



CENTRO DE ESTUDIOS FINANCIEROS

VIRIATO, 52	28010 MADRID	914 44 49 20
PONZANO, 15	28010 MADRID	914 44 49 20
G. DE GRÀCIA, 171	08012 BARCELONA	934 15 09 88
ALBORAYA, 23	46010 VALENCIA	963 61 41 99

www.cef.es

info@cef.es

Índice Tema 7

1. Conceptos básicos.
2. Estructura de una librería.
3. Gestión de librerías: el programa «AR» y controles de cambios.
4. Ejemplo.
5. Gestión de medios magnéticos.
6. Control de cambios.





CENTRO DE ESTUDIOS FINANCIEROS

VIRIATO, 52	28010 MADRID	914 44 49 20
PONZANO, 15	28010 MADRID	914 44 49 20
G. DE GRÀCIA, 171	08012 BARCELONA	934 15 09 88
ALBORAYA, 23	46010 VALENCIA	963 61 41 99

www.cef.es

info@cef.es

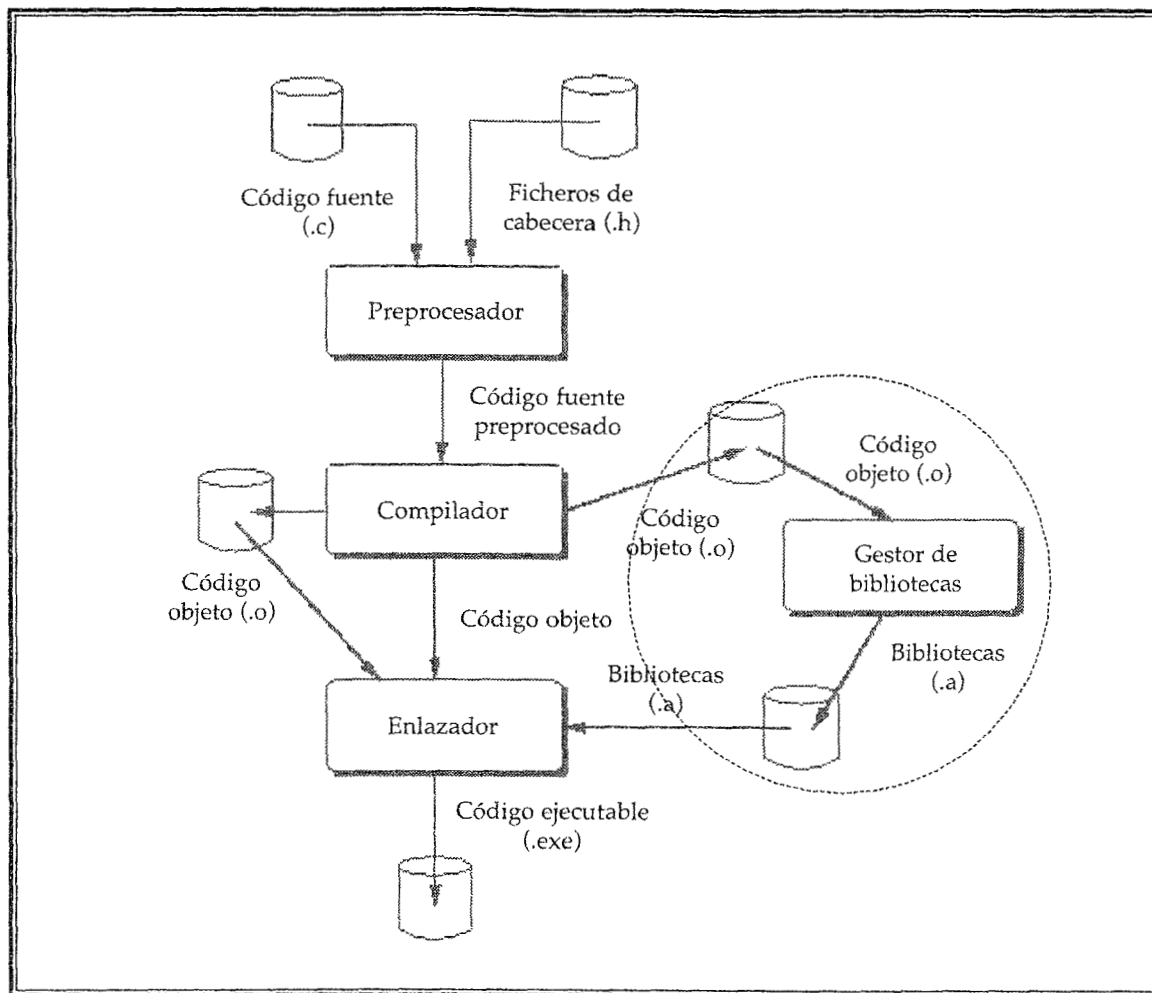
TEMA 7

Gestión de librerías de programas. Gestión de medios magnéticos. Controles de cambios.

