Tema 1: Introducción

509: Informática básica

2006/2007

Índice

Presentación asignatura

"Informática Básica" es una asignatura de carácter **obligatorio** que se imparte en el primer curso de la titulación de Ingeniería Técnica en Diseño Industrial (ITDI) y que pretende proporcionar al estudiante conocimientos básicos sobre la **estructura y el funcionamiento de un ordenador**, los conceptos fundamentales de la **programación** y el **manejo básico** de un ordenador personal (Sistema Operativo) y sobre algunas aplicaciones de interés (Diseño de páginas web e Internet). Consta de

- 4,5 créditos de teoría y problemas (programación),
- 3 créditos de prácticas (diseño de páginas web y programación).

Se imparte durante el primer semestre del curso. Por tanto, las convocatorias **ordinarias** de examen son en **febrero** y en **septiembre**.

Teoría:

- Visión general de la informática (hardware y software).
- Concepto de Sistema Operativo.
- Codificación interna de la información en los ordenadores (representación multimedia).
- Principios básicos de la comunicación entre ordenadores (redes e Internet).
- Conceptos fundamentales de programación (lenguaje Python).

En prácticas se utiliza mayoritariamente **software libre**. Veáse:

`http://www.gnu.org/philosophy/free-sw.es.html`

- Entorno operativo Windows XP.
- Herramientas de comunicación en Internet (navegador Firefox, Filezilla, pasarela Webmail).
- Diseño de páginas web (NVU), edición de imágenes/retoque fotográfico (GIMP) y animaciones (Flash, SVG).
- Desarrollo de programas sencillos con Python (PythonG).

Objetivos

- Conocer la estructura y el funcionamiento básico de un ordenador y aprender la utilidad básica de un Sistema Operativo.
- Conocer los principios básicos de la comunicación entre ordenadores.
- Conocer el hardware del PC, en especial dispositivos orientados a entrada y salida gráfica.
- Realizar programas sencillos mediante el lenguaje de programación Python.
- Comprender los fundamentos del diseño de páginas web, edición de imágenes y de las principales herramientas de comunicación en Internet.

Teoría

Las clases de teoría presentan los conceptos básicos.

Algunos de los conocimientos impartidos en clases de teoría son necesarios para realizar con éxito las prácticas.

Tema	**Título**	**Sesiones** (aprox.)
1	Introducción	2
2	Sistemas Operativos	2–3
3	Codificación de la información	3
4	Redes de ordenadores. Internet	2–3
5	Hardware	3
6	Introducción a la programación. El lenguage Python	15

Prácticas

Las sesiones de prácticas se desarrollan en aulas informáticas.

Práctica	**Materia**	**Sesiones** (aprox.)
1	El entorno de trabajo Windows XP	1
2	Internet (correo-e, navegación, FTP, SFTP)	1
3	Diseño de páginas web (NVU, GIMP, Flash, SVG)	6–7
4	Programación en Python	5–6

Además, las **sesiones** de teoría dedicadas a la realización de **problemas** versarán sobre cuestiones de **programación**.

En reprografía encontraréis una copia reducida de las **transparencias** usadas en clase. También podéis descargaros el fichero correspondiente, en formato PDF, de la **página web** de la asignatura. En el tema 6 ("Introducción a la programación") no se usan transparencias, pero se ha elaborado un cuaderno de apuntes que incluye todos los conceptos, problemas y ejercicios vistos en clase.

Es importante que asistáis a clase con la copia reducida de las transparencias. De ese modo tendréis que tomar menos apuntes y podréis prestar más atención a las explicaciones.

Para realizar cada sesión de prácticas se dispone de un **enunciado** donde se explican una serie de conceptos que podréis probar mediante la ejecución de una serie de ejemplos y la realización de los ejercicios propuestos.

Dicho enunciado estará disponible en **reprografía** (al menos) el día antes de la sesión práctica correspondiente. También podréis descargar de la **página web** de la asignatura el fichero PDF correspondiente.

Es importante que leáis este enunciado *antes* de empezar cada sesión práctica, anotando todas aquellas dudas que os puedan surgir para que podáis resolverlas consultando al profesor de prácticas. De lo contrario, es posible que no dispongáis de tiempo suficiente.

Para poder realizar el examen es necesario obtener APTO en prácticas. Para ello, habrá que **entregar** algunos de los ejercicios planteados en los bloques 3 (diseño web) y 4 (programación).

Evaluación

Examen consistente de tres partes:

- prueba de **teoría** (sin apuntes): 20 cuestiones tipo test sobre los **5 primeros temas** (4 opciones, 3 preguntas contestadas erróneamente anulan una bien). **Duración: 30 minutos**. **Puntuación:** $2{,}5$ **ptos.**
- prueba de **problemas** (con apuntes): cuestiones y problemas sobre programación en Python. **Duración: 2 horas**. **Puntuación:** 5 **ptos.**
- prueba de **prácticas** en **aula informática** (con apuntes): los ejercicios a realizar con ordenador versarán sobre los contenidos vistos en las 8 primeras sesiones de prácticas (todo excepto programación). **Duración: 1 hora**. **Puntuación:** $2{,}5$ **ptos.**

Además, existe la posibilidad de obtener **hasta** $1{,}5$ puntos adicionales por la realización de **controles y trabajos voluntarios** que se irán planteando **a lo largo del curso**. El examen será el **19 de enero de 2007**.

Bibliografía

- Apuntes y transparencias de la asignatura disponibles en reprografía y en la página web.
- "Introducción a la informática: hardware, software y teleinformática", de Miguel Ángel Sánchez Vidales, Publicaciones Universidad Pontificia de Salamanca (2001).
- "Digital Multimedia", de Nigel Chapman y Jenny Chapman, John Wiley & Sons (2000).
- "Internet. Manual de referencia", de H. Hahn, McGraw-Hill (1994).
- "CAD/CAM Theory and Practice", de I. Zeid, McGraw-Hill (1991).
- "Learning Python", de M. Lutz y D. Ascher, O'Reilly & Associates (1999).

Página web

Hay una página web con información de la asignatura en el **Aula Virtual** de la UJI:

`http://aulavirtual.uji.es`

A través de la página puedes:

- descargar apuntes y boletines de prácticas,
- descargar software gratuito utilizado en las prácticas,
- participar en foros de opinión y consulta,
- informarte de las novedades que puedan afectar a la asignatura,
- consultar soluciones a ejercicios, las calificaciones obtenidas, exámenes de cursos anteriores, etc.

