamiento. La finalidad de esta técnica es poder identificar el uso que se le da de forma directa con el usuario. Se pude enseñar a los usuarios a que sepan plasmar la idea de mejoramiento de la interface, un usuario podría expresar lo que desea, por ejemplo: “Quiero que, en la parte superior de la interface gráfica, se pueda tener acceso a mis archivos almacenados”. Con este tipo de análisis se interactúa de forma directa con el usuario para que el desarrollador pueda ofrecer un mejoramiento en el diseño de las interfaces.

4.2.1. Principios sobre el diseño de interfaces

Recuerda que el diseño de interfaces es lo que el usuario podrá ver y la manera en que se comunica con la computadora y para que cumpla con su función debe cubrir ciertos aspectos básicos.

Los principios que se deben considerar durante la etapa de diseño de interfaces son los siguientes:

- **Sencillez.** Para evitar que los usuarios tengan complicaciones con las interfaces al momento de estar utilizando el sistema, éstas deben basarse sobre el principio de la sencillez, pues las hace más fáciles de entender e implementar.
- **Integridad.** Con la finalidad de permitir utilizar todas las funciones del sistema se debe basar su diseño en el principio de la integridad para que permita el uso completo.
- **Eficiencia.** Para los usuarios, las llamadas al sistema deben ser de forma transparente, es decir no ocultar procesos o ser muy claros en los manejos de los mismos; esto se logra si la implementación de los mecanismos es de forma eficiente.[1]

El considerar estos principios, garantiza al desarrollador que el uso del sistema pueda ser del agrado de los usuarios, por lo tanto, no debemos omitirlo al momento de diseñar nuestras interfaces.

4.2.2. Paradigmas

Recuerda que paradigma es el esquema o patrón que siempre sigue siendo algo tan cotidiano que damos por hecho que así se tiene que hacer.

Otros de los puntos que se deben considerar para poder iniciar con el diseño de las interfaces, te los presentamos a continuación, mediante los **Paradigmas**.

Paradigmas de programación más comunes:

- **Imperativos.** Este tipo de paradigma es considerado como el más común para el desarrollo de sistemas imponiendo acciones o cosas como su nombre lo dice acciones que son obligatorias.
- **Funcional.** Este tipo de paradigma se basa en un tipo de programación en forma de funciones matemáticas.
- **Lógica.** Para este paradigma se tienen definidas reglas lógicas que resuelven problemas planteados por el sistema.
- **Orientado a objetos.** En este tipo de paradigma, la finalidad es encapsular estado y operaciones en objetos que poseen clases y herencias, los cuales se pueden comunicar entre ellos.
- **Estructurado.** En el paradigma estructurado, el código se divide en bloques y estructuras, que pueden o no comunicarse entre ellas; puede ser controlado por secuencias, selección e interacción.

- **Guiado por eventos.** La estructura de este paradigma y su ejecución de los programas van determinados por los sucesos que ocurran en el sistema o que ellos mismos provoquen.

Los paradigmas tienen una importancia clave en la programación, pues por medio de estos se pueden crear planteamientos y bases para poder desarrollar un software[1].

4.2.3. Llamadas al sistema

Durante el desarrollo de las interfaces, las llamadas al sistema deben cumplir la regla de: menor número de llamadas menor trabajo al procesador, debido a que es importante contar con un paradigma de datos unificador, el cual se encargará de este problema y puede ser de mucha utilidad durante el proceso de diseño no sobrecargando de trabajo al procesador[1].

Si se ignora este mecanismo unificador de datos, se obtiene como consecuencia, muchas llamadas al sistema y éstas pueden afectar, pues no deben interferir en la potencia del *hardware*; si este cuenta con un mecanismo bastante eficiente para hacer algo sobre los mecanismos de acceso o, si una llamada al sistema es rápida, los usuarios siempre podrán construir interfaces más cómodas con base en ella.

Como se mencionó con anterioridad, el sistema operativo, para ser eficiente deberá ofrecer el menor número de llamadas al sistema, para ello debe de cumplir con los criterios de:

- Contar con un paradigma unificador de datos.
- Manejo de forma general de llamadas al sistema.
- Procedimientos de biblioteca sencillos y específicos.
- Determinar la implementación de las llamadas al sistema sobre conexiones o sin conexiones.

