ILE a RPG v IBM i

Vladimír Župka, 2024

Starší programové modely

OPM (Original Program Model)	RPG, COBOL, CL, Basic, PL/I
EPM (Extended Program Model)	Fortran/400, Pascal/400, C/400

ILE – Integrated Language Environment

ILE C	od verze 2.3 z roku 1993
ILE RPG, ILE COBOL, ILE CL	od verze 3.2 z roku 1994
ILE C++	od verze 5.1 z roku 2001

Účel:

- propojení programů psaných v různých jazycích, zejména RPG a C
- opětovné využívání programů členěním do procedur, modulů a servisních programů
- statické volání procedur zvyšuje výkonnost programů
- aktivační skupiny umožňují oddělené zpracování aplikací

ILE objekty

*MODULE	modul
*SRVPGM	servisní program
*BNDDIR	spojovací seznam (binding directory)

Procedura

Procedura je společný termín pro **hlavní proceduru** (main procedure) a **podproceduru** (subprocedure).
Hlavní procedura se také nazývá *hlavní program* a je volaná *dynamicky*.
Podprocedura je volaná *staticky*.
Podprocedura, která vrací hodnotu, se také nazývá **funkce**.

Modul

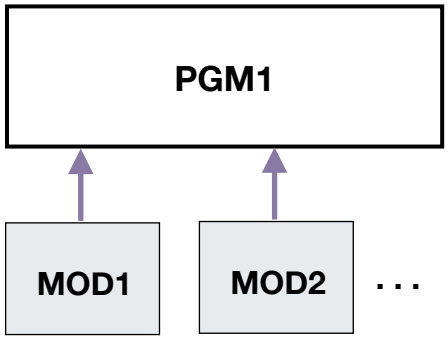
Modul je objekt typu ***MODULE** a skládá se z procedur. Není samostatně spustitelný.

Vzniká kompilací ze zdroje typu **RPGLE** příkazem

CRTRPGMOD Create RPG module (nebo klávesou **F15**)

Může definovat
- Hlavní proceduru - UEP (user entry procedure). V RPG se zvlášť neoznačuje. Zároveň kompilátor generuje proceduru - PEP (program entry procedure) pro dynamické volání. PEP je vyvolána při spuštění programu a volá dále (staticky) hlavní proceduru UEP.
- Exporty - procedury nebo data k dispozici jinému modulu pomocí klíčového slova EXPORT.
- Importy - volání procedur z jiného modulu nebo čtení dat (klíčovým slovem IMPORT) z jiného modulu.
- Debug data - údaje pro ladicí program.

ILE program složený z modulů



Program - objekt typu ***PGM** - je vytvořen (spojen) z modulů příkazem

CRTPGM Create Program

```
CRTRPGMOD MODULE(MOD1)
CRTRPGMOD MODULE(MOD2)

CRTPGM PGM(PGM1) MODULE(MOD1 MOD2 ...) ENTMOD(MOD2)        vstupní modul je MOD2
```

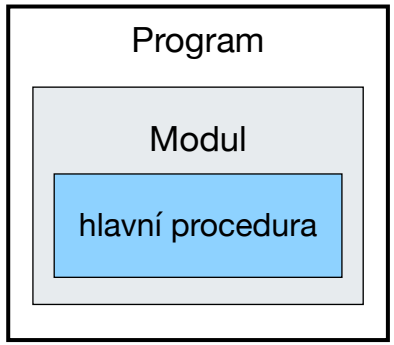
Vstupní modul musí obsahovat hlavní proceduru.

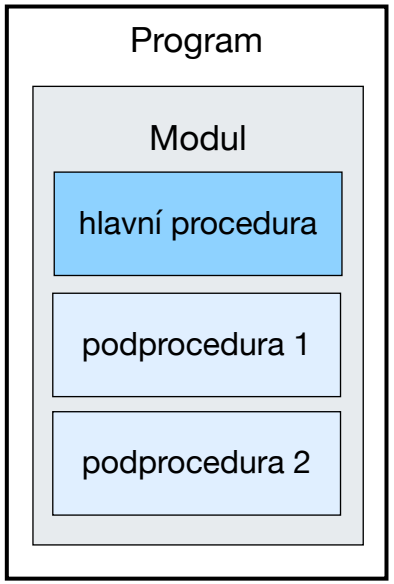
ILE program složený z jednoho modulu

Program je vytvořen ze zdroje typu **RPGLE** příkazem

CRTBNDRPG Create Bound RPG Program (nebo klávesou **F14**)

Kompiláror nejprve vytvoří modul do knihovny QTEMP a ten pak "spojí" do programu.





