Tarea 2: Sistemas Operativos y Software de un Sistema Informático

Francisco Javier Sueza Rodríguez

8 de diciembre de 2022

Centro:	IES Aguadulce
Ciclo Formativo:	Desarrollo Aplicaciones Web (Distancia)
Asignatura:	Sistemas Informáticos
Tema:	Tema 2 - Software de un Sistema Informático

Índice

1. Caso Práctico

Ada, fundadora de la empresa AguadulSoft, ha visto una oportunidad excepcional de promocionar la empresa aprovechando una exposición temporal denominada "Historia del software: Sistemas Operativos"que se va a alojar en la localidad.

La empresa, a la que se le ha pedido colaboración, realizará un análisis de los distintos Sistemas Operativos así como de otros tipos de software que han surgido a lo largo de la historia.

También va a averiguar qué licencias se utilizan en los sistemas operativos y distintos programas que se utilizan en la actualidad.

2. Actividades

2.1. Actividad 1: Tipos de Aplicaciones Informáticas

Indica, justificando su uso, dos ejemplos de aplicaciones de propósito general y otros dos de propósito específico para cada uno de los siguientes entornos productivos:

- Agencia de Viajes
- Agencia de Publicidad y Marketing

2.1.1. Solución

Estas dos empresas usaran tanto aplicaciones de propósito general como específico de su propio campo. En la siguiente lista se muestran dos ejemplos de cada tipo de software para cada una de las empresas.

- **Agencia de Viajes**
  - **Aplicaciones de Propósito General**
    - **Hoja de Cálculo**: es muy probable que usen hojas de cálculo, ya que los viajes tienen muchos complementos y añadidos, y una hoja de cálculos será necesaria para calcular el precio final del paquete elegido por el cliente. Probablemente usen **Excel** o alguna alternativa libre como **Libreoffice Calc** o **Calc de Openoffice**.
    - **Herramientas de Acceso de Bases de Datos**: también deberán emplear alguna herramienta de acceso y gestión de bases de datos, para poder consultar y administrar los destinos disponibles, precios, paquetes, etc...
  - **Aplicaciones de Propósito Específico**
    - **Aplicaciones Ad-hoc Especializadas**: para la gestión de la agencia de viajes, los clientes, billetes, comunicación con terceros (hoteles, guías turísticos,..) se utilizará algún **CRM** especializado, como por ejemplo **CRM Travel**.
    - **Herramientas de Administración de Bases de Datos**: además de un CRM, se deberá usar una bases de datos y herramientas para su administración, ya que este tipo de aplicaciones necesitan una base de datos como soporte. Así, podrá centralizarse toda la información de la empresa y permitirle al CRM hacer una gestión adecuada de toda esta información. Una de estás aplicaciones podría ser **OracleDB**, una de las más empleadas a nivel mundial, o **MySQL**, otra de la más usadas también.

- **Agencia de Publicidad y Marketing**
  - **Aplicaciones de Propósito General**
    - **Generador de Presentación**: una herramienta necesaria para exponer información será un generador de presentación, bien para presentar la estrategia de una nueva campaña y sus elementos claves o para presentar datos relativos a una campaña ya realizar, entre otras cosas. Podrían usar **Microsoft Powerpoint**, por ejemplo.
    - **Herramientas para la Comunicación**: seguramente también usen herramientas para la comunicación rápida y directa entre los diferentes departamentos, ya que el trabajo de esta empresa debe estar bien coordinado entre los diferentes equipos que conformen un mismo proyecto. Este software podría ser **Slack**, uno de los más usados.
  - **Aplicaciones de Propósito Específico**
    - **Herramientas de Diseño Gráfico y Maquetación**: esta herramienta será básica para el diseño de campañas publicitarias, especialmente aquellas pensadas para cartelería o cualquier medio impreso. Uno de los más usados es **Adobe Photoshop**, aunque también podrían usar alternativas como libres como **Gimp** o **Inkscape**, o incluso usar la herramienta colaborativa **Figma**, cuyo uso se esta expandiendo recientemente.
    - **Herramientas de Análisis de Datos**: la empresa necesitará diferentes herramientas de análisis de datos para conocer el impacto de las campañas, punto clave para saber si han tenido éxito o no. Esto requiere de herramientas que sean capaces de procesar grandes cantidades de datos, conocidas como **Big Data**. Una de las empleadas es **HubSpot**, especializada en el análisis de datos de marketing.