[1] Si quieres complementar tu subtema, puedes encontrar más información sobre los *paradigmas* en la página 861 del libro de Tanenbaum (2003). La ficha completa la encontrarás en tus fuentes de consulta.

Otra consideración que va relacionada con la **interfaz de llamadas al sistema** es su visibilidad; algunos sistemas operativos como Unix, el cual reconoce llamadas adicionales, es decir, la lista de procesos es pública. Por el contrario, en el caso de Windows, la lista de llamadas al sistema no se ha hecho pública. La diferencia entre hacer públicas o no las listas, ayuda a los implementadores a la flexibilidad pues, pueden cambiar las llamadas a sistemas reales.

La cantidad y tipo de llamadas al sistema va en relación con el sistema operativo y varían de uno a otro. Existen, por lo general, las llamadas al sistema para la ejecución de ficheros que contienen programas, y pedir más memoria dinámica puede ser una solución temporal pero que nos exige tener más recursos de memoria.
Tenenbaum (2003) nos dice al respecto que:

> Las llamadas orientadas a conexiones en comparación con las llamadas sin conexión. Las llamadas estándar para leer un archivo de Unix y Win32 son orientadas a conexiones. Primero se abre un archivo, luego se lee y, por último, se cierra. Algunos protocolos de acceso a archivos remotos también son orientados a conexiones. Por ejemplo, para usar FTP, el usuario primero inicia sesión en la máquina remota, lee los archivos y luego termina la sesión (p. 866).

Un sistema operativo funciona solicitando al procesador las actividades a realizar, si dichas actividades requieren llamar varias veces al procesador hacen que el sistema se haga lento y, por tanto, obsoleto y poco competitivo al mercado. Muchas de estas llamadas poseen una interfaz para que el usuario sea capaz de autorizar que el proceso se realice de manera óptima, muchas de esas llamadas al sistema tienen un orden lógico para realizarse, en donde se requiere que se abra un archivo, se realice una operación y posteriormente se cierre pero, un gran problema es que si el archivo se usa continuamente, se plantea la solución de colocarlo en una memoria temporal dinámica, lo cual sólo soluciona la cuestión de acceso rápido, pero presenta la posibilidad de consumir muchos recursos. Por lo tanto, es labor del diseñador establecer cómo va a manejar las llamadas al sistema.

4.3. Implementación y desempeño

Recuerda que un sistema operativo, aparte de controlar el hardware, crea un entorno para el usuario–máquina, esto vuelve la tarea difícil ya que se requiere, como lo hemos mencionado, algo que sea atractivo y a la vez funcional. Es decir, que todos los componentes de un equipo físico trabajen como parte de una misma máquina logrando así que cada componente se integre a otro, cada uno en su función y traspasando la información que tenga que enviar a otro componente.

La implementación de las aplicaciones en general se puede clasificar en tres apartados, de acuerdo con su función y lo que utilizan:

- Integración de los componentes.
- Objetos distribuidos.
- Servicios.

Un sistema operativo está formado por un conjunto de sistemas o de procedimientos en los cuales puede darse el caso que dos o más procedimientos diferentes tengan el mismo objetivo; existe la posibilidad que uno sea más rápido que otro, pero también, puede ser que el sistema más rápido puede ser menos confiable que el sistema más lento.

Si en estas partes de sistemas o procedimientos tratamos de realizar optimizaciones complejas, el sistema operativo en su estructura puede generar errores, para evitar éstos es conveniente aplicarlos sólo si son necesarios. Existen diferentes técnicas que pueden ser utilizadas para mejorar el desempeño en la estructura del sistema, las cuales son recomendables investigar y cuales conviene implementar, de acuerdo con lo que queremos diseñar: éstas son métodos de ordenamiento de almacenamiento.

4.3.1. Estructura del sistema

Si recuerdas, en unidades anteriores, hablamos sobre la importancia de que el sistema operativo administre el procesador, memoria y dispositivos de E/S, así como la administración de archivos, seguridad y protección del sistema operativo; como ya conoces estos temas, ahora al momento del diseño debemos preguntarnos sobre la parte que corresponde a la estructura del sistema operativo. Comprenderás porqué algunos sistemas operativos cuentan con más elementos que otros, ya que dependen de estos componentes y su evolución en el mercado, el cómo están arquitectónicamente construidos, y por qué debes al momento de diseñar adicionar o no, servicios al sistema para que sea compatibles con las novedades del mercado o lo que la sociedad demanda.