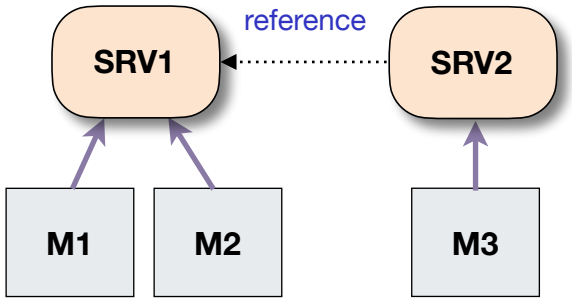
Servisní program

modul
další moduly
servisní program :
moduly, servisní programy
další servisní programy

Objekt typu ***SRVPGM** – vytváří se příkazem

CRTSRVPGM (Create Service Program)

z modulů a jiných servisních programů.




```
CRTSRVPGM   SRVPGM(SRV1) MODULE(M1 M2)
CRTSRVPGM   SRVPGM(SRV2) MODULE(M3) BNDSRVPGM(SRV1)
```

- obsahuje *tabulku* s adresami procedur a dat – *exportů*, které dává k dispozici navenek
- obsahuje jednu nebo více *signatur* pro kontrolu verzí
- není samostatně spustitelný

ILE program složený z modulů a servisních programů

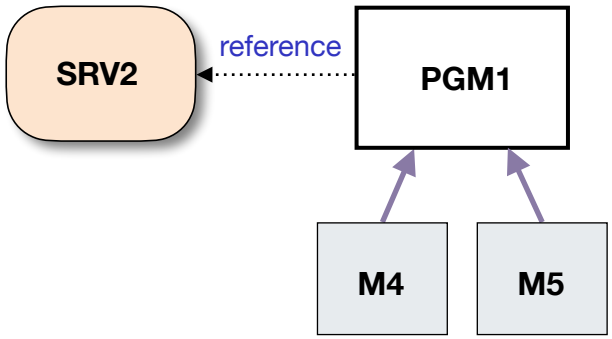
vstupní modul
další moduly
servisní program :
moduly, servisní programy
další servisní programy

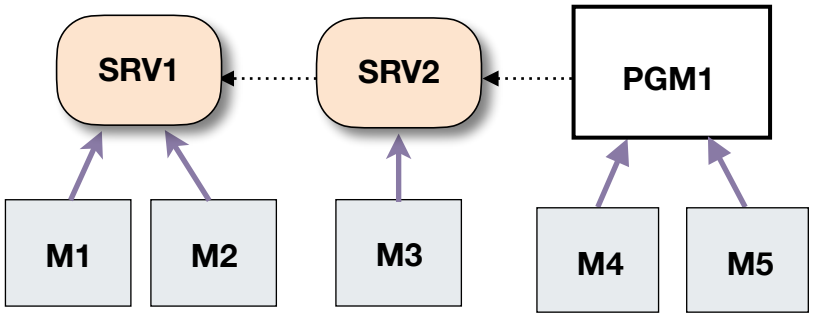
Objekt typu ***PGM** – vytváří se příkazem

CRTPGM (Create Program)

z modulů a servisních programů.

```
CRTPGM   PGM(PGM1) MODULE(M4 M5) ENTMOD(M5) BNDSRVPGM(SRV2)
```

