Tema 1

Introducción a la física computacional

Bienvenidos a la asignatura de *Física Computacional I*. La asignatura de Física Computacional se ha añadido recientemente al temario de los estudios de Ciencias Físicas. En esta asignatura aprenderemos a usar diversas herramientas computacionales que nos serán útiles en el futuro, sea cual sea la actividad que realicemos como físicos.

El objetivo de esta asignatura es aprender a manejar herramientas computacionales básicas, tanto de cálculo simbólico como de cálculo numérico. En la primera mitad del curso veremos, como ejemplo representativo de programa de cálculo simbólico, una introducción al programa gratuito de código abierto MAXIMA[1] y en la segunda parte, como ejemplo de lenguaje de programación útil para cálculo numérico, veremos una introducción al lenguaje C[2]. En la elección de estos lenguajes informáticos nos hemos limitado a aquellos programas gratuitos de libre acceso, que al mismo tiempo sean suficientemente representativos como herramientas de cálculo simbólico, en un caso, y como lenguaje de programación en el otro. De todas formas, en el caso del lenguaje C éste se ha convertido en el lenguaje de programación estándar, de modo que más que un ejemplo representativo se trata del lenguaje de programación por excelencia. En el caso del MAXIMA existen alternativas comerciales más potentes (como p. ej. el MAPLE o el MATHEMATICA) cuyo uso está más extendido, sin embargo, para este curso hemos optado por el paquete de cálculo simbólico gratuito de código abierto. En este sentido el principal objetivo es aprender a organizar el trabajo de una manera ordenada y eficiente, lo que nos resultará útil en el futuro independientemente de cuál sea el programa de cálculo simbólico que empleemos.

1.1. Matemáticas en Física

Como todos sabemos, la Física es la ciencia que estudia las leyes que gobiernan el comportamiento de todo cuanto se conoce (materia, energía, espacio, tiempo, ...). Aunque realmente nadie sabe por qué es así, lo cierto es que, cuando se analizan cuidadosamente, los sistemas físicos parecen estar regidos por una serie de *leyes físicas*, dadas por unas ecuaciones matemáticas más o menos sencillas de formular y de entender, y cuya resolución es, con frecuencia, difícil. Aunque la apreciación sobre la *sencillez* de las leyes físicas es algo subjetiva, el cumplimiento de dichas leyes

[1] http://maxima.sourceforge.net/

[2] http://en.wikipedia.org/wiki/C_programming_language

no lo es. Con mucha frecuencia las leyes de la física toman la forma de *ecuaciones diferenciales*, o bien ordinarias para magnitudes dependientes sólo del tiempo, o bien en derivadas parciales para magnitudes dependientes del espacio y el tiempo, es decir, para magnitudes descritas por medio de *campos*. Por este motivo el trabajo del físico siempre está ligado a la manipulación de objetos matemáticos (vectores, matrices, funciones, ecuaciones, . . .) y a la realización de cálculos.

Aparte de proporcionar cierto grado de conocimiento sobre cómo funciona el universo en general, el conocimiento de las leyes que rigen el funcionamiento de los sistemas físicos nos permite realizar predicciones cuantitativas muy precisas sobre cuál será el estado de un sistema dentro de un tiempo a partir del conocimiento de su estado actual y de las interacciones a las que está sujeto. Para realizar estas predicciones normalmente es necesario realizar ciertos cálculos numéricos (normalmente un número muy elevado de ellos), en los que los ordenadores se han convertido en la herramienta fundamental desde finales del siglo XX.

El hecho constatado de que las leyes de la física tengan forma matemática hace que las matemáticas en física, y en general *en las Ciencias Naturales*, sean mucho más que una mera herramienta. Si el objetivo de la física es describir las leyes que rigen el funcionamiento de todo cuanto se conoce, el conocimiento del lenguaje en el que, aparentemente, están codificadas estas leyes es una parte fundamental de la física, motivo por el cual los temarios de la carrera de Ciencias Físicas son bastante generosos en asignaturas de Matemáticas en todas las universidades del mundo. La enorme relevancia de las matemáticas en las ciencias físicas se analiza con mayor profundidad en el famoso ensayo de Eugene P. Wigner (premio Nobel de física en 1963): “The Unreasonable Effectiveness of Mathematics in the Natural Sciences”, Communications in Pure and Applied Mathematics 13 (1960).

1.1.1. Echemos un vistazo rápido a las Matemáticas que estudiaremos en Físicas

El tipo de objetos matemáticos que maneja normalmente un físico es lo que determina el temario de matemáticas que se estudian en Físicas. En este sentido, y resumiendo mucho, se podría decir que las matemáticas que se estudian en Físicas están orientadas al objetivo de saber resolver ecuaciones diferenciales, tanto *ordinarias* como *en derivadas parciales*. El motivo es que, en general, las leyes físicas son ecuaciones diferenciales que expresan cómo cambian con el tiempo las magnitudes físicas que describen un sistema concreto. Esto no es tan sorprendente si tenemos en cuenta que, en general, cualquier ley física que exprese un principio de conservación (p. ej. de la energía o del momento lineal) para una magnitud descriptible por medio de un *campo* (una función del espacio y del tiempo) llevará a una relación que deben cumplir las derivadas de esa función respecto de sus variables, es decir, a una ecuación diferencial en derivadas parciales, o a una ecuación diferencial ordinaria si la función considerada sólo es función del tiempo.

Por ejemplo, las ecuaciones diferenciales ordinarias (EDOs) son fundamentales en la mecánica de Newton, donde para realizar predicciones debemos resolver la segunda ley de Newton, que es una ecuación diferencial ordinaria de segundo orden (ya que las derivadas de mayor orden que incluye son de orden 2)

$$\boldsymbol{F} = m\frac{d^2\boldsymbol{r}}{dt^2} \tag{1.1}$$

cuya solución queda determinada de manera única cuando se conocen las dos condiciones iniciales dadas para la velocidad y la posición inicial del móvil

$$\left.\frac{d\boldsymbol{r}}{dt}\right|_{t=0} = \boldsymbol{v}_0, \qquad \boldsymbol{r}(t=0) = \boldsymbol{r}_0 \tag{1.2}$$

Una situación similar se da en el caso de todos los modelos simplificados que se emplean con frecuencia en diversos campos para describir determinados procesos por medio de sistemas dinámicos, cuya evolución temporal está determinada por EDOs del tipo

$$\frac{dx_i}{dt} = f_i(x_1, x_2, \ldots, x_n, t), \qquad x_i(t=0) = x_{i,0}, \qquad i = 1, 2, \ldots, n \tag{1.3}$$

Las EDOs con "condiciones iniciales" (como los ejemplos anteriores) no son las únicas importantes, también son fundamentales las EDOs con "condiciones de contorno", que aparecen, por ejemplo, al resolver una ecuación diferencial en derivadas parciales por medio del método de separación de variables, que ya se estudiará en su momento.

Cuando se manejan magnitudes físicas que no son sólo funciones del tiempo, lo más habitual es que las leyes físicas sean ecuaciones diferenciales en derivadas parciales. Las ecuaciones diferenciales en derivadas parciales (EDPs) se han convertido en el lenguaje básico de la física desde el momento en que aparece el concepto de "campo" como herramienta matemática apropiada para la descripción de diversas variables físicas cuyo valor es función del espacio y el tiempo (como p. ej. la temperatura o la velocidad de un fluido, o la intensidad de un campo magnético en una región en la que hay cargas en movimiento), o para la descripción de interacciones que se transmiten a distancia, como p. ej. las interacciones electromagnéticas o la interacción gravitatoria (de hecho, el concepto de campo está ya implícito en la ley de gravitación universal de Newton).

Esta situación se cumple en todas las escalas de la física, desde la escala *cósmica* (descrita por las ecuaciones del campo gravitatorio de la Relatividad General de Einstein) hasta la escala *microscópica* (descrita por las ecuaciones de la Mecánica Cuántica: Schrödinger, Klein-Gordon, Dirac, ...), pasando por la escala *macroscópica* en la que nos movemos nosotros, es decir, la escala que incluye longitudes características que van desde (pongamos) los milímetros a (pongamos) los kilómetros, descrita por ecuaciones como las de Navier-Stokes para los fluidos. En todas estas escalas las ecuaciones diferenciales en derivadas parciales son también el lenguaje fundamental que permite describir las fuerzas electromagnéticas (por medio de las ecuaciones de Maxwell). Aparte de esto las EDPs son también la herramienta fundamental para modelizar fenómenos sencillos que dependen de más de una variable (ecuación de ondas, ecuación de Laplace, ecuación de la difusión, ...), y también son importantes en disciplinas nuevas de la física, como p. ej. el análisis de imágenes digitales (tan importante en teledetección y en medicina), donde frecuentemente se integra una ecuación de difusión para suavizar una imagen, o se plantea una EDP (la ecuación del flujo óptico) para determinar el movimiento de los patrones detectados en una secuencia de imágenes. En el transcurso de la carrera de Ciencias Físicas uno irá descubriendo poco a poco muchas de estas leyes físicas (algunas de ellas pertenecen a temas más o menos avanzados, que sólo se estudian en programas de doctorado), su formulación matemática precisa, las ecuaciones que las describen y cómo se resuelven.

El otro factor que determina las matemáticas que debe conocer un físico es la naturaleza de los objetos matemáticos a los que se aplican las leyes de la física. Tal y como

hemos dicho antes, las variables físicas normalmente son funciones, a veces sólo del tiempo (como p. ej. el valor promedio de la temperatura en una habitación), aunque con mucha frecuencia serán funciones tanto del espacio como del tiempo (como p. ej. el campo de temperaturas en una habitación), por este motivo el físico debe conocer todo lo relativo al álgebra y al cálculo con funciones de varias variables (derivadas parciales, integrales múltiples, etc.). Aunque con cierta frecuencia las funciones que se consideran en física son funciones reales de variable real, esto no siempre es así, por ejemplo en Mecánica Cuántica la función de ondas que describe los orbitales atómicos es una función que depende de variables reales pero que toma valores complejos, por otra parte también hay algunos casos en los que resulta conveniente describir ciertas magnitudes físicas por medio de funciones de variable compleja. Además, el análisis de funciones de variable real es incompleto si no se estudia un poco de variable compleja, de la misma manera que el estudio de los polinomios reales de variable real es incompleto si no se estudia algo de variable compleja, ya que a veces las raíces de estos polinomios son complejas. En particular, hay varios resultados muy relevantes que se aplican a funciones reales de variable real a los que sólo se puede llegar por medio del análisis complejo (p. ej. el cálculo de integrales impropias o las transformadas de Fourier), de modo que algo de variable compleja es imprescindible y por este motivo las titulaciones en física generalmente incluyen esta rama de la matemática en sus temarios.

Otra disciplina matemática útil para el físico es la geometría diferencial y el cálculo tensorial. Generalmente todas las magnitudes físicas suelen tener un *rango tensorial* bien definido. Por ejemplo, hay magnitudes físicas que se describen por medio de funciones escalares (que son *tensores de orden 0*), como la temperatura o la masa, otras están descritas por medio de funciones vectoriales (*tensores de orden 1*), como la velocidad o las fuerzas, y hay ciertas magnitudes que se describen por medio de objetos tensoriales un poco más complicados, como p. ej. el estado de tensiones en el interior de un material, descrito por un *tensor de orden 2* (que ya se estudiará en su momento). Por otra parte en el trabajo rutinario que realiza un físico aparecen con muchísima frecuencia curvas y superficies, por ejemplo en la descripción de las trayectorias de un móvil en el espacio, o de las líneas de corriente en un fluido en movimiento, o en el estudio de las superficies equipotenciales de un campo eléctrico, o de las fuerzas de tensión superficial que aparecen en las superficies de los líquidos (cuya intensidad depende de la curvatura de la superficie). En la manipulación de todos estos objetos geométricos entran ingredientes de cálculo tensorial y de geometría diferencial, que son por tanto disciplinas matemáticas relevantes para la física.

