Dr. Hajnal Éva: Haladó programozás

HALADÓ PROGRAMOZÁS

DLL használata és készítése
Feladat

DLL készítése

- 1 solution, 2 projekt: DLL készítő + DLL használó programok
- A DLL-t készítő alkalmazás típusa: Class Library
- A DLL-t használó alkalmazás referenciái közé fel kell venni a DLL file-t (a Visual Studio segít: akár magát a projektet is fel tudjuk venni, mint referenciát)
- A DLL file alapértelmezetten mindig a készülő EXE file mellé másolódik
- A referencia hozzáadása után a DLL-t használó alkalmazásban a DLL névtere és osztálya(i) szokványosan elérhetőek

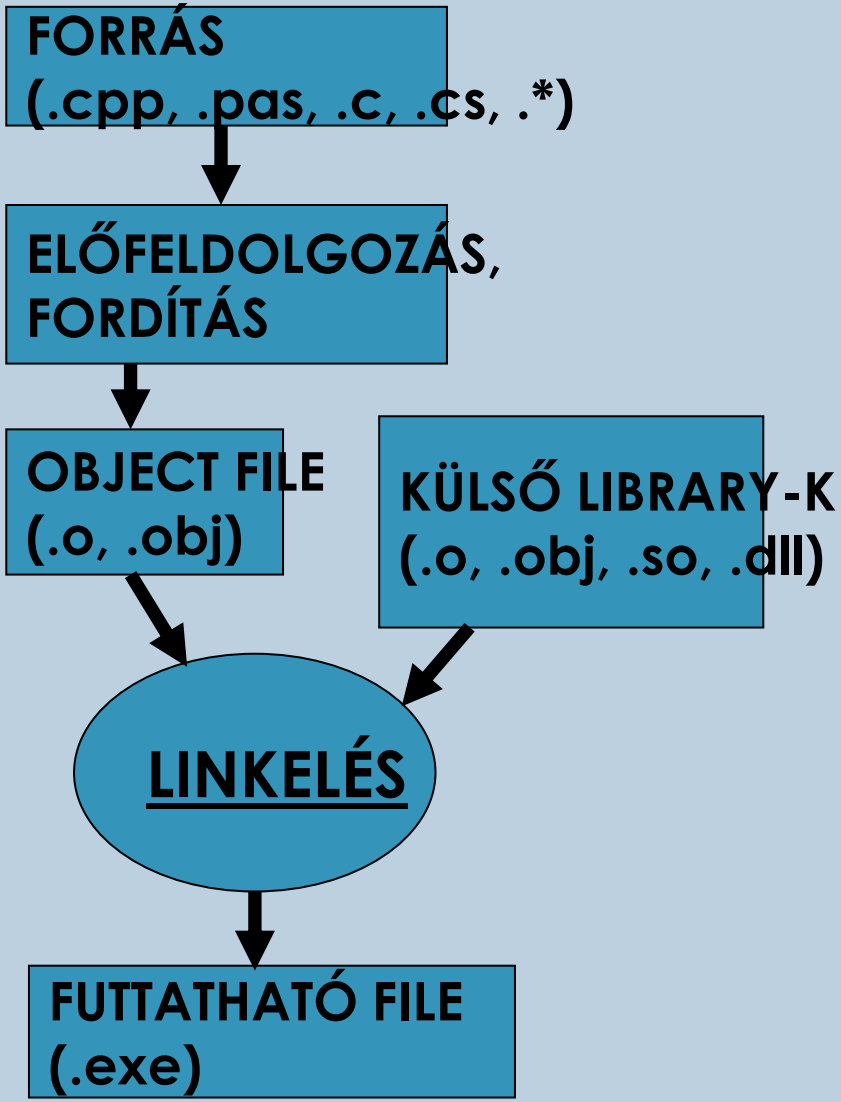
DLL ALAPJAI

Haladó Programozás

Dynamic Link Library / Shared Object

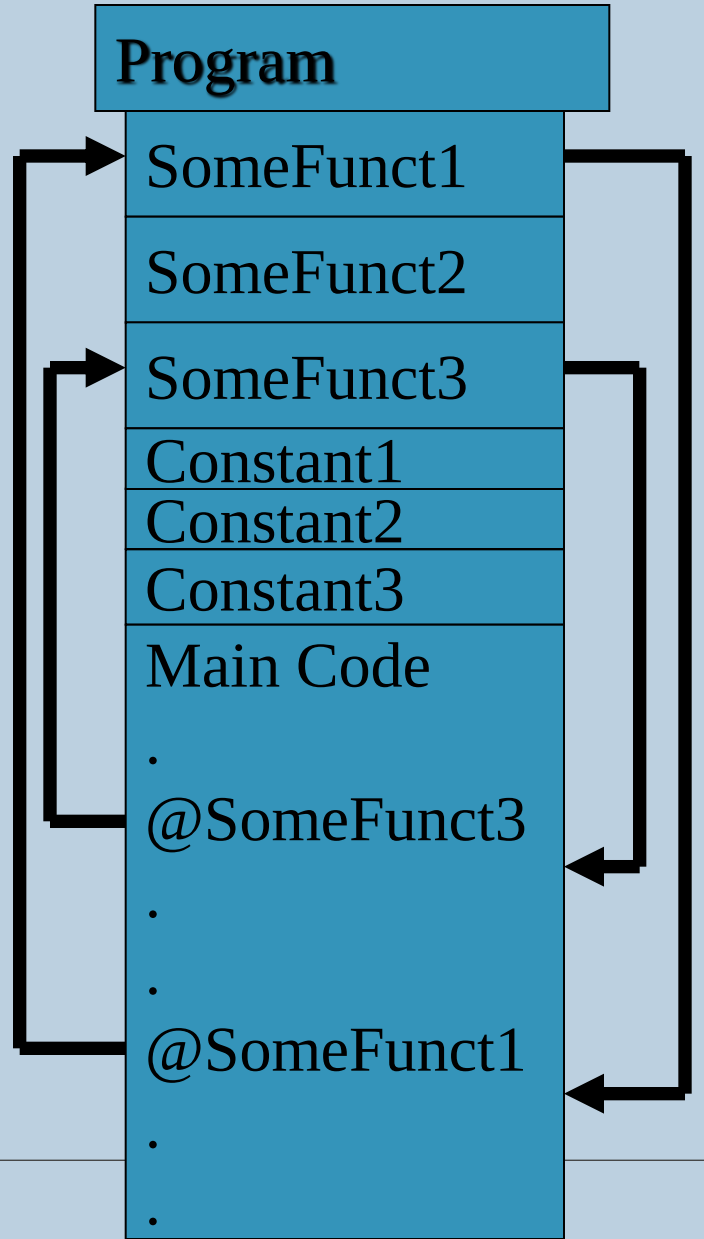
- Program modul, ami tartalmazhat programkódot, megosztott erőforrásokat, melyek elérhetők más modulokból
- Külön fordított, a programhoz ténylegesen csak a futási időben kapcsolódik (= dynamic)
- A memóriába csak egyszer töltődik be, több program is használhatja (= shared, .NET alatt ez korlátozottan igaz)
- A mai operációs rendszerek felhasználó számára elérhető programjai/moduljai kizárólagosan így működnek (bár több nyelvben is lehetséges statikus fordítást kérni, nem illik)
- Windows OS alatt meg kell különböztetni a natív és a felügyelt DLL-t:
  - **Natív: OS/CPU számára értelmezhető bytekód: procedurális kód, egyszerű típusok, közvetlen HW elérés is akár**
  - **Felügyelt: Egy (vagy több) .NET CLR osztály van benne**

