Bibliotecas / Librerías

Tres Tipos

- **Estaticas:** Simple colección de programas objeto. Una vez compilado el código es parte de nuestro binario. En Linux extensión .a
- **Compartidas (Shared):** El código no formará parte de nuestro programa. Se cargan en memoria junto con nuestro programa cuando se lo invoca. En Linux extensión .so
- **De carga dinámica (DL – Dynamic Loaded):** No forman parte de nuestro programa. Solo se cargan en memoria si son requeridas/invocadas (no con el programa).

Estáticas

- Compilamos solo el archivo objeto, no el binario usando -c:

```
$ gcc -c holalib.c
```

- Luego creamos la libreria usando otro programa el ar:

```
$ ar rcs libhola.a holalib.o
```

- Del comando ar (de archive) usamos 3 modificadores:
  - r: Reemplazar archivo
  - c: Crear archivo
  - S: Actualizar el indice de la biblioteca
- También se puede usar el modificador “t” para listar lo que tiene la biblioteca:

```
$ ar t libhola.a
```

Estáticas: Enlazando...

- Primero compilamos (sin enlazar) el programa que usará la libreria:

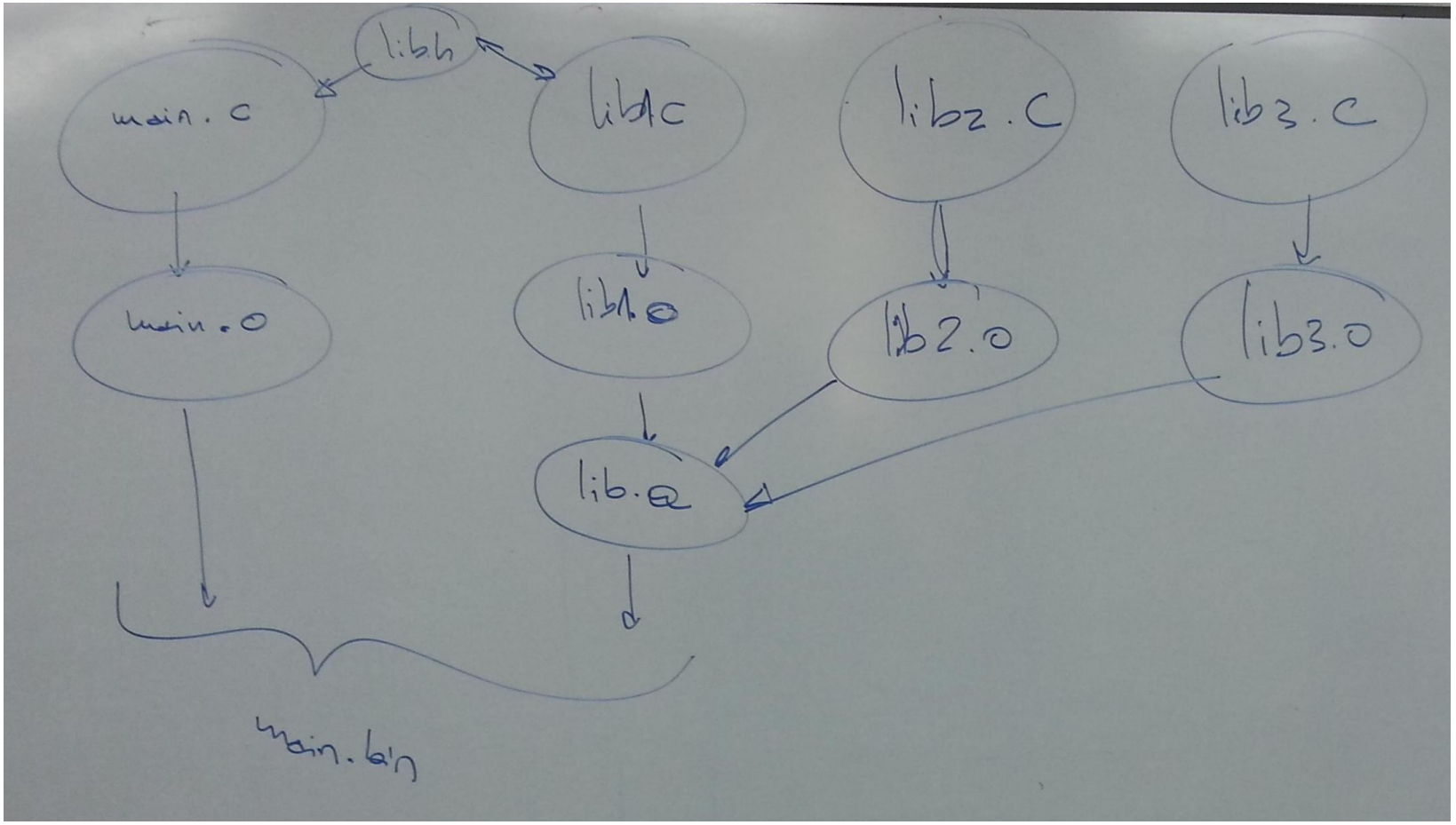
```
$ gcc -c -otest_holalib.o test_holalib.c
```