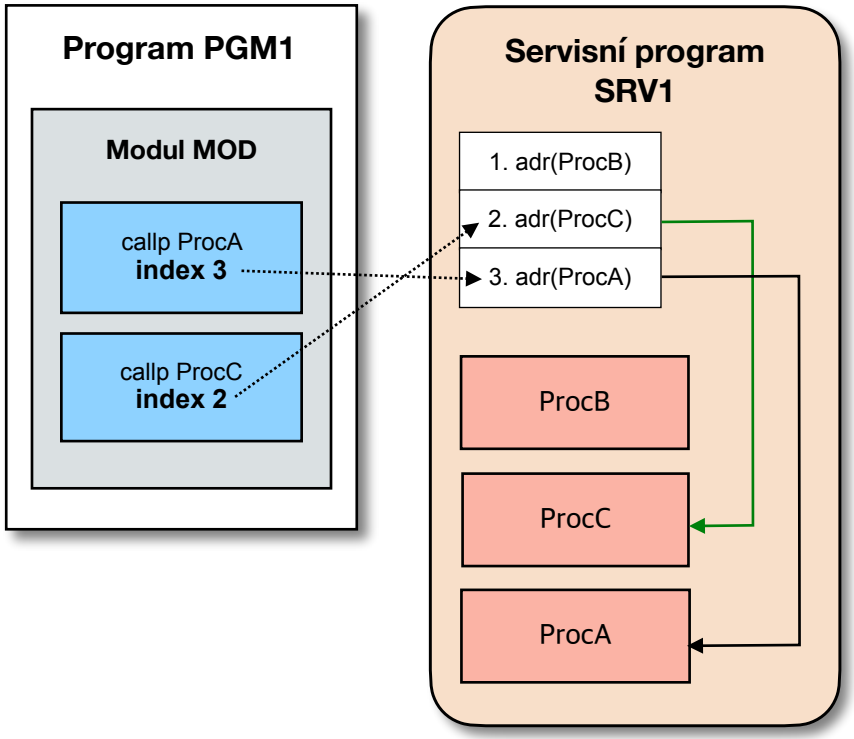




- Jeden z přímo připojených modulů musí být označen jako vstupní modul a musí tedy obsahovat hlavní proceduru, která se vyvolává dynamicky.
- Ostatní procedury lze volat jen staticky (bound call) z hlavní procedury a z procedur v připojených modulech a servisních programech.

Připojení servisního programu

```
CRTPGM   PGM(*CURLIB/PGM1) MODULE(MOD) BNDSRVPGM(SRV1)
```



- Servisní program je k programu připojen volně - referencí
- Program i servisní program zůstávají samostatnými objekty
- Do programu se zapíší referenční údaje ze servisního programu:
  - *indexy* na položky v tabulkách adres exportů (procedur)
  - nejnovější (*CURRENT) *signatura* servisního programu

Signatura, export a spojovací text

Signatura je 16bajtový údaj vyjadřující verzi servisního programu. Servisní program může být, a je zpravidla připojen k mnoha programům. Signatura slouží ke kontrole, zda nová verze servisního programu bude vyhovovat programům, k nimž byl dříve připojen.

```
CRTSRVPGM ... EXPORT(*SOURCE) SRCFILE(QSRVSRC) SRCMBR(SRV1)
```

- signaturu a exporty určuje spojovací text zdrojového typu BND (binder source) - nekompiluje se
- pojmenovaný nejlépe shodně se jménem servisního programu

Spojovací text **SRV1** určuje tři exportované podprocedury a signaturu VER01:

```
STRPGMEXP   PGMLVL(*CURRENT) SIGNATURE('VER01')
EXPORT      SYMBOL(ProcB)
EXPORT      SYMBOL(ProcC)
EXPORT      SYMBOL(ProcA)
ENDPGMEXP
```

Signatura odráží počet a pořadí exportů.

```
CRTSRVPGM ... EXPORT(*ALL)
```

- signaturu generuje systém
- exportuje všechny podprocedury a proměnné s klíčovým slovem EXPORT