Continuando en la línea de analizar la naturaleza de los objetos matemáticos con los que describimos la realidad vemos que, en muchos casos, la descripción que se hace de ciertos sistemas físicos es estadística (esto es lo que sucede, p. ej., en la física cuántica o en la de fluidos turbulentos), de modo que la estadística es otra materia indispensable para un físico.

Para terminar este breve vistazo global debemos mencionar los espacios funcionales, es decir, espacios cuyos elementos son funciones (los denominados espacios de Hilbert). En general las magnitudes físicas se describen por medio de funciones matemáticas, muchas de las propiedades (relevantes en física) de estas funciones sólo pueden estudiarse en el contexto de los espacios funcionales. En particular los espacios de Hilbert constituyen el marco matemático básico de la mecánica cuántica, pero además constituyen el contexto matemático en el que las ecuaciones diferencia-

les (ordinarias y en derivadas parciales) son inteligibles, y en este contexto es donde se formulan y analizan los métodos que se emplean para resolverlas.

El estudio de las diversas asignaturas de matemáticas que se contemplan en físicas puede resultar en ocasiones un poco árido, sobre todo si no tenemos claro cuál es la finalidad de lo que se está estudiando. Esperamos que este brevísimo esquema sobre el infinitamente apasionante mundo de las matemáticas sea útil en este sentido, y que las herramientas de cálculo simbólico y numérico que veremos en la primera parte de esta asignatura sirvan de ayuda para su estudio en el futuro. Por otra parte es muy importante tener presente que aunque existen paquetes informáticos capaces de realizar cálculos simbólicos y numéricos, si no tenemos una comprensión profunda de las matemáticas que estamos empleando no seremos capaces de sacar partido a ninguna herramienta computacional, por potente que sea.

A todo esto, cuando en este apartado nos hemos referido a las leyes que rigen el comportamiento de un cierto sistema físico hemos dado por sentado que esas leyes son conocidas. ¿Qué sucede en caso contrario? La experimentación sistemática (racional y objetiva) ha sido desde siempre la herramienta fundamental en física (en general en el método científico) para deducir estas leyes, o al menos para formular modelos que nos permitan encontrar una aproximación válida de las mismas. En este sentido es donde la gran potencia alcanzada por los ordenadores desde finales de siglo XX nos ofrece la posibilidad de sustituir algunos experimentos reales por *experimentos virtuales*, en los que el *experimentador* introduce unas reglas de evolución sencillas y posteriormente observa el comportamiento de ese sistema (calculado por medio de un ordenador en una *simulación numérica*), y analiza si las reglas de evolución propuestas reproducen el comportamiento observado en la realidad. Esto es lo que se hace en Física Computacional y aprender a usar ordenadores para ello es el objetivo de la segunda parte de esta asignatura.

1.1.2. Qué hacen y qué no hacen los ordenadores

Esto puede parecer una trivialidad pero conviene tener presente que los ordenadores hacen sólo tareas sistemáticas, repetitivas, tediosas, y las hacen muy rápido; por eso son tan útiles. En muchos problemas habituales en física nos encontramos con que necesitamos realizar una cantidad apreciable de cálculos numéricos (o de manipulaciones simbólicas), que aparte del volumen de trabajo que representan no ofrecen ninguna dificultad, ese es el tipo de trabajo que debemos re-dirigir al ordenador. De todas formas, desde el punto de vista del investigador en física, el objetivo último de estos cálculos no es solamente encontrar el resultado, la parte físicamente más importante viene después con la **interpretación física** del resultado obtenido y la extracción de **conclusiones** que nos permitirá hacer predicciones *cualitativas* correctas para casos similares sin necesidad de hacer ningún cálculo. El objetivo en física no es generar una solución numérica sino comprenderla, es decir, alcanzar cierto grado de *conocimiento* sobre el comportamiento del sistema que se está estudiando.

Por eso no podemos esperar que los ordenadores nos resuelvan todos los problemas. De hecho, normalmente la parte más difícil de un cálculo numérico es plantear el problema de la manera adecuada, una vez que logramos eso, el resto es fácil. Para ser capaces de plantear un problema de la manera correcta es necesario tener cierto grado de conocimiento sobre lo que se está haciendo, sobre lo que significan las ecuaciones que estamos intentando resolver y también sobre qué es exactamente lo que

hace el método numérico que estamos empleando. Sólo teniendo los conocimientos adecuados de física y matemáticas podremos entender por qué funciona un cálculo numérico (o por qué no funciona cuando el método falle) y, de esta manera, hacer un uso realmente eficiente de un ordenador. Es un error (que, desgraciadamente, empieza a ser habitual) pensar que los ordenadores nos evitarán la tarea de tener que dominar algo de matemáticas para ser capaces de trabajar en física.

En esta asignatura vamos a centrarnos en los aspectos técnicos del uso de algunos paquetes informáticos. Es decir, nos centraremos en el *¿cómo se hace?*, más que en el *¿por qué se hace?*. Responder a la segunda pregunta (¿por qué resolvemos este problema con este método?, ¿por qué lo planteamos de esta forma?) es algo que irá quedando claro, poco a poco, a medida que se avance en los estudios, a lo largo de toda la carrera. De esta forma, con esta asignatura pretendemos ofrecer un complemento a las asignaturas tradicionales de matemáticas. Esperamos sinceramente que disponer de los conocimientos técnicos sobre ordenadores impartidos en esta asignatura será de gran utilidad en los cursos posteriores, en los que necesitará poner en práctica todas estas herramientas.

1.2. Uso eficiente de ordenadores para trabajo científico. Sistema operativo Linux

Es muy probable que muchos alumnos de esta asignatura sean aficionados (probablemente muy expertos) a los ordenadores y la programación, para ellos esta asignatura será, sin duda alguna, muy sencilla. Este apartado está dirigido principalmente a los que todavía no lo son. Para aquellos alumnos que no estén demasiado familiarizados con la programación vamos a incluir a continuación algunas indicaciones generales.

Como es sabido, los ordenadores son máquinas capaces de almacenar y manipular información, almacenada en su interior en código binario. Para la manipulación de la información los ordenadores disponen de un procesador, que puede ser programado para realizar distintas operaciones. En última instancia, el procesador de un ordenador maneja información en código binario. Esto significa que, a nivel del procesador, la información está codificada en una inmensa secuencia de *ceros* y *unos*, asignados a unas variables elementales, dadas por las direcciones de memoria sobre las que opera el procesador, de modo que la actividad del procesador se reduce a mover estos *ceros* y *unos* de unas direcciones de memoria a otras, y a intercambiar sus valores siguiendo unas pocas reglas simples y definidas. Resulta obvio que este nivel *fundamental* de funcionamiento (denominado *bajo nivel*) queda muy lejos del nivel que a nosotros nos resulta inteligible, basado no en *ceros* y *unos*, sino en archivos de texto, de audio, de vídeo, programas ejecutables, de gráficos, comunicaciones, etc. (este nivel se denomina *alto nivel*).

El programa que permite al usuario comunicarse con el procesador es el Sistema Operativo (generalmente programado en lenguaje C). El funcionamiento de un ordenador, el tipo de cosas que podremos hacer con él y la forma de hacerlas, está muy condicionado por el sistema operativo que tenga instalado, y por lo que éste nos permita hacer.

1.2.1. Qué podemos esperar y qué no debemos permitir de un ordenador

De un ordenador debemos esperar que realice de manera eficiente todas las tareas que le encarguemos, y nada más. No deberíamos permitir que un ordenador sea una amenaza para la integridad e inviolabilidad, de la información que depositamos en él. El ordenador debe trabajar para nosotros, y no al contrario, por este motivo no deberíamos permitir que el mantenimiento del ordenador suponga una carga de trabajo considerable.

Para el tipo de tareas que nos interesan en esta asignatura, la elección de Sistema Operativo más eficiente, segura y económica es el sistema operativo (de tipo Unix) denominado Linux, sin ninguna duda. En sus inicios (hacia principios de los 1990) el SO Linux tenía el inconveniente de ser algo más complicado de usar que otros conocidos sistemas operativos comerciales. Sin embargo, en la actualidad esto ya no es una limitación.

La superioridad de los sistemas operativos tipo Unix está fuera de discusión en la actualidad. Todos los super-ordenadores (vectoriales, clusters, etc.) dedicados a tareas de cálculo exhaustivas en centros de investigación (o en universidades, etc.) funcionan con sistemas o bien Unix o bien Linux, concretamente el 95 % de los 500 super-ordenadores más rápidos del mundo usan Linux o alguna de sus variantes[3]. Los sistemas operativos tipo Unix, como el Linux son también los escogidos en la mayoría de los casos para los servidores que proporcionan conexión a Internet a un número elevado de usuarios.

Las ventajas del Linux frente a otros sistemas operativos son enormes. Por un lado el propio SO es gratuito, así como todas las actualizaciones. Además es completo, es decir, incluye paquetes para cualquier tarea que queramos hacer con el ordenador, como p. ej. comprimir y descomprimir archivos, escribir documentos de todo tipo, manipular archivos de audio y vídeo, incluye compiladores para muchos lenguajes de programación, entre ellos C, etc., y todos estos paquetes son gratuitos. Otra de las grandes ventajas es la seguridad. El SO Linux se basa en una jerarquía de permisos y de usuarios. Para usar el ordenador uno necesita estar registrado como usuario (con un *login* y una contraseña o *password*, y con un determinado directorio de trabajo, que es el único sitio en el que tiene *permiso de escritura*), de modo que un posible atacante ya se encuentra con dificultades. Para modificar algo que afecte a la configuración del aparato uno necesita conectarse como administrador (root). Además de esto el ordenador mantiene un registro de todas las conexiones realizadas (con éxito o fallidas), de modo que es fácil detectar si se está siendo víctima de un intento de invasión. Pero aparte de todo esto la mayor ventaja de Linux es la eficiencia con la que emplea los recursos de hardware disponibles (procesador o procesadores y memoria), adaptándose a —y sacando el mejor partido de— lo que tenga disponible. Para una introducción algo más extensa a este SO se puede consultar `http://en.wikipedia.org/wiki/Linux`.

1.2.2. Instalación del SO Linux

El sistema operativo Linux nace en 1991, escrito por Linus Torvalds y publicado en un servidor de una incipiente Internet. Como sistema operativo, Linux (o Linux 0.1) era muy básico: un núcleo (también llamado *kernel* o máquina virtual Linux) y una

[3] `http://www.top500.org/statistics/list/`

serie de herramientas Unix del proyecto GNU. En la actualidad, ambas cosas permanecen, aunque muy mejoradas por infinidad de programadores a lo largo de los años y adaptadas a la "moda" del momento: entornos gráficos y herramientas fácilmente utilizables desde su instalación. Esta adaptación ha llevado a la existencia de múltiples distribuciones de Linux, muchas gratuitas (como p. ej. Fedora, Ubuntu o Debian) y otras comerciales (como RedHat, CentOS o Suse). En este apartado nos centraremos en dos distribuciones gratuitas (o libres) muy extendidas: Fedora (heredera de RedHat) y Ubuntu (basada en Debian).

Para instalar un sistema operativo Linux, lo más sencillo es descargarse de Internet una imagen del CD de instalación. Para Fedora, esta imagen se puede descargar de

`https://getfedora.org/`

Para Ubuntu, se puede descargar de

`http://www.ubuntu.com/download`

En el momento de escribir estas notas, la versión última de Fedora es la 23, y la de Ubuntu la 15.10. Esto no significa nada más que, desde que se iniciaron estos dos proyectos, ha habido 23 y 15 cambios relativamente importantes en los sistemas de instalación y en las versiones del *kernel* linux que contienen; después del punto, se suelen indicar cambios menores a los que se puede actualizar una distribución anterior, sin necesidad de reinstalarlo todo (aunque, como veremos a continuación, esto no es tan difícil).