1. CONCEPTOS BÁSICOS.

- Definición: una librería, también denominada biblioteca, es un fichero que está compuesto por una colección de otros ficheros llamados miembros de la librería.
- La creación de librerías (o bibliotecas) aporta modularidad y portabilidad a los programas.
- La estructura de una librería posibilita la extracción de sus miembros.
- Cuando se añade un fichero a una librería, los datos de éste y su información administrativa (permisos, fechas, propietario, grupo, etc.) se introducen en él.
- Existen múltiples herramientas para la gestión de librerías. Se tomará como referencia el modelo propuesto por GNU C (compilador gcc).

El modelo de compilación propuesto por este compilador responde al siguiente esquema:

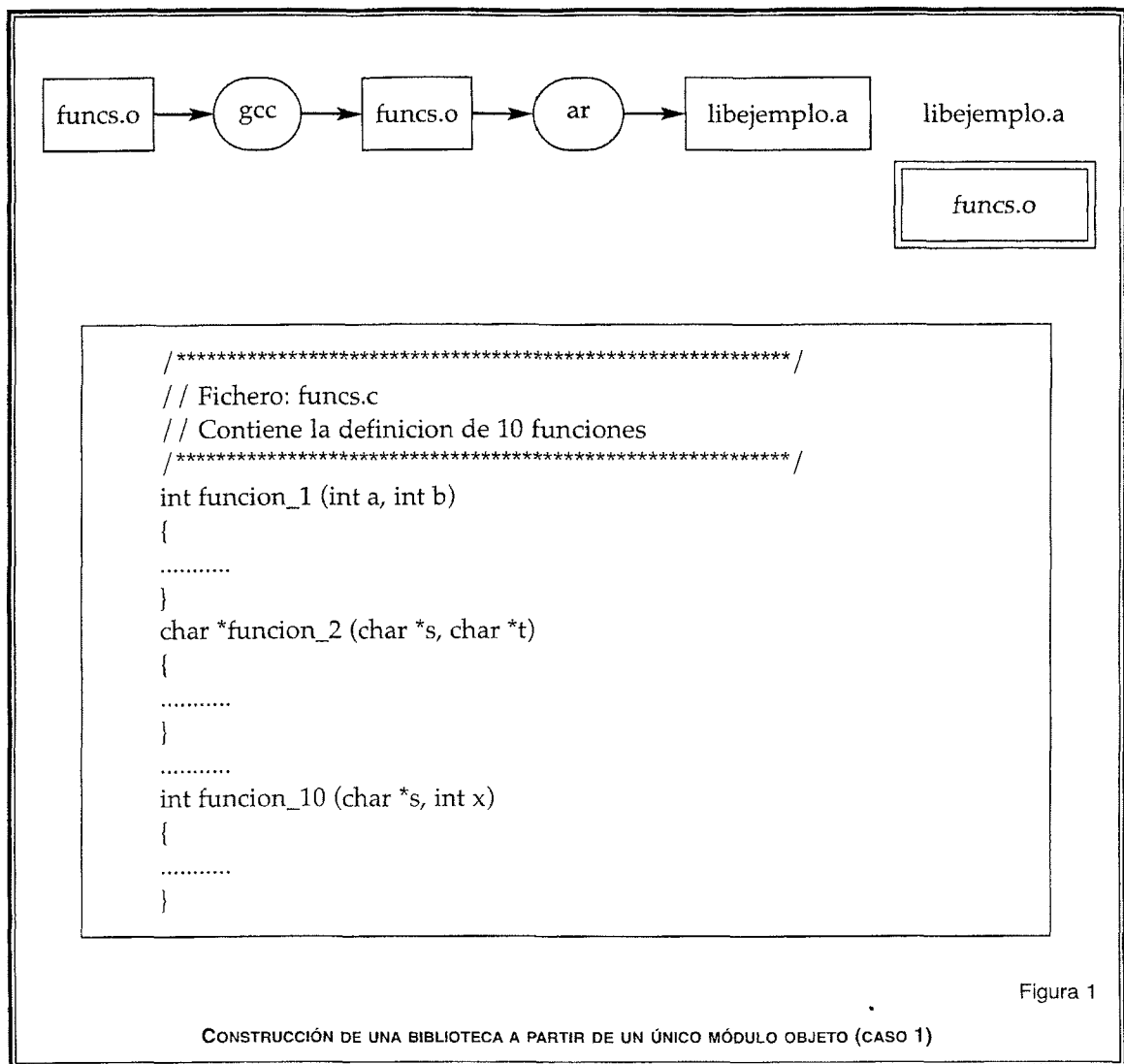


2. ESTRUCTURA DE UNA LIBRERÍA.

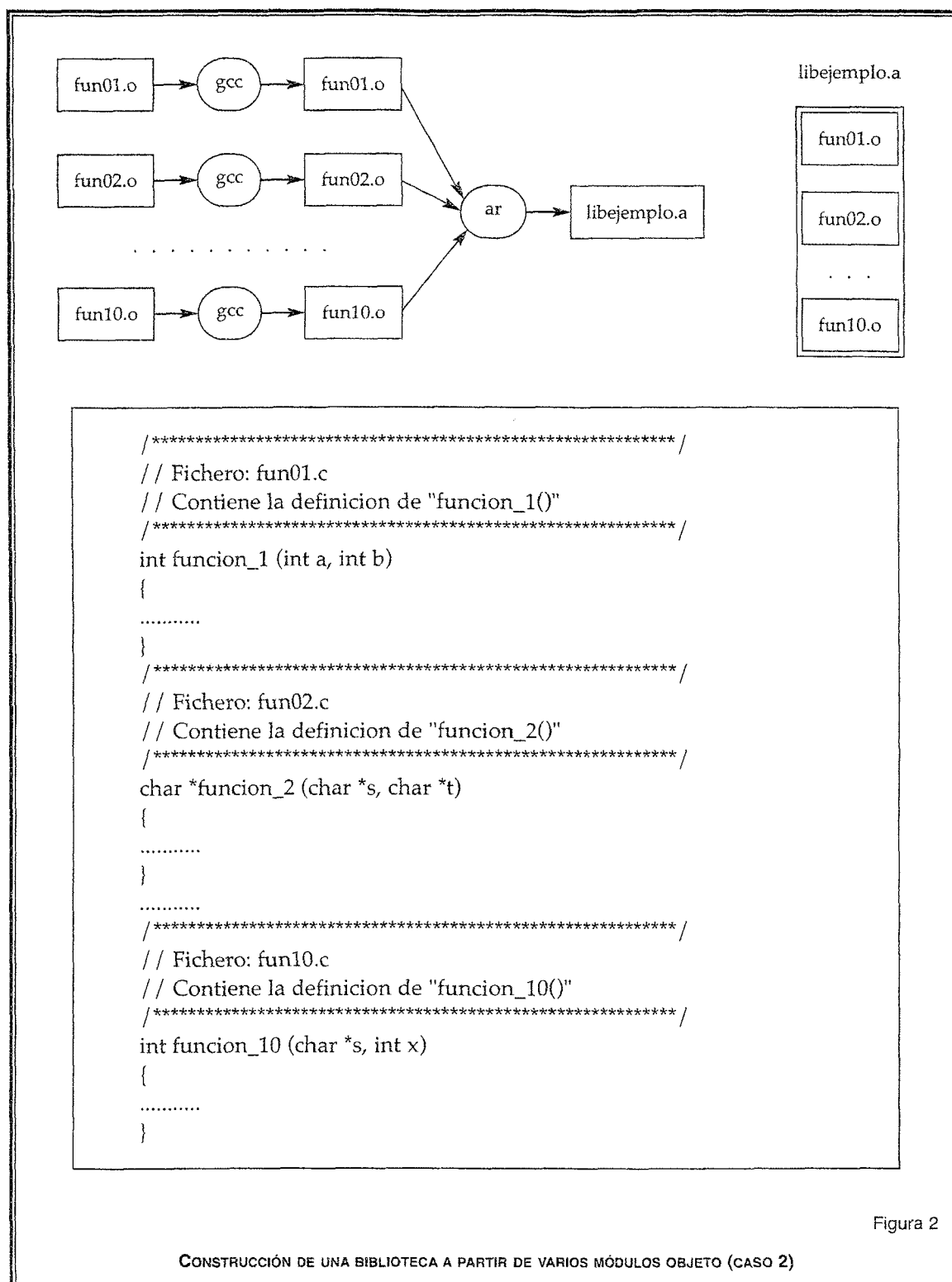
Una biblioteca o librería se estructura internamente como un conjunto de módulos objeto (extensión .o). Cada uno de estos módulos puede ser el resultado de la compilación de un fichero de código fuente (extensión .c) que puede contener, a su vez, una o varias funciones. La extensión por defecto de los ficheros de biblioteca es .a y se acostumbra a que su nombre empiece por el prefijo lib.