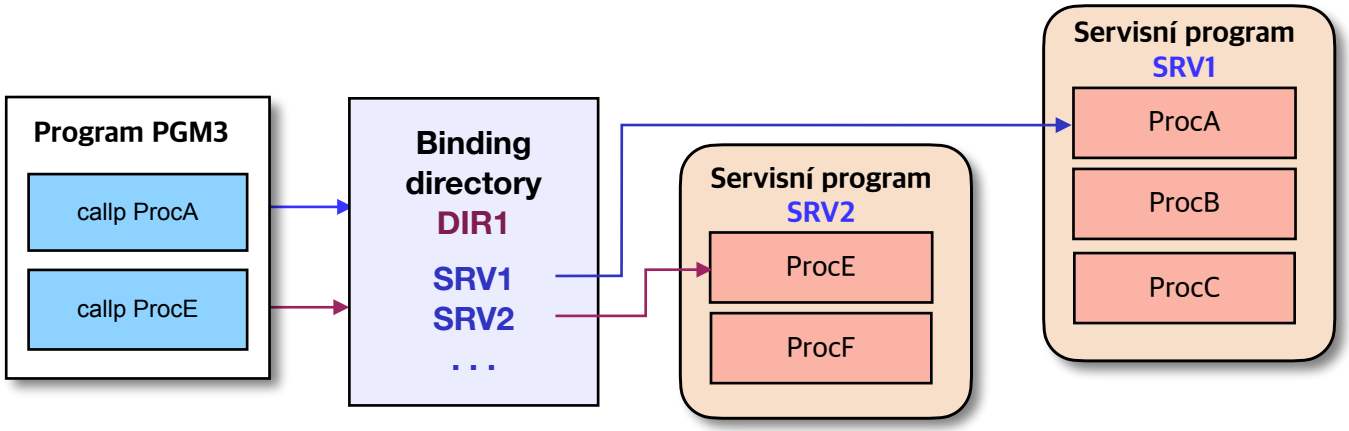
Spojovací seznam (binding directory)

Spojovací seznam je objekt typu *BNDDIR.

```
CRTBNDDIR   BNDDIR(*CURLIB/DIR1)
ADDBNDDIRE  BNDDIR(*LIBL/DIR1) OBJ(*LIBL/SRV1)
ADDBNDDIRE  BNDDIR(*LIBL/DIR1) OBJ(*LIBL/SRV2)
ADDBNDDIRE  BNDDIR(*LIBL/DIR1) OBJ(*LIBL/MOD1)
...
```

Spojovací seznamy se zapisují v parametru příkazu CRTPGM nebo příkazu CRTSRVPGM.

```
CRTPGM PGM(PGM3) BNDDIR(DIR1 ...)
```



Spojovací seznamy lze také zadat v parametru kompilace RPG modulu nebo programu

```
CRTRPGMOD ... BNDDIR(DIR1 {: DIR2...})
CRTBNDRPG ... BNDDIR(DIR1 {: DIR2...})
```

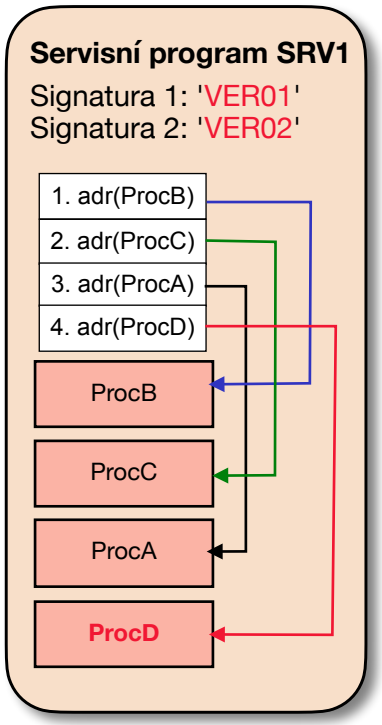
nebo v řídicím příkazu zdrojového modulu

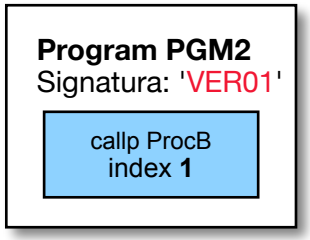
```
CTL-OPT BNDDIR('DIR1' {: 'DIR2'...})
```

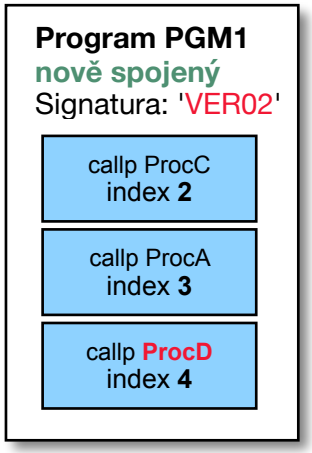
Spojovací seznam QC2LE obsahuje servisní programy a moduly s funkcemi jazyka C a systému UNIX.