Es importante tener en cuenta la etapa de análisis del sistema, ya que es aquí donde se tendrá línea para continuar con la etapa del desarrollo, para saber lo que se tiene que hacer, cómo hacerlo y cuál será su límite. La arquitectura de un sistema representa un paso hacia el desarrollo del mismo, una vez que se analiza y definen los requerimientos del usuario obtenidos por el análisis hecho de todos los puntos anteriores, el siguiente paso es definir la estructura que contendrá el sistema operativo.

La estructura de un sistema operativo está dada por su tipo y antes de seguir, primero se definen los tipos de sistemas operativos que existen, posteriormente se explica qué estructura maneja cada uno de ellos.

Hoy en día existen diferentes tipos de sistemas operativos, dentro de los cuales están los más comunes:

- **Monotarea**: sistema que sólo puede ejecutar una tarea a la vez.
- **Multitarea**: es aquel que tiene la capacidad de ejecutar varias tareas.
- **Monousuario:** en este tipo cada proceso puede ser ocupado por un usuario único durante el tiempo de ejecución.

- **Multiusuario**: el sistema que tiene esta característica permite ser ocupado por dos o más usuarios durante el tiempo de ejecución de sus procesos.
- **Monoproceso:** como su nombre lo dice solo puede gestionar un solo procesador.
- **Multiproceso**: permite gestionar a través de dos o más procesadores para dividir la carga entre ellos al realizar una actividad.
- **Cliente-Servidor:** cuando el sistema desarrollado tiene esta característica distribuye las tareas entre diferentes llamadas al sistema. En donde el cliente hace una petición y el servidor da respuesta a la misma.

Todos esos tipos de sistemas operativos tienen su estructura definida por sus funciones a realizar. Así para cada tipo de sistema operativo y sus funciones a realizar hay diferentes tipos de estructuras tales como:

- **Por capas y anillos.** En este tipo de estructura se contemplan seis capas para su formación[2]:
    - Capa 0, capa que se encarga de la administración de multiprogramación básica del microprocesador.
    - Capa 1, en esta capa se encarga de gestionar el espacio en la memoria.
    - Capa 2, para esta capa su principal actividad es comunicar entre cada proceso y la consola del operador.
    - Capa 3, en esta capa la gestión se enfoca sobre los dispositivos de E/S.
    - Capa 4, esta capa se gestionan los programas de los usuarios.
    - Capa 5, es donde se alojan los procesos del operador del sistema.

[2] Si deseas complementar la información, acude a la página 287 del libro de Tanenbaum (2003). La ficha completa la encontrarás en tus fuentes de consulta.

- **Arquitectura Máquina virtual**. Esta alternativa de software se vuelve cada vez más útil para el manejo de servidores y economizador de *hardware*, de tal forma que el sistema emula la existencia de *hardware* y el software convierte las peticiones a la máquina virtual en operaciones sobre la máquina física. Existen diferentes tipos de virtualización[2]:

    - *IBM VM*, las máquinas virtuales se basaban en mono-tarea.
    - *Java VM*, esta máquina permite la ejecución de códigos binarios en distintas plataformas.
    - *VMWare*, tiene la facultad de poder ejecutar varias máquinas con distintas sesiones,
    - *Citrix*, al igual que *vmware* puede ejecutar varias máquinas con distintas sesiones y con sus características propias del software que permite el manejo de dispositivos entre sus máquinas.

- **Arquitectura kernel monolítico.** Esta arquitectura se define por la existencia de una colección de procedimientos y, de forma independiente, cada procedimiento podrá invocar a cualquiera de los otros, cada vez que requiere hacerlo, por medio de su interface determinante a través de parámetros y resultados.

- **Arquitectura microkernel y multihilado.** Este tipo de arquitectura se basa en obtener la mayor funcionalidad del kernel, se limita para que se pueda ejecutar en modo privilegiado y que permita las modificaciones y extensiones de forma simple y sencilla.