2.2. Actividad 2: Licencias de Software

Indica para cada una de las siguientes aplicaciones, el tipo de licencia que utiliza, intentando dar en términos generales si se trata de una licencia de software con código abierto, con código cerrado o de dominio público (sin licencia), y de manera específica la licencia exacta que utiliza.

Por ejemplo: El programa de edición de gráficos vectoriales llamado Inkscape, usa una licencia GPLv3+, que es una licencia de software con código abierto, no permisiva, de tipo copyleft fuerte (software libre).

Las aplicaciones son las siguientes:

- Microsoft Office 2021.
- Google Chrome.
- VLC Media Player.
- Avast Free Antivirus.
- Adobe Acrobat Pro.

2.2.1. Solución

En esta actividad vamos a indicar bajo que licencia están desarrolladas diferentes aplicaciones, especificando si la licencia es libre y propietaria, así como sus principales características.

- **Microsoft Office 2021**: esta suite ofimática se encuentra bajo la licencia **Microsoft CLUF**, una **licencia propietaria** de tipo **EULA**, que no permite distribución del software ni incluye su código fuente, además, solo permite su uso para un único usuario. [1]
- **Google Chrome**: este navegador se encuentra bajo los **Términos de Servicio Adicionales de Chrome y Chrome OS** [2], que establece una **licencia propiertaria freeware** para los binarios de Google Chrome, que permite su uso de forma gratuita y por tiempo ilimitado.

  En cambio, este navegador esta basado en **Chromium** [3], por lo que la mayoría de sus componentes software están bajo la licencia **3-clause BSD**, una licencia de **código abierto** de tipo **permisiva** [4], que garantiza el uso y distribución del software y su código fuente.
- **VLC Media Player**: este reproductor de vídeo esta bajo una **licencia de código abierto**, en concreto, esta licencia es la **GPLv2.1+** [5], una licencia de tipo **copyleft fuerte** que garantiza la distribución tanto del software como de su código fuente siempre cuando se conserve la licencia en los productos derivados.
- **Avast Free Antivirus**: la versión gratuita de **Avast Antivirus** usa una **licencia propietaria**, la **Avast EULA**, de tipo **freeware** y **EULA**. Su uso es gratuito por tiempo ilimitado para un solo usuario, prohibiendo explícitamente su uso para uso comercial o su uso por parte de cualquier compañía o entidad gubernamental. [6]
- **Adobe Acrobat Pro**: los productos de Adobe se rigen por la licencia **Adobe EULA**, una **licencia propietaria** de tipo **EULA**. La licencia tiene dos modalidades, una **freeware** y otra **comercial**. En este caso, la versión **Pro** de Adobe Acrobat usa la modalidad **comercial**. [7]

2.3. Actividad 3: Sistemas operativos: Última versión, requisitos hardware,...

Realiza la siguiente tabla añadiendo una fila por sistema operativo con la última versión existente de:

- Windows 11 (canal de disponibilidad general).
- Ubuntu Desktop LTS (última versión con soporte de larga duración).
- iOS (para iOS, en lugar de requisitos hardware", incluye una lista con los dispositivos que soportan la última versión).
- iOS (para iOS, en lugar de requisitos hardware", incluye una lista con los dispositivos que soportan la última versión).

Sistema Operativo	Última versión	Requisitos hardware			Licencia	Dispositivos en los que normalmente se instala
		Procesador	RAM	Espacio de almacenamiento		
Windows 11						
Ubuntu Desktop LTS						
iOS		*(lista de dispositivos compatibles)*				
Android						

Figura 2.1: Tabla para rellenar sobre SOs

2.3.1. Solución

En este punto vamos a ver información básica sobre los diferentes sistemas operativos más extendidos en la actualidad. En la siguiente figura se muestra una tabla con toda la información pedida.

Dentro de la sección de requisitos de hardware se incluirán los requisitos mínimos, no los recomendados. El colo de la tabla también puede diferir de la que se muestra en el enunciado, pero el contenido es el solicitado.

