

**Tecnológico Nacional de México**

**Instituto Tecnológico de Reynosa**

Materia: Sistemas Operativos 2

ACTIVIDAD05: Investigación exhaustiva del Tema02: Procesos y comunicación con sus 4 subtemas

Tema 2: Procesos y comunicación

Alumno: Castillo Jr. Gregorio

Numero de control: 19580589

Correo electrónico: L19580589@reynosa.tecnm.mx

7mo Semestre Matutino Salón 7

Docente: Mario José Santiago Sánchez

Fecha de entrega: 06/10/2022



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Consecutivo** | **Numero de**  **control** | **Apellido Paterno** | **Apellido Materno** | **Nombres(s)** | **Correo electrónico**  **Institucional** | **Firma de que está**  **autorizando que se**  **entregue esta Publicación** | **Fotografía del rostro de cada**  **Integrante del equipo (selfie)** |
| **1** | **19580589** | **Castillo Jr** |  | **Gregorio** | **L@19580589@reynosa.tecnm.mx** | Abrir foto |  |
| **2** | **19580595** | **Flores** | **Acosta** | **Sheila Lizeth** | **L@19580595@reynosa.tecnm.mx** | No hay descripción disponible. |  |
| **3** | **19580867** | **Morales** | **Calixto** | **Daniel Alexander** | **L@19580867@reynosa.tecnm.mx** | No hay descripción disponible. |  |

Equipo #12

# Tabla de contenidos

[Tabla de contenidos 3](#_Toc115678382)

[Contenido 4](#_Toc115678383)

[2.1 Procesos 4](#_Toc115678384)

[2.2 Comunicación 10](#_Toc115678385)

[2.3 Nombres 13](#_Toc115678386)

[2.4 Sincronización 15](#_Toc115678387)

[Conclusiones 21](#_Toc115678388)

[Herramientas y recursos 22](#_Toc115678389)

[Bibliografía 23](#_Toc115678390)

# Contenido

## 2.1 Procesos

***Definición***

Se le conoce como un programa o mandato que se ejecuta de manera real en el sistema. Estos procesos existen dentro de las jerarquías llamadas como “parde-hijo”, ya que un proceso iniciado por un programa es un proceso llamado “padre” y un proceso es “hijo” es el producto del “padre”.

El proceso padre puede tener varios hijos (procesos), pero un hijo (proceso) solo puede tener un padre.

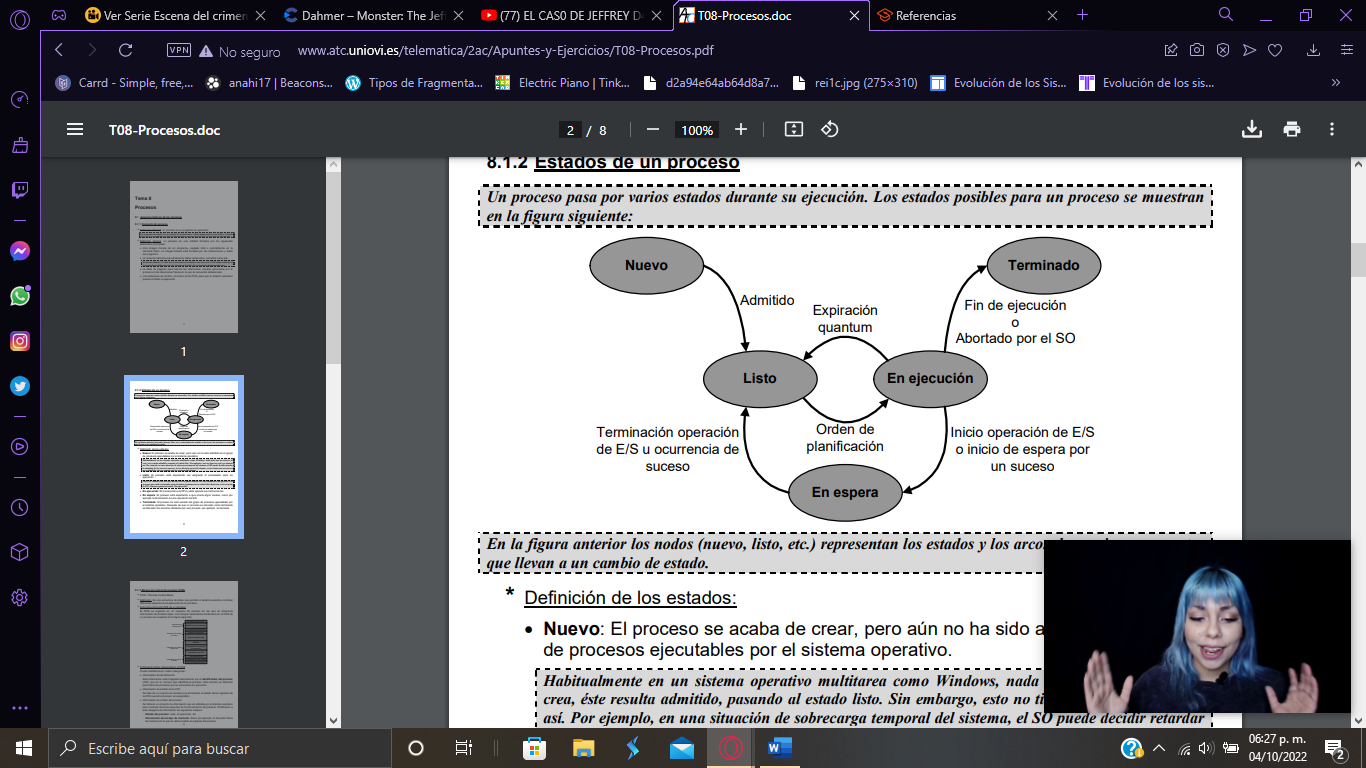
***Dentro de…***

El sistema nos asigna un número de identificación de proceso (PID), cada que el proceso es iniciado. Si el mismo programa inicia varias veces, este tendrá un numero PID distinto cada vez.

