Herramientas gratuitas para el trabajo científico

Satu Elisa Schaeffer†, Vanesa Avalos Gaitán
Saúl Isaí Caballero Hernández, Yajaira Cardona Valdés
Ramón García Alejo, Gabriela García Ayala,
Irma Delia García Calvillo, Sergio Madrigal Espinoza,
Miguel Mata Pérez, Dexmont Alejandro Peña Carrillo,
Gabriela Chamorro Sotelo y José Juan García Moreno

2007 (actualizaciones menores: 8 de septiembre de 2015)

Prefacio

Este documento contiene los materiales de enseñanza de un taller de verano organizado en el 2007 en Pisis de la Facultad de Ingeniería Mecánica e Eléctrica de la Universidad Autónoma de Nuevo León. El tema principal del taller fue la utilización de herramientas gratuitas de cómputo para las varias tareas de un estudiante, un tesista o un científico. La mayoría de las herramientas están disponibles para los sistemas operativos basados en Unix, Microsoft Windows o Mac OS. Ya que el documento ya está algo viejo, habrá enlaces que ya no funcionan igual como herramientas algo obsoletas, pero la mayoría de esta información sigue siendo vigente y útil (y consultado por bastante alumnos cada año.)

Los autores agradecen a los doctores J. Arturo Berrones Santos, Roger Z. Ríos Mercado, Deniz Özdemir e Igor Litvinchev de Pisis y el doctor Gregorio Toscano Pulido de Cinvestav por sus comentarios.

En el 2015, algunas de las secciones más obsoletas fueron eliminadas y unas notas breves agregadas sobre temas contemporáneos.

† Autor de contacto, correo electrónico elisa.schaeffer@uanl.edu.mx

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Unix en breve

Como prerrequisito de las siguientes sesiones del taller, los participantes deben aprender el manejo básico de sistemas operativos tipo Unix por un terminal textual. En el Cuadro 1.1, se mencionan algunas de las instrucciones útiles.

1.1. Sistema de archivos y carpetas

En Unix, por lo general se puede nombrar los archivos con terminaciones arbitrarias — el nombre de archivo no implica necesariamente nada sobre el formato de sus contenidos. Es muy importante recordar que en Unix sí se diferencia entre mayúsculas y minúsculas.

La carpeta inicial en que se entra por defecto al abrir un terminal de instrucciones está ubicada en `/home/usuario`, donde `usuario` es el usuario de la persona quien ha hecho login en el sistema. En cualquier momento, solamente al ejecutar la instrucción `cd` regresaremos el terminal a esta carpeta inicial. Para ver los contenidos de una carpeta, se utiliza la instrucción `ls` y para movernos a otra carpeta se añade el nombre de la carpeta después de la instrucción `cd`. El siguiente ejemplo ilustra estas instrucciones, junto con la instrucción `cd ..` que nos regresa un nivel en nuestro sistema jerárquico de carpetas:

```
> ls
CURRICULUM.doc         info                  red_contactos.doc
Mail                   mail                  replace_accents_html
Makefile               nsmail                taller
OpenOffice.org1.1.4    other                 temp
PID295149.pdf          papers                tesis
SemillaBarros2006.doc  pedro                 todo
admin                  pics                  visitas
backup                 prosper.pdf           work
docs                   public_html
> cd visitas/
> ls
carta_udp_abril_2007.jpg  coasesor.jpg
carta_udp_abril_2007.pdf  invitacionelisa.doc
```

Cuadro 1.1: Instrucciones básicas de sistemas tipo Unix. Casi todas aceptan *parámetros* para modificar su comportamiento.

`cd` ⟨nombre de carpeta⟩	Cambiar carpeta
`ls` ⟨nombre de carpeta⟩	Ver contenidos de una carpeta
`less` ⟨nombre de archivo⟩	Ver contenidos de un archivo
`rm` ⟨nombre de archivo⟩	Remover un archivo
`cp` ⟨nombre de archivo⟩ ⟨nombre de archivo⟩	Copiar un archivo
`mv` ⟨nombre de archivo⟩ ⟨nombre de archivo⟩	Mover/renombrar un archivo
`mkdir` ⟨nombre de carpeta⟩	Crear una carpeta nueva
`rmdir` ⟨nombre de carpeta⟩	Remover una carpeta
⟨instrucción⟩ > ⟨nombre de archivo⟩	Enviar/redirigir la salida a un archivo
⟨instrucción⟩ < ⟨nombre de archivo⟩	Leer la entrada de un archivo
⟨instrucción⟩ | ⟨instrucción⟩	Enviar la salida como entrada
`..`	Carpeta anterior
`.`	Carpeta actual
`~`	Carpeta inicial

```
> cd ..
> ls
CURRICULUM.doc         info                 red_contactos.doc
Mail                   mail                 replace_accents_html
Makefile               nsmail               taller
OpenOffice.org1.1.4    other                temp
PID295149.pdf          papers               tesis
SemillaBarros2006.doc  pedro                todo
admin                  pics                 visitas
backup                 prosper.pdf          work
docs                   public_html
```

Para saber cuales son carpetas y cuales archivos, así como ver qué *permisiones de acceso* tenemos en los diferentes archivos y carpetas, podemos utilizar `ls -l`:

```
> cd
> cd work/
> ls -la
total 5852
drwx------  12 elisa    faculty     2048 Jun  4 15:05 .
drwx--x--x  45 elisa    faculty     3072 Jun  4 15:04 ..
-rw-r--r--   1 elisa    faculty   110434 Oct  6  2006 OfertaEconomica.gif
-rw-r--r--   1 elisa    faculty    62976 May 23 12:26 Paraevaluadorespracticum.doc
-rw-r--r--   1 elisa    faculty    29696 Apr 20 09:32 aceptacion-verano-07.doc
drwxr-xr-x   4 elisa    faculty      512 Jun  4 15:04 admin
drwxr-xr-x   3 elisa    faculty      512 May  9 13:08 clases
-rw-r--r--   1 elisa    faculty     6803 May 23 11:19 diapositivas_sergio.txt
-rw-r--r--   1 sergio   alumni      3791 Mar  7 11:55 gnuplot.txt
drwxr-xr-x   2 elisa    faculty      512 Jun  4 15:03 hiring
-rw-r-----   1 elisa    faculty  2142828 May 16 11:43 inst_elisa.tar.gz
-rw-r--r--   1 elisa    faculty    69315 May 23 12:26 invitacion_practicum_2007.jpg
-rw-r-----   1 elisa    faculty   121193 May 25 06:25 nips15.ps.gz
drwxr-xr-x   5 elisa    faculty     1024 Jun  4 15:05 opetus
drwxr-xr-x   2 elisa    faculty      512 Jun  4 15:04 perla
```

```
-rw-rw-r--   1 elisa    faculty     1615 May 28 10:33 pifi_libros1.txt
-rw-r--r--   1 elisa    faculty   322646 Apr 18 10:22 proyecto.pdf
drwxr-xr-x  11 elisa    faculty     1024 Jun  4 15:05 research
drwxr-xr-x   2 elisa    faculty      512 Jun  4 15:05 servsoc
-rw-r-----   1 elisa    faculty     2577 Dec  5  2006 solver.tar.gz
-rw-------   1 elisa    faculty    57127 Mar  7 13:03 taller.tex
drwxr-xr-x   2 elisa    faculty      512 Jun  4 15:03 tesistas
drwxr-xr-x   2 elisa    faculty      512 Jun  4 15:05 verano
drwxr-xr-x   2 elisa    faculty      512 Jun  4 15:05 visitas
>
```

Cada línea que comienza con la letra `d` es una carpeta. Las nueve letras que siguen son las permisiones: los primeros tres son las permisiones del mismo usuario `rwx` que significa que tiene permiso de leer (`r`), escribir (`w`) y ejecutar un archivo (`x`). El símbolo `-` implica que no hay permiso. Las tres siguientes letras son del *grupo de usuarios*. Por ejemplo, en `yalma.fime.uanl.mx`, `elisa` pertenece al grupo `facultad`, mientras `sergio` pertenece al grupo `alumni` y `ramon` al grupo `guest`. Las permisiones de grupo aplican a todos los usuarios que pertenecen al mismo grupo junto con el usuario quien es dueño del archivo. Las tres últimas letras representan las permisiones de *todos* los usuarios del sistema. El número de la segunda columna tiene que ver con el número de enlaces al archivo. Después viene el usuario y el grupo del dueño del archivo. Después viene el tamaño en bytes, la fecha, hora de la última modificación y el nombre del archivo o carpeta.

Esta vista oculta algunos archivos de sistema. Por definir la opción `-a` en `ls`, uno puede ver *todos* los archivos. Los archivos ocultos están nombrados así que el primer símbolo de su nombre es el punto `.`:

```
> cd
> ls -la
total 7354
drwx--x--x  45 elisa    faculty     3072 Jun  4 15:18 .
drwxr-xr-x  14 root     root         512 Jan 15 09:17 ..
-rw-------   1 elisa    faculty     6218 Apr 18 12:36 .ICEauthority
-rw-------   1 elisa    faculty      304 Jun  4 13:53 .TTauthority
drwx------   3 elisa    faculty      512 Mar 12 14:54 .Trash
-rw-------   1 elisa    faculty      605 Jun  4 13:53 .Xauthority
-rw-r--r--   1 elisa    faculty       40 Mar  9 12:50 .Xdefaults
drwxr-xr-x   2 elisa    faculty      512 Feb 22 13:58 .acrobat
-rw-r--r--   1 elisa    faculty      237 Jun  4 13:04 .acrosrch
-rw-r--r--   1 elisa    faculty    14018 Jun  4 12:13 .addressbook
-rw-------   1 elisa    faculty    19624 Jun  4 12:13 .addressbook.lu
drwx------   2 elisa    faculty      512 Mar 13 12:21 .adobe
-rw-r--r--   1 elisa    faculty      201 May 25 17:32 .bash_aliases
-rw-------   1 elisa    faculty     7136 Jun  4 13:56 .bash_history
-rw-r--r--   1 elisa    faculty      213 Mar 13 14:35 .bash_profile
-rw-r--r--   1 elisa    faculty      458 Mar 13 14:35 .bashrc
drwx------   3 elisa    faculty      512 Mar 12 15:58 .config
-rw-r--r--   1 elisa    faculty      767 Mar 14 12:44 .cshrc
drwxr-xr-x   2 elisa    faculty      512 Mar  9 12:50 .desktop
drwxr-xr-x  12 elisa    faculty      512 Jun  4 13:53 .dt
-rwxr-xr-x   1 elisa    faculty     5111 Oct  4  2006 .dtprofile
drwxr-xr-x   3 elisa    faculty      512 Nov  7  2006 .emacs.d
-rw-------   1 elisa    faculty       16 Oct  4  2006 .esd_auth
-rw-r--r--   1 elisa    faculty        0 Mar 26 13:28 .fonts.cache-csw-1
drwx------   4 elisa    faculty      512 Mar 13 14:51 .gaim
drwx------   4 elisa    faculty      512 Jun  4 13:54 .gconf
drwx------   2 elisa    faculty      512 Jun  4 13:57 .gconfd
drwxr-xr-x  21 elisa    faculty     1024 May 25 17:26 .gimp-2.2
```

```
drwx------   3 elisa    faculty      512 Mar 26 13:28 .gnome2
drwx------   2 elisa    faculty      512 Mar 26 13:28 .gnome2_private
drwxr-xr-x   2 elisa    faculty      512 Mar 13 14:22 .gstreamer-0.10
-rw-r--r--   1 elisa    faculty       92 Mar 13 14:22 .gtkrc-1.2-gnome2
drwxr-xr-x   2 elisa    faculty      512 Mar 13 11:16 .icons
drwxr-xr-x   3 elisa    faculty      512 Mar  9 12:48 .iiim
-rw-r--r--   1 elisa    faculty      682 Jun  1 11:05 .ispell_english
drwxr-xr-x   4 elisa    faculty      512 Mar 13 13:32 .java
drwxr-xr-x   3 elisa    faculty      512 Mar  9 12:50 .kde
-rw-r--r--   1 elisa    faculty      628 May 23 11:35 .log
-rw-r--r--   1 elisa    faculty      202 Mar  1 18:00 .login
-rw-r--r--   1 elisa    faculty     2424 Mar  9 12:50 .mailcap
drwx------   3 elisa    faculty      512 Oct  4  2006 .metacity
-rw-r--r--   1 elisa    faculty      635 Mar  9 12:50 .mime.types
drwx------   5 elisa    faculty      512 Mar 14 12:35 .mozilla
drwxr-xr-x   4 elisa    faculty      512 Mar  9 13:52 .nautilus
drwx------   5 elisa    faculty      512 Mar  2 13:58 .netscape
-rw-------   1 elisa    faculty    17425 Jun  1 08:34 .pinerc
-rw-r--r--   1 elisa    faculty      189 Aug 15  2006 .profile
-rw-------   1 elisa    faculty     1367 May 10 16:23 .recently-used
-rw-------   1 elisa    faculty        2 Feb 28 12:42 .sh_history
-rw-r--r--   1 elisa    faculty      166 Feb 22 14:24 .sig
drwx------   2 elisa    faculty      512 Jun  4 13:53 .solregis
drwx------   2 elisa    faculty      512 May 25 22:48 .ssh
-rw-r--r--   1 elisa    faculty       77 Mar  9 12:50 .sversionrc
drwx------   3 elisa    faculty      512 Mar 12 15:58 .thumbnails
-rw-r--r--   1 elisa    faculty       31 May 25 17:26 .xfigrc
-rw-r--r--   1 elisa    faculty    70110 Mar  9 13:52 .xftcache
-rw-r--r--   1 elisa    faculty    72704 May 28 09:53 CURRICULUM.doc
drwx------   2 elisa    faculty      512 Mar  2 13:56 Mail
-rw-r--r--   1 elisa    faculty      465 May 25 14:58 Makefile
drwxr-xr-x   5 elisa    faculty      512 May 10 16:21 OpenOffice.org1.1.4
-rw-r--r--   1 elisa    faculty   281765 Jan 30 12:48 PID295149.pdf
-rw-r--r--   1 elisa    faculty  3049984 Jan 24 14:11 SemillaBarros2006.doc
drwxr-xr-x   3 elisa    faculty      512 Mar 12 12:43 admin
drwx------   4 elisa    faculty     2048 May 17 17:47 backup
drwxr-xr-x   2 elisa    faculty      512 May  9 13:08 docs
drwx------   4 elisa    faculty     1024 Jun  4 14:57 info
drwx------   2 elisa    faculty     2560 Jun  4 14:56 mail
drwx------   2 elisa    faculty      512 Mar  2 13:57 nsmail
drwx------   5 elisa    faculty      512 Nov 28  2006 other
drwxr-xr-x   2 elisa    faculty     1536 May 28 11:04 papers
drwxr-xr-x   2 elisa    faculty      512 Jun  4 14:58 pedro
drwx------   2 elisa    faculty      512 May 28 10:39 pics
-rw-r--r--   1 elisa    faculty    56450 May 23 11:24 prosper.pdf
drwxr-xr-x  13 elisa    faculty      512 Jun  1 15:08 public_html
-rw-r--r--   1 elisa    faculty    48128 May 31 11:38 red_contactos.doc
drwxr-xr-x   2 elisa    faculty      512 May  9 13:08 taller
drwxr-xr-x   3 elisa    faculty      512 May 23 18:23 temp
drwxr-xr-x   2 elisa    faculty      512 May  9 13:07 tesis
drwx------   2 elisa    faculty     7168 Sep  6  2006 todo
drwx------  12 elisa    faculty     2048 Jun  4 15:05 work
>
```

La instrucción para cambiar las permisiones es `chmod` y se utiliza de la siguiente manera; primero se define primero se define a quien o a quienes le haremos las modificaciones, después se realizan los cambios deseados, y por último va el nombre del archivo o carpeta de que se trata:

```
> cd
> cd temp
> ls -la
total 166
```

```
drwxr-xr-x   3 elisa    faculty      512 Jun  4 15:21 .
drwx--x--x  45 elisa    faculty     3072 Jun  4 15:18 ..
-rw-r--r--   1 elisa    faculty      228 May 23 18:23 Makefile
-rw-r--r--   1 elisa    faculty       50 May 23 18:21 b.cpp
-rw-r--r--   1 elisa    faculty       10 May 23 18:20 b.h
-rw-r--r--   1 elisa    faculty     4536 Mar 14 10:20 better.pdf
-rw-r--r--   1 elisa    faculty    15743 Mar 14 10:18 better.ps
-rw-r--r--   1 elisa    faculty      276 Mar 13 12:16 doc.dvi
-rw-r--r--   1 elisa    faculty     4769 Mar 13 12:22 doc.pdf
-rw-r--r--   1 elisa    faculty    17142 Mar 13 12:21 doc.ps
-rw-r--r--   1 elisa    faculty       92 Mar 13 12:16 doc.tex
drwxr-xr-x   3 elisa    faculty      512 Mar 22 15:26 parcial
-rwxr-xr-x   1 elisa    faculty     9988 May 23 18:23 programa
-rw-r--r--   1 elisa    faculty     1480 Mar 14 10:09 prueba.dvi
-rw-r--r--   1 elisa    faculty    15743 Mar 14 10:11 prueba.ps
-rw-------   1 elisa    faculty      925 Mar 14 10:14 prueba.tex
-rw-r--r--   1 elisa    faculty      407 May 23 18:22 test.cpp
> chmod a-rw prueba.ps
> ls -la prueba.ps
----------   1 elisa    faculty    15743 Mar 14 10:11 prueba.ps
> chmod u+rw prueba.ps
> ls -la prueba.ps
-rw-------   1 elisa    faculty    15743 Mar 14 10:11 prueba.ps
> chmod g+r prueba.ps
> ls -la prueba.ps
-rw-r-----   1 elisa    faculty    15743 Mar 14 10:11 prueba.ps
> chmod a+rw parcial/
> ls -la parcial/
total 12
drwxrwxrwx   3 elisa    faculty      512 Mar 22 15:26 .
drwxr-xr-x   3 elisa    faculty      512 Jun  4 15:21 ..
-rw-r--r--   1 elisa    faculty     2548 Mar 22 15:26 parcial.tar.gz
drwxr-xr-x   2 elisa    faculty      512 Mar 22 15:26 temp
> ls -la
total 166
drwxr-xr-x   3 elisa    faculty      512 Jun  4 15:21 .
drwx--x--x  45 elisa    faculty     3072 Jun  4 15:18 ..
-rw-r--r--   1 elisa    faculty      228 May 23 18:23 Makefile
-rw-r--r--   1 elisa    faculty       50 May 23 18:21 b.cpp
-rw-r--r--   1 elisa    faculty       10 May 23 18:20 b.h
-rw-r--r--   1 elisa    faculty     4536 Mar 14 10:20 better.pdf
-rw-r--r--   1 elisa    faculty    15743 Mar 14 10:18 better.ps
-rw-r--r--   1 elisa    faculty      276 Mar 13 12:16 doc.dvi
-rw-r--r--   1 elisa    faculty     4769 Mar 13 12:22 doc.pdf
-rw-r--r--   1 elisa    faculty    17142 Mar 13 12:21 doc.ps
-rw-r--r--   1 elisa    faculty       92 Mar 13 12:16 doc.tex
drwxrwxrwx   3 elisa    faculty      512 Mar 22 15:26 parcial
-rwxr-xr-x   1 elisa    faculty     9988 May 23 18:23 programa
-rw-r--r--   1 elisa    faculty     1480 Mar 14 10:09 prueba.dvi
-rw-r-----   1 elisa    faculty    15743 Mar 14 10:11 prueba.ps
-rw-------   1 elisa    faculty      925 Mar 14 10:14 prueba.tex
-rw-r--r--   1 elisa    faculty      407 May 23 18:22 test.cpp
>
```

1.2. Expresiones regulares

1.2.1. Definición

Una expresión regular es una cadena de texto que describe un conjunto de cadenas de texto. La expresión regular también es llamada patrón o "pattern" en ingles.

Las expresiones regulares se utilizan para describir un conjunto de cadenas de texto sin tener que listar todas las cadenas, por ejemplo para listar las palabras casa y caza se puede escribir ca(s|z)a. A esto se le llama que el patrón mapea las palabras.

Una expresión regular consiste de constantes y operadores que denotan conjuntos de cadenas y operadores sobre estos conjuntos.

1.2.2. Operadores

| Alternar. Permite separar alternativas. En ca(s|z)a se alterna entre la s y la z

() Agrupación. Indica el alcance y precedencia de los operadores. En ca(s|z)a solamente se puede reemplazar el carácter en la tercera posición por s o z.

? Este carácter busca la ocurrencia 0 ó 1 veces de la ultima expresión. Por ejemplo go?l mapearía las palabras gl y gol.

* Este carácter busca la ocurrencia 0, 1 o cualquier número de veces que ocurra la ultima expresión. Por ejemplo go*l mapearía las palabras gl, gol, gool, goool, gooool, etc.

+ Busca la ocurrencia de al menos una vez la ultima expresión. Por ejemplo go+l mapearía las palabras gol, gool, goool, gooool, etc. Nótese que a diferencia de * gl no es mapeado.

1.2.3. Constantes

En una expresión regular un carácter se mapea solo a si mismo (una a, mapea solo la letra a, por ejemplo). Sin embargo existen meta-caracteres que tienen funciones especiales. El el cuadro 1.2 se muestra la sintaxis típica de expresiones regulares.

1.2.4. `grep`

`grep` toma una expresión que es el patrón de la línea de instrucciones, lee la entrada o una lista de archivos, e imprime solamente las líneas que contengan alguna coincidencia con el patrón definido. Con la opción `-c`, las ocurrencias mismas no están incluidas en la salida de `grep`, solamente el

Cuadro 1.2: Sintaxis de expresiones regulares.

`.`	Mapea cualquier carácter (uno solo). Si se coloca entre [], mapea el carácter ".". Por ejemplo a.cd, mapea abcd, accd, adcd, etc. Sin embargo [a.cd], mapea "a", ".", "c" o "d".
`[]`	Mapea cualquier carácter que se encuentre entre los paréntesis. Por ejemplo [abcd], mapea los caracteres "a", "b", "c" o "d".
`[^]`	Mapea cualquier carácter que no se encuentre entre los paréntesis.
`^`	Mapea el inicio de una línea.
`$`	Mapea el final de una línea.
`()`	Define una sub-expresión.
`[A-Z]`	Mapea todas las letras mayúsculas.
`[a-z]`	Mapea todas las letras minúsculas.
`[0-9]`	Mapea todos los números.

número total de veces que ocurre el patrón en la entrada al `grep`. Con la opción `-n` podemos añadir el número de la línea a cada ocurrencia. Con la opción `-v`, se busca por las líneas que *no* coinciden con el patrón, o sea, el complemento del resultado de la instrucción sin definir `-v`.

Por ejemplo, para buscar en el archivo `archivo.txt` por todas las líneas que contengan "mace", "nace" y "oace", sirve ejecutar `grep [m-o]ace archivo.txt`. Para buscar por las líneas que contengan la terminación "ace" y que no empiecen con las letras "m", "n" y "o", se ejecuta `grep [^m-o]ace archivo.txt`.

1.2.5. sed

`sed` es un "editor de flujo" que realiza muchas acciones sobre texto. `sed` se puede utilizar en diferentes sistemas operativos. `sed` ayuda a modificar el contenido de un fichero entre otras cosas. Sus opciones más comunes están en el cuadro 1.3.

(Otra opción bastante útil es `tr`.)

En `sed`, no se usa `-` para definir las opciones. El su sintaxis vienen las opciones adjuntadas al nombre del fichero: por ejemplo, para reemplazar las ocurrencias de la palabra "viejo" con la palabra "nuevo" en `archivo.dat` se ejecuta `sed 's/viejo/nuevo/g' archivo.dat`. Para omitir todas las líneas que contengan la palabra "Rey" o "rey", sirve `sed '/[Rr]ey/d' gente.txt`.

Cuadro 1.3: Algunas de las opciones más comunes de `sed`.

`s/.../`	reemplazar la primera ocurrencia de un patrón en cada línea con una cadena definida (por ejemplo, `s/algo/otro/` para sustituir "otro" donde dice "algo")
`g`	hace sustituciones generales de todos los patrones localizados (por ejemplo `s/algo/otro/g`)
`a`	añade una línea definida *después* cada línea afectada (por ejemplo `sed 'ahola'` va a añadir "hola" al fin de cada línea)
`i`	inserta una línea definida *antes* de cada línea afectada
`c`	cambia cada línea afectada por la línea definida
`d`	omitir cada línea afectada (por ejemplo `5,7d` es para omitir líneas desde la quinta hasta la séptima)
`$`	el símbolo de fin de archivo (por ejemplo en `sed '5,$d' datos.dat` quitaría las líneas desde la quinta hasta el fin del archivo)

1.3. Otras herramientas pequeñas

1.3.1. `cat`

`cat` envía el contenido del fichero a la salida por defecto (o sea, la pantalla si no defines una redirección con > por ejemplo). Enviando varios archivos se puede *concatenar* uno o varios ficheros: solamente hay que redirigir la salida a un archivo que no sea ninguno de los archivos de entrada. En el cuadro 1.4 se muestra las opciones más comunes de `cat`. Un ejemplo del sintaxis es la siguiente instrucción que junta los contenidos de todos los archivos los nombres de cuales comienzan con `datos` y tienen terminación `.dat` a un sólo archivo que se llama `todos_los_datos.dat`, juntando todas las líneas en blanco sucesivas en una sola línea en blanco y asignando un número a cada línea no en blanco:

```
cat -bs datos*.dat > todos_los_datos.dat
```

Cuadro 1.4: Algunas de las opciones más comunes de `cat`.

`-n`	Numera todas las líneas.
`-b`	Numera las líneas que no están en blanco.
`-s`	Junta en una sola línea las líneas en blanco sucesivas.
`-v`	Muestra los caracteres de control como si fueran visibles.
`-e`	Muestra los caracteres de fin de línea `$`.
`-t`	Muestra los tabuladores como `Î`.

1.3.2. `sort`

La instrucción `sort` sirve para agrupar u ordenar un fichero o la salida de un programa. Dado las ordenes que uno ponga en la línea de instrucción es la forma en que se van acomodar los ficheros. El cuadro 1.5 muestra las opciones más comunes.

Cuadro 1.5: Algunas de las opciones más comunes de `sort`.

`-n`	ordenar en orden numérico de cadenas
`-r`	obtener el orden reverso
`-d`	orden de "diccionario" — solamente considerar símbolos alfabéticos y blancos
`-f`	no diferenciar entre mayúsculas y minúsculas
`-m`	combinar archivos que ya están ordenadas
`-o nombre.dat`	enviar la salida a archivo `nombre.dat`
`-k3`	definir la llave de ordenamiento en posición tres
`-t#`	utiliza el símbolo especificado (en este caso #) como separador en vez de blanco
`-g`	ordenar en orden numérico general

Su sintaxis es tal que primero se ponen las opciones y después el archivo o los archivos que contienen los datos. En el ejemplo siguiente, tomamos la salida de `ls -la` y sorteamos los archivos según su tamaño:

```
> ls -la */index.html | sort -n -k5
-rw-r--r--   1 elisa    faculty     4341 May 11 17:32 io/index.html
-rw-r--r--   1 elisa    faculty     4400 Apr 25 09:31 aa/index.html
-rw-r--r--   1 elisa    faculty     5801 May 17 14:44 verano/index.html
-rw-r--r--   1 elisa    faculty     6343 May 10 10:49 taller/index.html
-rw-r--r--   1 elisa    faculty     6518 Feb 15 10:31 prog/index.html
-rw-r--r--   1 elisa    faculty     9985 May 28 12:18 seminar/index.html
>
```

1.3.3. `cut`

`cut` es una herramienta de línea de instrucciones de Unix que se utiliza para sacar/cortar o copiar secciones las líneas de entrada (sea una redirección con `|` o un fichero). Sus opciones más comunes están en el cuadro 1.6. Por ejemplo, para cortar los nombres de los archivos de la salida siguiente de `ls -l p*.pdf`,

```
-rw-rw-rw- 1 elisa elisa 59958 2007-03-26 14:13 problemas_de_matching_y_flujos.pdf
-rw-rw-rw- 1 elisa elisa 35230 2007-03-26 14:13 problemas_de_optimizacion.pdf
-rw-rw-rw- 1 elisa elisa 48295 2007-03-30 11:18 programacion_entera.pdf
-rw-rw-rw- 1 elisa elisa 56356 2007-03-26 14:13 programacion_lineal.pdf
-rw-rw-rw- 1 elisa elisa 61993 2007-03-26 14:13 programas_duales.pdf
```

se puede utilizar por ejemplo `ls -l p*.pdf | cut -c49-82` para obtener la salida deseada

```
problemas_de_matching_y_flujos.pdf
problemas_de_optimizacion.pdf
programacion_entera.pdf
programacion_lineal.pdf
programas_duales.pdf
```

Cuadro 1.6: Algunas de las opciones más comunes de `cut`.

-b	elegir solamente los bytes definidos por el rango (por ejemplo `-b3-15`)
-c	elegir solamente los caracteres definidos por el rango (por ejemplo `-c4-26`)
-d	definir el delimitador del campo (por ejemplo `-d:`)
-f	elegir solamente los campos/columnas definidos, separado por el delimitador
-s	suprimir las líneas que contengan el carácter delimitador (solamente con `-f`)
–complement	solamente dejar pasar lo que no pertenece en el rango definido

1.3.4. `touch`

`touch` es una herramienta para cambiar fechas y horas de acceso o modificación de archivos. Se ejecuta con `touch archivo.dat` para el archivo del nombre `archivo.dat`. Si no existe el archivo nombrado, touch va a crear un archivo nuevo vacío — si no es deseable la creación en falta de existencia, con la opción `-c` se puede evitar la creación. Con la opción `-m`, solamente la fecha y hora de modificación está actualizada a la fecha actual, mientras con la opción `-a` se cambia solamente la fecha y hora de acceso; sin opciones cambian las dos fechas. También se puede cambiar la fecha y hora para ser lo misma de un otro archivo, digamos `modelo.txt`, se ejecuta `touch archivo.dat modelo.txt`. Para manualmente cambiar la fecha y hora, se ejecuta `touch` con la opción `-t` así que se determina una cadena en formato MMDDhhmm (mes, día, hora y minuto). Para realizar la operación con varios archivos, basta con definir la lista o la expresión regular: por ejemplo, `touch *.txt` cambia todos los archivos de tipo `.txt` para tener la fecha y hora actual como su fecha y hora de modificación y acceso.

1.3.5. `echo`

`echo` es una instrucción muy simple: repite en la salida por defecto que se da como argumento a `echo`: por ejemplo, `echo hola` va a dar la salida “hola”. Se puede redirigir la salida a un archivo normalmente con `>`: con `echo hola >hola.txt` se crea un archivo del nombre `hola.txt` con los contenidos `hola`. Con la opción `-n` se suprime el fin de línea. Si se quiere incluir caracteres especiales, hay que

desactivarlos con solo poner un \ justo antes del símbolo deseado: `echo \*hola\*`.

1.4. Shell

El sintaxis de algunas cosas, como por ejemplo ajustar variables ambientales como `PATH`, depende de cuál *shell* se está utilizando. Otras opciones son `bash`, `sh`, `ksh` y `jsh`.

Al iniciar `tcsh`, el sistema se adapta a la configuración guardada en el archivo `.cshrc` en la carpeta de inicio del usuario. Por ejemplo,

```
#ident  "@(#)local.cshrc        1.2     00/05/01 SMI"
umask 022
set path=(/bin /usr/bin /usr/ucb /etc .)
set path=(/usr/local/bin ${path})
set path=(/opt/Acrobat5/bin /usr/sfw/bin ${path})
set path=(/usr/local/teTeX/bin/sparc-sun-solaris2.9 ${path})

setenv LD_LIBRARY_PATH=/usr/lib:/usr/local/lib
setenv LD_PATH=/usr/local/lib

set prompt="%T %B%m%b\:%~%# "

if ( $?prompt ) then
        set history=32
endif

alias pine 'pine -i'
```

En el ejemplo, primero se establece la variable `path` que contiene la lista de carpeta en las cuales el sistema va a buscar por programas para ejecutar al entrar como una instrucción el nombre de un programa. Después se establece dos variables para el uso del cargador (inglés: loader) de programas escritos en C++. Después se ajusta el número de instrucciones ya ejecutadas que serán recordadas por el shell. Al final se establece un *alias* que reemplaza la instrucción `pine` dada por el usuario con `pine -i`.

Para que se pongan en efecto los cambios hechos en `.cshrc`, hay que ejecutar `source .cshrc`, abrir otro terminal o hacer logout y login de nuevo.

En general, si prefieren utilizar `bash`, hay dos opciones. Una es iniciar `bash` manualmente con la instrucción `bash`, y la otra es editar el archivo `.login` en la carpeta de inicio del usuario `/home/miusuario/` para contener una línea extra (la última del ejemplo).

```
# Copyright (c) 2001 by Sun Microsystems, Inc.
# All rights reserved.
#
# ident "@(#)local.login        1.7     01/06/23 SMI"
stty -istrip
# setenv TERM 'tset -Q -'
if ( -f /bin/bash ) exec /bin/bash --login
```

Con `bash`, las variables ambientales vienen del archivo `.bashrc` en la carpeta de inicio:

```
export PS1='\a\074\t\076${USER}@${HOSTNAME}:${PWD/$HOME/~}> '

export HISTCONTROL=ignoredups
shopt -s checkwinsize
[ -x /usr/bin/lesspipe ] && eval "$(lesspipe)"

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib:/usr/local/lib
export LD_PATH=$LD_PATH:/usr/local/lib
```

En el ejemplo, primero se ajusta como se ve el *prompt* [1] para que sea la hora actual y `usuario@yalma:` con la carpeta actual. Después hay ajustes a cosas como eliminar duplicados del historial y ajustar al tamaño de la ventana. Los alias se ha incluido de otro archivo que se llama `.bash_aliases` y contiene por ejemplo las siguientes instrucciones:

```
alias casa='ssh usuario@otro.servidor.mx'
alias pine='pine -i'
```

Las variables del cargador se ajusta con la instrucciónexport, a la cual corresponde en `tcsh` la de `setenv`.

1.5. Resolución de problemas

1.5.1. Páginas de ayuda: `man`

`man` es una instrucción "ayudante" en las terminales, ya que si no sabes para qué sirve cierta instrucción, `man` da un mini-manual de cómo utilizarla. El idioma de estas páginas depende de la configuración del sistema Unix. Para usar `man` se define como parámetro el nombre del programa o instrucción que quieras utilizar y te mostrará todo lo referente a este programa, por ejemplo:

```
> man mv
Reformatting page.  Please Wait... done

User Commands                                           mv(1)

NAME
     mv - move files

SYNOPSIS
     /usr/bin/mv [-fi] source target_file

     /usr/bin/mv [-fi] source...   target_dir

     /usr/xpg4/bin/mv [-fi] source target_file
```

[1] En bash, las posibilidades de ajustar el prompt son varios; se puede incluir también la fecha. Buscando con Google por "bash prompt" se encuentra fácilmente instrucciones detalladas en español y inglés.

```
     /usr/xpg4/bin/mv [-fi] source...  target_dir

DESCRIPTION
     In the first synopsis form, the mv utility  moves  the  file
     named  by the source operand to the destination specified by
     the target_file. source and target_file  may  not  have  the
     same  name. If target_file does not exist, mv creates a file
     named target_file. If target_file exists, its  contents  are
     overwritten.  This  first  synopsis form is assumed when the
     final operand does not name an existing directory.

     ... (cortado por razones de presentación) ...

     In order to preserve the source file characteristics,  users
     must  have  the  appropriate  file  access permissions. This
     includes being super-user or having the same owner id as the
     destination file.

OPTIONS
     The following options are supported:

     -f    mv will move the file(s) without prompting even if  it
           is  writing over an existing target. Note that this is
           the default if the standard input is not a terminal.

     -i    mv will prompt  for  confirmation  whenever  the  move
           would  overwrite  an  existing  target. An affirmative
           answer means that the move should proceed.  Any  other
           answer prevents mv from overwriting the target.

  /usr/bin/mv
     Specifying both the -f and the -i options is not  considered
     an error. The -f option will override the -i option.

  /usr/xpg4/bin/mv
     Specifying both the -f and the -i options is not  considered
     an  error.  The  last  option  specified  will determine the
     behavior of mv.

OPERANDS
     The following operands are supported:

     source
           A path name of a file or directory to be moved.

     target_file

SunOS 5.9            Last change: 7 Jun 2001                   2

User Commands                                             mv(1)

           A new path name for the file or directory being moved.

     target_dir
           A path name of an existing  directory  into  which  to
           move the input files.

USAGE
     See largefile(5) for the description of the behavior  of  mv
     when encountering files greater than or equal to 2 Gbyte ( 2
    **31 bytes).

ENVIRONMENT VARIABLES
     See environ(5) for descriptions of the following environment
```

```
     variables  that  affect  the  execution of mv: LANG, LC_ALL,
     LC_CTYPE, LC_MESSAGES, and NLSPATH.

EXIT STATUS
     The following exit values are returned:

     0     All input files were moved successfully.

     >0    An error occurred.

ATTRIBUTES
     See attributes(5) for descriptions of the  following  attri-
     butes:

  /usr/bin/mv
     ____________________________________________________________
    |       ATTRIBUTE TYPE        |       ATTRIBUTE VALUE       |
    |_____________________________|_____________________________|
    | Availability                | SUNWcsu                     |
    |_____________________________|_____________________________|
    | CSI                         | Enabled                     |
    |_____________________________|_____________________________|
    | Interface Stability         | Stable                      |
    |_____________________________|_____________________________|

  /usr/xpg4/bin/mv
     ____________________________________________________________
    |       ATTRIBUTE TYPE        |       ATTRIBUTE VALUE       |
    |_____________________________|_____________________________|
    | Availability                | SUNWxcu4                    |
    |_____________________________|_____________________________|
    | CSI                         | Enabled                     |
    |_____________________________|_____________________________|
    | Interface Stability         | Standard                    |
    |_____________________________|_____________________________|

SunOS 5.9           Last change: 7 Jun 2001                   3

User Commands                                            mv(1)

SEE ALSO
     cp(1), cpio(1), ln(1), rm(1), setfacl(1),  chmod(2),  attri-
     butes(5), environ(5), fsattr(5), largefile(5), standards(5)

NOTES
     A -- permits the user to mark explicitly the end of any com-
     mand  line  options, allowing mv to recognize filename argu-
     ments that begin with a -. As an aid to  BSD  migration,  mv
     will  accept  -  as a synonym for --. This migration aid may
     disappear in a future release.

SunOS 5.9           Last change: 7 Jun 2001                   4

>
```

Algunas páginas de `man` son muy largas. Para avanzar una página, se puede utilizar el blanco (spacebar) y para avanzar una línea, enter. El programa que se usa para mostrar páginas `man` internamente es `more` y para aprender su uso, basta con `man more`.

Si no existe en el sistema una página para alguna instrucción, `man` avisa el usuario:

```
> man gimp
```

```
No manual entry for gimp.
```

1.5.2. Listado y manejo de procesos: `ps` y `kill`

La instrucción `ps` ayuda a verificar qué programas e instrucciones tenemos en ejecución en este momento. Al escribirlo en la línea de instrucciones te dará una lista de programas que estás utilizando actualmente con una clave de acceso y un tiempo de uso que llevas utilizando este programa, por ejemplo:

```
> ps
PID     TTY     TIME   CMD
9658    pts/23  0:00  run-mozi
9664    pts/23  0:03  firefox-
14580   pts/23  0:01  xemacs-2
9666    pts/23  0:00  gconfid-2
9416    pts/23  0:00  tcsh
9642    pts/23  0:00  firefox
15228   pts/23  0:00  ps
>
```

donde `PID` es un número único de identificación del proceso y `CMD` identifica a qué programa/instrucción corresponde el proceso. Para ver *todos* los procesos de la computadora, se pone `ps -A` y para ver todos los procesos de *un cierto usuario* se pone `ps -u usuario`, por ejemplo:

```
> ps -u yajaira
   PID TTY      TIME CMD
 29725 ?        0:10 metacity
 29712 ?        0:00 esd
 29696 pts/16   0:00 Xsession
   508 pts/31   0:00 run-mozi
 29711 ?        0:00 sh
   544 ?        0:02 nautilus
 29665 ?        0:00 utaction
 29692 ?        0:00 dsdm
 29651 ?        0:00 utslaunc
   938 pts/31   0:01 emacs
 29723 ?        0:01 gnome-sm
 29709 pts/16   0:01 xscreens
 29681 pts/16   0:00 sdt_shel
 29860 ?        0:00 gnome-pt
 29736 ?        0:01 galf-ser
  4634 ?        1:52 Xsun
 29647 ?        0:02 utaudio
 29859 ?        0:03 gnome-te
 29698 pts/16   0:00 gnome-se
 29731 ?        1:20 nautilus
 29683 pts/16   0:00 tcsh
   516 pts/31   0:28 mozilla-
 29714 ?        0:00 bonobo-a
 29598 ?        0:00 Xsession
 29729 ?        0:07 gnome-pa
 29707 pts/16   0:01 gconfd-2
 29717 ?        0:00 gnome-se
 29773 ?        0:01 nautilus
   984 pts/31   0:06 acroread
```

```
29738 ?        0:00 mixer_ap
  479 pts/31   0:00 mozilla
29863 pts/31   0:00 tcsh
```

Para *eliminar* un proceso no deseado (por ejemplo, un programa que ya no responde), se usa la instrucción `kill`.

Primero hay que utilizar `ps` para identificar el número del proceso que vamos a eliminar, por ejemplo `984` (el `acroread` de `yajaira`) y con esta escribamos `kill 984`. Después, si el proceso no aparece muerto y siga en la lista de `ps`, intentamos con una señal más fuerte de terminación, `kill -9 984`. Con este parámetro `ps -9` se fuerza el proceso a terminar, mientras `ps` es más suave y simplemente "sugiere" al proceso que se cierre. Nota que solamente es posible eliminar procesos del usuario mismo — para matar un proceso de otro usuario, hay que avisar a un administrador que lo eliminen ellos.

Capítulo 2

Comunicación por Internet

Para tener una conexión a Internet en el domicilio, hay que contar con un servicio de banda ancha (típicamente la conexión será vía telefónica o cable). El costo mensual es cerca de 400 pesos por una conexión de velocidad de 512kb/s. Al tener acceso a Internet, se pueden formar diferentes tipos de conexiones entre la computadora del dominio a servidores y/o programas de tipo P2P (peer-to-peer, "entre iguales").

2.1. Conexiones entre computadoras

En esta sección, revisamos los métodos más importantes para realizar transferencia de información entre computadoras a través de Internet, tanto obteniendo acceso directo al sistema operativo y los discos duros de la otra computadora como intercambiando mensajes o datos con la computadora con o sin la participación de otro usuario.

2.1.1. ssh

El ssh (Secure SHell) es un protocolo de *comunicación cifrada* para transferir información entre las computadoras. Es decir, un tercero no puede (fácilmente) "escuchar" la transmisión y obtener información como contraseñas o los otros datos enviados. La cifra que usa es la del algoritmo RSA.

Es necesario tener el *servidor* ssh instalado y ejecutando en la computadora *a la cual se quiere conectar* y un programa tipo cliente (más liviano), que se ejecuta en la computadora *desde la cual se forma la conexión*.

En sistemas basados en Linux, un cliente de ssh suele ser incluido en la instalación básica. Para Windows, una opción gratuita de los clientes de ssh para terminales textuales es *PuTTY* [32].

Con ssh, se necesita determinar el *nombre del servidor* (por ejemplo, `yalma.fime.uanl.mx`) con el cual

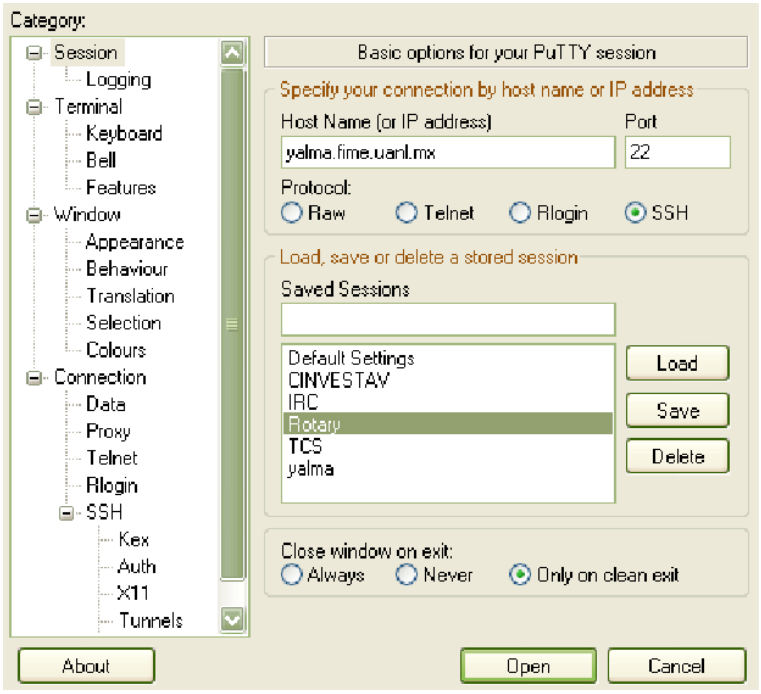


Figura 2.1: Pantalla inicial de PuTTY.

se quiere establecer comunicación o alternativamente su dirección IP (por ejemplo 148.234.29.130), tener una *cuenta de usuario* en el servidor y conocer la *contraseña*. En Unix/Linux, funciona como en el ejemplo siguiente:

```
> ssh micuenta@yalma.fime.uanl.mx
micuenta@yalma.fime.uanl.mx's password:
Last login: Thu Feb  8 09:30:14 2007 from tu.compu.algo.mx
Sun Microsystems Inc.   SunOS 5.9       Generic May 2002
You have mail.
>
```

Después de la pantalla de inicial de la figura 2.1, PuTTY abre una ventana con el texto `login as:`, donde se ingresa el usuario:

```
login as: miusuario
miusuario@yalma.fime.uanl.mx's password:
```

Al haber hecho eso, la conexión está establecida.

2.1.2. Ftp

Ftp (File Transfer Protocol) es un protocolo para transferencia de archivos entre computadoras (a un servidor que cuenta con ciertos servicios habilitados)— el tráfico por Ftp *no* está cifrado (o sea,

alguien capturando los paquetes de IP puede ver la contraseña y toda la información que está siendo transferida), pero existen versiones que utilizan SSL (Secure Sockets Layer) para cifrar la transferencia.

Se necesita conocer el nombre o dirección IP del servidor y tener una cuenta de usuario.