Veamos, con un ejemplo, cómo se estructura una biblioteca. La biblioteca libejemplo.a contiene un conjunto de 10 funciones. Los casos extremos en la construcción de esta biblioteca serían:

1. Está formada por un único fichero objeto, por ejemplo, func.o que es el resultado de la compilación de func.c. Este caso se ilustra gráficamente en la figura 1.



2. Está formada por 10 ficheros objeto, por ejemplo, `fun01.o`, `fun02.o`, ..., `fun10.o` resultado de la compilación de 10 ficheros fuente, por ejemplo, `fun01.c`, `fun02.c`, ..., `fun10.c` que contienen, cada uno, la definición de una única función. Este caso se ilustra gráficamente en la figura 2.



Una vez construida la biblioteca, y para generar el fichero ejecutable que utiliza una función de ésta, el enlazador actúa de la siguiente manera: enlaza el módulo objeto que contiene la función main() con el módulo objeto (completo) de la biblioteca donde se encuentra la función utilizada. De esta forma, en el programa ejecutable sólo se incluirán los módulos objeto que contienen alguna función llamada por el programa.

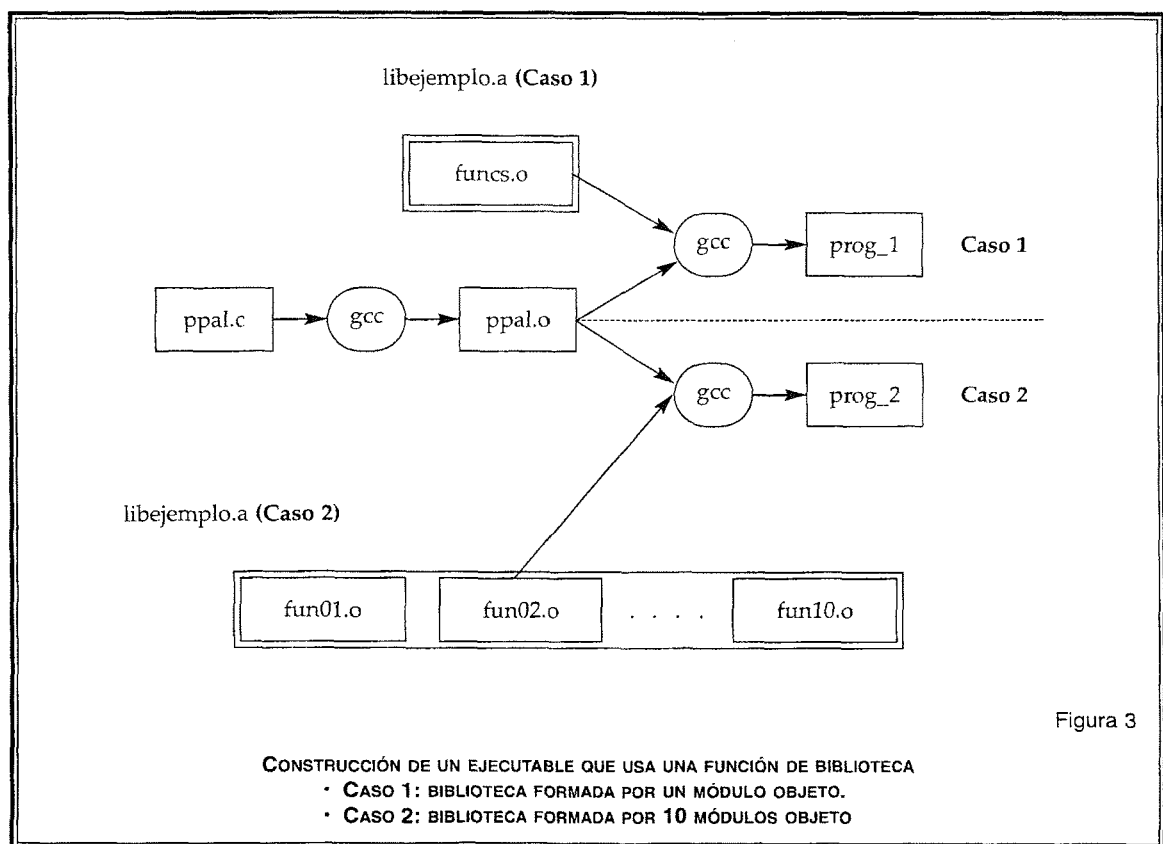
Por ejemplo, supongamos que ppal.c contiene la función main(). Esta función usa la función funcion_2() de la biblioteca libejemplo.a. Independientemente de la estructura de la biblioteca, ppal.c se escribirá de la siguiente forma:

```
#include "ejemplo.h" // prototipos de las funciones de "libejemplo.a"

int main (void)
{
    .....
    cad1 = funcion_2 (cad2, cad3);
    .....
}
```

Cada biblioteca llevará asociado un fichero de cabecera que contendrá los prototipos de las funciones que se ofrecen (funciones públicas) que actúa de interface entre las funciones de la biblioteca y los programas que la usan. En nuestro ejemplo, independientemente de la estructura interna de la biblioteca, ésta ofrece 10 funciones cuyos prototipos se encuentran declarados en ejemplo.h.