Cuando iniciamos un proceso en el sistema, este proceso utiliza una parte de recursos disponibles en el sistema. Cuando se ejecuta más de un proceso, un planificador (que se encuentra incorporado al SO) proporciona a cada proceso su parte del tiempo del sistema, dependiendo de las prioridades establecidas. Las prioridades pueden llegar a cambiarse mediante algunos comandos dependiendo del sistema.

***Estados de un proceso***

Los procesos siempre pasan por varios estados durante la ejecución. A continuación, se mostrará un diagrama de los estados correspondientes por los que pasa el proceso.



**Nuevo:** es cuando el proceso se acaba de crear, pero aun no ha sido admitido en el grupo de los procesos ejecutables por el SO.

**Listo:** es cuando el proceso esta esperando ser asignado al procesador para su ejecución.

**En ejecución:** pasa cuando el proceso tiene la CPU y esta ejecuta las instrucciones.

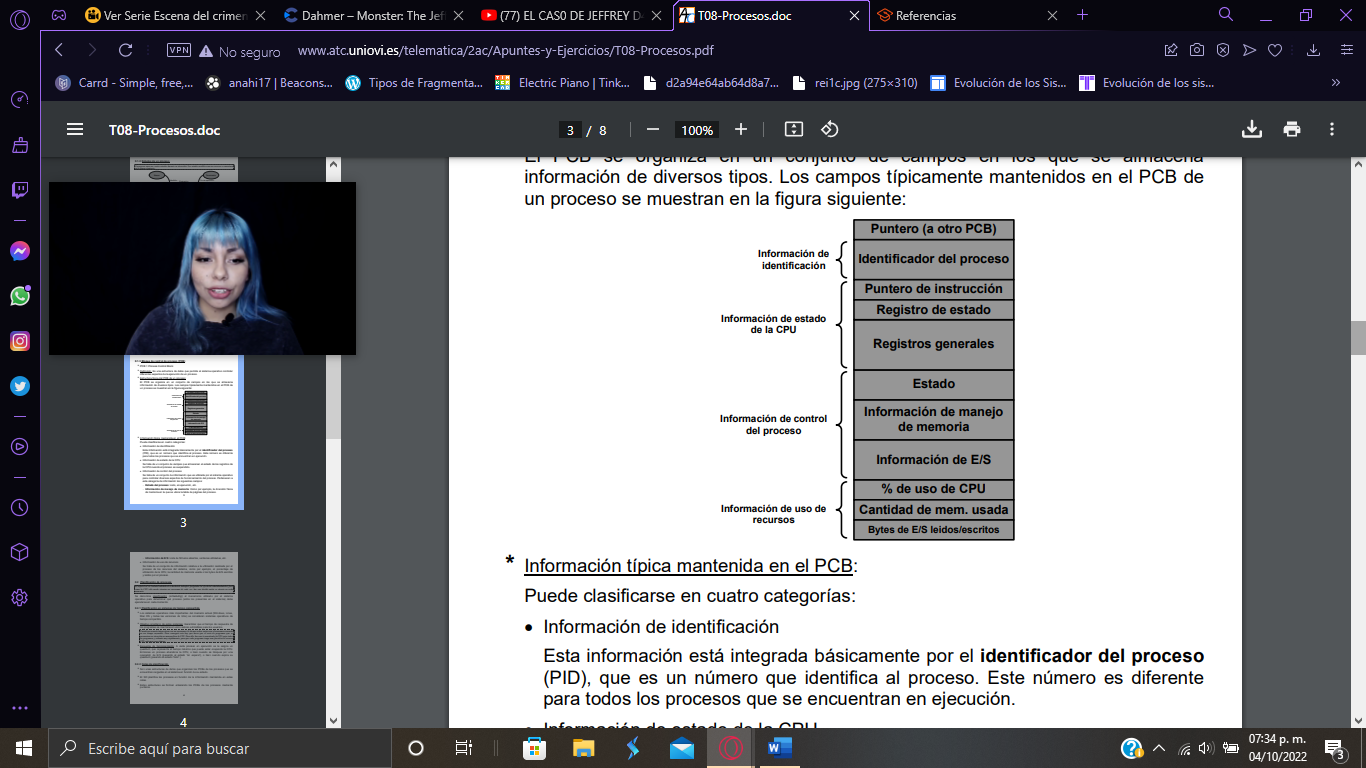
**En espera:** el proceso espera a que ocurra algún suceso, como por ejemplo la terminación de una operación de entrada/salida.

**Terminado:** Es cuando el proceso se saca del grupo de procesos ejecutables por el sistema operativo. Después de que el proceso sea marcado como finalizado, todos los recursos utilizados por ese proceso serán liberados, como, por ejemplo: la memoria.

***Bloque de control de proceso (PCB)***

Es un tipo de estructura de datos que permite al SO controlar distintos aspectos durante la ejecución de nuestro proceso.

La estructura típica del bloque de control de proceso es organizada en un conjunto de campos en los que se almacena la información de distintos tipos. Los típicos campos mantenidos del PCB de un proceso son:



***Planificación de procesos***

Se denomina planificación al mecanismo que es utilizado por el SO para determinar que proceso debe ejecutarse en cada momento.

***Planificación en sistemas de tiempo compartido***

* Tenemos los sistemas más conocidos del mercado actual: Windows, Linux, Mac OS y todas las versiones Unix, estas son consideradas sistemas operativos de tiempo compartido.
* Objetivo prioritario de estos sistemas: nos garantizan que el tiempo de respuesta de los programas se mantienen en unos valores admisibles para los usuarios.
* Esquema de función: cada proceso que este en ejecución se le asigna un quantum, que representa el máximo que puede tener ocupado la CPU, o cuando se bloquea por una operación E/S, en pocas palabras cuando expira su quantum.