En terminales textuales (como en el *command prompt* de Windows o las consolas de sistemas de tipo Unix):

```
> ftp
ftp> open yalma.fime.uanl.mx
Connected to yalma.fime.uanl.mx.
220 yalma FTP server ready.
Name (yalma.fime.uanl.mx:algo): micuenta
331 Password required for micuenta.
Password:
230 User micuenta logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Las instrucciones básicas de Ftp son:

`open` ⟨servidor⟩	abrir una conexión a servidor
`close`	cerrar la conexión actual
`bin`	transferencia en modo binario
`ascii`	transferencia de puro texto
`ls`	ver los contenidos de la carpeta actual
`cd` ⟨carpeta⟩	cambiar carpeta en el servidor
`lcd` ⟨carpeta⟩	cambiar carpeta en la computadora local
`prompt`	confirmaciones on/off
`put` ⟨archivo⟩	cargar un archivo
`get` ⟨archivo⟩	descargar un archivo
`mput` ⟨archivos⟩	cargar varios archivos
`mget` ⟨archivos⟩	descargar varios archivos
`bye`	cerrar todas las sesiones y salir

Existen herramientas gráficas de Ftp, algunas puramente gratuitas u otras que solamente son gratuitas para uso no-comercial [30, 7, 22]. Con las herramientas gráficas, la transferencia de archivos funciona típicamente como copiar archivos de una carpeta a otra en una interfaz de usuario gráfica de cualquier PC. Como ejemplo de un cliente de Ftp para Windows, incluimos capturas de pantalla de WS_FTP [20], la edición limitada de cual es gratuita. En `http://www.ujaen.es/sci/redes/ftp/wsftp/` hay una manual de uso en castellano de WS_FTP; la herramienta está disponible para descargar de `yalma.fime.uanl.mx`, de la página `http://yalma.fime.uanl.mx/~pisis/ftp/pc-software/`.

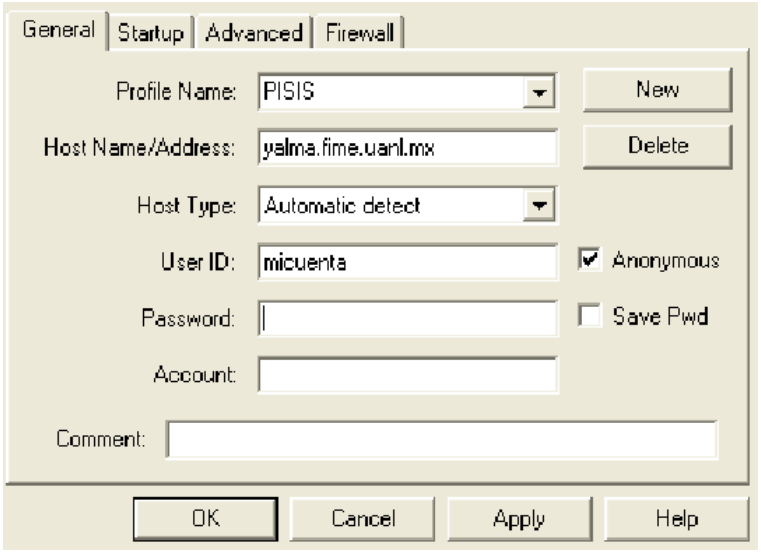


Figura 2.2: Pantalla inicial de conexión de WS_FTP.

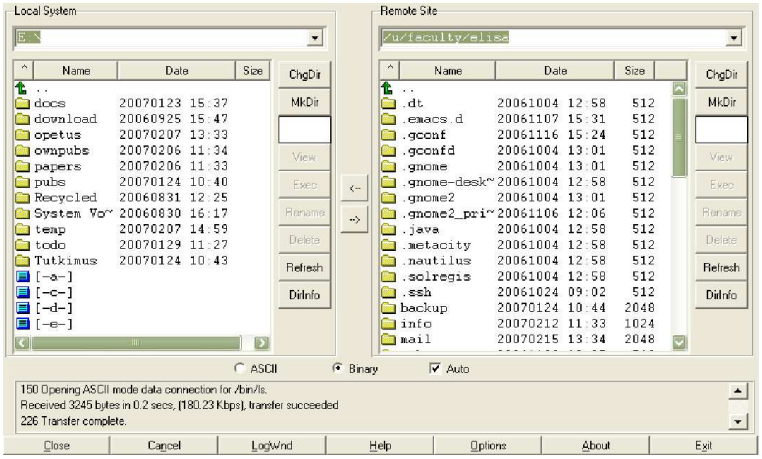


Figura 2.3: Vista de archivos WS_FTP al haber formado una conexión.

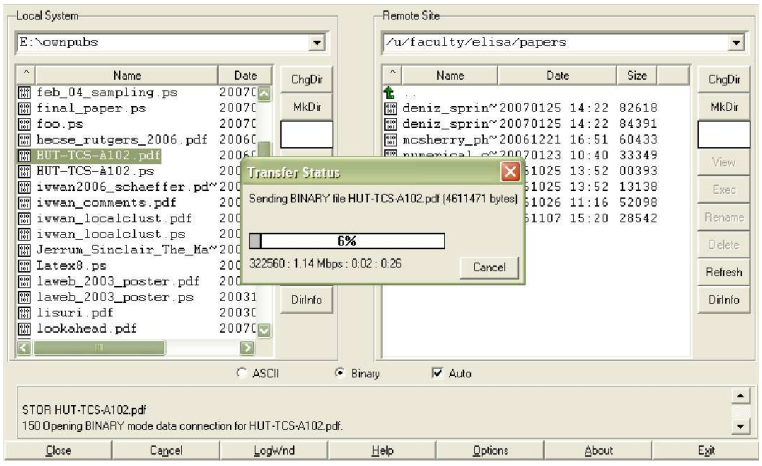


Figura 2.4: Pantalla de conexión de WS_FTP.

2.1.3. scp

Para copiar archivos con comunicación cifrada, se recomienda en Unix la instrucción `scp`. Para copiar algo (digamos `archivo.txt` de la carpeta donde se ejecutará la instrucción) del disco duro local a un servidor (digamos `yalma.fime.uanl.mx`, a la carpeta de inicio del usuario) que cuente con SSH (con usuario `miusuario`), se pone

```
scp archivo.txt miusuario@yalma.fime.uanl.mx:/home/miusuario/
```

y para copiar del servidor al sistema local (en la carpeta inicial del usuario)

```
scp miusuario@yalma.fime.uanl.mx:/home/miusuario/archivo.txt .
```

En Windows, hay que descargar WinSCP [28] u otra herramienta parecida. El funcionamiento de WinSCP es muy parecido a un cliente Ftp gráfico — además, también cuenta con Ftp, y entonces sirve para dos diferentes protocolos de transmisión.

2.2. Navegadores

Para acceder a sitios/páginas Web (o sea, comunicarse con servidores del protocolo Http), se necesita un *navegador*. No todos los navegadores son iguales con respecto a velocidad, eficiencia del uso, etcétera.

En la actualidad (2015), una buena opción no discutida aquí es Google Chrome (o Chromium en Linux).

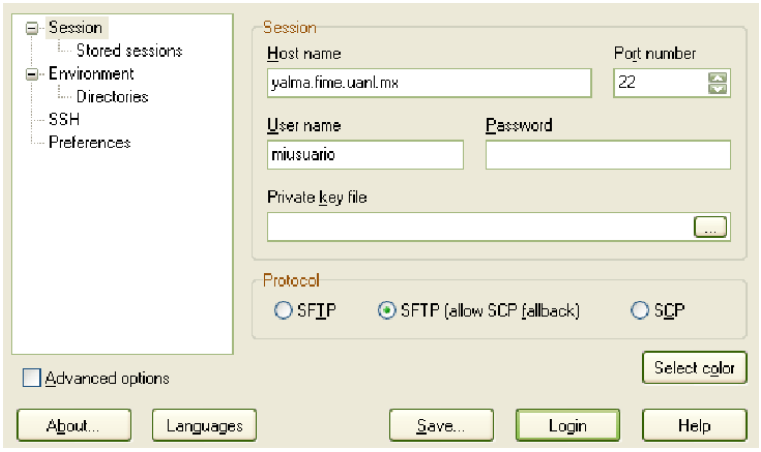


Figura 2.5: Pantalla de inicial de WinSCP.

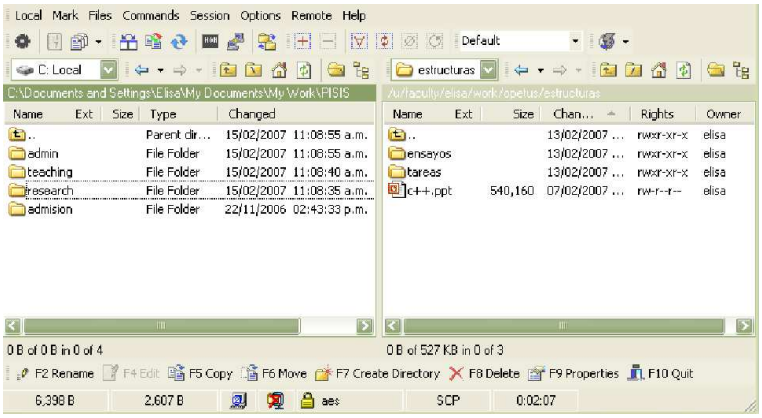


Figura 2.6: Vista de archivos de WinSCP.

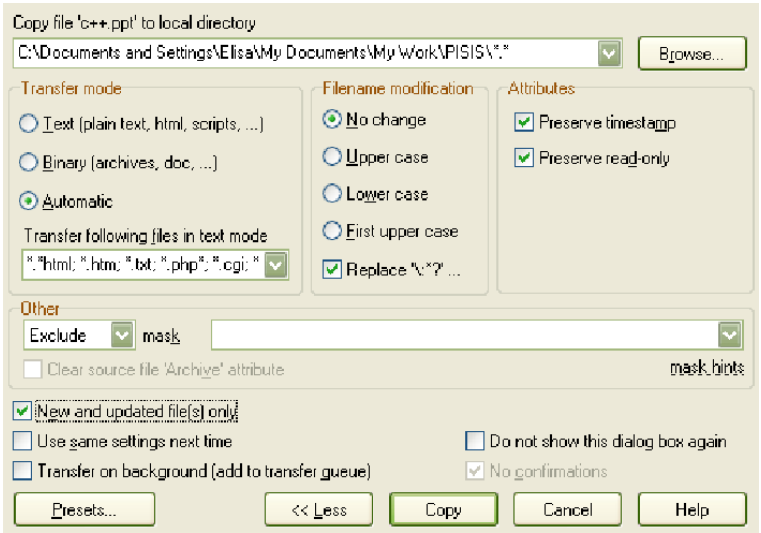


Figura 2.7: Ajustes de copiado de WinSCP.

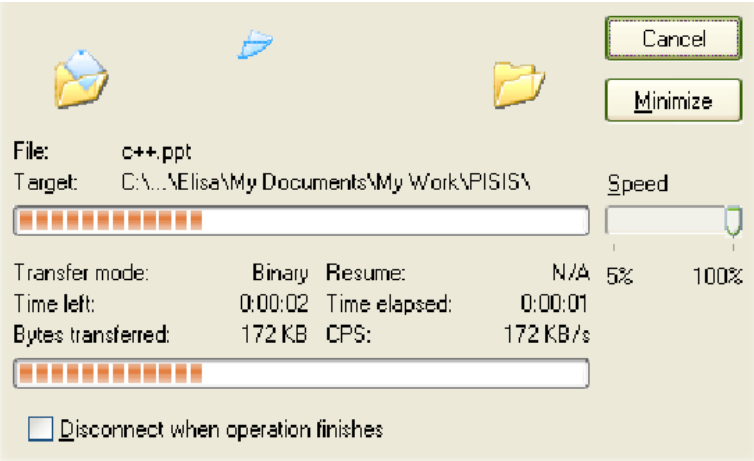


Figura 2.8: Progreso de copiar de WinSCP.

2.2.1. Mozilla Firefox

Una opción recomendable disponible en varios idiomas para sistemas operativos tipo Windows, Unix o Mac OS, es el navegador *Mozilla Firefox* [23]. Es muy flexible, fácil de usar, altamente compatible y por lo general muy rápido. Con componentes opcionales gratuitos (inglés: add-ons), se puede añadir funcionalidad útil al navegador.

2.2.2. `lynx`

A veces hace falta ver una página Web aunque no haya acceso a una terminal gráfica. Para sistemas tipo Unix, existe el navegador `lynx` que funciona con puro texto. Las instrucciones esenciales están en el Cuadro 2.1.

Una funcionalidad útil de `lynx` es que permite descargar archivos. Por ejemplo, si la página actual contiene un enlace a un archivo, al teclear "d" cuando en enlace está seleccionada produce un diálogo de descarga, donde se elige "Save to disk" (con ⟨enter⟩) y edita (si es necesario) el nombre del archivo para guardar. Desafortunadamente, por lo menos por ahora, `yalma.fime.uanl.mx` no cuenta con `lynx`.

2.2.3. `wget` — descargar sin navegar

Hay veces cuando uno quiere descargar un archivo específico de un cierto URL, pero abrir un navegador solamente para eso parece muy pesado. En esas situaciones, la instrucción `wget` ayuda. Por ejemplo, para descargar la versión actual de este mismo documento, basta con ejecutar

```
wget http://yalma.fime.uanl.mx/~elisa/teaching/taller/taller.pdf
```

Cuadro 2.1: Las instrucciones básicas de `lynx`.

h	ayuda (help)
q	salir (quit)
⟨espacio⟩	avanza por una pantalla
b	retrocede por una pantalla
⟨abajo⟩	al enlace siguiente
⟨arriba⟩	al enlace anterior
⟨izquierda⟩	seguir el enlace elegido
⟨derecha⟩	volver a la página anterior
d	descargar (download)

y el archivo será descargado.

```
> cd temp/
> ls
countsize    planets      primo        total.awk
datos.txt    primero      teclado.txt
> wget http://yalma.fime.uanl.mx/~elisa/teaching/taller/taller.pdf
--07:45:01--  http://yalma.fime.uanl.mx/%7Eelisa/teaching/taller/taller.pdf
           => ‘taller.pdf’
Connecting to yalma.fime.uanl.mx:80... connected!
HTTP request sent, awaiting response... 200 OK
Length: 2,596,434 [application/pdf]

    0K ..........  ..........  ..........  ..........  ..........   1% @  24.41 MB/s
   50K ..........  ..........  ..........  ..........  ..........   3% @  48.83 MB/s
  100K ..........  ..........  ..........  ..........  ..........   5% @  24.41 MB/s
  150K ..........  ..........  ..........  ..........  ..........   7% @  48.83 MB/s
  200K ..........  ..........  ..........  ..........  ..........   9% @  24.41 MB/s
  250K ..........  ..........  ..........  ..........  .......... 11% @  48.83 MB/s
  300K ..........  ..........  ..........  ..........  .......... 13% @  48.83 MB/s
  350K ..........  ..........  ..........  ..........  .......... 15% @  48.83 MB/s
  400K ..........  ..........  ..........  ..........  .......... 17% @  24.41 MB/s
  450K ..........  ..........  ..........  ..........  .......... 19% @  48.83 MB/s
  500K ..........  ..........  ..........  ..........  .......... 21% @  48.83 MB/s
  550K ..........  ..........  ..........  ..........  .......... 23% @  24.41 MB/s
  600K ..........  ..........  ..........  ..........  .......... 25% @  48.83 MB/s
  650K ..........  ..........  ..........  ..........  .......... 27% @  48.83 MB/s
  700K ..........  ..........  ..........  ..........  .......... 29% @  48.83 MB/s
  750K ..........  ..........  ..........  ..........  .......... 31% @  24.41 MB/s
  800K ..........  ..........  ..........  ..........  .......... 33% @  48.83 MB/s
  850K ..........  ..........  ..........  ..........  .......... 35% @  48.83 MB/s
  900K ..........  ..........  ..........  ..........  .......... 37% @  48.83 MB/s
  950K ..........  ..........  ..........  ..........  .......... 39% @  24.41 MB/s
 1000K ..........  ..........  ..........  ..........  .......... 41% @  24.41 MB/s
 1050K ..........  ..........  ..........  ..........  .......... 43% @  24.41 MB/s
 1100K ..........  ..........  ..........  ..........  .......... 45% @  48.83 MB/s
 1150K ..........  ..........  ..........  ..........  .......... 47% @  48.83 MB/s
 1200K ..........  ..........  ..........  ..........  .......... 49% @  24.41 MB/s
 1250K ..........  ..........  ..........  ..........  .......... 51% @  48.83 MB/s
 1300K ..........  ..........  ..........  ..........  .......... 53% @  48.83 MB/s
 1350K ..........  ..........  ..........  ..........  .......... 55% @  48.83 MB/s
 1400K ..........  ..........  ..........  ..........  .......... 57% @  24.41 MB/s
 1450K ..........  ..........  ..........  ..........  .......... 59% @  48.83 MB/s
 1500K ..........  ..........  ..........  ..........  .......... 61% @  48.83 MB/s
 1550K ..........  ..........  ..........  ..........  .......... 63% @  24.41 MB/s
```

```
 1600K ......... ......... ......... ......... ......... 65% @  48.83 MB/s
 1650K ......... ......... ......... ......... ......... 67% @  48.83 MB/s
 1700K ......... ......... ......... ......... ......... 69% @  24.41 MB/s
 1750K ......... ......... ......... ......... ......... 70% @  48.83 MB/s
 1800K ......... ......... ......... ......... ......... 72% @  48.83 MB/s
 1850K ......... ......... ......... ......... ......... 74% @  48.83 MB/s
 1900K ......... ......... ......... ......... ......... 76% @  24.41 MB/s
 1950K ......... ......... ......... ......... ......... 78% @  48.83 MB/s
 2000K ......... ......... ......... ......... ......... 80% @  48.83 MB/s
 2050K ......... ......... ......... ......... ......... 82% @  24.41 MB/s
 2100K ......... ......... ......... ......... ......... 84% @  48.83 MB/s
 2150K ......... ......... ......... ......... ......... 86% @  48.83 MB/s
 2200K ......... ......... ......... ......... ......... 88% @  24.41 MB/s
 2250K ......... ......... ......... ......... ......... 90% @  48.83 MB/s
 2300K ......... ......... ......... ......... ......... 92% @  48.83 MB/s
 2350K ......... ......... ......... ......... ......... 94% @  48.83 MB/s
 2400K ......... ......... ......... ......... ......... 96% @  48.83 MB/s
 2450K ......... ......... ......... ......... ......... 98% @  48.83 MB/s
 2500K ......... ......... ......... .....              100% @  34.75 MB/s

07:45:01 (36.96 MB/s) - 'taller.pdf' saved [2596434/2596434]

> ls
countsize    planets      primo        teclado.txt
datos.txt    primero      taller.pdf   total.awk
>
```

Si ya existe un archivo con el mismo nombre que el archivo que se descarga con `wget`, `wget` renombra el suyo con un número corriente: `taller.pdf.1`, `taller.pdf.2`, etcétera.

(Una opción es `curl` que ya viene con las Mac, mientras `wget` ocupa ser descargado e instalado aparte en ellas.)

2.3. Correo electrónico

2.3.1. Componentes de un correo electrónico

En esta sección se explica el significado de las partes esenciales de la cabecera del correo electrónico para que sea más fácil y seguro su uso eficiente.

- `To:` — las direcciones de correo de los destinatarios principales del mensaje. Hoy en día es común limitar el número total de recipientes por mensaje por ejemplo a unos 50, con la meta de limitar el envío de mensajes no deseados tipo "spam".
- `From:` — la dirección del correo electrónico de la persona que envía el mensaje. Hay que tener cuidado ya que con mucha facilidad se puede falsificar el remitente del mensaje para tratar de enviar un virus o tratar de robar información personal.
- `Date:` — fecha y hora en que el mensaje se ha enviado. Note que la fecha y hora depende de la máquina donde esté trabajando o mandando el mensaje ya que se pueden manipular, ya

que la hora que marca el correo es la de la máquina y si esta tiene desfasada o mal puesta la hora y fecha es la que marcará en el correo de salida.

- `Subject:` — asunto del que trata el mensaje. Es una parte importante del correo, es lo que lo distingue de los demás. Si es muy importante el asunto a tratar se le da énfasis al `Subject:` para que la persona que va a recibir el mensaje se de cuenta del asunto y la importancia que se le debe dar. Es de buen estilo poner algo concreto y conciso.
- `Cc:` — destinatarios en copia: aquí se agregan los correos electrónicos a los que quiere que se les mande el mensaje a parte del destinatario principal escrito en el `To:`. Estos se agregan, los envía y son visibles para todos receptores.
- `Bcc:` — copia carbón ciega: es como el `Cc:`, con la diferencia que en esta parte el receptor *no* podrá ver los correos agregados a los que se les envió el mensaje.
- `Attchment:` — archivos adjuntos: se puede enviar texto, archivos de diferentes extensiones, archivos comprimidos o programas ejecutables. Hay que tener mucho cuidado ya que puede recibir virus por este medio al tratar de abrir un ejecutable disfrazado o un archivo, es muy común que las máquinas se infecten de virus por descargar por este medio.
- `Reply-To:` — al contestar el mensaje, el programa de correo utiliza normalmente la dirección en `From:`, pero con esta opción se puede poner otra dirección (posiblemente de otra persona) como la dirección predeterminada a cual responder.

No siempre se encuentra una manera directa de manipular todas estas opciones en todos los servicios de correo electrónico comunes.

2.3.2. `pine`

Cuando ya establecida una conexión tipo SSH a un servidor de Unix(como `yalma.fime.uanl.mx`), se puede acceder correo electrónico que llega a la cuenta indicada con programas de modo de operación textual, como `pine`. El uso de `pine` es simple: la parte baja de la "ventana" muestra las instrucciones más comunes (vea Cuadro 2.2). `Control-x` significa que hay que oprimir las teclas `Control` y la letra `x` al mismo tiempo.

2.3.3. Firma automática

El `pine`, como la mayoría de programas para acceder correo electrónico, permite definir una firma para ser incluida en los mensajes enviados. Esa firma se escribe en un archivo con el nombre `.sig` o `.signature`. En tal archivo, en la primera línea hay que poner **dos guiones y un espacio blanco**. Es cortesía común evitar líneas largas y mantener la firma corta (unos 2–5 líneas debe ser suficiente). Se suele poner el nombre y algo de información de contacto (teléfono, afiliación, etcétera), por ejemplo

```
--
```

Cuadro 2.2: Las instrucciones fundamentales de `pine`.

	Navegación
`i`	ir a la carpeta de entrada (inbox)
`c`	escribir un nuevo mensaje (compose)
`l`	ir a la lista de carpetas (list of folders)
`m`	ir al menú principal
`q`	salir del programa (quit)
	En una carpeta
`s`	guardar en una carpeta (save)
`r`	contestar (reply)
	Al escribir un mensaje
`Control-x`	enviar mensaje
`Control-o`	posponer mensaje
`Control-c`	cancelar mensaje
	Al escribir un mensaje: área de texto
`Control-r`	insertar un archivo
`Control-t`	verificar ortografía (en inglés)
`Control-j`	alinear el texto
`Control-t`	verificar ortografía (en inglés)
	Al escribir un mensaje: cabecera
`Control-t`	elegir archivo para adjuntar (En Attchment)
`Control-t`	elegir un recipiente de (En Attchment)

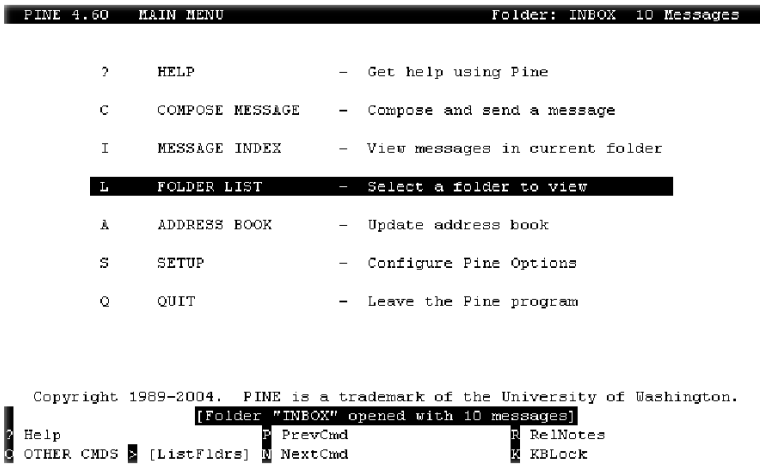


Figura 2.9: Pantalla inicial de `pine`.

```
Lic. Nombre Apellido
Programa de Posgrado en Ingeniería de Sistemas
Estudiante                    +52 81 1234 4567
FIME / UANL         usuario@yalma.fime.uanl.mx
```

En pine, si no se quiere utilizar la firma siempre, se puede incluir el archivo `.sig` como si fuera cualquier archivo por teclar `Control-r` en el área de mensaje. Para incluirla automáticamente en cada mensaje, en "Main Menu" (M), elegir "Setup" (S), "Config" (C), hay que poner en `signature-file` el nombre del archivo que contenga la firma. Si se prefiere tener la firma incluida casi siempre, es una buena idea configurarla a ser incluida siempre y quitarla con unas repeticiones de `Control-k` en el editor (sea `pico` o `emacs`) los pocos casos cuando es no deseada. Cuando se contesta mensajes ("Reply", R), se puede definir si la firma se pone al final de todo el mensaje o antes del mensaje posiblemente incluido en la respuesta en `Setup/Config/signature-at-bottom`.

2.3.4. Gmail

Es útil contar con algún correo "libre" con accesso a través de cualquier navegador. Existen varias opciones, como los de Yahoo! y MSN. En este documento, presentamos uno de los más sencillos, el Gmail de Google [15]. Abrir una cuenta de Gmail era *por invitación* (algún amigo necesitaba enviarles una invitación por su cuenta de Gmail), pero actualmente permiten inscribirse sin invitaciones. El correo mismo se puede elegir libremente, si no está en uso ya por otra persona. Es una buena idea abrir por lo menos una cuenta con su nombre y apellido para uso "formal" (`nombre.apellido@gmail.com`) y utilizar otra cuenta (u otro servicio, como el de Hotmail de Microsoft) donde se busca ser anónimo (`loquesea@hotmail.com`).

En Gmail, se puede buscar por mensajes recibidos y enviados. El filtro de correo basura (ingl. *spam*) de Gmail es bastante bueno. Para *mensajes instantáneos*, Gmail incorpora una herramienta Google Talk [16], pero Google Talk está también disponible como un cliente independiente del navegador. En la Sección **??**, veremos otros programas de mensajero instantáneo por Internet.

2.3.5. Reenvío automático

En sistemas tipo Unix, se puede crear un archivo de nombre `.forward` para determinar a cuál dirección debería dirigir el correo llegando a la cuenta: hay que poner en el archivo una sola línea que indique la dirección a cuál quiere dirigir el reenvío.

En sistemas como Gmail, existe una opción para hacer lo mismo (en la página de *Settings*, bajo *Forwarding and POP*).

Capítulo 3

Manejo de archivos

3.1. Almacenaje y compresión de datos

Para copiar varios archivos para almacenar o tomar respaldos, es a veces conveniente preparar un "paquete" de ellos: un sólo archivo que consiste de varios y de esa manera se puede con una herramienta *extraer* uno o todos los archivos guardados. Existen varias herramientas para esto, una de las más básicas es `tar`. Para *crear* un paquete, se usa la opción `-c` y para *abrir* un paquete, la opción `-x`. Hay que definir también el nombre del archivo de paquete.

Con ese tipo de paquetes, igual como con cualquier archivo grande, es posible que sus contenidos tengan bastante redundancia, por lo cual se puede *comprimir* el archivo para que ocupe menos espacio en el disco duro. Los algoritmos y en consecuencia las herramientas para compresión de datos son numerosos — en el mundo de Unix lo más común es el formato `.gz`: la compresión se hace con `gzip` y la descompresión (o sea, recuperación de los datos) con `gunzip`. Nota que `gzip` siempre *reemplaza* el archivo original y solamente queda la versión comprimida.

El ejemplo siguiente ilustra la creación de un paquete con 17 archivos tipo `.eps`, su compresión, transferencia a otra parte, descompresión y apertura:

```
> cd temp/
> ls
b1.eps       b5.eps       ch3.eps      p2.eps       s2.eps       total.awk
b2.eps       b6.eps       ch4.eps      p3.eps       s3.eps
b3.eps       ch1.eps      datos.txt    p4.eps       s4.eps
b4.eps       ch2.eps      p1.eps       s1.eps       teclado.txt
> tar -cvf figuras.tar *.eps
a b1.eps 361K
a b2.eps 602K
a b3.eps 957K
a b4.eps 792K
a b5.eps 938K
a b6.eps 752K
a ch1.eps 290K
a ch2.eps 309K
a ch3.eps 422K
```

```
a ch4.eps 174K
a p1.eps 87K
a p2.eps 163K
a p3.eps 230K
a p4.eps 90K
a s1.eps 169K
a s2.eps 234K
a s3.eps 328K
a s4.eps 197K
> ls
b1.eps       b5.eps       ch3.eps      p1.eps       s1.eps       teclado.txt
b2.eps       b6.eps       ch4.eps      p2.eps       s2.eps       total.awk
b3.eps       ch1.eps      datos.txt    p3.eps       s3.eps
b4.eps       ch2.eps      figuras.tar  p4.eps       s4.eps
> ls -lh figuras.tar
-rw-r--r--   1 elisa    faculty     6.9M Jun 22 15:38 figuras.tar
> gzip figuras.tar
> ls
b1.eps          b6.eps          datos.txt       p4.eps          teclado.txt
b2.eps          ch1.eps         figuras.tar.gz  s1.eps          total.awk
b3.eps          ch2.eps         p1.eps          s2.eps
b4.eps          ch3.eps         p2.eps          s3.eps
b5.eps          ch4.eps         p3.eps          s4.eps
> ls -lh figuras.tar.gz
-rw-r--r--   1 elisa    faculty     1.4M Jun 22 15:38 figuras.tar.gz
> cp figuras.tar.gz ~/public_html/temp/
> cd ~/public_html/temp/
> gunzip figuras.tar.gz
> tar -xvf figuras.tar
x b1.eps, 368856 bytes, 721 tape blocks
x b2.eps, 616182 bytes, 1204 tape blocks
x b3.eps, 979302 bytes, 1913 tape blocks
x b4.eps, 810822 bytes, 1584 tape blocks
x b5.eps, 960401 bytes, 1876 tape blocks
x b6.eps, 769466 bytes, 1503 tape blocks
x ch1.eps, 296656 bytes, 580 tape blocks
x ch2.eps, 316141 bytes, 618 tape blocks
x ch3.eps, 431254 bytes, 843 tape blocks
x ch4.eps, 177337 bytes, 347 tape blocks
x p1.eps, 88475 bytes, 173 tape blocks
x p2.eps, 166073 bytes, 325 tape blocks
x p3.eps, 234921 bytes, 459 tape blocks
x p4.eps, 92108 bytes, 180 tape blocks
x s1.eps, 172860 bytes, 338 tape blocks
x s2.eps, 238623 bytes, 467 tape blocks
x s3.eps, 335016 bytes, 655 tape blocks
x s4.eps, 200802 bytes, 393 tape blocks
> ls *.eps
b1.eps   b4.eps   ch1.eps  ch4.eps  p3.eps   s2.eps
b2.eps   b5.eps   ch2.eps  p1.eps   p4.eps   s3.eps
b3.eps   b6.eps   ch3.eps  p2.eps   s1.eps   s4.eps
>
```

Desde la página `http://www.7-zip.org/` se puede descargar una herramienta para la línea de instrucciones de Microsoft Windows que permite crear archivos comprimidos de diferentes formatos y abrir archivos comprimidos.

3.2. Control de versiones

(En la actualidad, la opción recomendable es `git`, `https://git-scm.com/`), que tiene los mismos principios que la herramienta discutida aquí.)

`cvs` es una aplicación de informática, que nos ayuda a registrar y actualizar cada movimiento de un proyecto cualquiera desarrollado en forma colaborativa entre varias personas o por la misma persona en varias computadoras.

La forma en que trabaja `cvs` es haciendo un *repositorio* de los datos del proyecto en una carpeta específica e inicializarlo con `cvs init` para empezar a usarlo. El nombre de la carpeta no importa, pero es importante que todos los usuarios de la máquina que necesitan acceso a los datos tengan permiso de accederlo — sí, hay que tener confianza en los otros usuarios[1]. Por ejemplo,

```
> mkdir repositorio
> chmod a+rw repositorio
> cvs -d /u/faculty/elisa/repositorio/ init
>
```

La opción `-d` específica a `cvs` donde está ubicado el repositorio. Un repositorio es como una base de datos o un depósito de datos donde se guardar la información cada vez que le demos la orden de hacerlo y guardara cada cambio hecho y no desechará la información cambiada o eliminada, sino que la guarda por si en un futuro deseamos volverla a utilizar.

Si el proyecto ya cuenta con algunos datos, hay que importarlas al repositorio con `cvs import`: si los datos están en la carpeta `tmp`, movemos allí y realizamos la importación, definiendo un mensaje con la opción `-m`, definiendo un nombre para el proyecto dentro del repositorio (`diap`) e información sobre quién y porqué está haciendo el import.

```
> cd tmp/
> cvs -d /u/faculty/elisa/repositorio/ import -m "Diapositivas"  diap elisa inicio
N diap/diapositivas.tex
N diap/b1.eps
N diap/b2.eps
N diap/b3.eps
N diap/b4.eps
N diap/b5.eps
N diap/b6.eps
N diap/ch1.eps
N diap/ch2.eps
N diap/ch3.eps
N diap/ch4.eps
N diap/p1.eps
N diap/p2.eps
N diap/p3.eps
N diap/p4.eps
N diap/s1.eps
N diap/s2.eps
N diap/s3.eps
```

[1] Si es posible crear un grupo de trabajo en el servidor, mejor, pero eso es algo que pueden hacer los administradores de sistemas tipo Unix.

```
N diap/s4.eps

No conflicts created by this import
>
```

Ahora el repositorio ya contiene toda la información y podemos, si queremos, eliminar las copias provisionales:

```
> cd ..
> rm -rf tmp/
>
```

Ahora hay que sacar una copia para trabajar localmente — *nunca* se modifican manualmente los contenidos del repositorio. Para sacar una copia de los contenidos, usamos `cvs checkout`:

```
> cvs -d /u/faculty/elisa/repositorio/ checkout diap
cvs checkout: Updating diap
U diap/b1.eps
U diap/b2.eps
U diap/b3.eps
U diap/b4.eps
U diap/b5.eps
U diap/b6.eps
U diap/ch1.eps
U diap/ch2.eps
U diap/ch3.eps
U diap/ch4.eps
U diap/diapositivas.tex
U diap/p1.eps
U diap/p2.eps
U diap/p3.eps
U diap/p4.eps
U diap/s1.eps
U diap/s2.eps
U diap/s3.eps
U diap/s4.eps
>
```

Ahora podemos libremente editar las copias en la carpeta `diap` así generada. Después de terminar de modificar los datos, hay que sincronizar con el repositorio con `cvs commit`:

```
> cd diap/
> gimp s1.eps &
[1] 16026
> emacs diapositivas.tex  &
> cvs commit -m "Cambio de letra en el documento y ajuste de colores de un dibujo"
cvs commit: Examining .
Checking in diapositivas.tex;
/u/faculty/elisa/repositorio/diap/diapositivas.tex,v  <--  diapositivas.tex
new revision: 1.2; previous revision: 1.1
done
Checking in s1.eps;
/u/faculty/elisa/repositorio/diap/s1.eps,v  <--  s1.eps
new revision: 1.2; previous revision: 1.1
done
```

```
[1]-  Done                    gimp s1.eps
[2]+  Done                    emacs diapositivas.tex
>
```

Si queremos escribir un comentario más extenso sobre los cambios realizados, basta con dejar fuera lo de `-m` y `cvs` abrirá automáticamente un editor para escribir un mensaje (normalmente `emacs` o `pico`, depende de la configuración):

```
Lo que hice era cortar un párafo de texto del comienzo que no me pareció
adequado justo allí.
CVS: ----------------------------------------------------------------------
CVS: Enter Log.  Lines beginning with 'CVS:' are removed automatically
CVS:
CVS: Committing in .
CVS:
CVS: Modified Files:
CVS:    diapositivas.tex
CVS: ----------------------------------------------------------------------
```

Se puede crear otra copia de trabajo para otro usuario, otra computadora, etcétera. Para acceder a `cvs` a través de `ssh`, hay que añadir información del servidor, usuario y protocolo en `-d`: ejecutando en otra máquina,

```
> cvs -d :ext:elisa@yalma.fime.uanl.mx:/u/faculty/elisa/repositorio checkout diap
elisa@yalma.fime.uanl.mx's password:
cvs checkout: Updating diap
U diap/b1.eps
U diap/b2.eps
U diap/b3.eps
U diap/b4.eps
U diap/b5.eps
U diap/b6.eps
U diap/ch1.eps
U diap/ch2.eps
U diap/ch3.eps
U diap/ch4.eps
U diap/diapositivas.tex
U diap/p1.eps
U diap/p2.eps
U diap/p3.eps
U diap/p4.eps
U diap/s1.eps
U diap/s2.eps
U diap/s3.eps
U diap/s4.eps
>
```

Ahora modificamos la otra copia:

```
> emacs diapositivas.tex &
[1] 7972
> cvs commit
cvs commit: Examining .
Checking in diapositivas.tex;
```

```
/u/faculty/elisa/repositorio/diap/diapositivas.tex,v  <--  diapositivas.tex
new revision: 1.4; previous revision: 1.3
done
[1]+  Done                    xemacs diapositivas.tex
>
```

En la *primera* copia, para poder ver los cambios hechos en la segunda, hay que realizar una actualización de los datos. Es recomendable hacer eso cada vez que volvemos a trabajar sobre el proyecto. Hacemos entonces un `cvs update` para retirar las versiones más recientes del repositorio:

```
> cvs update
cvs update: Updating .
U diapositivas.tex
>
```

Para añadir un archivo nuevo, se usa `cvs add` con el nombre del archivo como parámetro. Después hay que ejecutar `cvs commit` para que tome efecto la adición. Para remover un archivo del repositorio, primero hay que eliminarlo de la copia de trabajo y después ejecutar `cvs delete` con el nombre del archivo ya eliminado. Igualmente hay que ejecutar `cvs commit` para que tome efecto la eliminación.

Si intentamos hacer un `cvs commit` cuando no tenemos una versión "fresca" de los datos, `cvs` va a marcarnos un error. También si dos personas hacen un commit que modifica la misma parte del mismo archivo o un `cvs update` modificaría algo que ya modificaste pero no has hecho un `cvs commit`, `cvs` avisa y marca en el archivo en qué parte hay conflictos.

Definiendo las variables ambientales `CVSROOT` y `CVSEDITOR`, por ejemplo, podemos evitar definir algunos datos en la línea de instrucciones (en `bash` se hace con `export`).

Para aprender más detalles sobre `cvs`, hay un manual bueno en formato Wiki en `http://ximbiot.com/cvs/wiki/` y una introducción en español en `http://acm2.asoc.fi.upm.es/~chernando/doc/cvs/`.

Para acceder un repositorio cd `cvs` desde Microsoft Windows, una buena opción es Tortoise CVS (`http://www.tortoisecvs.org/`) que integra al Windows Explorer. En la figura 3.1 se muestra la forma de hacer un `cvs checkout` desde Explorer y en la figura 3.2 se muestra la vista de unas carpetas y archivos que vienen de un repositorio de `cvs`. Existe una versión más avanzada de `cvs` que se llama Subversion para los que quieren más.

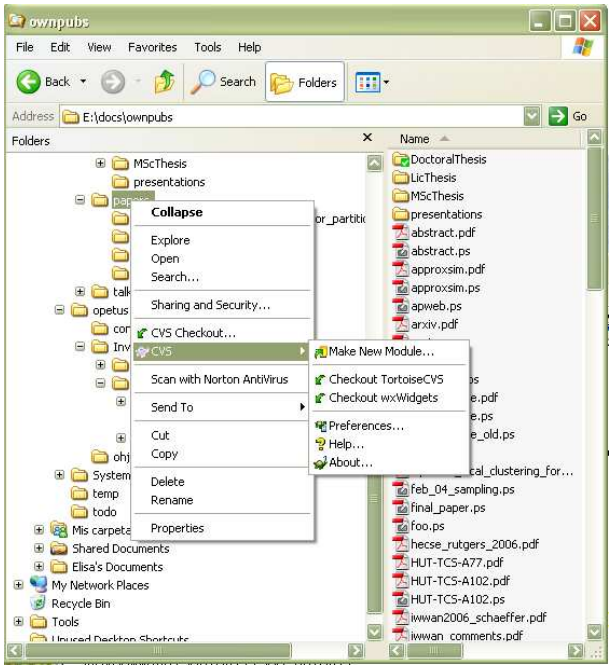


Figura 3.1: Tortoise CVS integrado a Windows Explorer — hacer click con el botón derecha del ratón muestra sus funciones para ajustar la configuración o hacer un `checkout` nuevo.

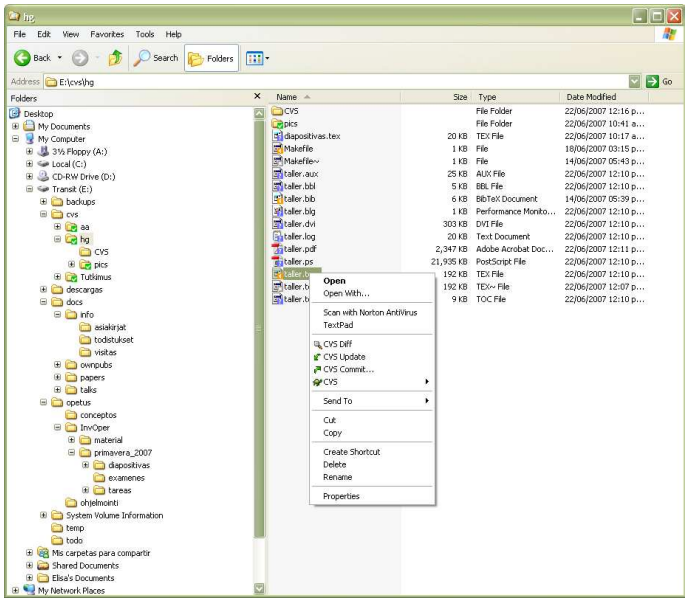


Figura 3.2: Tortoise CVS integrado a Windows Explorer — hacer click con el botón derecho del ratón muestra sus funciones; eligiendo archivos se puede realizar las acciones `cvs commit`, `cvs update`, `cvs add` y `cvs delete`, entre otras.

Capítulo 4

Preparación de documentos (de texto)

Este tema divide en cuatro sesiones: una sesión para temas 4.1–4.5, una sesión para cada uno de los temas 4.6, 4.7 y 4.8.

4.1. Emacs

Emacs es un editor libre de textos que cuenta con varias extensiones para diferentes tipos de tareas. Es ideal para la programación de escala pequeña o intermediada y preparación de documentos en HTML o con LaTeX, por ejemplo.

Su uso más eficiente es por instrucciones cortas del teclado, aunque las interfaces gráficas también permiten operar por iconos o menús. En la figura 4.1, se muestra XEmacs en su estado inicial.

El cuadro 4.1 define algunas de las instrucciones más básicas de Emacs.

Emacs automáticamente guarda versiones intermedias de los documentos editados por si acaso algo va mal. Una copia del archivo `miarchivo.dat` está generada al inicio de Emacs con el nombre `miarchivo.dat~` y a cada rato la versión actual se guarda en `#miarchivo.dat#`.

4.2. Pico y nano

Se ejecuta los editores `pico` y `nano` por escribir su nombre en la línea de instrucciones. Hay diferentes formas de abrir y utilizar el programa pico ya sea primero creando un archivo o modificando uno: `pico archivo.dat` crea un archivo con el nombre `archivo.dat` si todavía no existe, y abre el archivo con el nombre especificado si uno existe.

Para el uso más fácil el usuario tiene instrucciones para realizar con más rapidez las tareas a realizar. El Cuadro 4.2 contiene algunas opciones con las que cuenta este programa.

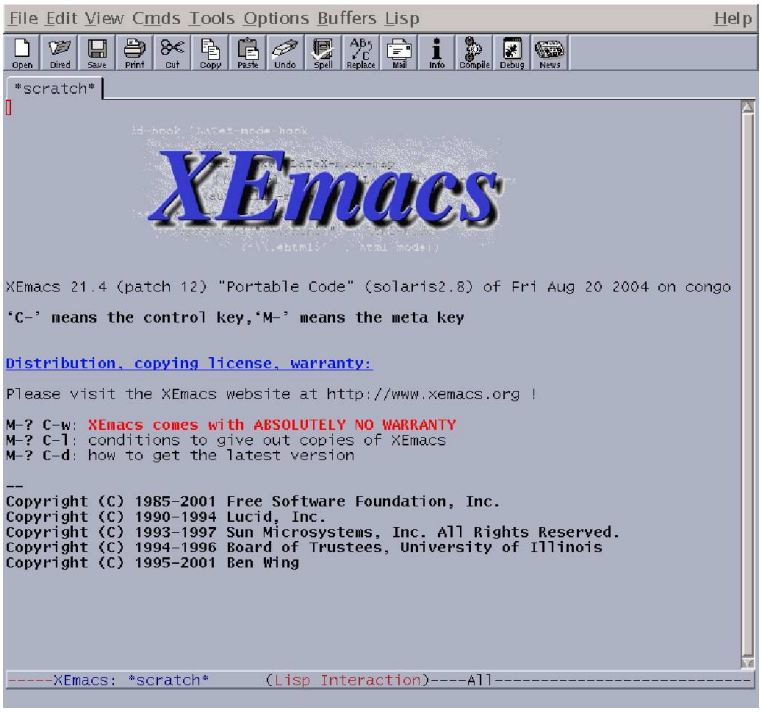


Figura 4.1: XEmacs en el estado inicial con ningún archivo abierto.

Cuadro 4.1: Las instrucciones esenciales de Emacs.

`Control-g`	volver
`Control-s`	buscar en el archivo
`Control-a`	ir al comienzo de la línea actual
`Control-e`	ir al fin de la línea actual
`Control-k`	cortar línea(s)
`Control-y`	pegar las líneas cortadas
`Control-`(espacio)	marcar comienzo de región
`Control-w`	cortar la región marcada
`Control-x Control-s`	guardar archivo
`Control-x Control-f`	abrir archivo
`Control-x Control-c`	salir del programa
`Control-x u`	deshacer (undo)
`Control-x k`	cerrar archivo
`Control-x` ⟨número⟩	dividir la ventana en ⟨número⟩ partes
`Esc-x replace-string`	reemplazar todos
`Esc-x query-replace`	reemplazar preguntando
`Esc-x ispell`	verificar ortografía (en inglés)
`Esc-q`	insertar cortes en líneas largas del párrafo

La ventana de Pico está compuesta de varias partes. En la parte superior del lado izquierdo aparecerá la versión del programa. En la parte superior en el centro aparece el nombre del archivo que se esta

Cuadro 4.2: Algunas de las instrucciones del editor `pico`.

`Control-f` (o flecha derecha)	Mover adelante un carácter
`Control-b` (o flecha izquierda)	Mover hacia atrás un carácter
`Control-p` (o encima de flecha)	Levantar una línea
`Control-n` (o abajo flecha)	Bajar una línea
`Control-a`	Mover al principio de la línea actual
`Control-e`	Mover al extremo de la línea actual
`Control-v`	Mover adelante una pantalla de texto
`Control-y`	Mover al revés una pantalla de texto
`Control-j`	Justificar el párrafo actual
`Control-c`	Posición actual del cursor de la impresión
`Control-g`	Exhibir el texto de ayuda dentro del pico
`Control-x`	Salir del Pico
`Control-k`	Corta la información escrita
`Control-u`	Pega la información antes cortada
`Control-t`	Te lleva a buscar archivos

escribiendo. En la parte superior del lado derecho está el estado en el que se encuentra el archivo. Todo lo que se encuentra en blanco es el cuerpo del archivo es donde vas a escribir el archivo que vas a realizar.

4.3. Open Office

OpenOffice.org [25] es una "suite ofimática" de software libre y código abierto que incluye seis herramientas básicas. Además está disponible para muchas y diversas plataformas como Windows, Unix, Linux y Mac (en sus diferentes versiones). Las seis herramientas básicas de OpenOffice son

- Procesador de textos OpenOffice.org *Writer*,
- Editor de fórmulas OpenOffice.org *Math*,
- Hoja de cálculo OpenOffice.org *Calc*,
- Editor de dibujos y gráficos OpenOffice.org *Draw*,
- Editor de presentaciones OpenOffice.org *Impress* y
- Editor de páginas web OpenOffice.org *Web*.

Además ofrece un lenguaje de macros OpenOffice.org *Basic*, y una interfaz gráfica de base de datos.

Una de las ventajas que tiene este software es una gran variedad de opciones que puedes utilizar. Aparte que es muy parecido a los demás software utilizados para realizar tareas o trabajo de oficina.

4.3.1. OpenOffice.org Writer

El "procesador de palabras" es un hoja donde podemos realizar documentos, archivos o alguna tarea donde necesitemos explicar o desarrollar para entregar. Gracias a Writer, tenemos un mundo de herramientas a la mano para hacerlo de la mejor manera y aparte con mucha facilidad, ya que no es un programa muy complicado.

Este programa es muy parecido al de Microsoft Office; de hecho tiene las mismas funciones que utiliza Word, con la diferencia de que el procesador de palabras que aquí explicamos tiene muchas más opciones. Una de las opciones más importantes es que desde el procesador de palabras puedes *abrir y generar documentos PDF*.

Writer tiene un sin fin de herramientas para realizar el trabajo que necesitas, y casi todas las herramientas están a la vista del usuario, ya se que en pequeños dibujos damos un click o abriendo desde la barra de herramientas las opciones que buscamos.

4.3.2. OpenOffice.org Math

Este programa se utiliza normalmente para introducir fórmulas dentro del procesador de palabras. Sirve para hacer fórmulas con caracteres especiales que normalmente no vienen en el teclado común. También desde el mismo se puede trabajar para hacer las fórmulas y después pegarlas al procesador de palabras.

Math es una programa que ofrece muchas ventajas para el usuario, ya que su gran variedad de caracteres especiales hace que puedas realizar trabajos complicados nada más es de arrastrar el carácter deseado a la hoja del procesador de palabras, o a la misma hoja del programa Math para realizar la fórmula deseada. Dentro de esta pantalla hay una ventana pequeña dentro de la pantalla (en el lado derecho en la parte superior la figura 4.2.

4.3.3. OpenOffice.org Calc

La hoja de cálculo de OpenOffice.org es un programa muy sencillo que nos ayuda a hacer una gran infinidad de tareas, desde hacer sumas hasta hacer reportes especializados según necesidad.

Con este programa podemos realizar gráficas, meter datos y armar un listado de personal u objetos. Es muy sencilla ya que funciona parecido al Microsoft Excel. Nada más hay que seguir los pasos básicos y podrás realizar la tarea que deseas sin mucho esfuerzo. Dentro de las gráficas que podemos realizar, hay muchas opciones a elegir, desde las más simples hasta hacerlas en 3D dándole una excelente presentación a su trabajo

Dentro de la barra de herramientas se encuentran todas las opciones a utilizar para trabajar con mayor facilidad. También las herramientas que se usan con más frecuencia se encuentran en pequeños iconos: solamente con un click podemos utilizar esa opción para nuestro uso.

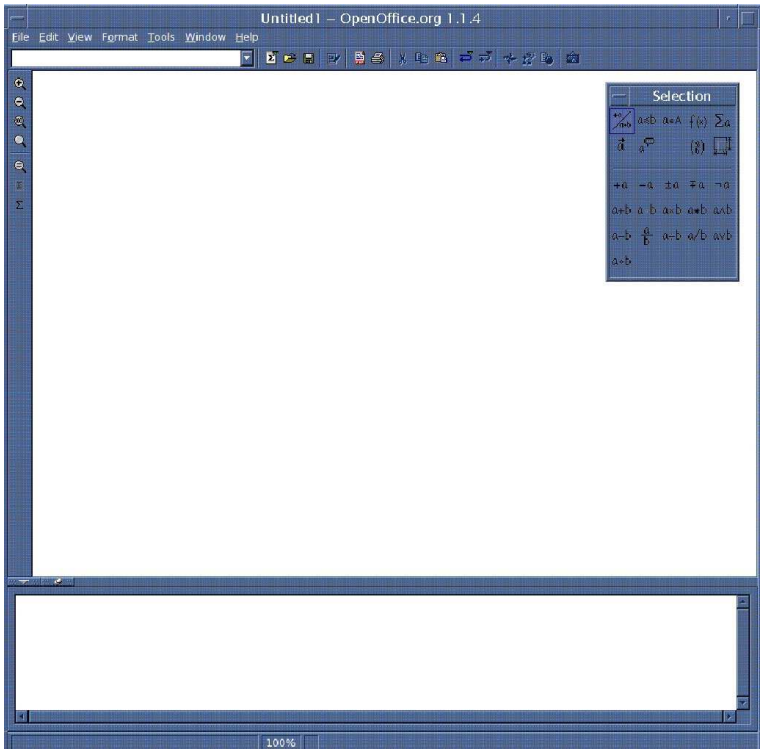


Figura 4.2: Opciones de OpenOffice.org Math para escoger un carácter para una fórmula.

Al igual que otras hojas de cálculo, hay instrucciones muy sencillas de realizar. Por ejemplo: para hacer una suma solamente poner `=sum (A1+B1)` en la celda `C1`, y le dará la suma de los valores de las celdas `A1` y `B1` en la celda `C1`. Para la multiplicación con el símbolo $*$ y división con el símbolo `/` es prácticamente lo mismo.

4.3.4. OpenOffice.org Draw

Este programa realiza dibujos y gráficos dentro de una hoja en blanco, o también se los puede introducir a un procesador de palabras o de cálculo.

Dentro de Draw puedes abrir cualquier archivo de OpenOffice; esto sirve por si necesitas un dibujo especializado o algún gráfico dentro del documento.

4.3.5. OpenOffice.org Impress

Impress ayuda a crear diapositivas, presentaciones y otros tipos de documentos para hacer una buena presentación. Si ya ha manejado el PowerPoint de Microsoft, no se le hará muy difícil manejar el Impress, ya que trabajan de forma muy similar.

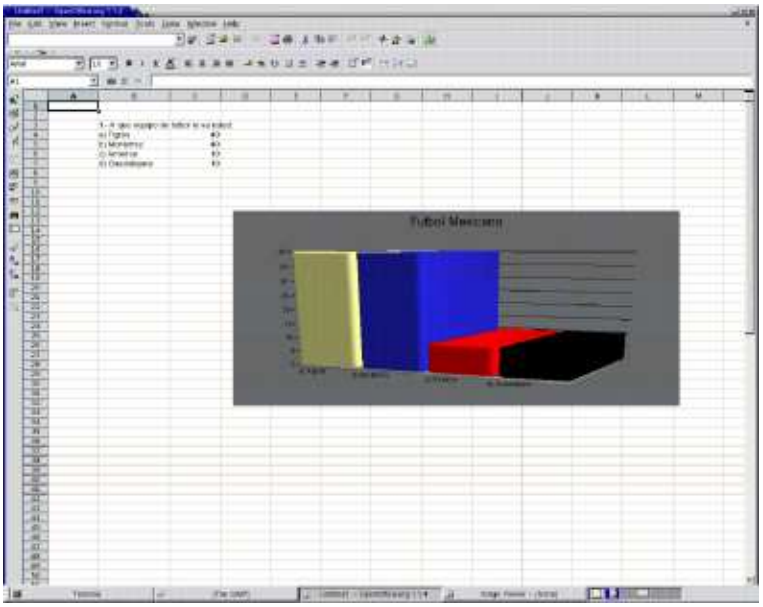


Figura 4.3: OpenOffice.org Calc: una gráfica de barras.

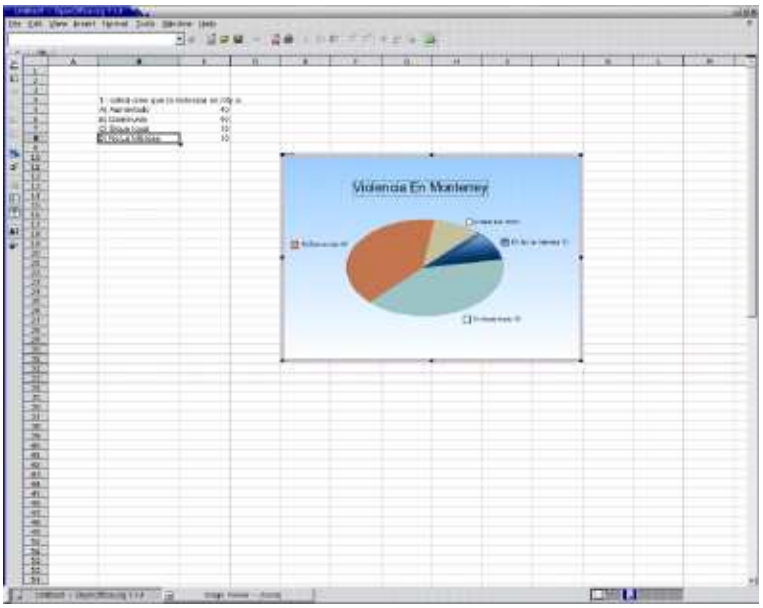


Figura 4.4: OpenOffice.org Calc: una gráfica tipo pay.

4.4. Verificación de ortografía: `ispell`

En Emacs, se puede ejecutar `ispell` para todo el documento actual con `Esc-x ispell`. Si solamente se requiere verificar una región, hay que marcar la región en Emacs tecleando `Ctrl-espacio` donde empieza y tecleando `Esc-x ispell-region` donde termina la región. Para cambiar la diccionario en uso, se utiliza `Esc-x ispell-change-dictionary`, cuando Emacs pregunta

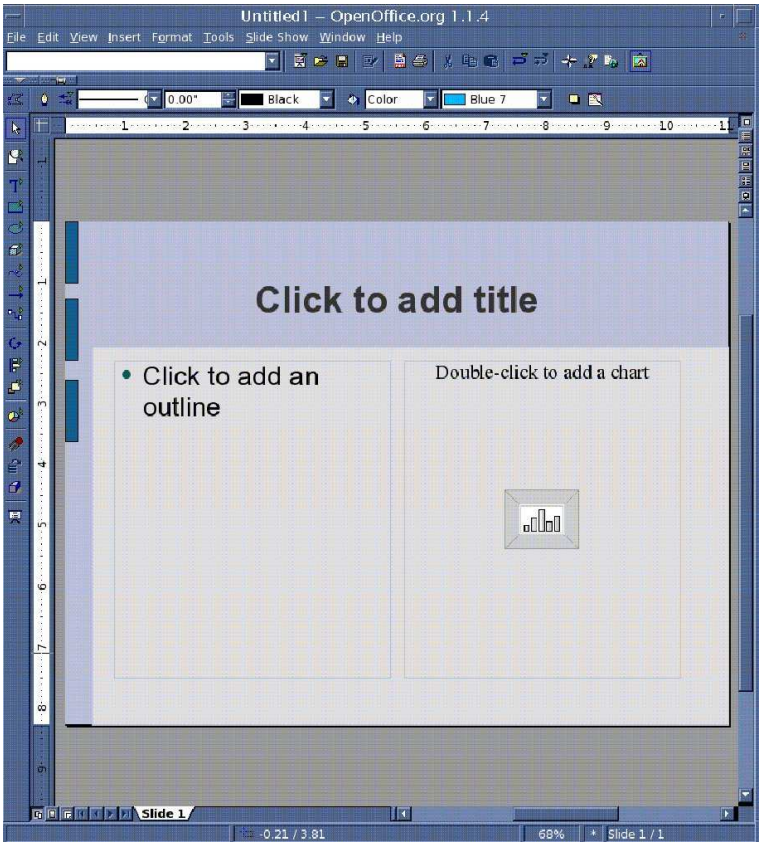


Figura 4.5: OpenOffice.org Impress.