Conceptos básicos

Informática: Conjunto de conocimientos científicos y técnicas que hacen posible el **tratamiento automático de la información** por medio de **ordenadores**.

- **Tratamiento de la información**: Aplicación sistemática de uno o varios **programas** sobre un conjunto de datos para utilizar la información que contienen. Ejemplos: ordenar un conjunto de datos, dar formato a un texto, calcular estadísticas, etc.
- **Ordenador**: **Máquina electrónica** dotada de una memoria de gran capacidad y de métodos de tratamiento de la información, capaz de resolver problemas aritméticos y lógicos gracias a la utilización automática de **programas** registrados en ella.

Un ordenador trabaja con dos tipos de informaciones:

- **Instrucciones**: le indican qué operaciones o transformaciones tiene que hacer.
- **Datos**: Los elementos sobre los que actúan o que generan las instrucciones. Pueden ser cosas tan distintas como una temperatura, una altura, un sonido, una imagen, un documento, etc.

Un **programa** informático es la secuencia de instrucciones que realizan una o varias tareas. El ordenador **carga en su memoria** los programas y los ejecuta.

En general, un ordenador puede ejecutar diversos tipos de programas con lo que se tiene una gran **flexibilidad** para resolver distintos problemas.

Hardware: conjunto físico de todos los dispositivos y elementos internos y externos del ordenador.

Software: conjunto de datos y programas. No tiene existencia material (aunque se presente sobre un soporte físico como DVD-ROM, CD-ROM, disquete, etc.).

Una analogía: en un libro

- el papel, las tapas, la tinta constituyen el **hardware**,
- el texto escrito, lo que se dice, es el **software**.

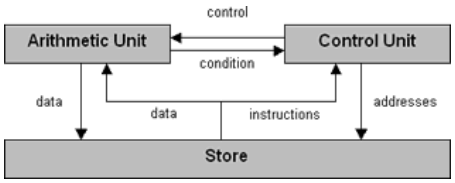
Hardware: lo que puedes partir con un hacha.

Software: lo que sólo puedes maldecir.

Hardware. Estructura básica de un ordenador

- **1833**: Charles **Babbage** ideó una máquina (mecánica) de cálculo programable mediante tarjetas perforadas: la **máquina analítica**. La máquina nunca se terminó de construir: la tecnología necesaria no estaba lo suficientemente madura.
- **1948**: John von Neuman propuso utilizar la estructuración ideada por Babbage para construir ordenadores electrónicos: **unidad de control**, **unidad aritmético-lógica** y **memoria**. A esto le incorporó dos **nuevas ideas**:
  - **Representación binaria de la información**: ordenadores binarios, idea ya utilizada por **Konrad Zuse** en **1938** para su Z1.
  - Los programas a ejecutar por el ordenador y los datos manejados por éstos debían **almacenarse en su memoria**.

Las reglas de diseño de la **arquitectura von Neuman** continúan vigentes hoy en día para la construcción de ordenadores: la unidad de control y la unidad aritmético-lógica componen la **Unidad Central de Proceso** (siglas en inglés: CPU) que es el componente fundamental del **procesador** principal de cada ordenador, todos usan memoria RAM para almacenar tanto los datos como los programas que se ejecutan para tratarlos y todos utilizan el sistema de representación binario.



Ahora bien, luego, cada ordenador se diseña conforme a una cierta **arquitectura específica**, que tiene que ver con:

- el tipo de procesador empleado,
- circuitería,
- modo de interconexión entre los elementos,
- etc.

La arquitectura más popular hoy en día es la conocida como PC (del *Personal Computer* diseñado por IBM a principios de los 80) o x86. Razones:

- es económica,
- posee buenas prestaciones para gran tipo de aplicaciones,
- dispone de un gran número de aplicaciones (programas) disponibles para uso doméstico, ofimática, multimedia, bases de datos, programación profesional, etc.

Existen otras arquitecturas: p.e. Apple Macintosh (los populares **Mac**).

- Los procesadores empleados son distintos a los del PC: PowerPC de Apple/IBM/Motorola (aunque los nuevos modelos también llevan Intel) vs. Intel Pentium o AMD Athlon.
- La circuitería es distinta: los procesadores se conectan de distinta manera con el resto de componentes, aunque éstos son similares.

Los programas se construyen para **funcionar en una arquitectura concreta**: no podemos ejecutar un programa para PC en un Mac y viceversa. Como veremos posteriormente, aún hay más restricciones.

Los Mac son ampliamente utilizados en el **mundo del diseño**: edición de vídeo, programas 3D, creación de páginas web, maquetación de libros, multimedia, etc. Sus inconvenientes:

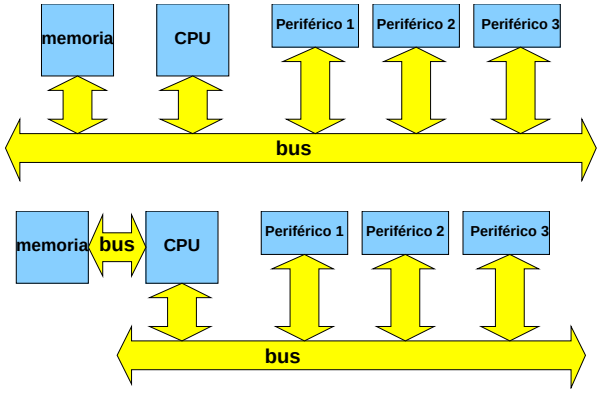
- El precio (en comparación con un PC).
- Son cajas "cerradas": al no poderse "abrir" no son ampliables, ni escalables, ni configurables (clónicos). P.e.: no podemos cambiar la tarjeta de sonido, ni ampliar memoria RAM.
- No se dispone de la enorme cantidad de programas que se tiene para PC, aunque esto cada vez es menos cierto gracias al **software libre**.

Nos centraremos en la arquitectura PC. Sus principales elementos son:

- La **CPU** o **microprocesador**: el "cerebro" del ordenador;
- la **memoria principal**: almacén de datos y programas que está usando el ordenador;
- las **tarjetas de expansión** (p.e. vídeo y sonido);
- el **bus**: pistas de comunicación entre elementos;
- la **placa base**: circuitería a la que se conectan los anteriores elementos;

- los **periféricos**:
  - de **entrada**: teclado, ratón, joystick, micrófono, escáner, CD-ROM. . .
  - de **salida**: monitor, impresora, altavoces. . .
  - de **entrada/salida**: discos duros, CD-RW. . .
- la **fuente de alimentación**;
- la **caja**: armazón sobre el que se montan los elementos internos del ordenador.

