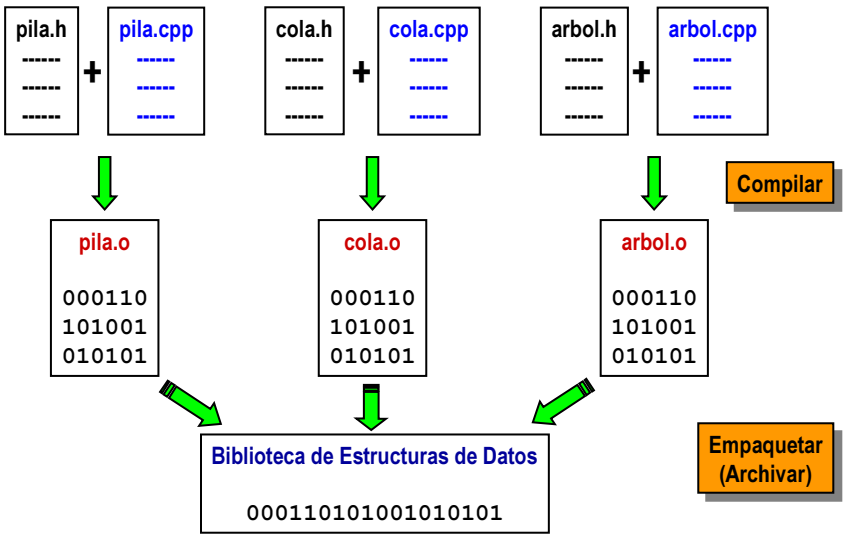
1. HERRAMIENTAS DE AYUDA A LA CREACIÓN DE PROGRAMAS:

1.1. Compilación separada (make)
1.2. Entornos de desarrollo
1.3. Bibliotecas de programas: estáticas y dinámicas
1.4. Herramientas de documentación
1.5. Análisis de rendimiento de programas (profiling)
1.6. Control de versiones (CVS)

Bibliotecas (*libraries*)

- Biblioteca: colección de elementos (clases, funciones) relacionados entre sí.
- Las bibliotecas proporcionan servicios a las aplicaciones.
- Cualquier usuario puede crear sus propias bibliotecas para organizar mejor una aplicación y compartirlas con otras aplicaciones.

Construir una biblioteca



Bibliotecas estáticas

- **Al generar el ejecutable se enlazan todos los códigos objetos compilados y los extraídos de las bibliotecas empleadas.**
- **El ejecutable es autocontenido: incluye todo el código necesario para la ejecución.**
- **Problemas:**
  - ✓ Archivo ejecutable grande.
  - ✓ La actualización de una biblioteca implica actualizar los ejecutables que la incluyen.

Bibliotecas dinámicas (compartidas)

- **El enlace con los elementos de la biblioteca se realiza en tiempo de ejecución.**
- **Dos formas de cargar:**
  - ✓ Al inicio de la ejecución (*shared libraries*).
  - ✓ El uso durante la ejecución de elementos contenidos en la biblioteca dinámica activan la carga de los mismos (*dynamic loaded libraries*).
- **Ventajas:**
  - ✓ Disminuye el tamaño de los ejecutables.
  - ✓ Los elementos de la biblioteca no se multiplican en cada uno de los programas que los usan.
  - ✓ La actualización de la biblioteca dinámica es inmediatamente visible en las aplicaciones que la usan.

Gestión de bibliotecas

- **Objetivo: Crear y mantener (añadir/eliminar elementos) bibliotecas.**
- **Mecanismos de gestión diferente para bibliotecas estáticas y dinámicas.**

Gestión de bibliotecas estáticas

- **Programa `ar`:**
  - ✓ Genera archivos, múltiples ficheros bajo el aspecto de uno solo.
  - ✓ Mantiene un índice de contenido del archivo.

- **Ejemplo de uso: crear la biblioteca de estructuras de datos del ejemplo.**
  - ✓ Se suponen creados y comprobados los ficheros objeto: `pila.o`, `cola.o` y `arbol.o`.
  - ✓ Inclusión en la biblioteca "`libestructuras.a`":

```
ar rcs libestructuras.a pila.o cola.o arbol.o
```

Gestión de bibliotecas estáticas (2)

```
ar rcs libestructuras.a pila.o cola.o arbol.o
```

Opciones:
r = inserta reemplazando
c = crear si no existe
s = crear/actualizar índice

Nombre de la biblioteca. El sufijo "a" es típico para esta clase de bibliotecas.

Los ficheros objetos que se incorporan a la biblioteca.

```
ar rc libestructuras.a pila.o cola.o arbol.o
ranlib libestructuras.a
```

Estas dos sentencias son equivalentes a la anterior

- **Usar la biblioteca:**
  - ✓ enlazar, mediante la opción -l, a la hora de compilar.