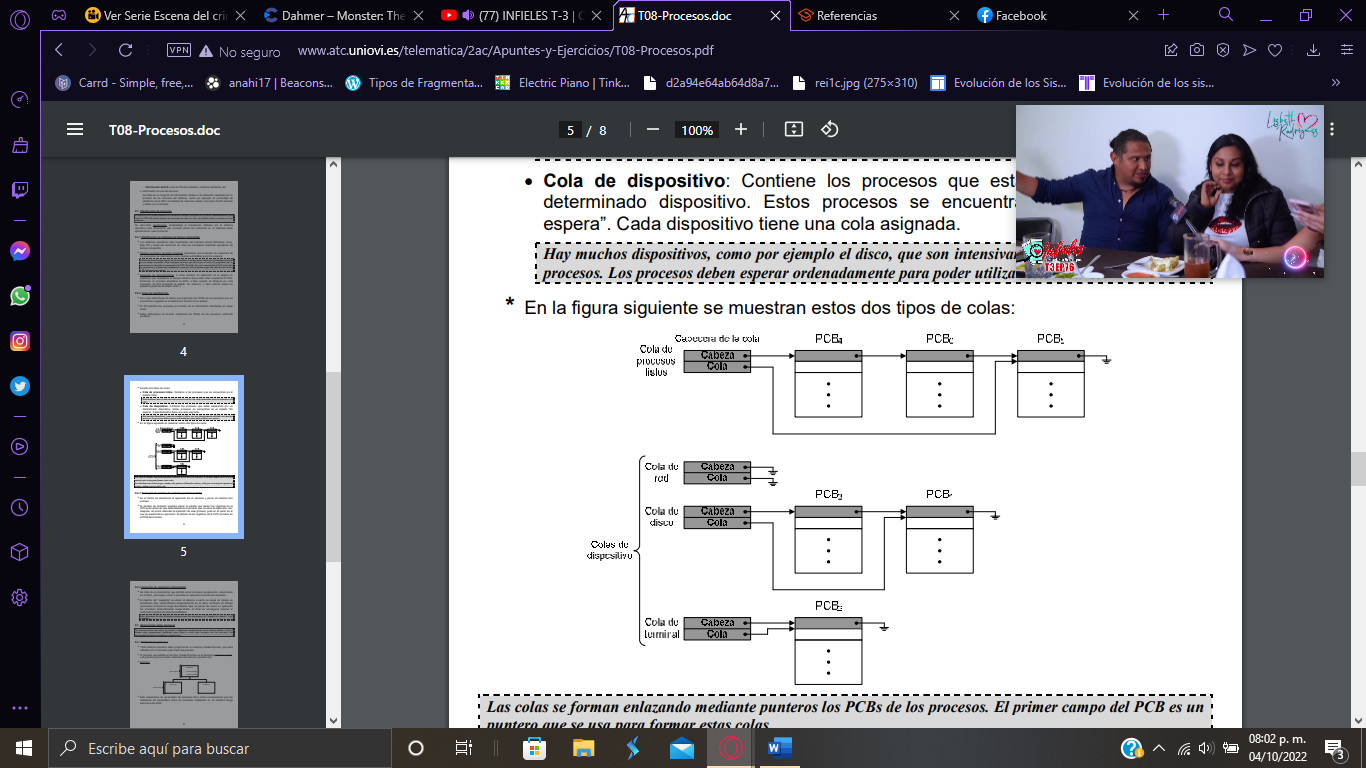
***Colas de planificación***

Las colas son estructuras de datos que organizan PCB de los procesos que están cargados en el sistema en función de su estado. Las estructuras se forman enlazando los PCB de los procesos mediante punteros.

El sistema operativo planifica los procesos en funciones de la información mantenida en estas colas.

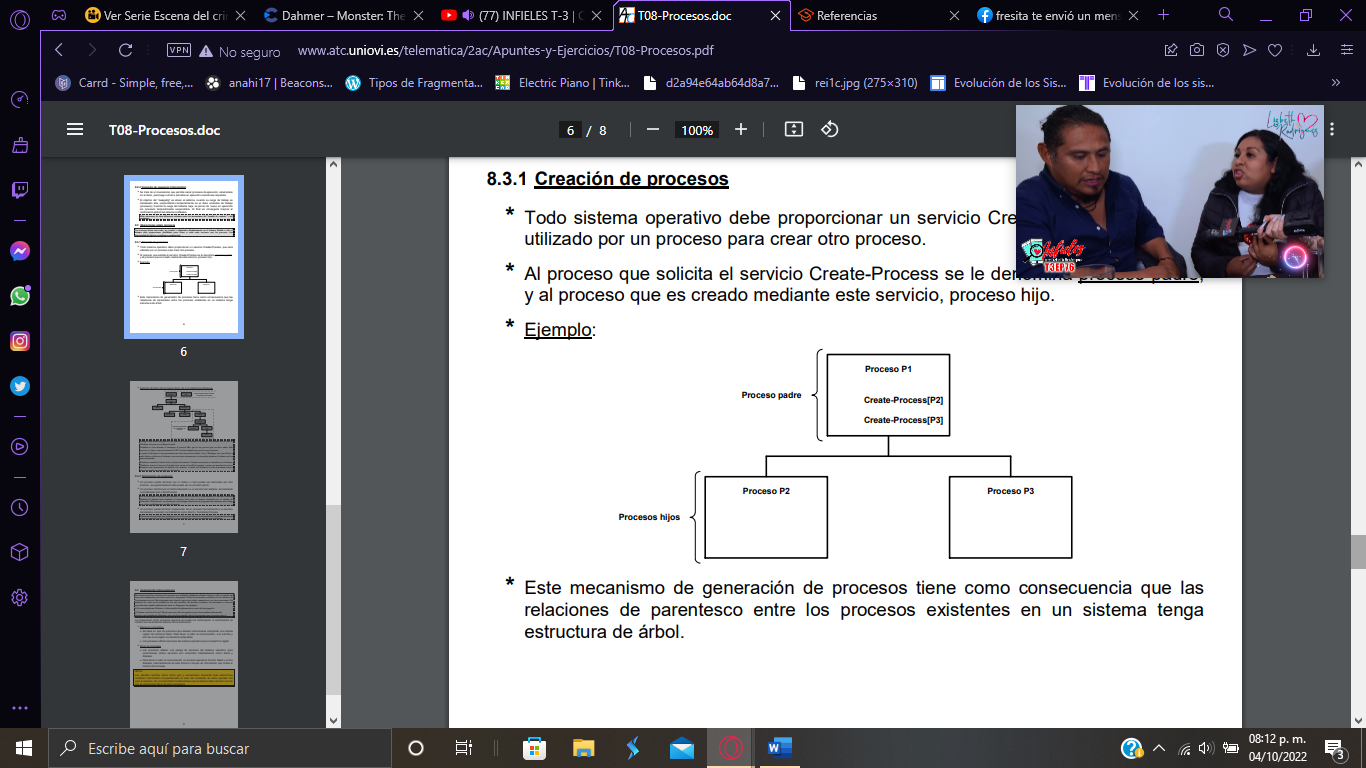
Aquí existen dos tipos de colas:

* Cola de procesos listos: contiene procesos que se encuentran en el estado “listo”.
* Cola de dispositivo: contiene los procesos que están esperando por un determinado dispositivo. Los procesos son encontrados en el estado de “espera”. Cada dispositivo tiene una cola asignada.

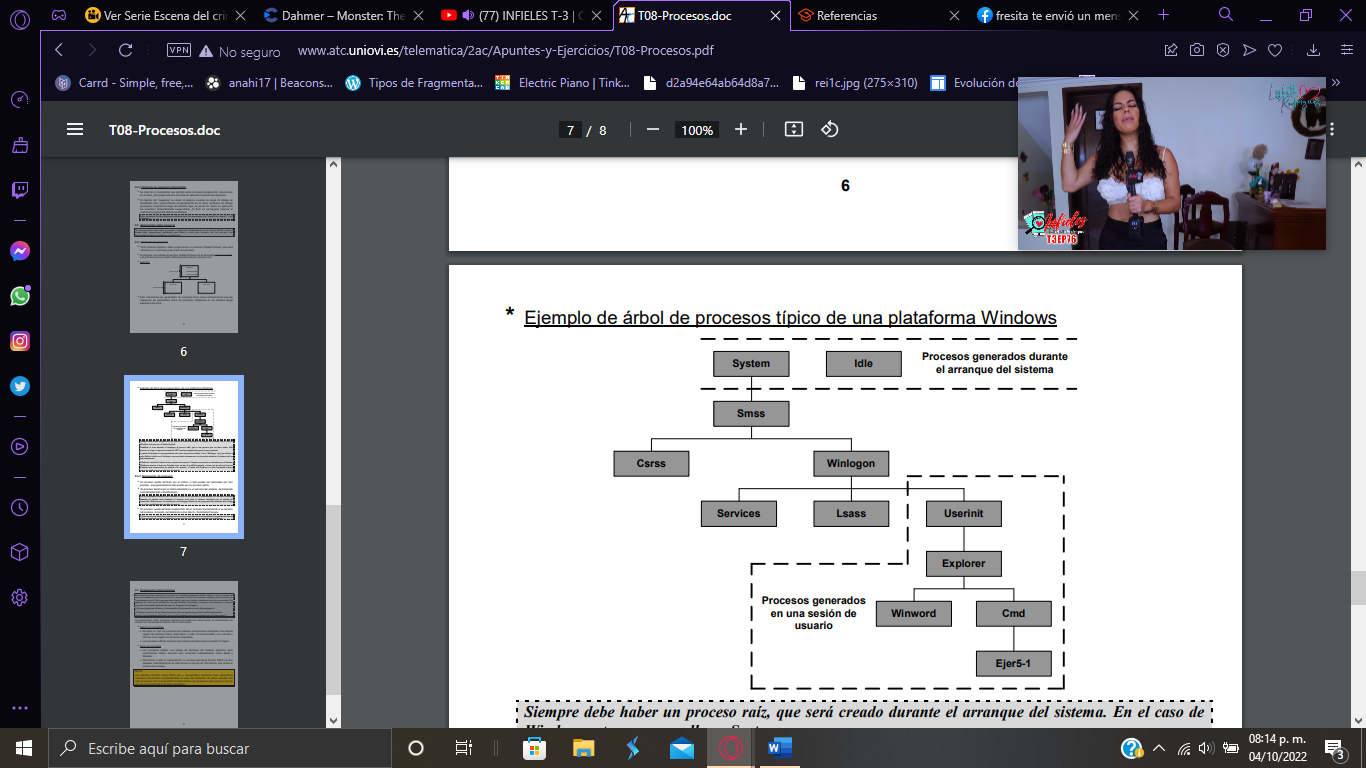


***Creación de procesos***

El SO debe proporcionar un servicio create-process, que se utilizara por un proceso para poder crear otro proceso. El proceso que necesita el servicio create-process es denominado como el padre y el proceso que es creado mediante este servicio es el hijo.



Este tipo de mecanismo de generación de procesos tiene como consecuencia que las relaciones de parentesco entre distintos procesos existentes en un sistema tengan la estructura de árbol.



**Hilos**

Los hilos son una unidad básica de utilización en el CPU, esta contiene un id de hilo, el propio programa counter, conjunto de registros, y una pila. Se representa a nivel del SO con una estructura llamada thread control block. Los hilos suelen compartir con otros hilos que pertenezcan al mismo proceso la sección de código, la selección de datos, etc.