A continuación se dan los cinco pasos para instalar Linux, con pequeñas notas sobre las diferencias entre Fedora y Ubuntu (cuando las haya).

1. Lo primero antes de instalar cualquier sistema operativo en una máquina, es saber qué tipo de máquina tenemos; esto lo determina el procesador o, mejor dicho, la familia de procesadores a la que pertenece el nuestro. En la siguiente tabla se sintetiza la elección para PCs:

Fabricante/Modelo	Familia
Intel (excepto los indicados en las siguiente fila) AMD (excepto los indicados en la siguiente fila) VIA (C3, C7)	i386 (Intel 386)
Intel (Atom 230, 330; Core 2 Duo; Centrino Core 2 Duo; Xeon; Core i5; Core i7) AMD (Athlon 64 y x2; Sempron64; Opteron) Apple (MacBook, MacBook Pro, y MacBook Air)	x86_64 (Intel x86 de 64 bits)
Intel (Itanium, Itanium2)	IA64 (Intel architecture 64 bits)
Apple (Macintosh G3, G4, G5; PowerBook)	ppc

   Hasta fechas no muy lejanas los modelo más habituales eran los i386, en la actualidad son los x86_64. Además, si se instala un sistema operativo para i386 en una máquina con procesador x86_64, aunque no se sacará todo el rendimiento de la misma, el sistema operativo funcionará. No es así para las otras familias de procesadores.

2. Una vez creado el CD de instalación, y reiniciada la máquina para que arranque desde el CD/DVD, se carga el sistema operativo Linux. Lo primero que hará éste

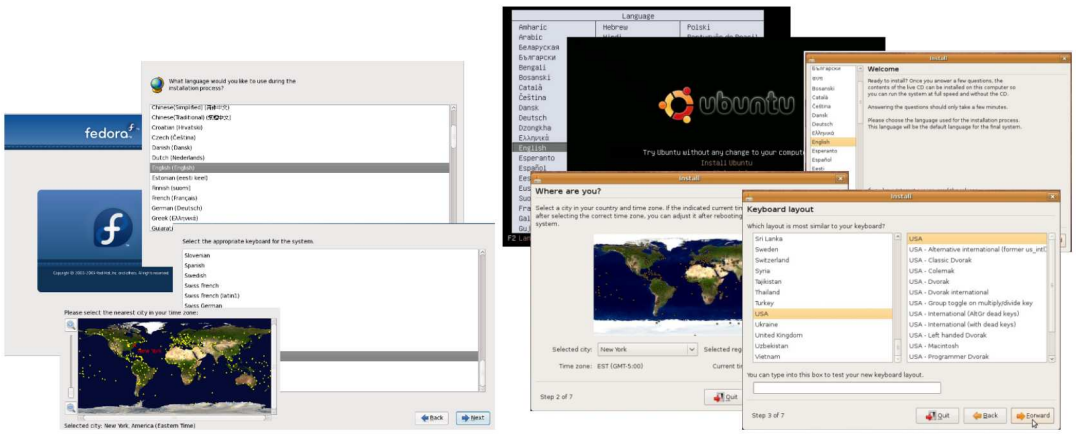

Figura 1.1: Inicio de la instalación de Linux: configurar idioma, teclado, fecha y hora.

será ejecutar un programa que comprueba el tipo de máquina en la que está: tipo de placa base, el tipo y tamaño de disco duro, si hay un teclado y un ratón, si hay una tarjeta gráfica y un monitor, si hay tarjeta de red, etc. Lo primero que veremos, destinado a nosotros será un mensaje de bienvenida al instalador de la distribución, junto con una serie de pasos guiados iniciales de configuración de la interfaz de usuario. Por ejemplo, habrá que seleccionar el idioma de la instalación: por defecto será Inglés, pero se puede seleccionar Español (de España) de entre la larga lista de idiomas soportados. También habrá que elegir el teclado que se tiene: la variante de idioma (Español), la distribución y número de teclas del teclado (usualmente QUERTY, 115 teclas, el habitual en España), la zona horaria en la que se encuentra[4]. De esta manera, todo lo demás será más fácil.

3. Una vez estudiada la máquina y configurada la interfaz de usuario, lo primero que se debe hacer es buscar sitio en el disco duro para instalar Linux. Aquí hay dos posibilidades: utilizar todo el disco duro para Linux o compartir el disco duro con otro sistema operativo ya instalado (como Microsoft Windows). [5]Dentro de cada disco habrá que crear una o más particiones, que son "regiones" en el disco en las que se encuentran los datos de un sistema operativo. Si ya existe una partición que queremos mantener, podemos cambiar su tamaño y hacer sitio para Linux en nuestro disco duro; alternativamente (o a continuación), se puede seleccionar una instalación por defecto que crea las particiones adecuadas para instalar Linux.

4. Después de hacer espacio en disco, el instalador copia en el disco duro los ar-

[4]En Fedora, por ejemplo, la zona horaria se indicará cuando se indique la ubicación en la red de la máquina, dos pasos más adelante

[5]Los discos en Linux tienen nombres (en función de su tipo) y letras (en función de su orden en la placa madre del ordenador). Así, por ejemplo, el primer disco IDE es el /dev/hda, mientras que el segundo disco SATA es el /dev/sdb. El "prefijo" /dev/ sólo indica que son dispositivos del sistema y da la ruta a ellos, porque en Unix todo, hasta los propios discos, es tratado como un fichero. Dentro de un disco existen particiones: zonas del disco (físicas o lógicas) en las que se encuentran datos relacionados entre sí. Por ejemplo, el "disco C:" de un Windows, es realmente la partición 1 del disco donde está instalado Windows; en Linux esta partición aparecerá con un nombre como "/dev/sda1" (primer disco, "a", partición 1), e instalaremos Linux a continuación, en la partición "/dev/sda2" (mismo disco, siguiente partición).

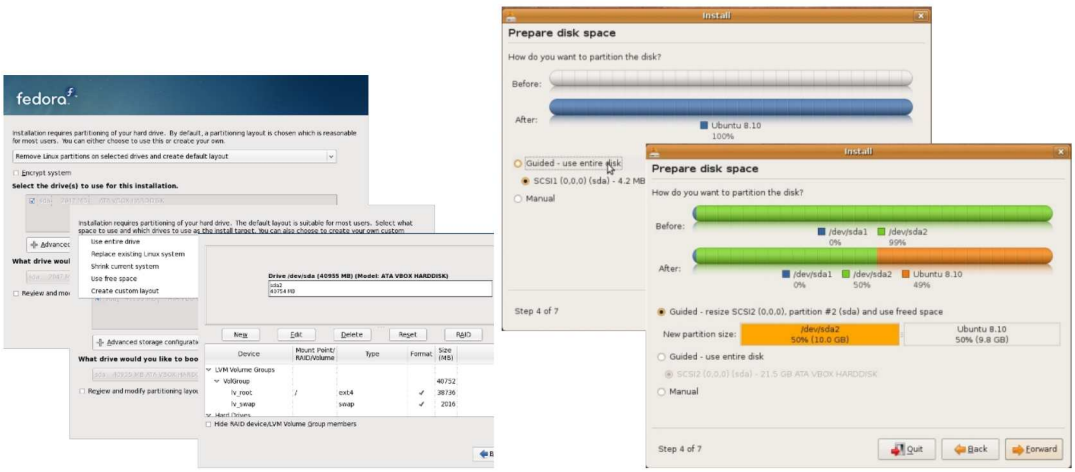

Figura 1.2: Particionado del disco para instalar Linux.

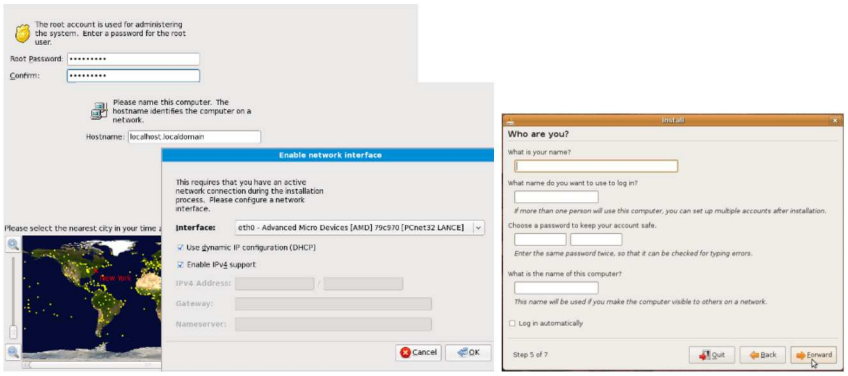

Figura 1.3: Identificación del usuario y de la máquina en la red.

chivos mínimos para tener en él un sistema operativo. Este sistema operativo es muy primitivo: no tiene interfaz gráfica ni prácticamente aplicaciones, pero tiene lo bastante para completar el proceso de instalación. Habitualmente, en este punto, se pedirán el nombre y la contraseña del administrador del sistema ("root") y de un usuario habitual de la máquina[6]. También se pedirá un nombre para la máquina y se buscará configurar la conexión a internet y "poner" la maquina en red[7]. Si alguna operación no se hace en este momento, siempre se podrá hacer una vez instalado el sistema operativo.

5. A continuación, se instalarán las aplicaciones. En Ubuntu, se hace una instalación de aplicaciones básicas (aunque muy completa, con todo lo que uno necesita para empezar a trabajar). En Fedora, se pregunta el uso que se dará a la máquina: de escritorio, servidor, etc. y se instalarán las aplicaciones y servicios adecuados a esa configuración. Ambos métodos de instalación son equivalentes

[6] Ubuntu, con el pretexto de facilitar la administración del sistema a los usuarios, no crea un usuario "root", sino que proporciona ciertos privilegios de administración al usuario habitual que se indica en este paso.

[7] Para esto es necesario tener conectada la máquina a una red, router o módem que Linux buscará e identificará, y los parámetros de conexión: DHCP o dirección IP, dirección de la puerta de enlace (*gateway*) y de los servidores DNS, etc.

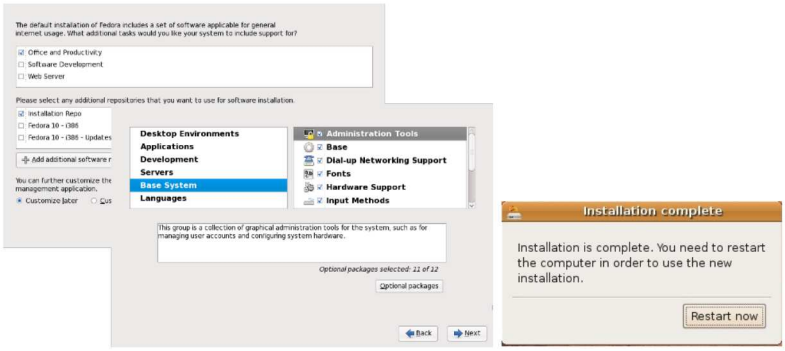

Figura 1.4: Selección del tipo de instalación, de las aplicaciones, y final de la instalación

desde el momento en que luego se podrán añadir o eliminar aplicaciones con los gestores de aplicaciones (que veremos más adelante). Cuando Linux termina de instalar las aplicaciones, avisa que la máquina se reiniciará y se arrancará Linux por primera vez.