Así se conectan *grosso modo*:



Software. Lenguajes de programación

El **soporte lógico** o **software** de un ordenador se refiere tanto al **conjunto de programas** (proporcionados por el fabricante, adquiridos a empresas específicas y/o desarrollados por el propio usuario) como a **los datos** que éstos manipulan.

Para que un ordenador realice diversas funciones hay que detallarle **todo** lo que tiene que hacer, y **sólo** lo que tiene que hacer. Esto se lleva a cabo mediante los *programas*.

El ordenador es una máquina inútil sin software. Por ejemplo, ¿qué podríamos hacer con un PC sin Sistema Operativo (software que lo "maneja")? Por su funcionalidad podemos distinguir tres grandes grupos de software:

1.- **software de control**: facilita el uso del hardware y lo administra, haciendo eficiente el uso del ordenador, y/o supervisa la ejecución de otros programas. Ejemplo: Sistema Operativo, administrador de impresión, etc.

2.- **software de tratamiento**: los programas que utilizan los usuarios para su trabajo (diseño gráfico, procesamiento de textos, cálculo, edición de audio y vídeo, etc.) u ocio (escuchar música, ver películas, "chatear", juegos, etc.). A su vez, podemos distinguir:
   a.- **software de programación**: programas que facilitan la realización de las propias aplicaciones de los usuarios, es decir, programas que ayudan a programar (intérpretes, compiladores, depuradores, editores de texto, etc.).
   b.- **software de aplicación**: programas que implementan una aplicación específica (el resto: procesadores de texto, hojas de cálculo, diseño asistido, dibujo vectorial, reproductores MP3, reproductores DivX, etc). También se incluyen los programas desarrollados por los usuarios para una aplicación concreta.

3.- **software de diagnóstico y mantenimiento**: los programas que utilizan las personas responsables del mantenimiento y puesta al día del hardware y software del sistema (antivirus, comprobadores de memoria, reparadores de disco, programas para copia de seguridad, instalación y actualización de programas, comprobadores de integridad, etc.).

Realmente, un **programa** es un **algoritmo** descrito en un **lenguaje de programación** que el ordenador es capaz de interpretar.

Denominamos **proceso** a un programa en ejecución junto con los datos que manipula.

Un **algoritmo** es

- coloquialmente: una **receta** para resolver un problema;
- formalmente: descripción precisa (no ambigua) y ordenada de una **secuencia de operaciones** bien definidas que permiten obtener un resultado a partir de unos datos de entrada en un **tiempo finito** y utilizando una **cantidad finita de memoria**.

Los **lenguajes de programación** son lenguajes **artificiales** que definen, mediante unas reglas muy precisas y rigurosas

- qué palabras (símbolos) se pueden usar (**léxico**),
- qué frases **gramaticalmente correctas** se pueden formar con las palabras (**sintaxis**),
- qué significan las frases válidas en el lenguaje (**semántica**).

A la labor de desarrollar programas se le llama **programación**, y a las personas que realizan esta labor se les conoce como **programadores**.

Ahora bien: sabemos que los ordenadores operan con el **sistema binario**. Desde el punto de vista de los lenguajes de programación, ¿qué quiere decir esto?

Que los ordenadores sólo entienden de secuencias formadas por 0s y 1s, es decir, sólo entienden **el lenguaje máquina**: una descripción **binaria** del juego de instrucciones básicas del procesador.

Conclusión obvia: la estructura de los lenguajes máquina está **muy adaptada a los circuitos de los ordenadores**. **Diferentes tipos de ordenadores** (arquitecturas) presentan **lenguajes máquina distintos**.

Ejemplo de instrucción: `00011010 00000110 00001011`

Ventaja:
- Los programas escritos en lenguaje máquina son **directamente interpretables** por la CPU: son muy eficientes (usan la memoria justa y se ejecutan muy rápidamente).

Desventajas:
- Escribir programas en lenguaje máquina es una tarea de titanes (muy engorroso y lento), amén de escasamente productivo.
- Es difícil encontrar errores y corregir fallos.
- Los programas son muy difíciles de entender e interpretar por los programadores.
- La dependencia del lenguaje máquina respecto a la CPU hace que los programas en lenguaje máquina sólo puedan ejecutarse en el **procesador para el cual fueron escritos**, con lo que los programas **no** son **portables** de un ordenador a otro.

Conclusión: **nadie** escribe programas **directamente** en lenguaje máquina pero **todos los programas necesitan ser traducidos a él** ya que es lo único que entiende el (procesador del) ordenador.

Ciertas tareas críticas de programación (p.e. partes de un S.O.) requieren **gran eficiencia** aún **a costa de** la **productividad** y la **portabilidad**.

¿Qué puede hacer el programador en este caso?. . . Usar un **lenguaje ensamblador**

Los **lenguajes ensambladores** surgieron para **facilitar** las labores de programación, ya que emplean **etiquetas** (palabras cortas o abreviaturas en inglés) y utilizan símbolos que usamos los humanos (cifras) en lugar de engorrosas secuencias de bits. Se les considera los primeros **lenguajes simbólicos**.

A tener en cuenta: **son muy dependientes del lenguaje máquina para el cual se desarrollan**.

Ejemplo de instrucción: `ADD 6,11`

Normalmente, cada sentencia en lenguaje ensamblador se corresponde con una sentencia en lenguaje máquina, aunque el ensamblador presenta ciertas ventajas sobre este último:

- Es más **fácil** escribir programas (aunque sigue siendo complicado), manteniendo la **eficiencia** del código máquina.
- Los programas se pueden depurar mejor ya que son **algo más comprensibles**.

La desventaja más importante es que **la dependencia total** del lenguaje ensamblador con **respecto al ordenador** sigue **impidiendo la portabilidad** de los programas.

El lenguaje ensamblador **debe traducirse a lenguaje máquina** para poder ser ejecutado por el ordenador, pero **la traducción es directa y sencilla**.

Programar en ensamblador sigue siendo difícil. Y lo peor: los programas **siguen sin ser portables** ¿Qué podemos hacer?