Sistema Operativo	Última Versión	Requisitos Hardware			Licencia	Dispositivos en los que se instala normalmente
		Procesador	RAM	Almacenamiento		
Windows 11	22h2	1 Ghz con 2 o más núcleos y de 64 bits	4 GB	64Gb	Microsoft CLUF (Propietaria EULA)	Desktops, Laptops, MS Surface Pro
Ubuntu Desktop LTS	22.04.1	2 Ghz dual core	4 GB	25 GB	GPL (Copyleft Fuerte)	Desktop, Laptops
iOS	16.1.2	iPhone SE, iPhone 8 (Plus), iPhone X (XR, XS y XS Max). iPhone 11 (Pro y Pro Max), iPhone 12 (Mini, Pro y Pro Max), iPhone 13 (Mini, Pro y Pro Max), y iPhone 14 (Pro, Pro Max y Plus)			IOS Software License (Propietaria EULA)	Móviles iPhone
Android	13	2.9 Ghz octa-core	12 GB	128 GB	Apache 2.0 y GPL (Libre permisiva y Coplyleft Fuerte	Móviles y Tablets

Figura 2.2: Tabla características SOs

Para el apartado de **Android**, se ha incluido el teléfono **Samsung Galaxy S21 Ultra**. Actualmente no hay móviles que salgan con Android 13, ya que esta versión es muy reciente (Agosto de 2022), pero si hay muchos que se han actualizado a esta versión desde la versión 12, este Samsung es uno de ellos.

2.4. Actividad 4: Arquitectura Interna de un Sistema Operativo

Explica en qué consisten las siguientes arquitecturas de sistemas operativos:

- Monolítica.
- Microkernel.
- Híbrida.

Indica, además, un ejemplo de SO para cada arquitectura.

2.4.1. Solución

En este punto vamos a explicar diferentes arquitecturas de Kernel, explicando un poco sus características y poniendo un ejemplo de un sistema operativo que las use.

- **Arquitectura Monolítica**

  En este tipo de arquitectura **todas las funciones** se implementan **dentro del núcleo**, es decir, todos los servicios del sistema operativo se ejecutan dentro de hilo principal del Kernel. También se incluyen dentro del núcleo **todos los drivers** necesarios para que funcionen los diferentes dispositivos. Este tipo de arquitectura ha sido empleado tradicionalmente por los sistemas tipo Unix. [8]

  Algunos desarrolladores, como Ken Thompson, mantienen que es "más fácil de implementar" que otras arquitecturas, como los microkernels, aunque son núcleos más difíciles de mantener, principalmente por la dependencia entre los componentes del sistema. [9]

  Algunos ejemplos de este tipo de kernel son los sistemas **GNU/Linux**, **BSD** (FreeBSD, OpenBSD, NetBSD), y los sistemas basados en **UNIX System V**, como **Solaris** o **HP-UX**. El sistema **MS-DOS** también usaba esta arquitectura.

- **Microkernel**

  En este tipo de arquitectura, muchas de las **funcionalidades** que históricamente se implementaban dentro del kernel se mueven **fuera de éste**, en un conjunto de "**servidores**" que se comunican a través de un kernel con el mínimo posible de código. Estos servidores, al contrario que en otros kernel, se ejecutan en el **espacio de usuario** en vez de en el **espacio de sistema**, manteniéndose en éste solo lo imprescindible, como el **IPC** (Inter Process Comunitacion), que permite la comunicación entre los diferentes procesos. [9]

  Esta arquitectura tiene varias **ventajas** sobre la monolítica, como que son **más fácil de mantener**, los **parches** se pueden **testear** en instancias separadas y son más rápidos de desarrollar. Pero también tiene algunos **inconvenientes**, como que los **programas necesitan más memoria** para poder ejecutarse, la **gestión de procesos** es más **compleja**, y el rendimiento es menor que en los kernel monolíticos. [9]

  Algunos ejemplos de sistemas operativos con microkernel son **AmigaOS**, **Minix** y **HarmonyOS**, entre otros.

- **Arquitectura Híbrida**

  Estos tipos de kernel son **similares** a los **microkernel**, con la diferencia de que **incluyen más funcionalidades** dentro de **núcleo** para mejorar el rendimiento de éste, combinando, por así decirlo, las arquitecturas monolítica y microkernel. La idea es tener una estructura similar a un microkernel, pero implementando esa estructura de forma monolítica, así, no tienen los beneficios de tener los servicios del SO en el espacio de usuario, pero tampoco tienen la sobrecarga en el paso de mensajes que esto puede ocasionar, siendo en este aspecto más similar a un kernel monolítico. [10]

  El ejemplo más conocido de este tipo de kernel son los sistemas **Microsoft Windows**, que desde la versión **Windows NT** se lleva empleando en todos sus sistemas operativos, incluido la ultima versión, **Windows 11**. También hay otros sistemas que emplean este tipo de arquitectura, como **BeOS**, **Netware** o **ReactOS**.