```
Use new dictionary (RET for current, SPC to complete):
```

y para ver la lista de opciones instaladas actualmente, basta con teclear espacio. Por ejemplo,

```
Click <mouse-2> on a completion to select it.
In this buffer, type RET to select the completion near point.

Possible completions are:
american                       brasileiro
british                        castellano
castellano8                    czech
dansk                          default
deutsch                        deutsch8
english                        esperanto
esperanto-tex                  francais
francais-tex                   francais7
german                         german8
italiano                       nederlands
nederlands8                    norsk
norsk7-tex                     polish
portugues                      russian
slovak                         svenska
```

y los de `castellano` (para documentos escritos en LaTeX) y `castellano8` (para documentos con 8-bit símbolos) corresponden al español, mientras `american` y `british` son de inglés.

4.5. Formatos de distribución de documentos

4.5.1. PostScript

La terminación de archivos en formato PostScript es `.ps`. Varios programas incorporan "impresoras virtuales", o sea impresión a archivos tipo PostScript.

Para convertir documentos de puro texto a PostScript, en Unix se usa `enscript`. Para juntar varias páginas de un documento tipo `.ps`, en Unix se usa `psnup`.

4.5.2. Adobe Pdf

Pdf es un formato de documentos desarrollado por la empresa Adobe Systems Incorporated. Los documentos mismos contienen todos los datos necesarios para mostrarlos sin cambios en cualquier plataforma/computadora. Es un formato preferible cuando se envía un documento (ensayo/tesis/informe) para evaluación a otra persona, porque siempre es fácil abrir y/o imprimir el documento con herramientas gratuitas y muy raramente hay problemas de compatibilidad.

Para *abrir* documentos de formato Pdf, en Unix se usan las instrucciones `acroread` y `xpdf`, entre otras. Ambos toman como parámetro de la línea de instrucciones el nombre del archivo a abrir (o varios). En Windows, se necesita descargar e instalar Acrobat Reader [2].

En Unix, hacer *conversión* a Pdf de otros formatos se hace por instrucciones como `ps2pdf` (conversión de PostScript, y ya en la sección anterior se explicó cómo transformar otros tipos de documentos a formato PostScript). En Windows, la manera más fácil de convertir todo tipo de documentos a Pdf es instalar alguna de las *impresoras virtuales* como el PrimoPDF [1] o PDF Creator [27] que también permite combinar varios documentos de Pdf a uno solo. En la figura 4.6, se muestra como después de haber instalado PrimoPDF, se aparece como una impresora normal.

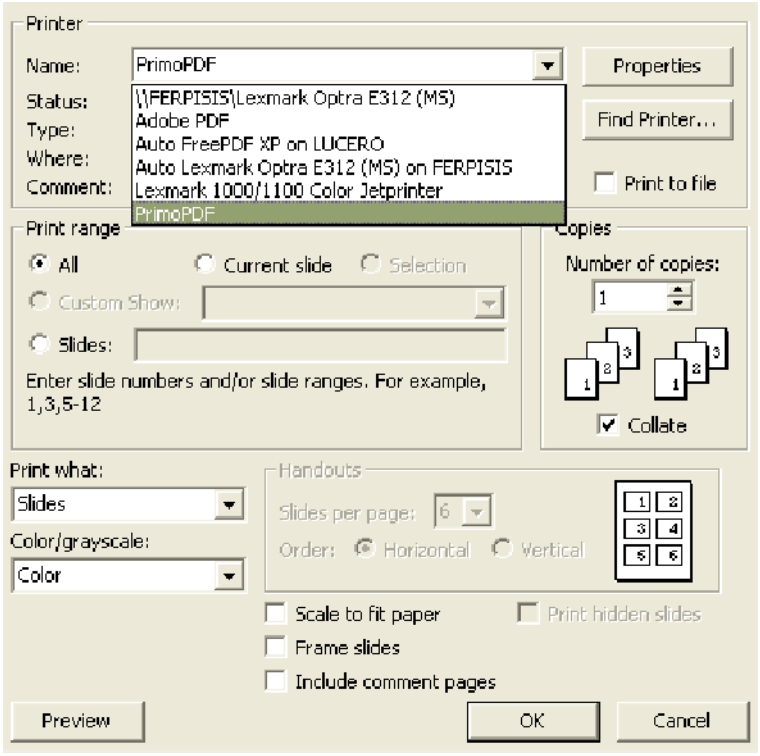


Figura 4.6: Impresión en Microsoft PowerPoint: PrimoPDF en la lista de impresoras disponibles.

Figura 4.7: Opciones de PrimoPDF al haber impreso a PrimoPDF.

4.6. LaTeX

Cada documento de LaTeX debe tener un tipo que se determina con una instrucción `documentclass`— las opciones incluyen por ejemplo `article` y `report`. Los contenidos del documento van entre las instrucciones
`begin{document}` y
`end{document}`. Para compilar un documento de ejemplo guardado en `doc.tex` con los siguientes contenidos

```
\documentclass{article}
\begin{document}
Hola
\end{document}
```

se ejecuta primero la instrucción `latex`:

```
> latex doc
This is TeX, Version 3.14159 (Web2C 7.4.5)
(./doc.tex
LaTeX2e <2001/06/01>
Babel <v3.7h> and hyphenation patterns for american, french, german, ngerman,
nohyphenation, loaded.
(/usr/local/teTeX/share/texmf/tex/latex/base/article.cls
Document Class: article 2001/04/21 v1.4e Standard LaTeX document class
(/usr/local/teTeX/share/texmf/tex/latex/base/size10.clo)) (./doc.aux) [1]
(./doc.aux) )
Output written on doc.dvi (1 page, 216 bytes).
Transcript written on doc.log.
```

Si el documento contiene elementos dinámicos como por ejemplo una tabla de contenidos, referencias internas, etcétera, LaTeX avisa cuando es necesario compilar de nuevo para que queden bien puestos todos los elementos:

```
> latex doc
This is TeX, Version 3.14159 (Web2C 7.4.5)
(./doc.tex
LaTeX2e <2001/06/01>
Babel <v3.7h> and hyphenation patterns for american, french, german, ngerman,
nohyphenation, loaded.
(/usr/local/teTeX/share/texmf/tex/latex/base/article.cls
Document Class: article 2001/04/21 v1.4e Standard LaTeX document class
(/usr/local/teTeX/share/texmf/tex/latex/base/size10.clo)) (./doc.aux)

LaTeX Warning: Reference ‘hola’ on page 1 undefined on input line 8.

[1] (./doc.aux)

LaTeX Warning: There were undefined references.


LaTeX Warning: Label(s) may have changed. Rerun to get cross-references right.

 )
```

```
Output written on doc.dvi (1 page, 276 bytes).
Transcript written on doc.log.
>
```

Después de repetir, cuando ya no aparece lo de "Label(s) may have changed", se puede ver el documento que resulta con la instrucción `xdvi doc.dvi`. Para generar un documento en formato PostScript, se ejecuta `dvips -o doc.ps doc.dvi`:

```
> dvips -o doc.ps doc.dvi
This is dvips(k) 5.92b Copyright 2002 Radical Eye Software (www.radicaleye.com)
' TeX output 2007.03.13:1216' -> doc.ps
<texc.pro><f7b6d320.enc><texps.pro>. <cmbx10.pfb><cmr10.pfb>[1]
>
```

Para ver el archivo `doc.ps`, se utiliza `gv doc.ps`. Para convertirlo en PDF, se ejecuta `ps2pdf doc.ps doc.pdf` y el resultado se puede ver con `acroread doc.pdf` u otro programa para abrir documentos tipo PDF.

4.6.1. Estructura básica

Podemos iniciar con el siguiente documento.

```
\documentclass{article}
\begin{document}

Hola mundo.

\end{document}
```

Este es el documento mínimo, a partir de ahora lo iremos sofisticando tanto como podamos.

La estructura general de un documento en LaTeX es:

```
\documentclass[opciones]{clase}
Preámbulo
\begin{document}
Documento
\end{document}
```

El parámetro *clase* en la primera línea indica el tipo de documento que se creará. Puede elegirse, por ejemplo, de entre las siguientes clases:

`article`: Si lo que se desea es escribir un artículo o un informe pequeño. Ideal para los trabajos escolares.

`book`: Si lo que se desea es escribir un libro. Por ejemplo, una tesis.

`report`: Esta es una clase intermedia entre un artículo y un libro. Como su nombre lo sugiere, es ideal para informes más grandes.

`slide`: Para presentaciones con diapositivas.

También puede crear su propia clase. De esto no hablaremos en este documento.

El parámetro *opciones* sirve para añadir algunas especificaciones al documento. Las opciones, de ser más de una, deberán separarse por comas. En caso de no especificar ninguna, en cuyo caso pueden omitirse incluso los corchetes, el documento tomará las opciones por defecto. Algunas de las opciones más útiles son:

`10pt, 11pt, 12pt`: Establece el tamaño básico del texto. La opción por defecto es 10pt.

`a4paper, letterpaper`: Especifica el tamaño de papel. También se pueden elegir `a5paper`, `b5paper`, `executivepaper` o `legalpaper`. La opción por defecto es `letterpaper`.

`titlepage, notitlepage`: Indica si debe crearse una página con el título del documento o no. La opción por defecto es **no** en `article`, y **sí** en `book` y `report`.

`twocolumn`: Crea el documento en dos columnas.

`twoside, oneside`: Especifica si el documento será a dos caras o a una sola. La opción por defecto es **una** para `article` y `report`, y **dos** para `book`.

`openright, openany`: Indica si los capítulos inician en una página derecha (dejando una página en blanco si es necesario), o en la siguiente página. La opción por defecto es **siguiente página derecha** para `book`, **siguiente página** para `report`. La opción no está disponible para `article`, pues en dicha clase no hay capítulos.

En LaTeX se pueden añadir algunos *paquetes*, con los cuales, disponemos de nuevas utilidades. Para agregar un paquete debe colocarse en el preámbulo la instrucción

`\usepackage[`*opciones*`]{`*paquete*`}`

Existe un sin número de paquetes, cada uno con sus opciones específicas. La documentación de cada paquete es distribuida junto con él. Aquí sólo mencionaremos, para poner un ejemplo útil, el paquete `babel`.

Paquete `babel`: El idioma por defecto de LaTeX es el inglés. Para el uso de otros idiomas debe cargarse el paquete `babel`. Para el idioma español la opción es `spanish`.

4.6.2. Escritura básica

Las *instrucciones* de LaTeX siempre inician con una *barra invertida* (\). Es importante recordar que LaTeX es sensible a mayúsculas y minúsculas, por lo que ha de tenerse cuidado en cómo se escriben las instrucciones.

Una orden en LaTeX siempre consiste de una barra invertida seguida de un nombre formado sólo por letras, o de una barra invertida seguida por un carácter especial.

LaTeX ignorará los espacios en blanco que se encuentren tras una instrucción, por lo que si se desea agregar un espacio en blanco ha de indicarse un argumento vacío a la instrucción colocando un `{}` al final de la orden.

Uno o más *espacios* en el código crean un sólo espacio en el documento. Un salto de linea en el código crea un espacio en el texto. En el siguiente ejemplo, y en adelante, el símbolo ␣ denota un espacio en blanco.

```
El␣siguiente␣␣␣texto␣␣que
se␣ha␣␣␣␣␣␣␣␣escrito␣de␣este␣modo,
␣␣␣se␣ ve␣␣␣␣␣de␣␣este␣␣␣␣␣␣otro.
```

El siguiente texto que se ha escrito de este modo, se ve de este otro.

Dos o más saltos de linea en el código crean un solo salto de linea en el texto. Si se desean más espacios, horizontal o vertical, vea la sección correspondiente más adelante.

Cuando LaTeX encuentra un `%` en el texto toma el resto de la línea como *comentario*, por lo que será ignorado en la compilación del documento.

```
Esto es un % comentario
% y más comentario
ejemplo.
```

Cuyo resultado es:

Esto es un ejemplo.

El signo `%` también puede ser útil para partir líneas demasiado largas en las que, por alguna razón, no se desea o no se puede partir en un espacio en blanco.

```
Esto es ot%
ro ejemplo.
```

Cuyo resultado es:

Esto es otro ejemplo.

Hay ciertos signos que LaTeX *reserva* para usos especiales. Estos caracteres especiales son

```
\ { } % # $ & _ ^ ~
```

Si se desea obtener dichos signos han de usarse las siguientes instrucciones:

```
$\backslash$ \{ \} \% \# \$ \& \_ \^{} \~{}
```

¿Para qué están reservados? Este documento aclara el uso de algunos de ellos, el uso del resto será aclarado conforme vaya avanzando en el uso del LaTeX.

4.6.3. Caracteres especiales

LaTeX utiliza código ASCII, por lo que algunos caracteres latinos de uso común no pueden, por defecto, ser ingresados directamente desde el teclado.

Para las *comillas* angulares, « y », deben usarse las órdenes

```
\guillemotleft y \guillemotright
```

Para las comillas voladas dobles “ y ” debe usarse `‘‘` y `’’`; y en el caso de que se deseen las comillas simples ‘ y ’ debe usarse `‘` y `’` . En todo caso, nunca deben usarse para este propósito las comillas que aparecen en el teclado (salvo que estén editando en `emacs`, que lo interpreta bien en el modo LaTeX).

Aquí ’ es el apostrofo y ‘ el acento grave. Cabe recordar que en español deben utilizarse en primera instancia las comillas angulares « y », reservando los otros tipos para cuando deban entrecomillarse partes de un texto ya entrecomillado: «Antonio me dijo: “Vaya ‘cacharro’ que se ha comprado Julián”».

Si se tiene cargado el paquete `babel`, las comillas pueden escribirse con << y >>.

En el caso de los *acentos* para las vocales, por ejemplo, para obtener é deberá emplearse `\’e`. Análogamente el resto, a excepción de la letra i, la cual exige trato especial, pues para evitar obtener algo como í deberá usarse `\´{\i}`, donde `\i` obtiene el símbolo ı, una i sin punto. El siguiente cuadro muestra cómo obtener otros símbolos que podría necesitar al escribir.

Acentos y caracteres especiales							
¡	!‘	¿	?‘	ñ	\~n	ü	\"u
ó	\’o	ò	\‘o	ô	\^o	ö	\"o
õ	\~o	ő	\H o	ŏ	\u o	ǒ	\v o
ȯ	\.o	ọ	\d o	ǫ	\c o	ç	\c c
ō	\=o	o̲	\b o	o͡o	\t oo	ß	\ss
æ	\ae	Æ	\AE	œ	\oe	Œ	\OE
å	\aa	Å	\AA	ø	\o	Ø	\O
ł	\l	Ł	\L	ı	\i		

Cuando se tiene cargado el paquete `babel`, si se ha definido la opción `spanish` entonces los acentos pueden ingresarse simplemente por `'e`, incluso para í (`'i`), ya que el paquete los sustituirá por defecto. Análogamente `~n` para ñ y `"u` para ü.

Si además se tiene cargado el paquete `inputenc` con la opción `latin1`, todos los caracteres anteriores podrán ser ingresados directamente del teclado.

Para agregar *puntos suspensivos* al texto se sugiere no usar tres puntos, sino la instrucción `\dots`. Observe la diferencia:

```
No así... sino así\dots
No así... sino así…
```

Para una lista completa de los símbolos disponibles, véase

`http://www.ctan.org/tex-archive/info/symbols/comprehensive/symbols-a4.pdf`

4.6.4. Guiones y guionado

LaTeX utiliza cuatro tipos de guiones para escritura: El guión simple (`-`) se emplea para composición de palabras; el guión largo (`--`) se usa entre dos números para declarar un rango; la raya ortográfica (`---`); y un guión de partición de palabras (`\-`) del que hablaremos adelante. A continuación un ejemplo de uso de los tres primeros.

```
físico-químico                    físico-químico
páginas 4--12                     páginas 4–12
Sí ---dijo él---, por supuesto.   Sí —dijo él—, por supuesto.
```

LaTeX parte las palabras cuando es necesario al final de un renglón. Por defecto, para hacer la partición, suele emplear reglas inglesas de *silabeo*. Afortunadamente, uno puede marcar su propio silabeo. Existen, cuando menos, dos formas de hacer esto.

Edición directa: Si una palabra ha sido mal partida por LaTeX, uno puede marcarle dónde partirla mediante un `\-` en el lugar en que uno desea la partición. Si se indican más de un lugar de partición, LaTeX elegirá el que quede mejor. Por ejemplo:

`electro\-óptico`, `elec\-tro\-óp\-tico`.

Forma global: Si la palabra en cuestión es usada frecuentemente se convierte en una potencial causa de problemas. En este caso es mejor definir su silabeo en el preámbulo mediante la instrucción `\hyphenation`, de esta manera, en todo el documento se respetará la forma indicada. Por ejemplo: `\hyphenation{elec-tro-óp-tico}`.

Por último, se recomienda fuertemente activar la opción `spanish` en el archivo `language.dat` (no se dirá aquí cómo hacer esto), y entonces LaTeX empleará las reglas de silabeo en castellano. Sin embargo, a pesar de que el español es un lenguaje muy regular, existen algunas excepciones que suelen crear problemas, por lo que no será extraño que algunas palabras se resistan. Para tales palabras haga lo anterior.

4.6.5. Documentos estructurados

Para colocar un *encabezado* (título, autor, etc.) a nuestro documento basta incluir la siguiente línea en el cuerpo del documento, es decir, después del `\begin{document}`.

```
\maketitle
```

Pero para ello es importante haber declarado los campos *título*, *autor* y, opcionalmente, la *fecha*, como se indica a continuación.

Para especificar el *título* del documento debe incluirse la siguiente línea en el preámbulo.

```
\title{Aquí va el título del documento}
```

Para especificar el *autor* del documento debe incluirse la siguiente línea en el preámbulo.

```
\author{Autor}
```

En caso de que el documento tenga más de un autor, la forma de separarlos es con la palabra reservada `\and`, la cual debe separar cada autor aunque sean más de dos.

```
\author{Autor 1 \and Autor 2 \and Autor 3}
```

La *fecha* es un parámetro opcional. En caso de que se desee especificar una fecha en el encabezado del documento deberá incluirse la siguiente línea en el preámbulo.

```
\date{Fecha a imprimir}
```

En caso de no poner esta línea, LaTeX colocará por defecto la fecha en que se compile el documento. Si lo que se desea es no colocar una fecha en el encabezado del documento basta escribir la instrucción pero dejando vacío el parámetro de entrada de la función, es decir `\date{}`.

Cuando se escribe un artículo (clase `article`) es común que el documento necesite un *resumen*. Para agregar un resumen, basta colocar el texto en el entorno `abstract` como se muestra a continuación.

```
\begin{abstract}
  Aquí debe escribirse el resumen.
\end{abstract}
```

Secciones

Una de las mayores fortalezas de LaTeX es la facilidad con que pueden crearse documentos estructurados. LaTeX maneja las siguientes opciones por defecto para definir la jerarquía de las partes del documento.

```
\part{...}
\chapter{...}
\section{...}
\subsection{...}
\subsubsection{...}
\paragraph{...}
\subparagraph{...}
```

Los puntos suspensivos deben reemplazarse por el nombre que llevará la sección. Las secciones `part` y `chapter` no pueden emplearse en la clase `article`, como era de esperarse.

Si deseamos que una de estas partes no aparezca numerada y por tanto no aparezca en el índice, basta agregar un $*$. Por ejemplo, supongamos que no se desea que cierto capítulo esté numerado, entonces debe escribirse `\chapter*{...}`. Análogamente para los demás.

El efecto de `paragraph` y `subparagraph` es idéntico a la vista, lo que cambia es la jerarquía con que son tratados. Por ejemplo, si colocamos en el preámbulo la línea

```
\setcounter{tocdepth}{5}
```

los párrafos y los sub-párrafos serán listados en el índice y se sangrarán según su jerarquía.

4.6.6. Índices

El *índice de contenido* o índice general se obtiene incluyendo la siguiente instrucción donde se desee que aparezca el índice.

```
\tableofcontents
```

En el índice general aparecerán en forma estructurada todas las secciones que hayan sido definidas según la sección anterior.

En caso de que se desee, se puede agregar un *índice de las figuras* usadas en el documento. Para ello basta colocar la siguiente instrucción donde se desee que aparezca el índice.

```
\listoffigures
```

También se puede agregar un *índice de las cuadros o tablas*. Para ello basta colocar la siguiente instrucción donde se desee que aparezca tal índice, y todos los objetos indicados dentro de un entorno

table serán enlistadas.

```
\listoftables
```

4.6.7. Referencias cruzadas

Para hacer referencias cruzadas, LaTeX dispone de instrucciones muy intuitivas. Primero debe colocarse una etiqueta al punto de referencia

```
\label{etiqueta}
```

Donde etiqueta es una palabra clave asignada por el usuario. Ahora que tenemos el punto de referencia sólo basta citarla

```
En la sección \ref{etiqueta} trataremos...
```

Obteniendo por resultado

```
En la sección 4.1 trataremos...
```

Se supone que 4.1 es la sección activa en que se encuentra el punto de referencia. Como es de esperarse, en caso de modificar el documento, todas las referencias son actualizadas.

En realidad el número arrojado por la instrucción \ref es el del capítulo, sección, subsección, figura, cuadro, teorema, o cualquier otro entorno que conlleve numeración y en el cual se encuentre la instrucción \label.

También puede usarse la instrucción \pageref{etiqueta} con la cual, como su nombre sugiere, lo que se obtiene es la página en que se encuentra el punto de referencia.

4.6.8. Subdocumentos

En caso de estar elaborando un proyecto largo, digamos un libro, podemos crear subdocumentos, digamos uno para cada capítulo, e incluirlos en un documento maestro con la instrucción

```
\include{subdocumento}
```

Donde subdocumento es el nombre del documento .tex que se desea incluir en ese lugar del documento maestro.

Otra instrucción de gran ayuda cuando se trabaja con subdocumentos es

```
\includeonly{subdocumento1,subdocumento2,...}
```

La cual debe colocarse en el preámbulo, y cuyo efecto es el de incluir en el trabajo sólo aquellos subdocumentos citados con \include que se encuentren entre los argumentos del \includeonly.

4.6.9. Notas al pie de página

La instrucción `\footnote{}` imprimirá el texto que sea colocado entre las llaves en una nota al pie de la página. Por ejemplo:

> El buen cristiano debe estar precavido frente a los matemáticos y todos aquellos que hacen profecías vacías. Existe el peligro de que los matemáticos hayan hecho un pacto con el diablo para oscurecer el espíritu y confinar al hombre en el infierno.[1]
>
> San Agustín, De genesi ad Litteram, libro II, xviii, 37

La nota al pie de página ha sido obtenido con:

```
... en el infierno.\footnote{Conviene aclarar que ...}
```

4.6.10. Mejorando el entorno

LaTeX tiene definidos un grupo de *tamaños* para el texto.

Tamaños	
tiny	`\tiny`
scriptsize	`\scriptsize`
footnotesize	`\footnotesize`
small	`\small`
normalsize	`\normalsize`
large	`\large`
Large	`\Large`
LARGE	`\LARGE`
huge	`\huge`
Huge	`\Huge`

Para poner un texto en determinado tamaño debe hacerse, por ejemplo, mediante `{\tiny pequeña}`. Si no se coloca entre llaves el efecto continuará hasta el fin del documento o hasta que se especifique otro tamaño. El tamaño `normalsize` es la opción por defecto, la instrucción sólo es necesaria en caso de que se desee volver al tamaño normal tras haber definido otro tamaño.

También puede cambiarse la letra a cualquier tamaño que defina el usuario mediante la orden

```
\fontsize{12}{12}\selectfont
```

Donde, el primer número es el tamaño de la letra en puntos y el segundo es la altura que ha de darse a la línea en puntos.

[1] Conviene aclarar que San Agustín llama matemáticos a los astrólogos.

La ventaja de los tamaños predefinidos (lista anterior), es que su medida se calcula de acuerdo al tamaño de letra definida para el documento, de tal manera que se conserve la jerarquía en los tamaños.

LaTeX emplea los *tipos de letras* CMR (*Computer Modern Roman*) creadas por Knuth. Éstas se pueden clasificar en cuatro *formas* (shape), tres *familias* (family) y dos *series* (serie).

Formas	
Recta	up
Itálica	it
Inclinada	sl
VERSALITAS	sc

Familias	
Roman	rm
Sans Serif	sf
`Typewriter`	tt

Series	
Medio	md
Negrita	bf

Para poner un texto en determinado tipo, digamos inclinada, debe hacerse mediante una instrucción `{\sl texto}`. Si no se coloca entre llaves el efecto continuará hasta el fin del documento o hasta que se especifique otro tipo. También puede hacerse en una forma menos breve, `\textsl{texto}`. Obviamente, para cualquier otro tipo basta reemplazar el `sl` por el correspondiente.

Las opciones por defecto son **recta**, **roman** y **medio**, y no es necesario especificarlas salvo cuando se desea volver a ellas tras haber declarado otra.

Cabe señalar que la forma breve tiene el inconveniente de que no hace combinaciones, por ejemplo:

```
{\sf Esto debería ser {\bf Sans Serif y negrita} y no lo es.}
```

Esto debería ser **Sans Serif y negrita** y no lo es.

Para conseguir ese efecto debemos recurrir al método menos abreviado:

```
\textsf{Esto debe ser \textbf{Sans Serif y negrita} y lo es.}
```

O incluso éste aún menos breve:

```
\sffamily{Esto debe ser {\bfseries Sans Serif y negrita} y lo es.}
```

Ambos ofrecen el siguiente resultado:

Esto debe ser **Sans Serif y negrita** y lo es.

También se puede especificar un tipo de letra dentro de un entorno, lo cual será de utilidad en caso de que el texto sea más largo, por ejemplo, que abarque varias líneas o párrafos.

```
\begin{itshape}
  Texto largo.
\end{itshape}
```

Para llamar la atención sobre un texto puede cambiarse el tipo o el tamaño de letra para el texto que se quiere *resaltar*, tal como se ha indicado antes. Sin embargo, LaTeX dispone de una mejor forma de hacer esto. Cuando se desea resaltar o destacar un texto se puede usar la instrucción **enfatizar** que se emplea mediante `\em` o `\emph{}`. Por ejemplo, cualquiera de las siguientes líneas

```
Este es un {\em texto resaltado} en una oración.

Este es un \emph{texto resaltado} en una oración.
```

obtiene el siguiente resultado.

Este es un *texto resaltado* en una oración.

No debe pensarse que resaltar un texto es equivalente a ponerlo en cursiva, pues en realidad LaTeX elegirá el tipo adecuado para que el texto resalte, por ejemplo, en un texto en cursiva el efecto sería como sigue.

```
{\it Este es otro {\em texto resaltado} en una oración.}
```
Este es otro texto resaltado *en una oración.*

Así que no debe abusarse de esto, si lo que se desea es colocar un texto en cursiva use `it`, pero si lo que se desea es resaltar use `em`.

Subrayar no es usual en un texto impreso, y en realidad se sugiere siempre enfatizar. Aún con ello, si se desea subrayar un texto, puede hacerse con la siguiente instrucción.

```
Este es un \underline{texto} subrayado.
```
Este es un texto subrayado.

El texto que se encuentre dentro de un entorno `verbatim` se respetará tal como ha sido escrito. Este entorno es ideal para escribir código, pues el texto no será compilado por LaTeX. Por ejemplo, el código escrito:

```
\begin{verbatim}
 Todo lo que sea escrito aquí será respetado
 y no será compilado por lo que los %comentarios
 y las {\bf instrucciones} serán ignoradas.
\end{verbatim}
```

Produce:

```
Todo lo que sea escrito aquí será respetado
```

```
y no será compilado por lo que los %comentarios
y las {\bf instrucciones} serán ignoradas.
```

El entorno `verbatim*` marca, además, los espacio en blanco:

```
Todo␣lo␣que␣sea␣escrito␣aquí␣será␣respetado
y␣los␣espacios␣en␣blanco␣␣␣␣serán␣marcados.
```

También se dispone de una versión para crear el mismo efecto dentro de un párrafo, lo cual se consigue con `\verb+texto+`. El signo + funge de símbolo delimitador. Se puede usar cualquier carácter excepto las letras, * o caracteres en blanco. También está disponible la instrucción `verb` en su versión con asterisco.

Como ya se ha podido ver, un texto en edición directa es colocado en tipo de letra `Typewriter`. No ha de usarse este entorno para colocar texto en ese tipo de letra, si lo que se desea es colocar texto en `Typewriter`, use un entorno `tt`.

4.6.11. Espacios horizontales y verticales

Como ya se mencionó, espacios dados con la barra espaciadora o con líneas en blanco son ignoradas por LaTeX. Si se desea dejar espacios en el texto se puede hacer alguna de las siguientes alternativas.

Un espacio extra de la barra espaciadora se obtiene mediante `\␣`. Así se pueden agregar tantos como se desee, aunque no se recomienda.

Otras formas de obtener espacios entre palabras se pueden ven el siguiente cuadro.

Este espacio.	`Este espacio.`
Este espacio.	`Este\,espacio.`
Este espacio.	`Este\ espacio.`
Este espacio.	`Este\enskip espacio.`
Este espacio.	`Este\quad espacio.`
Este espacio.	`Esta\qquad espacio.`

Espacios verticales entre párrafos pueden obtenerse con las siguientes instrucciones.

Este espacio.	`Este espacio.`
Este espacio.	`Este espacio.\smallskip`
Este espacio.	`Este espacio.\medskip`
Este espacio.	`Este espacio.\bigskip`
Este espacio.	`Este espacio.`

Además en LaTeX podemos obtener espacios de cualquier longitud mediante las instrucciones `\hspace{7mm}` para espacio horizontal y `\vspace{1.5cm}` para espacio vertical, donde, en realidad se debe indicar

la longitud deseada en las unidades que mejor convengan. El siguiente cuadro muestra algunas de las unidades empleadas por LaTeX.

Unidades de medida	
mm	milímetro
cm	centímetro = 10 mm
in	pulgada = 25.4 mm
pt	punto $\approx$ 1/3 mm
em	el ancho de una m en el tipo de letra en uso
ex	el alto de una x en el tipo de letra en uso

La instrucción `\hspace` es ignorada al principio de línea. Análogamente la instrucción `\vspace` es ignorada al inicio de página. Esto es con la intención de no dejar espacios innecesarios, sin embargo, si es el efecto que se desea, pueden emplearse las versiones con asterisco `\hspace*{}` y `\vspace*{}`.

Si se desea *iniciar una nueva página*, puede usarse la instrucción

```
\newpage
```

Si se desea bajar un renglón, pero sin empezar nuevo párrafo puede usarse `\\` o `\newline`. Por ejemplo,

Un párrafo en
dos líneas.

se obtiene con

```
Un párrafo en \\
dos líneas.
```

Si además se agrega una asterisco `\\*`, LaTeX prohíbe que se produzca un salto de página tras el salto de línea.

4.6.12. Justificado y centrado

Por defecto, LaTeX justifica el texto a ambos lados. Los entornos `flushleft` y `flushright` crean párrafos justificados a la izquierda y a la derecha respectivamente. Por su parte, el entorno `center` centra el texto.

Estos entornos se usan de la siguiente manera.

```
\begin{flushright}
 Este es un texto          \\
 justificado a la derecha, \\
 como se puede ver.        \\
\end{flushright}
```

He aquí su efecto:

Este es un texto
justificado a la derecha,
como se puede ver.

Si no se coloca el \\ para especificar el fin de línea, entonces LaTeX lo hace automáticamente cortando en renglón en el último espacio que no exceda el ancho del texto.

4.6.13. Listas

LaTeX provee de tres entornos para hacer listados. Cada elemento de una lista debe iniciarse con un `\item` solo o seguido de un objeto según el entorno que se esté empleando. Los entornos se pueden anidar indistintamente, en los ejemplos sólo se anidan de un solo tipo pero esto es para resaltar su uso.

El entorno `enumerate`

Crea una lista numerada. Observe el efecto que se crea al anidarlos:

```
\begin{enumerate}
\item Uno
\item Dos
\begin{enumerate}
\item Aquí inicia una sublista a Dos.
\item Y sigue.
\end{enumerate}
\item Tres, continúa y termina la lista original.
\end{enumerate}
```

Y el resultado es:

1. Uno
2. Dos
    a) Aquí inicia una sublista a Dos.
    b) Y sigue.
3. Tres, continúa y termina la lista original.

El entorno `itemize`

Crea una lista con viñetas. Si en el ejemplo anterior se reemplaza cada `enumerate` con un `itemize` se obtiene el siguiente efecto:

- Uno
- Dos
  - Aquí inicia una sublista a Dos.
  - Y sigue.
- Tres, continúa y termina la lista original.

El entorno `description`

Crea una lista para describir objetos. Requiere que se indiquen los nombres de los objetos a describir.

```
\begin{description}
\item[Uno] Este es el primero.
\item[Dos] Este es el segundo.
\item[más uno] Sucesivamente.
\item[Cuarenta y cuatro] Y termina.
\end{description}
```

Y el resultado es:

Uno Este es el primero.

Dos Este es el segundo.

más uno Sucesivamente.

Cuarenta y cuatro Y termina.

4.6.14. Instrucciones propias

Uno de los mayores atractivos de LaTeX es la posibilidad de crear nuestras propias instrucciones. En este documento daremos algunos ejemplos que esperamos motiven la creatividad.

La instrucción `\def` nos provee de una herramienta para definir instrucciones que serán reemplazadas. Ejemplos:

```
\def\yo{Escriba aquí su nombre}
\def\uanl{\textsc{Universidad Autónoma de Nuevo León}}

Mi nombre es \yo{}.
En la actualidad, la \uanl{} camina hacia la excelencia.
```

Cuyo resultado es:

Mi nombre es Escriba aquí su nombre.
En la actualidad, la Universidad Autónoma de Nuevo León camina hacia la excelencia.

Por lo que, si es un texto que será escrito frecuentemente, será de gran utilidad.

Para crear tus propias instrucciones, que tomen parámetros y hagan con ellos alguna acción, puede usarse la siguiente instrucción:

```
\newcommand{instrucción}[núm]{definición}
```

Donde, *instrucción* será el nombre de la nueva instrucción (debe iniciar con una barra invertida), *núm* es un número del 1 al 9, que es la cantidad de argumentos que va a requerir la instrucción, y *definición* debe describir lo que se desea haga la instrucción. En la definición, cada parámetro se invoca con el símbolo #.

Primer ejemplo:

```
\newcommand{\resalte}[1]{\emph{#1}}
Este es un \resalte{texto resaltado}.
```

Y el resultado es:

Este es un *texto resaltado*.

Claro que nadie hace esto para resaltar texto, pero sin duda da una perspectiva de lo que se puede hacer. He aquí un ejemplo más sofisticado:

```
% Define la fórmula general de segundo grado
\newcommand{\fgsg}[3]{\frac{-(#2) \pm \sqrt{#2^{2}-4(#1)(#3)}}{2(#1)}}

Observe su efecto:

$\fgsg{a}{b}{c}$
$\fgsg{4}{1}{3}$
$\fgsg{3}{-2}{1}$
```

Observe su efecto:

$$\frac{-(b)\pm\sqrt{(b)^2-4(a)(c)}}{2(a)}$$
$$\frac{-(1)\pm\sqrt{(1)^2-4(4)(3)}}{2(4)}$$
$$\frac{-(-2)\pm\sqrt{(-2)^2-4(3)(1)}}{2(3)}$$

Como la instrucción requiere tres parámetros, cada uno se coloca entre llaves independientes. Los signos $ son necesarios para entrar al entorno matemático y que las formulas se produzcan correctamente, pero de eso se hablará en otra ocasión.

4.6.15. Cuadros y figuras

LaTeX permite la creación de tablas mediante el ambiente `tabular`, el cual tiene la siguiente sintaxis:

```
\begin{tabular}{posiciones}
  cuarpo de la tabla
\end{tabular}
```

donde `posiciones` corresponde a las indicaciones para la colocación de las columnas dentro del espacio destinado para cada una de ellas, puede tomar las roguinetes opciones

r $\rightarrow$ centrado a la derecha
l $\leftarrow$ centrado a la izquierda
c $\leftrightarrow$ centrado al centro

Dentro del cuerpo de la tabla se usa el carácter & para indicar la separación entre columnas, es decir, si se tienen n columnas se tendrán $n-1$ &'s. También se usa $\backslash\backslash$ para indicar el termino de un renglón, cuando se trata del ultimo renglón $\backslash\backslash$ se omiten.

Ejemplo

	Agricultura	Manufactura
Agricultura	0.293	0
Manufacturas	0.014	0.207
Energía	0.044	0.010
Pesca	0.234	0

La tabla anterior fue producida por

```
\[
\begin{tabular}{lcl}
            & Agricultura & Manufactura\\
Agricultura  & 0.293       & 0          \\
Manufacturas & 0.014       & 0.207      \\
Energía      & 0.044       & 0.010      \\
Pesca        & 0.234       & 0
\end{tabular}
\]
```

LaTeXpermite la colocación de líneas horizontales y verticales que abarque toda o sólo una parte de la tabla.

Esto se hace mediante las siguientes instrucciones:

\hline	→	produce una linea horizontal del tamaño de la table
\cline{j-k} con $j \leq k$	→	produce una línea horizontal de la columna j a la k

Para obtener linea verticales se debe indicar esto a la derecha y/o izquierda de cada posición, por ejemplo:

	Agricultura	Manufactura
Agricultura	0.293	0
Manufacturas	0.014	0.207
Energía	0.044	0.010
Pesca	0.234	0

fue producida por

```
\begin{center}
\begin{tabular}{|l|c|l|} \hline
             & Agricultura & Manufactura\\ \hline
Agricultura  & 0.293       & 0          \\
Manufacturas & 0.014       & 0.207      \\
Energía      & 0.044       & 0.010      \\
Pesca        & 0.234       & 0          \\ \hline
\end{tabular}
\end{center}
```

Otro ejemplo:

Fuente de variación	gl	SC	CM	Valor F
Tratamientos	3	0.004695	0.001565	20.87
Error	16	0.0012	0.000075	
Total	19	0.005895		

que fué producida por

```
\begin{center}
\begin{tabular}{lrlcr}\hline
Fuente de    &    &          &          &         \\
variación    & gl &      SC  &    CM    & Valor F \\ \hline
Tratamientos & 3  & 0.004695 & 0.001565 & 20.87   \\
Error        & 16 & 0.0012   & 0.000075 &         \\ \cline{2-5}
Total        & 19 & 0.005895 &          &         \\ \hline
\end{tabular}
\end{center}
```

LaTeX permite agrupar en una tabla dos o más columnas, esto se hace mediante `multicolumn{num. col.}{posición}{col.}`, donde `num. col.` es el número de columnas a agrupar, en `posición` se indica la posición, debe ser sólo una, `col.` se refiere al texto que aparecerá en la única columna existente, por esta razón es que solo se pone una posición.

Para entender mejor el funcionamiento de éste, veamos el siguiente ejemplo:

Ritmo de Crecimiento de la Industria Manufacturera

	Decenio anterior		
País	Decenio	2o. quinquenio	1970
hline Argentina	5.6	5.0	4.4
Brasil	7.0	10.4	11.0
Chile	5.5	3.6	1.3
Guatemala	7.6	8.2	3.5
México	9.1	8.8	8.7

La tabla anterios fué producida por

```
\begin{center}
{\footnotesize Ritmo de Crecimiento de la Industria Manufacturera }\\
\begin{tabular}{|l|c|c|c|} \hline \hline
         & \multicolumn{2}{c|}{Decenio anterior} &   \\
País     & Decenio & 2o. quinquenio              & 1970 \\ hline
Argentina & 5.6     & 5.0                        & 4.4 \\ \hline
Brasil    &  7.0    & 10.4                       &  11.0  \\ \hline
Chile     &  5.5    & 3.6                        &   1.3  \\ \hline
Guatemala & 7.6     & 8.2                        &   3.5   \\ \hline
México    &  9.1    & 8.8                        &  8.7 \\ \hline \hline
\end{tabular}
\end{center}
```

Para crear un cuadro enumerado, se usa

```
\begin{table}[ht!]
\caption{Aquí va la explicación.}
\label{tabla:id}

... aquí va el contenido, que típicamente es un ambiente tabular

\end{table}
```

donde lo de `ht` es la posición preferida (`t` = arriba, `b` =abajo, `h`=aquí mismo, etcétera), `\caption` define la descripción que aparece encima del cuadro (y si se lo coloca justo antes de `\end{table}`, debajo del cuadro), y `\label` se usa para referencias cruzadas.

Similarmente, para crear una figura enumerada, existe

```
\begin{figure}[ht!]
```

```
... aquí va el contenido, que típicamente es un archivo tipo .eps

\caption{Aquí va la explicación.}
\label{fig:id}
\end{figure}
```

Para incluir una gráfica, el formato recomendable es *Encapsulated PostScript* (`.eps`), generado nativamente por `xfig` (sección 5.1.1) y creado por conversión en Gimp (sección 5.1.2).

Con `\usepackage{graphicx}` podemos usar la instrucción

```
\includegraphics[width=65mm]{dibujo.eps}
```

y alternativamente con `\usepackage{epsfig}` funciona

```
\epsfig{file=dibujo.eps, width=65mm}
```

donde `dibujo.eps` es el nombre del archivo de la gráfica y `65mm` es el ancho el millimetros para la gráfica. Es común poner la instrucción de incluir una imagen dentro de un `\centerline{...}` para que sea justificada al centro de la página.

4.6.16. Fórmulas matemáticas

El *modo matematico* es indispensable para la creación de fórmulas matemáticas:

- Los espacios son ignorados. Para insertar espacios, para insertar espacios donde el usuario crea conveniente, LaTeX tiene una serie de instrucciones de control.
- Los caracteres alfabéticos se encuentran en tipo math–italic, que es, por su espaciado ligeramente diferente al tipo \it.
- Muchas secuencias de control, como por ejemplo la que produce las letras Griegas, solamente trabajan en modo matemático.
- No se puede empezar un parágrafo adentro de una fórmula.

Los delimitadores del modo matemático son:

$ . . . $	para fórmulas cortas dentro de un renglón.
\[. . . \]	para fórmulas centradas.

Existen ambientes definidos para modo matemático, éstos se verán más adelante.

En las siguientes secciones se presentan instrucciones que deben manejarse en modo matemático.

Subíndices y superindices

Subíndice	_
Superindice	^

***Ejemplo* 1** *Si queremos obtener*

$$x_2, y^1$$

escribimos

```
\[x_2,\, y^1\]
```

Las instrucciones _ y ^ pueden ser combinados

***Ejemplo* 2** *Cuando son combinados, no es necesario un orden especifico*

$$x_1^2, y_1^x$$

producido mediante

```
\[x_1^2, y^x_1\]
```

cuando el subíndice o superíndice tiene mas de dos elementos se usan {…} así, para obtener

$$x_{11}^2, y_1^{x^2}$$

escribimos

```
\[x_{11}^2, y^{x^2}_1\]
```

Fracciones

Se pueden producir fórmulas como x/y y desplegarlas así x/y, pero LaTeX para el caso de fracciones largas utiliza

\frac{numerador}{denominador}

***Ejemplo* 3**

$$x = \frac{y + z/2}{y^2 + 1} = \frac{z + y}{1 - \frac{x}{y}}$$

lo cual fue producido por

```
\[
x = \frac{y + z/2}{y^2 +1} = \frac{z + y}{1 - \frac{x}{y}}
\]
```

Ésta puede ser usado también dentro del texto para producir fracciones como $\frac{x}{2}$, por medio de $\frac{x}{2}$.

Raices

En LaTeX por medio de \sqrt{parámetro} se produce la raiz cuadrada de parámetro; sin embargo para producir la raiz n-ésima se usa \sqrt[n]{parámetro}.

***Ejemplo* 4**

$$\sqrt{x^3 + 1}, \sqrt[4]{y^2 + 2}$$

fué producido por

```
\[
\sqrt{x^3 + 1}, \sqrt[4]{y^2 + 2}
\]
```

Al igual que las fracciones sed pueden incluir dentro del texto, sólo que debe llevar $ ... $.

Sumatorias e Integrales

En LaTeX las sumatorias se producen con \sum_{ subíndice}^{superíndice}, donde los límites pueden omitirse. Segun sea el delimitador del modo matemático que se ponga el desplegado varía

***Ejemplo* 5**

$$\sum_{i=1}^{n} x$$

es producido por

```
\[
\sum_{i=1}^{n} x
\]
```

ó si es dentro de una linea de texto como ésta, $\sum_{i=1}^{n} x$ *es obtenida por*

```
$\sum_{i=1}^{n} x$
```

Para el caso de las integrales, al igual que las sumatorias se ven diferentes cuando estan dentro de una linea de texto. Se usa \int_{subíndice}{superíndice}.

***Ejemplo* 6**

$$\int_a^b f(x)dx$$

obtenido por

```
\[
\int_{a}^{b} f(x) dx
\]
```

Es importante señalar que la manera de producir dentro de una linea la sumatoria se puede modificar usando \limits

***Ejemplo* 7** *...* $\sum_{i=1}^{n} x$ *no se ve tan bien como como* $\sum\limits_{i=1}^{n} x$*, la cual se obtiene escribiendo*