Usar **lenguajes de alto nivel**. Éstos **apenas dependen** del ordenador y son **estándar**: hay organismos e instituciones que describen sin ambigüedad el lenguaje y velan por su **normalización** con el objetivo de lograr su independencia de las máquinas.

Los **lenguajes de alto nivel** son **simbólicos**, están **formalmente descritos** (de un modo matemático), tienen una **gran potencia expresiva** y son **bastante cercanos al lenguaje humano** (concretamente al lenguaje matemático).

Ejemplo de instrucción: `a = 6 + 11`

Ventajas:

- Los programas son **portables** debido a la **independencia** del lenguaje de alto nivel respecto a las máquinas, lo que significa que los programas se pueden ejecutar con poca o ninguna modificación en distintos tipos de ordenadores.
- Es más **fácil escribir** programas, se incrementa la **productividad** y se **reducen costes**.
- Se dispone de un conjunto de instrucciones **mucho más potente**: existen **instrucciones complejas ya predefinidas**.
- Los programas son más **fáciles de entender**, lo cual facilita tanto su **depuración** como su **modificación**.

Desventajas respecto a lenguajes máquina y ensambladores:

- Como el lenguaje es prácticamente independiente de la máquina, se **aprovechan menos** sus **recursos internos**
- **Aumenta** el **consumo de memoria** de los programas.
- El **tiempo de ejecución** de los programas **es mayor**.

Las ventajas, fundamentalmente **potencia**, **productividad** y **legibilidad** **compensan sobradamente** las desventajas, ya que los recursos de memoria, espacio en disco y tiempo de CPU son cada vez **más baratos**.

Hay centenares de lenguajes de programación disponibles.

- Algunos son más adecuados que otros para según qué tipo de problemas.
  - Sistemas: C, C++.
  - Administración de sistemas: Perl, Python.
  - Web: PHP, Perl, Python.
  - Inteligencia Artificial: Lisp.
  - Propósito general: C, C++, Pascal, Python.
- Debes ser consciente que hay lenguajes más "potentes" que otros, más "portables" que otros, más "eficientes" que otros, . . .
- Algunos permiten expresar "más con menos", pero a costa de la velocidad o el consumo de memoria.
  - C, C++ o Pascal son lenguajes cuyos programas se ejecutan rápidamente.
  - Python o Perl son más lentos, pero sus programas suelen ser entre 3 y 10 veces más cortos.

Nosotros emplearemos Python, ya que:

- es muy potente y expresivo,
- dispone de una gran cantidad de funciones predefinidas y
- permite desarrollar programas de cierta envergadura de manera sencilla y eficaz (su curva de aprendizaje es pequeña).

Veamos un ejemplo de programa (escrito en C).

```
#include <stdio.h>

void main(char *argv[], int argc)
{
  printf("Saludos a todos.\n");
}
```


Pero, ¿y la **traducción** al **lenguaje máquina** (secuencias de bits)?

A la vista del programa presentado en la transparencia anterior esto no parece una tarea ni mucho menos tan sencilla como la traducción del lenguaje ensamblador.

En este caso, se emplean programas "especiales", **específicos para cada lenguaje**, que realizan la traducción de cualquier programa a lenguaje máquina.

Estos programas se denominan **compiladores** o bien **intérpretes**. Para que un programa pueda funcionar en un ordenador **debe ser compilado** previamente.

¿Cuál es la diferencia entre **compilador** e **intérprete**?

Los lenguajes que usan un compilador (**compilados**) requieren que los programas se escriban enteros y se almacenen en un fichero. El compilador **necesita leer todo el programa** para poder efectuar la traducción. Ejemplos: C, Pascal. Sus programas suelen ser más eficientes: se ejecutan más rápidamente, usan menos memoria y dan más control al programador.

Los lenguajes que usan un intérprete (**interpretados**) no requieren la escritura completa de los programas, aunque sí resulta conveniente almacenarlos en un fichero (para no perderlos :-). El intérprete **traduce a lenguaje máquina** instrucción por instrucción. Ejemplos: Python, Perl. Sus programas suelen ser más fáciles de escribir y depurar, pero consumen más recursos (memoria y tiempo de ejecución).

Normalmente, suele ser más fácil aprender a programar con un lenguaje interpretado.

Ficheros. Organización jerárquica de la información

Los datos que maneja el ordenador pueden residir en la memoria principal o en periféricos de almacenamiento masivo, como los discos duros, CD-ROM, etc.

En el segundo caso, la información se presenta en forma de **ficheros**.

Un **fichero** es una colección de informaciones relacionadas entre sí y definidas por su creador.

Los ficheros permiten:

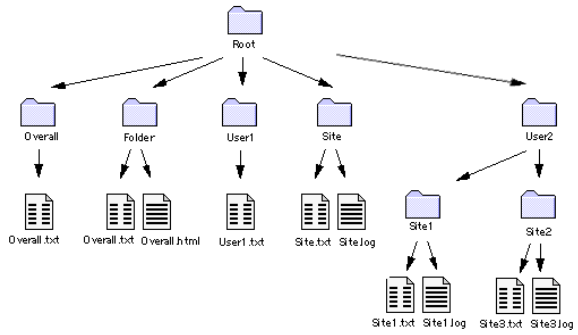
- **agrupar** una serie de **datos** en una unidad lógica,
- **abstraer** de los **detalles de representación** y soporte físico: p.e., en los discos duros la información se registra y lee con técnicas electromagnéticas, en los CD-ROM se lee con técnicas ópticas, pero eso a nuestros ficheros "les da igual".

Tipos de fichero:

- **Ejecutables**: sus datos son secuencias de instrucciones (programas) que el ordenador puede ejecutar.
- **De datos**: información de cualquier tipo (texto, imagen, sonido. . .).

En un disco duro puede haber decenas de miles de ficheros.

Los ficheros se organizan en una **jerarquía** o **árbol de directorios**.



Sistemas Operativos

El **Sistema Operativo** (S.O.) es un **programa** que se encarga de:

- **Controlar y administrar los recursos** del ordenador (memoria, disco duro, teclado, ratón, etc.). Distribuye los recursos del sistema (espacio en disco, memoria, tiempo de CPU) entre los usuarios y los programas, protegiendo a unos de otros.
- **Ofrecer una visión abstracta y homogénea** independiente de la arquitectura, facilitando el manejo de los dispositivos y escondiendo la complejidad del hardware. Esto es, ofrece una **máquina virtual** con el objetivo de facilitar la utilización del ordenador por parte del usuario.

