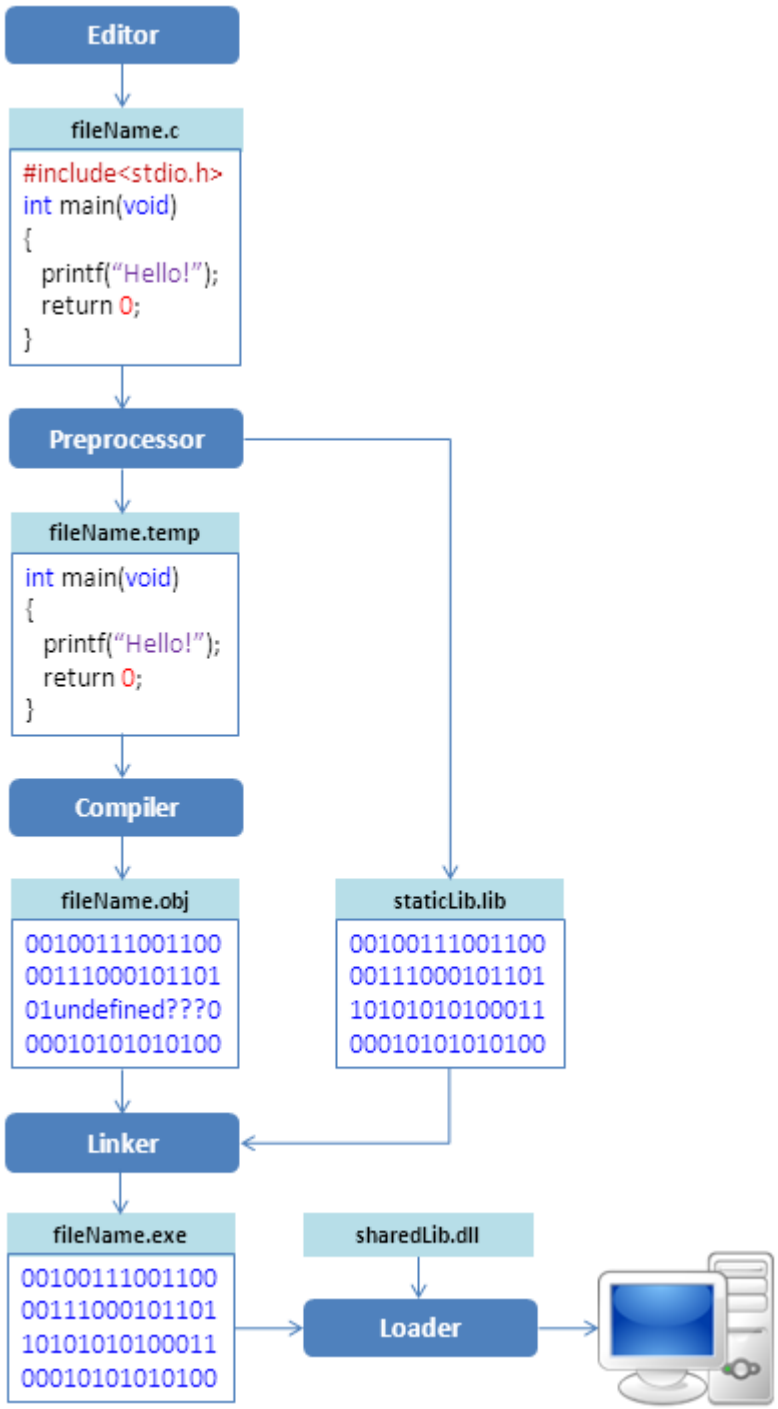
# Hogyan jön létre a forráskódból a végrehajtható állomány?

A **C** programozási nyelv használatakor a szoftverfejlesztési folyamat több munkafázisból áll. Minden fázis más-más eszköz használatát igényli. Az alkalmazások fejlesztéséhez: *szerkesztő*, *fordító*, *linker* és *betöltő* eszközök szükségesek. A **C** programozáshoz használt **IDE** (interaktív fejlesztő környezet) legtöbbje (például Code Blocks, Eclipse, VS Code, Visual Studio, IntelliJ, Codelite, Clion stb.) beépítve biztosítja ezeket, a szükséges eszközöket.

A **C** nyelven írt programokat futtatásuk előtt le kell fordítani. Ez azt jelenti, hogy a forrásfájlokat (a programozók számára is olvasható szöveges fájlokat) olyan **gépi kódú** állománnyá kell átalakítanunk (fordítanunk), amely egy mikroprocesszor (CPU) számára közvetlenül végrehajtható. A forráskód megírásától, a bináris állomány keletkezéséig a folyamat több lépcsőben zajlik. Ebből következik, hogy egy, már lefordított futtatható állomány, nem lesz kompatibilis/futtatható bármilyen CPU-val, hanem csak azokkal amik értik a benne szereplő utasításkódokat.

A következő ábrán láthatjuk, hogyan jutunk el a szöveges programtól a gép által is végrehajtható bináris, gépi kódú reprezentációhoz.