Para la elección de la estructura óptima de la biblioteca hay que considerar que la parte de la biblioteca que se enlaza al código objeto del programa principal es el módulo objeto completo en el que se encuentra la función de biblioteca usada. En la figura 3 mostramos cómo se construye un ejecutable a partir de un módulo objeto (ppal.o) que contiene la función main() y una biblioteca (libejemplo.a).



Sobre esta figura, distinguimos dos situaciones diferentes dependiendo de cómo se construye la biblioteca. En ambos casos el fichero objeto que se enlazará con la biblioteca (con más precisión, con el módulo objeto adecuado de la biblioteca) se construye de la misma forma: el programa que usa una función de biblioteca no conoce (ni le importa) cómo se ha construido la biblioteca.

- Caso 1: el módulo objeto `ppal.o` se enlaza con el módulo objeto `funcs.o` para generar el ejecutable `prog_1`. El módulo `funcs.o` contiene el código objeto de todas las funciones, por lo que se enlaza mucho código que no se usa.
- Caso 2: el módulo objeto `ppal.o` se enlaza con el módulo objeto `fun02.o` para generar el ejecutable `prog_2`. El módulo `fun02.o` contiene únicamente el código objeto de la función que se usa, por lo que se enlaza el código estrictamente necesario.

En cualquier caso, como usuario de la biblioteca se actúa de la misma manera en ambas situaciones. La secuencia de tareas para generar un ejecutable (suponiendo que la biblioteca ya existe) será:

1. Generación del objeto `ppal.o`: `gcc -c ppal.c -o ppal.o`
2. Generación del ejecutable `prog_x`: `gcc ppal.o libejemplo.a -o prog_x`

Como diseñador de la biblioteca sí debemos tener en cuenta la estructura que vamos a adoptar para ésta, considerando que si las bibliotecas están formadas por módulos objeto con muchas funciones cada una, los tamaños de los ejecutables serán muy grandes. En cambio, si las bibliotecas están formadas por módulos objeto con pocas funciones cada una, los tamaños de los ejecutables serán más pequeños.

3. GESTIÓN DE LIBRERÍAS: EL PROGRAMA «AR» Y CONTROLES DE CAMBIOS.

El programa gestor de bibliotecas de GNU es `ar`. Con este programa es posible crear y modificar bibliotecas existentes: añadir nuevos módulos objeto, eliminar módulos objeto o reemplazarlos por otros más recientes. La sintaxis de la llamada a `ar` es:

`ar [-]operación [modificadores] biblioteca [módulos objeto]`

donde:

- Biblioteca, es el nombre de la biblioteca a crear o modificar.
- Módulos objeto es la lista de ficheros objeto que se van a añadir, eliminar, actualizar, etc., en la biblioteca.
- Operación, indica la tarea que se desea realizar sobre la biblioteca. Éstas pueden ser:
 - r Adición o reemplazo. Reemplaza el módulo objeto de la biblioteca por la nueva versión. Si el módulo no se encuentra en la biblioteca, se añade a la misma. Si se emplea, además, el modificador `v`, `ar` imprimirá una línea por cada módulo añadido o reemplazado, especificando el nombre del módulo y las letras `a` ó `r`, respectivamente.

- d Borrado. Elimina un módulo de la biblioteca.
 - x Extracción. Crea el fichero objeto cuyo nombre se especifica y copia su contenido de la biblioteca. La biblioteca queda inalterada.
 - t Listado. Proporciona una lista especificando los módulos que componen la biblioteca.
- Modificadores, Se pueden añadir a las operaciones, modificando de alguna manera el comportamiento por defecto de la operación. Los más importantes (y casi siempre se utilizan) son:
 - s Indexación. Actualiza o crea (si no existía previamente) el índice de los módulos que componen la biblioteca. Es necesario que la biblioteca tenga un índice para que el enlazador sepa cómo enlazarla. Este modificador puede emplearse acompañando a una operación o por sí solo.
 - v Verbose. Muestra información sobre la operación realizada.