1.2.3. Estructura y funcionamiento básico del SO Linux

Lo que se ha instalado es un sistema operativo y aplicaciones para este sistema operativo. Entre estas aplicaciones se encuentra el "servidor X", que es la aplicación que proporciona a las demás aplicaciones gráficas el entorno de ventanas, y el "entorno de escritorio", que es la aplicación que proporciona una visión integrada de todas las aplicaciones, para que el usuario las vea parecidas unas a otras y le sea intuitivo su uso, con el resto del sistema operativo (sistema de ficheros, administración de hardware, etc.). Sea cual sea la distribución de Linux que usemos, existen muchas opciones distintas para el "entorno de escritorio", las dos más habituales son el entorno "Gnome" (`http://www.gnome.org`) y el entorno "KDE" (`http://www.kde.org`). Existen otros muchos, como p. ej. el Xfce (`http://www.xfce.org`), especialmente orientado a ordenadores con procesadores lentos y poca memoria. En nuestra instalación de Linux podemos seleccionar que se instale más de un entorno y posteriormente probarlos todos para decidir cuál es el que más nos gusta (al abrir una sesión el ordenador nos permitirá escoger el entorno gráfico que queremos usar). Además, si tenemos instalados más de un entorno de escritorio desde cualquiera de ellos podremos usar las aplicaciones asociadas a los otros. Si tenemos instalado el Gnome y el KDE, desde una sesión con Gnome podremos usar, por poner un ejemplo, cualquiera de los editores de textos típicos del KDE (el Kwrite, o el Kate), y viceversa, desde una sesión en KDE podremos usar cualquiera de las aplicaciones incluidas con el Gnome (como el editor de textos Gedit o el Geany).

Linux basa su filosofía en el reparto de tareas. Un archivo de configuración, por ejemplo, contiene sólo instrucciones de configuración para una aplicación. El archivo ejecutable de esta aplicación sólo contiene (o debe contener) las instrucciones para que la aplicación funcione, pero no instrucciones que sean comunes con otras aplicaciones o datos de ésta. Las instrucciones comunes entre aplicaciones forman bibliotecas de funciones. Los datos, forman los archivos compartidos de datos. Las aplicaciones que realizan tareas especializadas que son útiles a las demás aplicacio-

nes, se instalan como servicios (el caso del servidor X o el caso de un servidor web). Esta filosofía explica dónde se guardan los ficheros destinados a cada tarea, como se muestra en la siguiente tabla.

Tarea	Directorio	Ejemplo
Configuración del sistema (y de las aplicaciones)	`/etc`	`/etc/hostname` `/etc/gtk/gtkrc.iso-8859-15`
Aplicaciones básicas del SO	`/bin`	`/bin/ls` `/bin/kill`
Bibliotecas de funciones del SO	`/lib`	`/lib/libc.so.6` `/lib/libx86.so.1`
Aplicaciones para los usuarios	`/usr/bin`	`/usr/bin/gedit` `/usr/bin/wxMaxima`
Bibliotecas de funciones de las aplicaciones para los usuarios	`/usr/lib` `/usr/lib64`	`/usr/lib/libgnome-2.so` `/usr/lib/mozilla/plugins/mozplugger.so`
Datos compartidos de las aplicaciones	`/usr/share`	`/usr/share/maxima/5.13.0/xmaxima/maxima-icon.png`
Datos propios de los usuarios	`/home`	`/home/usuario`

El reparto de tareas, hace que cada tarea la realice sólo un actor (aplicación, servicio, biblioteca, archivo de datos) y que las demás se aprovechen de lo bien que lo hace. Este diseño modular permite mejorar partes sin que haya que cambiar el todo y es la clave del éxito de los sistemas Unix. El sistema operativo se encarga de facilitar la interconexión de aplicaciones, servicios, bibliotecas, y su acceso a archivos de datos fácilmente accesibles al usuario (o al administrador)[8].

1.3. Mantenimiento e instalación de paquetes informáticos adicionales

Las aplicaciones que se han instalado inicialmente pueden eliminarse, ampliarse o actualizarse, posteriormente. Como se acaba de explicar, en Unix, y en Linux particularmente, la clave del funcionamiento es la colaboración entre aplicaciones, bibliotecas y datos. Por eso, cuando se instala una nueva aplicación, es necesario asegurarse de que están instalados aquéllos otros elementos de los que depende. De esto se encargan los sistemas de gestión de paquetes. Los dos más utilizados en Linux son “dnf” en distribuciones Fedora y afines y “apt” en Debian, Ubuntu y distribuciones afines[9]. Estas herramientas permiten mantener actualizado el sistema para protegerlo de posibles fallos de seguridad que se van descubriendo (algo que siempre está sucediendo:

[8] Desgraciadamente, algunas aplicaciones están perdiendo esta filosofía y se están convirtiendo en “mastodónticas”, lentas, y difíciles de mantener. Esto se debe, sobre todo, a que se busca que estas aplicaciones sean “portables” a otros sistemas operativos como Microsoft Windows que no proporcionan las ayudas a las aplicaciones que da Unix, por lo que es inevitable introducir código redundante que haga tareas que el sistema operativo ya hace, pero no comparte con las aplicaciones que se ejecutan.

[9] En distribuciones Fedora anteriores a la 22 en lugar de “dnf” se empleaba “yum” y en distribuciones Debian anteriores a la 8 se empleaba “apt-get”.

sólo es totalmente seguro el ordenador sin conexión a Internet, detrás de una puerta bien cerrada y... preferiblemente, apagado) o tener las últimas versiones de las aplicaciones libres. Para ello se emplean "repositorios" o almacenes de paquetes.

1.3.1. Mantenimiento automático de la máquina

Un paquete es un archivo que contiene en su interior, además de los ficheros que es necesario copiar en el disco, información sobre qué otros paquetes es necesario tener instalados para su correcto funcionamiento. Estos archivos suelen tener extensiones ".deb" (para Debian, Ubuntu y similares) o ".rpm" (para Fedora y otras distribuciones herederas de Red Hat).

Muchos paquetes vienen incluidos en el CD de instalación, pero la totalidad de ellos no cabrían ni en varios DVDs y, además, pronto quedarían anticuados. Para ello existen los "repositorios" (transliteración del inglés "repository", que significa almacén). Los repositorios son sitios de Internet (servidores HTTP o FTP) que contienen, muy actualizados, los paquetes disponibles para una versión de una distribución. Como consultar los paquetes de un repositorio es costoso, lo que se hace es mantener en el propio ordenador una lista de los paquetes que contienen y sus informaciones de dependencias. Este listado se actualiza con la instrucción (ejecutada como root desde la línea de comandos de un terminal)

`dnf update` (Fedora)

`apt update` (Debian/Ubuntu)

Conviene realizar esta operación antes de instalar nuevos paquetes, para garantizar que al descargar éstos se descargarán las últimas versiones de los paquetes necesarios para cumplir las dependencias (una modificación en un paquete de una aplicación, que le añada nuevas propiedades, puede requerir una nueva biblioteca de funciones que no esté todavía instalada en nuestro sistema).

1.3.2. Instalación de paquetes informáticos adicionales

Para instalar un paquete, lo primero que necesitamos es conocer su nombre. Los nombres de paquetes contienen toda la información para garantizar que son los adecuados a nuestra máquina: el nombre de la aplicación o biblioteca y su versión (además de la arquitectura de la máquina, de 32 o 64 bits, para la que está compilado). Cuando queremos buscar una aplicación nueva y no sabemos su nombre exacto o sólo sabemos lo que hace, debemos buscar el nombre correcto del paquete. Para ello ejecutaremos:

`dnf search` *paquete* (Fedora)

`apt search` *paquete* (Debian/Ubuntu)

Por ejemplo, si buscamos un sistema de álgebra por ordenador, podemos buscar

`apt search algebra` (Ubuntu)

El resultado será una inmensa lista con todos los paquetes que sirven para hacer alguna manipulación algebraica, seguidos de una breve descripción de los que son. Así encontraremos, entre muchos

- axiom - *A general purpose computer algebra system: main binary and modules*
- gap - *Groups, Algorithms and Programming computer algebra system*

- `jacal` - *Interactive symbolic math system*
- `maxima` - *A computer algebra system -- base system*
- `wxmaxima` - *a wxWidgets GUI for the computer algebra system maxima*
- `yacas` - *Computer Algebra System*

Para instalar una aplicación, emplearemos también los "dnf" y "apt". Así, por ejemplo, si queremos instalar un programa para hacer gráficas ("plot" en Inglés), lo buscaremos primero con

`dnf search plot` (Fedora)
`apt search plot` (Debian/Ubuntu)

lo que nos encontrará (entre muchos otros), el paquete "gnuplot":

`gnuplot` - *A command-line driven interactive plotting program*

Para instalarlo ordenaremos

`dnf install gnuplot` (Fedora)
`apt install gnuplot` (Debian/Ubuntu)

En ese momento se nos informará de todas las dependencias que tiene este paquete con otros, que se instalarán además del gnuplot; también se recomendarán otros paquetes (como el gnuplot-doc, que contiene la ayuda de la aplicación), etc. Hecho esto, se tendrá la aplicación instalada. Además la búsqueda nos habrá sugerido paquetes ayudantes, como

`plotdrop` - *A minimal GNOME frontend to GNUPlot*

que podremos instalar también para facilitarnos el uso de Gnuplot. Cuando seamos usuarios avanzados, sin embargo, eliminaremos esta aplicacioncita con

`dnf remove plotdrop` (Fedora)
`apt remove plotdrop` (Debian/Ubuntu)

En esa acción, el gestor de paquetes se encargará de no eliminar nada que pueda afectar a otras aplicaciones (y en caso de que alguna aplicación utilizara "plotdrop" como auxiliar para crear gráficas, nos prohibiría desinstalarlo... sin antes haber desinstalado la otra aplicación dependiente).

1.4. Procesadores de texto: LaTeX y LyX

El uso de ordenadores en física no se limita a la realización de cálculos, sino que se extiende a todo lo que atañe a la elaboración, manipulación, almacenamiento y distribución de información. Aunque el contenido de esta asignatura se centra en el uso de ordenadores para cálculo, en este apartado pasaremos revista rápidamente al tema de la generación de documentos, centrándonos en los tipos de documentos científicos más habituales: libros y artículos (tanto en formato impreso como electrónico) y presentaciones y pósters (empleados con frecuencia para exponer trabajos en congresos de investigación).

Tomando como ejemplo representativo de documento científico un libro de texto de Física General de primero vemos que el documento tiene cierta complejidad dado que incluye, además de texto, multitud de ecuaciones (muchas de ellas con caracteres no estándar, como letras griegas), figuras, tablas, referencias bibliográficas, etc. Para generar documentos científicos necesitaremos, por tanto, un programa (denominado *procesador de texto*) que, aparte de formatear textos, sea capaz de generar

ecuaciones y tablas, incluir gráficas, manejar volúmenes relativamente grandes de referencias bibliográficas y también manejar referencias cruzadas internas dentro de cada documento. En la actualidad existen muchos procesadores de texto que tienen esta capacidad, unos gratuitos y otros no.

⋆ ¿Es necesario emplear un procesador de texto?

Durante los estudios universitarios tendrá que realizar diversos documentos para su posterior presentación y en muchos casos evaluación. Por otra parte también es cierto que en la actualidad se da por hecho que cualquier persona que haya superado unos estudios universitarios (o incluso que esté realizando los últimos cursos de dichos estudios) debe ser capaz de generar documentos impresos, o electrónicos, con una calidad de presentación profesional. En un pasado no demasiado lejano, y por tanto familiar a muchos estudiantes de esta universidad, se consideraba como algo normal realizar estos documentos “a máquina” o incluso “a mano”, pero (algunos dirán que lamentablemente) aquellos tiempos ya pasaron. En la actualidad el estándar comúnmente aceptado es que (casi) cualquier documento que vayamos a presentar debe haber sido realizado con un ordenador por medio de un *procesador de texto*. Aparece entonces la cuestión fundamental siguiente:

1.4.1. ¿Qué podemos esperar y qué no debemos permitir de un procesador de texto?

De un procesador de texto debemos esperar que nos permita producir fácilmente documentos, con un aspecto final de alta calidad, que incluyan todas las características que hemos mencionado antes. En principio el procesador de texto debería hacer automáticamente todo el trabajo de formatear adecuadamente el texto y generar automáticamente las numeraciones de ecuaciones, figuras, tablas y referencias, así como elegir su posición adecuada, de manera que podamos centrarnos en el trabajo que realmente nos interesa que es escribir de la manera más clara posible la idea que queremos transmitir.