Od verze 6.1 není zápis zapotřebí, protože tyto objekty se staly součástí systémového spojovacího seznamu.

Údržba servisního programu








```
/* První blok - pro současnou verzi */
STRPGMEXP PGMLVL(*CURRENT) SIGNATURE('VER02')
EXPORT      SYMBOL(ProcB)
EXPORT      SYMBOL(ProcC)
EXPORT      SYMBOL(ProcA)
EXPORT      SYMBOL(ProcD)
ENDPGMEXP

/* Druhý blok - pro předchozí verzi */
STRPGMEXP   PGMLVL(*PRV) SIGNATURE('VER01')
EXPORT      SYMBOL(ProcB)
EXPORT      SYMBOL(ProcC)
EXPORT      SYMBOL(ProcA)
ENDPGMEXP
```

1. Do servisního programu SRV1 přidáme proceduru ProcD v příslušném modulu a modul zkompilujeme.
2. Upravíme spojovací text.
3. Vytvoříme znovu servisní program SRV1 příkazem CRTSRVPGM. Servisní program nyní obsahuje dvě signatury - VER01 a VER02.
4. Příkazem CRTPGM připojíme servisní program k novému programu PGM1, který používá novou proceduru. Tento program si zapíše signaturu VER02.
5. Program PGM2 zůstane beze změny se signaturou VER01.
6. Oba uvedené programy lze spustit, protože jejich signatury se najdou v připojeném servisním programu.

Aktivační skupina (activation group)

Aktivační skupina je podstruktura úlohy (jobu).

Složení aktivační skupiny

Prostředky nutné k provozu programu:

- Proměnné programu (statická a automatická paměť)
- Dynamická paměť alokovaná ze systému
- Dočasné prostředky pro řízení dat
  - open data paths (ODP)
  - commitment definitions
  - SQL kurzory
- a další

Druhy aktivačních skupin

- předvolená aktivační skupina (default activation group) existuje stále
  - skupina číslo 1 - Hostí systémové programy, např. QCMD
  - skupina číslo 2 - Hostí aplikační programy
- pojmenovaná, při ukončení programu nezaniká
- systémová, při každé aktivaci programu se vytvoří nová; zaniká při ukončení programu
- převzatá od volajícího programu

Přidělení aktivační skupiny

Při spojování je přidělena programu nebo servisnímu programu.

CRTPGM
- ACTGRP(jméno) pojmenovaná
- ACTGRP(*NEW) systémová
- ACTGRP(*CALLER) převzatá
- ACTGRP(*ENTMOD) pojmenovaná určitým jménem podle paměťového modelu
  - QILE pro STGMDL(*SNGLVL)
  - QILES pro STGMDL(*TERASPACE)

CRTSRVPGM
- ACTGRP(jméno) pojmenovaná
- ACTGRP(*CALLER) převzatá

Při kompilaci programu složeného z jednoho modulu se přiděluje v příkazu

CRTBNDRPG (F14)
- DFTACTGRP(*NO), případně ještě ACTGRP (*NEW | *CALLER | jméno) s podprocedurami
- DFTACTGRP(*YES), bez podprocedur. Nemusí se zapisovat.