**Virtualización**

Esta ejecuta los procesos en una maquina física de otras máquinas virtuales, en pocas palabras, se consigue que una computadora pueda trabajar como varias computadoras juntas de manera que el software o hipervisor separa el sistema operativo de las aplicaciones hardware.

**Clientes**

Los clientes son los que realizan peticiones a otro programa, al servidor, quien da la respuesta. También se puede aplicar a programas que se ejecutan sobre una sola computadora, aunque la manera mas ventajosa es un SO multiusuario distribuido a través de una red de computadoras.

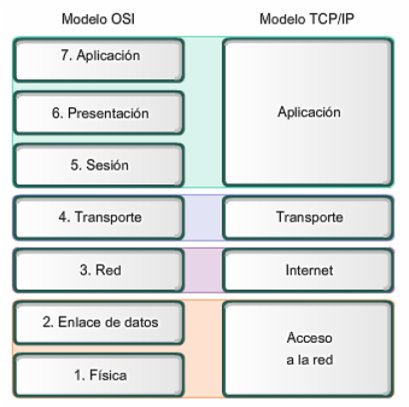
**Migración de código**

Es un movimiento de programación de un sistema a otro. Existen tres niveles distintos de la migración de código con complejidad, un costo y riesgo creciente. La migración simple implica el paso a un idioma de versión nueva. Un segundo nivel de migración mas complicado seria pasar a un lenguaje de programación diferente. La migración a una plataforma u SO completamente nuevo es el tipo de migración más complejo.

## **2.2 Comunicación**

Cuando hablamos de la comunicación tenemos que tener muy en cuenta cual de estos tipos tenemos, esto debido a que no es la misma forma de comunicar un sistema distribuido y uno unificado, esto se debe de forma principal en como es que se ejecutan y realizan los diferentes procesos dentro dichos sistemas, mientras que por un lado tenemos los sistemas distribuidos, donde pequeñas partes son recibidas por múltiples ordenadores para al final juntarlas y tener un resultado final, por el otro lado tenemos a un solo ordenador trabajando en la elaboración, de dicho proceso, con esto no significa que uno sea mejor que otro, si no que depende mucho en cuanto a las circunstancias y las necesidades con las que contemos y que es lo que queramos realizar’

Dicho esto podemos encontrar diversos métodos para establecer conexiones entre nuestros equipos, uno de ellos es el protocolo con capas, este se utiliza ya que en los sistemas distribuidos existe la falta de memoria compartida, ya que al ser diferentes equipos, estos no pueden estar conectados a una sola memoria, tanto por la limitación de conexiones, como la localización, ya que si estamos usando un sistema distribuido, podemos tener un equipo A en un país y un equipo B en otro, imposibilitando el uso de una memoria compartida, es con ello cuando se hace uso de la transferencia de mensajes

Cuando se empieza el proceso de envió de mensajes de una computadora A hacia una computadora B, este empieza a construir un mensaje en su propio espacio de direcciones, tras esto el sistema ejecuta una llamada para que nuestro sistema operativo busque el mensaje y lo envié a través de la red hacia el equipo B, este proceso puede ser multidireccional, es decir, no es necesario hacer un envió uno a uno, si no que nos permite enviar uno a muchos, siempre y cuando así lo queramos

Por últimos y esto es más como un paréntesis, tenemos que saber que, para evitar conflictos entre estas redes, tanto A como B deben coincidir en el significado de los bits que se envían

De igual forma podemos diferenciar distintos modelos, como lo son el OSI o el TCP/IP

Empezando por el modelo OSI, este esta diseñado para permitir la comunicación de los sistemas abiertos, esto se refiere a que cualquier equipo puede hacer conexión, siempre y cuando cuente con las reglas estándares para comunicarse, además encontramos que establece el formato, contenido y significado de los mensajes recibidos y enviados, estos son los que constituyeron los protocolos especializados para la comunicación y como es que esta debe desarrollarse, del mismo modo es que este mismo protocolo distingue entre dos tipos generales, con ello, antes de intercambiar los datos el emisor y el receptor deben: Establecer en forma explícita una conexión, esclarecer el protocolo a utilizar, Al finalizar, deben terminar la conexión esto para evitar vulnerabilidades del sistema,

El modelo cliente- servidor socket

Este modelo es incluso más “sencillo” ya que permite establecer una conexión sin la necesidad de estar conectados a una red, esto debido a que estamos haciendo una conexión directa entre las computadoras, lo bueno de este modelo es que no representa una complicación como lo haría OSI o el TCP/IP, de este modo quien ejecuta los requerimientos necesarios para establecer a conexión, es el servidor, con ello es que nos brinda ciertos beneficios, los cuales están entre ellos, que si este protocolo falla, se nos regresan datos de la ejecución, además de no establecer una conexión hasta que se utilice dicha conexión y por si fuera poco la pila de protocolo es aún más corta que en otros protocolos y por lo tanto es mucho más eficiente

Direccionamiento en C

En este protocolo es esencial el conocer la dirección a la cual se está mandando el mensaje, esto nos facilita el hecho de que elimina la ambigüedad acerca de quién es el receptor, pudiendo identificarlo en todo momento, sin embargo, nos trae un gran problema, el cual consiste en la identificación de problemas

Para solucionar esto, podemos encontrar distintas formas para arreglarlo, entre ellas, está el hecho de no necesitar de coordenadas globales, del mismo modo pueden repetirse los nombres de los procesos en distintas máquinas, del mismo modo podemos utilizar una variable local de forma machine.local en vez de machine.process, este último método lo que nos permite es mandar un numero entero aleatorio de 16 o 32 bit, tras ello el sistema inicia una llamada para indicar al núcleo que se desea escuchar al local-id, y tras ello el núcleo sabe a cuál proceso debe darle el mensaje