No deberíamos permitir, por tanto, que tareas poco interesantes desviasen nuestra atención de lo que es realmente importante. Al procesador de texto debemos exigirle que realice de manera eficaz todo el trabajo de formateo, sin necesidad de tener que invertir horas posteriormente en corregir o ajustar el aspecto del documento. Otra cosa que no deberíamos permitir de ninguna manera es que un procesador de texto nos convierta en sus esclavos, obligándonos a usar un determinado programa para acceder a la documentación, ni tampoco que tengamos que pagar periódicamente nuevas licencias para disponer de actualizaciones que realmente no queremos.

Afortunadamente en la actualidad existen diversas opciones de software libre (como p. ej. el LibreOffice, que se instala como “suite ofimática” en cualquier distribución del SO Linux) que nos permiten generar nuestros propios documentos y también abrir documentos generados, no solo con el LibreOffice, sino también con otros conocidos procesadores de textos comerciales de uso habitual. Por este motivo es muy aconsejable instalar el LibreOffice, que además de textos nos permite también generar hojas de cálculo y presentaciones, así como abrir y modificar hojas de cálculo y presentaciones generadas con otros paquetes comerciales habituales.

De todas formas, para generar fácilmente documentos científicos de gran calidad la mejor herramienta no es el LibreOffice, sino el LaTeX. El LaTeX es un programa diseñado precisamente para que podamos concentrarnos en escribir lo que queremos transmitir, olvidándonos totalmente de los detalles del formateo del documento. Básicamente, para generar un documento con LaTeX lo que hacemos es escribir un archivo de *texto plano* (normalmente con extensión `.tex`) que contiene todo el texto que queremos incluir, las ecuaciones, las tablas, los nombres de los archivos que contienen las figuras que queramos incluir, etc., junto con una serie de órdenes o instrucciones que le indican a LaTeX el significado (no el aspecto) de cada parte del documento. En este sentido el LaTeX es un lenguaje que tiene sus propios comandos.

El éxito del LaTeX se debe a que la calidad tipográfica y potencia de este sistema (para manejar documentos extensos y complejos) es muy superior a la de cualquier otro procesador de textos. Además este procesador de textos es extremadamente eficiente a la hora de sacar partido de los recursos computacionales disponibles (memoria y velocidad del procesador del ordenador), no es lógico que para escribir un mero documento con algunas ecuaciones y unas pocas figuras tengamos que comprarnos un ordenador nuevo cada año.

El uso de LaTeX está muy extendido en el mundo de las publicaciones científicas, especialmente (pero no exclusivamente) en los campos de matemáticas, física, informática e ingeniería. En la actualidad la inmensa mayoría de las publicaciones científicas de esos campos se escriben en LaTeX, tanto en lo referente a libros como a revistas periódicas. La superioridad de la calidad de la presentación de los documentos escritos en LaTeX en comparación con los realizados con los procesadores de texto comerciales habituales está fuera de toda duda, basta con una breve búsqueda en Internet para obtener abundante documentación en este sentido y por este motivo lo recomendamos fuertemente.[10]

Como todo en esta vida, el LaTeX no está exento de inconvenientes, los principales son los siguientes:

hay que aprender a usarlo y

- es un poco más rígido en su funcionamiento que los procesadores de texto tradicionales.

No es que sea muy complicado escribir documentos en LaTeX, pero hay que dedicar un poco de tiempo a aprender cómo funciona, al menos a nivel básico. Por otra parte, en LaTeX no se puede empezar a escribir un documento (una hoja en blanco) sin más,

[10] En cualquier caso, aunque nuestra recomendación es el procesador LaTeX, el uso de este procesador no es de ninguna manera obligatorio. Lo que sí deben aprender forzosamente para estar dentro de lo que en la actualidad se considera estándar es a manejar *algún* procesador de texto, el que cada uno elija con total libertad, y deben saber usarlo lo suficientemente bien como para realizar documentos con una presentación buena. De lo contrario estarían fuera de lo que se considera estándar. Si por ejemplo realiza sus documentos "a mano" y cuando se pide un archivo PDF los escanea y guarda en PDF (una práctica poco frecuente pero no del todo extinta), en ese caso estaría haciendo algo por debajo del estándar actual y consecuentemente estaría produciendo una mala impresión, independientemente de la calidad del contenido de dicho documento. Un estudiante universitario debe ser capaz de usar un procesador de textos y realizar documentos con una calidad de presentación buena. En el caso de estudiantes de ciencias esos documentos incluirán con mucha frecuencia fórmulas matemáticas, gráficas, tablas, referencias bibliográficas, enlaces a páginas web, etc., de modo que es necesario aprender a hacer todo eso. Qué procesador de texto usar es algo que cada uno debemos elegir de manera personal, de la misma forma como elegimos qué tipo de ordenador usar y con qué sistema operativo.

sino que debemos saber de antemano qué tipo de documento queremos escribir, a fin de cargar la hoja de estilos adecuada, la cual define de manera bastante rígida la estructura del documento. Los tipos de documento más habituales son "artículo" o "libro" (hay más, pero esos dos son los que más se usan). En cada uno de estos casos comenzaríamos el documento LaTeX con una instrucción del estilo de:

```
\documentclass[11pt,twoside,a4paper]{article}
\documentclass[11pt,twoside,a4paper]{book}
```

Al hacer esto estamos diciendo al procesador LaTeX que vamos a escribir un documento con una determinada estructura (posiblemente con un título y autor, capítulos, secciones, etc.) y en LaTeX sucede que introducir variaciones que se salgan de la norma en esa estructura seleccionada es complicado, por eso comentábamos antes que LaTeX "es más rígido" en su funcionamiento que los procesadores de texto tradicionales. Si el documento que quiere realizar no tiene ninguna estructura, es decir, si lo que está buscando es la versión electrónica de una verdadera "hoja en blanco", entonces LaTeX no es la mejor elección (a menos que sea un usuario experto). En ese caso la mejor elección sería un procesador de textos tradicional, como el LibreOffice, que es gratuito y funciona extraordinariamente bien en cualquier sistema operativo. Por el contrario, si lo que quiere escribir es un libro o un artículo, lo más probable (lo más lógico y lo más sensato) es que quiera ceñirse al estilo y formato que le diga su editorial, el cual especifica cosas como márgenes, tipos de letra, espaciados, etc. En LaTeX todos estos detalles de formato quedan definidos al cargar la hoja de estilos correspondiente (article, book, etc.) de modo que el autor del documento puede centrarse en el contenido del documento, no en su forma. En ese caso LaTeX, cuya filosofía de funcionamiento puede resumirse en la conocida frase:

> *"deje el diseño de estilos de documentos a los profesionales de diseño de estilos de documentos y céntrese en el contenido de su documento"*

es la mejor elección sin ninguna duda.

Como decíamos antes, LaTeX es parecido a un lenguaje de programación. Para elaborar documentos con LaTeX lo que se hace es escribir en un archivo de *texto plano* el código LaTeX que posteriormente, al ser interpretado por el programa LaTeX, genera finalmente el documento formateado final, en el correspondiente archivo de tipo `PDF`. Al principio puede parecer que esto es mucho más complicado que los procesadores de texto habituales, ya que tenemos que aprender algo de código para poder escribir un documento. Sin embargo, si disponemos de un ejemplo de código fuente en LaTeX, que contenga las instrucciones más habituales, el esfuerzo que tenemos que realizar para ponernos en marcha es realmente mínimo.[11] Al cabo de muy poco tiempo el balance entre el tiempo que hemos perdido aprendiendo a usar el LaTeX y el tiempo que hemos ganado, al no tener que pasar horas ajustando detalles ("mover cajitas" y/o márgenes, etc.) en documentos formateados con procesadores de texto malos, es extremadamente positivo. Además, para facilitarnos esta tarea en la web tenemos una fuente de información virtualmente infinita, donde podremos localizar instantáneamente cualquier instrucción LaTeX que necesitemos usar y que no conozcamos, incluyendo

[11] Por supuesto que la situación es distinta si queremos ser usuarios de LaTeX a nivel profesional, como por ejemplo el responsable de la hoja de estilos de una publicación periódica (p. ej. Physical Review Letters, formateada en LaTeX con la hoja de estilos REVTeX4) o de una editorial como p. ej. Springer-Verlag o Cambridge University Press. Para usar el procesador LaTeX a ese nivel es necesario invertir cierto tiempo de estudio. Para generar documentos a nivel de usuario normal el tiempo de aprendizaje es mínimo.

ejemplos de uso, plantillas (*templates*) e incluso documentos completos que podemos emplear como puntos de partida para generar los nuestros, cambiando el texto pero conservando las instrucciones.

En http://en.wikipedia.org/wiki/LaTeX puede encontrarse un resumen sobre este programa de procesamiento de textos, su historia y ejemplos de uso. En la actualidad este procesador de textos es el empleado para formatear los documentos de la práctica totalidad de las publicaciones científicas en los campos de matemáticas, física y por supuesto ciencias de la computación o informática (incluyendo libros de texto, artículos de investigación en revistas científicas, etc.), y en gran parte de las publicaciones de ingeniería, economía y otros campos. Por ejemplo las publicaciones generadas por las editoriales de la American Mathematical Society, American Physical Society, Cambridge University Press, Oxford University Press, Springer-Verlag, CRC press, Taylor & Francis y un larguísimo etcétera están generadas con LaTeX.

De todas formas, si por el motivo que sea decidimos no invertir absolutamente nada de tiempo en aprender un poquito de código LaTeX, en la actualidad existen programas que nos permiten generar documentos en LaTeX al estilo de los procesadores de texto tipo WYSIWYG (*What You See Is What You Get*) como el LibreOffice, es decir, visualizando el aspecto final del documento al mismo tiempo que lo escribimos y accediendo a los comandos del procesador de textos por medio de opciones en menús, sin necesidad de estudiar código LaTeX. Un ejemplo es el LyX, empleado para escribir gran parte de estos apuntes. LyX también es gratuito y de código abierto y puede instalarse con cualquier distribución de Linux (o Windows). Con el LyX podemos ir generando el documento de la misma forma que lo haríamos con el LibreOffice, generando las instrucciones de LaTeX que necesitemos por medio de las funciones disponibles en los menús que aparecen en la parte superior de la ventana del LyX. Con esto podemos generar documentos de la máxima calidad tipográfica, sin necesidad de estudiar código LaTeX. En cualquier caso, aunque al principio puede parecer que es mucho lo que uno necesita estudiar para generar un simple documento, en la práctica el número de comandos de LaTeX que se emplean es muy reducido, y cuando uno está familiarizado es algo más rápido escribir el código directamente en LaTeX (por ejemplo, para insertar ecuaciones) que navegar por los menús del LyX, aunque esto ya es una cuestión de gustos. El LyX está bastante extendido pero no es el único procesador de tipo WYSIWYW (*What You See Is What You Want*) que nos permite manejar LaTeX, otro ejemplo muy extendido es el TeXMACS (también gratuito y de código abierto), que además puede usarse como *front end* del programa MAXIMA.

Resumiendo, los programas como el LibreOffice están muy bien para escribir una nota, pero para generar un texto de alta calidad a nivel profesional el mejor procesador es LaTeX. Para generar documentos usando este procesador, sin necesidad de aprender código LaTeX, existe la posibilidad de usar el programa LyX. Para el alumno interesado indicamos al final de esta sección algunas direcciones de internet donde encontrará ejemplos de código que le serán útiles para ponerse en marcha. Al final de esta sección y en la página web de la asignatura pueden encontrarse algunos ejemplos sencillos de código LaTeX.