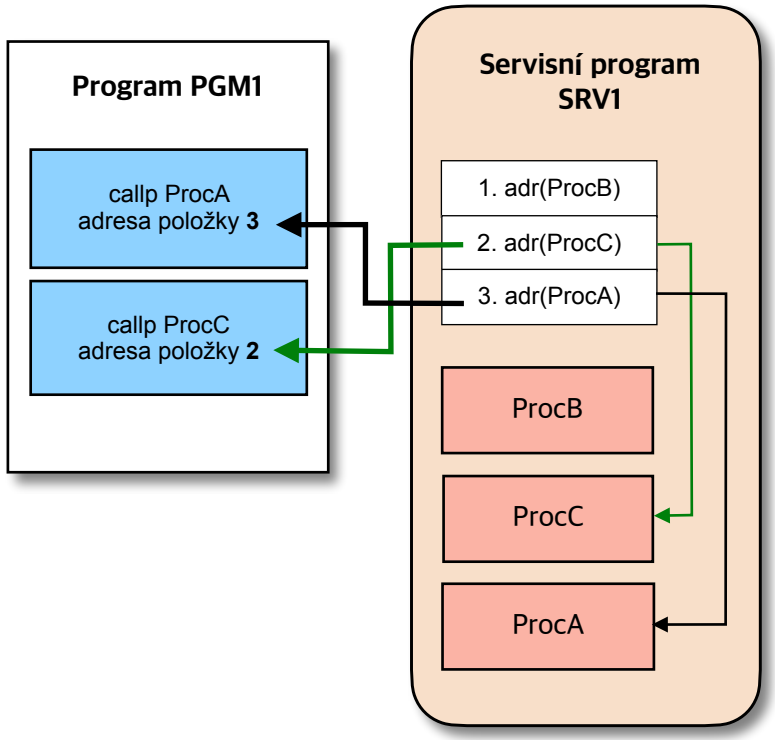
Aktivace programu

ILE program se při spuštění aktivuje v aktivační skupině, pro kterou byl vytvořen nebo v aktivační skupině volajícího programu (*CALLER).

Jestliže aktivační skupina v jobu již existuje, použije ji.

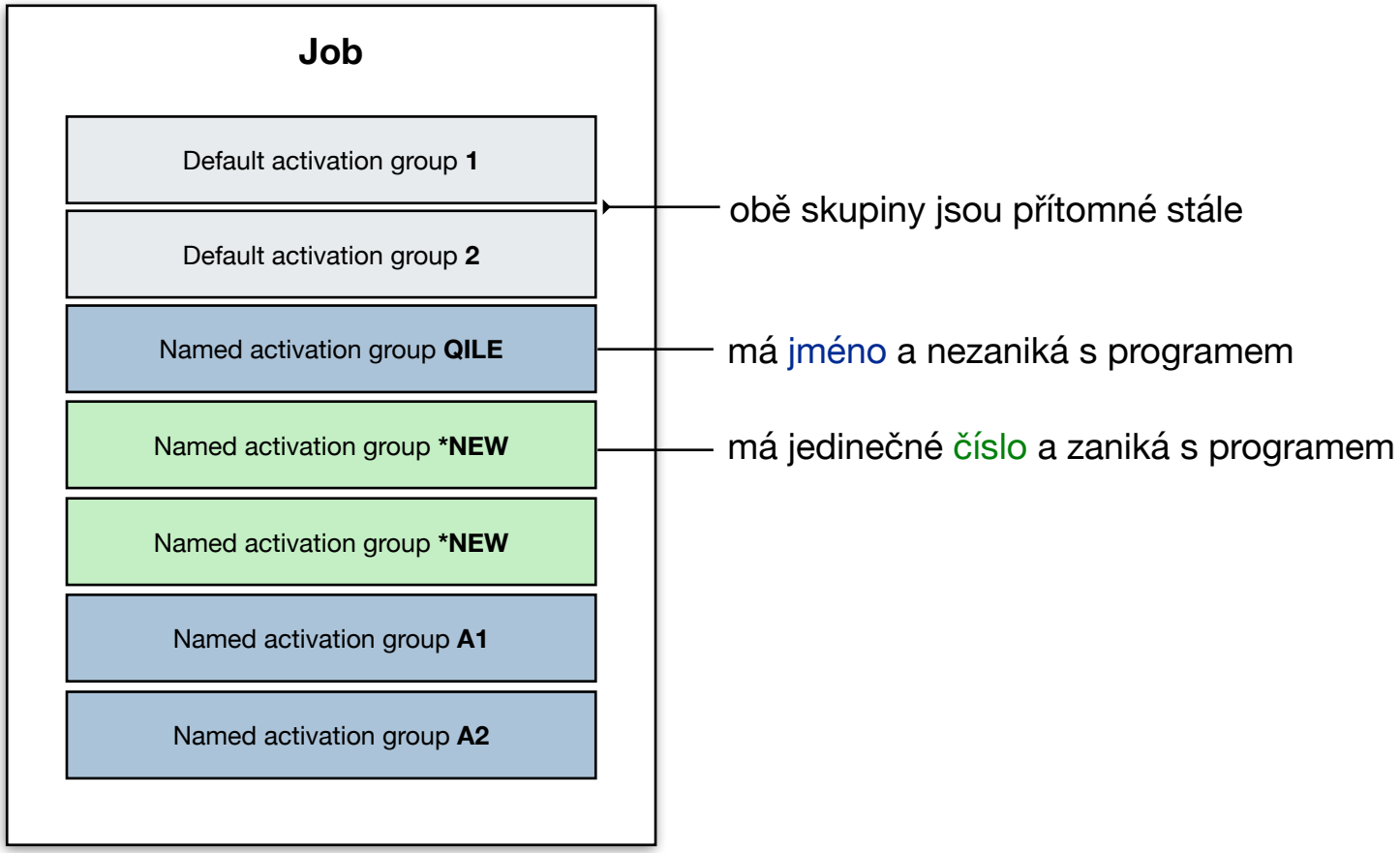
Jestliže příslušná aktivační skupina v jobu neexistuje, probíhá aktivace programu takto:

- vytvoří novou aktivační skupinu programu
- vytvoří paměť pro statická data programu
- dokončí připojení servisních programů:
  - zavede do paměti servisní programy
  - pro všechny moduly servisního programu vyhradí statickou paměť
  - kontroluje signaturu servisniho programu
  - na místo indexů dosadí adresy ukazující do tabulky adres exportů



Aktivační skupiny v jobu

Při spuštění jobu se automaticky vytvoří dvě předvolené aktivační skupiny.
Tak jak jsou postupně volány aplikační programy, vytvářejí se nové aktivační skupiny nebo se používají ty existující. Aktivační skupiny odpovídají zápisům v zásobníku volání call stack.



Program spojený ze dvou modulů

Modul **OBSAH_CALL** volá podproceduru **'obsah'**

```
// Program bez RPG cyklu
CTL-OPT MAIN(OBSAH_CALL) ; // určuje lineární hlavní proceduru

// lineární hlavní procedura
DCL-PROC OBSAH_CALL;  // formálně je zapsána jako podprocedura

    // prototyp pro volanou podproceduru
    DCL-PR OBSAH   packed(15: 5)      // vracená hodnota
                   EXTPROC('obsah'); // malá písmena!
        *n    packed(15: 5) value;   // parametry
        *n    packed(15: 5) value;
        *n    packed(15: 5) value;
    END-PR;

    dsply OBSAH ( 3: 4: 5 ); // statické volání podprocedury

END-PROC;

// nepotřebuje příkaz RETURN
```

Modul **OBSAH_PRC** zahrnuje pouze podproceduru **'obsah'**

```
CTL-OPT NOMAIN; // modul nemá hlavní proceduru

// podprocedura (funkce)
DCL-PROC  OBSAH  EXPORT ;  // podrocedura je volána z jiného modulu
    // rozhraní procedury - procedure interface
    DCL-PI *N       packed(15: 5)      // vracená hodnota
                    EXTPROC('obsah'); // malá písmena!
       a    packed(15: 5) value;     // parametry
       b    packed(15: 5) value;
       c    packed(15: 5) value;
    END-PI;
    dcl-s  s  packed(15: 5);
    if a >= b + c or b >= a + c or c >= a + b;
       return -1;
    endif;
    s = (a + b + c) / 2;       // poloviční součet stran
    return %sqrt(s * (s - a) * (s - b) * (s - c));  // výpočet obsahu
END-PROC obsah;
```

Spojení obou modulů do programu **OBSAH_PGM**

```
CRTPGM PGM(OBSAH_PGM) MODULE(OBSAH_CALL OBSAH_PRC)
```

