



75-08 Sistemas Operativos
Lic. Ing. Osvaldo Clúa
2008

Facultad de Ingeniería
Universidad de Buenos Aires

Bibliotecas

Bibliotecas (Library)

- Colección de subprogramas usados en el desarrollo de Software.
- Contienen "código auxiliar" y datos que brindan servicios a distintos programas.
- Permiten el desarrollo modular.
- Convierten los servicios en "commodities" entre los cuales se puede elegir.

Tipos de Bibliotecas

- Estáticas:
 - Cuando los subprogramas se incluyen en el ejecutable.
- Dinámicas: pueden cargarse:
 - antes de cargar el programa
 - en el momento de cargar el programa
 - durante la corrida del programa
- Compartidas

Bibliotecas Estáticas

- El link-editor es quien las incorpora al archivo ejecutable.
 - En linux lo hace el **GNU ld**
 - Invocado al usar `--static` (ver laboratorio)
 - En windows, terceras partes hicieron un **Linker para .net**. (Para evitar instalar el **.net Framework**).
 - Facilitan la instalación de los programas en otras máquinas.
 - Evitan que los troyanos ataquen programas sensibles.

Bibliotecas Dinámicas

- El link-editor solo indica las llamadas. Éstas se resuelven:
 - Durante la carga o
 - Durante la ejecución.
- Un uso común de estas bibliotecas son los **Plugins**.
 - Extienden la funcionalidad de las aplicaciones.
 - Dieron origen a un mercado como <http://www.toolfarm.com/>

Enlace dinámico

- Se debe solucionar el problema de la **relocación**:
 - No se puede depender de posiciones “absolutas” de memoria ni en el programa ni en las bibliotecas.
 - Se deben calcular las direcciones cada vez que se carga el programa o la biblioteca.
 - Una forma de acelerar la carga es usar **prelinking** en Linux o **prebinding** (BSD o Mac OS-X).
 - En general se usa una Import Table como acceso indirecto a las direcciones de la biblioteca para simplificar el proceso de linking.

Búsqueda de las Bibliotecas dinámicas

- En Linux hay un "search path" formado por:
 - posición fija indicada en el programa+ archivo de configuración + variable de ambiente.
 - El orden de búsqueda es inverso (última modificación manda).
 - En Windows los **ActiveX** se buscan en el registry. El resto en el indicado en una llamada a **SetDllDirectory()**, System32, System y Windows.
 - El control de versiones tampoco es muy serio.

Windows DLL

- Además de dinámicas son compartidas.
- Se guardan en un PE.
- Tienen extensiones como `ocx`, `dll`, `fon`, `drv`
- Pueden contener código, datos o `recursos` (iconos, menus, bitmaps, templates, fonts, etc)
- Proveen modularidad para desarrollo y mantenimiento ...
 - A costa del `DLL hell` que terminó al aparecer `.net`.

Funcionamiento de las DLL

- En **Win32** el archivo de las DLL está organizado en secciones:
 - Cada sección tiene sus propios atributos (ejecutable, compartible, solo lectura, etc).
 - Generalmente hay una sola copia de las secciones de código en memoria (se comparte).
 - En cambio las secciones de datos son privadas (aunque pueden compartirse).
 - Las bibliotecas **comprimidas** (con **UPX** por ejemplo) permanecen privadas.

Funcionamiento de las DLL (2)

- Los símbolos exportados tienen como identificador un número y un nombre.
 - Solo los nombres se mantienen entre distintas versiones.
 - Las tablas están ordenadas por nombre.
 - Se encuentra el número por una búsqueda binaria.
 - Se accede a la función por el número
 - Se puede hacer un bind a una rutina de una versión específica.

Funcionamiento de las DLL (3)

- load-time dynamic linking
 - El linking se hace al cargar el programa.
 - El programador no tiene control si no se encuentra la DLL
- run time dynamic linking
 - Se hace por medio de un llamado a `LoadLibrary()`
 - En programador puede intervenir si hay error.
- para saber que bibliotecas se usan, hay programas como `Dependency Walker`

Bibliotecas en Linux

- Las bibliotecas comienzan con `lib`
- Las de enlace estático terminan en `.a` y se manejan con `ar (1)`
- Las de enlace dinámico en `.so`
- Además se agrega un número de versión.
 - Por ejemplo la biblioteca `foo` es `libfoo.a` y `dynfoo` es `libdynfoo.so`
- Las dependencias se ven con `ldd (1)`

Position Independent Code (PIC o PIE)

- Es código que puede ejecutar correctamente en forma independiente de su posición en la memoria.
 - opción `-fPIC` o `-fPIE` (Executable) de `gcc(1)`
 - usan `global offset tables` (GOTs) para las variables globales
 - ver el trabajo de laboratorio.
- Las PIC son compartidas, las PIE no (usadas en *S. O. seguros*)
- Las DLL de Windows no son PIE ni PIC

Run Time Linking en Linux

- Se hace por medio de la interface de programador del linker dinámico.
 - `son dlopen()`, `dlsym()`, `dlclose()`, `dLError()`
 - ver el laboratorio para un ejemplo.
- También se usa por razones de seguridad (*exec shield*)
 - Evita el ataque de *vuelta a libc* (variante de Buffer Overflow explicado *acá*).