  Por ejemplo, el sistema operativo Unix puede ejecutarse en PC, Macintosh, Sparc, etc., arquitecturas muy distintas entre sí. Pero un usuario o programador de Unix puede controlar esos ordenadores del mismo modo ya que la **máquina virtual** (interfaz) que ofrece el sistema es la misma en todos los casos.

Además, el S.O. también puede verse como un **programa de control**, en el sentido en que planifica la ejecución de los procesos, **detectando** y **recuperando** en la medida de lo posible los errores que pudiesen producirse.

Desde este punto de vista se pretende evitar el uso inadecuado de la máquina. El S.O. tiene la potestad de finalizar la ejecución de aquellos procesos que realizan operaciones "incorrectas".

De esta forma, se evita que, por ejemplo, un programa produzca resultados anómalos debido a que otro ha modificado sus datos, accediendo inapropiadamente a la memoria que aquél ocupa.

En conclusión, el propósito del S.O. es proveer un ambiente en el cual

- el usuario pueda ejecutar programas y tener acceso a (manejar) los dispositivos de manera adecuada, cómoda y convenientemente;
- se utilice el hardware de una manera eficiente, permitiendo la compartición de recursos por los usuarios de forma (casi)óptima.

Cuando un ordenador se enciende, **carga** el programa Sistema Operativo en su **memoria principal** (RAM) y éste se ejecuta de inmediato.

El S.O. toma control del hardware y ofrece al usuario uno o más **entornos de trabajo** (o **entornos de usuario**): conjuntos de utilidades que, mediante una interfaz adecuada, permiten al usuario ejecutar programas, controlar procesos, gestionar sus ficheros, etc. Un ejemplo de entorno de trabajo: el escritorio de Windows.

Hay entornos de trabajo

- **orientados a línea de órdenes**: el usuario escribe órdenes en una consola que el ordenador ejecuta;
- **gráficos**: el usuario acciona sobre elementos gráficos con uno o más dispositivos especiales (p.e. ratón) para ejecutar las acciones.

Partes fundamentales que componen un S.O.:

Gestión de procesos: Ejecución de programas y representación interna de los mismos en forma de procesos. Planificación, envío de señales y finalización de procesos.

Entrada/salida: Sincronizar entrada y salida de datos, atrapar interrupciones y ofrecer utilidades para que los programadores puedan capturar datos de entrada y mostrar resultados en sus programas.

Gestión de la memoria: Asignar y liberar memoria y llevar un control fidedigno del estado de la memoria principal en todo momento.

Sistema de ficheros: Representación lógica del sistema físico de almacenamiento secundario (discos duros, disquetes, etc.).

Codificación de la información

- Sistema **analógico**: los datos están representados mediante una magnitud física que varía de forma **continua**. Ejemplo: barómetro.
- Sistema **digital**: los datos están representados mediante una magnitud física que varía de forma *discreta* (sólo hay una serie de estados posibles). Ejemplo: medidor de carga en un condensador (cargado o no).

Un sistema digital en el que los **valores posibles** son **sólo dos** se llama **binario**. En la actualidad, **los ordenadores son digitales binarios** (en el pasado existieron ordenadores de tipo mecánico o electromecánico).

Ahora bien, el ordenador es capaz de capturar sonido y/o vídeo, magnitudes continuas. ¿Cómo podemos explicar esto?

Si es preciso **manipular datos analógicos** se utilizan dispositivos periféricos adicionales que se comunican con el ordenador mediante **conversores analógico-digitales** (p.e. una tarjeta de sonido es capaz de digitalizarlo).

En un ordenador digital binario, los componentes más básicos son dispositivos electrónicos con **dos estados estables** (representados mediante 0 y 1).

Se hace necesario **codificar la información** en el **sistema binario** para que un ordenador pueda tratarla ya que todas las informaciones se representan mediante **códigos binarios**. ¿Qué podemos codificar? **Prácticamente todo**. En el Tema 3 trataremos este asunto en mayor profundidad.

Una **codificación** es una **transformación** de los elementos de un conjunto en los de otro, de tal manera que a cada elemento del primer conjunto le corresponde un elemento distinto del segundo.

Cualquier información que deseemos tratar automáticamente en un ordenador, representada mediante alguno de los códigos de comunicación habituales entre las personas (lenguaje, imágenes, sonidos,. . .), **deberá tener asociado su código binario** que el ordenador debe ser capaz de interpretar.

En los dispositivos de entrada y salida del ordenador se efectúan automáticamente los cambios oportunos para que en el exterior la información sea directamente comprendida por los usuarios (ya que el ser humano no entiende de "códigos binarios").

Redes

Consideramos como una **red de ordenadores** a cualquier conjunto de computadores que pueden **comunicarse entre sí**.

Una red puede abarcar únicamente **dos ordenadores** o **varios millones**. Más aún, **distintas redes pueden estar comunicadas** entre sí.

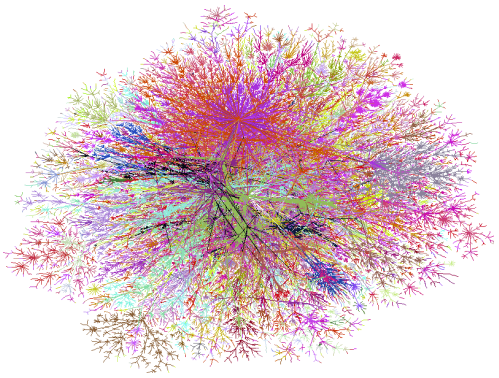
Esto forma las "interredes" o **internets**, la más famosa de las cuales es **la red de redes mundial** conocida simplemente como **Internet**.