- Luego vamos a enlazar el programa binario con la libreria previamente creada:

```
$ gcc -odemo test_holalib.o -L. -lhola
```

  - El -l indica que librería usar. Como siempre comienzan con “lib” y terminan con “.a” solo ponemos lo que va en el medio: “hola”
  - El -L ha explicito al linker donde buscar la librería, en nuestro caso “.” por que esta en el dir actual.
  - Si no lo hacemos explicito toma el del sistema. Esto depende de la distribución. En debian se encuentra en /etc/ld.so.conf.d/libc.conf

Estáticas


libh
main.c
lib1.c
lib2.c
lib3.c
main.o
lib1.o
lib2.o
lib3.o
lib.a
main.bin

Bibliotecas Compartidas

- Como dijimos antes estas Bibliotecas no incorporan el codigo de sus funciones al programa que las invoca, sino que resuelven las referencias en el momento de arranque del programa que las usa. Son útiles para no repetir código si varios programas necesitan usar la misma librería.
- Las Bibliotecas compartidas tienen un nombre al que se denomina “soname” que en general difiere del nombre del archivo, que la contiene:

```
$ ls -l /usr/lib/libopencv_highgui*
$ objdump -p /usr/lib/libopencv_highgui.so.2.3.1 | grep -i soname
```

- Este nombre se asigna mediante el parametro -soname que se pasa al linker ld en el momento de su construccion. Es mejor para el manejo de estas Bibliotecas tener separados soname y nombre de archivo.

Bibliotecas Compartidas: Compilando...

- Directorios comunes de librerias:
  - `/lib , /usr/lib , /usr/local/lib`
- Para compilar:

```
$ gcc -fPIC -g -c -Wall holalib.c
```

- La opcion fPIC indica al compilador que genere codigo independiente de la posicion que ocupe en memoria.
- Finalmente en lugar de archivarlo le pedimos al linker (a través de gcc) que cree el archivo de librería compartida:

```
$ gcc -shared -Wl,-soname,libhola.so.1
       -o libhola.so.1.0.1 holalib.o -lc
```

- El modificador -Wl tiene argumentos que deberan ser enviados al linker uno a uno (cada parametro separado con una o mas comas)

Bibliotecas Compartidas: Instalando...

- La instalación de una biblioteca compartida depende de la distribución usada. Por ejemplo ambos Ubuntu y Debian usan la utilidad ldconfig, pero Ubuntu usa también $LD_LIBRARY_PATH. Lo que anda en ambas es usar el archivo de configuración /etc/ld.so.conf y la utilidad: ldconfig
- Para ser mas prolijos crearemos un directorio propio en /usr/local/lib

```
$ sudo mkdir /usr/local/lib/personal
$ sudo mv ./libhola* /usr/local/lib/personal
```

- Lo agregamos al archivo de configuración

```
$ cat /etc/ld.so.conf
```

  - include /etc/ld.so.conf.d/*.conf

```
$ sudo vim  /etc/ld.so.conf.d/personal.conf
```

  - # Personal libraries
  - /usr/local/lib/personal

- Luego ejecutamos ldconfig para que deje nuestra libreria lista para su uso:

```
$ sudo ldconfig
```