```
$\sum \limits_{i=1}^{n} x$
```

Límites anidados

En LaTeX podemos tener mas de un subíndice o supraíndice en una sumatoria, para ésto se usa \atop

***Ejemplo* 8**

$$\sum_{\substack{0\leq i\leq m\\ 0\leq j\leq n}} M(i,j)$$

obtenido mediante

```
\[
\sum_{0 \leq i \leq m \atop 0 \leq j \leq n} M(i,j)
\]
```

Si queremos que los limites se vean mejor se usa \scriptstyle

***Ejemplo* 9**

$$\sum\limits_{\substack{0\leq i\leq m\\ 0\leq j\leq n}} M(i,j)$$

para ésto solo añadimos \scriptstyle como se muestra

```
\[
\sum_{\scriptstyle 0 \leq i \leq m \atop
      \scriptstyle 0 \leq j \leq n} M(i,j)
\]
```

Arreglos

Para hacer un arreglo en modo matemático se usa el ambiente

\begin{array}{argumentos}

...

\end{array}

donde en argumentos se especifica el numero de columnas y alineacion de cada una, ésto se hace usando las siguinetes letras

- **l** (left): para alinear a la izquierda; es decir, dentro del espacio que tiene reservado, la columna, se corre hacia la derecha.
- **r** (right): para alinear a la derecha; es decir, dentro del espacio que tiene reservado, la columna, se corre a la derecha.
- **c** (center): para centrar; es decir, dentro del espacio que tiene reservado, la columna, se centra.

Para especificar el fin de una columna y el inicio de otra, se pone &. Si se trata de la ultima columna no se pone. Asi, habra una & menos con respecto al número de columnas en cada renglon.

Si hay más de un renglón,se pone \\ al final de cada uno, y cuando es el ultimo renglón no se pone nada.

Como el ambiente es de modo matemático, deben de usarse los delimitadores $...$ ó [...] .

***Ejemplo* 10**

$$\begin{array}{lcr}
(\alpha \times \beta) + \eta & \xi + \varphi & \sigma - \rho \\
\alpha + \eta & \varphi + \zeta & \sigma \\
\beta & \varphi + \zeta \times \gamma & \sigma\rho\lambda
\end{array}$$

que fué producido por

```
\[
\begin{array}{lcr}
(\alpha \times \beta) + \eta & \xi + \varphi & \sigma - \rho\\
```

```
\alpha + \eta & \varphi + \zeta & \sigma\\
\beta &\varphi + \zeta \times \gamma & \sigma \rho \lambda
 \end{array}
\]
```

$$\begin{array}{ccccc}
H_2O_{(s)} & + & 6{,}0KJ & \rightarrow & H_2O_{(l)} \\
H_2O & \rightarrow & H_{2(s)} & + & 6{,}0KJ \\
Na_{(s)} & + & 2{,}6KJ & \rightarrow & Na_{(l)}
\end{array}$$

producido por

```
\[
\begin{array}{lclcl}
H_2 O_{(s)} & +          & 6.0KJ    & \rightarrow & H_2 O_{(l)}\\
H_2 O       & \rightarrow & H_{2(s)} & +          & 6.0KJ\\
Na_{(s)}    & +          & 2.6KJ    & \rightarrow & Na_{(l)}\\
\end{array}
\]
```

Delimitadores

Existen expresiones matemáticas encerradas por *delimitadores*, como paréntesis, corchetes, llaves, etc., para los cuales el tamaño se ajusta a la expresión, LaTeX cuenta con los coamndos

$$\backslash\text{left} \qquad \backslash\text{right}$$

delimitadores izquierdo y derecho respectivamente.

Matrices

***Ejemplo* 11**

$$\left(\begin{array}{ccccc}
H_2O_{(s)} & + & 6{,}0KJ & \rightarrow & H_2O_{(l)} \\
H_2O & \rightarrow & H_{2(s)} & + & 6{,}0KJ \\
Na_{(s)} & + & 2{,}6KJ & \rightarrow & Na_{(l)}
\end{array}\right)$$

producido por

```
\[
\left(
\begin{array}{lclcl}
H_2 O_{(s)} & +          & 6.0KJ    & \rightarrow & H_2 O_{(l)}\\
H_2 O       & \rightarrow & H_{2(s)} & +          & 6.0KJ\\
Na_{(s)}    & +          & 2.6KJ    & \rightarrow & Na_{(l)}\\
\end{array}
\right)
\]
```

$$\left[\begin{array}{lcr} (\alpha \times \beta) + \eta & \xi + \varphi & \sigma - \rho \\ \alpha + \eta & \varphi + \zeta & \sigma \\ \beta & \varphi + \zeta \times \gamma & \sigma\rho\lambda \end{array}\right]$$

obtenido por

```
\[
\left[
\begin{array}{lcr}
(\alpha \times \beta) + \eta & \xi + \varphi & \sigma - \rho\\
\alpha + \eta & \varphi + \zeta & \sigma\\
\beta &\varphi + \zeta \times \gamma & \sigma \rho \lambda
 \end{array}
\right]
\]
```

Funciones

$$f(x) = \begin{cases} 4 & \textit{si } x \leq -4 \\ |x| & \textit{si } -4 < x < 0 \\ x+4 & \textit{si } x \geq 0 \end{cases}$$

producipo por

```
\[
f(x)=\left\{
\begin{array}{cl}
4   & \mbox{si } x\leq -4\\
|x| & \mbox{si } -4<x<0\\
x+4 & \mbox{si } x\geq 0
\end{array}
     \right.
\]
```

$$f(x) = \begin{cases} 4 & \textit{si} \quad x \leq -4 \\ \frac{1}{x+4} & \textit{si} \quad -4 < x < 0 \\ x+4 & \textit{si} \quad x \geq 0 \end{cases}$$

```
\[
f(x)=\left\{
\begin{array}{ccc}
4             & \mbox{si} & x\leq -4\\
\frac{1}{x+4} & \mbox{si} &-4<x<0\\
x+4           & \mbox{si} &x\geq 0
\end{array}
     \right.
\]
```

$$v_y = \left.\frac{\partial v}{\partial y}\right|_x = -\frac{\left|\begin{array}{ccc} F_u & F_y & F_w \\ G_u & G_y & H_w \\ H_u & H_y & H_w \end{array}\right|}{\left|\begin{array}{ccc} F_u & F_v & F_w \\ G_u & G_v & H_w \\ H_u & H_v & H_W \end{array}\right|}$$

producido por

```
\[
v_y = \left. \frac{\partial v}{\partial y}\right|_x =
     -\frac{\left|\begin{array}{ccc}
                   F_u & F_y & F_w \\
                   G_u & G_y & H_w \\
                   H_u & H_y & H_w
                  \end{array}\right|}
           {\left|\begin{array}{ccc}
                   F_u & F_v & F_w \\
                   G_u & G_v & H_w \\
                   H_u & H_v & H_W
                   \end{array}\right|}
\]
```

Fórmulas Multilíneas

En matemáticas muchas veces existe la necesidad de producir en papel estructuras de varios renglones, las cuales deben ser presentadas de manera explícita y estética. Para éstas estructuras haremos uso del ambiente

```
\begin{eqnarray*}
  Fórmulas multilíneas
\end{eqnarray*},
```

que distribuye las ecuaciones en tres columnas alineadas mediante &'s, la primer columna es alineada antes del primer &, la segunda entre los &'s y la tercera es alineada después del segundo &, y cada renglón es separado por \\.

***Ejemplo* 12** *La cantidad de calor puede calcularse mediante la ecuación*

$$Q = m \times \Delta t \times c$$

donde

$$\begin{aligned} Q &= \textit{cantidad de calor} \\ m &= \textit{masa} \\ \Delta t &= \textit{cambio en temperatura} \\ c &= \textit{capacidad calorífica específica} \end{aligned}$$

que fué producido por

```
\begin{eqnarray*}
Q        & = & \mbox{cantidad de calor}\\
m        & = & \mbox{masa}\\
\Delta t & = & \mbox{cambio en temperatura}\\
c        & = & \mbox{capacidad calorífica específica}
\end{eqnarray*}
```

***Ejemplo* 13**

$$\begin{aligned} m\frac{d\vec{u}}{dt} &= -\int_s P_s\vec{S} - \int_s \tau \cdot d\vec{S}, \\ m &= \int_V \rho dV = \textit{mass}, \\ d\vec{S} &= \textit{vector surface area element} \\ \vec{N} &= \rho \cdot \widetilde{\rho + \pi} \end{aligned}$$

que fué producido por

```
\begin{eqnarray*}
m \frac{d \vec{u}}{dt} & = & -\int_s P_s \vec{S} -
                            \int_s \tau \cdot d \vec{S}, \\
                     m & = &  \int_V \rho dV =
                            \mbox{mass}, \\
             d \vec{S} & = &  \mbox{vector surface area element}\\
               \vec{N} & = &  \rho \cdot \widetilde{\rho + \pi}
\end{eqnarray*}
```

Fórmulas numeradas

Entre las ventajas que LaTeX ofrece en la producción de fórmulas, se encuentra la numeración automática de éstas, así, LaTeX le asigna un contador interno y físico a cada una de ellas. Esto se hace usando los siguinetes ambientes

```
\begin{equation}
  fórmula
\end{equation}
```

***Ejemplo* 14**

$$\int_{-\infty}^{\int_\nu^\alpha e^2} f(t)dt \tag{4.1}$$

que fué producida por

```
\begin{equation}
\int_{-\infty}^{\int_\nu^\alpha e^2} f(t) dt
\end{equation}
```

```
\begin{eqnarray}
  Fórmulas multilíneas
\end{eqnarray}
```

Este ambiente es similar al eqnarray∗, con la diferencia que se le asigna un contador a cada línea limitada por \\[2]. Es importante mencionar que el contador de una línea puede ser suprimido mediante el uso de nonumber antes de \\

***Ejemplo* 15**

$$\int_a^b f(x)dx = \lim_{\Delta x\rightarrow 0}\sum_{i=1}^{n-1}\Delta x f(a+i\Delta x) \tag{4.2}$$

$$\int_a^\infty f(x)dx = \lim_{b\rightarrow\infty}\int_a^b f(x)dx$$

$$\int_a^b f(x)d\varphi = \text{ínf}(P,\ f,\ \varphi) = \sup(P,\ f,\ \varphi) \tag{4.3}$$

ésto fué producido por

```
\begin{eqnarray}
\int_a^b{f(x)dx}       &=& \lim_{\Delta x\rightarrow 0}
                           \sum_{i=1}^{n-1} \Delta x f(a+i\Delta x) \\
\int_a^\infty{f(x)dx}  &=& \lim_{b\rightarrow \infty}
                           \int_{a}^{b}{f(x)dx} \nonumber \\
\int_a^b{f(x)d\varphi} &=& \inf (P,\: f,\: \varphi) =
                           \sup (P,\: f,\: \varphi)
\end{eqnarray}
```

Matrices y Arreglos Numerados

```
\begin{equation}
  Arreglos, Matrices
\end{equation}
```

***Ejemplo* 16**

$$\lambda E - A = \left[\begin{array}{cccc} \lambda E_1 - A_1 & & & \\ & \lambda E_2 - A_2 & & \\ & & \ddots & \\ & & & \lambda E_s - A_s \end{array}\right] \tag{4.4}$$

producido por

```
\begin{equation}
\lambda E -A = \left[ \begin{array}{cccc}
   \lambda E_1 - A_1 &                   &        & \\
                     & \lambda E_2 - A_2 &        & \\
                     &                   & \ddots & \\
                     &                   &        & \lambda E_s - A_s
                      \end{array}\right]
\end{equation}
```

[2] indica que termina una línea

***Ejemplo* 17**

$$f(x) = \begin{cases} 4 & si \quad x \leq -4 \\ \frac{1}{x+4} & si \quad -4 < x < 0 \\ x+4 & si \quad x \geq 0 \end{cases} \tag{4.5}$$

producido por

```
\begin{equation}
f(x)=\left\{
\begin{array}{ccc}
4             & \mbox{si} & x\leq -4\\
\frac{1}{x+4} & \mbox{si} & -4<x<0\\
x+4           & \mbox{si} & x\geq 0
\end{array}
     \right.
\end{equation}
```

4.6.17. Diapositivas

¿Por qué hacer diapositivas con LaTeX? A pesar de que existe software especializado para la creación de diapositivas y de que éste no es muy costoso e incluso gratuito (Impress de OpenOffice, por ejemplo) la elaboración de filminas con LaTeX tiene algunas de las siguientes ventajas:

- El ambiente LaTeX permite la elaboración de formulas matemáticas con facilidad y gran calidad.
- Con un simple lector Pdf podemos ver nuestras presentaciones en cualquier sistema operativo y sin tener que utilizar el programa creador para su vizualización.
- LaTeX nos da en general todas las ventajas de un lenguage del tipo "What You See Is What You Mean" contra los lenguages tipo "What You See Is What You Get".

Algunas de las clases de documentos (documentclass) más comunes para la elaboración de diapositivas con LaTeX son: `slides`, `seminar`, `prosper`, `beamer` y `chaksem`. A continuación daremos una explicación breve de cada una de estas clases.

`Slides`

La clase slides, viene junto con LaTeX por defecto y es la más simple de todas. Permite la elaboración de diapositivas de manera rápida y sencilla. Es ideal para cuando hay mucha prisa o cuando de verdad sea necesaria la exposición de un documento enteramente escrito en LaTeX (libro, artículo, etc.). Sin embargo, no hay mucha diferencia entre crear diapositivas con slides y ver un documento pdf en modo de presentación (lo cual se logra tecleando `Control-l`). Por esta razón, no daremos más explicacciones sobre esta clase y pasaremos a las siguientes.

Seminar

Aunque sea más apropiada para la elaboración de diapositivas que `slides`, `seminar` no deja de ser en esencia una clase "austera". La ventaja de seminar sobre el resto de las clases que veremos es que permite la elaboración de diapositivas de manera rápida y sencilla. Además, su simpleza de formato hace de `seminar` una clase ideal para la impresión de diapositivas. Para mayor información sobre esta clase consulta `http://www.tug.org/applications/Seminar/`.

A continuación mostramos un ejemplo sencillo para elaborar diapositivas con `seminar`. El resultado de este ejemplo se muestra en la figura 4.8.

```
\documentclass{seminar}
\usepackage{amsfonts}
\usepackage[latin1]{inputenc}
\begin{document}

%---PREÁMBULO---
\title{Las cadenas de Markov}
\author{Sergio David Madrigal Espinoza}
\date{19 de junio de 2007}
\maketitle

%---PRIMERA DIAPOSITIVA---
\begin{slide}{Definición}

Una cadena de Markov, que recibe su nombre del matemático ruso Andrei
Markov, es una serie de eventos, en la cual la probabilidad de que
ocurra un evento depende del evento inmediato anterior.

\end{slide}

%---SEGUNDA DIAPOSITIVA---
\begin{slide}{Representación matemática}

Una cadena de Markov es una secuencia $X_{1},X_{2},X_{3},\ldots$ de
variables aleatorias. El rango de estas variables, es llamado espacio
de estados, el valor de $X_n$ es el estado del proceso en el tiempo
$n$. Si la distribución de probabilidad condicional de $X_{n+1}$ en
estados pasados es una función de $X_n$ por sí sola, entonces:

\begin{eqnarray*}
P(X_{n+1}=x_{n+1}|X_n=x_n, X_{n-1}=x_{n-1}, \ldots, X_2=x_2, X_1=x_1)=\\
P(X_{n+1}=x_{n+1}|X_n=x_n)
\end{eqnarray*}

\end{slide}

%---TERCERA DIAPOSITIVA---
\begin{slide}{Algunas aplicaciones}
\begin{itemize}
\item Número esperado de iteraciones que hará un algoritmo.
\item Para el pronóstico del clima.
\item Estudios socioeconómicos.

\end{itemize}
\end{slide}
\end{document}
```

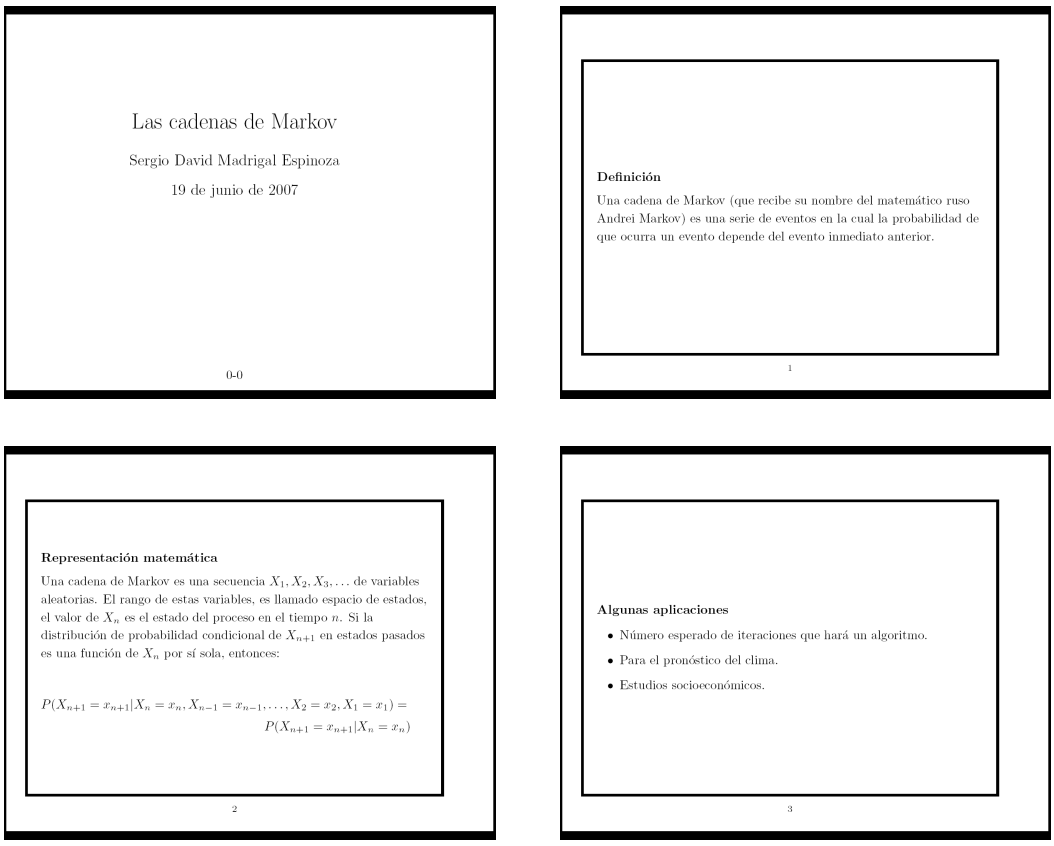


Figura 4.8: Diapositivas hechas con `seminar`.

`prosper`

Algunas de las ventajas de esta clase son:

- Gran variedad de plantillas, acompañadas cada una de su propio fondo.
- Permite crear enlaces de una diapositiva a otra.
- Permite crear enlaces a internet.
- Tiene siete diferentes efectos de trancisión de diapositivas.

A continuación se muestra un ejemplo de diapositivas para la clase `prosper`. Para poder ejecutar el siguiente ejemplo, es necesario que tengas en la carpeta el archivo `troispoints.sty`. En general, basta con cualquier otro estilo de tu elección, siempre y cuando hagas los arreglos necesarios en el programa. El resultado de este ejemplo se muestra en la figura 4.9. Si deseas ver ejemplos más avanzados consulta el siguiente sitio web `http://prosper.sourceforge.net/`.

```
\documentclass[letterpaper,pdf,troispoints,slideColor,colorBG]{prosper}
%letterpaper -> tamano carta,
%pdf -> dispone el documento en diapositivas para pdf,
%troispoints -> estilo de diapositiva,
%slideColor -> formato de color,
%colorBG -> tipo de color.
\usepackage{amsmath} % Imprime fuentes matemáticas de alta calidad

%---PREÁMBULO---
\title{Las bondades de la clase prosper}
\author{Sergio David Madrigal Espinoza}
\institution{Universidad Autónoma de Nuevo León}

\begin{document}
\maketitle % Diapositiva de presentación hecha con los datos del preámbulo

%---DIAPOSITIVA---
\begin{slide}[Split]{Enlaces} % Inicia una diapositiva con el efecto ‘‘Split’’
                              % y con el titulo ‘‘Enlaces’’
\begin{itemize} % Inicia una lista
\item Si pulsas \href{http://prosper.sourceforge.net/}{{\blue aquí}} iras
directo a la página de \texttt{Prosper} (siempre y cuando tu Acrobat Reader
este bien configurado).
\item Teclea \texttt{CTRL-L} para entrar o salir de pantalla completa.

\item Pulsa \hyperlink{LAST}{{\green final}}\hypertarget{SECOND}{}
para ir a la última diapositiva.
\end{itemize}
\end{slide}

%---OTRA DIAPOSITIVA---

\overlays{3}{
\begin{slide}[Dissolve]{Trabajando con fórmulas}
{\small % reduce el tamano del texto para esta diapositiva
La siguiente fórmula modela una serie de datos con tendencia aditiva y
estacionalidad multiplicativa.
\begin{equation}\label{modelo}
y_{(i-1)m+j}=l+x_{(i-1)m+j}p_{j-m}+\varepsilon_{(i-1)m+j}
\end{equation}

\FromSlide{2} %El texto debajo aparece en el segundo y hasta el ultimo clic.
donde $l$ es el nivel, $p_{j-m}$ imita los efectos de la estacionalidad y de
la tendencia y $\varepsilon_{(i-1)m+j}$ es una fuente de variación.

\FromSlide{3}%
La fórmula \ref{modelo} es un método simple para el pronóstico de datos
con tendencia aditiva y estacionalidad multiplicativa.}
\end{slide}
}
%---úLTIMA DIAPOSITIVA---
\begin{slide}[Glitter]{La última diapositiva}
  Esta es la \hypertarget{LAST}{última} diapositiva. ?‘Quieres ir a la
  \hyperlink{SECOND}{{\green segunda}}?
\end{slide}
\end{document}
```

Figura 4.9: Diapositivas hechas con prosper.

beamer

La clase beamer es similar a la clase prosper, con la ventaja de que trabaja conjunta y directamente con pdflatex aunque también puede hacerlo con dvips. Otra diferencia a prosper es que beamer no necesita cargar estilos de diapositivas pues ya vienen definidos junto con la clase. Los estilos de diapositivas de beamer son bastante ordenados y dan una idea general de la presentación en cada diapositiva lo cual facilita el entendimiento para la audiencia. A pesar de estas ventajas en cuanto a estilos de diapositivas de beamer sobre prosper, vale decir este último tiene estilos más llamativos y vistosos, los cuales estan inpirados en Microsoft Power Point, lo cual podría parecer una ventaja a más de uno.

A continuación mostramos un programa de LaTeX hecho con la clase beamer. El resultado se muestra en la figura 4.10. Si quieres ver más ejemplos de beamer consulta http://latex-beamer.sourceforge.net/.

```
\documentclass{beamer}
\mode<presentation>
{
  \usetheme{Warsaw} % Estilo de diapositivas: JuanLesPins, Berkeley, etc.
  \setbeamercovered{transparent} %Crea un efecto de transparencia para el
                                 %texto siguiente.
}
```

```
\usepackage{amsfonts}
\usepackage[latin1]{inputenc}
\usepackage[spanish]{babel}
\usepackage{times} % Tipo de letra

\title[MSNE] % úsalo si el titulo es muy largo
{Un Método Simple para el Pronóstico de una Serie de Tiempo No Estacionaria}

\author[Madrigal, Garza y Villarreal] % úsalo si hay varios autores
{S. D. Madrigal Espinoza, R. Garza Morales y C. E. Villarreal Rodríguez}

\institute[] % Un requisito casi siempre obligatorio
{Posgrado en Ingeniería de Sistemas\\
 Universidad Autónoma de Nuevo León}

\date[CCDL 2007] % Hay que abrebiar el título de la conferencia
{Congreso para la Creación de Diapositivas con \LaTeX, 2007}
% Utiliza el nombre de la conferencia o sus siglas.
% Esta información es útil solo para quienes leen el este programa

%Las siguientes instrucciones sirven para que el contenido
%de la conferencia aparesca en cada diapositiva.

\AtBeginSubsection[]
{
  \begin{frame}<beamer>{Contenido}
    \tableofcontents[currentsection,currentsubsection]
  \end{frame}
}

\begin{document}

%---PREÁMBULO---
\begin{frame}
  \titlepage %Crea la diapositiva de presentación
\end{frame}
%---DIAPOSITIVA---
\begin{frame}{Contenido}
 \tableofcontents %Despliega el contenido en una diapositiva.
\end{frame}

\section{Introducción}

\subsection{El problema básico}

%---OTRA DIAPOSITIVA---
\begin{frame}{Pronóstico de datos con tendencia aditiva y estacionalidad
              multiplicativa}
   \begin{itemize}
     \item Se dice que una serie de tiempo tiene tendencia aditiva cuando su
           crecimiento puede representarse en terminos de sumas o bien, cueando
           éste es proporcional al tiempo.
   \pause
     \item La estacionalidad multiplicativa es aquella culla amplitud crece a
           través del tiempo y como su nombre lo indica, suele ser representada
           como un producto.
   \end{itemize}
\end{frame}

\subsection{Trabajo previo}

%---UNA DIAPOSITIVA MÁS---
\begin{frame}{El método multiplicativo de Holt-Winters (MMHW)}
Características del MMHW
 \begin{itemize}
```

```
   \item Captura por separado los efectos del nivel la tendencia y la
         estacionalidad y los integra en el pronóstico.
      \pause
   \item Da pesos exponenciales mayores a las observaciones más recientes.
         \pause
   \item Es el método más popular para el pronóstico de este tipo de datos.
         \pause
   \item Cuenta con bases estadísticas tan solidas como la de los modelos
         ARIMA.
 \end{itemize}
\end{frame}

\section{La contribución}
\subsection{El MSNE}

%---UNA MÁS---
\begin{frame}{El MSNE}
Bajo la suposición de que los
datos tienen tendencia aditiva y estacionalidad multiplicativa,
podemos aproximar su comportamiento con la siguiente relación
\pause
\begin{equation}
y_{t}=l+x_{t}bs_{t}+\varepsilon_{t}.  \label{1}
\end{equation}%
\pause
Algunas diferencias fundamentales entre el MSNE y el MMHW son:
\pause
 \begin{itemize}
   \item El MMHW da pesos mayores a las observaciones más recientes, lo cual
         no hace el MSNE.
      \pause
   \item El MSNE usa menos variables que el MMHW.
      \pause
   \item Al contrario del MMHW, el MSNE cuenta con fórmulas cerradas para la
         estimación de sus parámetros.
         \pause
   \item Los parámetros del MSNE minimizan el error cuadrado medio de manera
         global.
 \end{itemize}
\end{frame}

\section*{Conclusiones} %El asterisco evita que las conclusiones aparescan en
                        %el contenido.
%---úLTIMA DIAPOSITIVA---
\begin{frame}{Conclusiones}
  \begin{itemize}
    \item En este trabajo ofrecimos un \alert{método simple} para el pronóstico
          de datos con tendencia aditiva y estacionalidad multiplicativa.
    \item La simplicidad de este modelo trae con sigo \alert{grandes ventajas}.
    \item El MSNE ha superado al MMHW en \alert{algunos casos}.
  \end{itemize}
  \vskip0pt plus.5fill % Para bajar 5 filas después de la primera
  \begin{itemize}
  \item
    Trabajos futuros
    \begin{itemize}
    \item Experimentación.
    \item Desarrollo de una base estadística para el MSNE.
    \end{itemize}
  \end{itemize}
\end{frame}
\end{document}
```

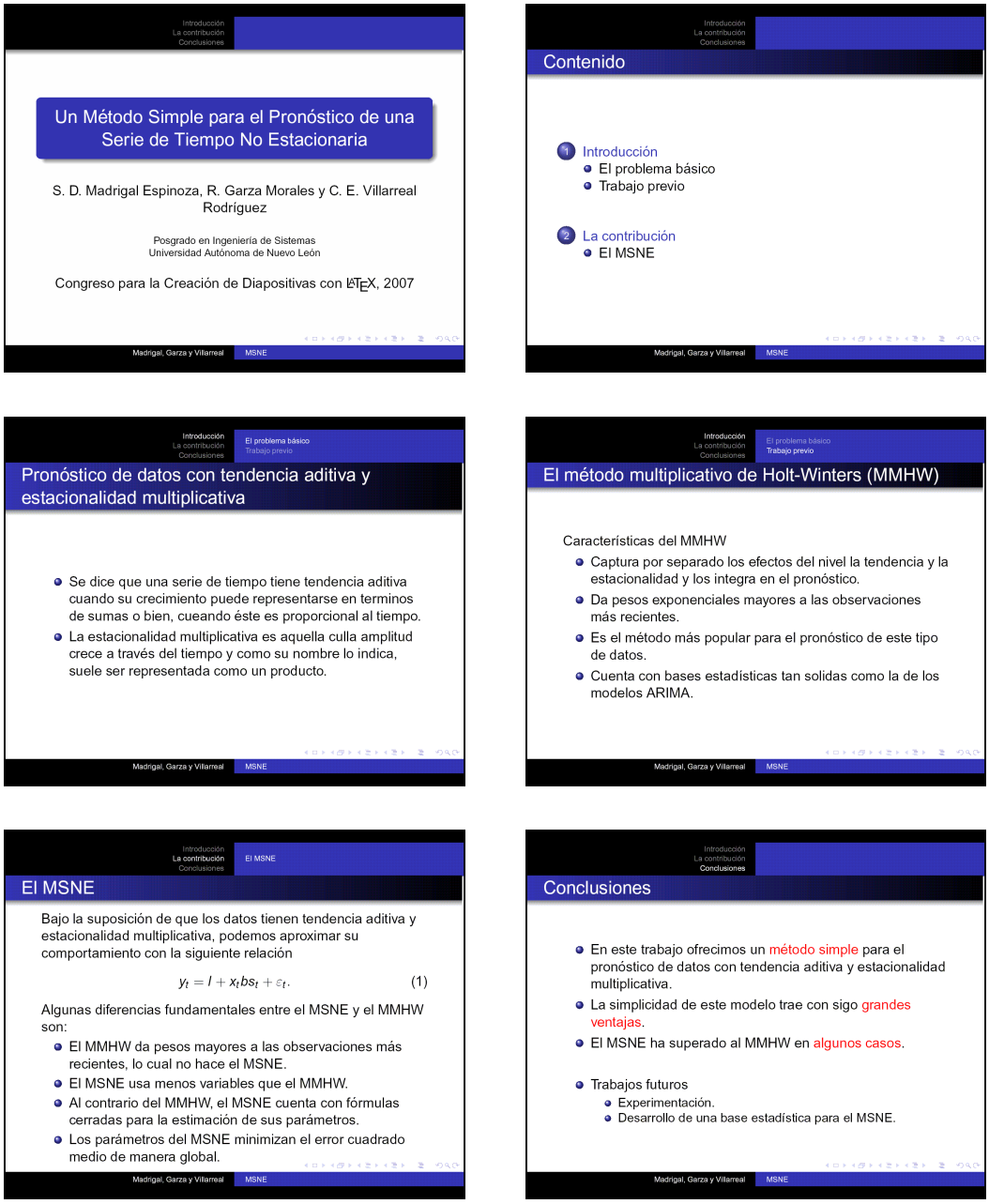


Figura 4.10: Diapositivas hechas con `beamer`.

`chaksem`

La clase `beamer` es ideal para presentaciones orientadas a investigación donde no es necesario comprender el tema en su totalidad, sino sólo captar la idea principal. En cambio, si es necesario explica completamente un tema como ocurre en una en un salón de clases normal, la estructura de

beamer sería inapropiada. Una alternativa para esta situación es la clase chaksem. La estructura de chaksem permite la elaboración rápida y sencilla de diapositivas.

En la figura 4.11 se muestran algunas diapositivas elaboradas con el siguiente programa. Para poder utilizar chaksem es necesario instalar los archivos chaksem.cls (y para algunos ejemplos haskell.sty). Para mayor información sobre chaksem consulta http://www.cse.unsw.edu.au/~chak/presentation/.

```
\documentclass[online]{chaksem}
\usepackage[T1]{fontenc}
\usepackage{alltt} %Para colorear letras.
\usepackage{amsmath} %Caracteres matemáticos de alta calidad.
\usepackage{pstricks} %Para crear líneas.

\makeatletter
\let\toprulecol=\dorange %Color de la barra superior.
\let\botrulecol=\dblue   %Color de la barra inferior.

\begin{document}

%---PREAMBULO E ÍNDICE---
\begin{slide}
  \begin{center}
    \vspace*{\fill} %Crea un espacio entre la línea superior y el título.
    \Large %Letras grandes para el título.
    {\dblue Breve Introducción a la\\
      Programación Lineal}
    \normalsize
      \begin{center}
      \vspace*{\fill}
      S. D. Madrigal Espinoza\\
      {\small\dgreen Universidad Autónoma de Nuevo León}
      \end{center}
      \small\texttt{sergio@yalma.fime.uanl.mx}
    \par\vfill
    \rule{.25\textwidth}{.5pt}
    %Para crear una línea entre el texto y el índice, el cual es muy breve.

    \par\vfill\bigskip
    \hspace*{\fill}\begin{minipage}{.75\textwidth}
    %Para crear un brebe índice dentro de la primer diapositiva

       \begin{slumerate}
       \item \normalsize Orígenes
       \item \normalsize Un ejemplo clásico: El problema de la mezcla
alimenticia
       \item \normalsize El modelo matemático
       \end{slumerate}
      \end{minipage}\hspace*{\fill}
  \end{center}%
  \vspace*{-.5\bigskipamount}
\setfooter{{\upshape Hecho con la clase chaksem de \LaTeX}}
\end{slide}

%---DIAPOSITIVA---
\begin{slide}
  \let\toprulecol=\dgreen %Color de la primer regla de esta diapositiva.
  \let\botrulecol=\dblue
  \heading{Orígenes} %Encabezado de diapositiva.

  \begin{second}
    \begin{slitemize}
      \item El problema de programación lineal fue concebido y resuelto por
```

```
L. V. Kantorovich y G. B. Dantzig de manera independiente.
      \item Dantzing publicó el artículo "Programación en una estructura
lineal"
de donde se adoptó el termino.
      \item Los modelos de programación lineal se cuentan entre los más
utilizadosactualmente.
    \end{slitemize}
  \end{second}

\end{slide}

%---DIAPOSITIVA---
\begin{slide}
   \heading{Un ejemplo clásico: el problema de la mezcla alimenticia}
    \begin{itemize}
    \item Tenemos "$n$" ingredientes alimenticios (indexados por "$j$").
    \item Cada ingrediente contiene cierta cantidad del nutriente "$i$".
    \item Los ingredientes deben ser mezclados para que satisfagan los
requerimientos alimenticios de cierta población.
    \item Existe un costo por cada unidad del ingrediente "$j$".
    \item El objetivo es minimizar el costo total satisfaciendo los
requerimientos nutricionales.
    \end{itemize}
\end{slide}

%---DIAPOSITIVA---
\begin{slide}
   \heading{El modelo matemático}
  \begin{eqnarray*}
  \text{Minimizar } & \displaystyle\sum_{j=1}^{n}c_jx_j\\
  \text{Sujeto a } & \displaystyle\sum_{j=1}^{n}x_j=b\\
   & l_ib \le \displaystyle\sum_{j=1}^{n}a_{ij}x_j \le u_ib \quad \forall \;i\\
   & 0 \le x_{j} \le u_j \quad \forall \; j
  \end{eqnarray*}
\end{slide}
\end{document}
```

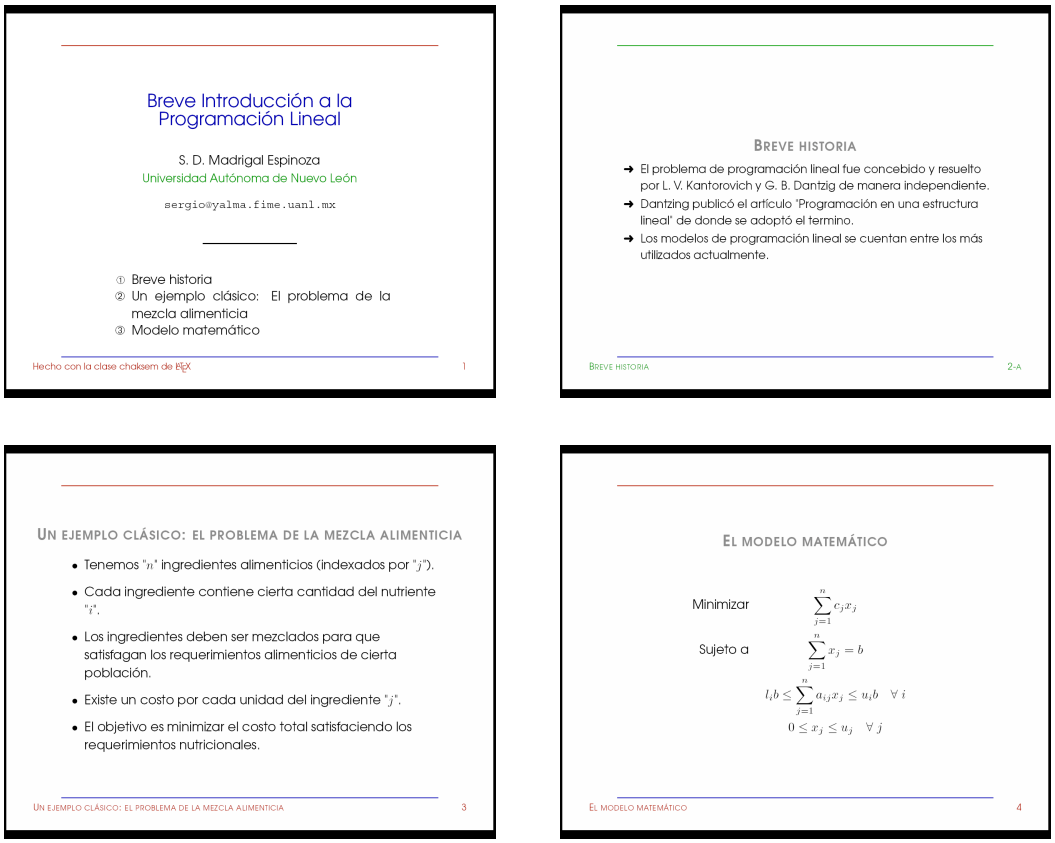


Figura 4.11: Diapositivas hechas con chaksem.

4.6.18. Carteles

a0poster y requerimientos del sistema

a0poster es una clase de TEX diseñada por G. Kettl y M. Weiser para crear posters científicos. La versión más reciente y en la que se desarrollan nuestros ejemplos es la 1.22b.

La importancia de crear posters en TEX radica básicamente en aquellas ventajas que ya representa en sí mismo la creación de un documento científico en TEX por ejemplo, la introducción de fórmulas, el control sobre la edición del documento, etcétera. Así, si uno ha creado ya un artículo en TEX y por alguna razón necesita preparar alguna presentación o algún poster podrá utilizar parte del documento que ya había escrito.

Para poder utilizar la clase a0poster fundamentalmente necesitarás dos archivos para crearlo, el archivo class (a0poster.cls) y el archivo que contiene las adaptaciones de los tamaños de letra (a0size.sty), éstos archivos deben ubicarse en un directorio en el que puedan ser leídos (por ejemplo en el mismo directorio en el que se encontraría un archivo del tipo mi_poster.tex). Para la compilación del archivo será necesario disponer tanto de LATEX como de dvips.

Los archivos de la clase a0poster así como una pequeña guía se pueden obtener de la red en

```
http://www.ctan.org/tex-archive/macros/latex/contrib/a0poster/
```

Sobre tamaño de papel y letra

a0poster está basado en la clase artículo (article) por lo que podrán utilizarse todos los mecanismos que están definidos en esta clase.

La clase a0poster trabaja distintos tamaños de papel, entre ellos los que se mencionan en la tabla posterior:

Opción	*Descripción*
landscape	formato horizontal, por default
portrait	formato vertical
a0b	"DIN A0 grande"[3]. Este tamaño es un poco más amplio que el formato A0, utiliza el ancho completo de una impresora HP Designjet 650C. También por omisión.
a0	DIN A0
a1	DIN A1
a2	DIN A2
a3	DIN A3
posterdraft	reduce la salida del archivo postscript al tamaño DIN A4, por lo que podrían efectuarse pruebas de impresión en una impresora ordinaria DIN A4.
draft	**obsoleto**– hace lo mismo que *posterdraft*, pero mientras las opciones se pasan a otros paquetes, está opción puede collisionar con otros paquetes (por ejemplo `graphics`).
final	genera el archivo postscript de tamaño original; por default.

Acerca de los tamaños estándar de papel, consulta Wikipedia en `http://es.wikipedia.org/wiki/Formato_de_papel`.

Obsérvese que además dispone de los siguientes tamaños de letra:

```
\tiny          12pt
\scriptsize    14.4pt
\footnotesize  17.28pt
\small         20.74pt
\normalsize    24.88pt
\large         29.86pt
\Large         35.83pt
\LARGE         43pt
\huge          51.6pt
\Huge          61.92pt
\veryHuge      74.3pt
\VeryHuge      89.16pt
\VERYHuge      107pt
```

Así el inicio del documento `mi_poster.tex` podría verse como se muestra a continuación

```
\documentclass[portrait,a0b]{a0poster}
\usepackage{babel,graphicx,pstricks}
\begin{document}
```

Posicionamiento de texto y gráficas

Para ayudar a simplificar más la elaboración del poster en un archivo llamado `a0poster.sty` se han definido macros, el cual se puede obtener en la red por ejemplo de `http://yalma.fime.uanl.mx/~elisa/temp/poster/`

y debe estar en el mismo directorio que en el que se encuentra `mi_poster.tex`. Este paquete permite por ejemplo insertar texto en columnas (`\col{}`), realizar el encabezado del poster, etcétera, de tal manera que sólo tendríamos que declarar en en preámbulo `\usepackage{a0poster}` para poder utilizar los mecanismos definidos en el mismo. Cabe mencionar que todo puede ser modificado de acuerdo a los gustos y necesidades de quien crea el poster. Por ejemplo, todos los mecanismos definidos para la clase `article` funcionan para `a0poster`, así `\section{}`, `\subsection{}` y `\subsubsection{}` siguen funcionando perfectamente para organizar la información, sin embargo si por alguna razón quisieramos indicar sólo el título de algo sin numeración alguna, podríamos utilizar `\paragraph{}`, el cual está definido en `a0poster.sty`.

El acomodo del texto y las gráficas es algo que aún queda a cargo del creador del poster, por lo general en un poster se organiza la información en columnas, para ello en el preámbulo del archivo `mi_poster.tex` podría utilizarse el paquete `\usepackage{multicols}`, la estructura es muy simple y funciona bastante bien

```
\begin{multicols}{3} % El número 3 indica el número de columnas
Texto, y más texto\\
Fórmulas\\
Gráficas y demás
\end{multicols}
```

La otra opción para la generación de las columnas y la insersión de gráficas se simplifica bastante con ayuda de las macros creadas en el archivo `a0poster.sty`. Así, nuestro archivo `mi_poster.tex` se vería así

```
\documentclass[a0poster]
\usepackage[spanish]{babel}
\usepackage[ansinew]{inputenc}
\usepackage{a0poster}
\usepackage{graphicx}

\begin{document}
\col{
\paragraph{Nombre de la sección}
Todo lo que quieras en la primera columna
}
\col{
Todo lo que quieras en la segunda columna
}
\col{
Todo lo que quieras en la tercera columna
}
\end{document}
```

Obsérvese que no debe haber líneas en blanco entre la llave que termina la columna anterior y la declaración de la siguiente columna.

La ventaja que presenta el \col es que uno controla la cantidad de información que pone en cada columna (en a0poster.sty está programado para tres culumnas, si necesitas más podrías ir a modificar el archivo y adecuarlo a tus necesidades), mientras que con \multicols distribuye el texto de acuerdo a la cantidad de columnas que se especifíquen, el uso de cualquiera de las dos opciones que se han manejado para distribuir el contenido del poster en columnas quedan a gusto del creador del poster.

El paquete \usepackage{grachicx} permite insertar imágenes .eps para ello deberá estar declarado en el preámbulo (los archivos de las gráficas preferentemente deberían estar en donde está mi_poster.tex). Para poder insertar gráficas por este medio deberemos definir el ancho sobre el que ahora insertaremos la figura, para ello podríamos definir una variable con una longitud acorde al número de columnas que hemos creado, en nuestro caso tres

```
\newlength{\figwidth}
\setlength{\figwidth}{25cm}
```

La declaración \figwidth debería realizarse de preferencia el en preámbulo del documento. Una vez hecho lo anterior podríamos insertar una imagen utilizando el ambiente minipage

```
\begin{minipage}{\figwidth}
\begin{center}
\includegraphics[width=10cm]{interpola.eps}
\end{center}
\end{minipage}
```

En los tutoriales de http://www.tug.org.in/tutorial/ y http://www.tex.uniyar.ac.ru/doc/pst_ug.pdf podría revisarse cómo crear figuras con PsTricks. También deberá incluirse en el preámbulo el paquete

```
\usepackage{pstricks}
```

Insertaremos gráficos con extensión .eps con el paquete \usepackage{epsfig}:

```
\begin{center}
\epsfig{file=interpola.eps, width=4cm}
\end{center}
```

Edición del poster

Como siempre uno tiene múltiples opciones cuando se está creando algo en LaTeX, el encabezado no será la excepción y la elección de nueva cuenta depende de las necesidades y gustos de quien crea

el poster. En el encabezado por lo general se encontrarán algunos logotipos, el título del poster, el autor, y datos del mismo.

El paquete `a0poster` contiene la definición que nos permite crear el encabezado del poster. De nueva cuenta uno podría editar el archivo `a0poster.sty` por ejemplo para agregar datos que uno quiere que siempre aparezcan, como los datos del autor, los logotipos, etcétera. Si se desea utilizar esta opción después del `\begin{document}` uno debería escribir

```
\title{}
\author{}
\inst{}
\email{}
\makeheader
```

La segunda opción es por si decidimos no utilizar el paquete `a0poster`, entonces uno tendría que definir "a mano" el encabezado del poster, una sugerencia sería

```
\bigskip
\begin{center}
{\veryHuge \textsc{ Fórmulas baricéntricas ...}}\\[1cm]
\end{center}
\bigskip

\begin{minipage}[b]{.2\textwidth}
\begin{center}
\includegraphics[width=10cm]{uni.eps}
\end{center}
\end{minipage}
\begin{minipage}[b]{.6\textwidth}
\begin{center}
{\huge \sl Yajaira Cardona, ...} \\ [.5cm]
{\LARGE Sociedad Matemática Mexicana}\\[.5cm]
{\Large \rm Universidad
Autónoma de Tabasco}\\[.5cm]
{\large Octubre 2006}\\[.5cm]
\end{center}
\end{minipage}
\begin{minipage}[b]{.2\textwidth}
\begin{center}
\includegraphics[width=12cm, height=10cm]{logopisis.eps}
\end{center}
\end{minipage}
```

La presentación final del poster podría ajustarse a las preferencias del creador del mismo, por ejemplo márgenes, fondos, colores, etcétera.

Por medio del paquete `pstricks` con `\red` la fórmula o texto precedente aparecerá en color rojo. Los colores red, blue, yellow, cyan y magenta, los de escala de grises, white, lightgray, gray, darkgray y black, también están definidos, aunque también podrían definirse otros colores, por ejemplo

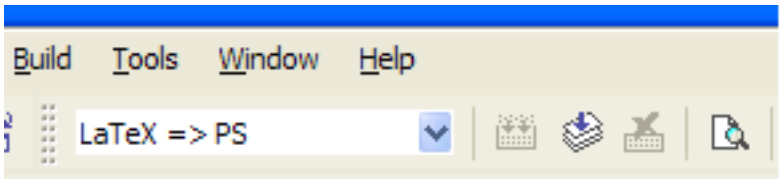
```
\newrgbcolor{DarkOrange}{1 .498 0}
\newrgbcolor{darkblue}{0.0 0.0 0.5}
\DarkOrange Fórmula
\darkblue Texto
```

El rango de números va de 0 a 1 y describen un color en el sistema `rgb` (red, green, blue). Aunque también podrían definirse colores en otro sistema como CMYK (acrónimo de Cyan, Magenta, Yellow y Key[4]).

También es posible crear cajas, cajas sombreadas, muchísimas cosas que hacen más llamativo el poster, `psTricks` (ver los tutoriales mencionados con anterioridad) es una buena herramienta para crear ese tipo de objetos.

Procesamiento

Para Windows, puedes bajar gratuitamente TEXnicCenter, ver `http://texniccenter.en.softonic.com/`. Es muy sencillo generar un archivo con extensión `.ps`, sólo se indica `latex→dvips`, tal y como se muestra en la figura



Se ejecutan las siguientes líneas en la consola para generar los archivos `mi_poster.ps` y `mi_poster.pdf` ambos de tamaño A0

```
latex mi_poster.tex

dvips -o poster.ps mi_poster.dvi

ps2pdf poster.ps poster.pdf
```

Si al generar el archivo con extensión `.ps` marca errores, deberás ejecutar primero `bash`.

Y las siguientes líneas para generar también los archivos `poster.ps` y `poster.pdf` pero ahora en tamaño carta

[4]Negro

```
psresize -W841mm -H1189mm -pletter poster.ps poster_carta.ps

ps2pdf mi_poster_carta.ps poster_carta.pdf
```

El psresize permite crear reescalar y centrar un documento en un tamaño de papel diferente. Ver

```
http://es.tldp.org/Paginas-manual/man-pages-es-extra-0.8a/man1/psresize.1.html
```

para más información.

En la página del taller, puedes encontrar los archivos a0poster.sty, a0poster.cls, a0size.sty, además los gráficos interpola.eps, .eps y el logotipo de la Universidad de Nuevo León uni.eps y el de Postgrado de Ingeniería de Sistemas logopisis.eps, además los siguientes archivos

1. mi_poster_v1.tex, no utiliza a0poster.sty
2. mi_poster_v2.tex, utiliza a0poster.sty

Además hemos creado un documento Esqueleto_poster.tex que podría servir como una base para comenzar a desarrollar posters. Éste trabaja con a0poster.sty

4.7. Páginas de web

En esta sección aprenderás hacer páginas personales donde podrás tener tu espacio en la red. (Por ejemplo DropBox se puede usar para almacenar una página personal en la carpeta Public.)

4.7.1. Página inicial de una carpeta: index.html

Al entrar con un navegador a una carpeta existente del sistema de archivos de un servidor Http, si existen permisiones de entrar y leer la carpeta, el servidor va a enviar al navegador un archivo de nombre index.html (si cuenta con Ssl, es index.shtml y en algunos sistemas basados en Windows todavìa index.htm). El navedagor muestrà la pàgina guardada en el archivo recibido.

Si no existe tal archivo, el navegador va a mostrar la lista de los contenidos de la carpeta, es decir, el servidor crea en una manera dinàmica un documento de Html que contiene todo lo que se puede ver con ls -l en la dicha carpeta en el servidor mismo. Se puede bloquear la generaciòn de tal lista por crear un archivo con el nombre index.html, aunque no contenga nada, por ejemplo con la instrucciòn

```
touch index.html
```

que nada más crea un archivo vacío. En la siguiente sección se explica la sintaxis básica del lenguaje Html.

4.7.2. El lenguage Html

HTML (HyperTextMarkup Languaje) es un lenguaje para dar estructura de documentos mostrados en navegadores de Web. La primera instrucción a utilizar es el de <html> que es como se abre y cierra la codificación de la pagina: <html> marca el inicio del documento y </html> su fin. (Toda aquella instrucción que tenga un slash significa que es el fin de el ciclo de esa instrucción).

Con <head> pondremos los titulo de la pagina y se escribe de la siguiente manera:

```
<html>
<head>
<title>El texto que aparece en la barra del navegador</title>
</head>
```

Los contenidos de la página misma van en la segunda parte que se llama <body>:

```
<html>
<head>
<title>...</title>
</head>
<body>
...
</body>
```

Las instrucciones que se pueden utilizar dentro de <body> son numerosos. Entre los más básicas son las siguientes:

- <h1>...</h1>: título del documento
- <h2>...</h2>: título de una sección
- <h3>...</h3>: título de una subsección
- <h4>...</h4>: título de una subsubsección
- ...: referencia a otra página — por ejemplo, en el lugar donde se incluye la secuencia siguiente, <a ref="http://www.uanl.mx/»UANL, se define como la página donde se va a conectar el enlace la página principal de la UANL y antes de cerrar la instrucción, el nombre corto del enlace
- ... : un párrafo de texto
-
: un cambio de línea forzada

- `<font color="#RRVVAA" size="Y" face="...">...</font>`: típo de letra (face), tamaño y el color definido en formato rojo-verde-azul (RGB) con tres números hexadecimales entre 00 y FF para utilizar entre el comienzo y el fin de la instrucción misma

Dentro de la instrucción`<body...>` se puede manipular los colores del fondo de pantalla, el color de las palabras escritas y las imágenes que estan dentro de la pagina personal. Estas son algunas instrucciones que manipulan el fondo de pantalla y las palabras de la página:

- `<body ... text="..." ...>` sirve para darle color de todo el texto que no tiene otro color definido.
- `<body ... bgcolor="..." ...>` sirve para darle color al fondo de la pantalla.
- `<body ... background="http://..." ...>` sirve para darle una imagen al fondo de la pantalla.

Los colores se pueden poner mediante códigos hexadecimales o con el nombre del color en ingles:

black	`#000000`	(negro)
maroon	`#990000`	(marrón)
green	`#009900`	(verde)
grey	`#999999`	(gris)
navy	`#000099`	(azul fuerte)
purple	`#990099`	(morado)
red	`#FF0000`	(rojo)
yellow	`#FFFF00`	(amarillo)
blue	`#0000FF`	(azul)
white	`#FFFFFF`	(blanco)

Para crear *listas*, hay dos opciones: listas numeradas con `<ol>...</ol>` y listas no numeradas con `<ul>.... </ul>`. Los elementos de las listas se crea en ambos casos con `<li>...</li>`.

Los siguientes ejemplos explican concretamente cómo lograr ciertos efectos.

- **Imágenes**: `<img src="http://yalma.fime.uanl.mx/~elisa/pics/uanl.jpg">` nos sirve para cargar una imagen a nuestra página personal desde un archivo que tengamos y plasmarlo en la página. La parte de la instrucción de `src` es la que va a buscar el navegador para cargar la imagen y mostrarla a la web.
- **Tablas**: `<table>...</table>` es una instrucción muy completa: dentro de ella se pueden utilizar muchas instrucciones para su mejor manejo. La primera función de esta instrucción es la de hacer una tabla (vea figura 4.12 para un ejemplo) en nuestra página y con las siguientes instrucciones se configura el formato de cómo va a quedar la tabla al final de presentación:
  - `<caption>...</caption>` muestra el titulo de la tabla

- `<table ... cellspacing="..." ... >` ayuda a administrar el espacio que hay *entre* cada cuadro dentro de la tabla por definir un número que determina cuántos pixeles de espacio se pone.
- `<table ... cellpadding="..." ...>` ayuda a administrar el espacio que hay *dentro de* cada cuadro dentro de la tabla por definir un número que determina cuántos pixeles de espacio se pone.
- `<table ... border=" ...>` define el grosor de nuestra tabla en pixeles.
- `<tr>..</tr>` define una fila de la tabla.
- `<th>..</th>` define un encabezado de una fila o columna de la tabla.
- `<td>...</td>` contiene los datos dentro de un cuadro de la tabla.