- **Arquitectura orientada a objetos**. Para este tipo de arquitecturas los recursos que provee el sistema operativo son por medio gráfico u objetos.

- **Arquitectura cliente-servidor**. En esta arquitectura el modelo principal de trabajo se encarga de dividir los procesos independientes que operan entre ellos, lo cual es de utilidad para intercambiar información, recursos, procesos y servicios.

En base a tus requerimientos establecidos, el siguiente paso es que escojas el tipo de sistema operativo que vas a diseñar, por ejemplo: monotarea, multitarea, multiusuario, cliente servidor, etc. y sobre esto decidir cuál de las estructuras le conviene al soporte de tu sistema para que funcione óptimamente.

Por ejemplo: Windows que es un sistema multiusuario, multitarea, multiproceso y aplica cliente servidor, hace una combinación de estas estructuras ya que soporta la instalación de máquinas virtuales, tiene comportamiento de la arquitectura cliente servidor, soporta el multiprocesamiento y manejo de hilos.

Los nuevos desarrollos de sistemas operativos tienen un panorama más amplio para su desarrollo. Y en cada subsistema, aplica el tipo de estructura de acuerdo con lo que quiere realizar. Como ya conoces los tipos de sistemas y las estructuras puedes escoger cuales vas a aplicar en tu sistema operativo.

4.3.2. Mecanismos en comparación con políticas

Es necesario diferenciar que los mecanismos son parte de la estructura del sistema operativo y tienen estructuras, arreglos y uniones. Las políticas van enfocadas a los procesos de cada usuario; la separación de estos conceptos ayuda a la coherencia y estructura en el diseño de un sistema operativo, ya que estarán ordenados y clasificados de acuerdo a lo que hacen y lo que necesitan para operar.

Si aplicas y diseñas el código necesario para que funcione todo lo que quieres que tu sistema realice, se generan miles de líneas de código, logrando por instantes perder coherencia en las formas. Tanenbaum (2003), dice: “Otro principio que ayuda a la coherencia arquitectónica, además de que evita que las cosas crezcan demasiado y las mantiene bien estructuradas, es el de separar el mecanismo y las políticas” (pág. 871). La intención de permitir la mezcla de las estructuras contenidas por arreglos de enteros, de

caracteres, miembros de la estructura y de uniones, permite definir un nuevo tipo de datos el cual puede ser utilizado como si fuera un tipo primitivo.

La **ortogonalidad**, es la capacidad de poder combinar estos distintos tipos de formas independientes, la cual se basa en los principios de la sencillez e integridad para el desarrollo de un sistema. Puede ser utilizado para separar los conceptos de procesos y subprocesos, donde un proceso es un contenedor de recursos y un subproceso es una entidad.

Si al momento de programar tu sistema en el lenguaje en el que decidas hacerlo agrupas tus elementos y los manejas como un nuevo tipo de datos hará la programación más simple y, si aparte separas los procesos de acuerdo con lo que requiere el usuario desde la ortogonalidad, harás que el código sea mucho más simple de usar.

4.3.3. Estructuras estáticas o dinámicas

El desarrollo de la arquitectura del sistema operativo se refiere a la estructura que contendrá el sistema o forma del mismo, este tipo de estructuras dinámicas o estáticas, son usadas en las líneas de código y se pueden clasificar de dos tipos:

- **Dinámicas**. Este tipo de estructuras se distingue por ser más flexibles y permiten adaptarse a todos los recursos disponibles, pero tiene como desventaja que requieren de un gestor de memoria dentro del sistema operativo para estar guardando en memoria los cambios que se producen al adaptarse a los recursos.
  En programación viste que las estructuras dinámicas puedan modificar su tamaño siendo posible un incremento en el mismo de acuerdo con tus necesidades, pero requieres quien lo administre.

- **Estáticas**. Se caracterizan por su fácil comprensión; su programación es más simple y su uso es más rápido ya que no existen estos cambios y sus datos quedan fijos.

Al quedar fijo el tamaño, de antemano se limita y no es necesario que alguien más administre el crecimiento.

Estas estructuras las puedes entender, si en tus líneas de código de programación defines arreglos, y si pueden ser de dos tipos: el que puede ir creciendo en su tamaño de acuerdo con las necesidades o el que tiene un tamaño definido.