Klasszikus fordítás

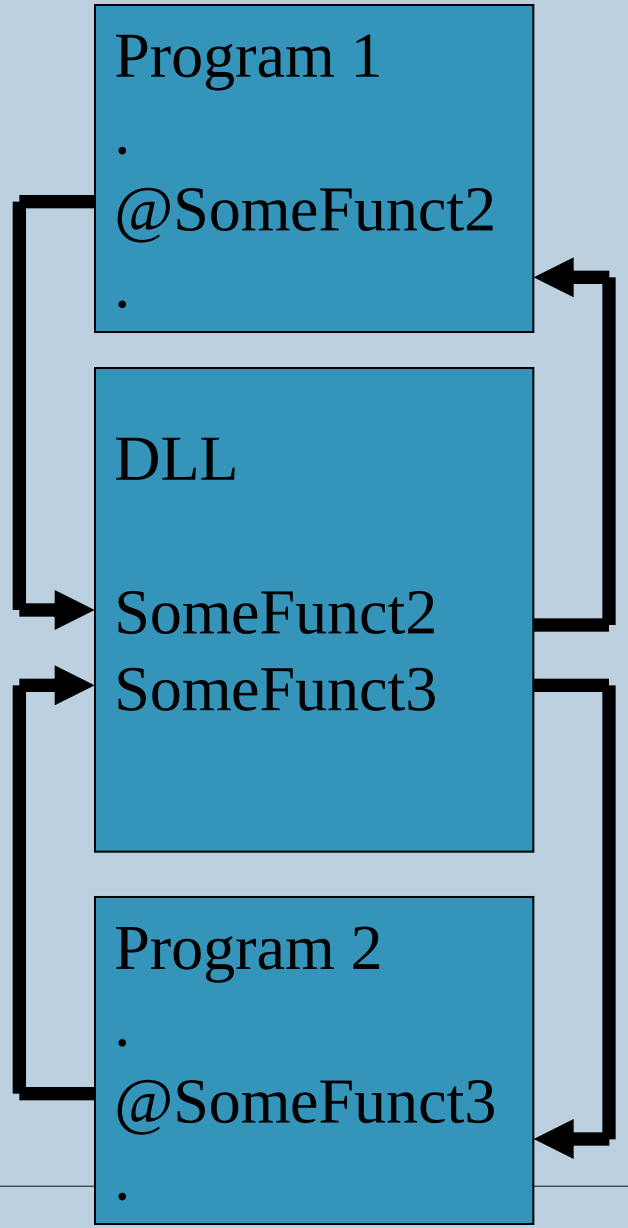


- Object File: Egy köztes kód-reprezentáció, a fordító generálja a forráskód lefordítása után
- Tartalmazza: az értelmezett, fordított kódot és relokációs adatokat, utóbbit a linker használja a futtatható állomány generálásához
- A külső library kód static linking esetén bekerül az EXE/ELF állományba; dynamic linking esetén nem