```
<html>
<body>
<table border="1" cellpadding="3" cellspacing="0">
<tr><th>Primero</th><th colspan="2">Segundo</th></tr>
<tr><td rowspan="2">Cosa</td><td>Otra cosa</td><td>Algo m&aacute;s</td></tr>
<tr><td>Detalle</td><td>Otro detalle</td></tr>
</table>
</body>
</html>
```

Primero	Segundo	
Cosa	Otra cosa	Algo más
	Detalle	Otro detalle

Figura 4.12: Arriba, un ejemplo de una definición de una tabla en Html, y abajo, la estructura que resulta.

Los acentos y caracteres especiales en documentos de Html están escritos con palabras de código para mostrarlos bien en diferentes navegadores y ambientes. Algunos de los más importantes están en el Cuadro 4.3.

Varias herramientas ofrecen generación automática de Html y existen convertidores para diferentes tipos de documentos a Html (como por ejemplo a LaTeX).

Cuadro 4.3: Algunos símbolos especiales en Html.

á	á
é	é
í	í
ó	ó
ú	ú
á	Á
é	É
Í	Í
ó	Ó
ú	Ú
ñ	ñ
ñ	Ñ
<	<
>	>
&	&
ª	ª
º	º
¿	¿
¡	¡

4.8. Referencias bibliográficas

Para colocar la bibliografía editándola directamente en un documento LaTeX puede usarse el entorno `thebibliography` en la cual cada entrada bibliográfica se hará mediante la instrucción `\bibitem`.

```
\begin{thebibliography}{X}

\bibitem{Baz} {\sc Bazaraa, M. S.}, {\sc J. J. Jarvis} y {\sc H.
D. Sherali}, {\it Programación lineal y flujo en redes}, segunda
edición, Limusa, México, DF, 2004.

\bibitem{Dan} {\sc Dantzig, G. B.} y {\sc P. Wolfe},
<<Decomposition principle for linear programs>>, {\it Operations
Research}, {\bf 8}, págs. 101--111, 1960.

\end{thebibliography}
```

Y se creará la siguiente salida en el documento. La numeración es creada por LaTeX.

[1] Bazaraa, M. S., J. J. Jarvis y H. D. Sherali, *Programación lineal y flujo en redes*, segunda edición, Limusa, México, DF, 2004.

[2] Dantzig, G. B. y P. Wolfe, «Decomposition principle for linear programs», *Operations Research*, **8**, págs. 101–111, 1960.

El parámetro X al inicio del entorno es un número que no sea excedido por la cantidad de entradas bibliográficas, en este caso podría ser 9. Lo que se encuentra entre llaves después de cada `\bibitem` es una palabra clave con la cual se podrán hacer citas bibliográficas.

Ahora, para hacer una *cita bibliográfica* ha de usarse la instrucción `\cite{}`.

```
Como se puede ver en \cite{Baz} ...
```

Con lo cual se obtiene el siguiente resultado.

Como se puede ver en [1] ...

Se pueden agregar algunas notas extras al citar, por ejemplo

```
Como se puede ver en \cite[pág.\ 20--21]{Baz} ...
```

Con el siguiente resultado.

Como se puede ver en [1, pág. 20–21] ...

La disposición de las entradas y su formato está completamente a nuestra libertad (y responsabilidad). LaTeX numerará las entradas bibliográficas según aparezcan en el entorno `thebibliography`. Conforme las entradas sean agregadas o quitadas de la lista, LaTeX actualizará automáticamente la numeración de las obras en la bibliografía y en los lugares en que fueron citadas.

También se pueden personalizar las etiquetas con que han de citarse las obras, agregando a cada entrada de la bibliografía la etiqueta deseada de la siguiente manera.

```
\bibitem[DanWol60]{Dan} {\sc Dantzig, G. B.} y ...
```

Ahora se produce el siguiente efecto al citarlo, se sigue citando igual (con la clave asignada y no con la etiqueta), en este caso `\cite{Dan}`.

Como se puede ver en [DanWol60] ...

4.8.1. BibTex

También se puede emplear BibTex, una poderosa herramienta hermana de LaTeX especialmente diseñada para el apoyo a la bibliografía. Lo primero que necesitamos para emplear BibTex es una *Biblioteca Virtual*, esto es una colección de archivos con extensión `.bib` en el que se encuentra la bibliografía que requiere nuestro documento (la creación de estos archivos se discutirá más adelante).

Después basta indicar en el documento las librerías que serán usadas. Si son más de una se deben separar por comas.

Por ejemplo, supongamos que tenemos dos archivos donde se encuentra la bibliografía a emplear, `librero1.bib` y `librero2.bib`, entonces, para indicar a BibTex que la bibliografía ha de buscarse en tales archivos es necesario agregar la siguiente linea.

```
\bibliography{librero1,librero2}
```

Y BibTex agregará de tales archivos sólo aquellas entradas que hallan sido citadas mediante un `\cite{}`. Si, por alguna razón, desea incluirse en la bibliografía una entrada que no es citada puede emplearse la orden `\nocite{}` lo cual no aparecerá en el documento.

Las entradas que se agregarán se enlistarán según el *estilo* que sea definido. Para definir el estilo es necesario agregar la siguiente linea.

```
\bibliographystyle{estilo}
```

Por defecto BibTex cuenta con los siguientes estilos.

`plain` Dispone las entradas de la bibliografía por orden alfabético. A cada una le asigna un número entre corchetes.

`unsrt` Dispone las entradas en el orden con que se fueron citando con `\cite` o `\nocite`.

`alpha` Ordena las entradas igual que `plain` pero los marcadores se construyen con una abreviatura del autor o autores y el año de publicación.

`abbrv` Ordena las entradas igual que `plain` y construye los marcadores de la misma forma, pero en la indicación de la referencia se emplean abreviaturas para los nombres de pila, meses y, en ocasiones, los nombres de las revistas.

Desafortunadamente los estilos predefinidos de BibTex tienen algunos inconvenientes. El primero que salta a la vista es que su idioma por defecto es el inglés, por lo que, por ejemplo, en una lista de autores aparecerá *and* precediendo al último autor. Además, la disposición de las partes de una entrada bibliográfica (autor, título, publicación, año, etc.) son ordenadas según un estándar que no suele coincidir con el existente en otros idiomas, como el español.

Para remendar esto, uno puede crearse su propio estilo, un archivo con extensión `.bst`. Esto se puede hacer *a mano*, lo cual requiere mayores conocimientos de TEX, o bien mediante la herramienta `custom-bib` que permite crear estilos al capricho (soporta idiomas), llenando un (nada breve) cuestionario, personalizando hasta el mínimo detalle, pero de esto no hablaremos en este documento.

Creando archivos `.bib`

Como se decía, un archivo `.bib` es una *base de datos* de entradas bibliográficas, el cual contiene las entradas en determinado formato. Por ejemplo, para agregar las entrada usadas en la sección

anterior (un artículo y un libro) al archivo, ha de hacerse como sigue.

```
@ARTICLE{Dan,
author = {Dantzig, G. B. and P. Wolfe},
year = 1960,
title = {Decomposition principle for linear programs},
journal = {Operations Research},
volume = 8,
pages = {101--111}
}

@BOOK{Baz,
  author = {Bazaraa, M. S. and J. J. Jarvis and H. D. Sherali},
  year = 2004,
  title = {Programación lineal y flujo en redes},
  edition = 2,
  publisher = {Limusa},
  address = {M{\'{e}}xico, DF}
}
```

Cada entrada consiste de tres partes: el **tipo** de entrada (en este caso `ARTICLE` y `BOOK`); una palabra **clave** (en el ejemplo `Dan` y `Baz`) con que ha de citarse; y los **campos**, es decir, la información de la entrada.

Los campos deben ser separados por comas, también la clave debe ser seguida de una coma. Debe tenerse en cuenta que en este archivo debe emplearse la escritura estándar de LaTeX (atención con los acentos y otros signos) independientemente de los paquetes que se hayan cargado al documento.

Para cada entrada debe definirse una serie de *campos*. Cada tipo de publicación contiene información diferente. Un libro y una revista requieren diferentes campos. Para cada tipo de entrada los campos se dividen en tres clases.

Requeridos: Si se omite un campo de este tipo se producirá un mensaje de advertencia y, algunas veces, el formato de la entrada en la bibliografía será incorrecta. Si la información de este campo no está disponible, es posible que no esté empleando el tipo de entrada correcto por lo que quizá sería mejor cambiar de tipo o bien, ignorar la advertencia.

Opcional: La información de un campo de este tipo será usada si está disponible, pero puede ser omitida y no causará el menor problema.

Ignorado: La información de un campo de este tipo será ignorada.

Todo campo que no sea **requerido** u **opcional** será ignorado, por lo que no será agregado a la entrada bibliográfica. Sin embargo, no es mala idea agregar tanta información relevante como sea

posible en una entrada. Por ejemplo, puede agregarse el resumen, con lo cual el lector podrá hacer búsquedas bibliográficas más fácilmente consultando los archivos `.bib`.

A continuación describimos los campos reconocidos por los estilos de bibliografía estándares. Cualquier entrada puede incluir otros campos, los cuales serán ignorados por esos estilos.

`address`: Usualmente la dirección de la editorial.

`author`: Nombre(s) del autor(es).

`booktitle`: Título del libro.

`chapter`: El número de un capítulo (o sección, etc.).

`edition`: La edición de un libro, por ejemplo, *Segunda*.

`editor`: Nombre(s) del (de los) editor(es).

`howpublished`: Forma en que fue publicada la obra.

`institution`: Institución responsable de un informe técnico.

`journal`: Nombre del periódico o revista.

`key`: Empleado para la alfabetización, referencias cruzadas y para crear una etiqueta cuando la información del autor no está disponible. No debe confundirse con la palabra clave usada en el `\cite` y que debe colocarse al inicio de la entrada.

`month`: El mes de publicación o, para un trabajo inédito, en el que fue escrito.

`note`: Cualquier información adicional que pueda ayudar al lector.

`number`: El número del periódico, la revista, el informe técnico o del trabajo en una serie.

`organization`: La organización responsable de una conferencia o publica un manual.

`pages`: Números de páginas.

`publisher`: El nombre de la editorial. No debe confundirse con el editor.

`school`: Nombre de la escuela donde fue escrita una tesis.

`series`: El nombre de una serie o conjunto de libros.

`title`: El título del trabajo.

`type`: El tipo de un informe técnico.

`volume`: El volumen de un periódico o una revista, o de algún libro que conste de volúmenes.

`year`: El año de publicación. Para un trabajo inédito, el año en que fue escrito. Generalmente debe consistir de cuatro dígitos, por ejemplo 1984.

A continuación describimos los *tipos de entrada*.

`ARTICLE:` Un artículo de un periódico o revista. **Campos requeridos:** author, title, journal, year. **Campos opcionales:** volume, number, pages, month, note.

`BOOK:` Un libro con una editorial explícita. **Campos requeridos:** author o editor, title, publisher, year. **Campos opcionales:** volume o number, series, address, edition, month, note.

`BOOKLET:` Un trabajo impreso y distribuido, pero que no tiene una editorial o institución responsable. **Campos requeridos:** title. **Campos opcionales:** author, howpublished, address, month, year, note.

`INBOOK:` Una parte de un libro, como un capítulo, una sección, un rango de páginas, etc. **Campos requeridos:** author o editor, title, chapter o pages, publisher, year. **Campos opcionales:** volume o number, series, type, address, edition, month, note.

`INCOLLECTION:` Una parte de un libro con título propio. **Campos requeridos:** author, title, booktitle, publisher, year. **Campos opcionales:** editor, volume o number, series, type, chapter, pages, address, edition, month, note.

`INPROCEEDINGS:` Un artículo de las memorias de un congreso. **Campos requeridos:** author, title, booktitle, year. **Campos opcionales:** editor, volume o number, series, pages, address, month, organization, publisher, note.

`MANUAL:` Documentación técnica. **Campos requeridos:** title. **Campos opcionales:** author, organization, address, edition, month, year, note.

`MASTERSTHESIS:` Una tesis de maestría. **Campos requeridos:** author, title, school, year. **Campos opcionales:** type, address, month, note.

`MISC:` Para cuando el resto falla. **Campos requeridos:** Ninguno. **Campos opcionales:** author, title, howpublished, month, year, note.

`PHDTHESIS:` Tesis de doctorado. **Campos requeridos:** author, title, school, year. **Campos opcionales:** type, address, month, note.

`PROCEEDINGS:` Las memorias de un congreso. **Campos requeridos:** title, year. **Campos opcionales:** editor, volume o number, series, address, month, organization, publisher, note.

`TECHREPORT:` Un informe publicado por una institución. **Campos requeridos:** author, title, institution, year. **Campos opcionales:** type, number, address, month, note.

`UNPUBLISHED:` Un documento (inédito), con un autor y un título, pero que no ha sido formalmente publicado. **Campos requeridos:** author, title, note. **Campos opcionales:** month, year.

Agregamos algunas notas que no debe dejar de leer para crear correctamente su archivo `.bib`.

Autor: Las indicaciones que se dan en este apartado aplican para los campos `author` y `editor`, sin embargo, nos referiremos indistintamente como *autor*.

En caso de ser más de uno, cada par de autores debe separarse por la palabra reservada `and`, sin importar si son más de dos. Si son demasiados, puede recurrirse a terminar la lista con un `and others`, y cualquier estilo estándar imprimirá un *el al.*[5]

Respecto a la forma de escribir el nombre de cada autor, es necesario aclarar que, para BibTex cada nombre consiste de cuatro partes: Nombre, von, Apellido y Jr. Cada parte consiste de una lista (que puede ser vacía) de *palabras*. La parte Apellido deberá incluirse siempre que ninguna otra parte esté, de tal manera que cuando sólo una *palabra* aparece, esta será siempre la parte Apellido.

En general, BibTex permite tres formas de escribir el nombre:

```
Nombre von Apellido
von Apellido, Nombre
von Apellido, Jr, Nombre
```

La primera forma no debe usarse si hay una parte Jr o la parte Apellido tiene más de una *palabra* y no hay parte von.

Por ejemplo, el nombre de *Per Brinch Hansen* deberá estar escrito como:

Brinch Hansen, Per

De esta manera, la parte Nombre estará dada por *Per*; la parte Apellido estará formada por dos *palabras*, *Brinch* y *Hansen*; mientras que las partes von y Jr están vacías. Si en cambio se escribiera:

Per Brinch Hansen

BibTex tomaría (erróneamente) *Per* y *Brinch* como la parte Nombre y *Hansen* como la parte Apellido.

Título: Algunos estilos de la bibliografía modifican los títulos colocando la inicial de cada palabra en mayúscula (salvo algunas palabras, con reglas no siempre claras). Otros estilos hacen exactamente lo contrario, colocar cada letra que no sea la inicial del título en minúscula.

Ambos estilos suelen dar problemas pues en más de una ocación debe respetarse la escritura de algunas palabras. Por ejemplo, siglas del tipo SMM, corren el riesgo de ser transformadas en Smm, o en smm; números cardinales como XIII podrían sufrir un maltrato parecido.

Para evitar este efecto, podemos colocar entre llaves aquellas formas que se desea sean respetadas tal como las colocamos. Ejemplos:

```
{XXXVIII} Congreso Nacional de la {SMM}
{CONACyT}
{1er} Taller de verano
```

[5] La abreviatura *et al.* viene del latín (y otros) y es legal en las publicaciones internacionales.

Cabe mencionar que en el idioma español sólo la primera palabra y los nombres propios deben iniciar con mayúscula, por lo que iniciar cada palabra de un título con mayúscula es incorrecto. Esta regla debe observarse incluso si se está citando una obra cuyo título aparece con ese estilo. Como excepción, se escriben con mayúscula los sustantivos y adjetivos que forman parte del nombre de publicaciones periódicas o de colecciones.

Comentarios: En un archivo `.bib` el símbolo % no es un caracter de comentario como en LaTeX, por lo que puede usarse directamente en caso de necesitarse. Ahora, para colocar un comentario en el archivo `.bib`, basta colocarlo sin signo alguno, pues BibTex considera como comentario toda entrada que no inicie con una @.

El campo `key`**:** Cuando se emplea el estilo `alpha` BibTex suele construir la etiqueta con la información del campo `author`, por ejemplo [Wol97] para Laurence A. Wolsey.

Para los tipos en que no hay un autor, como `MANUAL` o `PROCEEDINGS`, la etiqueta se construye con la información del campo `organization`. Es común que en estos casos se desee proponer la etiqueta. Por ejemplo para una organización como

```
organization = {The Association for Computing Machinery},
```

es muy común que exista una abreviatura con la cual la organización es reconocida, en este caso es preferible definir la entrada

```
key = {ACM}
```

Sin el campo key el estilo `alpha` tomaría las primeras tres letras de la organización para construir la etiqueta (ignorando el artículo), por lo que se crearía la etiqueta [Ass86]. Con el campo key, en cambio, la etiqueta sería [ACM86] el cual será más informativo para el lector.

No siempre será necesario crear una etiqueta con key, por ejemplo, la organización

```
organization = {Unilogic, Ltd.},
```

puede razonablemente conservar la etiqueta que se crearía por defecto, [Uni86].

Editores de archivos `.bib`**:** Existen varias herramientas que ayudan en la gestión de librerías BibTex, si no se desea hacer un archivo `.bib` *a mano*.

Una herramienta de la que se suele hablar bien es **BibDB**, también en su versión WinBibDB. Disponibles en `CTAN:support/bibdb`, o bien en su página

```
http://www.mackichan.com/BibDB/default.htm
```

Otra herramienta popular es `bibtex.el`, que es el modo BibTEX para `emacs`. Usualmente ya viene

incluido con emacs. En todo caso se puede obtener en su página

```
http://www.ida.ing.tu-bs.de/people/dirk/bibtex/
```

4.8.2. Bibliotecas electrónicas e índices de citas

En línea hay acceso a varias revistas y actas de congresos importantes. Entre los bases de datos más amplios están kis de las grandes editoriales:

- IEEE Xplore [19]
- ACM Digital Library [4]
- SpringerLink [31]
- ScienceDirect [12] de Elsevier

Un buen recurso también es el JSTOR (The Scholarly Journal Archive) en http://www.jstor.org/. Muchos de estos necesitan suscripción para poder descargar los artículos. Una lista de los bases de datos en los cuales tiene suscripción la UANL está en

```
http://www.dgb.uanl.mx/basededatos.php
```

y también se puede acceder contenidos de recursos en http://search.ebscohost.com/

Para encontrar citas y evaluar el impacto de una publicación científica, lo estándar es usar incides de citas. Por ejemplo http://portal.isiknowledge.com/ tiene una cita de índices muy utilizada para evaluar la calidad del trabajo científico. También se puede usar http://www.citebase.org/ para ver cuántas citas tiene un artículo; también muestra cuántas veces lo han descargado desde el sitio. El servicio de http://citeseer.ist.psu.edu/cs permite buscar por artículos y citas.

En http://libra.msra.cn/Default.aspx hay una versión aparentemente inicial de una herramienta para búsquedas académicas de citas e información bibliográfica. Un base de datos muy completo para encontrar detalles bibliográficos es

```
http://www.informatik.uni-trier.de/~ley/db/
```

También ACM Digital Library muestra citaciones entre los documentos incluidos en su base de datos; utilizando la versión "The Guide" incluye información bibliográfica de publicaciones que no son de del ACM mismo.

Para buscar por artículos en la Web, lo mejor **no** es el buscador de Google tradicional, sino

```
http://scholar.google.com.mx/
```

Capítulo 5

Gráficas

5.1. Dibujos

5.1.1. `xfig`

Xfig es un editor vectorial gráfico que se ejecuta bajo el sistema de ventanas X (en inglés, X Window System), distribuido y desarrollado libremente para plataformas compatibles con Unix. Quizá una de las primeras preguntas que deberíamos respondernos para comprender el significado de realizar nuestros gráficos en xfig, es ¿qué es un editor de gráficos vectoriales?, ¿qué es un gráfico vectorial?, ¿qué es un gráfico rasterizado? En Wikipedia encontramos las siguientes definiciones

Gráficos vectoriales: éstos gráficos también conocidos como modelados geométricos o gráficos orientados a objetos) son los que se conforman con primitivas geométricas tales como puntos, líneas, curvas o polígonos, de igual forma, son gráficos que se construyen por ordenador basándose en ecuaciones matemáticas.

Gráficos rasterizados: son aquellas imágenes que están configuradas sobre un conjunto de píxeles, tales como los bitmaps

La gráfica anterior muestra el ejemplo del efecto de un gráfico vectorial contra un gráfico rasterizado. La gráfica original se muestra a la izquierda. En la parte superior derecha se muestra una ampliación de 7× como una imagen vector. En la parte inferior derecha se muestra la misma ampliación pero utilizando una imagen bitmap. Como las imágenes rasterizadas están basadas en píxeles al ser escaladas pierden claridad, mientras que las imágenes basadas en vectores pueden ser escaladas indefinidamente sin perder nitidez. Es en este punto en donde radica la importancia de generar nuestras imágenes con un editor vectorial gráfico. Para entender mejor cómo se genera un gráfico vectorial, cuáles son las formas geométricas básicas (líneas, polígonos, círculos, elipses, curvas splines), las operaciones vectoriales permitidas (rotar, mover, estirar, unir, etcétera), ventajas y desventajas ver por ejemplo Wikipedia.

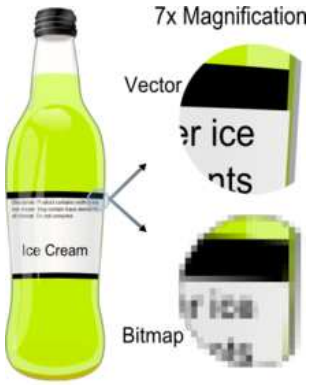


Figura 5.1: Vector vs. Bitmap (tomada de Wikipedia).

Un editor de gráficos vectoriales es un programa computacional que le permite a los usuarios componer y editar imágenes de gráficos vectoriales interactivamente en la pantalla de la computadora y guardarlas en algún formato de gráfico vectorial popular como EPS (Encapsulated PostScript), PDF, WMF (Metaarchivo de Windows) o SVG (Scalable Vector Graphics).

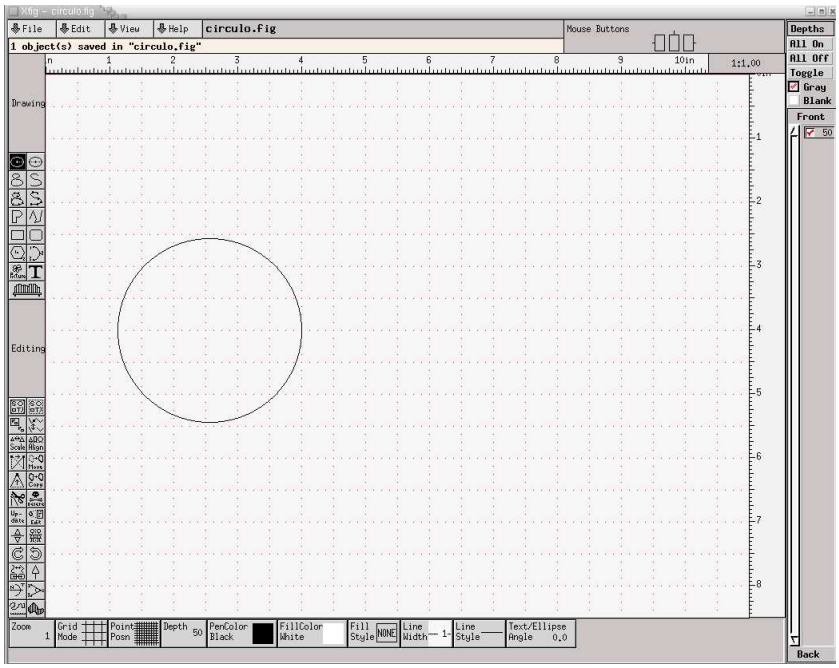
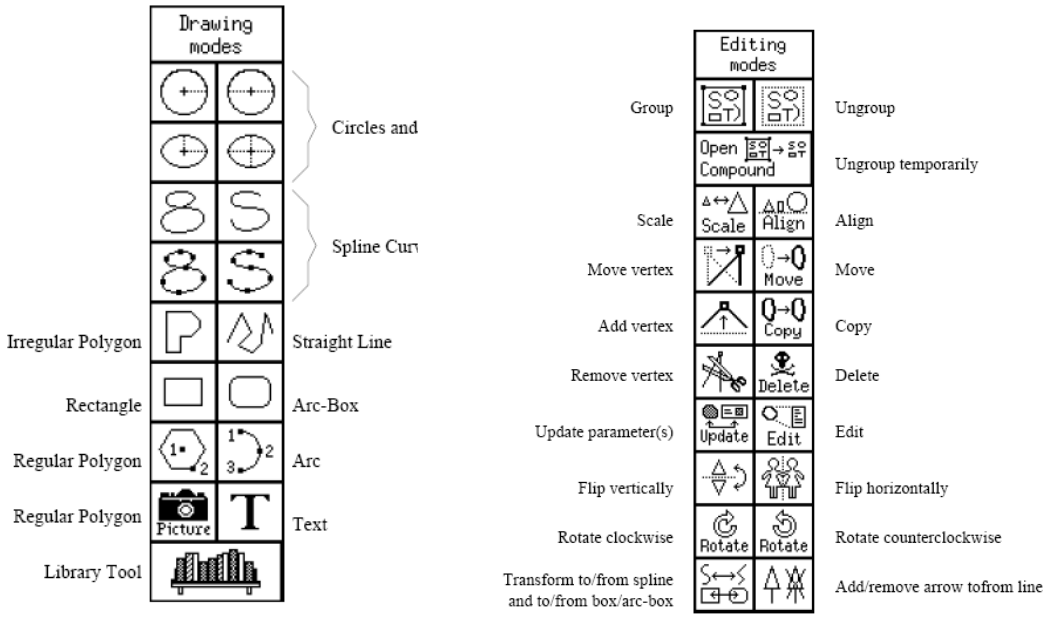


Figura 5.2: Xfig 3.2.4

Básicamente en xfig se dispone de dos conjuntos de botones. El primero de ellos es para dibujar las figuras, y el segundo para editarlas, en la figura



En la barra de menús de Xfig se encuentra una con la etiqueta *Help* en ella se puede acceder a un archivo `xfig-howto.pdf`, o bien al manual que se encuentra en `http://epb.lbl.gov/xfig/` en formato PDF. En los cuales podremos revisar la función de cada uno de los botones de creación o edición de dibujo.

Ahora nos concentraremos en generar imágenes y exportarlas a un archivo con formato EPS para insertarla en un documento de LaTeX. Una vez generada nuestra figura, en el menú File deberemos escoger la opción export, y aparecerá una ventana como la de la figura (5.3). En la primera línea de este menú, aparece otro menú llamado Language, en el podremos escoger desde EPS, PDF, GIF, JPEG, etcétera. En la parte de Output file deberemos teclear el que será el nombre de la figura. En Current Dir se indica el directorio en el que ha quedado nuestra figura. Recordemos que para insertar una figura en un archivo con formato TeX preferentemente las imágenes deben de estar en la misma carpeta, así mediante el paquete `\usepackage{epsfig}` podríamos insertar una figura así

```
\epsfig{file=pics/NombreArchivo.eps, width=5cm}
```

Xfig también permite guardar figuras en el formato Fig format, el cual es su formato nativo de sólo texto. En `http://epb.lbl.gov/xfig/fig-format.html` encontraremos una descripción del mismo. Para generar un archivo en formato FIG, en el menú File deberemos escoger la opción save as, y aparecerá una ventana como la de la figura (5.4). En Current Dir se indica el directorio en el se guardará nuestra figura.

Las siguientes 10 líneas corresponden a un archivo de formato FIG que contiene solamente una circunferencia como la que se muestra en (5.2). La primer línea contiene un comentario en el que se indica el nombre y la versión que estamos utilizando `# FIG 3.2`, en la segunda se indica la orientación del gráfico (horizontal), en la tercera se indica que aparecerá centrada, las unidades que se

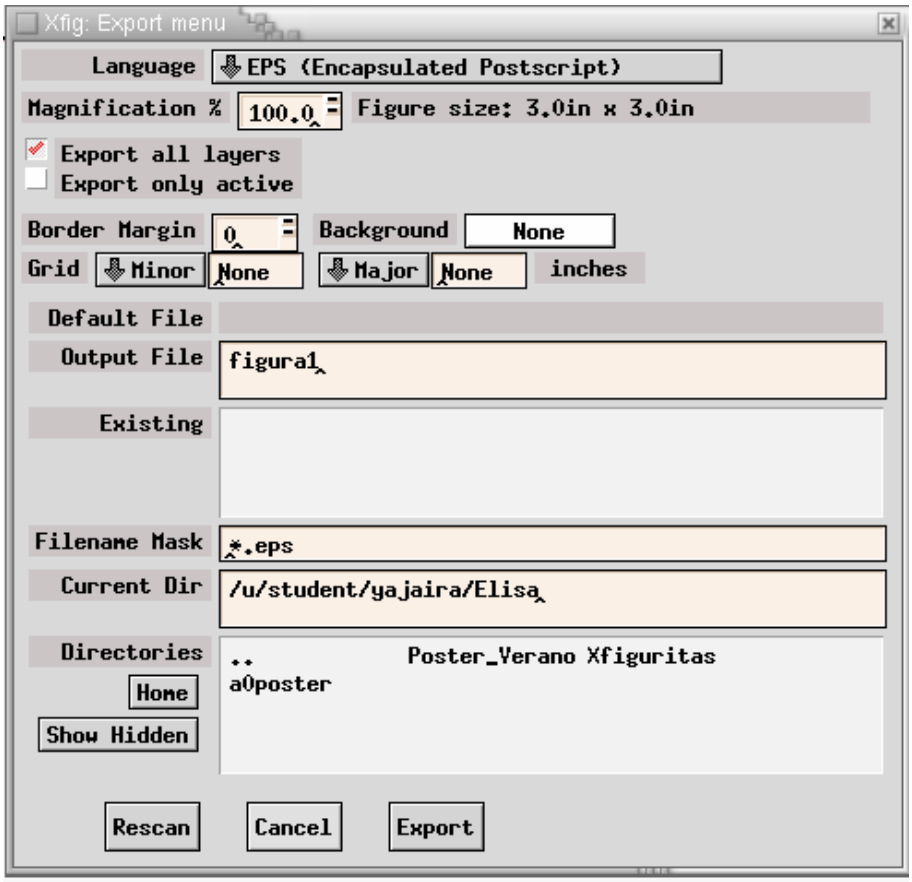


Figura 5.3: Seleccionar File→export

utilizan son pulgadas, el tamaño del papel es carta, el número 100.00 indica que se exportará e imprimirá a ese porcentaje, después se indica página individual, el número -2 indica que el nivel de transparencia para los colores es nulo, y el número 1200 indica la resolución a la que será impresa la figura (en píxeles) y el número dos al lado izquierdo de 1200 indica el sistema de coordenadas (superior izquierdo). El último renglón es el que describe el objeto que hemos creado, en nuestro caso la circunferencia. Por cada objeto que vayamos creando en nuestra figura se irán agregando líneas que las describan. El primer número que aparece indica que la figura definida es una elipse (generalización de circulo), el 3 indica que es un círculo definido por su radio, el 0 el estilo de línea (sólida), y así cada uno de los números indican características de la figura que van desde la descripción del objeto mismo, el color, el grosor de la línea, el relleno de la figura, la profundidad, el ángulo, el centro, el radio, etcétera.