El desarrollador del sistema operativo deberá determinar qué tipo de estructura le conviene utilizar para la implementación del desarrollo de su arquitectura; considerando espacios de procesos de usuario o espacio de procesos en núcleos, el dinámico puede ir creciendo en base a sus necesidades, pero requiere quien lo controle; y, el estático, que ya está definido haciéndolo más fácil, pero limitado.

En conclusión, las estructuras mencionadas en el tema anterior se refieren a la estructura general del sistema operativo y las estructuras mencionadas en este tema hacen referencia exclusivamente a la manera en que se va a establecer la programación o líneas de código.

4.3.4. Implementación descendente o ascendente

Una vez que ya tienes decididos todos los elementos a utilizar en el diseño, el siguiente paso es decidir cómo vas a implementar el sistema. Para la implementación de un sistema, existen diferentes tipos de técnicas que son de gran utilidad para llevar a cabo este proceso, éstas son:

- **Ocultación del *hardware***, esta técnica permite ocultar las interrupciones, convirtiéndolas en operaciones de sincronización entre hilos. El ocultar la arquitectura del hardware, permite facilitar la transportabilidad del sistema operativo. La fuente del sistema operativo debe ser única y una compilación condicional.

- ***Indirección*.** Para esta técnica deberá existir cierta flexibilidad por parte del sistema operativo, ya que, si por algún motivo el usuario da la entrada de un proceso por medio del teclado, al pulsar la tecla puede obtener un valor que no corresponde con lo deseado, para solucionarlo debe existir la posibilidad de utilizar configuraciones distintas de teclados.

- ***Reentrabilidad***. En esta técnica se permite la ejecución de forma simultánea de algún fragmento de código, la ejecución se puede dar por los casos dentro de un multiprocesador o en un *monoprocesador,* en éste pueden darse interrupciones cuando se ejecuten las mismas porciones del código al anterior.

Para saber más de estos conceptos, consulta el siguiente manual:
Jorba, J. (2004). *Administración Avanzada de GNU/Linux*, consultado el 20 de julio de 2023 en: http://lash.utrng.edu.mx/wp-content/uploads/2018/03/1.pdf

Para la implementación de un sistema, como recomendaciones iniciales para un buen proceso, podría ser el verificar la existencia de errores. Cuando una llamada al sistema falla puede ser porque los **ficheros no existen** dentro del módulo o bien pertenecen a otro. Se deben considerar la mayor parte de las pruebas al inicio del procedimiento para validar la ejecución de llamadas al sistema.

Una vez que se consideró ocultar el hardware, el uso de diferentes combinaciones para las entradas de datos, y que permite ir introduciendo datos en forma simultánea, debemos considerar de qué manera se va a implementar ésta; puede ser de dos maneras diferentes: ascendente y descendente. En la manera ascendente empieza por probar de forma separada que funcionen éstas y al final hacer una prueba combinando todas las llamadas.

Si por el contrario la manera de implementarlo será descendente, entonces se empieza la implantación con un esqueleto que incluye las llamadas a estos procesos que después se irán probando de manera individual.

4.3.5. Principios de optimización

Los principios de optimización son reglas que deben estar presentes en todos sistemas o módulos que se desarrollen para un sistema operativo. Estos son:

Primer principio: Una parte importante para el desarrollo de un software es que su funcionalidad sea óptima; es decir que funcione de forma rápida y común, que durante la implantación del sistema no se generen errores de diseño, por lo cual es conveniente optimizar sólo lo necesario dentro de la arquitectura del sistema.

Segundo principio: Deben considerarse las funciones que se necesitan adicionar a la estructura y validar si es necesario modificarla para su desempeño, dependiendo de lo que se requiere del sistema: sencillo y rápido o robusto y lento.

Tercer principio: Por lo general, antes de que se desee optimizar un sistema, se debe tener presente la dimensión de la arquitectura del sistema para darle **la seguridad, integridad y optimización** suficiente para un buen desempeño.

Si se cubren estos puntos, el sistema operativo será óptimo.