Comunicación RPC

Por su nomenclatura (Remote procedure call) y traducido como llamada procedural remota, este mecanismo nos permite conectarnos de una forma un tanto distinta, ya que esta lo que realiza es una ejecución, pero no en una maquina propia, si no en una maquina remota, la cual nos permite introducir comandos y ejecuciones desde remoto, permitiéndonos optimizar tiempo y esfuerzo

Para ello tenemos que tener en cuenta que es lo que realmente queremos realizar y desde que tipo de red lo estamos realizando, ya que estas pueden ser redes PAN, LAN, WAN y MAN, Cada una de las conexiones remotas resulta mucho más complicada que la anterior, por lo que la más sencilla es PAN y la más complicada es WAN, esto debido a que tenemos que identificar de forma muy clara, cual es la computadora con la cual nos estamos comunicando, ya que en una red PAN es mucho más sencillo debido a que existe un menor número de equipos cosa que no es así con las WAN donde estamos prácticamente conectadas a internet

## **2.3 Nombres**

Los nombres juegan un papel muy importante en todos los sistemas de cómputo. Se utilizan para compartir recursos, para identificar entidades de manera única, para hacer referencia a ubicaciones, y más. Un asunto importante con respecto a los nombres es que un nombre puede resolverse para la entidad a la cual hace referencia. De esta manera, la resolución de nombres permite a un proceso acceder a la entidad nombrada. Para resolver los nombres, es necesario implementar un sistema de nombres. En sistemas distribuidos y sistemas no distribuidos, la diferencia entre nombres radica

en la forma en que se implementan los sistemas.

En un sistema distribuido, con frecuencia la implementación de un sistema de nombres se distribuye por sí sola a través de múltiples máquinas. La manera en que se lleva a cabo esta distribución juega un papel importante en lo que se refiere a la eficiencia y la escalabilidad del sistema de nombres.

En este capítulo, nos concentramos en tres diferentes e importantes formas de utilizar los

nombres en los sistemas distribuidos.

Primero, después de explicar algunos temas generales relacionados con los nombres, daremos un vistazo a la organización e implementación de nombres amigables con el usuario. Ejemplos típicos de dichos nombres incluyen aquellos implementados para los sistemas de archivos y para la World Wide Web. La construcción de sistemas de nombres escalables a nivel mundial es una preocupación primordial para este tipo de nombres.

Segundo, los nombres se utilizan para ubicar a las entidades en una vía independiente de su ubicación actual. Como podemos darnos cuenta, los sistemas de nombres para nombres amigables con los usuarios no están configurados de manera particular para soportar este tipo de entidades de seguimiento.

La mayoría de los nombres ni siquiera dan clave alguna con respecto a la ubicación de la entidad. Se requieren organizaciones alternativas tales como las utilizadas para la telefonía móvil, en donde los nombres son identificadores independientes de la ubicación, y aquellas empleadas para las tablas hash distribuidas. Por último, con frecuencia las personas prefieren describir entidades por medio de distintas características, ello provoca una situación en la cual, necesitamos resolver una descripción mediante los atributos apropiados para una entidad que se adhiera a dicha descripción. Este tipo de resolución de nombres es muy difícil y le pondremos atención especial.

**NOMBRES PLANOS**

Líneas arriba explicamos que los identificadores son convenientes para representar a las entidades de manera única. En muchos casos, los identificadores son simplemente cadenas aleatorias de bits, a las cuales nos referimos convenientemente como nombres no estructurados o planos. Una propiedad importante de tales nombres es que no contienen información alguna con respecto a la ubicación del punto de acceso de su entidad asociada. A continuación, daremos un vistazo a la forma en que se pueden resolver los nombres, o, de manera equivalente, cómo podemos localizar una entidad cuando solamente proporcionamos su identificador.

**Tipos de nombres**

Hay tres enfoques principales para los esquemas de nominación.

El primer enfoque y el más sencillo, los archivos se nombran con una combinación del nombre de su anfitrión y su nombre local, lo que garantiza un nombre único dentro de todo el sistema.

El segundo enfoque popularizado por el sistema de archivos de red (NFS, Network File System), ofrece una forma de unir directorios remotos a directorios locales, lo que da la apariencia a un árbol de directorios coherentes.

El tercer enfoque es la estructura más compleja y difícil de mantener en la NFS, ya que cualquier directorio se puede unir a cualquier árbol de direcciones locales y la jerarquía resultante puede estar poco estructurada.

## **2.4 Sincronización**

**Sincronización:**Para poder ejecutar tareas en ambientes distribuidos, es importante que el orden en las secuencias de operaciones de procesos sea estricto y universal. De ahí la necesidad de poder ejecutar procesos distribuidos con una certeza de que iniciarán al mismo tiempo respecto a una unidad de tiempo general. Mantener una Base de Tiempo a través de una red impone ciertos problemas tecnológicos importantes, pero tal vez el más relevante es la propagación, sobre todo en aquellos sistemas que procesan grandes cantidades de datos a alta velocidad.

El término sincronización se define como la forma de forzar un orden parcial o total en cualquier conjunto de eventos, y es usado para hacer referencia a tres problemas distintos pero relacionados entre sí:

1.-La sincronización entre el emisor y el receptor.

2.-La especificación y control de la actividad común entre procesos cooperativos.

3.-La serialización de accesos concurrentes a objetos compartidos por múltiples procesos.

El concepto de sincronización existe también en los S.O.C. de implementar porque existe un reloj único, accesible  todos los recursos. En cambio en un S.O.D. se debe administrar establecer artificialmente este mecanismo, de ahí surgen los diversos algoritmos cuyos enfoques son:

a) Tiempo Global Distribuido.

b) Tiempo Global por Promedio Localizado.

c) Exclusión Mutua.

d) Algoritmos de Elección de Coordinador.

e) Algoritmos de Anillo Token Ring (no confundir con Topología Token Ring, es otra cosa).

f) Otros enfoques (combinaciones o alternativas).