4. EJEMPLO.

Este ejemplo muestra cómo crear una biblioteca (libadic.a) que consta de un único módulo objeto (adicion.o). Éste se construye a partir de adicion.c que contiene únicamente dos funciones.

```

/*****/

// Fichero: adicion.c

// Contiene las funciones "suma()" y "resta()"

/*****/

#include "adicion.h"

int suma (int a, int b) {return (a+b);}

int resta (int a, int b) {return (a-b);}

Las funciones de esta biblioteca se usan en el programa demo1.c

/*****/

// Fichero: demo1.c

// Usa las funciones "suma()" y "resta()" de "libadic.a"

/*****/

#include <stdio.h>

#include "adicion.h"

int main (void) {
printf ("%d\n", suma (2,5));
printf ("%d\n", resta(2,5));
return (0);
}

```

La biblioteca lleva asociado su fichero cabecera correspondiente, en este caso, `adicion.h` que contiene los prototipos de las funciones públicas de la biblioteca (en este caso, todas).

```
/* **** */
// Fichero: adicion.h

// Fichero de cabecera asociado a la biblioteca "libadic.a"

/* **** */

#ifndef ADICION
#define ADICION

int suma (int, int);
int resta (int, int);
#endif
```

Los comandos a ejecutar serían los siguientes:

```
gcc -c -o demo1.o demo1.c
gcc -c -o adicion.o adicion.c
ar rvs ./libadic.a adicion.o
a - adicion.o

gcc -L. -o demo1.exe demo1.o -ladic
```

• Otro ejemplo.

Este ejemplo muestra cómo crear dos bibliotecas: `libadic.a` y `libprod.a`. Ambas constan de un único módulo objeto (`adicion.o` y `producto.o`, respectivamente). El primer módulo objeto se construye a partir del fuente `adicion.c` (véase ejemplo anterior) y el segundo módulo objeto se construye a partir del fuente `producto.c`. Ambos ficheros contienen dos funciones.

```
/* **** */
// Fichero: producto.c

// Contiene las funciones "multiplica()" y "divide()"

/* **** */

#include "producto.h"

int multiplica (int a, int b) {return (a*b);}

int divide (int a, int b) {return (a/b);}
```

Las dos bibliotecas se usan en el programa `demo2.c`. Este programa usa las dos funciones de `libadic.a` y una de `libprod.a`.

```
/* **** */
```

```
// Fichero: demo2.c

// Usa las funciones "suma()" y "resta()" de "libadic.a"

// y la funcion "multiplica()" de "libprod.a"

/*****/

#include <stdio.h>

#include "adicion.h"

#include "producto.h"

int main (void)

{

printf ("%d\n", suma (2,5));

printf ("%d\n", resta(2,5));

printf ("%d\n", multiplica(2,5));

return (0);

}
```

De nuevo, cada biblioteca lleva asociado su fichero cabecera correspondiente, en este caso, `adicion.h` (véase ejemplo anterior) y `producto.h`.

```
/*****/

// Fichero: producto.h

// Fichero de cabecera asociado a la biblioteca "libprod.a"

/*****/

#ifndef PRODUCTO

#define PRODUCTO

int multiplica (int, int);

int divide (int, int);

#endif
```

5. GESTIÓN DE MEDIOS MAGNÉTICOS.

- La gestión de los medios magnéticos afecta a dos aspectos: estrategias de almacenamiento y salvaguarda de la información (backups).
- La gestión integral y centralizada de los datos en entornos heterogéneos es la situación más crítica para una organización.
- Razones para evolucionar hacia una gestión eficiente de los medios magnéticos:

- El 94 por 100 de la información de Europa todavía está en papel, y para la conversión al entorno e-business el almacenamiento electrónico de datos es un requisito previo clave.
- Las necesidades de almacenamiento crecen año a año a un ritmo de un 60 por 100.
- Los costes de gestión física del almacenamiento representan un 80 por 100 del gasto total de almacenamiento, relegando a un segundo término el precio del hardware.