2.5. Actividad 5: Gestión de Procesos

Sabemos las siguientes características sobre un sistema operativo:

1. Utiliza el algoritmo de planificación de procesos SJF (el trabajo más corto primero), el cual es un algoritmo no apropiativo.
2. Necesita ejecutar una serie de procesos, cuyos instantes de llegada y tiempos que tardan en ejecutarse se representan en la siguiente tabla:

Proceso	Llegada	Tiempo de CPU
A	0	4
B	1	8
C	3	5
D	5	2
E	8	1

Figura 2.3: Tabla de procesos: tiempos de ejecución y llegada

3. Los procesos se ejecutan en un sistema operativo ideal, es decir, en el que el sistema operativo no consume recursos de CPU.
4. Comenzamos a estudiar el sistema desde que entran nuestros procesos al sistema y considerando la Unidad de Tiempo 0 (UT0),
5. En la siguiente tabla se pueden apreciar los procesos que se ejecutan en estas condiciones desde la Unidad de Tiempo 0 (UT0) a la Unidad de Tiempo 1 (UT1).

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A	↓1	2																		
B		↓																		
C																				
D																				
E																				

Figura 2.4: Tabla de gestión de los procesos

↓	Proceso ha llegado y está listo para ejecutarse en ese instante
F	Proceso termina en ese instante
#	Proceso se está ejecutando (# representa el instante de ejecución)
	Proceso está esperando en la cola de procesos listos

Figura 2.5: Leyenda de la tabla de gestión de procesos

Respetando todas las restricciones dadas en el enunciado:

1. Completa la tabla anterior para las unidades de tiempo de la 2 a la 19, estableciendo el proceso que se ejecutará en cada unidad de tiempo e indicando los instantes de llegada de cada proceso. Utiliza la nomenclatura y/o simbología que aparece en la leyenda.
2. Razona tu respuesta especificando el estado en el que se encuentra cada uno de los procesos para las unidades de tiempo de la 2 a la 8.

Nota: En el apartado "3.2.- Planificación apropiativa y no apropiativa"de la unidad tienes vídeos con ejemplos resueltos de los principales algoritmos de planificación estudiados.

2.5.1. Solución

Hay que tener en cuenta que se esta usando una algoritmo **SJF** (Short Job First), por lo que los procesos con menos tiempo de ejecución tendrán más prioridad que los que tengan más. Además, es un algoritmo **no apropiativo**, por lo que una vez que un proceso está ocupando la CPU el sistema operativo no podrá bloquearlo. Teniendo esto en cuenta, vamos a responder a los dos apartados que se nos piden.

1. En primer lugar, vamos a mostrar la tabla de ejecución de los procesos. La nomenclatura es la que se especifica en el enunciado por lo que no se va a repetir aquí.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	15	17	18	19
A	↓1	2	3	4F																
B		↓											1	2	3	4	5	6	7	8F
C				↓	1	2	3	4	5F											
D						↓					1	2F								
E									↓	1F										

Figura 2.6: Tabla de ejecución de procesos resuelta.

2. Los resultados obtenidos en la tabla se deben, como ya hemos comentado, a que se esta usando un algoritmo SJF, por lo que si en un momento dado hay varios procesos a la espera, siempre tendrá prioridad el que tenga menos ciclos de computación. Ademas, una vez que este proceso ocupa la CPU, no hay forma de bloquearlo, ya que el algoritmos es de tipo **no apropiativo**. Vamos a analizar con más detalle los momentos de ejecución entre el punto 2 y el 8, incluidos.