## Editor

Egy **C** alkalmazás fejlesztésének első lépése a forráskód írása/szerkesztése. A forráskód tartalmazza az összes utasítást, amelyet a gépnek szöveges formátumban (az emberek számára érthetően/olvashatóan) végre kell hajtania. Használhatunk egyszerű szöveges szerkesztőt (például Notepad++) vagy az IDE által biztosított fejlettebb szerkesztőt. A forráskódot a **C nyelv szintaxisa**/nyelvi szabályai által előírtak szerint kell megírni, miután a forrásfájl elkészül, és **.c** kiterjesztésű fájlként kell lementeni. A szerkesztő támogatja a forráskód szintaktikai színezését (syntax highlighting), automatikus kódkiegészítést, töréspontok (**breakpoints**) elhelyezését, gyors navigációt a program egyes részei között, stb…

### Fordítás / Compiling

A forrásfájl lefordításához egy fordítóra van szükségünk (gcc, mingw, cc, stb..) C fordítók. Ha egy egyszerű szövegszerkesztőt használtunk a forráskódhoz, telepítenünk kell a **C** fordítót a számítógépünkre, és “kézzel”, parancssorból meg kell hívnunk a fordítót. Egy egyszerűbb megoldás egy IDE (interaktív fordítást segítő környet) használata (amelynek már tartalmazza a fordítót) és egy menügomb megnyomásával is lehet a folyamatot elvégezni.

A fordítási feladat a forrásfájl kezdeti feldolgozási szakaszát tartalmazza. **A kezdeti fázist előfeldolgozásnak nevezzük**. Az előfeldolgozást az előfeldolgozó (**preprocessor**) hajtja végre, amel a fordító része. Az előfeldolgozó pl. a # (hash) karakterrel kezdődő összes sor forráskódját megpróbálja megkeresni. Ezeket a sorokat fordító irányelveknek/direktíváknak nevezzük. (ha nem tudja feloldani a direktívákat, akkor hibaüzenetet kapunk)

A fordító egyik feladata, olyan funkciók beillesztése, amelyek a forráskódunkon kívül vannak meghatározva. Az előfeldolgozó eltávolítja az összes **fordítói direktívát** a forráskódból, de emlékszik, hogy milyen további fájlokat kell beépíteni a folyamat későbbi szakaszába. Az előfeldolgozás végén létrejön egy ideiglenes fájl, amely a felhasználó számára közvetlenül már nem látható, ez tartalmazza az összes behelyettesített információt/forráskódot.

Az előfeldolgozás befejezése után a **fordítási folyamat** elkezdődhet. A fordító alakítja át (konvertálja) a forrásfájlt objektumfájlba. Az objektumfájlt gépi kódnak is nevezzük, és a számítógép vagy a mikrovezérlő központi feldolgozó egysége értelmezheti (CPU).

Ekkor az objektumfájl kész, de hiányzik még néhány meghatározatlan hivatkozás. Ezek a meghatározatlan hivatkozások kóddarabok, amelyeket egy másik helyről kell beilleszteni. Esetünkben a meghatározatlan referencia a **printf()** függvény. A fordító tudja, honnan szerezhető be ennek a funkciónak a kódja, mert azt a fordító direkíva ezt már korábban meghatározta (**#include <stdio.h>**).

Az **stdio.h** egy fejléc fájl (kiterjesztés ***.h**), amely többek között magában foglalja a **printf()** függvény deklarációját/(meghatározását), de magát a gépi kódú megvalósítását még nem, mert ezt majd egy későőbbi (linker) fázisban kapja meg! Ebben a fázban a fejlécfájl beillesztésével meghatározzuk, hogy hol található a printf() függvény meghatározása. Itt csak az érdekes, hogy a függvény milyen paraméterekkel használható, illetve, milyen visszatérési értékkel dogozik.

A fordítás során statikus és dinamikus fordítást is megadhatunk, aminek a linkelésnél és a futtatáskor lesz jelentősége.

### Linkelés / Linking

Ebben a szakaszban össze kell állítanunk az alkalmazásunk által használt összes fájlt. Ez azt jelenti, hogy rendelkeznünk kell az objektum fájlokkal és a statikus könyvtári fájlokkal a külső funkciókhoz. A statikus könyvtári fájlok (**.lib**) tartalmazzák a forrásfájlunkban használt külső függvények végrehajtható kódját. Ebben a konkrét esetben a statikus könyvtárfájl tartalmazza a **printf()** függvény gépi kódját. A linkernek meg kell adni ezt a könyvtári fájlt. (nagy segítség, ha IDE-t használunk, akkor ezeket a beállításokat a háttérben az IDE elvégzi.)

A linker keresni fogja az összes objektumfájlt, és az összes meghatározatlan hivatkozást felváltja a könyvtári fájlokban szereplő hivatkozott gépikódra. Az összekapcsolási folyamat végén lesz egy

végrehajtható fájl (például ***.exe** Windows alkalmazásokhoz, ***.hex** mikrovezérlőkhöz).

### Betöltés/Indítás

Az utolsó lépés a programfájl betöltése a számítógép memóriájába, hogy végrehajtható legyen. Ezt egy betöltő hajtja végre, ami az adott operációs rendszer része. A futtatható program futtatásakor elindul a betöltő, amely betölti a programot a memóriába, és megkezdődik a végrehajtás.

Mikrokontroller alkalmazás esetén a betöltési feladat alapvetően a Flash memória újraprogramozása az újonnan létrehozott futtatható fájlunkkal (*.hex).

### Debug és Release fordítási mód

A C és C++ nyelvek érdekessége, hogy megkülönbözteti a nyomonkövetési információt is tartalmazó módot, az úgynevezett DEBUG és az optimalizált fordítást a RELEASE módot. A RELEASE mód olyan fordítási módot jelent, ahol a fordító jelentős optimalizálási képességéből adódóan az eredeti forráskód és a lefordított gépi kódú utasítások sorrendiségükben eltér(het)nek. Ez azt jelenti, hogy a forráskód alapján lépésenként nem föltétlenül lehetne haladni nyomkövetéskor.

From:
https://edu.iit.uni-miskolc.hu/ - **Institute of Information Science - University of Miskolc**

Permanent link:
**https://edu.iit.uni-miskolc.hu/tanszek:oktatas:szamitastechnika:complier_mukoedese**

Last update: **2023/10/03 16:45**