La sincronización de relojes en un sistema distribuido consiste en garantizar que los procesos se ejecuten en forma cronológica, y a la vez respetando el orden de los eventos dentro del sistema.

**Algoritmos de sincronización:**En los últimos años se ha abordado el problema de la sincronización de relojes de computadoras desde diferentes puntos de vista. Esto puede reflejarse al comparar varios de los algoritmos más populares que resuelven el problema. En todos los casos, es necesaria la comunicación que establece la misma referencia de tiempo para todas las entidades; pero para ello no basta la difusión de la referencia, necesariamente debe haber una especie de control de recursos similar al control de acceso a usuarios o a un mecanismo de direccionamiento en una topología de red; en el caso de faltar una referencia de tiempo universal se toma un consenso y elección entre todos los sistemas de tiempo involucrados.

**Algoritmo de Lamport:**El algoritmo de Lamport es uno de los primeros propuestos para la sincronización de sistemas distribuidos y se basa en la relación “sucede antes”, más la utilización de los mensajes entre las computadoras como indicadores precisos de esta relación. Más específicamente, un mensaje no puede ser recibido antes de ser enviado y, por lo tanto, si se tienen marcas de tiempo de los envíos de los mensajes se puede verificar si el tiempo actual es coherente con la definición de la relación “antes de”. Para ello se definen relojes lógicos.

Este algoritmo soluciona el problema de la escalabilidad, ya que solo exige un mensaje entrante y uno saliente para sincronizar cada máquina, pero no tiene un orden total de los eventos que suceden entre diferentes computadoras, excepto los relacionados directamente con envío y recepción de mensajes.

**Algoritmo de Cristian:**Este algoritmo utiliza los conceptos de sincronización interna y externa. La sincronización interna se refiere a mantener un grupo de relojes sincronizados, no importa qué hora tengan respecto al uso horario, pero que en el grupo sea la misma, o con un margen de diferencia. Cristian describe el método como probabilístico debido a que la sincronización se consigue solo si el tiempo de respuesta es suficientemente corto comparado con la precisión requerida. En otras palabras el algoritmo se basa en la uniformidad en la propagación de la señal de sincronía.

**Algoritmo Berkeley:**Este es un algoritmo basado en un reloj central, el cual mide las desviaciones de los relojes de todos los elementos de la red, y les obliga a compensar con ajustes de retraso del reloj a los equipos que está adelantados, y con ajustes de adelanto en el reloj a los equipos que están atrasados; a su vez el reloj central se ajusta en el valor promedio de la hora de todos los equipos y establece la hora a la que todos se sincronizarán con su respectivo ajuste (atrasarse o adelantarse).

**Relojes físicos:**Cuando utilizamos una base de tiempo Universal, se dice que utilizamos Relojes Físicos. Son útiles en aplicaciones que se manejan eventos secuenciados a alta velocidad, de tal suerte que no hay recursos disponibles para organizar la sincronización; En ciertos sistemas (por ejemplo, los sistemas de tiempo real), es importante la hora real del reloj. Para estos sistemas se necesitan relojes físicos externos. Por razones de eficiencia y redundancia, por lo general son recomendables varios relojes físicos, lo cual implica dos problemas:

Los relojes físicos deben ser iguales (estar sincronizados), no deben desviarse del tiempo real más allá de cierta magnitud. Debido a la distorsión del tiempo y las derivas en los relojes de los distintos CPU's, en ciertos sistemas es importante la hora real del reloj, por lo que se precisan relojes físicos externos (más de uno) y se deben sincronizar con los relojes del mundo real.

**Relojes lógicos:**La idea de un reloj lógico consiste en crear un sistema de convergencia del tiempo mediante la medición de las derivas, de tal suerte que la noción de tiempo universal se sustituye por la noción de un tiempo global auto-ajustable.

Los relojes lógicos representan una relación de orden parcial, y funcionan mediante la alteración del conteo de tiempo para mantener la sincronización mediante variables que afectan la hora de referencia. El reloj lógico  establece un orden total entre eventos si se añade el número del procesador, pero esto implica pagar un precio por la comunicación y por mantener el registro la tabla de CPU's.

**Relojes Lógicos**

Las computadoras poseen un circuito para el registro del tiempo conocido como dispositivo reloj.

Es un cronómetro consistente en un cristal de cuarzo de precisión sometido a una tensión eléctrica que:

* Oscila con una frecuencia bien definida que depende de:

o Al forma en que se corte el cristal.

o El tipo de cristal.

o La magnitud de la tensión.

* A cada cristal se le asocian dos registros:

o Registro contador.

o Registro mantenedor.

* Cada oscilación del cristal decrementa en “1” al contador.
* Cuando el contador llega a “0”:

o Se genera una interrupción.

o El contador se vuelve a cargar mediante el registro mantenedor.

* Se puede programar un cronómetro para que genere una interrupción “x” veces por segundo.
* Cada interrupción se denomina marca de reloj.

Para una computadora y un reloj:

* No interesan pequeños desfasajes del reloj porque:

o Todos los procesos de la máquina usan el mismo reloj y tendrán consistencia interna.

o Importan los tiempos relativos.

Para varias computadoras con sus respectivos relojes:

* Es imposible garantizar que los cristales de computadoras distintas oscilen con la misma frecuencia.
* Habrá una pérdida de sincronía en los relojes (de software), es decir que tendrán valores distintos al ser leidos.

Para ciertos algoritmos lo que importa es la consistencia interna de los relojes:

* No interesa su cercanía particular al tiempo real (oficial).
* Los relojes se denominan relojes lógicos.

Los relojes físicos son relojes que:

* Deben ser iguales (estar sincronizados).
* No deben desviarse del tiempo real más allá de cierta magnitud.