4.3.6. Uso de cachés

La caché es un medio de utilidad de almacenamiento temporal, su función es habilitar el acceso a datos que han sido recientemente consultados; esto permite hacer más ágil el proceso de lectura que el sistema hace de los datos al momento de realizar un proceso.

Como ya se mencionó en temas anteriores, el uso de cachés es con la intención de que aquellos datos que más se utilicen sean colocados al alcance de quien los necesita, evitando así las constantes llamadas al procesador y así evitar la necesidad de abrir, consultar y cerrar el archivo cada que sea utilizado. Por lo tanto, se emplea la técnica de usar cachés que guarda en espacios de memoria aquello que más se utiliza.

Existen diferentes tipos de caches:

- **Caché de bloques**. es un pequeño segmento de memoria *RAM,* que se añade a algún sitio del disco, con la finalidad de almacenar los datos que han sido leído con anterioridad, teniendo como resultado agilidad al cargar los datos.
- **Caché de paginación**, para este tipo de caché el almacenamiento se hace en memoria *RAM,* dentro del disco duro.

Según Tanenbaum (2003, el uso de cachés es similar en los distintos sistemas operativo, pues:

> La tarea del administrador de caché consiste en mantener en memoria los bloques del sistema de archivos que se han usado recientemente, para reducir el tiempo de acceso en las siguientes referencias. Windows tienen un solo caché integrado que se usa con todos los sistemas de archivos utilizados, incluidos NFTS, FAT-32, FAT-16 e incluso sistemas de archivos de CD_ROM. Esto implica que los sistemas de archivos no necesitan mantener sus propios cachés. (p. 849)

En conclusión, hacer un buen uso de cachés agiliza el acceso a información recurrente, siendo esta entendida como velocidad de respuesta, sin embargo, lo único que se está haciendo es tener, en palabras comunes, a la mano la información que más se utiliza.

Cierre de la unidad

Has concluido la cuarta unidad del curso. A lo largo de ésta, se vieron conceptos básicos sobre diseño de sistemas operativos, bases, interfaces, implementación, temas de diseño, la visión profesional, los principios del diseño y sus paradigmas, las llamadas al sistema y por último los subtemas de estructura, sus mecanismos, su implementación, optimización y el manejo de los cachés.

En caso de que los temas que se acaban de mencionar no te sean familiares o no los recuerdes, es aconsejable que revises nuevamente la unidad. Si los conceptos son parte de tus conocimientos, ya estás listo para terminar esta unidad didáctica de Programación de Sistemas Operativos.

Recuerda que todo proyecto que desarrolles debe tener como característica principal la ayuda, actualización e implementación de soluciones y mejoras estratégicas como parte de una organización, es posible también encontrar áreas de oportunidad en el contexto donde te desenvuelves siempre y cuando tus proyectos vayan dirigidos siempre al bien común.

Para saber más

Si deseas saber acerca de cómo se crea un diagrama de flujo, consulta la siguiente dirección electrónica:

Cómo crear un diagrama de flujo(s/f), consultado el 15 de Junio de 2012 https://www.yumpu.com/es/document/view/14486398/como-crear-un-diagrama-de-flujo-educarchile.

Si deseas saber más sobre reentrabilidad, ocultación del hardware e indirección puedes consultar:

Jorba, J. (2004). *Administración Avanzada de GNU/Linux*, consultada el 20 de julio de 2023 http://lash.utrng.edu.mx/wp-content/uploads/2018/03/1.pdf

Fuentes de consulta

Bibliografía básica

Candela, S. y García, C. (2007). *Fundamentos de Sistemas Operativos. Teoría y ejercicios resueltos*. España: Paraninfo.

Niño, J. (2011). *Sistemas Operativos Monoexpuesto.* Madrid: Editex.

Ortiz, H. (2005) *Sistemas Operativos Modernos*. Colombia: Medellín.

Silberschatz, A. (2006). *Fundamentos de Sistemas Operativos*. Madrid-México: Mc. Graw Hill.

Stallings, W. (2005). Sistemas *Operativos Modernos: Aspectos Internos y principios de diseño*. México: Pearson, Prentice Hall.

Tanenbaum, A. (2003). *Sistemas Operativos Modernos*. México: Pearson Educación. http://bibliotecas.ucasal.edu.ar/opac_css/index.php?lvl=notice_display&id=15735