```
#FIG 3.2
Landscape
Center
Inches
Letter
```

```
100.00
Single
-2
1200 2
1 3 0 1 0 7 50 -1 -1 0.000 1 0.0000 3075 4800 1727 1727 3075 4800 4800 4725
```

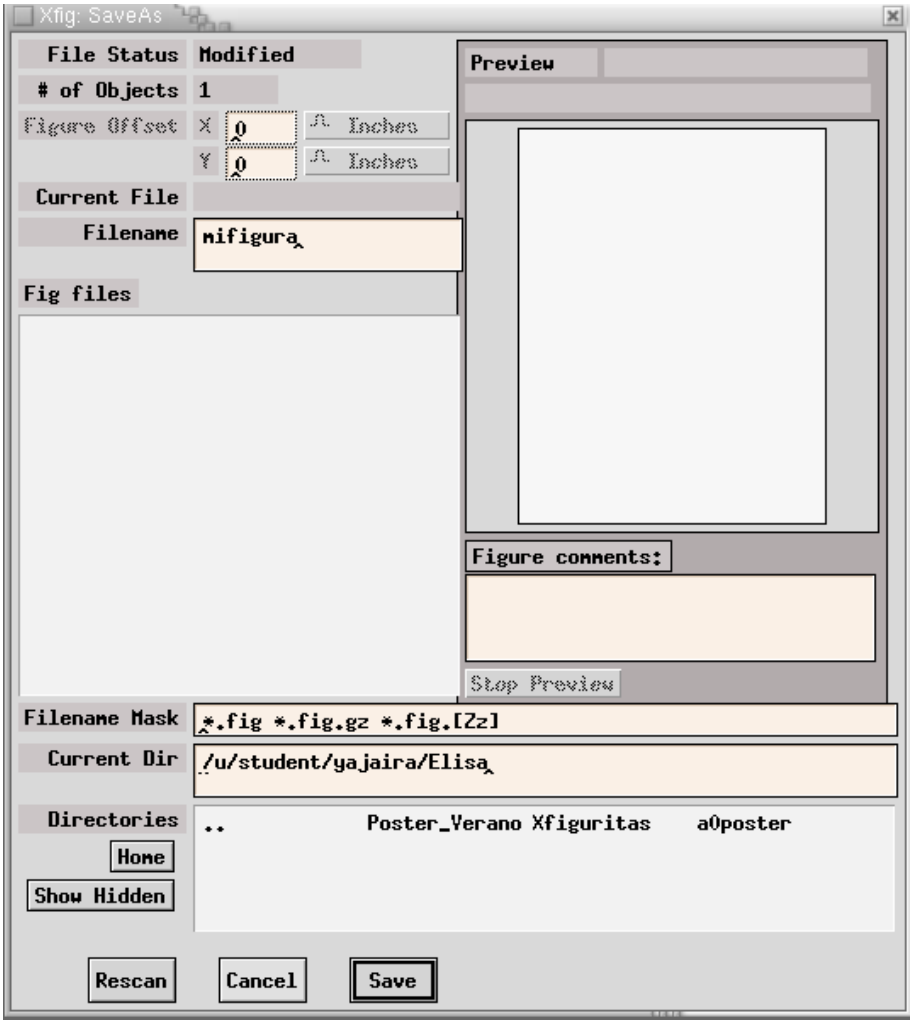


Figura 5.4: Seleccionar File→save

Para *convertir* dibujos de tipo `.fig` a otros tipos, en Unix se puede utilizar la instrucción `fig2dev` que conoce varios formatos de imágenes. Otra opción es usar una herramienta como el Gimp de la sección siguiente.

Siempre que sea necesario utilizar símbolos de latex o escribir algo en el formato de latex en texto dentro de figuras `xfig`, se puede realizar de la siguiente manera en Unix:

1. En cada elemento de texto del gráfico de xfig que se desea utilizar algún símbolo o texto bajo la sintaxis latex activar la opción: `special flag = special` y usar color de lápiz negro.
2. Escribir lo que se desee utilizando la sintaxis latex. Por ejemplo, si deseo incluir una etiqueta con el símbolo griego tau, escribir τ, o para algún valor al cuadrado, escribir x^2
3. Guardar el archivo y exportar a *Combined PS/latex*. Esto generara dos archivos, uno con extensión `.pstex_t` y otro `.pstex`.
4. Para generar una imagen con extensión .dvi crear el siguiente documento latex y compilar (incluir en la misma carpeta los archivo antes mencionados):

   ```
   \documentclass{article}
   \usepackage{epsfig}
   \usepackage{graphicx}
   \usepackage{color}
   \pagestyle{empty}
   \begin{document}
   \input{ nombre_archivo.pstex_t }
   \end{document}
   ```

   Una vez compilado este documento de latex, se procesarán los símbolos incluidos en la figura y aparecerán de la forma bonita en la que los queremos ver.
5. Finalmente se puede utilizar el formato `dvi` o convertir esta imagen a cualquier formato deseado. Para `.eps` ejecutar la siguiente instrucción: `dvips -E example.driver.dvi -o example.eps`
6. Para convertir de `.eps` a `.pdf`: `epstopdf nombre_archivo.eps`

Para incluir texto y matemáticas en formato LaTeX a dibujos de tipo `.fig`, `transfig` ayuda a automatizar este tipo de conversiones.

5.1.2. Gimp

Gimp (Gnu Image Manipulation Program) es un programa de manipulación de imágenes. Gimp está diseñado para manipular, crear o editar imágenes, ya sean fotografías o dibujos, con una variedad inmensa de extensiones para poder utilizar dentro del programa. Se puede utilizar en diferentes plataformas como Windows, Linux, Unix, etcétera.

Gimp es un programa muy práctico que nos ayudará a realizar varias tareas. Cuenta con actualizaciones gratuitas. Se recomienda que la resolución de la pantalla sea por lo menos de 1024×768, que es la resolución mínima aconsejada para utilizar este aplicación; se puede utilizar con resolución baja hasta 800×600, pero no se podrá trabajar con comodidad, ya que las ventanas a utilizar no se mostrarán todas o se mostrarán incompletas.

El programa abre con las siguientes opciones de uso: la *caja de herramientas* de la figura 5.5, opciones de herramientas de la figura 5.6, el acceso a *capas* de figura 5.7, el acceso a las *paletas* y el consejo del día (un ejemplo está en figura 5.8).

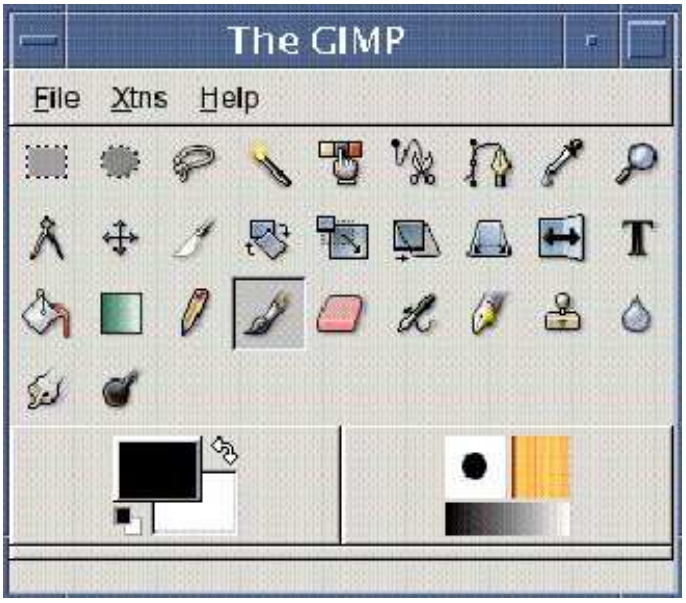


Figura 5.5: La caja de herramientas de Gimp: las opciones son selección, dibujo y transformación de imágenes. En esta ventana se encuentra las opciones en forma de dibujos para seleccionar los que se quiera hacer, desde la creación de un logo hasta editar fotografías. La parte posterior de la caja muestra dos opciones de fondo de pantalla o de la parte de enfrente de la pantalla.

El primer ejemplo del uso de Gimp es la creación de un logo. Es muy fácil con el apuntador: dirija a la caja de herramientas. Donde dice `Xnts`, da un click. Al abrirse las opciones, escoger la opción de `Script-fu`. De ahí va a `logos` y escoge el que más le llame la atención. Por ejemplo, si escoge el básico uno, dentro de la ventana que se le abre para las opciones, le dará las opciones del tamaño, color y el mensaje que le pondrá a su logo. Así tendremos hecho el primer ejemplo de cómo utilizar el Gimp.

Para *guardar* un archivo, va a archivos (inglés: files) y elija guardar (inglés: save). Ponga el nombre que le quiera dar al archivo. Gimp permite una gama extensa de tipos de archivo (file type). La extensión de `.xcf` es su propio formato.

Para crear una imagen desde la caja de herramientas, vaya al menú de archivo (files) y después elija "nuevo" (new). Aparecerá una pantalla con la siguiente información:

- Plantilla (template): los tamaño predeterminados para la imagen nueva (640×480, 800×600, etcétera).
- Tamaño de la imagen (image size): se especifica el tamaño de la imagen que se va a crear libremente por determinar la anchura (width) y la altura (heigth).
- Opciones avanzadas (advanced options): dentro de las opciones avanzadas se puede modificar o utilizar el modo de color, el fondo o agregar un comentario para la imagen.

Ya después de abrir una hoja en blanco, se puede aplicar algunas de las herramientas que se pue-

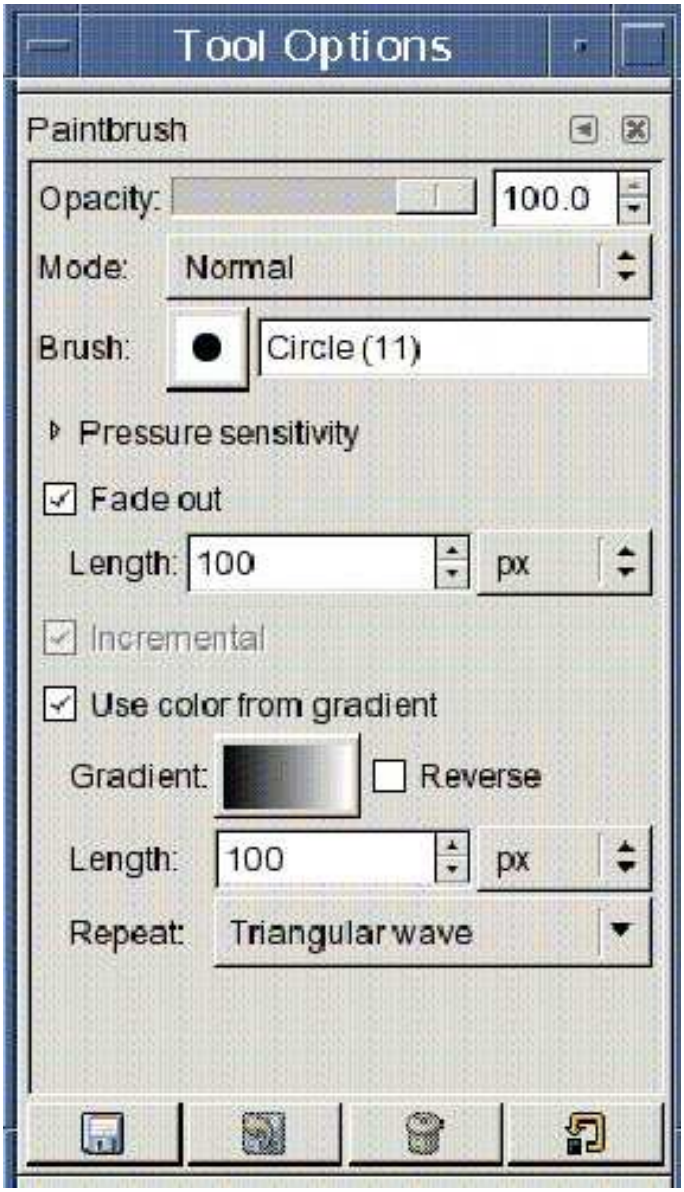


Figura 5.6: Las opciones de las herramientas de Gimp.

de utilizar para crear una imagen, mostradas en la imagen 5.5. Sus descripciones, en el orden de izquierda a derecha, de arriba hacia abajo, son

1. selecciona una área específica en forma de rectángulo la cual podemos utilizar para editar, crear o modificar una imagen
2. selecciona una parte de la imagen pero en forma de elipse y también podemos crear, modificar o editar una imagen a nuestro gusto
3. selecciona regiones dibujadas a mano

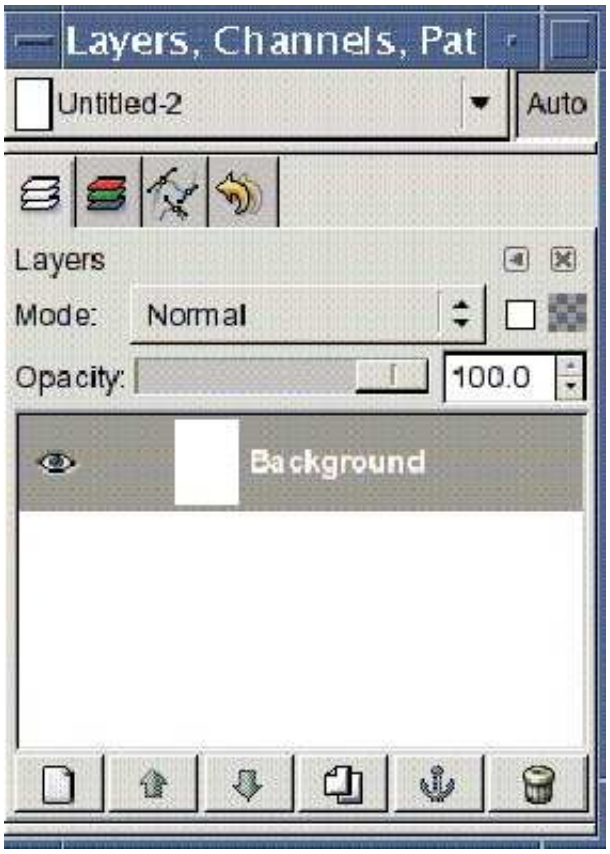


Figura 5.7: El selector de capas de Gimp.

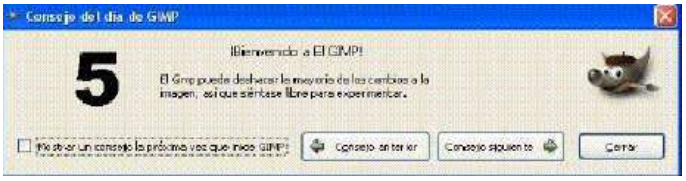


Figura 5.8: Gimp muestra opcionalmente al inicio un "consejo del día, que sirve bien para aprender más de sus funciones.

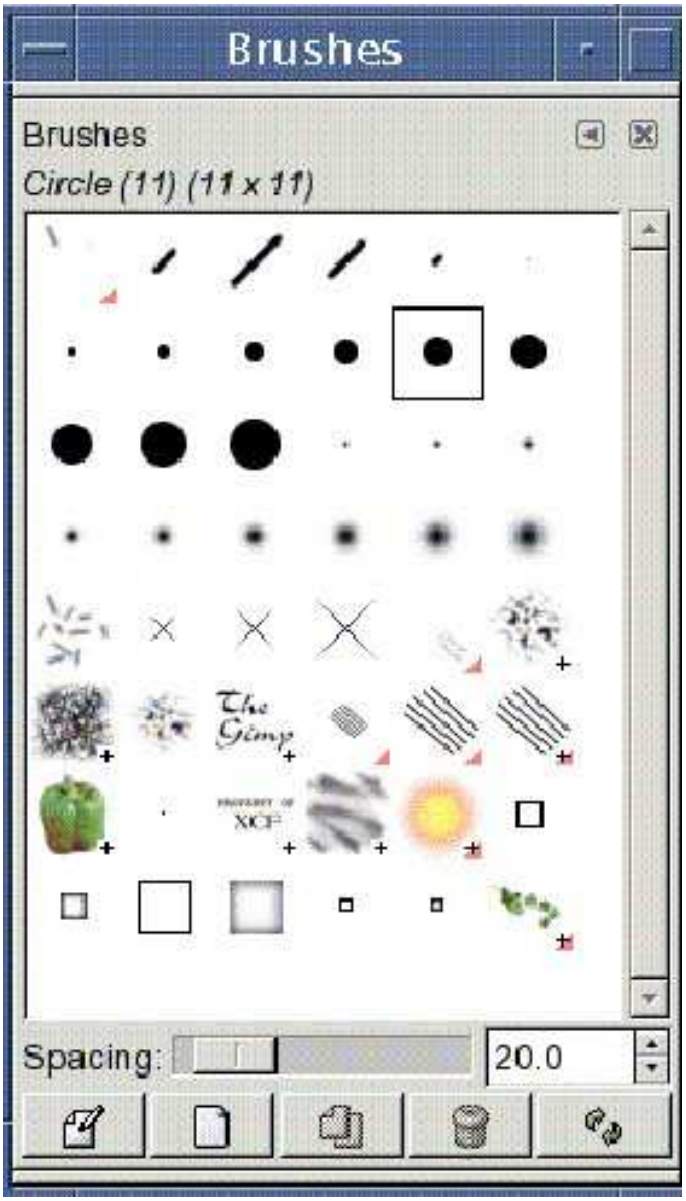
4. la *barita mágica* selecciona regiones contiguas
5. selecciona regiones por colores
6. selecciona formas de la imagen
7. crea o edita *rutas*
8. recoge colores de la imagen
9. zoom (aumenta o disminuye la imagen)
10. medidas de distancia y ángulos

11. mueve capas y selecciones
12. recorta o redimensiona la imagen
13. inclina la imagen
14. cambia la perspectiva de la capa
15. invierte la imagen o selección simétrica
16. añade texto a la imagen
17. rellena con un color o patrón un espacio en la imagen
18. rellena con un degradado de colores
19. pinta píxeles de bordes duros
20. pinta trazos borrosos de brocha
21. borra el color de fondo o la transparencia
22. aerógrafo de presión variable
23. dibuja con tinta
24. pinta usando patrones o regiones de la imagen
25. enfoca o desenfoca
26. borra la imagen
27. blanquea trazos

Como un último ejemplo, mostramos cómo *recortar* y moldear una imagen. Esta es una de las funciones más básicas y sencillas de Gimp. Lo que se tratara de hacer es sacar de una fotografía o una imagen alguna parte de ésta para poder utilizarla de diferentes formas. Lo primero que se debe hacer es abrir Gimp y en abrir seleccionamos la imagen que queremos trabajar.

Como se muestra en la imagen, hay cosas que no se ven bien. Entonces hay que modificarla para que nada más pueda salir el rostro de la persona que se encuentra en la imagen. Para lograr esto, utilizamos la opción de recortar la imagen. Esta opción se encuentra en la caja de herramientas: es la duodécima opción en figura 5.5. Una forma más rápida de obtenerla es oprimiendo las teclas Shift y "C": automáticamente se abrirá la opción de recortar la imagen.

Ya teniendo la opción habilitada, con el ratón daremos un click en una esquina de la zona a recortar y, manteniendo pulsado el botón izquierdo del ratón, lo arrastramos en diagonal para formar un marco de selección: la imagen a recortar se marcará del resto de la imagen mostrándole la zona seleccionada. Cuando ya se tenga la forma o la imagen deseada, daremos un click para recortarla y damos otro click al ratón para terminar el proceso del recorte.



5.2. Diagramas

5.2.1. Dia

Este programa nos sirve para hacer gráficos o diagramas de una forma fácil y sencilla ya que su interfaz es muy amigable y muy entendible para cualquier persona quien ha utilizado con anterioridad programas como el Gimp.

Para abrir este programa desde una terminal en la línea de instrucciones, escriba `dia` y inmediatamente se abrirá el programa con dos pantallas con las que vamos a trabajar. La primera es la ventana

Figura 5.9: A la izquierda, la foto original. A la derecha, una versión cortada.

del editor de diagramas (figura 5.10) y la segunda es la área donde se va a trabajar los diagramas (figura 5.11).

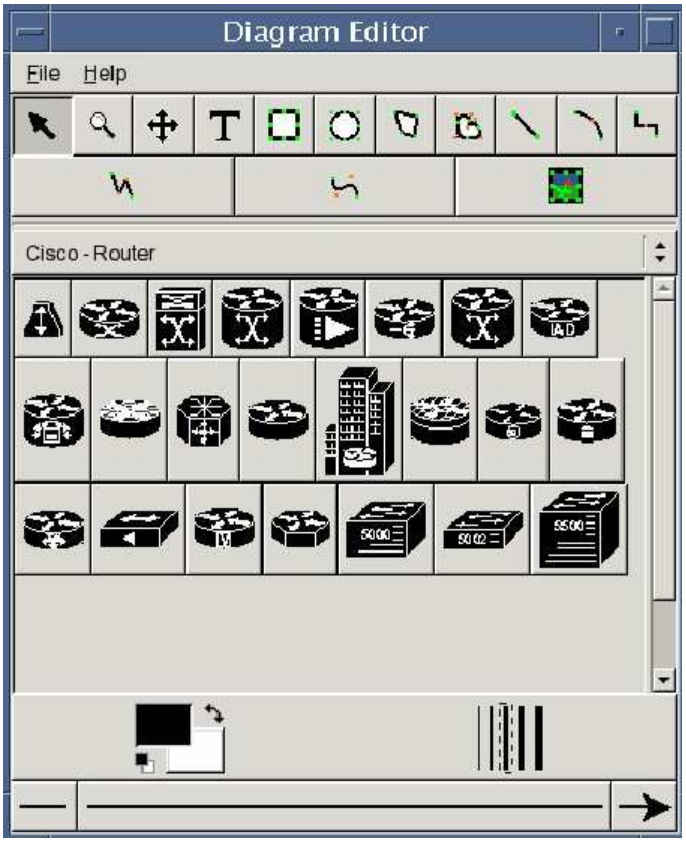


Figura 5.10: Editor de diagramas de Dia.

Dentro de la ventana del editor, en la parte inferior de la misma, hay dos opciones a escoger: la de archivo y la de ayuda. En archivo puede abrir una nueva hoja para editar una gráfica o guardar alguna hecha, entre otras actividades, y en ayuda puede encontrar desde Internet la ayuda, si no

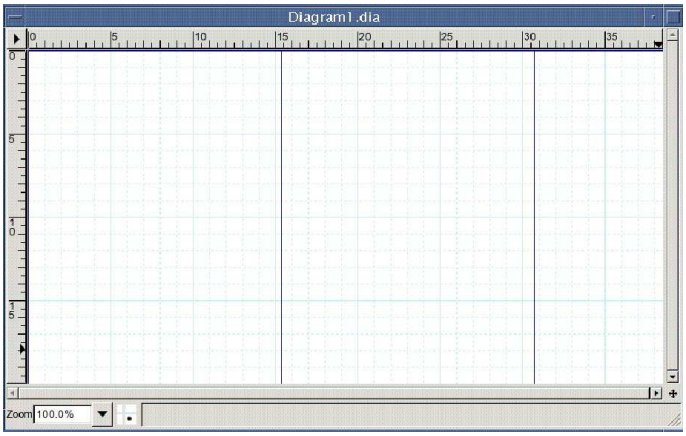


Figura 5.11: Composición de diagramas en Dia.

entiende algo con respecto al programa.

Es muy fácil utilizar este programa ya que sólo ocupa arrastrar con el ratón la imagen que quiera desde la ventana del editor de gráficas hacia la hoja de diagrama. Con las diferentes opciones que hay en el editor puede hacer muchas combinaciones

En la primera fila de la ventana del editor de gráficas hay once opciones para utilizar dentro de la hoja del diagrama:

1. Arrastrar algún objeto utilizado dentro de la gráfica.
2. El zoom.
3. Arrastrar toda la gráfica, no solamente un objeto como lo hace la primera opción.
4. Agregar texto a algún objeto en específico.
5. Agregar un cuadro al diagrama (se puede moldear al gusto).
6. Agregar una elipse al diagrama (se puede moldear al gusto).
7. Agregar un triángulo al diagrama (se puede moldear al gusto).
8. Agregar un círculo al diagrama (se puede moldear al gusto).
9. Agregar una línea recta para unir dos o más objetos.
10. Agregar una línea curvada para unir dos o más objetos.
11. Agregar una línea que puede quebrar hacia abajo o hacia arriba para unir dos o más objetos.

Estas son las opciones más básicas para poder elaborar algún diagrama sencillo. A continuación vienen tres opciones similares a las de arriba pero son más específicas y más prácticas:

1. Una línea muy similar a la línea recta con la diferencia de que le puede agregar más puntos para poder *quebrarla* dándole un click al botón de en medio del ratón para agregar punto. Se puede repetir para agregar varios puntos según sea la necesidad.
2. Una línea similar a la línea curvada con la diferencia que uno maneja la curva que quiere darle a la línea.
3. Agregar una imagen, ya sea una foto o algún dibujo de algo que necesite poner en el diagrama.

Después de estas opciones, hay una barra donde puede escoger diferentes tipos de objetos o figuras. Al darle un click, se despliega una lista de opciones que puede utilizar dentro de la diagrama. Son alrededor de treinta opciones, donde al escoger alguna, en la parte posterior de el editor de diagramas aparecerán las opciones que activaste en ese momento.

Como un ejemplo del uso de Dia, preparamos un diagrama de la topología de una configuración de ruteadores and switch de Cisco. Para hacer esta práctica, hay que conocer algunas imágenes de los routers y los switch que son los que nos dan la red normalmente en un edificio de trabajo donde se encuentre una red internet. La topología del ejemplo es muy básica, ya que sólo vamos a ocupar lo siguiente: dos router, un switch y dos computadoras, conectando la primera computadora al switch, el switch al primer router, el primer router al segundo y el segundo router a la segunda computadora.

Lo primero que hay que hacer es abrir un diagrama nuevo desde el editor y buscar en las opciones donde dice “Assorted”. Ahí le damos un click para que se abran las opciones y buscamos “cisco-computer”. Escoger dos computadoras — no importa el modelo en esta práctica.

Después en la misma opción le das click para buscar ahora “cisco-hub”. para escoger un switch. También se puede escoger cualquiera al gusto. Por último en la misma opción hacer click en “cisco-router” y escoger dos routers del mismo tipo — pueden ser de cualquier serie, nada más que sean del mismo.

Ya teniendo puestos los objetos sobre la hoja de diagrama, nos vamos a la primera línea de opciones, donde vamos a pegar con líneas los diferentes dispositivos de la forma indicada. Cada uno de los objetos tiene diferentes formas de unirse:

- La primera computadora se une con una línea sencilla al switch.
- El switch se une con una línea sencilla al primer router.
- Los router se unen con una polilínea, haciendo un quebrado en esta línea.
- El primer router se une a la primera computadora con una línea curveada.
- El segundo router se una a la segunda computadora con una línea sencilla.

Ya si quiere que se vea más estética, agregue unos cuadros que se encuentran en la primera línea de opciones del editor y dele nombre al objeto. También dentro de esa misma opción puede ponerle título al ejercicio, como se hizo en el ejemplo — el resultado está en la figura 5.12. Una opción muy útil es darle doble click al objeto que quiera para cambiarle el estilo de letra o de color.

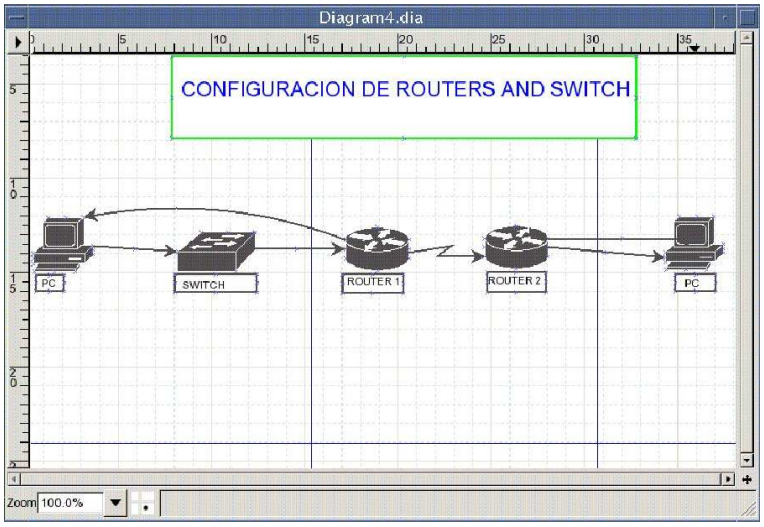


Figura 5.12: Un ejemplo de diagrama hecho por Dia con componentes de una red tipo Cisco.

5.2.2. Gnuplot

Gnuplot es un programa gratuito y programable ideal para graficar funciones y datos. Una buena manera de empezar a trabajar con gnuplot será meterse a su cuenta y crear una carpeta en la cual creará y guardará los documentos necesarios (`.txt`, `.eps` y `.plot`) para el funcionamiento de Gnuplot. En este escrito llamaremos a esta carpeta `gnuplot/`. Una vez que tenga su carpeta, debe entrar a ella y crear un documento del tipo `mi_grafica.plot` para lo cual le recomendamos utilice un buen editor de textos como el `emacs`. Para empezar, veremos como puede graficar algo simple; el seno y el coseno de x por ejemplo. Entra a `mi_grafica.plot` y escribe

```
set term postscript eps
set xrange [1:15]
set yrange [-4:4]

plot cos(x) title 'El coseno de x' with points pointtype 7,\
     sin(x) title 'El seno de x' with lines linetype 2,\
     (cos(x)+sin(x))/2 title 'El promedio' with linespoints pointtype 2 linetype 3
```

Guarde `mi_grafica.plot` y cierre el archivo. Abra una terminal y de la instrucción `cd ~/gnuplot/`.

Al ejecutar, Gnuplot crea un documento en el cual pueda ver el gráfico. Aquí le recomendamos utilizar "Encapsulated PostScript" conocido por su acrónimo `eps`. Con la línea `set term postscript eps` pedimos a Gnuplot a crear un documento de este tipo. Para ejecutar Gnuplot, se debe usar la línea de instrucciones y escribir `gnuplot mi_grafica.plot > mi_grafica.eps` — si todo sale bien, su programa no reportará errores. Para visualizar el gráfico que ha creado necesita el programa *Ghostview* (un software gratuito). Escriba la instrucción `gv mi_grafica.eps &` para mostrar el gráfico que ha creado.

Si ha seguido al pie de la letras estas instrucciones, su gráfico debe ser igual al de la figura 5.13. El gráfico debe estar en el rango $[1, 15]$ para las x y $[-4, 4]$ para las y y usar puntos gruesos para graficar

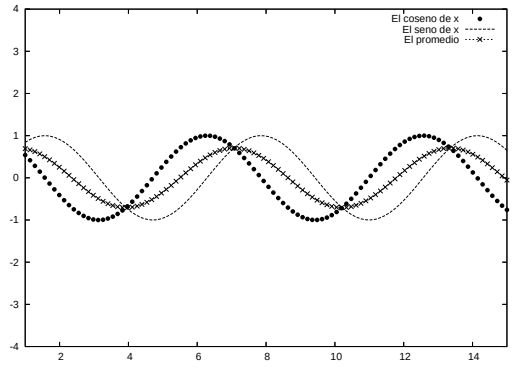


Figura 5.13: Un ejemplo de Gnuplot con funciones matemáticas.

el coseno de x, líneas finamente trozadas para el seno de x y los símbolos $\times$ unidos por segmentos de línea recta para graficar el promedio. Si queremos darle color a las líneas y puntos debemos agregar la palabra `color` al final de la primera línea del programa. Si queremos utilizar diferentes tipos de línea y punto basta con cambiar el número a la derecha de `linetype` y `pointtype`.

Para más detalles, puede consultar diferentes manuales de gnuplot como el que aparece en la dirección `http://www.duke.edu/~hpgavin/gnuplot.html` — ahí encontrará muchas otras funciones matemáticas que puede graficar.

Digamos que queremos hacer una gráfica con los siguientes datos:

```
# mes inf prnstc sup
1 35,200 39,929 44,657
2 34,883 39,611 44,339
3 32,732 37,460 42,187
4 33,446 38,173 42,901
5 33,450 38,177 42,905
6 35,079 39,806 44,534
7 37,501 42,229 46,956
8 34,710 39,437 44,165
9 33,650 38,378 43,105
10 36,328 41,056 45,784
11 44,441 49,169 53,897
12 56,718 61,447 66,175
```

que son los pronósticos (`prnstc`) para las ventas nacionales de vehículos subcompactos por mes con intervalos de confianza inferiores (`inf`) y superiores (`sup`). Para empezar, debemos guardar el archivo en un documento de texto (nombrándolo por ejemplo `.txt` o `.dat`). Digamos `pronosticos_autos.txt`. Ahora, la manera de crear un programa que grafique todos los datos del documento se muestra a continuación

```
set term postscript eps 20

plot 'pronosticos_autos.txt' using 1:2 title 'Inferiores' with linespoints, \
     'pronosticos_autos.txt' using 1:3 title 'Pronostico' with linespoints, \
     'pronosticos_autos.txt' using 1:4 title 'Superiores' with linespoints
```

Guarda el documento y sigue los mismos pasos del primer ejemplo. Si todo sale bien, su gráfico se debe ver como el de la figura 5.14.

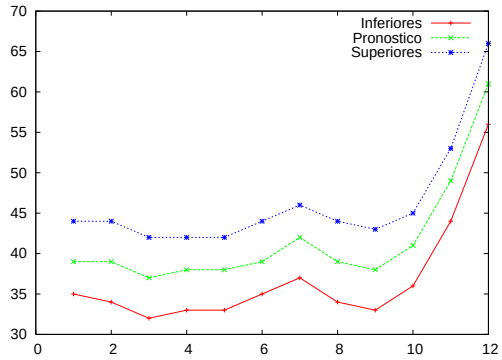


Figura 5.14: Un ejemplo de Gnuplot con datos de un archivo.

A continuación se muestra un ejemplo de una serie de instrucciones para hacer “el volcán” de la figura 5.15.

```
set term postscript enhanced eps 20 # enhanced para notación tipo LaTeX
set size 1, 1  # cambia el tamaño de la imagen
set style data lines  # une los puntos graficados
set hidden3d  # oculta las líneas del fondo
set view 60, 30, 1, 1  # determina el ángulo del gráfico

set xlabel 'Eje x'
set ylabel 'Eje y'
set zlabel 'Eje z'
set isosamples 25,25;  #es el número de puntos a graficar por eje

set xrange [-2.5:2.5]
set yrange [-2.5:2.5]
set zrange [0:1]

splot 2*(x**2+y**2)*exp(-(x**2+y**2)) title "2(x^2 + y^2) e^{- (x^2 + y^2)}"
```

Gnuplot es capaz de utilizar diversos sistemas de coordenadas para realizar gráficas. Uno de ellos es el sistema de coordenadas esféricas. A continuación se muestra un ejemplo de lo que podemos hacer con este sistema de coordenadas esféricas:

```
set terminal postscript eps 20
set dummy u,v
set angles degrees
set parametric
set view 60, 50, 1.0, 1.3
set samples 32, 32
set isosamples 30, 30
set mapping spherical
set yzeroaxis lt 0 lw 1.000
set ticslevel 0
set noxtics
set noytics
```

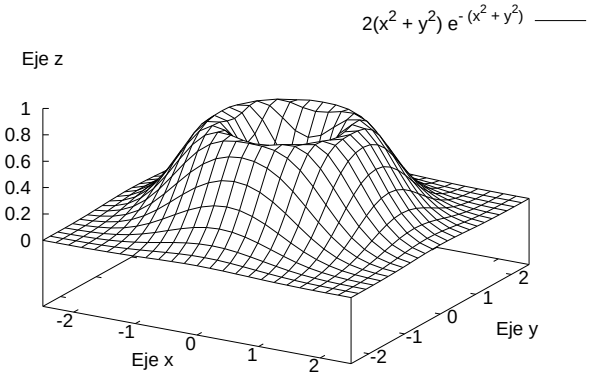


Figura 5.15: Un ejemplo de Gnuplot en tres dimensiones.

```
set noztics
set urange [-90:90] noreverse nowriteback
set vrange [0:360] noreverse nowriteback
set pointsize 1.5

set hidden3d offset 1 trianglepattern 3 undefined 1 altdiagonal bentover

set title "La tierra en 3D"
set urange [-90:90] noreverse nowriteback
set vrange [0:360] noreverse nowriteback
splot cos(u)*cos(v), cos(u)*sin(v), sin(u) notitle with lines lt 2, \
 'world.dat' notitle with lines lt 1 lw 2.5, \
 'mtynl.dat' title "MTY, NL" with points pointtype 7 linetype 4
```

Antes de compilar el programa, asegúrese de crear un archivo `mtynl.dat` con los contenidos

```
# Monterrey, Nuevo León
-101 27
-101 27
```

y copiar el archivo `world.dat` de la carpeta `/usr/local/doc/gnuplot/demo/` (dependiendo de la ubicación de la instalación) por ejecutar la siguiente instrucción en la carpeta donde desea colocar la copia:

```
cp /usr/local/doc/gnuplot/demo/world.dat .
```

En la Figure 5.16 se muestra un mapa de la tierra en el cual se indica con un punto el sitio que ocupa la ciudad de Monterrey en el estado de Nuevo León.

Gnuplot también puede usarse para la construcción gráfica de algunas herramientas de la calidad como se muestra a continuación.

En estadística descriptiva, un diagrama de *caja y bigote* es una manera conveniente de describir gráficamente un compendio de cinco errores que consisten en la observación más pequeña, el cuartil

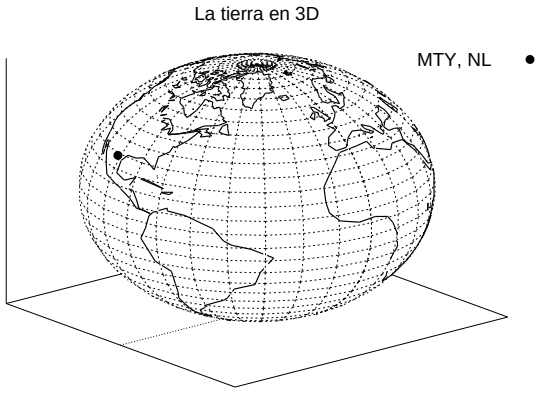


Figura 5.16: Un ejemplo de Gnuplot con coordinadas esféricas.

menor, la media, el cuartil superior y la observación más grande. A continuación se muestra como programar un diagrama de caja y bigote

```
set term postscript eps color
set boxwidth 0.9 absolute # determina el grosor de las cajas
set title "Diagrama de caja y bigote"
set xrange [ 0.00000 : 25.0000 ]
set yrange [ 0.00000 : 30.0000 ]
plot 'caja_y_bigote.dat' using 1:3:2:6:5 with candlesticks lw 2 notitle, \
# lee el valor que abre la caja, luego el dato menor y el mayor
# y por último el cierre de caja
  'caja_y_bigote.dat' using 1:4:4:4:4 with candlesticks lt -1 notitle
# grafica el valor que está dentro de la caja
```

En la figura 5.17 se muestra la gráfica correspondiente a este programa. Es recomendable acomodar el dato menor en la segunda columna e ir en forma creciente hasta la última. Por la naturaleza del diagrama con caja y bigote, debe haber seis columnas, siendo la primera la del eje de las abscisas.

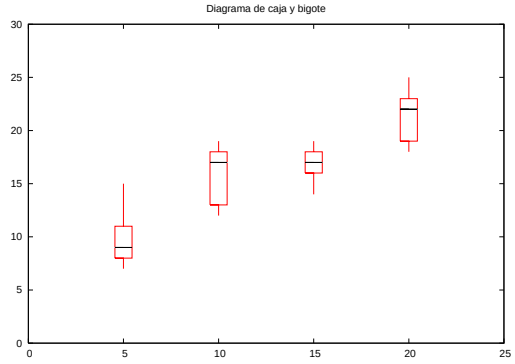


Figura 5.17: Una gráfica caja-bigote con Gnuplot.

En estadística, un *histograma* es una representación gráfica de una variable en forma de barras, donde la superficie de cada barra es proporcional a la frecuencia de los valores representados. En el eje vertical se representan las frecuencias y en el eje horizontal los valores de las variables, normalmente señalando las marcas de clase, es decir, la mitad del intervalo en el que están agrupados los datos. Se utiliza cuando se estudia una variable continua, como franjas de edades o altura de la muestra, y, por comodidad, sus valores se agrupan en clases, es decir, valores contiguos. En la figura 5.18 se muestra un histograma hecho a partir de la función

$$\frac{\exp \frac{-x^2}{2}}{\sqrt{2\pi}}. \tag{5.1}$$

```
set term postscript eps color
set encoding iso_8859_1 # sirve para incluir acentos
set boxwidth 0.5 absolute
set style fill solid 1.000000 border -1 # sirve para rellenar las barras
set samples 13 # determina cuantas barras serán graficadas
set title "Histograma de una aproximación de la distribución normal estándar"
set yrange [ 0.00000 : 0.4500 ]
plot [-3:3] exp(-x*x/2)/sqrt(2*pi) notitle with boxes
```

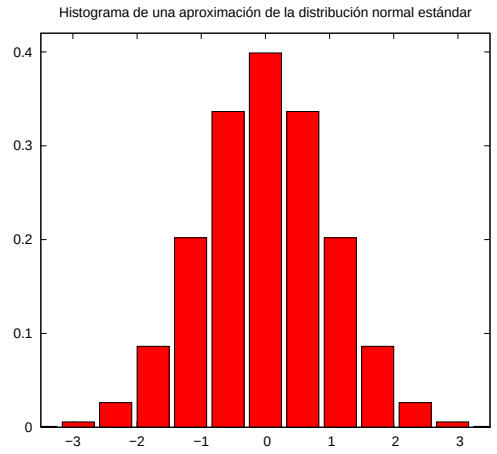


Figura 5.18: Un histograma con una función de Ec. 5.1.

Un *diagrama de Pareto* es una gráfica de barras para datos de conteo. Presenta la frecuencia de cada conteo en el eje vertical y el tipo de conteo o clasificación sobre el eje horizontal. Siempre arreglan los tipos de conteo en orden descendente de frecuencia u ocurrencia; esto es, el tipo que ocurre con mayor frecuencia esta a la izquierda. Seguido por el tipo que ocurre con la siguiente mayor frecuencia, y así sucesivamente. Hacer un diagrama de Pareto con gnuplot es muy similar a hacer un histograma o cualquier otro diagrama de barras. En la figura 5.19 se muestra un diagrama de Pareto hecho para conocer cuál es la mayor razón de los tiempos muertos en cierta fabrica. El programa se muestra a continuación.

```
set term postscript eps color
set boxwidth 0.5 absolute
set style fill solid 0.250000 border
```

```
set ylabel 'Porcentaje de tiempo muerto'
set title "Ejemplo de un diagrama de Pareto"
set yrange [ 0.00000 : 42.000 ]
set xrange [ 0.00000 : 6.000 ]

# las etiquetas del eje x
set xtics   ("Falla de\n colorimetro" 1.0, "Reactivos" 2.0, \
  "Tubo\n deformado" 3.0, "Falla de\n electrodo" 4.0,"Otros" 5.0)
set bmargin 3 # espacio extra para las etiquetas

plot 'pareto.dat' using 1:2 notitle with boxes
```

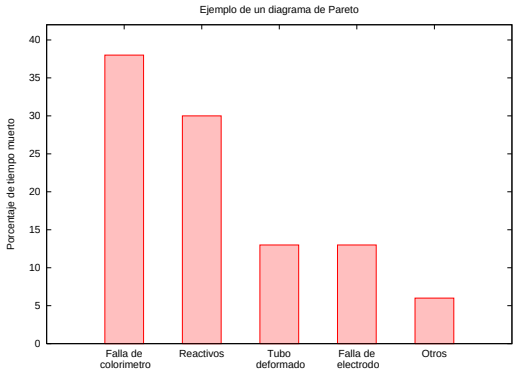


Figura 5.19: Diagrama de Pareto.

El sitio en la red en el cual puedes obtener más información y ejemplos sobre este y otros tipos de gráficas es `http://gnuplot.sourceforge.net/demo_4.0/` (para aprender `gnuplot` bien, es recomendable copiar de algunos ejemplos y realizar modificaciones para ajustar el resultado a ser lo deseado). Una guía muy completa sobre `gnuplot` que se puede obtener en `http://www.gnuplot.info/docs/gnuplot.pdf`.

Capítulo 6

Programación

6.1. Programación imperativa y orientada a objetos

6.1.1. C y C++

Los programas escritas en C son programas imperativos. C es un lenguaje que permite el desarrollo rápido de programas pequeñas bastante eficientes que no necesitan interacción fuerte en forma gráfica con el usuario.

Libros gratuitos disponibles en HTML o PDF en línea incluyen los de [26] y Eckell [9, 11].

```
> gcc -v Reading specs from
> /usr/local/lib/gcc/sparc-sun-solaris2.9/3.4.2/specs Configured with:
> ../configure --with-as=/usr/ccs/bin/as --with-ld=/usr/ccs/bin/ld
> --disable-nls Thread model: posix gcc version 3.4.2
```

mientras `/usr/local/bin/g++` es simplemente un alias para `gcc` con algunas librerías adicionales incluidas en el proceso de crear el ejecutable:

```
> g++ -v Reading specs from
> /usr/local/lib/gcc/sparc-sun-solaris2.9/3.4.2/specs Configured with:
> ../configure --with-as=/usr/ccs/bin/as --with-ld=/usr/ccs/bin/ld
> --disable-nls Thread model: posix gcc version 3.4.2
```

De hecho, `gcc` no es un sólo compilador, pero un conjunto de compiladores. Desde su página `http://gcc.gnu.org/` uno puede descargar otras herramientas útiles también.

En la página `http://www.thefreecountry.com/compilers/cpp.shtml` hay una lista larga de compiladores gratuitos para C y C++. El uso de *librerías* adicionales es común en los programas escritos en C o C++. Un ejemplo de tales librerías es BOOST [5].

Para fines de demostración, incluyamos un programa muy pequeña que no hace nada muy impresionante: para intentar a compilar, ejecutar y modificar el programa, se supone que lo tengan guardado en un archivo `cprog.c`:

```
#include <stdio.h>

int main(int argc, void* args) {
  int i, prev = 0, curr = 1, temp;
  printf("Hello, world!\nF(1-10): 0 ");
  for (i = 0; i < 10; i++) {
    temp = curr;
    curr = prev + curr;
    prev = temp;
    printf("%d ", curr);
  }
  printf("\nBye!\n");
  return 0;
}
```

Al compilar, por defecto el ejecutable creado reside en el archivo `a.out`, pero con la opción `-o` se puede dirigirlo directamente bajo otro nombre:

```
> gcc cprog.c
> ./a.out
Hello, world!
F(1-10): 0 1 2 3 5 8 13 21 34 55 89
Bye!
> gcc -o fibo cprog.c
> ./fibo
Hello, world!
F(1-10): 0 1 2 3 5 8 13 21 34 55 89
Bye!
>
```

Los programas de C++ están compuestas por clases, mientras los de C están estructurados en subrutinas. En ambos lenguajes, se suele separar definiciones de implementación. En C, las definiciones de pone en archivos tipo `.h` y en C++ en archivos tipo `.hpp`. El código de C se pone en archivos `.c` y lo de C++ en archivos tipo `.cpp`.

Como un ejemplo simple de un programa en C++, tenemos el archivo siguiente de definiciones bajo el nombre `cuenta.hpp`:

```
class CuentaBancaria {

private:
  double balanceActual;

public:
  CuentaBancaria();
  double balance();
  void depositar(double monto);
  int retirar(double monto);
};
```

y el código siguiente en `cppprog.cpp`:

```
#include <iostream>
#include "cuenta.hpp"

using namespace std;

CuentaBancaria::CuentaBancaria() {
  balanceActual = 0.0;
}

double CuentaBancaria::balance() {
  return balanceActual;
}

void CuentaBancaria::depositar(double monto) {
  balanceActual += monto;
  return;
}

int CuentaBancaria::retirar(double monto) {
  if (balanceActual >= monto) {
    balanceActual -= monto;
    return 0;
  }
  return -1;
}

int main(int argc, void* args) {
  CuentaBancaria* bc = new CuentaBancaria();
  cout << "Cuenta nueva con balance de " << bc->balance() << " pesos." << endl;
  cout << "Depositando 500 pesos..." << endl;
  bc->depositar(500.0);
  cout << "Balance actual: " << bc->balance() << endl;
  bc->retirar(200.0);
  cout << "Balance actual: " << bc->balance() << endl;
  delete bc;
  return 0;
}
```

Al compilar y ejecutar, obtenemos lo siguiente:

```
> g++ cppprog.cpp
> ./a.out
Cuenta nueva con balance de 0 pesos.
Depositando 500 pesos...
Balance actual: 500
Balance actual: 300
> g++ -o banco cppprog.cpp
> ./banco
Cuenta nueva con balance de 0 pesos.
Depositando 500 pesos...
Balance actual: 500
Balance actual: 300
>
```

6.1.2. Java

Java es un lenguaje orientado a objetos que tiene mucho en común con C y C++. En Java, ya muchas de las operaciones "básicas" de C/C++ se hace con objetos.

Se usa la instrucción`/bin/javac` para compilar y la instrucción `/bin/java` es para ejecutar programas ya compilados escritos en Java.

El siguiente ejemplo es prácticamente lo mismo que lo de C++ de la sección anterior; escribimos todo en un sólo archivo `CuentaBancaria.java`:

```
public class CuentaBancaria {
    private double balanceActual;

    public CuentaBancaria() {
        this.balanceActual = 0.0;
    }

    public void depositar(double monto) {
        this.balanceActual += monto;
        return;
    }

    public boolean retirar(double monto) {
        if (this.balanceActual >= monto) {
            this.balanceActual -= monto;
            return true;
        } else {
            return false;
        }
    }

    public String toString() {
        return ("Balance actual: " + this.balanceActual);
    }

    public static void main(String[] args) {
        CuentaBancaria b = new CuentaBancaria();
        System.out.println("Nueva cuenta creada.");
        b.depositar(500.0);
        System.out.println(b);
        b.retirar(200.0);
        System.out.println(b);
        return;
    }
}
```

y al compilar y ejecutar obtenemos:

```
> javac CuentaBancaria.java
> java CuentaBancaria
Nueva cuenta creada.
Balance actual: 500.0
Balance actual: 300.0
>
```

El fuente "estándar" de documentación y versiones de Java para descargar para instalar es `http://java.sun.com/`.

Eckel tiene también un libro gratuito sobre Java [10]. En cualquier caso, lo más importante es el Java API en `http://java.sun.com/j2se/1.4.2/docs/api/` (para la versión que se usa).

6.2. Lenguages "script"

Los lenguajes script son lenguajes de programación que no necesitan compilación previa a ejecución.

En la actualidad definitivamente conviene echar un vistazo a `python`, `https://www.python.org/`.

6.2.1. awk

awk es un lenguaje tipo script de programación que es fácil de combinar con las herramientas existentes de Unix.

La área fuerte de awk es realizar modificaciones a archivos. En esta sección solamente revisamos lo más básico sobre awk, ya que es un lenguaje bastante rico. Toda la sintaxis de awk está sensitiva a minúsculas y mayúsculas. awk se puede utilizar en cualquier sistema operativo de Unix moderno. Los programas escritos en awk se puede ejecutar directamente de la línea de instrucciones o alternativamente desde un archivo. Ni el nombre ni la terminación de este archivo tiene importancia.

awk procesa sus datos de entrada línea por línea, dividiendo cada línea en "columnas". El separador por defecto es todo tipo de espacio, pero por modificar el valor de la variable FS (inglés: field separator) se puede cambiar este valor. El primer ejemplo logra que se imprima solamente la primera columna de los datos de entrada en la pantalla: `awk '{printf $1}' fichero` donde `fichero` es el nombre del archivo de datos de entrada. También se puede enviar datos desde otro programa: en el ejemplo siguiente utilizamos como entrada la salida de `ls -l`,

```
> ls -l
total 6620
drwx------   2 elisa    faculty      512 Mar  2 13:56 Mail
drwxr-xr-x   5 elisa    faculty      512 May 10 16:21 OpenOffice.org1.1.4
-rw-r--r--   1 elisa    faculty   281765 Jan 30 12:48 PID295149.pdf
-rw-r--r--   1 elisa    faculty  3049984 Jan 24 14:11 SemillaBarros2006.doc
drwxr-xr-x   3 elisa    faculty      512 Jun 14 12:01 admin
drwx------   4 elisa    faculty     2048 Jun 14 12:00 backup
drwxr-xr-x   4 elisa    faculty      512 Jun 14 17:36 cvs
drwxr-xr-x   5 elisa    faculty      512 Jun 14 17:36 cvsroot
drwxr-xr-x   2 elisa    faculty      512 May  9 13:08 docs
drwx------   4 elisa    faculty     1024 Jun  4 14:57 info
drwx------   2 elisa    faculty     2560 Jun 15 10:45 mail
drwx------   2 elisa    faculty      512 Mar  2 13:57 nsmail
drwx------   5 elisa    faculty      512 Nov 28  2006 other
drwxr-xr-x   2 elisa    faculty     1536 May 28 11:04 papers
drwxr-xr-x   2 elisa    faculty      512 Jun  4 14:58 pedro
drwx------   2 elisa    faculty      512 Jun 14 12:01 pics
drwxr-xr-x  13 elisa    faculty      512 Jun  1 15:08 public_html
drwxr-xr-x   2 elisa    faculty      512 Jun 14 17:36 temp
drwxr-xr-x   2 elisa    faculty      512 May  9 13:07 tesis
drwx------   2 elisa    faculty     7168 Sep  6  2006 todo
```

```
drwx------  13 elisa    faculty     2048 Jun 15 12:03 work
```

y elegimos únicamente la columna de los tamaños de los archivos:

```
> ls -l | awk '{print $5}'

512
512
281765
3049984
512
2048
512
512
512
1024
2560
512
512
1536
512
512
512
512
512
7168
2048
>
```

Con el uso de variables, podemos también sumar estos valores para poder imprimir el tamaño total de los archivos. La notación de operaciones aritméticas es muy parecido a varios otros lenguajes como C y Java: `a + b` es una suma de las variables `a` y `b`, mientras `a * b` es su producto, etcétera.

Con este ejemplo, el largo de la línea de instrucciones ya se pone excesivo, por lo cual preparamos este ejemplo en un archivo:

```
BEGIN {total = 0}
{print $5; total = total + $5}
END {print "Total = " total}
```

Todo el código en el bloque `BEGIN {...}` se ejecuta una vez antes de empezar a procesar la entrada. Aquí damos valor inicial cero a una variable `total`. El código del bloque intermedio se ejecuta para cada línea de entrada, y al final se ejecuta una vez el código del bloque `END{...}`. El separador de instrucciones es `;`.

Para ejecutar el programa desde el archivo, utilizamos la opción `-f`:

```
> ls -l | awk -f total.awk

512
512
281765
```

```
3049984
512
2048
512
512
512
1024
2560
512
512
1536
512
512
512
512
512
7168
80
80
2048
Total = 3354949
>
```

De hecho, el bloque `BEGIN` no es realmente necesario, como cada variable en `awk` tiene el valor inicial cero. No es obligatorio incluir los bloques `BEGIN` y `END` y se puede incluir solamente uno de los dos según la necesidad.

Si no queremos llamar explícitamente a `awk`, podemos "esconderlo" en el archivo de instrucciones por incorporar la línea `#!/bin/awk -f` al comienzo del archivo y darle permisiones de ejecutar el archivo por lo menos al usuario mismo:

```
> less total.awk
#!/bin/awk -f

{print $5; total = total + $5}
END {print "Total = " total}
> chmod u+x total.awk
> ls -l | ./total.awk

512
512
281765
3049984
512
2048
512
512
1503
512
1024
2560
512
512
1536
512
512
512
512
512
7168
```

```
76
73
2048
Total = 3356441
>
```

Aquí /bin/awk es la ubicación de awk en el sistema — la ubicación de todo programa se puede averiguar en Unix por ejecutar which con la instrucción de interés. Por ejemplo,

```
> which awk
/bin/awk
> which sh
/bin/sh
> which sed
/bin/sed
> which firefox
/opt/csw/bin/firefox
> which emacs
/usr/local/bin/emacs
```

Nota que which solamente busca por las carpetas definidas en la variable ambiental PATH del usuario,

En el ejemplo siguiente, incorporamos awk con sed para renombrar varios archivos: vamos a renombrar cada archivo con terminación .txt a tener la terminación .text. Primero utilizamos la opción -1 (no es una ele, es el número uno) con ls (nota que es el dígito uno, no la letra ele) para darnos los puros nombres de los archivos, después de que usamos awk para imprimir las instrucciones de renombrar, sed a hacer el cambio y finalmente sh para ejecutarlo todo:

```
> ls *.txt
diapositivas_sergio.txt  gnuplot.txt              pifi_libros1.txt         acceptance.txt
> ls -1 *.txt | awk '{print "mv "$1" "$1}' | sed s/txt/text/2 | sh
> ls *.text
diapositivas_sergio.text  gnuplot.text              pifi_libros1.text         acceptance.text
```

Por último, mostramos un ejemplo con una cláusula condicional if. Queremos obtener un listado de archivos de tamaño mayor o igual a 2048 bytes (o sea, dos megabytes). La primera versión del programa es

```
#!/bin/awk -f

BEGIN {print "Archivos de 2MB y mayores:"}
{if ($5 >= 2048) print $0}
```

donde $0 refiere a la línea entera. Intentamos con este programa:

```
> chmod u+x grandes
> ls -l | ./grandes
Archivos de 2MB y mayores:
-rw-r--r--   1 elisa    faculty   281765 Jan 30 12:48 PID295149.pdf
```

```
-rw-r--r--   1 elisa    faculty  3049984 Jan 24 14:11 SemillaBarros2006.doc
drwx------   4 elisa    faculty     2048 Jun 14 12:00 backup
drwx------   2 elisa    faculty     2560 Jun 15 10:45 mail
drwx------   2 elisa    faculty     7168 Sep  6  2006 todo
drwx------  13 elisa    faculty     2048 Jun 15 12:17 work
```

Está incluyendo también carpetas en la salida. Para eliminar carpetas, tomamos en cuenta que todas esas líneas comienzan con la letra `d`. Utilizamos la operación de comparación con una expresión regular: `$1 /d/` es verdad si la primera columna contiene la letra `d`. Nosotros queremos que *no* la contenga, por lo cual incluimos en el mismo `if` como una segunda condición `$1 ! /d/` y juntamos las dos condiciones a ser cumplidas con `&&` que significa "y":

```
#!/bin/awk -f

BEGIN {print "Archivos de 2MB y mayores:"}
{if (($1 !~ /d/) && ($5 >= 2048)) print $0}
```

Con esta ya obtenemos el resultado deseado:

```
> ls -l | ./grandes
Archivos de 2MB y mayores:
-rw-r--r--   1 elisa    faculty   281765 Jan 30 12:48 PID295149.pdf
-rw-r--r--   1 elisa    faculty  3049984 Jan 24 14:11 SemillaBarros2006.doc
>
```

Para ver la sintaxis completa de `awk`, hay un tutorial muy completo en `http://www.gnu.org/manual/gawk/html_node`.

6.2.2. sh

Los shell script sirven para situaciones donde hay que ejecutar con frecuencia una serie de instrucciones y uno quiere ahorrar el esfuerzo de tener que escribirlos cada vez. También ayuda si hay que correr por ejemplo un experimento que consiste de varias instrucciones pero uno no quiere esperar en persona en el terminal para escribir la instrucción siguiente. En esta sección veremos cómo escribir un script tipo Bourne shell. Se escribe en un archivo de texto, por ejemplo con `emacs`, y la primera línea *siempre* es `#!/bin/sh` (o realmente la ubicación verdadera de `sh`, lo que se ve con la instrucción `which sh`). La última línea del script debería ofrecer un valor de salida para el script: cero si todo estuvo bien (`exit 0`) y otro valor si un error ocurrió.

Para poder ejecutar el script, el archivo tiene que tener permiso de ejecución por lo menos para el usuario mismo, lo que se logra con `chmod a+x` junto con el nombre del archivo donde se guarda el script. En el primer ejemplo escribimos un script pequeño que no hace nada útil mas que simplemente mostrar la sintaxis de los shell script:

```
> emacs -nw primero
File Edit Options Buffers Tools Help
#!/bin/sh
```

```
echo "hola a todos"
echo "la fecha y hora actual:"
date
exit 0
----:---F1  primero           (Fundamental)--L5--All---------------------------
Wrote /u/faculty/elisa/primero
> chmod u+x primero
> ./primero
hola a todos
la fecha y hora actual:
Fri Jun 22 16:31:53 CDT 2007
>
```

Entonces, el script ejecuta cada línea escrita en la línea de instrucciones, una a la vez. Podemos incluir parámetros desde la línea de instrucciones:

```
> emacs -nw primero
File Edit Options Buffers Tools Insert Help
#!/bin/sh
echo "hola, " $1
echo "la fecha y hora actual:"
date
exit 0
----:---F1  primero           (Shell-script[sh])--L2--All----------------------
> ./primero elisa
hola, elisa
la fecha y hora actual:
Fri Jun 22 16:34:14 CDT 2007
>
```

Los parámetros de línea de instrucciones están guardados a variables especiales `$1`, `$2`, etcétera. La variable `$0` tiene en este caso el valor `./primero`. Para no olvidar nosotros mismos qué es lo que hace el script, podemos añadir comentarios, o sea, líneas que contengan el símbolo #:

```
#!/bin/sh
# toma como parametro el nombre del usuario e imprime fecha & hora
echo "hola, " $1
echo "la fecha y hora actual:"
date
exit 0 # salir con estado normal sin errores
```

Podemos declarar variables y ejecutar operaciones aritméticas básicas con enteros — para más operaciones matemáticas contamos con `expr` (ver `man expr` para más información):

```
#!/bin/sh
# toma como parametro el nombre del usuario e imprime fecha & hora
echo "hola", $1
echo "la fecha y hora actual:"
date
sum=0
count=0
for file in ‘ls -1‘
do
  lines=‘wc -l $file | awk ’{print $1}’‘
```

```
  echo "El archivo" $file "tiene" $lines "lineas."
  sum=‘expr $sum + $lines‘
  count=‘expr $count + 1‘
done
echo "En total son" $sum "lineas en los" $count "archivos."
prom=‘expr $sum / $count‘
echo "Promedio redondeado:" $prom
exit 0
```

Copiando del PDF, hay que cuidar mucho poner los símbolos al comienzo y final de las expresiones correctamente. Cuando lo corremos, obtenemos lo siguiente:

```
> ./primero elisa
hola, elisa
la fecha y hora actual:
Fri Jun 22 17:55:14 CDT 2007
El archivo countsize tiene 23 lineas.
El archivo datos.txt tiene 25 lineas.
El archivo primero tiene 19 lineas.
El archivo primero~ tiene 19 lineas.
El archivo teclado.txt tiene 8 lineas.
El archivo total.awk tiene 4 lineas.
En total son 98 lineas en los 6 archivos.
Promedio redondeado: 16
>
```

Hay que tomar en cuenta que esto no va a funcionar si los nombres de los archivos o las carpetas contienen espacio blanco. Por lo general no es una buena idea utilizar blancos en los nombres por razones de compatibilidad. También se confunde con carpetas y enlaces — como ejercicio, puedes modificar la línea que elije el nombre del archivo en una variable así que no considere carpetas o enlaces.

Para mostrar cómo implementar *condiciones y bucles* en shell scripts, el siguiente ejemplo intenta verificar con un algoritmo muy simple si o no un entero es primo:

```
#!/bin/sh
# toma como parametro un entero mayor a uno y determina si es primo
echo "Determinando si" $1 "es primo:"
cont=si
n=$1
i=2

# mientras i es menor que n
while [ $i -lt $n ]
do

# dividimos por i
div=‘expr $n / $i‘

# calculamos el producto del resultado con i
prod=‘expr $div ’*’ $i‘
# tomamos la diferencia
mod=‘expr $n - $prod‘

# si el diferencia es cero
if [ $mod -eq 0 ]
```

```
 then
    # tenemos un divisor
    echo $n "no es primo, tiene divisores" $i "y" $div"."
    exit 0
 fi
i=‘expr $i + 1‘
done
echo $n "es primo."
exit 0
```

El algoritmo es muy genuino — sería mucho mejor no dividir con otros números pares después de dos. Intenta como ejercicio modificar el algoritmo para no hacerlo. Además, bastaría probar divisores hasta la raíz cuadrada del número de entrada. Algunos ejemplos de la función del programa simple, guardado en el archivo `primo` con permisiones de ejecución, son los siguientes:

```
> chmod u+x primo
> ./primo 2947
Determinando si 2947 es primo:
2947 no es primo, tiene divisores 7 y 421.
> ./primo 3969
Determinando si 3969 es primo:
3969 no es primo, tiene divisores 3 y 1323.
> ./primo 7
Determinando si 7 es primo:
7 es primo.
```

Nota que el algoritmo ejecuta por mucho tiempo con primos y muy poco con números que tienen divisores pequeños. Se puede mejorar por manejar los pares aparte y por cambiar la cota superior del `while` a uno que solamente avanza hasta la raíz cuadrada de la entrada. Estas modificaciones hacen que el algoritmo corre mucho más rápidamente, lo que se puede probar por ejemplo con la entrada 3457, que sí es primo.

Vale la pena revisar a `bash` que es muy parecida a `sh` como herramienta pero un poco más amigable.

6.2.3. Perl

Perl es un lenguaje script, esto es el código se compila antes de ejecutarlo, a diferencia de otros lenguajes como C, en donde el código se compila y traduce a lenguaje maquina.

Script básico

El siguiente script ilustra el uso del lenguaje Perl. La primera línea del script debe contener los caracteres de #! seguidos de la ruta en donde se encuentra el ejecutable de perl, con esto le decimos a la maquina que el script lo interpretará con perl.