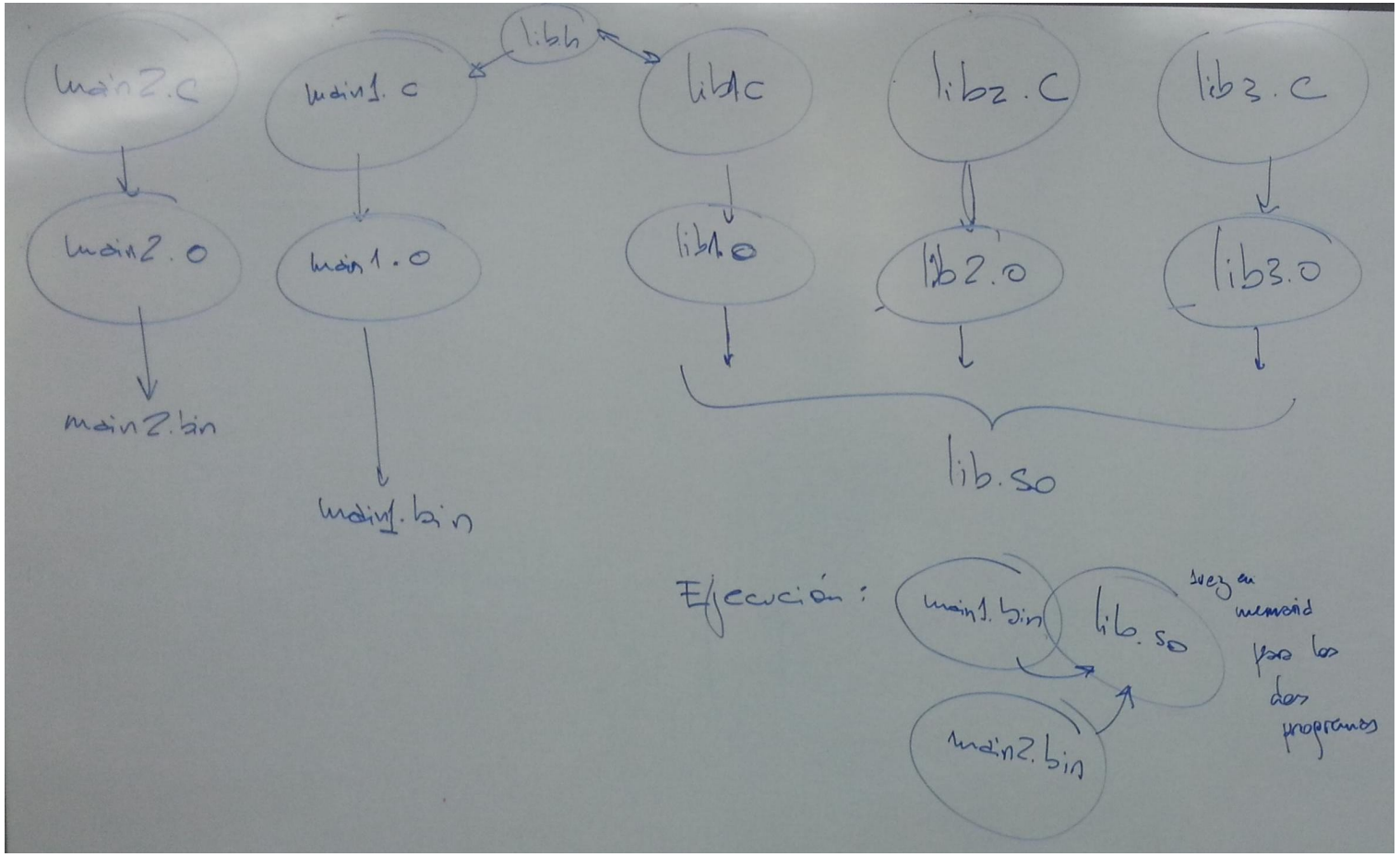
Bibliotecas Compartidas: Usando...

- Luego compilamos la aplicación que lo va a usar:

```
$ gcc -Wall -c -o test_holalib.o test_holalib.c
```

- Luego necesitamos para el compilador/linker tener también:
  - `cd /usr/local/lib/personal`
  - `sudo ln -sf libhola.so.1 libhola.so`
- Y ahora podemos crear el binario
  - `gcc -o demo test_holalib.o -L/usr/local/lib/personal/ -lhola`

Compartidas



Bibliotecas de carga dinámica

- Estas no se cargan al comenzar el programa si no que deben ser cargadas/pedidas por el programa invocante.
- Su principal función es para crear modulos o plugins que solo se cargaran si son requeridos.
- Para invocarlas/cargarlas se debe usar una librería particular, pero para compilarlas/instalarlas usamos el mismo método que para compartidas.
- La libreria que nos permitirá manejar la carga dinámica es dlfcn.h