• **Software para la gestión del almacenamiento y backups.**

a) Soluciones comerciales:

i) Tivoli Storage Management. Permite gestionar de un modo centralizado la información corporativa ubicada en cualquier sitio desde servidores NT a mainframes y desde SAN a LAN e Internet. Además permite realizar desde los clientes copias de seguridad integradas en toda la empresa, gestión de almacenamiento y recuperación en caso de desastre en más de 39 plataformas distintas, incluidas NT, UNIX, Novell, Linux, OS/400 y OS/390.

ii) VERITAS NetBackup:

- (1) Permite gestionar la información de organizaciones que almacenan información crítica en sistemas heterogéneos y la distribuyen alrededor de todo el mundo.
- (2) Soporta virtualmente todas las plataformas informáticas más conocidas.
- (3) Gestiona la recuperación de datos en caso de desastres.
- (4) Dispone de opciones de backup exclusivas, como Flashbackup que realiza backups consistentes «snap shot» de datos mientras está en uso el sistema.
- (5) Proporciona backup continuo para las aplicaciones más utilizadas en Windows NT, incluidas Microsoft SQL Server, Microsoft Exchange Server y Oracle.
- (6) Dispone de los mecanismos de reconocimiento automático de los SGBDRs más importantes del mercado.

iii) VERITAS TeleBackup: realización de backups en PCs/estaciones de trabajo y equipos móviles y remotos que ofrece protección y simplicidad para los usuarios de sistemas personales, así como el alto rendimiento y escalabilidad que requieren los entornos de TI corporativos.

b) Servidores Windows. Gestión del sistema de almacenamiento.

(a) DFS (Sistema de archivos distribuido). Permite a los usuarios acceder a una estructura de carpeta virtual que verán como una única estructura continua de carpetas, la cual está formada en realidad por carpetas que residen en distintos servidores esparcidos a través de la organización. Dos posibilidades: DFS independiente (hospedada en un servidor que no pertenece a un dominio de Windows) y DFS basadas en dominios.

- (b) Almacenamiento remoto-Administración de almacenamiento jerárquico. Permite dos capas de almacenamiento de datos. La primera capa, el almacenamiento local, es el disco duro estándar o la unidad de disco extraíble albergada en un volumen NTFS (como una unidad Jazz de Iomega). La segunda capa es la unidad de cinta y la biblioteca de cintas automatizada. Los datos están almacenados inicialmente en el almacén local, y una vez que los datos han caducado (no se ha accedido a ellas dentro del período de tiempo especificado), se copian o migran al medio de almacenamiento remoto, aunque también permanecen intactos o en la caché del almacén local para un rápido acceso. Almacenamiento remoto sólo administra físicamente los archivos de un volumen, no sigue enlaces DFS.

- (c) Carpetas compartidas. NETBios, permisos ...

c) Servidores Unix/Linux. Gestión del sistema de almacenamiento.

- (a) rsync es un programa para copiar archivos entre dos sistemas UNIX que utiliza un algoritmo propio. Para los archivos que ya existan en ambas máquinas es capaz de enviar de un equipo a otro tan sólo aquellas partes de los archivos que hayan sido modificadas, «sincronizando» de esta forma los contenidos de los dos equipos.
- (b) NFS (Network File System de SUN). Permite compartir datos entre varios ordenadores de una forma sencilla sin necesidad de validación entre máquinas de almacenamiento. Sigue el modelo cliente/Servidor. NFS no es un protocolo demasiado eficiente y puede ser lento. Está diseñado para redes locales.
- (c) Samba. Es un sistema de ficheros compartido similar a NFS, con la particularidad de que «habla el idioma» de Windows. Mediante Samba podremos acceder desde Linux a los «recursos compartidos» de Windows, y viceversa: instalar en un Linux un servidor de ficheros al que se pueda acceder desde el «entorno de red» de Windows. Se basa en el protocolo SMB que es usado por Microsoft Windows 3.11, 9x/ME y NT/2000 para compartir discos e impresoras.

6. CONTROL DE CAMBIOS.

a) Servidores WINDOWS: gestión de los backups:

(a) Tipos de backups:

- (i) Copia de seguridad normal. Una copia de seguridad normal, en el lenguaje de Windows, es una copia de seguridad total de todos los archivos y directorios seleccionados en el software de Copia de seguridad de Windows. Como parte de la tarea, el programa borra el bit de modificado de cada archivo. Este tipo de copia de seguridad es la base para futuras tareas que sólo realizan copias de seguridad de los archivos modificados.
- (ii) Copia de seguridad incremental. Durante una copia de seguridad incremental, el programa examina el bit de modificado y hace una copia de seguridad sólo de los archivos que han cambiado desde la última copia de seguridad incremental o normal. Al igual que con la copia de seguridad normal, esta tarea borra el bit de modificado de cada archivo que copia. Las copias de seguridad incremental utilizan la mínima cantidad de cinta y además ahorran tiempo puesto que no copian todos los archivos que

no han cambiado durante la tarea. Sin embargo, realizar una restauración es un inconveniente.