```
#!/usr/bin/perl
```

```
print "Hola mundo";
```

6.2.4. Sintaxis

Comentarios

Para comentar en perl es necesario colocar el carácter #, todo el texto que se encuentre después de este carácter será tomado como comentario.

6.2.5. Variables Escalares

Las variables escalares pueden contener enteros, reales, cadenas o referencias. El nombre de la variable va precedido por el signo de pesos ($). En Perl las variables se declaran con la palabra clave my, sin embargo no necesitan ser declaradas. Por ejemplo,

```
#!/usr/bin/perl

#Declaro una variable
my $hola;

#Asigno valores
$hola="Hola Mundo\n";
$adios="adios\n";

#Escribimos un poco en pantalla
print $hola;
$hola=23;
print "Mira un número: $hola \n";
print "Y $adios";
```

Operadores de variables escalares

El operador de asignación es el símbolo =, como se puede observar en el primer ejemplo de la página. También se pueden mezclar operadores aritméticos junto con la asignación. Por ejemplo,

```
#!/usr/bin/perl

#Asigno valores a variables
my $uno=123.67;
$dos=123123.2334;
$suma=$uno;
$suma+=$dos;

#Escribimos resultados en pantalla
print "La suma de $uno y $dos es = $suma \n";
```

Además se tienen los operadores autoincremento (++), autodecremento (–), potencia (**). Por ejemplo,

```
#!/usr/bin/perl

#Asigno valores a variables
my $contador=0;

#Escribimos en pantalla
print "Se contar:", $contador++ , ", ", $contador ++,"...\n";

print "2 elevado a $contador es:", (2**$contador), "\n";
```

Para concatenar cadenas se utiliza el operador punto.

```
#!/usr/bin/perl

#Asigno valores a variables
my $cad1="Hola";
my $cad2="Mundo";
my $cad3=$cad1." ".$cad2;

#Escribimos en pantalla
print $cad3;
```

6.2.6. Arreglos

Los arreglos van precedidos por el símbolo arroba (@). Cuando se acceda al elemento de un arreglo se utiliza el nombre del arreglo precedido por el carácter $, ya que es un escalar.

Los variables pueden contener diferente tipo de datos y pueden ser redimensionados solo indicando la posición del nuevo elemento. Por ejemplo,

```
#!/usr/bin/perl

#Declaramos la variable primer_array como un array
my @primer_array;

#asignamos unos cuatro valores al array
@primer_array=(1,"dos",3,"cuatro");

#Añadimos un quinto de forma individual
$primer_array[4]=5.5;

#Mostramos el tercer elemento del array
print "El tercero es= ".$primer_array[3]." \n";
```

Para sacar/insertar elementos se pueden usar las funciones pop y push. Que sacan o insertan, respectivamente, un elemento al final del arreglo, es decir, tratan el array como una pila. También podemos utilizar shift y unshift para sacar o insertar, respectivamente, un elemento del principio del array.

Para ver el tamaño (número de elementos) de un array se utiliza el símbolo de sostenido entre el símbolo y el nombre del array, es decir, con $#array. Este tamaño nos lo da contando desde 0, realmente nos da el último índice que existe en el array. Si el array no tuviese ningún elemento, su tamaño sería -1.

```
#!/usr/bin/perl

#asignamos unos cuatro valores al array
@matriz=(1,"dos",3,"cuatro");

#Añadimos con Push
push(@matriz, 5, 6, "siete");

#Mostramos el último elemento
print "El último es ". $matriz[$#matriz]."\n";

#sacamos con Pop
$uno=pop(@matriz);

print "He sacado $uno\n";

#Añadimos con unshift
unshift(@matriz, "cero", -1 );

#Mostramos el primer elemento
print "El primero es ". $matriz[0]."\n";

#sacamos con shift
$uno=shift(@matriz);
s
print "He sacado $uno\n";

print "La matriz tiene ".$#matriz." elementos\n";
```

En todo script de Perl existe el array ARGV que contiene los parámetros de entrada.

Para añadir dimensiones a los arrays, simplemente se añaden corchetes. Como en el siguiente ejemplo de arrays con más de una dimensión:

```
#!/usr/bin/perl

my @array3D=([],[],[]);

$array2D[0][0]=0;
$array2D[0][1]=1;
$array2D[1][0]=2;
$array2D[1][1]=3;

$array3D[0][0][0]=20;

print $array2D[1][1]." ".$array3D[0][0][0]."\n";
```

Otro tipo de arreglo son los asociativos en los cuales los elementos son referenciados mediante claves en vez de una posición. Para los array asociativos se utiliza el símbolo por ciento. Al igual que sucede con los arrays normales, cuando se accede a un elemento de un array asociativo se debe hacer referencia como un escalar con el símbolo pesos. Además la indexación por clave no se hace utilizando los corchetes, sino que se utilizan las llaves. Por ejemplo,

```
#!/usr/bin/perl

#asignamos valores a una tabla hash
my %colorfruta;

$colorfruta{"verde"}="kiwi";
$colorfruta{"amarillo"}="platano";
$colorfruta{"rojo"}="sandía";
$colorfruta{"naranja"}="naranja";

print "Una fruta verde es: ".$colorfruta{"verde"}."\n";

%dias=("lunes",L,"martes",M,"miercoles",X,"jueves",J,
  "viernes",V,"sabado",S,"domingo",D);
print "La representación del Martes es :".
  $dias{"martes"}." \n";
```

6.2.7. Manejo de archivos

La función open(manejador de archivo, archivo) abre un archivo.

El parámetro manejador de archivo es la variable con la que se hará referencia al archivo en el código. El parámetro archivo es el nombre del archivo que se abrirá. Además del parámetro se especifica el modo de apertura del archivo, es decir, lectura, escritura, concatenación.

```
open(FILE, "$file"); #Abre el archivo para lectura
open(FILE, "$>file"); #Abre el archivo para escritura
open(FILE, "$>>file"); #Abre el archivo para concatenar,
# es decir escribir al final del archivo.
```

Para escribir en un archivo se utiliza la función print FILE “texto”, en un archivo que haya sido abierto para escritura. Además para abrir la entrada estándar (teclado) y salida estándar (pantalla) se utilizan:

```
open(FILE, '-'); #Abre la entrada estandar
open(FILE, '>-'); #Abre la salida estandar
```

Para leer el archivo completo se puede utilizar la sentencia @lines = <FILE>; que lee todas las líneas que se encuentran en el archivo FILE, para leer linea por linea se utiliza una variable escalar en vez de un arreglo, esto sería $line = <FILE>; que lee la siguiente línea en el archivo. Un ejemplo:

```
#!/usr/bin/perl

#Ejemplo de lectura y escritura de archivos
open(KEYBOARD, '-');
print "Nombre del archivo a leer: ";
$fileNameInput = <KEYBOARD>;

open(FILEINPUT, "$fileNameInput");
#Abrir el archivo para lectura
```

```
@lines = <FILEINPUT>;

print "\nNombre del archivo donde se guardaran los datos:";
$fileNameOutput = <KEYBOARD>;


open(FILEOUTPUT, ">$fileNameOutput");
#Abrir el archivo para escritura

foreach  $line (@lines)
#Guardar el contenido del archivo de entrada
#en el archivo de salida
{
print FILEOUTPUT $line;
}

#Agregar este texto en el archivo de salida
print FILEOUTPUT "Este archivo fue creado por perl\n";
```

Estructuras de control

foreach Se utiliza para moverse a través de cada uno de los elementos en un arreglo

```
foreach $linea (@arreglo)
{
print $linea;
}
```

Comparaciones Los operadores de comparación son:

- == Igualdad numérica.
- != Operador diferente de, funciona numéricamente.
- eq Igualdad para cadenas de texto.
- ne Operador diferente de, funciona para cadenas de texto

Otra estructura de control es el for cuya sintaxis es:

```
for(inicializa; compara; incrementa)
{
sentencias;
}
```

Además de los anteriores también se tiene el ciclo while

```
while(condicion)
{
```

```
sentencias;
}

do
{
sentencias;
}
while(condicion);
```

Para arreglos asociativos se puede utilizar el ciclo while (my ($key, $value) = each %arreglo), para acceder al par llave, valor de un arreglo.

Condicionales Perl también admite la sentencia if, else;

```
if(condicion)
{
sentencias;
}
elsif(condicion)
{
sentencias;
}
else
{
sentencias;
}
```

6.2.8. Expresiones regulares

Uno de los aspectos mas poderosos de Perl es la manipulación de cadenas de texto. Para esto utiliza expresiones regulares.

Las expresiones regulares se expresan entre slashes y el mapeo se realiza con el operador = . Por ejemplo la siguiente expresión es verdadera si la palabra “algo” existe en la cadena de texto. $sentencia = /algo/.

Otra manera de aplicar una RE es con la variable $_, la cual permite hacer lo siguiente:

```
$_ = "Yo vivo en mi casa ";

if(s/casa/Casa/)
{
print "La cadena es " . $_ . "\n";
}
```

El operador ! se utiliza para obtener las cadenas que no mapean con la expresión utilizada.

Además de buscar cadenas de texto Perl también puede reemplazar estas cadenas para esto se utiliza la función s.

```
$sentencia =~ s/casa/Casa/
#Substituye la palabra casa por Casa.
```

El ejemplo anterior solamente reemplaza la primera ocurrencia en la cadena de texto, para reemplazar todas las ocurrencias se utiliza la función g.

```
$sentencia =~ s/casa/Casa/g
#Substituye la palabra casa por Casa.
```

Las expresiones regulares por omisión son sensibles a mayúsculas y minúsculas, para indicar que no sea especifica se utiliza la función i

```
 $sentencia =~ s/casa/Casa/gi
#Substituye la palabra casa por Casa, cAsa por Casa, etc.
```

Perl tiene la capacidad de recordar las expresiones regularas utilizadas recientemente, para esto utiliza las variables $1, ... $9.

```
if(s/casa/Casa) { print "ocurrencia $1"; }
```

Perl permite reemplazar carácter por carácter en una RE.

```
$sentencia =~ tr/abc/efg
#Reemplaza cada a por e, cada b por f, cada c por g.
```

Otra función muy útil en Perl es split, la cual divide una cadena de texto y regresa un arreglo. La función es de la forma split(RE, texto)

```
 $strColores = "blanco,negro,azul";
@colores = split(/,/, $strColores);
```

6.2.9. Subrutinas

Como todo buen lenguaje de programación Perl puede manejar subrutinas de código

```
sub rutina
{
print "Hacer algo";
print "El primer parametro vale @_[0]"
}

#Se llama a la rutina
&rutina;

#Se llama a la rutina y se le pasa un parametro
&rutina("valor1");
```

Para leer los parámetros pasados a la rutina se utiliza la variable _, la cual es un arreglo de los parámetros pasados a la rutina.

El resultado de una rutina es siempre la ultima operación evaluada.

Para declarar una variable que sea local en la subrutina se utiliza la función local($var1, $var2), donde $var1 y $var2 son las variables locales.

6.2.10. Ejemplo de uso de perl

El siguiente código reemplaza todos los acentos en un archivo HTML por su código para ser visualizado correctamente.

```
#!/usr/bin/perl

#Ejemplo que reemplaza los acentos en un
# archivo HTML por su codigo válido.

#Declarar los caracteres que se reemplazarán
my %specialChars = ("á" => "&aacute;", "é" => "&eacute;",
"í" => "&iacute;", "ó" => "&oacute;", "ú" => "&uacute;");

open(KEYBOARD, '-');
print "Nombre del archivo HTML a procesar: ";
$fileNameInput = <KEYBOARD>;

open(FILEINPUT, "$fileNameInput");
#Abrir el archivo para lectura

@lines = <FILEINPUT>;

print "\nNombre del archivo HTML procesado:";
$fileNameOutput = <KEYBOARD>;


open(FILEOUTPUT, ">$fileNameOutput");
#Abrir el archivo para escritura
foreach  $line (@lines)

#Guardar el contenido del archivo de entrada
#en el archivo de salida
{
$newString = $line;

while ( my ($key, $value) = each %specialChars )
{
$newString =~ s/$key/$value/g;
}

print FILEOUTPUT $newString;
}
```

6.3. Programación estadística, cálculo científico y simulación

Las herramientas gratuitas para programación estadística y cálculo científico son numerosas. Por ejemplo *scilab* (http://www.scilab.org/) es muy parecido a *Octave* (de Sección 8.1) y *scicos* es una caja de herramientas (inglés: toolbox) de scilab.

Para diagramas y análisis de datos, una opción más es *Grace* (http://plasma-gate.weizmann.ac.il/Grace/). *COIN* (COmputational INfrastructure for Operations Research, http://www.coin-or.org/index.html) es un conjunto de librerías y herramientas especiales para la investigación de operaciones que cuenta por ejemplo con solvers de código abierto para poder modificarlos libremente e integrarlos como componentes en software propio. Para programación en C o C++, también sirven la librerías de *GNU Scientific Library* (http://www.gnu.org/software/gsl/). Vale la pena para estudiantes y investigadores conocer a través de sus páginas web las posibilidades de estas herramientas gratuitas.

6.3.1. R for Statistical Computing

R es un software libre para cómputo estadístico y de gráficos. Se compila y ejecuta sobre una amplia variedad de plataformas Unix, Windows y MacOS. R se puede descargar desde el sitio oficial del proyecto en http://www.r-project.org/, ahí también encontrarán documentación referente a la instalación en las distintas plataformas en las que se ejecuta. La versión actual de R es la 2.5.1 liberada el 28 de junio de 2007.

Introducción a R

R es similar al sistema S que fue desarrollado por John Chambers de los laboratorios Bell. Provee una amplia variedad de técnicas estadísticas y gráficas (modelado lineal y no lineal, pruebas estadísticas, análisis de series de tiempo, clasificación y clustering entre otras). R es una suite integrada de componentes de software para manipulación de datos, cálculo y despliegue de gráficos, algunas de las características que posee son:

- Un componente para manipulación y almacenamiento de datos.
- Una suite de operadores para cálculos sobre arreglos, en particular matrices.
- Una amplia, integrada y coherente colección de herramientas intermedias para análisis de datos.
- Componentes para análisis y despliegue de gráficos.
- Un simple y efectivo lenguaje de programación Llamado S que incluye condicionales, ciclos, funciones recursivas definidas por el usuario y componentes de entrada y salida. (Por esto muchas de las funciones incluidas en el sistema están escritas en S.)

Usando R interactivamente

R al igual que muchos de los paquetes de Unix es case sensitive por lo que se debe considerar esto al definir variables y al utilizar instrucciones. Al usar R se nos presentará un prompt que espera instrucciones de entrada. El prompt por omisión es > que en Linux puede ser el mismo prompt del shell por lo que pudiera parecer que nada ha pasado.

En el siguiente ejemplo asumiremos que el prompt de Unix (o Linux) es `$` y el de R es >. Para iniciar el programa R usaremos `$ R` y para salir de R utilizaremos `>q()`.

La ayuda de R R tiene un modulo de ayuda similar a `man` en Unix. Para obtener información sobre alguna función en particular, por ejemplo para consultar sobre `solve` se escribe `>help(solve)`. Otra alternativa seria utilizar: `>?solve`.

Para características especificadas por caracteres especiales, el argumento debe ser encerrado en comillas dobles o simples, por convención se utilizan las dobles comillas. Un ejemplo es `>help( [[ )`.

Para ver ejemplos sobre algún tópico podemos usar `>example(topic)`. En la figura 6.1 se puede ver un ejemplo de éste ejecutado sobre una versión de R en Windows.

Ejecución desde un archivo o envío de salida a un archivo externo

Si se almacenan instrucciones en un archivo externo (digamos `command.R`), estos pueden ser ejecutados durante una sesión de R con `source("commands.R")`. La función `sink()` dirigirá toda la salida subsiguiente de la consola de un archivo externo: `>sink( record.lis )`. Ejecutar `>sink()` lo restablece de nuevo a la consola.

Permanencia de datos y eliminación de objetos Las entidades que R crea son llamadas objetos, pueden ser variables, arreglos de números, cadenas de caracteres, funciones o estructuras más generales construidas a partir de tales componentes.

Durante una sesión de R los objetos son almacenados por nombre. La colección de objetos almacenada actualmente es llama workspace.

`>objects()` se puede utilizar para desplegar los nombres de los objetos almacenados actualmente en R. Para eliminar objetos almacenados se cuenta con `rm`. Por ejemplo, `>rm(x, y, z, junk, temp)`.

Todos los objetos creados durante una sesión de R son permanentemente almacenados en un archivo para su uso futuro en otras sesiones. Al terminar una sesión se te da la oportunidad de guardar todos los objetos disponibles.

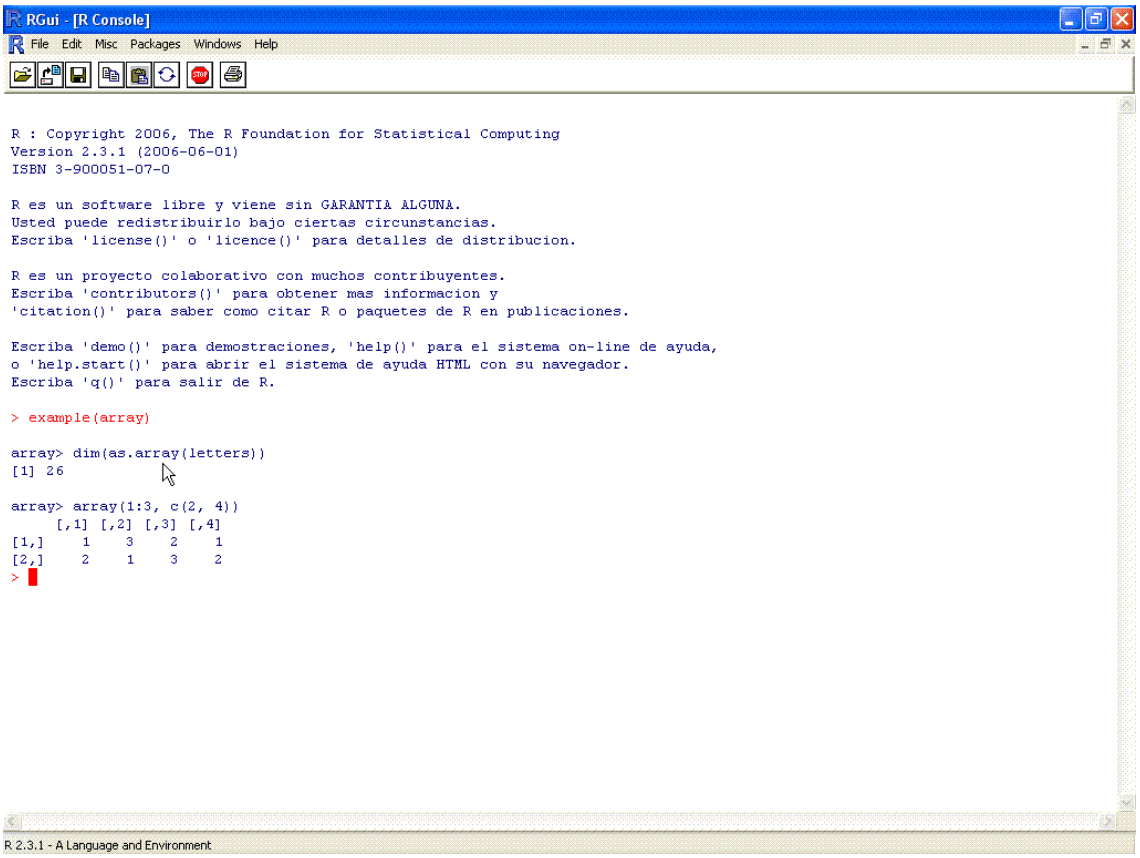


Figura 6.1: R en Microsoft Windows.

Manipulaciones simples; números y vectores

R opera sobre estructuras de datos con nombres. La estructura más simple es el vector numérico, el cual es una entidad que consiste de una colección ordenada de números. Para definir un vector llamado x que consista de cinco números, digamos 10.4, 5.6, 3.1, 6.4 y 21.7 usaremos:

```
>x <- c(10.4, 4.6, 3.1, 6.4, 21.7)
```

Esta es una operación de asignación usando la función `c()` que en este contexto toma un número arbitrario de argumentos vector y regresa un vector cuyo valor se obtiene al concatenar estos argumentos.

Nótese que el operador <- consiste de los caracteres "menor que" y "menos" apunta en dirección del objeto que recibe el valor de la expresión, debido a esto podemos hacer la asignación también en la otra dirección:

```
>c(10.4, 4.6, 3.1, 6.4, 21.7) ->x
```

También podemos realizar asignaciones usando la función assign(), una forma equivalente de hacer lo que se ha hecho arriba usando esta función seria:

```
>assign( x , c(10.4, 4.6, 3.1, 6.4, 21.7))
```

Veamos como construir un vector a partir de otros vectores, la expresión `>y <- c(x, 0, x)` creará un vector con 11 entradas consistiendo de 2 copias de x con un cero en la posición central.

Los vectores se pueden utilizar en expresiones aritméticas en cuyo caso las operaciones se ejecutan elemento por elemento. Los vectores que aparecen en la misma expresión no necesitan ser de la misma longitud, si no lo son, el valor de la expresión es un vector con la longitud del más grande de los vectores que aparecen en la expresión. El vector más pequeño se recicla hasta obtener la longitud del vector mas largo. En particular una constante simplemente se repite. Entonces la asignación `>v <- 2*x + y + 1` genera un vector v de longitud 11 (la de y) construido sumando elemento por elemento, `2*x` repetido 2,2 veces (para obtener la longitud de y), y repetido solo una vez y 1 repetido 11 veces.

Todos los operadores aritméticos elementales son los comunes `+`, `-`, `*`, `/` y `^` para elevar a potencia. Además están disponibles todas las funciones aritméticas comunes `log`, `exp`, `sin`, `cos`, `tan`, `sqrt`, etcétera. Dos de las funciones estadísticas disponibles son `mean(x)` la cual calcula la *media de la muestra* y `var(x)` que calcula la *varianza de la muestra*. El valor de la *media* lo podemos calcular como `>sum(x)/length(x)` y la *varianza* como

```
>sum((x-mean(x))^2) / (length(x) - 1)
```

R tiene un buen número de posibilidades para generar secuencias de números utilizadas comúnmente, una forma de generar una secuencia creciente seria `1:30` que generaría el vector `c(1, 2, 3, .., 29, 30)`, es importante considerar que el operador `:` tiene alta prioridad por lo que la expresión `2*1:15` generaría la secuencia `c(2, 4, ..., 28, 30)`.

Usando la función `seq()` podemos también generar secuencias numéricas, un ejemplo en el que indicamos la diferencia entre cada elemento de la secuencia seria

```
>seq(-5, 5, by=0.2) ->s3
```

que genera en `s3` el vector `c(-5.0, -4.8,  , 4.8, 5.0)`.

Vectores de caracteres Es frecuente utilizar vectores de caracteres en R para cosas como etiquetas de gráficos y algunas otras. En R las cadenas de caracteres se delimitan por comillas dobles o comillas simples, por ejemplo `"x-values"` o `"New iteration results"`. También es importante considerar que existen caracteres que no se pueden utilizar directamente en las cadenas de caracteres por lo que se hace uso de caracteres de escape al estilo de C. Para generar vectores de cadenas se pueden utilizar las mismas técnicas que se usan con números.

Otros tipos de objetos Los vectores son el tipo de objeto más importante en R, pero existen otros de los cuales podemos sacar provecho:

- *matrices* o más genéricamente *arreglos* que son vectores multidimensionales, y
- *funciones* que son por si mismas objetos en R que se pueden almacenar en el workspace del proyecto.

Básicos de probabilidad

R nos da la posibilidad de trabajar muchas de las operaciones relacionadas con distribuciones de probabilidad. Existen una gran cantidad de distribuciones de probabilidad disponibles en R, entre ellas la *distribución normal*, la *distribución t*, la distribución *binomial* y la *chi-cuadrada*.

Veamos unos ejemplos utilizando la distribución normal. Para obtener los diversos valores relacionados con una distribución normal contamos con las funciones dnorm, pnorm y qnorm entre otras. La función dnorm devuelve la altura de la distribución de probabilidad en cada punto. Dado un número x, la función `pnorm` calcula la probabilidad de que un número aleatorio normalmente distribuido sea menor o igual que ese número x. Esta función recibe el nombre de función de distribución acumulada.

Un ejemplo de `pnorm` para una distribución con media cero y varianza uno, tendríamos `>pnorm(0)`. Como la distribución tiene media cero, obtendríamos como resultado de `pnorm` el valor de 0.5.

La función `qnorm` funciona como la inversa de `pnorm`. La idea detrás de `qnorm` es que dada una probabilidad, `qnorm` regresa el valor para el cual la distribución *acumulada* genera esa probabilidad. Por ejemplo, para una distribución normal con media cero y varianza uno, la función `qnorm` regresaría el valor del Z-Score que encontramos comúnmente en libros de estadística. Por ejemplo,

- `>qnorm(0.5)` nos daría el valor de cero, mientras
- `>qnorm(0.5, mean = 1)` generaría el valor de 1.

En R encontramos muchas otras funciones relacionadas con distribuciones de probabilidad y variables aleatorias que no están dentro del alcance de este documento introductorio.

Funciones escritas por el usuario

Gráficos R nos da la posibilidad de mostrar datos de forma visual utilizando distintos tipos de gráficos; por ejemplo Strip Charts, Histograms, Boxplots, Scatter Plots y Normal QQ Plots. Aquí haremos uso de los Scatter Plots para mostrar algo de lo que se puede hacer con las herramientas que R nos da.

Un Scatter Plot provee una vista gráfica de la relación entre dos conjuntos de números, por ejemplo podríamos hacer uso de un Scatter Plot para obtener la gráfica de una distribución normal, veamos como podemos crear este gráfico.

Primero generaremos una secuencia de puntos sobre el eje x:

```
>x <- seq(-10, 10, by=0.1)
```

Después generaremos los valores correspondientes a las alturas en la función de probabilidad de la distribución normal, por lo que usaremos la función `dnorm` ya estudiada previamente: `>y <- dnorm(x)`.

Finalmente haciendo uso de la función `plot()` generaremos el gráfico deseado: `>plot(x,y)`. En la figura 6.2 ilustramos el proceso y el resultado que se obtiene.

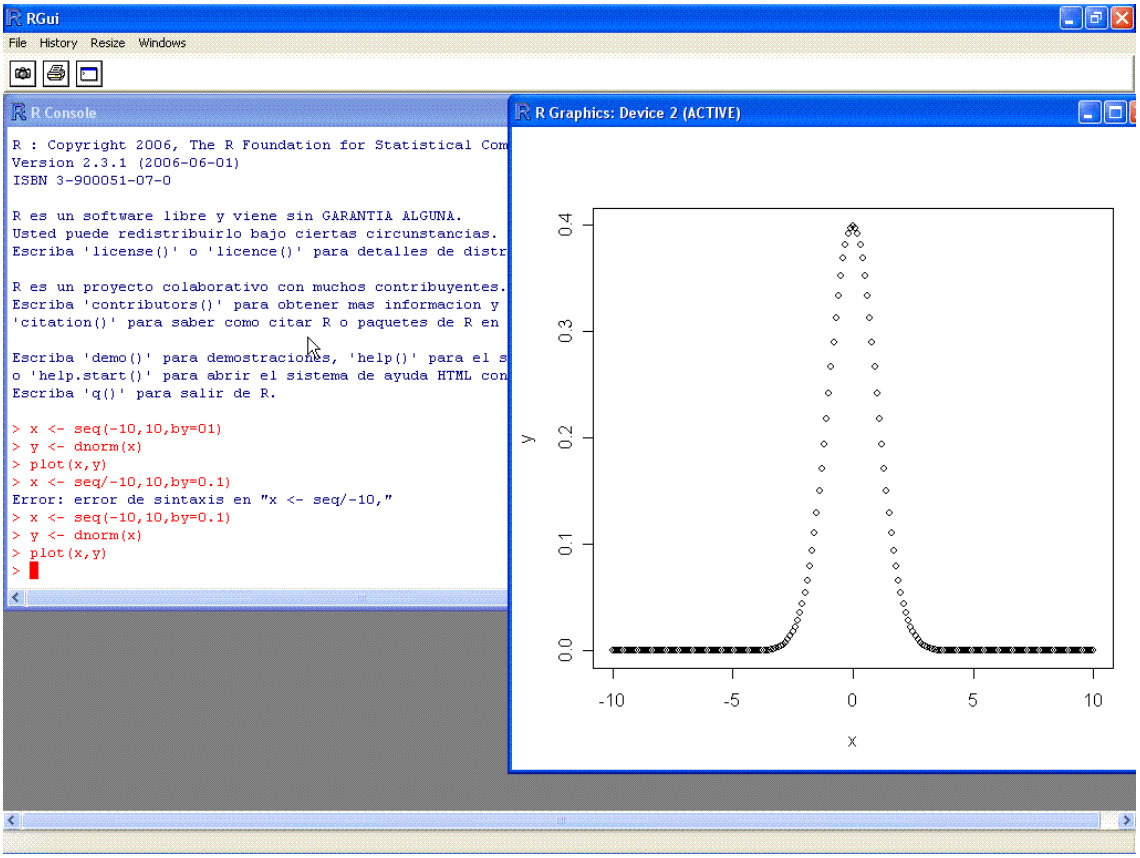


Figura 6.2: Una gráfica generada con R.

Existen muchas otras herramientas que podemos encontrar en R para gráficos más complejos, pruebas estadísticas, análisis numérico, regresiones, podemos crear nuestras propias funciones gracias al lenguaje S. Concluimos que R es una herramienta muy poderosa y versátil que solo está limitada por nosotros mismos.

6.3.2. RePast

RePast [3, 24] (Recursive Porus Agent Simulation Toolkit) es una plataforma para el modelado de sistemas de agentes en dos dimensiones. Es gratuito y funciona en varios sistemas operativos. Está disponible en

`http://repast.sourceforge.net/`

RePast está basado en *programación orientada a objetos* y está disponible para múltiples lenguajes de programación. El paquete ya incluye, por ejemplo, algoritmos genéticos, el método MCMC (Markov Chain Monte Carlo), redes neuronales y regresión. Cuenta con varios ejemplos para modificar, lo que permite desarrollo rápido. Suporta eventos discretos paralelos o secuenciales. RePast incorpora herramientas para registrar eventos y preparar diagramas dinámicas.

La manera más rápida de desarrollar una simulación con RePast es modificar un ejemplo existente. Al ejecutar la versión en Java de RePast desde `repast.jar`, se utiliza la instrucción `java -jar repast.jar`. Lo que aparece es la barra de control de RePast, mostrada en la figura 6.3.

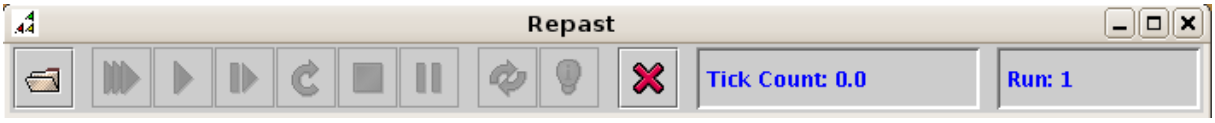


Figura 6.3: La barra de controles de RePast.

Con el botón de la carpeta a la izquierda, uno puede examinar los modelos de ejemplo incorporados en RePast. Son varios y muestran diferente funcionalidad de la herramienta; la figura 6.4 muestra la ventana de los modelos de ejemplo.

Al abrir un modelo, RePast primero abra una ventana de ajuste de parámetros y después al hacer click en "tocar" (el triángulo de la barra), abra la simulación y posiblemente una o más gráficas dinámicamente actualizadas. Un ejemplo de una simulación documentada en [29] está en la figura 6.5.

RePast dirige `System.out` de Java a una ventana *RePast Output* (ver figura 6.6, mientras `System.err` continua siendo el terminal o consola desde el cual se inició la ejecución.

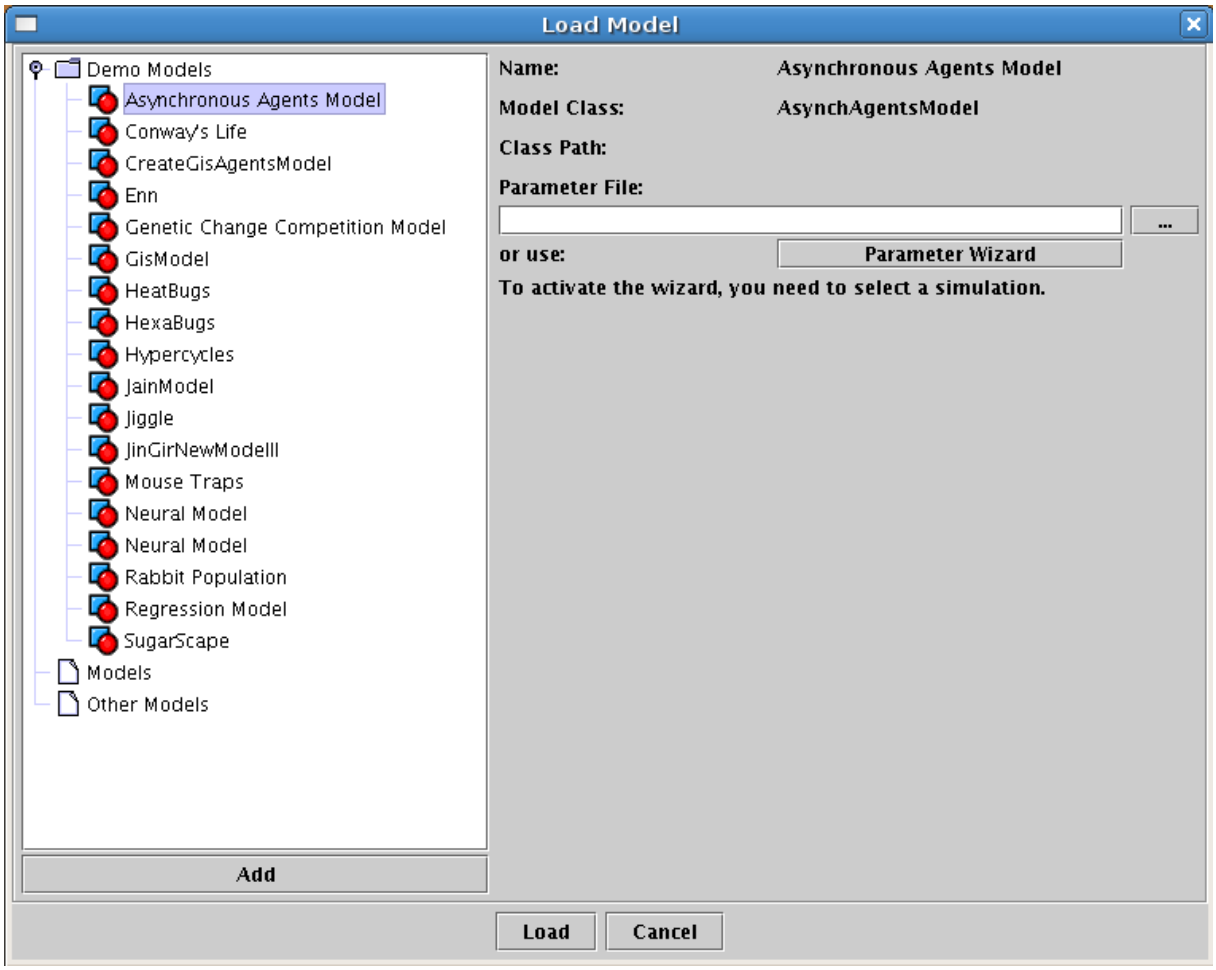


Figura 6.4: El dialogo de ejemplos de RePast para ejecución.

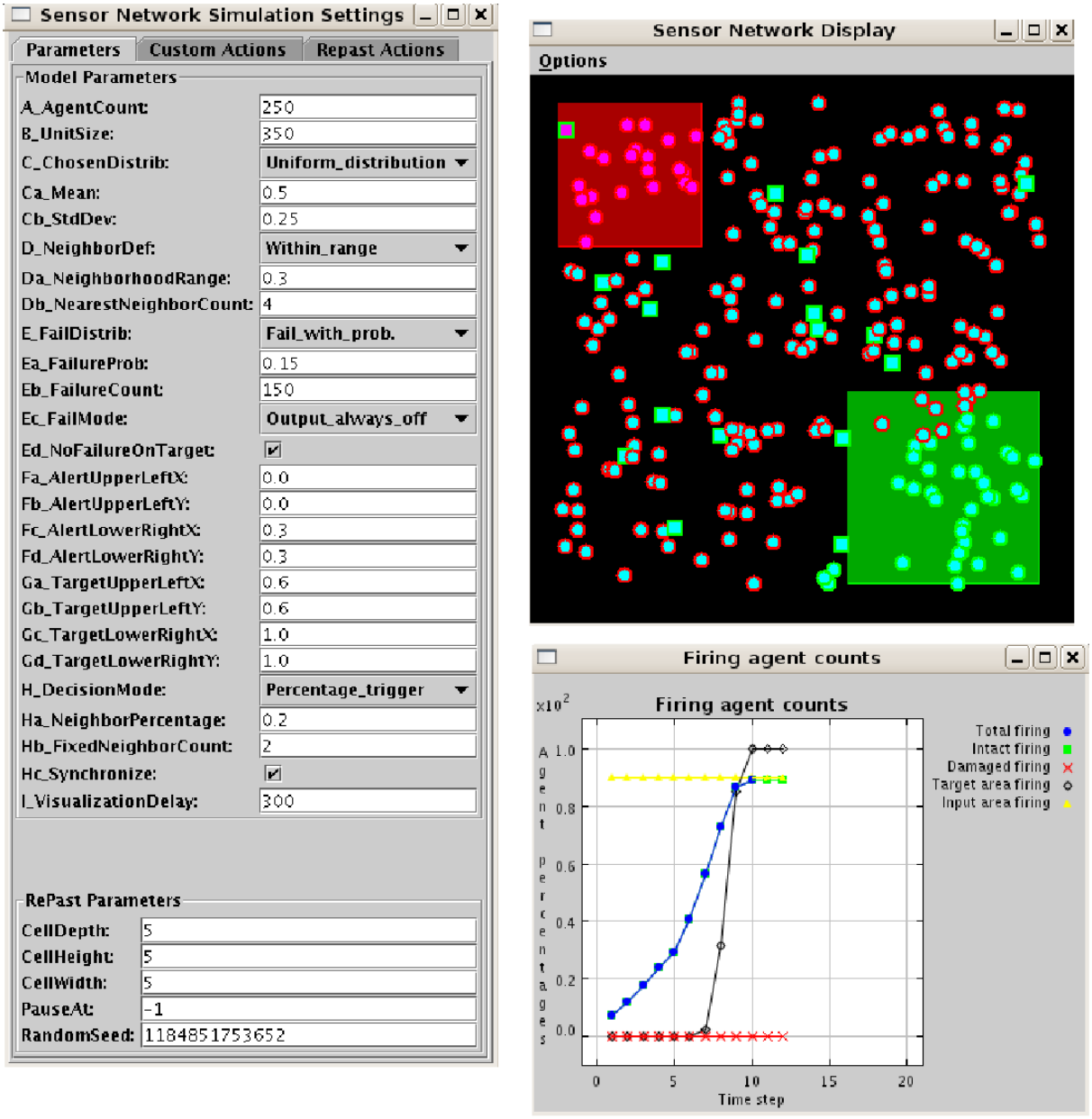


Figura 6.5: Un modelo de ejemplo hecho con RePast donde la alarma viaja a través de una red sensora desde un área de incendio a un área de vigilancia. A la izquierda está la ventana de parámetros, a la derecha arriba la simulación misma y abajo una gráfica que actualiza según los eventos de la simulación dinámicamente durante su ejecución.

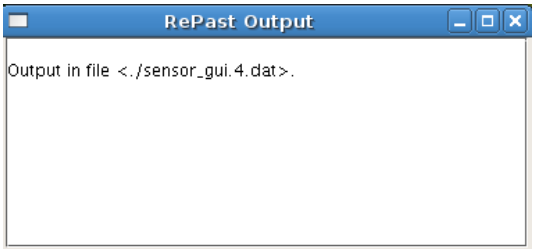


Figura 6.6: La ventanilla `System.out` de RePast.

Capítulo 7

Ejecución automatizada

7.1. Makefile

`make` y `gmake` son programas que ejecutan instrucciones según un *árbol de dependencia*. Las instrucciones se guardan en un archivo de texto que se nombra `Makefile`, en cual caso la ejecución se inicia con solamente la instrucción `make`. Si el nombre del archivo es otro, como por ejemplo `proyecto.make`, hay que utilizar la opción `-f`: `make -k proyecto.make`.

También se puede guardar información de inicialización en otro archivo `make.ini` que se lee primero antes de ejecutar las instrucciones de `Makefile`. Las operaciones de `make.ini` deberían ser suficientemente generales para aplicar a *cada* uso qe el usuario tiene para `make`. Los ambos archivos `Makefile` y `make.ini` tienen la misma estructura: pueden contener (entre cosas más avanzadas) los siguientes elementos:

1. **comentarios**: cada línea que comienza por el símbolo # es un comentario, igual como el resto de una línea que contiene el símbolo # desde su ocurrencia hasta el final de la línea,
2. **reglas**: las reglas con líneas que indican a `make` en qué momento y cómo generar unos ciertos archivos que aquí se llaman los *blancos* y están compuestas por *dependencias* y posiblemente una o más líneas de shell; una regla puede ser *explicita* (directamente declarada en el archivo `Makefile`) o de *inferencia* (reglas más generales),
3. **dependencias**: una línea de dependencia `X Y Z: A B C` indica que para preparar los blancos `X`, `Y` y `Z`, primero hay que tener listos los archivos `A`, `B` y `C` — no hay ningún límite para cuantos archivos se puede exigir para la preparación de los archivos de meta, y tampoco hay límite para el número de blancos,
4. **líneas de shell**: las líneas que comienzan con un **tabulador** (*nunca ningún otro tipo de espacio blanco, no funcionará*) y especifican cuáles instrucciones habrá que ejecutar en el shell para crear/actualizar/modificar los blancos; líneas consecutivas que comienzan con un tabulador son considerados líneas de shell de la misma regla, y

5. **macros**: con un macro, se puede reemplazar una secuencia que repite, una opción que es especialmente útil si los nombres de archivos de cuales depende mucho pueden cambiar; para definir un macro a la secuencia `a b c`, se añade `SEC = a b c` en el archivo antes de las reglas, y en cada regla donde se necesita — no importa en que parte — la secuencia `a b c`, se pone solamente `$(SEC)` o `${SEC}`.

Siempre hay que definir por lo menos una dependencia: `all`, así que en la línea de dependencia `all: X Y Z` se da la lista del "producto final" del proceso. Por ejemplo, para compilar un documento LaTeX con make a un documento tipo Pdf, la dependencia podría ser como `all: doc.pdf`, y para crear un ejecutable del nombre `a.out` de un programa escrito en el lenguaje C, sería `all: a.out`. Si el objetivo es producir varios archivos como el producto final sin que haya dependencia entre ellos, hay que definir varios archivos en la dependencia `all`.

Típicamente se incorpora también *objetivos alternativos* como por ejemplo `limpia:` con el propósito de remover archivos auxiliares que no se necesita. Por ejemplo, al general un documento LaTeX cualquiera del archivo `documento.tex`, siempre se generan automáticamente los archivos `documento.aux` y `documento.log`, y si se usa BibTex, también `documento.bbl` y `documento.blg`. Para remover estos, definir

```
# makefile general de documentos de LaTeX
DOC = documento

all: $(DOC).pdf

# generar el DVI
$(DOC).dvi: $(DOC).tex $(DOC).bib
        latex $(DOC)
        bibtex $(DOC)
        latex $(DOC)
        latex $(DOC)

$(DOC).ps: $(DOC).dvi
        dvips -o $(DOC).ps $(DOC).dvi

$(DOC).pdf: $(DOC).ps
        ps2pdf $(DOC).ps $(DOC).pdf

limpieza:
        rm $(DOC).aux $(DOC).log $(DOC).bbl $(DOC).blg
```

después de que tenemos la instrucción `make` para generar `documento.pdf` y `make limpieza` para remover los cuatro archivos auxiliares que ya no nos interesan cuando está listo el documento. Nota que el espacio enfrente de las líneas es un **tabulador** y no funcionará sin serlo.

Otro ejemplo, de programación en el lenguaje C++, produce el ejecutable del código fuente:

```
COMPILADOR = g++
EJECUTABLE=programa
OPCIONES= -Wall -o $(EJECUTABLE)
FUENTES= main.cpp funciones.cpp auxiliares.cpp
OBJETOS=$(FUENTES:.cpp=.o)
```

```
all: $(EJECUTABLE)

$(EJECUTABLE):$(FUENTES)
        $(COMPILADOR) $(OPCIONES) $(FUENTES)

limpieza:
        rm $(EJECUTABLE)
```

Si alguno de los archivos de cuales depende un archivo `X` han sido modificados después del tiempo de modificación de `X` mismo, `X` será regenerado al momento de ejecutar `make` que implica que sea necesario crear `X` como el producto final o el producto intermediado de `make`.

Una línea larga se puede dividir con el símbolo \:

```
# este es un comentario
algo que ocupa mucho espacio se ve mejor cortado en varias piezas

# lo de arriba es igual a lo de abajo
algo que ocupa mucho espacio \
se ve mejor \ # incluso puedo poner comentarios y no afecta nada
cortado en varias piezas
```

Cada instrucción de shell siempre devuelve su estado de salida, que es un número entero. El valor cero significa que todo fue bien, y un valor no cero significa que ocurrió algún error. `make` siempre examina el valor de salida de cada línea de shell ejecutada y *termina la ejecución* al encontrar un valor no cero.

Sin embargo, algunas instrucciones pueden tener estado de salida no cero aunque todo fue bien, por falta de cuidado del programador. Para los casos donde no es deseable que `make` termine ejecución por encontrar un estado de salida no cero, se puede añadir el prefijo `-` a la línea de shell para que `make` ignore el estado de salida.

Todas las variables ambientales de Unix son automáticamente considerados como marcos por `make`, por lo cual por ejemplo la expresión `$(PATH)` estará reemplazada por el valor de la variable `PATH` del sistema. Marcos pueden ser definidos también en la línea de instrucciones y las definiciones dadas en la línea de instrucciones reemplazan las definiciones dadas en el archivo `Makefile`. Por ejemplo, al ejecutar `make TARGET=report` reemplazará cada ocurrencia de `$(TARGET)` del archivo `Makefile` por `report` durante la ejecución de `make`, *sin* cambiar el archivo `Makefile`. Si la definición de un macro en la línea de instrucciones contiene espacio blanco, habrá que ponerlo en cita: `¨TARGET=data.txt analysis.txt¨`.

Existen marcos especiales para acceder ciertas partes de las reglas que se puede utilizar en las línea shell de una dependencia: `$@` refiere a la parte izquierda de la dependencia y `$^` refiere a la parte derecha, mientras `$<` significa “la primera de las dependencias”. Un ejemplo es el mismo `Makefile` usado para generar este documento:

```
DOC = taller

all: $(DOC).pdf
```

```
$(DOC).dvi: $(DOC).tex $(DOC).bib
        latex $<
        bibtex $(DOC)
        latex $<
        latex $<

$(DOC).ps: $(DOC).dvi
        dvips -Pcmz -o $@ $^

$(DOC).pdf: $(DOC).ps
        ps2pdf $^ $@
        scp $@ elisa@yalma.fime.uanl.mx:\
/home/elisa/public_html/teaching/taller/$@

clean:
        rm -f $(DOC).aux $(DOC).dvi $(DOC).log \
$(DOC).bbl $(DOC).blg
```

También se puede generar marcos basados en otros macros, que resulta útil en muchos casos de programación. Si por ejemplo queremos utilizar el marco `FUENTES` que ya contiene la lista de los archivos de código fuente en el lenguaje C para obtener una lista de los archivos objetos `.o` generados por el compilador, definimos `OBJECTOS = $(FUENTES,.c=.o)` que resulta que la parte antes de la igualdad = será reemplazado en cada ocurrencia en la significado de `$(FUENTES)` por la parte después: es decir, cada `.c` en `$(FUENTES)` será un `.o` en `$OBJECTOS`, pero nada más cambia.

`make` cuenta con varias definiciones más para ayudar a crear archivos `Makefile` más generales y dinámicas. Para más información, se recomienda el manual oficial de GNU Make [13].

7.2. Ejecución trasfondo

7.2.1. &, `Control-z` y `fg`

En Unix, se puede ejecutar un programa *trasfondo*, es decir, sin "bloquear" el terminal por lo cual se lanza la instrucción. Por ejemplo, para abrir `emacs`, si el terminal está gráfico, abre una ventana nueva para emacs, pero el terminal ya no acepta más instrucciones. Al añadir "&" al fin de la instrucción, el programa inicia trasfondo:

```
> emacs &
[1] 7286
>
```

La ventana donde está `emacs` funciona, y además la ventana terminal queda disponible para acceso. Nota que en cualquier caso no se puede cerrar la ventana terminal sin afectar a los programas iniciados desde ella.

En terminales no gráficos, ejecutar por ejemplo computaciones largas en el trasfondo es útil. La ejecución trasfondo también ayuda a "esconder" programas que continúan activos mientras se hace otra cosa. Varios programas pueden ser *suspendidas*, o sea, enviados al trasfondo, con teclear

`Control-z`. Con la instrucción `fg`, se trae programas del trasfondo al terminal. Si hay varios programas en el trasfondo, al dar la identidad del proceso como parámetro, `fg` puede elegir un dado programa para "devolver". Para ver los trabajos en trasfondo, se usa la instrucción `jobs` que imprime una lista enumerada de los trabajos[1] En el ejemplo siguiente, se comienza la ejecución de `pico` en el trasfondo, después iniciando también `irssi` en el trasfondo. Al ejecutar `jobs`, se obtiene una lista de los trabajos, y con `fg 2`, vuelve a manejar `irssi`. Al terminar `irssi`, de puede recuperar `pico` con solamente `fg`, como ya no queda nada más que un trabajo en el trasfondo.