Para sincronizar los relojes lógicos, Lamport definió la relación ocurre antes de (happens-before):

* Si “a” y “b” son eventos en el mismo proceso y “a” ocurre antes de “b”, entonces “a –> b” es verdadero.
* “Ocurre antes de” es una relación transitiva:

o Si “a –> b” y “b –> c”, entonces “a –> c”.

* Si dos eventos “x” e “y” están en procesos diferentes que no intercambian mensajes, entonces “x –> y” no es verdadero, pero tampoco lo es “y –> x”:

o Se dice que son eventos concurrentes.

### Sincronización.

Ya vimos en teoría que:

Sincronización:

* + Un mesaje no se puede recibir antes de haberlo enviado.
  + Tipos de sincronización del remitente:
    - * Send bloqueante: el proceso queda bloqueado hasta que el destinatario lee el mensaje.
      * Send no bloqueante: el proceso continúa inmediatamente.
  + Tipos de sincronización del destinatario:
    - * Receive bloqueante: el proceso se queda bloqueado si no hay ningún mensaje en su buzón.
      * Receive no bloqueante: el proceso no se bloquea.
      * Comprobación de llegada: el proceso mira a ver si hay mensajes en su buzón, pero no los lee.

En el caso de los buzones IPC de UNIX, el envío es siempre NO bloqueante. No hay ningún mecanismo para que el proceso se bloquee hasta que un destinatario lea su mensaje. La recepción es, por defecto, bloqueante.

Sincronización

La sincronización de procesos en los sistemas distribuidos resulta más compleja que en los centralizados, debido a que la información y el procesamiento se mantiene en diferentes nodos. Un sistema distribuido debe mantener vistas parciales y consistentes de todos los procesos cooperativos.

No es posible reunir toda la información sobre el sistema en un punto y que luego algún proceso la examine y tome las decisiones.

En general los algoritmos distribuidos tienen las siguientes propiedades:

1)- la información relevante está repartida entre muchas máquinas

2)- los procesos toman decisiones basadas solamente en la información local disponible

3) – debería poder evitarse un solo punto que falle en el sistema

4)- no existe un reloj común u otro tiempo global exacto

Si para asignar un recurso de E/S un proceso debe recolectar información de todos los procesos para tomar la decisión esto implica una gran carga para este proceso y su caída afectaría en mucho al sistema.

Idealmente, un sistema distribuido debería ser más confiable que las máquinas individuales. Alcanzar la sincronización sin la centralización requiere hacer cosas en forma distinta a los sistemas operativos tradicionales.

# **Conclusiones**

En conclusión, esta segunda unidad de sistemas operativos 2 fue uno de los temas más interesantes, ya que nos llamó mucho la atención lo importante que es el papel de la comunicación y procesos entre máquina-humano, ya que sin esto no podríamos hacer las maravillas que actualmente hacemos. Aun así, vemos que hay distintos de procesos y distinta comunicación, lo más destacable es la forma en que cada uno trabaja, en cómo se aplican los procesos para cada situación en la que se requiere. Sin olvidarnos del papel de cliente en este tema es que, sin él, las peticiones a otro programa o servidor no se podrían hacer o la migración de código, donde podemos mover nuestro programa hacia a otras plataformas logrando pasar de un lenguaje de programación a otro, eso sí, se muestran algunas complicaciones y los distintos niveles que existen.

Se dio un gran vistazo en forma de que los distintos procesos cumplen su crucial papel en los sistemas distribuidos desde la perspectiva de un sistema operativo, la administración y la calendarización de los procesos (dependiendo al que se trate). Dando a entender que lo más importante fue como nos permite la construcción de clientes y servidores de modo a comunicación/procesos.

# **Herramientas y recursos**

* Imágenes de Google
* Power Point para crear la exposición
* Youtube
* El documento de nuestra investigación del tema
* Scribbr para el APA
* Equipo de cómputo.

# **Bibliografía**

Procesos. (s. f.). Recuperado 4 de octubre de 2022, de <https://www.ibm.com/docs/es/aix/7.2?topic=processes->

Procesos. (2017, 11 agosto). pdf. Recuperado 4 de octubre de 2022, de <http://www.atc.uniovi.es/telematica/2ac/Apuntes-y-Ejercicios/T08-Procesos.pdf>

Hilos. (s. f.). pdf. Recuperado 4 de octubre de 2022, de <https://www.fing.edu.uy/tecnoinf/maldonado/cursos/so/material/teo/so05-hilos.pdf>

Profile, V. M. C. (s. f.). *Unidad II*. Recuperado 4 de octubre de 2022, de http://sistemasoperativos2beresainz.blogspot.com/p/unidad-2-comunicacion-en-los-sistemas.html

*Unidad 2: Procesos y comunicación*. (2016, 24 septiembre). Sistemas operativos ii. Recuperado 4 de octubre de 2022, de <https://soperativos2.wordpress.com/2016/09/22/unidad-2-comunicacion-en-los-sistemas-operativos-distribuidos/>

*Sistemas Distribuidos* (2.a ed.). (2008, 1 enero). Pearson Educación.

*Sistemas Operativos II - Segunda sesión*. (s. f.). Recuperado 5 de octubre de 2022, de <http://avellano.usal.es/%7Essooii/sesion2.html>

*Unidad 2 Comunicación en los sistemas operativos distribuidos*. (s. f.). Recuperado 5 de octubre de 2022, de <http://sistemasoperativos2-u2.blogspot.com/2016/09/sistemas-operativos-2-unidad-2.html>

*Unidad 2: Procesos y comunicación*. (2016, 24 septiembre). Sistemas operativos ii. Recuperado 5 de octubre de 2022, de <https://soperativos2.wordpress.com/2016/09/22/unidad-2-comunicacion-en-los-sistemas-operativos-distribuidos/?fbclid=IwAR05JpkXjfYtJrqRLIOHbIJazGQJ1KmkRH96v3qfGQ4CVffHmyhvp4O6sSA>