Lokální soubor v podproceduře

```
  // hlavní procedura volá podproceduru VRATIT_CENU
ctl-OPT dftactgrp(*no);  // kvůli podproceduře v modulu
dsply %editc(VRATIT_CENU('00001'): 'P'); // volání podprocedury
return;

  // podprocedura vrací cenu pro číslo materiálu
dcl-PROC VRATIT_CENU;
    dcl-F CENIKP keyed;  // příp. i STATIC
    dcl-DS cenik_ds  LIKEREC (CENIKPF0); // je nutná datová struktura

    dcl-PI *N    packed(10: 2);    // rozhraní podprocedury
        material like(cenik_ds.MATER) CONST; // nebo VALUE
    end-PI;

    chain material  CENIKP  cenik_ds;  // čte do datové struktury
    return cenik_ds.CENA;              // vrací hodnotu
end-PROC;
```

- Lokální soubor negeneruje popisy I a O s proměnnými.
- Pro datová pole je nutné použít datovou strukturu nebo funkci %fields u UPDATE.
- Soubor se otevře vždy při vstupu do podprocedury. Uzavírá se při výstupu z podprocedury a proměnné zanikají.
- Klíčové slovo STATIC u souboru nechá soubor otevřený a zachová jeho data pro příští volání.

Soubor jako parametr volání

Společný /INCLUDE člen `FILPAR_C` pro oba moduly

```
// Šablona souboru pro kompilaci
  dcl-f infile_t TEMPLATE extdesc('CENIKP') block(*yes);

// Šablona pro formát (záznam) souboru
  dcl-ds infmt_t TEMPLATE likerec(CENIKPF0);

// Prototyp procedury zpracující soubor
  dcl-PR PROC_INFILE;
     *n    LIKEFILE(infile_t);  // soubor ze šablony
     *n    likeds(infmt_t);     // formát ze šablony jako sdružená proměnná
  end-PR;
```

Modul `FILPAR` (hlavní program)

```
/include FILPAR_C

// Definuji soubor pomocí LIKEFILE ze šablony
  dcl-f inp_file LIKEFILE(infile_t) extfile('CENIKP');  // soubor pro výpočet

// Definuji formát souboru ze šablony pro příkaz READ
  dcl-ds inp_fmt LIKEDS(infmt_t);

// Volám podproceduru PROC_INFILE s dvěma parametery – soubor a formát
  PROC_INFILE (inp_file: inp_fmt);

  return;
```

Modul `FILPAR_P` (podprocedura)

```
ctl-opt NOMAIN;  // chybí hlavní procedura

/include FILPAR_C

dcl-proc PROC_INFILE export;
   dcl-PI *n;    // rozhraní (interface) podprocedury bez jména
      file  LIKEFILE(infile_t);  // soubor
      fmt   LIKEDS(infmt_t);     // formát - datová struktura
   end-PI;

   // soubor se otevře při vstupu do podprocedury
   read file fmt;  // operace READ musí použít datovou strukturu
   dow not %eof;
      dsply (fmt.MATER + ' ' + %editc(fmt.CENA: 'P'));
      READ file fmt;
   enddo;
   // soubor je lokální, zavře se automaticky
end-proc PROC_INFILE;
```

Spojení modulů do programu

```
CRTPGM PGM(FILPAR) MODULE(FILPAR FILPAR_P)
```

Statické volání (bound call)

Vyžaduje

- Prototyp ve volající RPG proceduře/programu: DCL-PR ... END-PR – není vždy povinný
- Rozhraní (procedure interface) - ve volané podproceduře: DCL-PI ... END_PI

Podprocedura může být zapsána ve stejném nebo jiném modulu, často v servisním programu.

Volání připojené podprocedury

- jménem funkce ve výrazu

```
DSPLY  OBSAH ( 3: 4: 5 );
```

- jménem podprocedury jako příkazu, případně s CALLP

```
CALLP PROC_INFILE (inp_file: inp_fmt);
```

Dynamické volání

Dynamické volání je volání programu. *Program nemůže vracet hodnotu* na rozdíl od podprocedury. Může však vracet výsledky prostřednictvím parametrů předávaných referencí.

Volající RPG program

```
DCL-PR PGM1 EXTPGM; // prototyp programu
   *n char(10);
   *n char(10);
END-PR;

dcl-s a char(10);
dcl-s b char(10);

CALLP  PGM1 (a: b); // volání programu s prototypem
```

Volaný program PGM1

```
DCL-PI *N;        // rozhraní (program interface)
   p1 char(10);   // je místo *ENTRY PLIST
   p2 char(10);
END-PI;
...
```