- (iii) Copia de seguridad diferencial. Una copia de seguridad diferencial es lo mismo que una copia de seguridad incremental exceptuando que el programa no elimina el bit de modificación de los archivos que copia a la cinta. Este tipo de tarea requiere más espacio en cinta que el utilizado en las tareas incrementales así como más tiempo, pero su ventaja radica en que cuando se realiza una restauración, se necesitan sólo las cintas que contengan la copia de seguridad normal y la más reciente copia de seguridad diferencial.
- (iv) Copia de seguridad diaria. Una copia de seguridad diaria copia sólo los archivos que han sido modificados en el día en que se ejecuta la tarea de copia de seguridad sin tener en cuenta el estado actual del bit de modificación. Este tipo de tarea tampoco borra el bit de modificación. Útiles cuando se quiere realizar una copia de seguridad extra en un día determinado, sin afectar a la estrategia de copia de seguridad establecida.
- (v) Copia de seguridad intermedia. Equivalente a una copia de seguridad normal, excepto que el programa no desactiva el atributo de modificado.

b) Servidores UNIX. Gestión de los backups:

(a) Tipos de backups = nivel de copia de seguridad:

- (i) Copia de seguridad completa. Realiza backups volcando en el dispositivo de copia los archivos o directorios deseados, sin tener en cuenta backups previos. Se correspondería con la orden: backup -0 ..
- (ii) Copia de seguridad incremental. Copia solamente los archivos que han cambiado desde la realización de otra copia total. Se correspondería con la orden: backup-1 ..
- (iii) Copia de seguridad progresiva. Copia solamente los archivos que han cambiado desde la realización de otra copia incremental. Se correspondería con la orden: backup-2 ..
- (iv) Copia de seguridad diferencial. Copia de seguridad sobre la última progresiva. Se correspondería con la orden: backup-n .. ($n > 2$).

b) Órdenes Unix para backups: backup, restore, cpio (para salvar ficheros individuales), tar (Tape Archiver) salva ficheros, dd (Device to Device) convierte y copia ficheros, dump o rdump (para hacer copias remotas o locales de los ficheros a salvaguardar).

c) Estrategia para la copia de seguridad:

La realización de una copia de seguridad de una red de manera efectiva es una tarea compleja que se debe planear mediante una aproximación. Para realizar una copia de seguridad de una red no basta con poner simplemente una cinta en la unidad y ejecutar el software. Una estrategia para la copia de seguridad debe tener en cuenta las siguientes preguntas:

- 1) ¿Cuánta información se debe copiar?
- 2) ¿De cuánto tiempo se dispone para realizar la copia de seguridad?

- 3) ¿Cada cuánto tiempo se debe realizar una copia de seguridad de la información?
- 4) ¿Quién se encarga de verificar el acabado de las copias de seguridad?
- 5) ¿Cuántas cintas (u otros medios) se planea utilizar?
- 6) ¿Cada cuánto tiempo se rescribirán las cintas?

d) Rotación de medios:

Un esquema de rotación de medios dicta cuántas cintas (u otro tipo de medios) se usan para realizar las copias de seguridad. Un esquema de rotación popular se conoce como el método del abuelo-padre-hijo ya que utiliza tres «generaciones» de cintas que representan respectivamente copias de seguridad mensuales, semanales y diarias. En este esquema de rotación se realiza una copia de seguridad completa cada mes y se guarda la cinta durante un año (preferentemente en algún sitio seguro ajeno a la empresa); ésta es el «abuelo». Además se realiza una copia de seguridad completa semanalmente que se guarda durante un mes; ésta es el «padre». Las copias de seguridad que representan el «hijo» se realizan diariamente y se guardan durante una semana ($12 + 4 + 7 = 23$ cintas). Las tareas diarias pueden ser copias de seguridad completas, incrementales o diferenciales.

e) Los Diez Mandamientos de los backups:

1. Haga copias de seguridad de todos los datos importantes.
2. Haga una copia de seguridad de los discos de instalación de los programas.
3. Actualice las copias de seguridad tan a menudo como pueda.
4. Revise el estado de sus copias de seguridad de vez en cuando.
5. Si le da pereza copiar todo el disco, al menos copie sus archivos de datos.
6. Si le da pereza copiar todos sus archivos de datos, al menos copie los más recientes o importantes.
7. No confíe en los disquetes como dispositivo de backup, su fiabilidad es ínfima.
8. Si no dispone de otra cosa, al menos haga copias en disquete.
9. Sobre todo si utiliza disquetes o cintas magnéticas, tenga más de un juego de copias, intercámbielos de forma rotatoria y renuévelos de vez en cuando.
10. Guarde las copias en lugar seguro, si no serán copias de seguridad inseguras. Nunca en la misma área o sala de los servidores.