Ventajas:

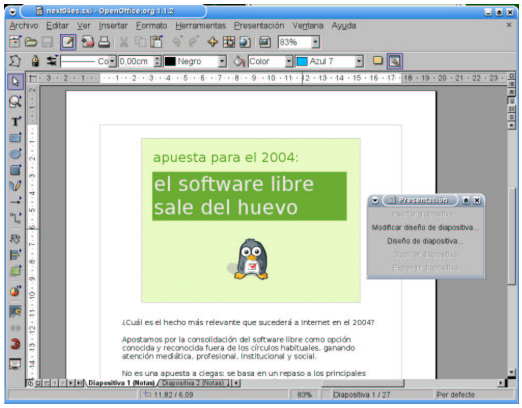
- Para las empresas:
  - Permiten **compartir recursos**. P.e. acceso a documentación, programas o máquinas remotas.
  - Aumento de la **fiabilidad**. P.e.: si un mismo dato se guarda en varias máquinas, cuando una falle se puede emplear otra.
  - **Ahorro**. La relación precio/rendimiento es mucho mejor en los ordenadores pequeños que en los grandes. Así, es preferible que cada usuario tenga su propio ordenador a que tenga una terminal de un ordenador grande. Esto ha extendido además el modelo cliente/servidor.
  - Proporcionan **escalabilidad**. Si se quiere más potencia se añaden nuevos ordenadores a la red.
  - La red supone un **medio de comunicación** que permite trabajar juntos a empleados separados geográficamente.

- Para las personas:
  - Permiten **acceso a información remota**. Especialmente con el crecimiento del servicio WWW.
  - Permiten diversas maneras de **comunicación interpersonal**. P.e. correo-e, mensajería instantánea, chat o vídeoconferencia.
  - Permiten nuevas maneras de **ocio**. P.e. juegos en red.

Internet es una red formada **interconectando infinidad de redes** de computadores entre sí.



Internet en 1998.



Impress de OpenOffice.org.

Aplicaciones de interés para el Diseño Industrial

Una **aplicación** es un **programa** que facilita al usuario **herramientas para llevar a cabo una tarea**. Podemos clasificar las aplicaciones según el tipo de tarea que ayudan a realizar. Así podemos encontrar:

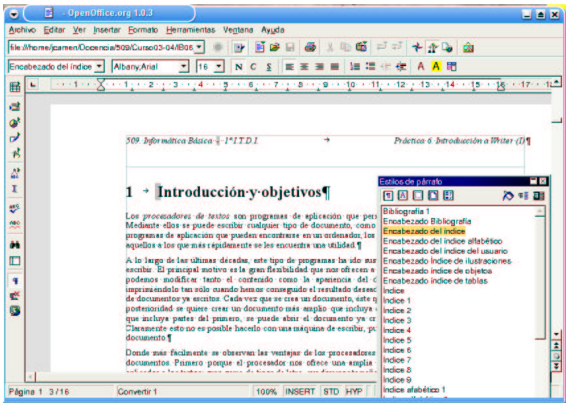
- **Procesadores de texto**: edición de libros, revistas, páginas web. . .

  Son programas que permiten dar formato (negrita, itálica, tamaño del tipo de letra, etc.) al texto, combinarlos con imágenes, etc. Muchos procesadores de texto permiten ver el resultado en pantalla al mismo tiempo que se edita el contenido.

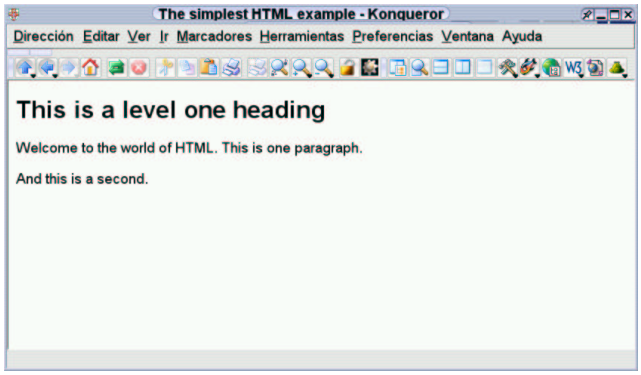
- **Navegación web**: acceso a información remota almacenada en páginas web.

  En realidad los navegadores son **intérpretes del lenguaje HTML**, es decir, permiten **visualizar** ficheros escritos usando este **lenguaje de etiquetas** (que no de programación) que es el que se utiliza en la elaboración de páginas web.

  También permiten acceder a ficheros de vídeo, audio, imágenes, PDF, etc. usando los **plugins** (herramientas específicas para el tratamiento adecuado de un determinado tipo de ficheros) adecuados. Algunos, como Internet Explorer se "integran" en entornos gráficos de trabajo y permiten acceder al contenido de ficheros y carpetas del sistema.



Writer de OpenOffice.org.



Un navegador web (Konqueror).

- **Presentación**: edición de transparencias, animaciones. . .

  Permiten editar una secuencia de "pantallas" que ayudan a transmitir una idea, presentar un producto, etc. Suelen incluir herramientas que permiten realizar "efectos especiales" y acceso a programas multimedia.

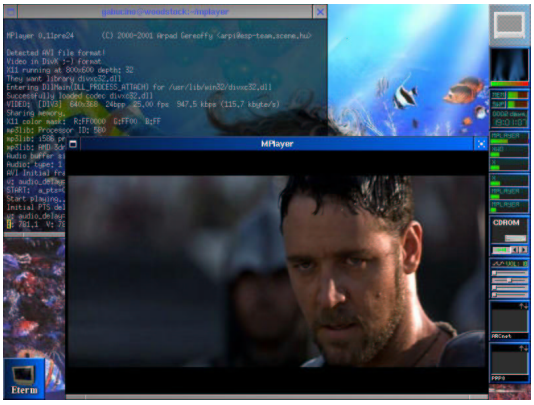
El navegador de la figura anterior está "interpretando" el contenido de este fichero:

```
<HTML>
 <HEAD>
  <TITLE>The simplest HTML example</TITLE>
 </HEAD>
 <BODY>
  <H1>This is a level one heading</H1>

  <P>Welcome to the world of HTML. This is one paragraph.</P>

  <P>And this is a second.</P>
 </BODY>
</HTML>
```

Aunque el contenido es texto, sigue una normas precisas que permiten al navegador mostrarlo con un formato visual atractivo.

- **Multimedia**: escuchar/grabar música, reproducir vídeos y películas, grabar imágenes, etc.

  Se denominan programas multimedia a aquéllos que permiten visualizar, editar y/o combinar datos diferentes entre sí: imágenes con sonido, imágenes con otras imágenes, captura y edición de vídeo, animaciones y efectos especiales, conversión entre formatos (WAV a MP3, MPEG a DIVX,. . .), etc.



Viendo a Russell Crowe en "Gladiator" con mplayer (en Linux).

Todo ingenierio competente debería **dominar** el uso de:

(a) un **procesador de texto**,

(b) un **programa para presentaciones** (optativa "Presentación de Diseños Asistida por Ordenador") y

(c) ciertos **servicios de Internet** (navegación, correo-e y transferencia de ficheros, que veremos mediante Mozilla Firefox y Filezilla).