Bibliotecas de carga dinámica: Funciones

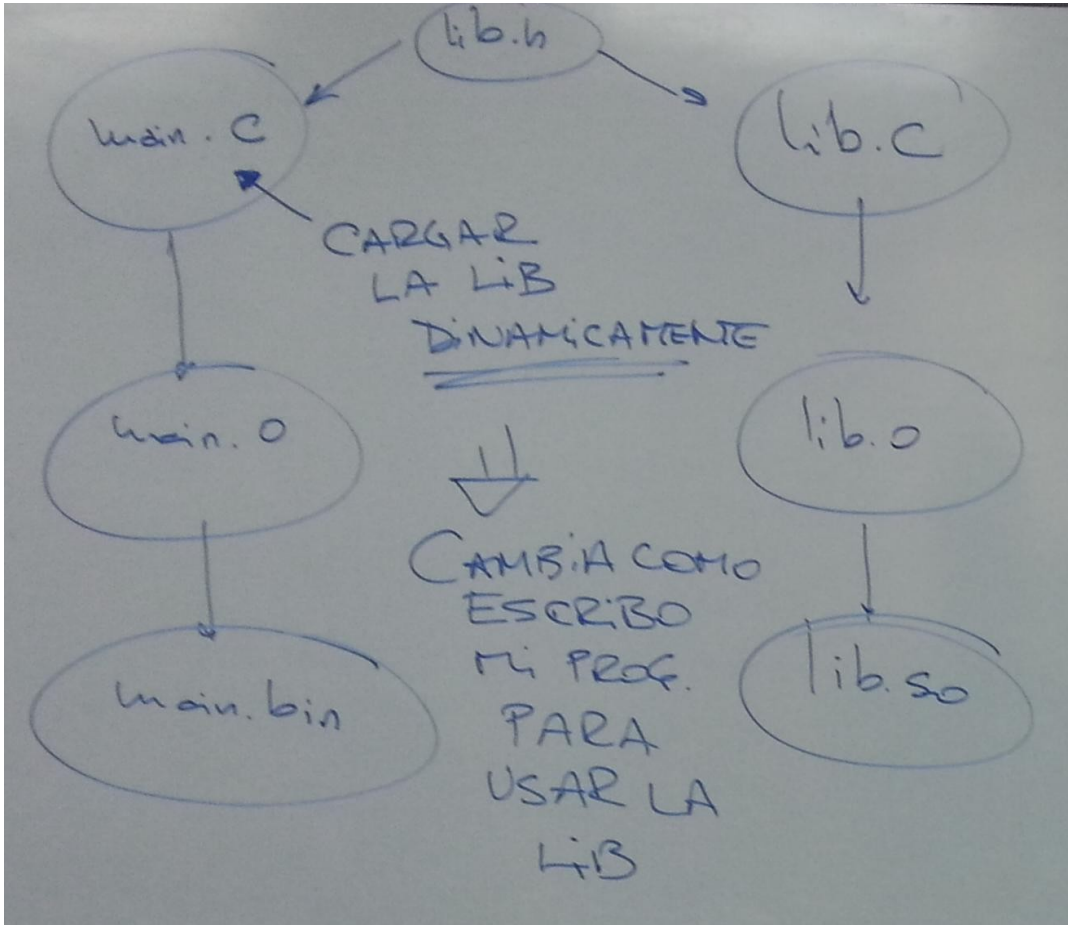
- dlopen()
  - Carga una librería y la prepara para su uso.
  - Devuelve un identificador para esa librería que se utiliza en el resto del programa para las referencias posteriores.
- dlerror()
  - Retorna un string que describe el error generado por las demás funciones de manejo de librerías dinámicas.
- dlsym()
  - Busca el valor de un símbolo presente en una librería ya abierta con dlopen().
  - Devuelve un puntero a la función o al elemento direccionado.
- dlclose()
  - Cierra la librería abierta con dlopen() permitiendo liberar el descriptor o handle que se tenía hasta el momento para identificar a nuestra librería dinámica.

Bibliotecas de carga dinámica: Compilando...