Un *clásico* para empezar a usar este procesador es el documento "The Not So Short Introduction to LaTeX 2_ε (*Or LaTeX 2_ε in 157 Minutes*)", disponible en

http://www.ctan.org/tex-archive/info/lshort/english/lshort.pdf

donde de forma bastante breve se presenta todo lo que uno necesita en la práctica

para saber usar este sistema[12]. La página central del programa LaTeX está en
`http://www.latex-project.org`

1.4.2. Escritura de documentos, presentaciones y pósters con LaTeX

Lo primero que debemos hacer para poder usar el procesador LaTeX es tenerlo instalado en nuestro ordenador. Para ello, desde una consola de línea de comandos debemos hacer (como root):

`dnf install texlive` (Fedora)
`apt install texlive` (Debian/Ubuntu)

una vez hecho esto ya estamos en disposición de escribir documentos, para ello creamos un archivo de texto plano (p. ej. `miarchivo.tex`) en el que escribimos el código LaTeX con el que se genera el documento. Las instrucciones

```
\documentclass[opciones]{article}
```

o

```
\documentclass[opciones]{book}
```

que mencionábamos antes nos permiten presentar el tema de las instrucciones en LaTeX. En la palabra `\documentclass` la barra `\` indica al procesador LaTeX que lo que viene a continuación no es texto normal del documento, sino una instrucción en lenguaje LaTeX, es decir, una orden o comando que especifica algo sobre cómo debe formatearse el documento. En este caso la instrucción es `\documentclass` que, como su propio nombre indica, es el comando que define el tipo, o clase, de documento que vamos a escribir, que queda especificado por el argumento "book" o "article" que va entre llaves al final de la instrucción. Los otros argumentos, los que van entre corchetes, son argumentos opcionales (es decir, se pueden omitir), e indican algunas opciones (como tamaño de la fuente, tamaño de la página, etc.). Todos los documentos en LaTeX comienzan con la instrucción `\documentclass`, seguida de otras instrucciones entre las que figura `\begin{document}` y terminan con la instrucción `\end{document}` con un significado bastante obvio. En la práctica no es necesario conocer estas instrucciones, ya que siempre podemos disponer de "plantillas de documentos" (o *templates*) que las incluyen y que podemos emplear como punto de partida para nuestros documentos.

Para procesar este archivo tecleamos en línea de comandos

```
latex miarchivo.tex
```

con esta orden LaTeX interpreta las instrucciones contenidas en `miarchivo.tex` y genera el archivo `miarchivo.dvi` (aparte de otros archivos auxiliares `miarchivo.aux` y `miarchivo.log`). El archivo `dvi` contiene el documento formateado con su aspecto final, de todas formas, la finalidad del archivo `dvi` es sólo ver cómo va quedando el documento a medida que lo escribimos, pero no es un formato diseñado para transmitir información a terceras personas. Para ello, una vez que hemos finalizado el documento debemos generar el correspondiente archivo en formato `pdf`, ésto puede hacerse usando los menús que existen en el programa usado para visualizar el archivo `dvi` (normalmente el `Okular`, o el `xdvi`, etc., todos ellos gratuitos e incluidos en cualquier distribución de Linux), o bien desde la línea de comandos del Linux, por medio del comando `dvipdf` (o del `dvipdfm`). Otra alternativa es compilar el documento con la instrucción

```
pdflatex miarchivo.tex
```

[12] Existe una traducción "oficial" al español en `http://osl.ugr.es/CTAN/info/lshort/spanish/lshort-a4.pdf`

que genera directamente el archivo `pdf` sin pasar por el `dvi`.

En la web existen multitud de páginas con información útil sobre LaTeX a cualquier nivel, desde simplemente saber un poco qué es todo eso, hasta el nivel de experto más avanzado y todos los niveles intermedios. Al final de esta sección se incluyen algunos enlaces que completan esta presentación. La primera referencia que citamos es la página de LaTeX en wikipedia, muy recomendable para tener una visión global sobre LaTeX y su funcionamiento. La segunda referencia es el clásico "The Not So Short Introduction to LaTeX" de T. Oetiker y cols., que incluye un curso práctico acelerado sobre el lenguaje LaTeX. Por medio de este curso en poco más de dos horas de trabajo (en 157 minutos para ser exactos) sabrá todo lo necesario para escribir documentos en LaTeX incluyendo figuras, tablas, ecuaciones, bibliografía, etc. Nuestra impresión personal es que la inmensa mayoría de los usuarios de este lenguaje hemos aprendido con "The Not So Short Introduction to LaTeX".

1.4.3. Ejemplos de documentos básicos en LaTeX

A continuación incluimos un ejemplo de cada uno de los tipos de documentos más habituales (*artículo* y *libro* por un lado, *póster* y *presentación* con Beamer por otro). El código de estos ejemplos está disponible en la carpeta LaTeX del curso virtual (ver archivos de texto plano con extensión `tex`) junto con los correspondientes documentos finales (archivos con extensión `pdf`), generados por medio del comando `latex + dvipdfm` para los tipos "article" y "book", y `pdflatex` para los que precisan el paquete Beamer. Todos estos ejemplos se han tomado de la web donde pueden localizarse fácilmente muchos más.

- **Ejemplo de artículo**

  Creamos un archivo de texto plano con el siguiente código (tomado de wikipedia):

```
\documentclass[12pt]{article}
\usepackage{amsmath}
\title{\LaTeX}
\date{}
\begin{document}
\maketitle
\LaTeX{} is a document preparation system for the \TeX{}
typesetting program. It offers programmable desktop publishing
features and extensive facilities for automating most aspects of
typesetting and desktop publishing, including numbering and
cross-referencing, tables and figures, page layout,
bibliographies, and much more. \LaTeX{} was originally written
in 1984 by Leslie Lamport and has become the dominant method for
using \TeX; few people write in plain \TeX{} anymore.
The current version is \LaTeXe.
% This is a comment, not shown in final output.
% The following shows typesetting power of LaTeX:
\begin{align}
       E_0 &= mc^2 \\
       E &= \frac{mc^2}{\sqrt{1-\frac{v^2}{c^2}}}
\end{align}
```

```
\end{document}
```

Como puede verse, el significado de los comandos que aparecen es muy sencillo y fácil de recordar. Como en cualquier lenguaje de programación, en el archivo donde escribimos el *código* también se pueden escribir *comentarios*, frases que el compilador sencillamente ignora, pero que son útiles para el programador (para el autor en este caso); para ello en LaTeX se emplea el carácter %.

- **Ejemplo de libro**

  Creamos un archivo de texto plano con el siguiente código (tomado de `http://www.rpi.edu/dept/arc/training/latex/Examples/exbook.tex`):

```
\documentclass[11pt]{book}          % Book class in 11 points
\parindent0pt \parskip10pt          % make block paragraphs
\raggedright                        % do not right justify

\title{\bf An Example of Book Class} % Supply information
\author{for \LaTeX\ Class}          %  for the title page.
\date{\today}                       %  Use current date.
% Note that book class by default is formatted to be printed back-to
   -back.
\begin{document}          % End of preamble, start of text.
\frontmatter              % only in book class (roman page #s)
\maketitle                % Print title page.
\tableofcontents          % Print table of contents
\mainmatter               % only in book class (arabic page #s)
\part{A Part Heading}     % Print a "part" heading
\chapter{A Main Heading}  % Print a "chapter" heading
Most of this example applies to \texttt{article} and \texttt{book}
   classes
as well as to \texttt{report} class. In \texttt{article} class,
   however,
the default position for the title information is at the top of
the first text page rather than on a separate page. Also, it is
not usual to request a table of contents with \texttt{article} class
   .

\section{A Subheading}    % Print a "section" heading
The following sectioning commands are available:
\begin{quote}             % The following text will be
part \\                   %   set off and indented.
chapter \\                % \\ forces a new line
section \\
subsection \\
subsubsection \\
paragraph \\
subparagraph
\end{quote}               % End of indented text
```

```
But note that---unlike the \texttt{book} and \texttt{report} classes
   ---the
\texttt{article} class does not have a ``chapter" command.

\end{document}              % The required last line
```

- **Ejemplo de póster y de presentación con el paquete Beamer:**

  Para compilar el ejemplo de póster y el de presentación es necesario tener cargado el paquete *Beamer*, que es un ingrediente adicional de LaTeX incluido por defecto en las versiones modernas. Beamer permite hacer presentaciones y pósters con LaTeX con un aspecto realmente bueno. Para emplear Beamer *no* se compila el documento con el comando `latex` (como en los ejemplos anteriores) sino que hay que usar el comando `pdflatex`. Al hacer esto `pdflatex` genera directamente el documento final (el póster o la presentación) en formato `pdf`, sin necesidad de pasar por el archivo intermedio `dvi`.

  Para ver el ejemplo de póster creamos un archivo de texto plano con el siguiente código, tomado de `http://tug.ctan.org/macros/latex/contrib/beamerposter/example.tex`:

```
\documentclass[final]{beamer} % beamer 3.10: do NOT use option
                              % hyperref={pdfpagelabels=false}
\mode<presentation>{
       %% check http://www-i6.informatik.rwth-aachen.de/~dreuw/
          latexbeamerposter.php for examples
\usetheme{Berlin}
       %% you should define your own theme e.g. for big headlines
          using your own logos
}
\usepackage[english]{babel}
\usepackage[latin1]{inputenc}
\usepackage{amsmath,amsthm, amssymb, latexsym}
%\usepackage{times}\usefonttheme{professionalfonts}
% times is obsolete
\usefonttheme[onlymath]{serif}
\boldmath
\usepackage[orientation=portrait,size=a0,scale=1.4,debug]{
   beamerposter}
       %e.g. for DIN-A0 poster
%\usepackage[orientation=portrait,size=a1,scale=1.4,grid,debug]{
   beamerposter}
       % e.g. for DIN-A1 poster, with optional grid and debug output
%\usepackage[size=custom,width=200,height=120,scale=2,debug]{
   beamerposter}
       % e.g. for custom size poster
%\usepackage[orientation=portrait,size=a0,scale=1.0,printer=rwth-
   glossy-uv.df]{beamerposter}
       % e.g. for DIN-A0 poster with rwth-glossy-uv printer check
% ...
```

```
%
\title[Fancy Posters]{Making Really Fancy Posters with \LaTeX}
\author[Dreuw \& Deselaers]{Philippe Dreuw and Thomas Deselaers}
\institute[RWTH Aachen University]{Human Language Technology and
   Pattern
       Recognition, RWTH Aachen University}
\date{Jul. 31th, 2007}
\begin{document}
\begin{frame}{}
       \vfill
       \begin{block}{\large Fontsizes}
              \centering
              {\tiny tiny}\par
              {\scriptsize scriptsize}\par
              {\footnotesize footnotesize}\par
              {\normalsize normalsize}\par
              {\large large}\par
              {\Large Large}\par
              {\LARGE LARGE}\par
              {\veryHuge veryHuge}\par
              {\VeryHuge VeryHuge}\par
              {\VERYHuge VERYHuge}\par
       \end{block}
       \vfill
\end{frame}
\end{document}
```

Para ver el ejemplo de presentación creamos un archivo de texto plano con el siguiente código (tomado de http://en.wikibooks.org/wiki/LaTeX/Presentations):

```
\documentclass[mathserif,serif]{beamer}
\begin{document}
\begin{frame}
       \frametitle{This is the first slide}
       %Content goes here
\end{frame}
\begin{frame}
       \frametitle{This is the second slide}
       \framesubtitle{A bit more information about this}
       %More content goes here
\end{frame}
% etc
\end{document}
```

Modifique el contenido de los ejemplos proporcionados para generar a partir de ellos sus propios documentos con formatos artículo, libro, póster o presentación. Si le surgen dificultades empleando Beamer deje este tema para más adelante, espere a estar razonablemente familiarizado con el uso de LaTeX en los tipos de documentos sencillos como “book” y sobre todo el básico “article”, antes de hacer presentaciones o pósters con Beamer.

