# *Dynamic Linking*

El uso de enlaces dinámicos o *dynamic linking* consiste en realizar el proceso de enlace o *linking* en tiempo de ejecución. El linking, como vimos en la materia, normalmente es el proceso por el cual se exponen funciones de una determinada biblioteca para incorporarla a un binario que las usa.  Por ejemplo, un archivo `utils.c` se compila a un archivo `utils.o` y ese archivo se enlaza o "linkea" con el archivo `main.o` que utiliza las funciones definidas en utils.c .

En el caso de dynamic linking, el binario final no contiene todas las funciones sino que espera su definición en otro lugar. Esto se realiza por múltiples razones:

- Permite reutilizar una misma biblioteca global para distintos binarios
- Esta biblioteca global puede actualizarse sin necesidad de recompilar los distintos binarios
- Usar binarios enlazados dinámicamente reduce su tamaño, lo que era útil antiguamente

En Linux, la forma de observar dependencias externas es con el programa ldd:

```
ldd /usr/bin/cat
    linux-vdso.so.1 (0x00007fad053dc000)
    libc.so.6 => /usr/lib/libc.so.6 (0x00007fad05000000)
    /lib64/ld-linux-x86-64.so.2 => /usr/lib64/ld-linux-x86-64.so.2
(0x00007fad053de000)
```

En este caso el programa `cat` está enlazado dinámicamente. El primer resultado (`linux-vdso.so.1`) es común a todos los binarios dinámicos y tiene que ver con la interfaz al kernel. El último resultado se corresponde con el intérprete del formato binario ELF en Linux. El segundo resultado es el más interesante: `libc.so.6` se corresponde a la biblioteca estándar de C, que se obtiene dinámicamente del archivo `/usr/lib/libc.so.6`[1].  Todos los binarios compilados dinámicamente que usen la biblioteca estándar de C (`<stdio.h>`, `<stdlib.h>`, etc.) van a apuntar entonces a ese mismo archivo, y si ese archivo no se encuentra múltiples programas básicos van a dejar de funcionar.

Estos archivos .so (por *shared object*, equivalente a los archivos .dll en Windows) son simplemente binarios compilados especialmente como bibliotecas compartidas usando el formato *Position Independent Code* (PIC), en el cual la definición de las funciones es independiente de la posición en memoria y por lo tanto pueden ser enlazadas desde otro ejecutable sin el riesgo de que al modificar las funciones se afecte la alineación del programa que las recibe.

---

[1] La extensión 6 indica la *versión* del .so, generalmente versiones distintas son incompatibles entre sí y el archivo .so puede ser un symlink a una versión específica.

Para crear un .so, se puede usar gcc:

```
gcc -shared -fPIC -o libfoo.so libfoo.c
```

por lo que el archivo `libfoo.so` va a ser una biblioteca compartida a la que se puede linkear dinámicamente. Para hacer uso de libfoo, se le tiene que indicar a gcc que haga uso de la librería (flag -l) y su ubicación actual para que verifique la correctitud de las llamadas a función (flag -L):

```
gcc -o prog prog.c -L./libfoo/ -lfoo
```

Sin embargo, si se ejecuta el programa normalmente se encuentra el siguiente error:

```
./prog
./prog: error while loading shared libraries: libfoo.so: cannot open
shared object file: No such file or directory
```

Puede verificarse con ldd que el linker no encuentra libfoo:

```
ldd ./prog | grep libfoo
    libfoo.so => not found
```

Esto es porque, por defecto, el linker solo busca los .so en directorios especiales como /usr/lib/. Para indicarle al linker otro directorio, se puede usar la variable de entorno LD_LIBRARY_PATH:

```
export LD_LIBRARY_PATH=$(pwd)/libfoo
./prog
```

y ahora sí el linker reconoce correctamente el programa.

## Fase 5

La fase 5 de la bomba usa una biblioteca que puede ser cargada dinámicamente desde el directorio ./libphase5 . Por ejemplo, para poder correr el ejecutable sin que muestre un mensaje de error es posible hacer:

```
LD_LIBRARY_PATH=$(pwd)/libphase5 ./bomb
```

El objetivo de la fase 5 es resolverla **sin modificar el archivo libphase5/libphase5.so**. Es decir, para que se considere aprobada es necesario que el archivo libphase5/libphase5.so sea exactamente igual al provisto por la cátedra. Es válido hacer cualquier otra modificación que permita resolver la fase, mientras se deje ese archivo intacto.