En el caso de Diseño Industrial, además resulta necesario conocer alguna de las muchas posibilidades que nos ofrece el "universo multimedia" (Tema 3).

Recientemente, el **diseño de páginas web** se está convirtiendo en una posible salida profesional para el ingeniero técnico en diseño.

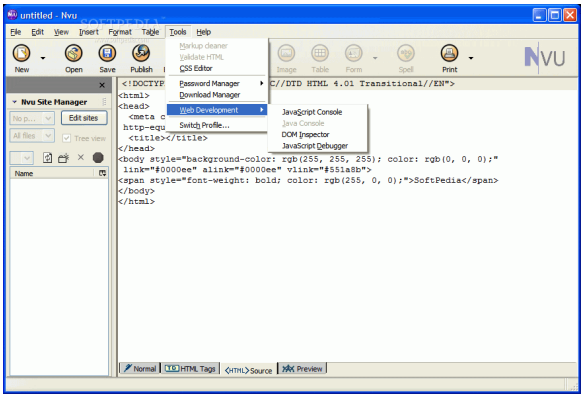
Sois muchos los que mostráis interés por conocer y aprender los fundamentos del diseño de páginas web.

Ciertamente, se trata de un campo muy interesante tanto desde el punto de vista profesional como desde el punto de vista del aprendizaje práctico.

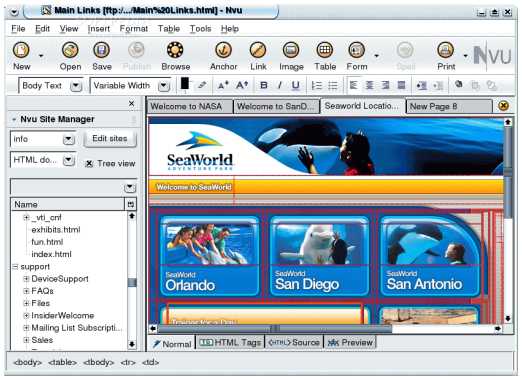
Vamos a introducir estos fundamentos mediante el uso de **NVU** en prácticas.

NVU es un editor WYSIWYG (what-you-see-is-what-you-get) de páginas web que funciona de una manera similar a la de un procesador de textos. Pero es más: es un sistema de diseño web para usuarios "de escritorio" similar a **FrontPage** y **DreamWeaver**. Es multiplataforma (Linux y Windows).

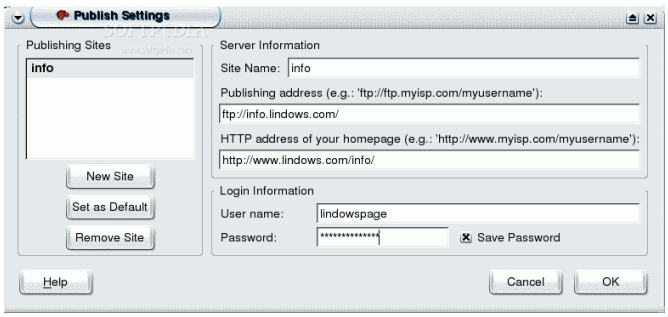
Podemos editar directamente el código HTML (usuario experto):



Pero también podemos usar las facilidades de edición y ver cómo va quedando nuestra web (usuario no experto):



Incluso podemos publicar nuestra página desde NVU si lo configuramos adecuadamente:



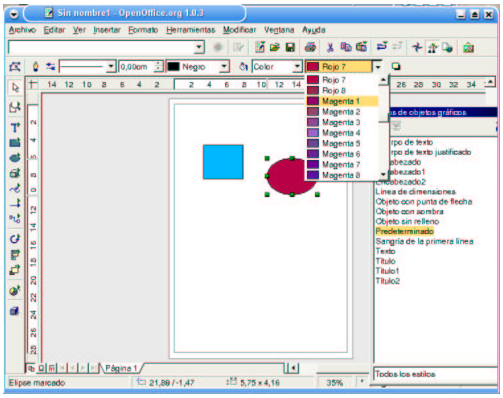
Pero además existen una serie de herramientas de gran interés en vuestra profesión y que seguramente os resultarán desconocidas.

En diversas asignaturas de la carrera tendréis la posibilidad de usar muchos de estos programas. Para que os vayáis familiarizando, vamos a presentar algunos junto con las tareas que ayudan a realizar.

- **Dibujo vectorial**: sirven para **dibujar objetos** almacenando la **información necesaria para construirlos** (p.e., de una circunferencia se puede almacenar su radio, las coordenadas del centro y el grosor y el trazo de la línea que la dibuja).

  Permiten **mayor flexibilidad en el dibujo**; se manipulan atributos de los objetos permitiendo escalados, giros, estrujamientos, modificaciones en el relleno, etc. Están orientados al **dibujo de precisión**: diseño de planos, diseño gráfico, . . . Por sus características **es fácil modificar objetos y crear otros más complejos por composición de objetos simples**. Los objetos suelen tener gran variedad de atributos susceptibles de ser modificados: contorno, relleno, tipo de trazo, etc.

  Programas: Sw. privativo (hay que adquirir licencia): **CorelDraw**, **FreeHand**, **Adobe Illustrator**. Sw. libre (gratuitos, la licencia permite hacer copias): **Xfig**, **Draw de OpenOffice.org**, **sodipodi**.

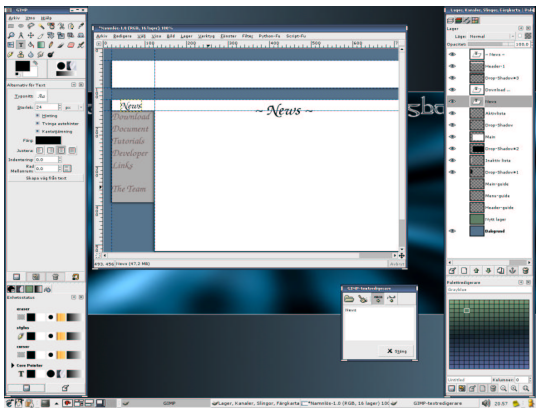


Cambiando el color de relleno del objeto elipse mediante Draw de OpenOffice.org.

