## Dinamikus szerkesztés

Nagyméretű programokban bizonyos eljárások csak nagyon ritkán szükségesek. Ezeket nem kell statikusan a programhoz szerkeszteni.

Csatoló táblázat (Linkage Segment): minden esetleg szükséges eljáráshoz egy csatoló blokkot (struktúrát) tartalmaz.

```
CSAT_STR  STRUCT
CIM       DD     FAR PTR CSATOLO
NEV       DB     '        '; 8 szóköz
CSAT_STR  ENDS
```



```
CSAT_STR  STRUCT
CIM       DD     FAR PTR CSATOLO
NEV       DB     '        '; 8 szóköz
CSAT_STR  ENDS
```

Pl. a dinamikusan szerkesztendő **ALFA** eljáráshoz az:

```
ALFA      CSAT_STR     <, 'ALFA'>
```

csatoló blokk tartozhat. Az eljárás csatolása, hívása:

```
          MOV    BX, OFFSET ALFA
          CALL   [BX].CIM
```

Első híváskor a **CSATOLO** kapja meg a vezérlést. A csatolandó eljárás neve a **[BX].NEV** címen található. A név alapján betölti és a programhoz szerkeszti a megfelelő eljárást.



```
CSAT_STR  STRUCT
CIM       DD     FAR PTR CSATOLO
NEV       DB     '        '; 8 szóköz
CSAT_STR  ENDS
```

A szerkesztés végeztével **CIM** -et a most a programhoz szerkesztett eljárás kezdőcímére változtatja, és erre a címre adja a vezérlést:

```
          JMP    [BX].CIM
```

**JMP**, és nem **CALL**, hogy a veremben a **CSATOLO** -t hívó programhoz való visszatérés címe maradjon.

További hívások esetén a **CSATOLO** közbeiktatása nélkül azonnal végrehajtásra kerül az imént csatolt eljárás.



**CSATOLO** használhatja és módosíthatja a program szerkesztésekor készült térképet (map) és a globális szimbólumok táblázatát.

Szokásos megszorítás: a csatolandó eljárás nem tartalmazhat **EXTRN** utasítást, és egyetlen, a memóriába bárhová betölthető modulból áll. Ekkor a szerkesztés magára a betöltésre, és ennek a tényét rögzítő adminisztrációra egyszerűsödik.

A dinamikusan szerkesztett eljárásokat könyvtárakba szokás foglalni (pl.: **.dll**), az eljárások általában többszöri belépést tesznek lehetővé (re-entrant).



Továbbfejlesztés:

A csatoló paraméterként kapja meg a felhívandó eljárás nevét.

A csatoló program hoz létre egy csatoló táblázatot.

A csatoló a táblázatban ellenőrzi, hogy csatolva van-e a kívánt eljárás. Ha nincs, akkor elvégzi a csatolást, ha pedig csatolva van, akkor közvetlenül meghívja az eljárást.



Menü vezérelt rendszer esetében, ha a kiválasztott menü elemhez tartozó szövegből generálni lehet az – esetleg csatolandó, majd – végrehajtandó eljárás nevét és az eljárást tartalmazó file nevét, akkor a program bővítését, javítását a megfelelő file-ok hozzáadásával vagy cseréjével és a menü szöveg file-jának cseréjével akár üzemelés közben is elvégezhetjük.

A csatolt program törlése: a törlendő eljárás csatoló blokkjában a **CIM** visszaállítása – illetve a csatoló tábla törlése – után az eljárás törölhető.

Szokásos, hogy egy dinamikusan szerkesztendő eljáráshoz tartozik egy számláló, melynek értéke a csatolás előtt **0**.
A hívó program először „bejelenti" az igényét az eljárásra. Ha a számláló **0**, akkor megtörténik a csatolás, és mindenképpen: számláló **++**.
Ha a továbbiakban már nem igényli az eljárást, akkor „elengedi": számláló **--**.
Ha a számláló **=0**, akkor senki sem igényli az eljárást, tehát törölhető.



**Programok hangolása**

| Két operációs rendszer | MULTIX | TSS/67 |
|---|---|---|
| Alkalmazott programozási nyelv | 95%-ban PL/I | assembly |
| Program lista | 3.000 oldal | 30.000 oldal |
| Programozók száma | 50 | 300 |
| Költség | 10 millió $ | 50 millió $ |

Egy programozó néhány évig egy nagyobb feladaton dolgozva havi átlagban csak kb. 100-200 (!!) ellenőrzött utasítást ír, függetlenül az alkalmazott programozási nyelvtől, és egy **PL/I** utasítás 5-10 assembly utasításnak felel meg.

Sokkal gyorsabb **TSS/67** ?



Irodalmi adatok alapján azt lehet mondani, hogy (hangolás előtti) nagyobb programok 1%-a felelős a program futási idejének kb. 50%-áért, 10%-a a 90%-áért.

Program hangoláson azt a folyamatot értjük, amikor megállapítjuk a kritikus részeket, és ezek gyorsításával az egész program futását felgyorsítjuk.

A kritikus részek felderítése: pl. idő szerinti megszakítások címének könyvelésével.



Tételezzük fel, hogy ugyanannak a feladatnak a megoldásához assemblyben 5-ször annyi utasításra (és időre) van szükség, mint probléma orientált nyelv esetén, és az elkészült program 3-szor olyan gyors. A probléma orientált nyelven készült változatának kritikus 10%-át assemblyben újra programozzuk.



**A költségek és futási idők alakulása:**

| | programozó év | futási idő |
|---|---|---|
| **Assembly nyelv** | **50** | **333** |
| **Probléma orientált nyelv** | **10** | **1000** |
| **Hangolás előtt** | | |
| **a kritikus 10%** | **1** | **900** |
| **a többi 90%** | **9** | **100** |
| **Összesen** | **10** | **1000** |
| **Hangolás után** | | |
| **a kritikus 10%** | **1+5** | **300** |
| **a többi 90%** | **9** | **100** |
| **Összesen** | **15** | **400** |



- A program probléma orientált nyelven történő elkészítésének és hangolásának ideje (és költsége) kb. harmada (15 programozó év) annak, mintha az egészet assemblyben készítenénk (50 programozó év),
- a sebessége csak 20%-kal gyengébb (333 helyett 400).