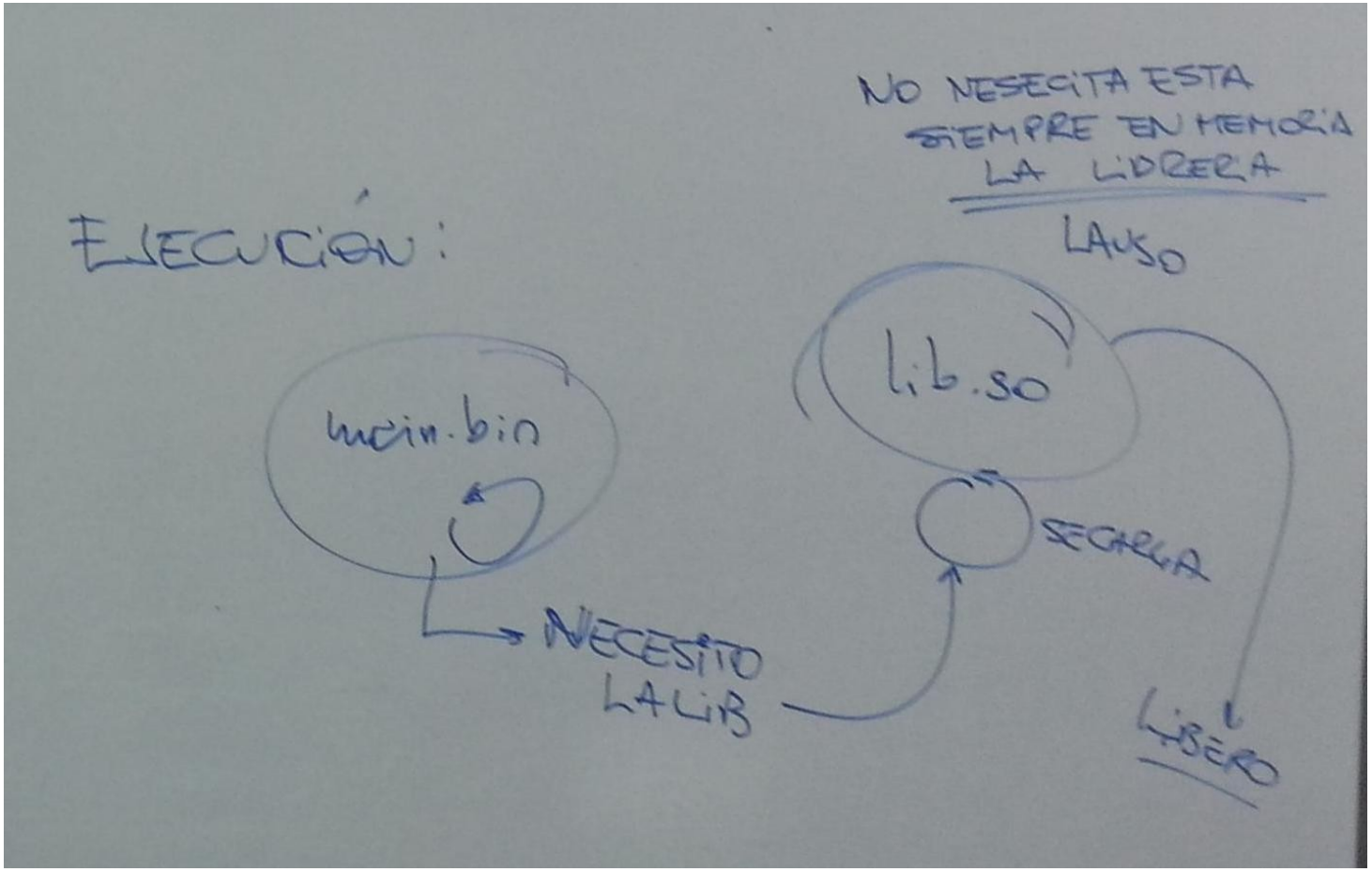
- Se busca en el archivo de cache /etc/ld.so.cache (que crea ldconfig) para ver si contiene la librería invocada/pedida por dlopen().
- El archivo que ahora llama a la librería deberá ser entonces diferente por que debe tener todo el manejo dinámico de la librería: test_holalib_dl.c

```
$ gcc -Wall -o demo test_holalib_dl.c -ldl
```

Dinámicas Compilación



Dinámicas Ejecución



Ventajas & Desventajas