   - **Momento 2**: en este momento de la ejecución, tenemos al proceso **A ejecutándose**, ya que llego en la unidad de tiempo 1, cuando no había ningún proceso más y comenzó a ejecutarse. El proceso **B** está en **espera**, ya que llego en el momento 2 cuando ya se estaba ejecutando el proceso A y al ser un algoritmo no apropiativo, el proceso A no se puede bloquear.
   - **Momento 3**: en este punto sigue **ejecutándose** el proceso **A**, que se encuentra en su último ciclo de computación, y el proceso **B** sigue en **espera**, pero además, acaba de entrar el proceso **C**, que tendrá que ponerse a la espera.
   - **Momento 4**: ahora el proceso **A** ha finalizado y teníamos 2 procesos a la espera, el **B** y el **C**. Como estamos usando el algoritmo SJF, se ejecutará el proceso con menos ciclos de computación, por lo que el proceso **C**, con **5 ciclos de computación**, tendrá prioridad sobre **B**, que tiene **8 ciclos**. Así que el proceso **C** comenzará su **ejecución** mientras **B** sigue a la **espera**.
   - **Momento 5**: el proceso **C** sigue en **ejecución**, mientras que el proceso **B** sigue a la espera y el proceso que acabo de entrar, el **D**, entrará en **espera** también.
   - **Momento 6**: aquí no ha cambiado nada, seguimos teniendo el mismo estado que en el punto anterior. El único cambio sería que el proceso en ejecución, el **C**, va por el 3 ciclo de computación.
   - **Momento 7**: misma situación que en el punto anterior, ningún cambio salvo un un ciclo más de computación para el proceso **C**.
   - **Momento 8**: en este punto, el proceso **C** está en su último ciclo de computación. Tenemos a los procesos **B** y **D** en espera, además acaba de entrar el proceso **E**, que pasará a estar a la espera también. Como ya es el último ciclo de computación de **C**, habrá que seleccionar el proceso que se ejecutará a continuación. Tenemos 3 procesos a la espera, proceso **B** con 8 ciclos de computación, el proceso **D** con 2 ciclos de computación y el proceso que acaba de llegar, el **E** con 1 ciclo de computación. Como el proceso con menos tiempo de computación es el que tiene prioridad, en el **siguiente ciclo** se **ejecutará** el proceso **E**.

2.6. Actividad 6: Gestión de Memoria

Supón un sistema en el que la gestión de memoria se realiza siguiendo un esquema de asignación de particiones variables en el que no es posible realizar compactación de memoria (no se pueden mover de sitio los procesos una vez hayan sido ubicados en memoria). La capacidad de la memoria es de 4000 KB, de los cuales 500 se encuentran ocupados por el sistema operativo, y el resto está disponible para ubicar los procesos que se ejecuten. La situación de partida, la podemos ver en la Figura 2.7.

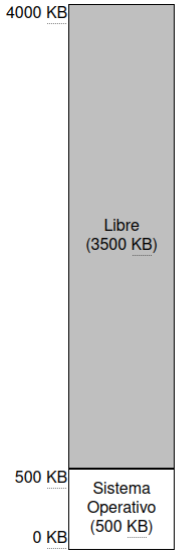

Figura 2.7: Imagen de la situación inicial de la memoria

Se van produciendo las siguientes llegadas y salidas de procesos, por orden, que requieren la asignación o liberación de trozos de memoria:

1. Inicialmente solo está cargado el sistema operativo, sin ningún otro proceso (situación de partida).
2. Llega un proceso A de 1200 KB de tamaño y se intenta cargar en memoria.
3. Llega un proceso B de 500 KB de tamaño y se intenta cargar en memoria.
4. Llega un proceso C de 1000 KB de tamaño y se intenta cargar en memoria.
5. Llega un proceso D de 400 KB de tamaño y se intenta cargar en memoria.
6. El proceso B termina su ejecución y se libera el espacio que estaba ocupando en memoria.
7. Llega un proceso E de 600 KB de tamaño y se intenta cargar en memoria.

Realiza lo siguiente:

(a) Si consideramos que el dibujo de arriba se corresponde con el punto 1 de la lista anterior, incluye dos dibujos más en los que se muestre cómo se encontraría la memoria después de los instantes 5 (tras llegar D y cargarse en memoria) y 6 (tras terminar B y liberarse su espacio).

(b) Explica qué ocurre en el punto 7, cuando llega E e intenta cargarse en memoria. Comenta si ocurre algún tipo de fragmentación y, en caso afirmativo, qué tipo de fragmentación sería.

2.6.1. Solución

En este punto vamos a ver como se gestionaría la memoria en un sistema operativo con esquema de asignación de particiones variables en el que no es posible la compactación, que como veremos a continuación, provocara la fragmentación de la memoria.

(a) En primer lugar, vamos a ver como quedaría la memoria en los instantes 5 y 6. El espacio en gris representa la memoria libre, y como podemos ver, tras el paso 6, quedan dos espacios de memoria libres de 400 y 500 KB.