```
> pico &
[1] 7552
> irssi &
[2] 7553

[1]+  Stopped                 pico
> jobs
[1]-  Stopped                 pico
[2]+  Stopped                 irssi
> fg 2
irssi
> fg
pico
>
```

7.2.2. `at`

Es posible enviar a un trabajo para ser ejecutado a otro momento sin estar "presente": por ejecutar `at -m1235 today` podemos crear un trabajo para ejecutar hoy a las 12:35 horas. Por poner `tomorrow`, sería mañana a la hora especificada. En el prompt que abra que define las instrucciones para ejecutar, una por línea, y al haberlos escritas todas, con `Control-D` se sale de `at`. Con `atq` uno puede ver lo que contiene la cola se trabajos que todavía no se ha ejecutado:

```
> at -m 13:31 today
at> ls -l > test.txt
at> sort test.txt > sort.txt
at> <EOT>
commands will be executed using /bin/tcsh
job 1183660260.a at Thu Jul  5 13:31:00 2007
> atq
 Rank     Execution Date     Owner     Job         Queue   Job Name
  1st   Jul  5, 2007 13:31   elisa   1183660260.a     a     stdin
```

Para saber más, ver `man at`.

7.2.3. `screen`

`screen` es una herramienta de "ventanas" para terminales textuales de sistemas operativos tipo Unix, que permite la creación de varias "ventanas" para poder ejecutar varias tareas simultáneamente. Lo

[1] Para ver todos los *procesos*, se utiliza `ps`.

mejor de screen es que uno puede salir del sistema (es decir, hacer logout) y dejar algunas tareas ejecutando en la máquina aunque el usuario no tiene una sesión interactiva iniciada en el sistema.

Para iniciar screen, se puede ejecutar simplemente la instrucción screen para inicial un shell para ejecutar instrucciones bajo del control de screen o opcionalmente definir directamente el programa para correr bajo del control de screen: por ejemplo, screen pine ejecuta la herramienta de acceso a correo electrónico pine (de Sección 2.3.2) en screen. El cuadro 7.1 muestra las instrucciones más importantes que se puede dar al screen.

Cuadro 7.1: Instrucciones básicas de screen. La notación Control-x significa que hay que teclear *al mismo tiempo* Control y x.

Control-a c	Crear una nueva ventana bajo screen, empezando con un shell
Control-a ?	Ver la lista de instrucciones posibles
Control-a n	Mover a la ventana siguiente el en orden de creación de las ventanas
Control-a p	Mover a la ventana anterior en el orden de creación de las ventanas
Control-a ⟨número⟩	Mover a la ventana número ⟨número⟩, donde la primera es cero
Control-a d	Dejar screen al trasfondo para correr incluso al salir del sistema

Para enviar el screen al trasfondo, hay que teclear primero Control-a y después d. El sistema imprime un mensaje [detached] y se vuelve al shell original de donde se inició el screen. Para salir completamente de screen así que terminen todos los programas ejecutando en el screen, solamente hay que ejecutar exit (o teclear Control-c), o en el caso que screen se inició con algún aplicación como pine, salir de la aplicación. Al haber terminado una instancia de screen, se ve el mensaje [screen is terminating].

Para *volver* a un screen existente, se ejecuta screen -r — poner solamente screen crea una nueva instancia de screen que no tiene acceso a las ventanas creadas en otras instancias. Si el usuario ya tiene varias instancias de screen ejecutando, el sistema imprime una lista de las posibilidades:

```
> screen -r
There are several suitable screens on:
        14058.pts-4.elisa       (Detached)
        14050.pts-4.elisa       (Detached)
Type "screen [-d] -r [pid.]tty.host" to resume one of them.
>
```

Si no existen instancias, ejecutar screen -r resulta en el mensaje There is no screen to be resumed. Para elegir a cuál instancia volver, hay que indicar o el número de identificación del proceso (en este caso, 14058 o 14050). En el caso que la instancia ya está activado por otra parte (por ejemplo, el usuario ha hecho un login de remoto y olvidado la instancia conectada), se puede "robar" la activación por screen -rd. Solamente quitar la activación sin activarla se hace con screen -d.

La lista dada por `screen -r` indica el estado de cada instancia. “Attached” significa que está activa en alguna parte y “Detached” que no hay acceso actual en ninguna parte. Intentando a activar una instancia ya activa por `screen -r` resulta en un error:

```
> screen -r
There are several suitable screens on:
        14278.pts-4.elisa       (Attached)
        14273.pts-4.elisa       (Detached)
        14269.pts-4.elisa       (Detached)
Type "screen [-d] -r [pid.]tty.host" to resume one of them.
> screen -r 14278
There is a screen on:
        14278.pts-4.elisa       (Attached)
There is no screen to be resumed matching 14278.
>
```

Para “compartir” na instancia así que sea activa en más que una sesión, se utiliza `screen -x`. De esta manera se puede colaborar entre varias personas con acceso al mismo shell o programa o dejar otra gente observar cómo se realiza alguna operación. Técnicamente es también posible compartir una instancia de `screen` entre múltiples usuarios: para instrucciones, busca por Google por “screen multiuser”.

Capítulo 8

Programación matemática

8.1. Octave

Octave [8] es algo entre un lenguaje de programación y una herramienta, de la misma manera que `gnuplot`: el usuario escribe instrucciones y Octave entrega resultados matemáticos y/o diagramas. En su funcionamiento es algo parecido a la herramienta comercial Matlab y su lenguaje es altamente compatible con lo de Matlab.

Matlab (MATrix LABoratory) fue creado inicialmente como un laboratorio de matrices que permitiera manipularlas fácilmente, realizar operaciones básicas entre ellas y factorizaciones matriciales. Ha tenido una gran evolución desde su primera versión y hoy en día es un software con grandes capacidades de visualización, rápida experimentación y cuenta con algoritmos muy eficientes tanto de Álgebra lineal como de otras áreas de matemáticas.

Octave nace como un software de apoyo a una clase de química en la Universidad de Texas y es como una versión libre que opera prácticamente igual que Matlab. Ha ido evolucionando, cuenta con algoritmos confiables y es una buena opción para operar con matrices y algo más.

Una de las capacidades del software es la fácil graficación tanto en dos como en tres dimensiones. Matlab cuenta con su propio editor de gráficas y Octave grafica utilizando en gnuplot, que también es software libre.

Una de las grandes ventajas de trabajar con Octave o Matlab es que no hay necesidad de declarar variables, las variables pueden ser escalares, vectores, matrices o números complejos. Para declarar las variables se puede hacer directamente del prompt o desde un programa, iniciemos en el prompt.

Se distinguen las variables mayúsculas de las minúsculas, es decir A y a pueden tener valores distintos. La forma de introducir instrucciones es la siguiente

variable = expresión

o simplemente

expresión

Por ejemplo al introducir $x = exp(5 - sin(\pi/2))$ produce

$$x = 54{,}5982$$

mientras que si no se asigna una variable a la expresión, el resultado obtenido se guarda en la variable ans, por ejemplo, si se introduce

$$exp(5 - sin(\pi/2))$$

se obtendrá

$$\text{ans} = 54{,}5982$$

Si el último carácter de una instrucción es un punto y coma, no se despliega en pantalla el resultado, pero si éste se asigna a una variable, guardará el nombre de la variable y su valor.

8.1.1. Vectores

Para introducir vectores se procede como sigue: El vector $x = (1 \quad 2 \quad 3)$, se introduce como

```
x=[1   2   3]
```

; para vectores columna se usa

```
x=[1   2   3]'
```

,

o bien

```
x=[1;2;3].
```

Para hacer referencia a un elemento del vector se escribe **x=(i)** donde i es la entrada del vector que se requiere, por ejemplo, si se declara el vector

```
x=[ -1   5   9]
```

para encontrar el elemento 2 del vector x, se teclea **x(2)**.

Se pueden generar vectores con puntos igualmente espaciados, por ejemplo para generar un vector con puntos igualmente espaciados en el intervalo $[0, 2\pi]$ se puede utilizar la instrucción

```
x=0:0.1:2*pi
```

mientras con la instrucción

```
x=linspace(0,2*pi,50)
```

se genera un vector con 50 puntos igualmente espaciados que parte de 0 y llega hasta 2π.

Se tienen algunas funciones definidas de forma intrínseca, que operan para escalares, matrices o vectores, por ejemplo la función seno, apliquemos esa al vector x que tenemos ya definido,

```
y=sin(x).
```

Como ***x*** es un vector, entonces **sin(*x*)** es también otro vector de la misma dimensión de ***x***, cuyas componentes son la función seno aplicada a cada una de las componentes del vector ***x***, esto es, *y*(*i*) = sen(*x*(*i*)).

Para graficar los vectores se utiliza la instrucción `plot`, en nuestro caso para graficar la función seno tecleamos `plot(x,y)`.

Se pueden poner etiquetas en los ejes o título a la gráfica, se puede cambiar de color o el estilo de la gráfica, se puede teclear **help plot**.

De hecho, esta es una de las ventajas del programa, tiene una ayuda en línea para todas las instrucciones, en cualquier momento si no recordamos cómo utilizarlo o el orden de los parámetros, o simplemente mas opciones de una instrucción, podemos pedir ayuda y nos da la información que se tenga de ella.

Supongamos ahora que se desea graficar $f(x) = 3x^3 + 2x - 3$ en el intervalo $[-2, 3]$ usando 50 puntos. La instrucción

```
x=linspace(-2,3,50)
```

genera las abscisas. Ahora, cómo encontramos *y* = *f* (*x*)?. El problema es que *x* es un vector, no un escalar, así que necesitamos realizar operaciones entre vectores. Si ***x*** es un vector y *a* un escalar entonces **a** + **x** es un vector que a cada componente de ***x*** le suma el escalar *a*.

a*x también es un vector que a cada componente de *x* la multiplica por el escalar *a*, y si ***z*** es otro vector de la misma dimensión de ***x***, entonces **x** + **z** es un vector de la misma dimensión de ***x*** donde la *i*-ésima componente de ***x*** + ***z*** viene dada por *x*(*i*) + *z*(*i*). Dados dos vectores de igual dimensión, ***x*** y ***z***, a menudo es conveniente generar un vector ***w*** tal que *w*(*i*) = *x*(*i*)**z*(*i*), esto se logra con la multiplicación elemento a elemento, la cual se define anteponiendo un punto antes del operador, en este caso tenemos que **w = x.*z**

Por ejemplo, **x.*x** es un vector cuyas componentes son $x(i)^2$. Otras operaciones elemento a elemento disponibles son

```
.*      ./      .^
```

Entonces para graficar $f(x) = 3x^3 + 2x - 3$, tenemos que

```
y=3*x.^3+2*x-3
```

genera el vector de ordenadas. Entonces la gráfica deseada se obtiene con las instrucciones

```
x=linspace(-2,3,50)
y=3*x.^3+2*x-3
plot(x,y)
```

8.1.2. Matrices

Considérese la matriz

$$a = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

Esta matriz se introduce de la siguiente forma:

```
a=[1 2;3 4]
```

o bien

$$a = [1 \quad 2 \\ 3 \quad 4]$$

Algo que resulta particularmente útil cuando no es posible escribir una instrucción en una sola línea, es teclear tres puntos al final del renglon, para indicar que se espera en la linea siguiente la continuación de la instrucción. Por ejemplo, para introducir la matriz a se usaría:

$$a = [1 \quad 2; ... \\ 3 \quad 4]$$

Para hacer referencia al elemento del renglón i y la columna j de la matriz a, se utiliza la siguiente notación:

$$a(i, j)$$

Por ejemplo, teclear **a(2,1)** y se obtendrá por resultado 3.

El i-ésimo renglón de a se obtiene por:

$$a(i, :)$$

y para la j-ésima columna de a se utiliza:

$$a(:, j)$$

Por ejemplo teclear **a(1,:)** para obtener el vector que contiene el primer renglón de la matriz a y utilice la instrucción **a(:,2)** para obtener la segunda columna.

Podemos alterar un sólo elemento de la matriz sin necesidad de modificarla toda. Por ejemplo **a(1,2)=5** nos produce

$$a = \begin{pmatrix} 1 & 5 \\ 3 & 4 \end{pmatrix}$$

y con **a(2,1)=a(2,2)** obtenemos

$$a = \begin{pmatrix} 1 & 5 \\ 4 & 4 \end{pmatrix}$$

Es posible conocer las dimensiones de una matriz (es decir, la cantidad de renglones y columnas que la componen) mediante la función size:

$$[\text{m,n}] = \text{size(A)}$$

Por otro lado, si v es un vector renglón o un vector columna, la instrucción n = length(v) asigna a n la dimensión de v.

También existen algunas instrucciones ya definidas, por ejemplo se puede calcular la inversa de la matriz A utilizando inv(A) De igual forma det(A) calcula el determinante de la matriz A, rank(A) proporciona su rango, por mencionar algunas.

Ejercicio Introducir la siguiente matriz, calcular su rango, inversa y determinante.

```
A=hilb(4)
inv(A)
det(A)
rank(A)
```

Operaciones con Matrices

Si A y B son dos matrices de igual dimensión, podemos efectuar la operación suma, resta, multiplicación, por ejemplo defina las matrices A y B como

$$A = \begin{pmatrix} 1 & 3 \\ 7 & 9 \end{pmatrix} \quad B = \begin{pmatrix} 2 & 6 \\ 4 & 5 \end{pmatrix}$$

y entonces podemos calcular su suma con la siguiente instrucción:

```
C=A+B
```

En forma análoga para la resta, la instruccion

```
C=A-B
```

La multiplicación viene dada por

```
C=A*B
```

cuidando las dimensiones para poderla efectuar.

Una vez definida una matriz A, la transpuesta se puede obtener con la instrucción

```
C=A'
```

también podemos redefinir A como su transpuesta, esto es, `A=A'`

Otra de las operaciones que podemos efectuar con matrices es elevarlas a una potencia dada, por ejemplo para elevar al cuadrado la matriz A, tenemos

$$C = A\text{^}2$$

de esta forma $C = A * A$. Y $C = A\text{^}3$ nos produce $C = A * A * A$.

En particular para resolver un sistema de ecuaciones lineales de la forma $Ax = b$, cuya solución es $x = A^{-1}b$, utilizamos la instrucción

$$x = A\backslash b$$

Ejercicio. Defina una matriz A de 3×3 y un vector b de 3×1, calcule

```
x=A \ b
```

y verifique que x es solución del sistema lineal por calcular `A*x` y comparar el con b.

Algunas matrices especiales

Matriz identidad La matriz identidad de orden n se obtiene por medio de la instrucción $a =$ eye(n) donde n es el orden de la matriz, así $a =$ eye(4) nos produce la matriz identidad de orden cuatro. También podemos obtener matrices no cuadradas, así $a =$ eye(m, n) nos produce una matriz de $m \times n$, con unos en la diagonal principal y ceros en las demás entradas.

Matriz de ceros Podemos producir una matriz cuyas entradas sean todas iguales a cero. La matriz de orden n con estas características se obtiene con la siguiente instrucción: $a =$ zeros(n) y $a =$ zeros(m, n) nos produce una matriz de orden $m \times n$ con entradas cero.

Matriz de unos Otra de las matrices muy utilizadas es la matriz con todas sus entradas iguales a uno. La matriz de orden n de unos se obtiene con: $a =$ ones(n) para matrices no cuadradas se obtiene con $a =$ ones(m, n). **Ejercicio.** ¿Cómo obtenemos una matriz con todas sus entradas iguales a dos?

Matriz aleatoria Podemos obtener matrices de orden n generadas aleatoriamente con distribución uniforme en $[0, 1]$ en la siguiente forma: $a =$ rand(n). Análogamente $a =$ rand(m, n) genera una matriz aleatoria de $m \times n$.

Algunas factorizaciones matriciales

Se cuenta con algoritmos eficientes para Álgebra lineal, en particular para factorizaciones matriciales. Algunas que podemos mencionar son las siguientes.

Factorización LU La factorización LU descompone la matriz en producto de dos matrices L triangular inferior y U triangular superior de tal forma que $A = L * U$, es el proceso de eliminación gaussiana. Si se tiene una matriz A definida, la forma de obtenerla la factorización es `[L,U]=lu(A)`

Factorización QR La factorización QR de una mattriz factoriza la matriz A en el producto de una matriz ortogonal y una triangular superior. La forma de obtenerla es `[Q,R]=qr(A)`

Eigenvalores y eigenvectores (Valores y vectores característicos o valores y vectores propios). Se puede calcular utilizando la instrucción **`e=eig(A)`**; si se utiliza de esta forma sólo da un vector con los eigenvalores, si se necesitan los eigenvectores se debe utilizar **`[V,E]=eig(A)`**. En la matriz **V** se tienen los eigenvectores y **E** es una matriz diagonal que tiene los eigenvalores en la diagonal principal.

Descomposición SVD Se obtiene con la instrucción **`[S,V,D]=svd(A)`**

En general esta es la forma de utilizar funciones: de lado izquierdo con corchetes se tienen los argumentos de salida, el orden es importante. Después viene el $=$ y nombre de la función y a la derecha con paréntesis los argumentos de entrada. En algunas funciones dependiendo del número de argumentos de entrada o salida es lo que se obtiene con la corrida del programa, un ejemplo de esto es la función **`eig`**.

8.1.3. Almacenar y recuperar variables

Antes de salir de Matlab u Octave las variables utilizadas pueden ser salvadas para ser usadas en otra sesión de la herramienta, esto se hace con save; con save temporal las variables actuales en el archivo temporal.mat.

save temporal x salva solamente la variable x, y save temporal x y z guarda las variables x, y y z. La instrucción load temporal restaura a memoria del programa todas las variables almacenadas en el archivo temporal.mat.

8.1.4. Algo de programación

Se pueden generar programas, Matlab tiene su propio editor, Octave no cuenta con editor propio se puede utilizar alguno del sistema como `emacs`. Los programas en deben tener extensión **`.m`** . y para correlos solo debemos teclear desde el prompt el nombre del programa sin extensión.

La programación es con el esquema modular, esto es, se generan subprogramas o módulos en archivos independientes que pueden ser utilizados por varios programas, esto permite reutilizar código y sólo llamar las rutinas necesarias. Se pueden listar los programas **pi_1** y **pi_2** para ver la sintaxis de programación, teclear **`type pi_1`** y **`type pi_2`**.

8.2. Optimización

8.2.1. Cplex

Ilog Cplex [18] es una herramienta de optimización, probablemente la más popular que hay actualmente. Cuenta con una edición gratuita para estudiantes (en combinación con Ampl). Con Cplex uno

puede solucionar problemas de optimización lineal y casos especiales:

- programas lineales (PL) básicos
- problemas de *flujos* (o sea, PL con una estructura especial)
- programación cuadrática (PQ): la función objectivo contiene términos cuadráticos
- programación entera mixta: PL o PQ con restricciones de (algunas) variables a valores enteros

El *Interactive Optimizer* de Cplex es un programa ejecutable para leer instrucciones de un archivo de entrada y para uso interactivo. La instrucción escplex. El ejemplo siguiente muestra el formato de entrada y el uso básico de Cplex:

```
Welcome to CPLEX Interactive Optimizer 9.0.0
with Simplex, Mixed Integer & Barrier Optimizers
Copyright (c) ILOG 1997-2003
CPLEX is a registered trademark of ILOG
Type 'help' for a list of available commands.
Type 'help' followed by a command name for more
information on commands.
CPLEX> enter example
Enter new problem ['end' on a separate line terminates]:
maximize x1 + 2 x2 + 3 x3

subject to -x1 + x2 + x3 <= 20
x1 - 3 x2 + x3 <=30
bounds
0 <= x1 <= 40
0 <= x2
0 <= x3
end
CPLEX> optimize
Tried aggregator 1 time.
No LP presolve or aggregator reductions.
Presolve time = 0.00 sec.
Iteration log . . .
Iteration: 1 Dual infeasibility = 0.000000
Iteration: 2 Dual objective = 202.500000
Dual simplex - Optimal: Objective = 2.0250000000e+002
Solution time = 0.01 sec. Iterations = 2 (1)
CPLEX> display solution variables x1-x3
Variable Name Solution Value
x1 40.000000
x2 17.500000
x3 42.500000
CPLEX> quit
```

Cplex ofrece acceso al solver desde programas escritos en los lenguajes C, Visual Basic, Fortran, etcétera. El Cplex *Callable Library* es una librería de funciones de C. En Unix, los archivos de la librería se llaman `libcplex.a`, `libcplex.so` y `libcplex.sl`, mientras en Microsoft Windows son `cplex.lib` y `cplex.dll`.

El *Concert Technology* de Cplex ofrece acceso programas escritos en los lenguajes C++ y Java (incluyendo .NET) y provee herramientas para crear y solucionar modelos, acceso a los resultados del

solver, y acceso a causas de errores. El archivo se llama `cplex.jar`. Lo siguiente es un ejemplo escrito en Java que accede a Cplex.

Para compilar el programa, se usa algo como (dependiendo de la instalación)

```
javac -classpath * /opt/ilog/cplex90/lib/cplex.jar:. Example.java
```

y para ejecutarlo se usa lo siguiente, escrito en *una sola línea* — aquí se corta por espacio limitado:

```
java -classpath /opt/ilog/cplex90/lib/cplex.jar:.
  -Djava.library.path=/opt/ilog/cplex90/bin/ultrasparc32_8_6.2/ Example
```

Los contenidos del programa — que incorporan un miniejemplo de un problema de optimización — son:

```
import ilog.concert.*;
import ilog.cplex.*;

public class Example {
  public static void main(String[] args) {
    try {

     IloCplex cplex = new IloCplex();
     double[] lb = {0.0, 0.0, 0.0};
     double[] ub = {40.0, Double.MAX_VALUE, Double.MAX_VALUE};
     IloNumVar[] x = cplex.numVarArray(3, lb, ub);
     double[] objvals = {1.0, 2.0, 3.0};

     cplex.addMaximize(cplex.scalProd(x, objvals));
     cplex.addLe(cplex.sum(cplex.prod(-1.0, x[0]),
                           cplex.prod( 1.0, x[1]),
                           cplex.prod( 1.0, x[2])), 20.0);
     cplex.addLe(cplex.sum(cplex.prod( 1.0, x[0]),
                           cplex.prod(-3.0, x[1]),
                           cplex.prod( 1.0, x[2])), 30.0);
     if (cplex.solve()) {
       cplex.output().println("Solution status = " + cplex.getStatus());
       cplex.output().println("Solution value = " + cplex.getObjValue());
       double[] val = cplex.getValues(x);
       int ncols = cplex.getNcols();
       for (int j = 0; j < ncols; ++j) {
         cplex.output().println("Column: " + j + " Value = " + val[j]);
       }
     }
     cplex.end();
    } catch (IloException e) {
      System.err.println("Concert exception '" + e + "' caught");
```

```
    }
  }
}
```

8.2.2. Gams

Gams [14] (General Algebraic Modeling System) es una herramienta para la programación matemática y optimización. Cuenta con una versión de demonstración que permite modelos con al máximo 300 restricciones, 300 variables, 2000 elementos no cero y 50 variable enteros.

Un programa de Gams es un archivo de texto con terminación `.gms`. Se puede utilizar cualquier tipografía, número de espacios, o de renglones. Los renglones que principien con una asterisco $*$, en la primer columna son considerados como comentarios.

En formato de Gams no distingue entre letras mayúsculas y minúsculas. Los nombres de las entidades en Gams deben empezar con una letra y pueden ser de hasta nueve caracteres. La única regla en cuanto a orden en Gams es que no se puede utilizar un argumento no sea declarado con anticipación. El punto y coma `;` indica el final de cualquier operación, ya sea de declaración o asignación. Gams maneja cuatro tipos de registros que deben ser declarados:

- Datos
  - set i rutas disponibles `/1,2,3,4/;`
  - `Parameter capacidad(i);`
  - `Table distancia(i,j);`
  - `Scalar f costo por retraso /90/;`
- Variables
  - `variable x(i,j), y;`
  - `binary variable y;`
  - `positive variable producción (j), inventario;`
- Ecuaciones
  - `equations`
  - `costo función objetivo`
  - `demanda(j) demanda en de la ciudad j;`
- Modelos
  - `model capacidades /all/;`
  - `model modelo1 /costo,funcion1,funcion2,funcion3/;`

Antes de correr el modelo GAMS ejecuta una compilación para localizar errores de sintaxis. De haber un error de compilación, avisa al usuario que hubo un error, e imprime el archivo de listado con el programa marcando con un número clave el error que ocurrió y en donde fue. En el manual de usuario aparece la lista de errores posibles; aunque por lo general son auto-explicativos. Por lo general basta con ejecutar `gams` (por ejemplo `gamsejemplo.gms`, ver cuadro 8.1).

Se puede especificar que solver utilizar, e incluso adicionar un archivo con parámetros para el solver como número de iteraciones, o de decimales a tomar en cuenta, datos de las derivadas, entre otros. Los tipos de Modelos en GAMS son los siguientes:

- `LP`: modelos líneales
- `NLP`: modelos *no-líneales*
- `MIP`: modelos *enteros-mixtos* líneales
- `MINLP`: modelos enteros-mixtos no-líneales
- `rMIP`: modelos mixtos líneales *relajados*
- `rMINLP`: modelos mixtos no-líneales relajados
- `DNLP`: modelos no-líneales con *discontinuidades* en la derivada
- `MCP`: modelos mixtos de *complementariedad*
- `CNS`: sistemas no-líneales *restringidos*

Después de correr un modelo, GAMS creará un archivo con el mismo nombre del programa, pero con terminación `.lst` en el mismo directorio donde se encuentre el programa. Se le puede pedir a GAMS el mandar resultados específicos de interés como puede ser formatos de salida, datos intermedios de en las iteraciones, etc.

En la página de GAMS en `http://www.gams.com`, se puede encontrar el manual del usuario, asi como diversos tutoriales y una *extensa* libreria de programas para modelos de todas índoles de la optimización. Dentro de la documentación disponible, se encuentran los manuales de cada solver, y las formas en que se puede interactuar con ellos.

Asignación o definición de funciones:

```
OBJ .. Z =e= SUM( (i,j), C(i,J)*X(i,j) );
ASI(J).. SUM( I, X(i,j)) =e= 1;
ASJ(I).. SUM(J, X(i,j)) =e= 1;
```

Una vez que se ha definido un modelo en GAMS, se puede entonces resolver con un llamado a un solver. El solver que utilizará GAMS será en este caso el que tiene prestablecido por default. Se puede escoger que solver utilizar, por ejemplo con la siguiente instrucción `OPTION MIP=cplex;` (para elegir CPLEX como el solver).

Cuadro 8.1: Estructura de un modelo Gams: declaración y asignacion de parámetros, declaración de variables, definición de ecuaciones, armar modelo y lamar a solver.

```
$ TITLE Test Problem
SETS
I corrientes /A, B, C, D/
J intercambiadores /1*4/;
TABLE C(i,j) Costo de Asignarle a la corriente i el intercambiador j
1 2 3 4
A 94 1 54 68
B 74 10 88 82
C 73 88 8 76
D 11 74 81 21;
VARIABLES X(I,J) , Z;
BINARY VARIABLES X(i,j);
EQUATIONS
ASI(J), ASJ(I), OBJ;
OBJ .. Z =e= SUM( (i,j), C(i,J)*X(i,j) );
ASI(J).. SUM( I, X(i,j) )=e= 1;
ASJ(I).. SUM(J, X(i,j) )=e= 1;
MODEL HEAT /ALL/;
solve HEAT using MIP minimizing Z;
```

`solve HEAT`	`using MIP`	`minimizing`	`Z;`
El nombre con el que se definó en modelo	Tipo de modelo	Minimizar o maximizar	Una variable declarada

Para variables, aplical las extensiones siguientes:

- `.l` = en el estado actual
- `.up` = cota superior
- `.lo` = cota inaferior
- `.m` = multiplicadores del simplex

Para conjuntos, Gams ofrece los operadores `card` y `ord`: por ejemplo, para un `set I /1*3/;`, `ord(i)` puede ser 1, 2 o 3 y `card(i)` es la cardinalidad del conjunto, en este caso tres.

También cuenta con un operador *condicional* `$` que se puede utilizar al definir ecuaciones o dentro de ecuaciones:

```
distancia(i,j)$(uso(i) ne 1)..
costo.. Sum((i,j)$(ord(i) ne ord(j)), variable (i,j)=l= 5
```

La instrucción `$include` permite incluir lo que existe en un archivo dentro del programa de Gams. Por ejemplo con `$include ex4.dat` dentro de un progrqama, entonces correrá con los datos del archivo `ex4.dat`. Se pueden correr varios ejemplos con tan solo cambiar este renglón. Gams cuenta con operaciones iterativos (tipo "while" de C/C++ o Java):

```
Set i iteration counter /1*30/;
Scalar UB=inf
LB = -inf
count;
Loop (i$((UB-LB) ge 0.001)
count=ord(i);
solve nlpmodel using nlp minimizing nlpobj;
UB $(nlpobj.l le UB) = nlpobj.l;
Milp_param(i)=nlp_var.l;
Solve milpmodel using mip minimizing milpobj;
LB$(milpobj.l ge LB0 = milpobj.l;
Nlp_param(i+1)= milp_var.l
);
```

También tiene una clausa condicional:

```
IF LOOP(i, solve nlpmodel using nlp minimizing nlpobj;
if((nlpobj.l ge milpobj.l),
   solve milpmodel using mip minimizing milpobj);
);
```

Se puede bajar la última versión de GAMS de la página de internet. No se requiere una licencia nueva para una versión nueva Hay que revisar que solver son los que estan habilitados en GAMS ya que estos dependen de la licencia que se haya contratado.

8.2.3. AMPL

AMPL (A Modeling Language for Mathematical Programming) [21] es un lenguage de modelado algebráico para optimizacioón líneal y no líneal con variables discretas y/o continuas. AMPL cuenta con una licencia estudiantíl gratuita con funcionamiento limitado.

8.2.4. Lindo y Lingo

LINDO (*Linear, Interactive, Discrete Optimizer*) (`http://www.lindo.com/`) es otro solver que cuenta con una versión de evaluación gratuita (de LINGO 10.0). LINDO es un solver interactivo para PL y además programación cuadrática y entera con funciones de análisis de sensibilidad. El formato de sus archivos de entrada es el siguiente:

```
max 2a + 3.5b + 4.1c - 2.6d  + 3.3e - 6.2f
st
a + b + c + d + e + f
```

El largo máximo de linea es 71 símbolos. El formato de salida es el siguiente:

```
LP OPTIMUM FOUND AT STEP      2
```

```
   OBJECTIVE FUNCTION VALUE

   1)      7.454545

VARIABLE        VALUE          REDUCED COST
    X         1.272727          0.000000
    Y         1.636364          0.000000


  ROW   SLACK OR SURPLUS     DUAL PRICES
   2)         0.000000          0.090909
   3)         0.000000          0.545455

NO. ITERATIONS=       2
```

Capítulo 9

Seguridad

9.1. Programas anti-virus

Un *anti-virus* es un programa que busca, detecta y elímina virus informáticos y cualquier tipo de programas que afecten el sistema operativo ya sea de una computadora o de una red interna. Antivirus es creado para evitar o prevenir que algún virus ataque nuestro sistema operativo en la computadora. O también para que no se propague y contagie a otras computadoras o se haga más grande el dano. Las funciones principales de los antivirus son *vacunar* por instalar un programa en la memoria que actua como un filtro de los programas ejecutados en tiempo real, *detectar* por examinar los archivos existentes en disco o los que se le indique en una ruta especifica y *eliminar* a los virus desactivando su estructura y despues reparando la de los archivos donde se alojaba el virus.

El antivirus lo que hace es compara el código de cada archivo con una base de datos de los códigos de los virus conocidos, por lo cual es importante actulizar tu computadora frecuentemente para poder descubrir las definiciones nuevas de tipos de virus, ya que los virus informáticos cada rato se crean nuevos y si no tienes actualizado tu computadora, puede llegarte alguno nuevo o de reciente creación que no lo tengas registrado en tu antivirus.

Una cosa importante para cualquier persona que tiene computadora es tener un antivirus ya que sin uno de estos, te pueden atacar un virus y estos te pueden robar información, borrar la información del disco disco, implantar información erronea o simplemente hechar a perder la instalación del sistema operativo de la máquina.

Existen diferentes herramientas antivirus que son gratuitas o cuenten con una versión de evaluación gratuita. Para mencionar algunas,

- `http://free.grisoft.com` para Windows.
- `http://pack.google.com.mx` para Windows XP o Vista.
- `http://www.clamwin.com` para Windows 98/Me/2000/XP y 2003.

- `http://www.pandasoftware.es/descargas/linux.htm` para Linux.
- `http://www.avast.com/eng/download-avast-home.html` para Linux.

9.1.1. Gusanos

Los *gusanos* (inglés: worm) son virus informáticos que se menten a la memoria y saturan la misma para que la pc se vuelva más lenta o para que se repite mucho una tarea simple hasta acabarte la memoria disponible. Estos se pueden contagiar mediante un correo electrónico o recibir un archivo de una persona desconocida de la red. Existen diferentes tipos de gusanos pero la mayoria ataca de la misma forma. Algunos ejemplos de gusanos conocidos son el Sasser y el Blaster.

9.1.2. Programas espias

Los *programas espias* (inglés: spyware) son programas que roban información a una persona o una empresa sin el concentimiento de las misma. Estos espias pueden obtener casi todo lo que buscan desde correo electrónico, direcciones, teléfonos información privada de una empresa que puede ser chica, media o grande o cualquier documentos restringido por un usario. Estos programas puedes obtenerlos mediante correos electrónicos, virus o por troyanos según sea el caso si te lo envian o si lo contraes por accidente.

Algunos programas espias son el Gator, el Bonzi Buddy y el Kazaa. Existen programas anti-spyware para darse cuenta de los programas espias. Estes inclyuen por ejemplo Spybot, Ad-Aware y SpywareBlaster.

9.1.3. Troyanos

Un *troyano* (inglés: troyan) es un programa que te mandan o que te regalan que parece de mucha utilidad para el usuario pero no es más que una trampa para que recibas un virus que deshabilita tu antivirus o si se tiene un servidor de seguridad. Este te puede llegar a tu correo electrónico como un regalo, pero en si cuando parece todo legítimo te sale la sorpresa de que se a infectado la máquina. Hace poco estuvo mandándose un correo electrónico donde te daba actualizaciones gratuitas y rápidas de plataformas comunes que hay en el entorno, pero no era más que un deshabilitador de antivirus.

9.1.4. AVG

La herramienta AVG [17] viene en tres versiones gratuitas: *Anti-virus AVG Free*, *AVG Anti-Spyware Free* y *AVG Anti-Rootkit Free* — la licencia es para uso personal no comercial solamente y solamente hasta tres computadoras pueden utilizar la misma licencia. Cuenten con actualizaciones gratuitos por internet. Son fáciles de instalar y operar en Windows.

- **Anti-virus AVG** es un antivirus que escanea archivos y correo electrónico.
- **Anti-Spyware AVG** es un anti-espía igualmente freeware.
- **Anti-Rootkit AVG** es para descubrir a los "rootkits" ya que estos se ocultan en su PC para también a su vez esconder a ciertos tipos de virus como los troyanos entre otros; este software los detecta y los borra.

9.2. Cortafuegos

Los cortafuegos (ingl. *firewall*) son elementos que podemos encontrar tanto hardware o software utilizado en una red de computadoras locales o de oficina para controlar las comunicaciones entre ellas, ya sea permitiendo la entrada o negando la misma ha personas ajenas a la institucióncon un control de reglas o politicas que hace la empresa para su privacidad y control de la red en la empresa.

Un cortafuegos bien instalado o configurado da una seguridad a la empresa y al encargado del area de redes ya que ayuda a poner seguridad a la empresa y restringiendo partes de la internet a los usuarios locales para que no se entretenga en paginas no aceptables para la gente de sistemas de la empresa.

Hay varios tipos de cortafuegos: cortafuego de capa de red o de filtrado de paquetes, cortafuego de capa de aplicacióny cortafuego personal. El primero trabaja bajo los permisos que le den las capas o niveles del modelo OSI ya qu trabaja en primero sobre la capa de 3 con los protocolos de red sobre todo con el TCP/IP donde se piden los permisos, despues va sobre la capa 4 de transporte donde se va a puerto origen y destino y ahi se le asignan otros permisos o restricciones y por ultimo cae a la capa 2 donde se va a la direccion MAC para comprobar o reafirmar las restricciones.

La capa de aplicación trabaja también con el modelo OSI con la capa de nivel siete que es la de aplicación. En esta capa los filtros o restricciones pueden adaptarse a los protocolos de red donde se verifica la entrada de los datos si son legibles o permitidos dentro de la empresa. Un ejemplo que manejan en la red es que puedes mediante las restricciones del URL delimitar las entradas del HTTP — este se le conoce como *proxy*. Y ya por ultimo el personal se instala por medio de un software delimitando ya sea el mismo softaware las limitantes o manualmente cuales quieres que sean las áreas donde se puede explorar o las zonas prohibidas que se desean cancelar.

Algunas ventajas de usar un cortafuego son que solamente deja entrar a personal autorizado a la organizacióno a la internet mediante una autorización o una clave y además dentro de la misma organización restringe información para que no cualquier usuario puede acceser a información confidencial y segmentar las áreas de trabajo para cada usuario dando la información que necesita. También agiliza la comunicación dentro de la empresa entre usuarios de diferentes niveles ya que es una comunicacion interna sin necesidad de meterse de lleno a la red y reconfigura los parámetros de seguridad.

Los desventajas incluyen que un cortafuegos no puede defenderse de ataques que no sean desde su punto de operación y tampoco pueden defenderse de usuarios internos que traicionen las políticas y

las delimitantes del cortafuegos sobre todo de los encargados de el área de sistemas. Otra desventaja es que no puede contrarestar virus que mediante un archivo o disco se hayan infectado la red interna por un usuario interno o por un agresor que entró a la institución y tuvo acceso a la red.

Para los sistemas operativos de Microsoft Windows, un ejemplo de los cortafuegos gratuitos para uso privado (no comercial) es *ZoneAlarm* [6]. Lo problemático con ZoneAlarm es que no se desinstala fácilmente — después de la desinstalación normal hay que limpiar algunas carpetas y archivos ocultos y además reconfigurar el propio cortafuegos del sistema operativo. Es en cualquier caso recomendable contar con un cortafuegos activo y muchos no quieren confiar en lo que viene con su sistema operativo.

Capítulo 10

Linux

10.1. Particiones del disco duro

Para instalar varios sistemas operativos en una computadora, uno tiene que tener un disco duro por cada sistema operativo o alternativamente *compartir* un disco entre dos o más sistemas.

El particionamiento de un disco duro se trata de la creación de *divisiones lógicas* para poder aplicar varios formatos lógicos de sistemas de archivos.

Los tres tipos de particiones son: primaria, extendida y lógica. Una partición primaria contiene un sistema operativo. Versiones antiguas de Microsoft Windows exigen que la partición conocida como `C:` en Windows sea una partición primaria. La partición primaria *activa* (típicamente una sola) contiene la información utilizada para cargar un sistema operativo en el momento de inicial la computadora.

Típicamente se puede crear un máximo de *cuatro* particiones primarias o extendidas por disco duro. Solamente una de las cuatro particiones permitidas puede ser extentendida, es decir, dividida en una o más particiones lógicas. El concepto de una partición extendida existe puramente para poder superar la limitación de tener no más que cuatro particiones definidas. Desafortunadamente no todos los sistemas operativos son capaces de iniciar de una partición lógica, por lo cual es recomendable en general utilizar particiones lógicas puramente para almacenamiento de datos. Windows asigna una letra diferente para cada partición primaria y para cada partición lógica.

Es bueno almacenar los datos en particiones distintas, porque así es posible intentar recuperarlas si otra parte del sistema falla. Por ejemplo, es posible reinstalar un sistema operativo, incluso formatear otras particiones, sin perder o dañar los datos guardados en otras particiones. Las particiones también sirven para mejorar el performans de la computadora en la presencia de discos duros muy grandes por estructurar los datos en particiones para acceso más rápido.

En Windows, existen varias herramientas (comerciales) para particionar discos duros, como el Partition Magic de PowerQuest

En sistemas tipo Unix por lo general se necesitan por lo menos tres particiones: una para el sistema raíz (inglés: root) que se conoce como la carpeta `/`, otra para datos de usuarios y una tercera para la memoria virtual (inglés: swap). Las particiones típicas adicionales incluyen `/home`, `/tmp`, `/usr`, `/var` y `/opt`.

El arte está en cómo determinar el *tamaño* asignado a cada partición. Situaciones donde se acaba el espacio en la partición de un sistema operativo o una parte de un sistema, mientras otras particiones todavía cuentan con mucho espacio libre, son causa de mucha frustración. Técnicamente es posible reajustar los tamaños de las particiones, pero por lo general es mucho mejor evaluar con anticipación la necesidad de espacio para cada partición. Por lo general, sistemas operativos de Microsoft Windows y sus aplicaciones necesitan mucho más espacio que los de tipo Unix.

Cada partición tiene definido su tipo de archivo. Sistemas operativos de Microsoft Windows típicamente necesitan ser instalados en particiones de tipo NTFS (New Technology File System). Para tener acceso a una partición de datos de sistemas operativos de tipo Windows y de tipo Linux, el formato recomendable es FAT32. Para Linux, el formato recomendable es `ext3` (un mejoramiento al formato `ext2`).

10.2. Distribuciones disponibles

Linux es un sistema de libre distribución por lo que se pueden encontrar todos los archivos y programas necesarios para su funcionamiento en una multitud de servidores conectados a Internet. La tarea de reunir todos los archivos y programas necesarios, así como instalarlos en tu sistema y configurarlo, puede ser una tarea bastante complicada y no apta para muchos. Por esto mismo, nacieron las llamadas distribuciones de Linux; empresas y organizaciones que se dedican a hacer el trabajo "sucio" para nuestro beneficio y comodidad.

Una distribución de Linux es simplemente un conjunto de programas recopilados a lo largo y ancho de sitios en Internet, organizados de tal manera que ofrezcan una solución particular o general hacia él o los usuarios. Estas distribuciones se pueden obtener a través de Internet, o comprando los CDs de las mismas, los cuales contendrán todo lo necesario para instalar un sistema Linux bastante completo y en la mayoría de los casos un programa de instalación que nos ayudara en la tarea de una primera instalación. Casi todos los principales distribuidores de Linux, ofrecen la posibilidad de bajarse sus distribuciones, vía FTP (sin cargo alguno).

Algunas de las distribuciones de Linux más populares son las siguientes:

RedHat `http://www.redhat.com` — Esta es una distribución que tiene muy buena calidad. La empresa que lo distribuye se encarga del soporte de la misma. Es necesario el pago de una licencia de soporte. Está enfocada a empresas.

Fedora `http://fedora.redhat.com` — Esta es una distribución patrocinada por RedHat y soportada por la comunidad. Es fácil de instalar y de buena calidad.

Debian `http://www.debian.org` — Denotada como la mejor del mundo. Es un proyecto totalmente no-comercial. Es posiblemente la distribución más estable y confiable, aunque no la más actualizada. Es también famosa por su reputación de ser difícil de instalar, a menos que el usuario tenga un profundo conocimiento del hardware de la computadora. Está enfocada primordialmente a desarrolladores, programadores, administradores de red y centros de computo de alto desempeño.

OpenSuSE Fácil de instalar. Versión libre de la distribución comercial SuSE.

Suse `http://www.suse.com` — Muy buena calidad, contenidos y soporte a los usuarios por parte de la empresa que la distribuye, Novell. Es necesario el pago de una licencia de soporte. Está enfocada a empresas.

Slackware `http://www.slackware.com` — Esta distribución es de las primeras que existió. Es extremadamente estable y segura, muy recomendada para servidores. La configuración no es fácil debido a que no ofrece herramientas de configuración gráficas, se mantiene con un instalador basado en texto.

Gentoo `http://www.gentoo.org` — Esta distribución es una de las únicas que han incorporado un concepto totalmente nuevo en Linux. Es una sistema inspirado en BSD-ports. Puedes compilar/optimizar vuestro sistema completamente desde cero. No es recomendable adentrarse en esta distribución sin una buena conexión a internet, un ordenador medianamente potente (si quieres terminar de compilar en un tiempo prudencial) y cierta experiencia en sistemas Unix. El proceso de instalación no es sencillo. No se recomienda para servidores con funciones críticas.

Ubuntu `http://www.ubuntu.com` — Distribución basada en Debian, con lo que esto conlleva y centrada en el usuario final. Muy popular y con mucho soporte en la comunidad. El entorno de escritorio por defecto es GNOME.

Kubuntu `http://www.kubuntu.com` — Distribución basada en Ubuntu, con lo que esto conlleva y centrada en el usuario final y facilidad de uso. La gran diferencia con Ubuntu es que el entorno de escritorio por defecto es KDE.

Mandriva `http://www.mandrivalinux.org` — Esta distribución fue creada en 1998 con el objetivo de acercar el uso de Linux a todos los usuarios, en un principio se llamo Mandrake Linux. Está enfocada a usuario de computo del hogar, oficina y escuelas. Es bastante fácil de instalar, amigable y con una gran cantidad de paquetes para comenzar a conocer Linux en serio.

La elección de una distribución depende de las necesidades del usuario y de gustos personales. Distribuciones como Debian, Slackware y Gentoo son distribuciones más avanzadas que requieren muchos conocimientos para poder ser usadas eficientemente. Mandriva, RedHat y SuSE son distribuciones más recomendadas para gente que comienza en el mundo de Linux.

La facilidad de actualización y mantenimento depende de la distribución elegida, pero la mayoría ya cuenta con herramientas semiautomáticas para descargar e instalar actualizaciones igual como software adicional.

Bibliografía

[1] activePDF Inc. Free PDF Creator — PrimoPDF. http://www.primopdf.com/.

[2] Adobe Systems Incorporated. Acrobat Reader. http://www.adobe.com/es/products/acrobat/readstep2.html.

[3] Mark Altaweel, Nick T. Collier, Tom Howe, Robert Najlis, Michael J. North, Miles Parker, Eric Tatara, and Jerry R. Vos. Recursive porus agent simulation toolkit. http://repast.sourceforge.net/.

[4] Association for Computing Machinery (ACM). ACM digital library. http://portal.acm.org/dl.cfm.

[5] David Abrahams Beman Dawes and Rene Rivera. Boost — free peer-reviewed portable C++ source libraries. http://www.boost.org/.

[6] Check Point Software Technologies Ltd. Zonealarm. http://www.zonelabs.com/store/content/company/products/znalm/freeDownload.jsp.

[7] Core FTP. Core FTP — free, secure FTP client for windows. http://www.coreftp.com/.

[8] John W. Eaton and many others. GNU octave — a high-level language for numerical computations. http://www.gnu.org/software/octave/.

[9] Bruce Eckel. *Thinking in C++*, volume 1: Introduction to Standard C++. Prentice Hall, 2 edition, April 2000. http://www.mindview.net/Books/DownloadSites.

[10] Bruce Eckel. *Thinking in Java*. Prentice Hall PTR, 4 edition, February 2006. http://www.mindview.net/Books/DownloadSites.

[11] Bruce Eckel and Chuck Allison. *Thinking in C++*, volume 2: Practical Programming. Prentice Hall, November 2003. http://www.mindview.net/Books/DownloadSites.

[12] Elsevier B.V. ScienceDirect. http://www.sciencedirect.com/.

[13] Free Software Foundation. GNU make manual. http://www.gnu.org/software/make/manual/make.html.

[14] GAMS Development Corporation. GAMS — general algebraic modeling system. http://www.gams.com/docs/intro.htm.

[15] Google. Gmail. http://mail.google.com/mail/.

[16] Google. Google Talk — Chatea y envía mensajes instantáneos a tus amigos de forma gratuita. http://www.google.com/talk/intl/es/.

[17] Grisoft. AVG anti-virus free edition. http://free.grisoft.com.

[18] ILOG. ILOG CPLEX. http://www.ilog.com/products/cplex/, http://www.ampl.com/DOWNLOADS/cplex101.html.

[19] Institute of Electrical and Electronics Engineers, Inc. (IEEE). IEEE xplore. http://ieeexplore.ieee.org/Xplore/dynhome.jsp.

[20] Ipswitch, Inc. WS_FTP — The world's most popular FTP client. http://www.ipswitch.com/Products/WS_FTP/, http://www.uned.es/csi/sai/software/wsftp/wsftppro_spanish.exe (en castellano).

[21] AMPL Optimization LLC. AMPL — a modeling language for mathematical programming. http://www.ampl.com/DOWNLOADS/index.html.

[22] Brian Masney. gFTP — a free multithreaded file transfer client for *NIX based machines. http://gftp.seul.org/.

[23] Mozilla Foundation. Mozilla firefox. http://www.firefox2.com/es/.

[24] Michael J. North, Nick T. Collier, and Jerry R. Vos. Experiences creating three implementations of the repast agent modeling toolkit. *ACM Transactions on Modeling and Computer Simulation*, 16(1):1–25, January 2006.

[25] OpenOffice.org. Openoffice.org — a multiplatform and multilingual office suite and an open-source project. http://www.openoffice.org/ (en inglés), http://es.openoffice.org/ (en español).

[26] Steve Oualline. *C Elements of Style*. M&T Books, 1992. http://www.oualline.com/style/.

[27] pdfforge.org. PDF Creator — a free tool to create PDF files from nearly any Windows application. http://www.pdfforge.org/products/pdfcreator/, http://sourceforge.net/projects/pdfcreator/.

[28] Martin Prikryl. WinSCP — Free SFTP and SCP client for Windows. http://winscp.net.

[29] Satu Elisa Schaeffer, Jonathan C. Clemens, and Patrick Hamilton. Decision making in distributed sensor networks. In *Proceedings of the Santa Fe Institute Complex Systems Summer School*, Santa Fe, NM, USA, 2004. Santa Fe Institute.

[30] SmartSoft Ltd. SmartFTP — an FTP client which allows you to transfer files between your local computer and a server on the internet. http://www.smartftp.com/.

[31] Springer GmbH. SpringerLink. http://springerlink.metapress.com/home/main.mpx.

[32] Simon Tatham, Owen Dunn, Ben Harris, and Jacob Nevins. PuTTY — a client program for the SSH, Telnet and Rlogin network protocols. http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html.