A continuación incluimos unas pocas referencias que les serán útiles para empezar a usar LaTeX:

- Documentación general

  `http://www.latex-project.org/`

  `http://en.wikipedia.org/wiki/LaTeX`

  `http://en.wikibooks.org/wiki/LaTeX`

  `https://tobi.oetiker.ch/lshort/lshort.pdf`

- Ejemplos

  `http://en.wikibooks.org/wiki/LaTeX/Sample_LaTeX_documents`

  `http://denethor.wlu.ca/latex`

  `http://pangea.stanford.edu/computerinfo/unix/formatting/latexexample.html`

  `http://www.cs.technion.ac.il/~yogi/Courses/CS-Scientific-Writing/examples/simple/simple.htm`

- BibTeX, para la gestión eficiente referencias bibliográficas

  `http://www.bibtex.org/`

  `http://en.wikipedia.org/wiki/BibTeX`

- Beamer, para hacer presentaciones y pósters

  `http://en.wikipedia.org/wiki/Beamer_(LaTeX)`

- Para usuarios de Windows

  `http://miktex.org/`

1.5. Archivos de texto plano y editores de texto

Un tema totalmente fundamental en computación es el de los archivos de **texto plano**. Los archivos de texto plano son los empleados para escrir **código** en cualquier lenguaje de programación. Por código nos referimos al conjunto de *instrucciones*, escritas en un cierto lenguaje, con la finalidad de que el ordenador las interprete, por medio del compilador o intérprete de comandos adecuado, y realice posteriormente la serie de operaciones que hayamos programado (p. ej. realizar una serie de cálculos, generar un documento con LaTeX, conectarse con otro ordenador, etc.).

Un archivo de texto plano, por tanto, no es un documento con instrucciones de formato complicadas, sino un archivo bastante básico, que contiene exclusivamente caracteres estándar ASCII, es decir, letras, números, signos estándar de puntuación, espacios y saltos de línea, etc., pero no caracteres no estándar (como p. ej. letras griegas o símbolos matemáticos no estándar), ni tampoco instrucciones de formato, es decir, instrucciones sobre cómo debe mostrarse ese texto (aparte de instrucciones totalmente básicas, como espacios o saltos de línea). Los archivos de texto plano no

contienen elementos complejos como textos resaltados o escritos con distintos tipos de fuente, distintos colores, tablas, figuras, etc. Como decíamos antes los archivos de texto plano son los que generalmente se emplean para escribir código en cualquier lenguaje de programación (Bash, C, C++, Fortran, LaTeX, Maple, Mathematica, Matlab, Maxima, Octave, Perl, TeX...). Este tipo de archivos contiene exclusivamente el conjunto de instrucciones, es decir, el código, del programa que sea, o en el caso de archivos tipo `tex` contienen el conjunto de instrucciones LaTeX que generan, por medio del comando `latex`, el documento formateado final.

1.5.1. Nombres y extensiones de archivos

Es muy importante tener en cuenta que los documentos generados por cualquier procesador de textos (LibreOffice, Word, etc.) **no** son archivos de texto plano. Los documentos con formato `pdf` tampoco son archivos de texto plano. El motivo es que estos tipos de archivos contienen elementos que no forman parte de esos caracteres que mencionábamos más arriba. También es importante darse cuenta que lo que define un archivo de texto plano es el contenido del archivo, no su extensión. Algunos sistemas operativos tienden a ocultar al usuario las extensiones de los tipos de archivos "conocidos" (lo que es, en muchos casos, una brecha de seguridad por la que se ejecuta código malicioso), y fomentan la confusión al dar la impresión que lo que define el tipo de un archivo es su extensión, no su contenido. Cuando escribimos un archivo de texto plano podemos darle el nombre de archivo y extensión que queramos, incluso podemos no darle ninguna extensión, y esto no afectará de ninguna manera al tipo de archivo, ya que no afecta a su contenido.

De todas formas, para organizar la información de manera sencilla e intuitiva y para facilitar la comunicación entre programadores existe la convención de asignar ciertas extensiones estándar a los archivos de texto plano con código fuente de diversos lenguajes. Como ya hemos mencionado se suele dar la extensión `.tex` a los archivos con código LaTeX, en programación en C es habitual dar la extensión `.c` a los archivos con código C. En este curso trabajaremos con wxMaxima, y cuando se escriben programas en Maxima se suelen usar las extensiones `.mac` o `.mc`, de manera indistinta. Para otros lenguajes se emplean otras extensiones, para un archivo de texto plano de tipo genérico, no asociado a ningún lenguaje de programación en particular, se suele emplear la extensión `.txt`. Insistimos en que no es la extensión del archivo (ni el icono que le asocia un explorador) lo que define su tipo, sino su contenido.

Aunque anteriormente hemos mencionado que un SO serio debería permitirnos dar a nuestros archivos el nombre y extensión que queramos, existe un problema cuando damos a los archivos, o a los directorios que los contienen, nombres con caracteres no estándar. Los sistemas operativos modermos permiten asignar a archivos y directorios nombres con caracteres arbitrarios, como por ejemplo "acentos" (...Física...) y/o espacios (...Física Computacional I...) entre otros. Aunque el sistema operativo lo permita, es muy frecuente que al final eso sea fuente de multitud de problemas, principalmente porque no todos los sistemas operativos (o en todas las regiones del mundo) se interpretan de igual modo estos caracteres; de modo que es algo que conviene evitar. Esto sucede, p. ej. en wxMaxima, cuando intentamos cargar un archivo por medio de la instrucción `batchload(concat(path, ''nombre-del-archivo''))` (siendo *path* una cadena de caracteres que indica el directorio donde está nuestro archivo): si, o bien en el *path*, o bien en el nombre del archivo tenemos algún nombre con acentos, espacios,

etc. el wxMaxima no entenderá correctamente el directorio donde se halla o el nombre del archivo en el disco, y no podrá cargarlo. Tener una ortografía y sintaxis correcta es muy importante cuando escribimos un texto de cualquier tipo, pero a la hora de poner nombres a archivos y directorios en un ordenador lo mejor para evitarnos disgustos posteriores es olvidarnos del uso de acentos, espacios, letra ñ, ... En el caso de los espacios, es recomendable sustituirlos por guiones "-", o guiones bajos "_". En el caso de los acentos sencillamente debemos omitirlos y para la letra "ñ"se puede sustituir sencillamente por "n", o por "nn" o por cualquier otra combinación de caracteres que nos resulte cercana ("nh", "gn", ...). De esta forma, en lugar de denominar "`Física Computacional I`" al directorio donde alojaremos el material de esta asignatura, nos ahorraremos muchos disgustos si lo denominamos "`FISICA_COMPUTACIONAL--I`".

1.5.2. Editores de texto

Finalmente, llegamos a la importante cuestión sobre software:

⋆ ¿qué programa se emplea para escribir archivos de texto plano?

Como hemos mencionado anteriormente, los archivos de texto plano **no** se escriben con procesadores de texto (LibreOffice, Word, etc.), ya que éstos generan documentos con caracteres no visibles (en general instrucciones de formato o contenido del archivo en formato comprimido) independientemente de cómo de sencillo o básico sea el documento que hemos escrito.

Para escribir archivos de texto plano se emplean unos programas bastante básicos denominados *editores de texto*. Es muy importante saber diferenciar entre un editor de texto y un procesador de texto. Como su propio nombre indica, lo único que permiten hacer los editores de texto es crear, mostrar y modificar archivos de texto plano, mientras que los procesadores de texto son programas bastante más complejos que permiten escribir documentos en los que, aparte del texto, existen instrucciones sobre cómo debe mostrarse dicho texto.

Existen multitud de editores de texto, algunos de ellos son los siguientes: gedit, nedit, kwrite, kate, geany, emacs, xemacs, vi, vim, etc. Para sistemas con Windows, aparte de los anteriores, puede usarse el notepad2 (por favor, no usen el notepad); también algunos de los editores de Linux están disponibles en Windows (p.ej. gedit, kate, geany). Los editores de texto están programados con la finalidad de ser útiles para escribir código en diversos lenguajes de programación. Por este motivo, la mayoría de los editores de texto modernos son capaces de reconocer la sintaxis diversos lenguajes de programación habituales y nos ofrecen la posibilidad de *resaltar esta sintaxis*. Para ello, estos editores nos muestran el texto empleando colores diferentes para distintos tipos de palabras clave (comandos o funciones del lenguaje) que el editor reconoce. Esto es muy útil a la hora de programar, ya que permite visualizar rápidamente errores de sintaxis comunes (como tener paréntesis mal cerrados) y no afecta al contenido de nuestro archivo de código fuente, que sigue siendo texto plano. Nótese que es el significado de las instrucciones o signos en ese lenguaje lo que determina el color o tipo de letra que le asigna el editor de texto; nosotros no podemos decidir que una instrucción particular aparezca de un color diferente a las demás, como haríamos con un procesador de textos (donde, por ejemplo, podríamos hacer que una palabra en particular apareciese en negrilla).

En muchos lenguajes de programación existen entornos de trabajo que incluyen en una única herramienta la ventana con el código fuente que estamos escribiendo, la ventana con la órdenes enviadas al compilador, la ventana con el output de nuestro programa y otras. Este tipo de entornos de trabajo pueden ser muy cómodos y útiles, pero en ocasiones generan la impresión equivocada de que el editor de texto empleado para escribir código en un cierto lenguaje sólo sirve para eso, y no puede utilizarse como un editor de texto cualquiera. En realidad esto no es así, cuando escribimos código fuente en un lenguaje de programación podemos emplear el editor de texto que queramos, independientemente del lenguaje de programación que estemos empleando. Por ejemplo, para programar funciones en Maxima una posibilidad es escribir el código de dichas funciones en la ventana del wxMaxima y posteriormente usar los menús para exportarlas en un archivo con extensión `.mac`, pero es mucho más cómodo usar un editor de texto ajeno al wxMaxima (como el KWrite) para escribir el código fuente, y sencillamente importar el código desde wxMaxima por medio de la orden `batchload(...)`.

Para finalizar incluimos a continuación algunos enlaces con información útil sobre editores de texto.

- Documentación general

  `https://en.wikipedia.org/wiki/Text_editor`

- Lista de editores de texto

  `https://en.wikipedia.org/wiki/List_of_text_editors`

- Comparativa de editores de texto

  `https://en.wikipedia.org/wiki/Comparison_of_text_editors`

La recomendación que hacemos en este sentido antes de decantarnos por un editor paricular es: *probar varios editores distintos*, naturalmente que sean compatibles con el SO que estemos empleando y, por supuesto, gratuitos.

1.6. Documentación científica

1.6.1. Tipos de publicaciones científicas más habituales

Tradicionalmente los estudios de ciencias físicas han estado totalmente centrados en los contenidos de física y matemáticas, dejando de lado cualquier aspecto técnico del desempeño de la profesión. Con el tiempo se ha observado que esto perjudicaba en cierta medida a muchos titulados en física, que, una vez finalizados sus estudios, debían emplear cierto tiempo en aprender este tipo de cosas. Un aspecto positivo de los nuevos temarios en física es prestar algo de atención a estas cuestiones. Con esta finalidad hemos presentado en este capítulo de introducción a la física computacional algunas notas generales sobre uso de ordenadores, sistemas operativos y procesadores de texto, y vamos a cerrar esta sección con una breve descripción de los tipos de publicaciones científicas más habituales, en particular los *artículos de investigación*.

Está fuera de discusión que muy pocos alumnos querrán leer artículos de investigación durante los primeros cursos del grado en física, pero no está de más tener algo de

cultura general sobre qué tipo de publicaciones científicas existen y cómo está estructurada en ellas la información. Los alumnos que se enfrenten al trabajo fin de grado o, tras finalizar éste, a un máster o, luego, al doctorado, encontrarán útil esta información. La investigación y la docencia son las salidas profesionales más conocidas para los titulados en física, pero no son en absoluto las únicas actividades profesionales que realizan los físicos. Además de éstas hay otras muchas profesiones relacionadas con el mundo de la tecnología donde trabajan físicos, junto con titulados de otras carreras, incluyendo, informática, ingeniería, meteorología, medicina, economía, ... De todas formas para un físico la investigación es un referente que constantemente aparece y es muy frecuente que los profesionales del mundo de la tecnología (y también de la docencia) que deseen mantenerse más o menos *al día* en su tema de trabajo, se vean obligados a leer periódicamente diversas publicaciones científicas. Por eso esperamos que este breve vistazo global al tema de las publicaciones científicas sea de interés, a largo plazo, para todos.