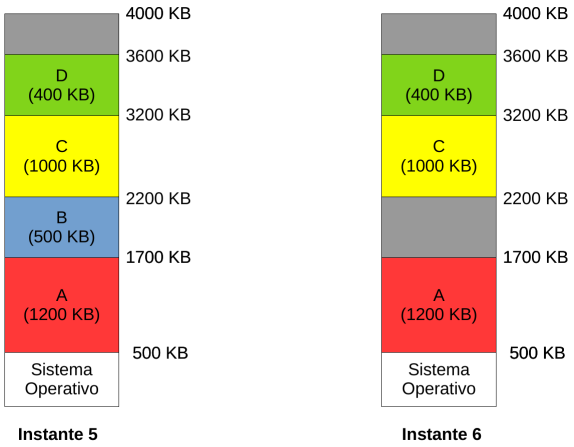

Figura 2.8: Estado de la memoria en los pasos 5 y 6

(b) Al entrar un proceso **E** de **600 KB**, no podría cargarse en memoria, ya que los espacios libres, como he comentado en el punto anterior, son de **400 KB** y **500 KB**. Esto provocaría un estado de **fragmentación externa** en la memoria, ya que la memoria externa a las particiones no es lo suficientemente grande como para alojar al nuevo proceso, por lo que en es punto, y hasta que algún proceso liberara memoria, no se podría usar.

2.7. Actividad 7: Estructura de Directorios y Rutas

Considera las siguientes estructuras de directorios para un equipo con SO Windows 10 (A) y uno con SO Lubuntu 22.04 (B):

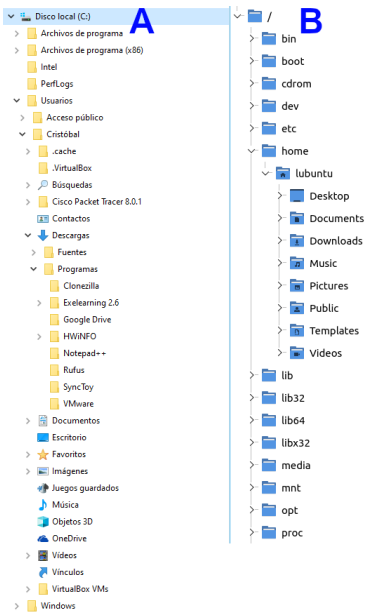

Figura 2.9: Estructuras de directorios para Windows y Lubuntu

Para el equipo con Windows (A), escribe las siguientes rutas:

- Una ruta absoluta al directorio "Fuentes".

- Una ruta relativa al directorio Rufus", considerando que el directorio de trabajo/activo actual sea Ïmágenes".
- **NOTA:** Las rutas en Windows no son sensibles a mayúsculas y minúsculas.

Para el equipo con Linux (B), escribe las siguientes rutas:

- Una ruta absoluta al directorio "Downloads".
- Una ruta relativa al directorio "Pictures", considerando que el directorio de trabajo/activo actual sea "media".
- **NOTA:** Las rutas en Linux sí son sensibles a mayúsculas y minúsculas.

2.7.1. Solución

(a). Rutas para **Windows**

- **Ruta Absoluta** a “Fuentes”:

  `C:\Usuarios\Cristóbal\Descargas\Fuentes\`

- **Ruta Relativa** a “Rufus” desde el directorio “Imágenes”:

  `..\Usuarios\Cristóbal\Descargas\Programas\Rufus\`

(b). Rutas para **Lubuntu**:

- **Ruta Absoluta** a “Downloads”:

  `/home/lubuntu/Downloads/`

- **Ruta Relativa** a “Pictures” desde el directorio “media”:

  `../home/lubuntu/Pictures`

Referencias

[1] Página oficial de Microsoft con los términos de uso.
https://www.microsoft.com/es-es/useterms.

[2] Wikipedia - Chrome.
https://www.google.com/intl/es/chrome/terms/.

[3] Términos de servicio adicionales de Google para Chrome.
https://www.google.com/intl/es/chrome/terms/.

[4] Wikipedia - Licencia BSD.
https://es.wikipedia.org/wiki/Licencia_BSD.

[5] Wikipedia - VCL media player.
https://es.wikipedia.org/wiki/VLC_media_player.

[6] EULA de avast.
https://static3.avast.com/10002736/web/o/legal/EULA_legacy/EULA_v1.14/EULA_v1.14_en-ww.pdf.

[7] Wikipedia - Adobe Acrobat.
https://es.wikipedia.org/wiki/Adobe_Acrobat.

[8] Wikipedia - Monolithic kernel.
https://en.wikipedia.org/wiki/Monolithic_kernel.

[9] Wikipedia - Kernel.
https://es.wikipedia.org/wiki/Adobe_Acrobat.

[10] Wikipedia - Hybrid kernel.
https://en.wikipedia.org/wiki/Hybrid_kernel.