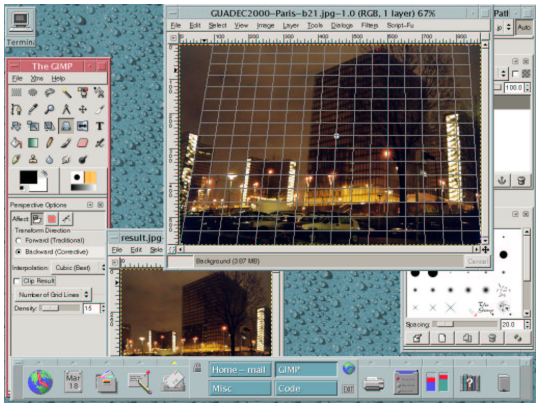
- **Retoque fotográfico** y **edición de imágenes**: permiten modificar y editar una imagen y/o fotografía o partes de la misma. Poseen varias funcionalidades para resaltar, ampliar, etc. una imagen. Se utilizan para fotografía digital, diseño, publicidad, etc. Para el diseño de páginas web es muy importante saber manejar una herramienta de estas características.

  Programas: **Photoshop** (hay que pagar por la licencia y es ilegal hacerse copias) y **The Gimp** (**software libre** y **multiplataforma** –tiene versiones para Windows y Linux). Con **The Gimp** no has de pagar nada y es perfectamente legal hacerse copias: `http://www.gimp.org`. Además, es la herramienta que vamos a usar en las prácticas.

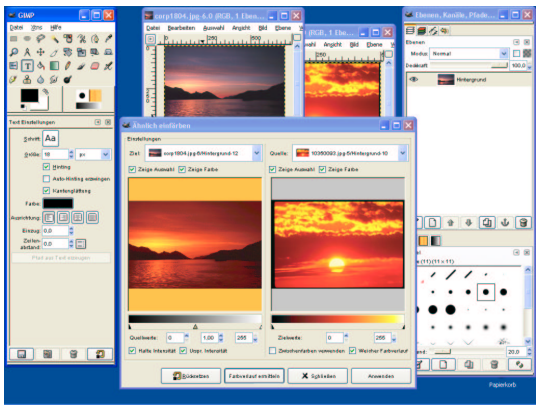
Retocando una foto con The Gimp.



Creando opciones de menú en una imagen para página web con The Gimp.



Transformando una imagen (perspectiva y escala) con The Gimp.



Realzando la foto de un paisaje con The Gimp.

- **CAD/CAM** (Diseño Asistido por Ordenador/Fabricación Asistida por Ordenador): son programas de **dibujo vectorial** (en el fondo) aunque con **capacidades muy avanzadas**: i) permiten la manipulación de objetos en tres dimensiones; ii) se pueden utilizar para generar distintos tipos de proyecciones (planos) y perspectivas de un objeto; iii) se pueden integrar con programas de fabricación automática (CAM). Se utilizan mucho en ingeniería y en diseño y fabricación.

  Son **muy precisos**, ya que suelen estar enfocados a la fabricación y suelen existir programas (o módulos) **específicos** para **diseño electrónico**, **arquitectura**, **diseño mecánico**, etc. Incluyen herramientas de **modelado**, **animación**, **análisis de elementos finitos** (resistencia de materiales, cálculo de esfuerzos y puntos de ruptura,etc.) y **cálculo de estructuras** (a nivel elemental).

  Programas: **Autocad**, **MicroStation** (requieren adquirir licencia). Sw. libre: **Qcad** limitado a 2D: `http://www.ribbonsoft.com/qcad.html`

Trabajando con MicroStation.

- **Modelado geométrico, animaciones, síntesis de imagen (*rendering*)**: Son programas similares, en cierto modo, a los de CAD, pero **menos precisos** aunque ofrecen **mayor calidad visual**.

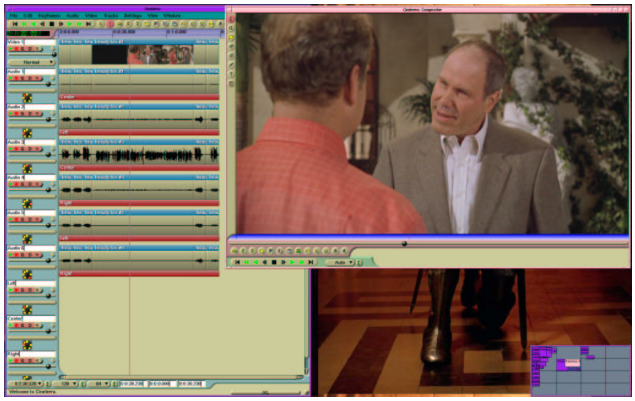
  Se usan en aquellas áreas del diseño donde no importa tanto la precisión como la calidad del objeto diseñado (textura, calidad imagen, sombreado,. . .): presentaciones, publicidad, diseño artístico, etc. Suelen incluir herramientas de modelado, animaciones y síntesis de imagen, pero algunos programas no las incluyen todas.

  Programas: requieren licencia: **3D Studio Max**, **Maya** (ambos con mejor modelado), **Lightwave**. Sw. libre: **Blender** y **POV-Ray** (éste es sólo para rendering –síntesis de imagen– e incluye un potente lenguaje para descripción de escenas).

Un personaje creado con Blender.

- **Edición de vídeo**: Son programas para la creación de contenidos avanzados (vídeo, películas). Suelen hacer fundamentalmente **tres cosas**: **capturar** (desde una cámara digital conectada al ordenador, por ejemplo), **componer** y **editar** audio y vídeo (añadir/suprimir escenas, pistas de audio, etc.). Algunos incluyen herramientas para hacer ciertos "efectos especiales".

  Programas: **Premiere** de Adobe (se requiere licencia); sw. libre: **Cinelerra** (Linux), **Kino** (Linux), **Virtualdub** (Windows), **Jahshaka** (multiplataforma).



Mezclando 6 canales de audio en una peli con Cinelerra.

- **Planificación de proyectos**: son programas que permiten **gestionar un proyecto**: **establecer objetivos**, **asignar recursos**, **calcular riesgos**, **planificar estrategias**, **asignar tareas**, ... Los programas más potentes permiten incluso **administrar grupos de trabajo** e incluyen herramientas de comunicación entre los mismos.

  Se pueden comunicar con otros programas (como hojas de cálculo) y son capaces de producir gráficas y diagramas de varios tipos: calendario del proyecto, diagramas de Gantt, etc.

  Programas: **Project Scheduler**, **Microsoft Project** (ambos requieren licencia).