Static linking

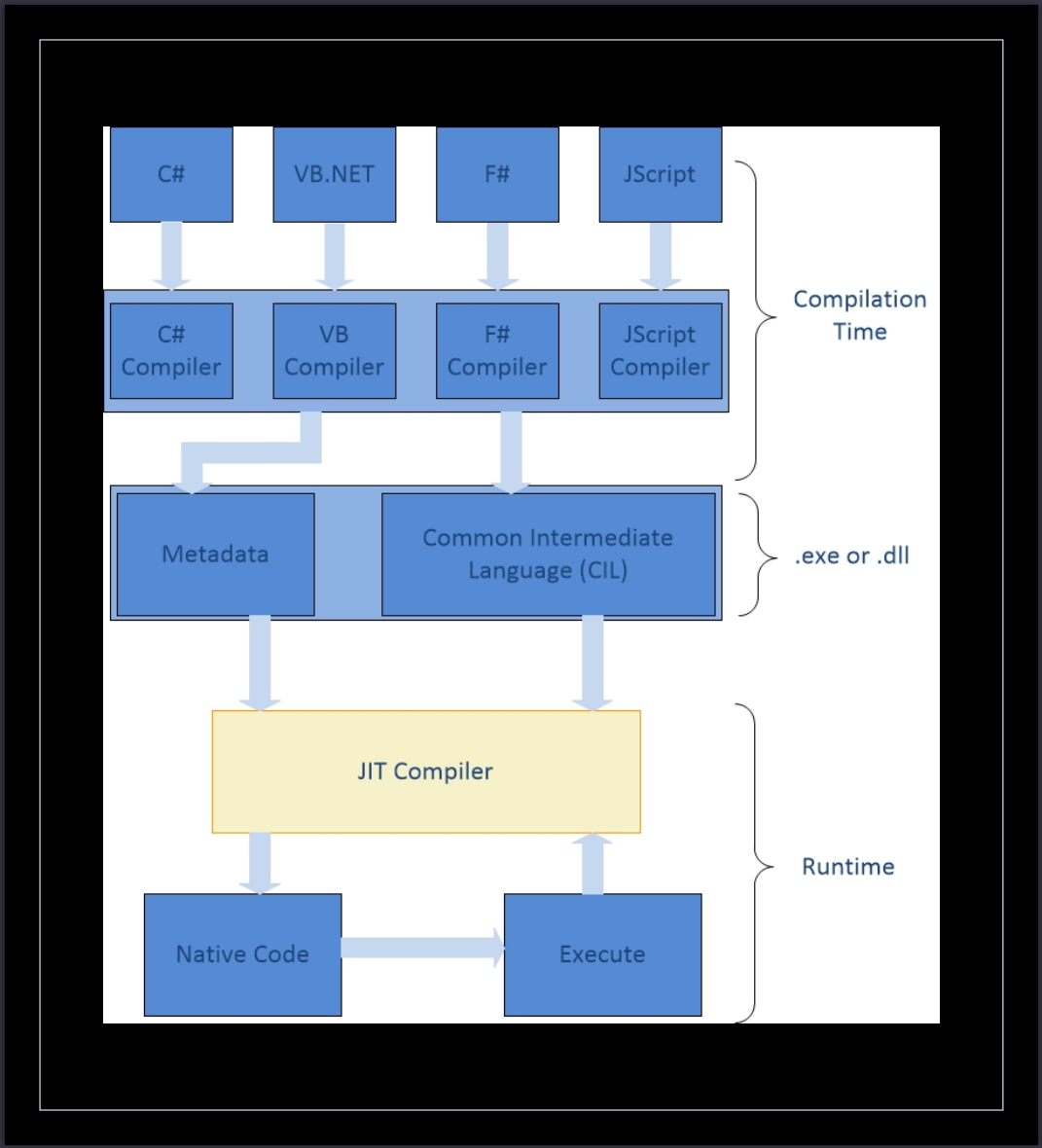


- Az összes használt függvény/ erőforrás a programban van
- Ezek helye a fordítóprogram számára ismert
- Így fordításkor az ezekre történő hivatkozás előre megadható, mert az adott kód a program saját címterületén található
- Ugyanazt a függvényt több program is külön-külön betölti ➔ pazarlás
- A Klasszikus (nem overlay) DOS programok (pl: Turbo Pascal)



Dynamic linking

- Egyes függvények/erőforrások a program saját címterületén kívül vannak
- Ezek helye a fordítóprogram számára nem ismert
- Így fordításkor az ezekre történő hivatkozás dinamikus, futási időben derül ki a pontos hely (a betöltést az OS végzi)
- Ugyanazt a függvényt több program is együtt használja ➔ megosztott erőforrás (~ Shared Object)
- A Legtöbb modern program így működik



.NET JIT

- JIT =
  Just-In-Time
- Metódusonként, az OOP elvek miatt a sebességvesztés kicsi
- NGEN.EXE használható,
  - natív képgenerátor
  - de sebességet nem mindig növel
- Hasonló elv, de sokban más:
  Java Hotspot VM

Felügyelt vs. natív DLL-ek

- Natív DLL
  - **Az aktuális könyvtárból vagy a %PATH%-ból töltődik be**
  - **Lassú betöltődés**
  - **%PATH% = a WINDOWS, SYSTEM, SYSTEM32 könyvtárak**

    ➔ DLL HELL
- Felügyelt DLL
  - **<u>Minden</u> függvény egy DLL hívás volt, amit eddig használtunk**
  - **Egy projekt „References” része tárolja azt, hogy mely felügyelt DLL-ek érhetőek el az adott projektből, ezt kell szerkeszteni**
  - **MSDN-en megtalálható, hogy melyik osztály/névtér melyik DLL-ben található**
  - **A felügyelt DLL-ek közös tárban regisztráltak (GAC), ezáltal kezelhetőek a különféle verziók és függőségek**
  - **Gyors betöltődés, ugyanolyan sebességű, mintha a saját kód lenne**

DLL elérése a .NET keretrendszerben

- Felügyelt DLL hívása
  - **Referencia hozzáadása: Project/Add reference**
  - **Ezután a DLL-ben tárolt névtér és az abban tárolt osztályok/metódusok a szokványos módon elérhetőek**
- Platform Invoke (P/Invoke: natív bytekód hívása felügyelt környezetből) ➔ DllImport attribútum
  - **using System.Runtime.InteropServices;**
  - ```
    [DllImport("winmm.dll", SetLastError = true)]
    static extern bool PlaySound(string pszSound,
                                 UIntPtr hmod, uint fdwSound);
    ```
  - ```
    string fname = @"c:\Windows\Media\tada.wav";
    PlaySound(fname, UIntPtr.Zero, 1);
    ```
  - **Szignatúrák, importok: www.pinvoke.net**

Köszönöm a figyelmet!