Programování ve Windows
Dynamické knihovny

Andrea Číková
Martin Osovský

Agenda

- K čemu jsou dobré?
- Jaký mají vztah ke spustitelným souborům a procesům?
- Jak se sestavují?
- Jak se načítají?
- Jak se hledají?
- Rebasing a binding
- API hooking

Význam dll knihoven

- Velký význam ve windows – sdílení kódu
  - kernel32.dll – procesy, vlákna a paměť
  - user32.dll – okna a zprávy
  - gdi32.dll – grafika a texty
  - advapi32.dll, comdlg32.dll, comctl32.dll
  - další pro ještě specializovanější věci (rpc ap.)
  - C runtime library

Příklady použití

- rozšíření možností aplikací (pluginy, umožnění úprav třetím stranám, API)
- zjednodušení procesu vývoje (ale aplikace by měla mít co nejméně souborů)
- šetření pamětí (dll se do paměti načítá jen jednou – crt)
- sdílení resourců (dialogy, ikony, řetězce)
- lokalizace

Příklady použití

- rozdíly v platformách
- zvláštní vlastnosti kódu v knihovnách
  - dá se hookovat
  - hostují COM a ActiveX komponenty
  - mohou hostovat .NET knihovny

Dll a spouštěná aplikace

- dll je PE modul, ke kterému se loader chová jinak než k exe modulu
- dll se mapuje do adresového prostoru procesu
  - implicitně při načítání aplikace
  - explicitně voláním funkce
- po načtení je kód k disposici všem vláknům procesu
- sdílení kódu používá copy on write (globální a statické proměnné)
- dll se rozpustí v paměti procesu, který ji načetl

dll a paměť

- statická vs. dynamická crt
- kód, který uvolňuje paměť by měl být pro jistotu ve stejném modulu jako ten, který ji rezervoval