Siguiendo los pasos establecidos en el **método científico**, el objetivo último de la investigación es transmitir los resultados obtenidos, de modo que otros investigadores puedan:

a. *reproducirlos* y de esta forma *verificarlos* y

b. tomarlos como *punto de partida* en su trabajo.

Esto se realiza por medio de una serie de publicaciones periódicas, de ámbito internacional, en las cuales se publica el trabajo realizado por los investigadores en forma de *artículos de investigación*. En la actualidad el lenguaje que se ha impuesto en el mundo de la investigación científica es el inglés (análogamente a como sucede en el mundo de la economía, la tecnología, la industria, la medicina, las comunicaciones, ...), de modo que para el científico es imprescindible ser capaz de manejar esta lengua.

Actualmente existen multitud de publicaciones científicas. Algunas de ellas son muy *generales*, como las prestigiosas revistas *Nature* o *Science*, donde se publican artículos que pueden ser de interés para muchos campos científicos distintos (física, biología, medicina, ...). En un siguiente nivel de especificidad nos encontramos con revistas todavía de carácter general, pero que ya afectan a un único campo científico; p. ej. en física son especialmente conocidas las publicaciones *Reviews of Modern Physics* y *Physical Review Letters*, editadas por la American Physical Society. A continuación nos encontramos con revistas especializadas en un tema en particular, como p. ej. el *Journal of Fluid Mechanics*, e incluso altamente especializadas, como puede ser p. ej. el *Journal of Non-Newtonian Fluid Mechanics*.

Normalmente los artículos publicados en revistas muy especializadas están dirigidos a un público experto en esa materia. En estas publicaciones se da por sentado que el lector posee un conocimiento profundo de los detalles particulares del tema de que se trata, de modo que el autor suele centrarse en los aspectos más relevantes de lo que ha sido su aportación al estado del conocimiento del tema (denominado habitualmente *estado del arte*). Si a esto sumamos que los editores de estas revistas suelen imponer límites más o menos estrictos en la extensión de cada artículo, encontramos que al final estas publicaciones sólo están al alcance de los que trabajan en ese tema. En las revistas más generalistas, en cambio, las publicaciones están dirigidas a un público más amplio, centrándose los trabajos en las implicaciones y consecuencias de la investigación realizada desde un punto de vista generalista, dejando de lado los detalles más técnicos.

Una vez dentro de un tema de investigación específico existen dos tipos de artículos de investigación. Por un lado están los artículos, digamos normales, en los que un grupo de investigadores publica una aportación puntual concreta, y por otro los artículos de *revisión del estado del arte* en los que un autor de reconocido prestigio (con una larga trayectoria de investigación a sus espaldas) realiza una puesta al día sobre la investigación en un tema en concreto de su especialidad. Los artículos de investigación normales suelen tener una extensión que va desde unas pocas páginas a unas pocas decenas de páginas, normalmente son extremadamente especializados y la cuestión que explican o resuelven es tremendamente específica, siendo de lectura muy difícil para los no familiarizados con el tema. En cuanto a los artículos de puesta al día o revisión del *estado del arte* de una materia en concreto, denominados *Reviews*, suelen ser muy extensos (entre 100 y 200 páginas es lo normal) y suelen incluir un número muy elevado de referencias bibliográficas, lógicamente. Cuando uno se introduce en el mundo de la investigación lo primero que debe hacer es localizar los artículos de *Review* del tema de que se trate y asimilar toda la información proporcionada por estas publicaciones sobre el *estado del arte* del tema considerado. Esto nos permitirá aprender qué cuestiones de nuestro tema de investigación han sido ya resueltas (y qué métodos se han empleado para ello) y, lo que es más interesante, qué problemas siguen todavía sin solución. Es muy importante estudiar a fondo los *Reviews* disponibles sobre el tema de investigación en el que uno empieza a trabajar, ya que de lo contrario es fácil caer en el error de hacer algo que ya está hecho, o peor aún, repetir errores ya cometidos por otros.

Dentro de los artículos normales, en los que uno publica sus resultados según los va obteniendo, un tipo particular de éstos son las *letters* (a veces llamadas *fast communications* o *short communications*). En ellos se publica de manera muy rápida y breve aquellos resultados que se consideran especialmente relevantes, para cuya consecución compiten diversos grupos de investigación. Cuando un grupo publica una *letter* es frecuente que posteriormente publique otro artículo más extenso, explicando sus resultados de una manera más amplia y detallada, analizando en profundidad las consecuencias de la investigación realizada no incluidas en la *letter*.

En la mayoría de los temas de investigación es muy frecuente que las revistas especializadas publiquen sólo un tipo determinado de artículos (o bien *reviews*, o bien *letters*, o bien artículos normales). Por ejemplo, las cerca de 50 revistas publicadas anualmente por la organización *Annual Reviews* (especializadas en sus correspondientes campos científicos y disponibles en la página `http://www.annualreviews.org`) contienen solamente artículos de revisión, algo parecido sucede en la prestigiosa *Reviews of Modern Physics*, donde se publican revisiones de temas pertenecientes a diversos campos de la física, mientras que la revista *Physical Review Letters* sólo publica *letters* (estas dos últimas publicaciones están disponibles en la página de la American Physical Society, `http://www.aps.org`, concretamente en `http://publish.aps.org/browse.html`).

Antiguamente mantenerse al día con la literatura de un tema de investigación era una tarea muy ardua. Uno a veces tenía grandes dificultades para acceder a publicaciones a las que su propia universidad no estaba suscrita (las suscripciones institucionales a estas publicaciones científicas suelen ser muy costosas). Hoy en día con Internet el acceso a la documentación científica es considerablemente más sencillo, y existen acuerdos entre distintas universidades que permiten el intercambio de este tipo de información de una manera muy rápida y eficiente. Aparte de los buscadores

habituales (Google) existe un portal de Internet especialmente orientado para esto

`http://isiwebofknowledge.com`

perteneciente a la agencia *Thomson Scientific's Institute for Scientific Information*. Para acceder a los servicios ofrecidos por esta agencia desde una institución suscrita (requisito indispensable) a este servicio en España (universidades, p. ej. UNED, centros de investigación públicos o privados, etc.) hay que ir a

`http://www.accesowok.fecyt.es/login/`

En este portal uno puede realizar todo tipo de búsquedas bibliográficas (por temas, por autores, por revistas, etc.). Así mismo cada publicación mostrada por este portal contiene enlaces virtuales a las publicaciones a las que cita y también a las publicaciones posteriores que citan a ésta, lo cual es extremadamente útil. Gracias a las herramientas de búsqueda que este portal de Internet proporciona, es muy sencillo mantenerse al día con lo que se publica en un tema y, cuando uno sabe qué es lo que busca, es muy sencillo encontrarlo.

1.6.2. Estructura de un artículo de investigación

En general la estructura típica de un artículo de investigación estándar es la siguiente:

- Título, autores e información de contacto
- Resumen (*Abstract*)
- Lista de palabras clave (*keywords*)
- Introducción (*Introduction*)
- Contenido del artículo dividido en secciones
- Conclusiones (*Conclusions*)
- Bibliografía (*Bibliography*)
- En ocasiones algunos apéndices adicionales (cuando la descripción de los detalles técnicos de alguna sección se hace demasiado extensa)

En todas las publicaciones científicas se hace énfasis en que el lenguaje debe ser tan claro y conciso como sea posible. El título sebe ser escueto y representativo del contenido del trabajo y la lista de palabras clave, que indica los temas de investigación con los que está relacionado el trabajo, debe ser tan precisa como sea posible. Todos los artículos de investigación comienzan con un breve resumen (el *Abstract*), del orden de unas 500 palabras a lo sumo (con frecuencia mucho menos), en él se debe escribir de la manera más sucinta y clara posible cuál es la aportación concreta de este artículo al tema de que se trate. La mayoría de los investigadores seleccionan los artículos que les parecen relevantes, o bien porque aparecen citados en otra publicación, o bien por la información contenida en el Abstract, por este motivo es de vital importancia que el Abstract describa de forma muy precisa (y escueta) la aportación realizada. El siguiente elemento más importante del artículo es la sección de conclusiones (siempre al final del artículo), en ella se destacan, de forma más extensa y concreta que en el Abstract, las aportaciones originales del trabajo realizado, situándolas en el contexto del tema

de investigación de que se trate, es decir, relacionando la aportación realizada con el *estado del arte* antes de esta publicación y con las cuestiones que todavía permanecen abiertas. En la actualidad el número de trabajos de investigación publicados cada mes es tan grande que en la práctica resulta imposible leer todo el material publicado sobre un tema en concreto, de modo que de estos tres elementos, el título, el abstract y las conclusiones, depende en gran medida que nuestro trabajo sea revisado o descartado directamente por los demás investigadores. Por supuesto, también influirá el prestigio de la revista en la que publiquemos nuestro trabajo y el hecho de que sea citado posteriormente por otros investigadores.

En cuanto a las secciones restantes del artículo, en la introducción suele hacerse un breve resumen del *estado del arte* del tema de que se trate, citando las referencias bibliográficas que contengan los avances realizados previamente, junto con una descripción de la motivación del presente estudio y los objetivos buscados. Para hacer énfasis en la relevancia del trabajo realizado es muy frecuente mencionar, en esta sección, las conexiones de nuestro trabajo con otros temas de investigación y sus posibles aplicaciones prácticas. Después de la introducción viene un número variable de secciones en el que se describe el trabajo realizado propiamente dicho. En estas secciones es donde se describirá con todo detalle los experimentos realizados (incluyendo materiales, métodos, detalles técnicos del dispositivo experimental, dificultades encontradas, resultados, tratamiento estadístico de los resultados obtenidos, ...), los cálculos numéricos realizados (incluyendo información sobre los algoritmos empleados, criterios de convergencia, tiempo de cálculo, resultados, ...) o los desarrollos matemáticos teóricos realizados (incluyendo información sobre la notación empleada, aproximaciones realizadas, resultados, ...). Todo el trabajo debe estar redactado de una forma clara y concisa, casi esquemática, al mismo tiempo que precisa y lo suficientemente completa como para que otro equipo de investigación sea capaz de reproducir totalmente nuestros resultados, incluyendo los desarrollos matemáticos teóricos, cálculos numéricos y resultados experimentales, a partir de la información contenida en nuestro trabajo junto con las publicaciones previas que allí se citan.

Finalmente, los artículos de revisión *Reviews* tienen una estructura similar, aunque debido a su extensión es muy frecuente que incorporen un índice (como el de un libro) entre el Abstract y la Introducción. En cuanto a los artículos de tipo *letter*, al ser muy cortos tienen una estructura más sencilla, de hecho con mucha frecuencia no están divididos en secciones, sino que sólo tienen el Abstract y una única sección a continuación, como si se tratara de una *carta*.

Con esto terminamos este breve resumen sobre los tipos más habituales de publicaciones científicas. Por supuesto que este resumen está muy lejos de ser exhaustivo, hay otros tipos de publicaciones, de acceso más restringido (y uso menos frecuente), que no hemos mencionado, como p. ej. las Tesis Doctorales, las memorias de Proyectos de Investigación (financiados por organismos públicos o privados), los informes anuales que realizan algunos centros de investigación, algunos registros de patentes, informes internos de empresas relacionadas con la tecnología, o las actas (*proceedings*) de congresos de investigación.