Ejemplo (Biblioteca estática)

```
1) Crear los códigos objetos:

       g++ -c -Wall pila.cpp
       g++ -c -Wall cola.cpp
       g++ -c -Wall arbol.cpp

2) Crear la biblioteca estática:

       ar rcs libestructuras.a pila.o cola.o arbol.o

3) Compilar un programa (demo) que use la biblioteca:

       g++ demo.cpp -o demo -L. -lestructuras

4) Ejecutar el programa:

       ./demo
```

Gestión de Bibliotecas dinámicas (shared)

```
1) Crear los códigos objetos:

       g++ -fPIC -c -Wall pila.cpp
       g++ -fPIC -c -Wall cola.cpp
       g++ -fPIC -c -Wall arbol.cpp

2) Crear la biblioteca dinámica:

       g++ -shared -Wl,-soname,libestructuras.so.0 \
           -o libestructuras.so.0.1 pila.o cola.o \
           arbol.o
```

Genera "Código Independiente de la Posición". Es un requisito.

"soname" de la biblioteca, no es el real

Nombre real del archivo que contiene la biblioteca.

Equivalente:
`ld -shared -o libestructuras.so.0.1 pila.o cola.o arbol.o`

Gestión de Bibliotecas dinámicas (shared) (2)

3) Establecer el "soname" como un enlace simbólico al nombre real de la biblioteca:

```
ln -sf libestructuras.so.0.1 libestructuras.so.0
```

4) Establecer el nombre de biblioteca que buscará el linker como un "link" al "soname" de la biblioteca :

```
ln -sf libestructuras.so.0 libestructuras.so
```

`libestructuras.so → libestructuras.so.0 → libestructuras.so.0.1`

Gestión de Bibliotecas dinámicas (shared) (3)

5) Compilar un programa (demo) que use la biblioteca:

```
g++ demo.cpp -o demo -L. -lestructuras
```

6) Ejecutar el programa:

```
export LD_LIBRARY_PATH = "."
./demo
```

Ejercicio (Juny 02)

- **Supongamos que tenemos los ficheros siguientes:**

`$HOME/SRC/ej1/fuente1.cpp` (permite generar **`fuente1.o`**)
`$HOME/SRC/ej1/fuente2.cpp` (permite generar **`fuente2.o`**)
`$HOME/SRC/ej1/cab1.h`
`$HOME/SRC/ej1/cab2.h`
`$HOME/SRC/ej1/lib/libcab.h` (usado por ambas fuentes).
`$HOME/SRC/ej1/lib/libfuente.cpp` (permite generar libfuente.o)

Escribir los **`makefiles`** que generen tanto la biblioteca libfuente.o como los ejecutables **`fuente1 y fuente2`**

La biblioteca se genera a partir de **`libfuente.o`**

`Fuente1` se genera a partir de **`fuente1.o`** y la biblioteca

`Fuente2` se genera a partir de **`fuente2.o`** y la biblioteca

`Makefile` en el directorio `ej1` (Bibl. estáticas)

```
CXX= g++
CXXFLAGS= -Wall
VPATH= ./lib
objs= fuente1.o fuente2.o

fuente1: fuente1.o libfuente.a
  $(CXX) demo1.o -o fuente1 -Llib -lfuente

fuente2: fuente2.o libfuente.a
  $(CXX) fuente2.o -o fuente2 -Llib -lfuente

fuente1.o: cab1.h libcab.h
fuente2.o: cab2.h libcab.h

libfuente.a:
  make libfuente.a -C lib

.PHONY: limpiar
limpiar:
  rm $(objs)
  rm fuente1 fuente2
  make limpiar -C lib
```

Makefile en el directorio lib (Bibl. estáticas)

```
CXX= g++
CXXFLAGS= -Wall
objs= libfuente.o

libfuente.o: libfuente.cpp

libfuente.a:
 ar rcs $@ $(objs)

.PHONY: limpiar
limpiar:
 rm $(objs)
 rm libfuente.a
```

Makefile en el directorio ej1 (Bibl. dinámicas)

```
CXX= g++
CXXFLAGS= -Wall
VPATH= ./lib
objs= fuente1.o fuente2.o

fuente1: fuente1.o libfuente.a
  $(CXX) demo1.o -o fuente1 -Llib -lfuente

fuente2: fuente2.o libfuente.a
  $(CXX) fuente2.o -o fuente2 -Llib -lfuente

fuente1.o: cab1.h libcab.h
fuente2.o: cab2.h libcab.h

libfuente.so:
  make libfuente.so -C lib

.PHONY: limpiar
limpiar:
  rm $(objs)
  rm fuente1 fuente2
  make limpiar -C lib
```

`Makefile` en el directorio `lib` (Bibl. dinámicas)

```
CXX= g++
CXXFLAGS= -Wall -fPIC
objs= libfuente.o

libfuente.o: libfuente.cpp

libfuente.so:
 ld -shared -o libfuente.so

.PHONY: limpiar
limpiar:
 rm $(objs)
 rm libfuente.so
```