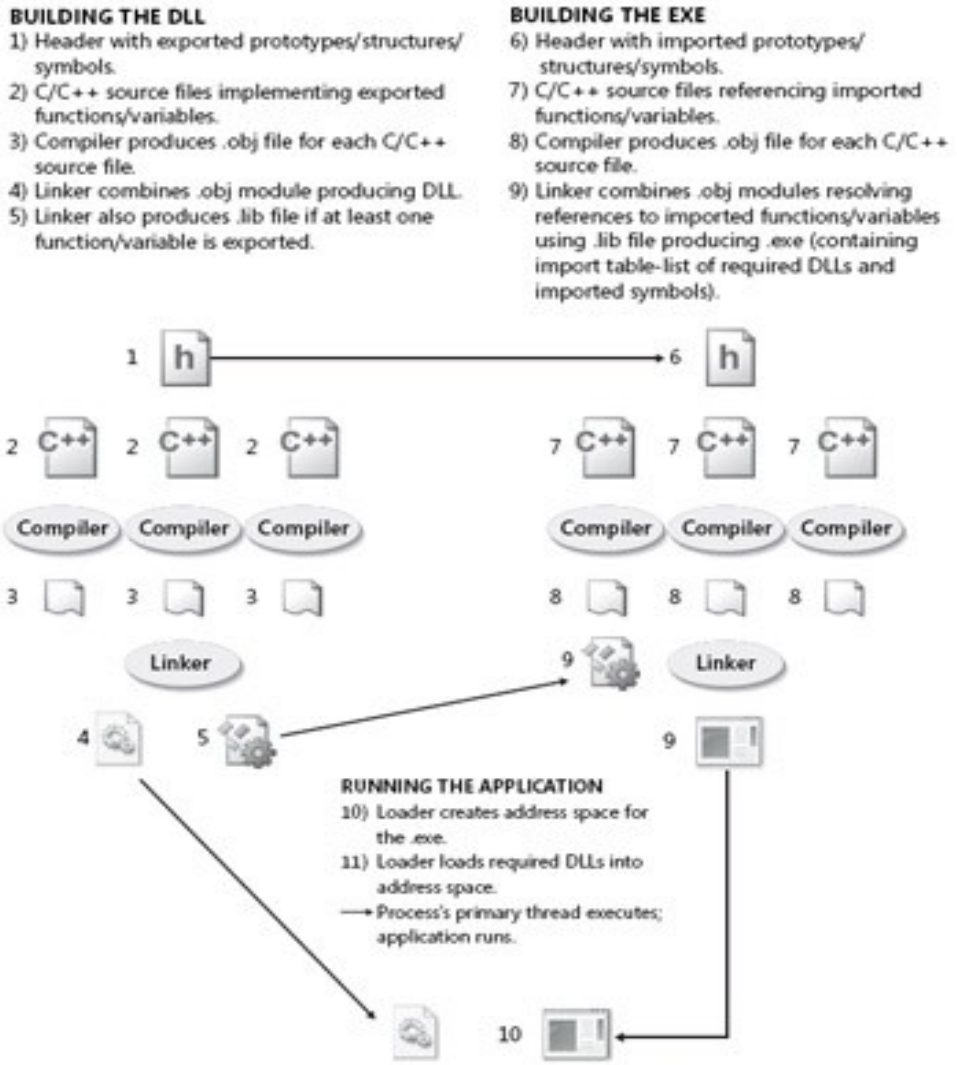
Sestavování a implicitní načítání

- je třeba dostat symboly do tabulky exportů dll modulu – declspec(dllexport)
- je třeba, aby se neprovádělo pozmněňování jmen – extern „C“
- v cílovém modulu je třeba deklarovat funkci jako extern nebo lépe jako declspec(dllimport)
- exportované funkce jsou v souboru .lib, který se linkuje do cílového modulu (pro dll je to jen seznam symbolů, jinak statická knihovna)
- je třeba použít sdílený hlavičkový soubor

Sestavování

- jiná možnost je použít def soubor se sekcí EXPORT následovanou jmény symbolů
- nebo parametr linkeru /export
- utilita DumpBin s parametry imports nebo exports

Sestavování - ilustrace



Načítání knihovny

- knihovna se načte do paměťového prostoru na tzv. module base address
- modul specifikuje preferred base address
- každý symbol má relative virtual address
- z import sekce exe se načte, které knihovny se mají načíst
- ty se načtou a s jejich import sekcemi se provede totéž
- pak se spočítají skutečné adresy symbolů a ty se vloží do import sekce exe (binding)

Načítání knihovny

- pokud nějaký modul nelze vložit na jeho PBA, provede se tzv. rebasing
- rebasing lze provést pro všechny moduly předem pomocí rebase.exe
  - je to složitý proces (musí se měnit všechny adresy u všech instrukcí v modulu)
  - kvůli copy on write to zabírá místo v systémovém page file
- binding lze provést předem pomocí bind.exe
  - zápis do import sekce také přesune paměť do systémového page file

Ruční načítání

- LoadLibraryEx
- parametr flags:
  - DONT_RESOLVE_DLL_REFERENCES
  - LOAD_LIBRARY_AS_DATAFILE
  - LOAD_LIBRARY_AS_IMAGE_RESOURCE
- LoadLibrary - bez flags
- Uvolnění knihovny FreeLibrary
- Proces si drží počet odkazů na nějakou knihovnu (je-li 0, uvolní ji z paměti - virtuální)

Načítání symbolu

- je třeba mít typedef na příslušný typ funkce a proměnnou tohoto typu
- GetProcAddress načte adresu příslušného symbolu v dll
- symbol voláme normální notací volání funkce

Vstupní bod Dll

- Dll může obsahovat funkci DllMain
- ta se volá
  - když se knihovna načte (hlavní vlákno, nebo vlákno, které zavolalo LoadLibrary)
  - když se vytvoří vlákno a knihovna je již načtena (příslušné vlákno)
  - když končí vlákno a knihovna je ještě načtena (příslušné vlákno)
  - když se knihovna uvolní z paměti (hlavní vlákno, nebo vlákno, které zavolalo FreeLibrary)

Vstupní bod

- parametr reason rozlišuje, proč byla DllMain zavolána
- při hodnotě `DLL_PROCESS_ATTACH` se neignoruje návratová hodnota, pokud je FALSE
  - při implicitním načítání zhavaruje celý proces
  - LoadLibrary vrátí NULL