Principy ILE v IBM i

Vladimír Župka, 2020

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

Procedura

RPG

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.

C, C++

Procedura je funkce. Hlavní procedura je funkce main( ) a je volaná *dynamicky*.
Ostatní funkce modulu jsou volány *staticky*.

COBOL

Procedura je COBOL program nebo ILE procedura.
Hlavní procedura je COBOL program volaný *dynamicky* (např. CL příkazem CALL).
ILE procedura je COBOL program volaný *staticky*:

- program v připojeném modulu (příkazem CRTPGM)
- program vnořený do programu.

CL

CL program je hlavní procedura volaná *dynamicky*. Podprocedura neexistuje.

Modul

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

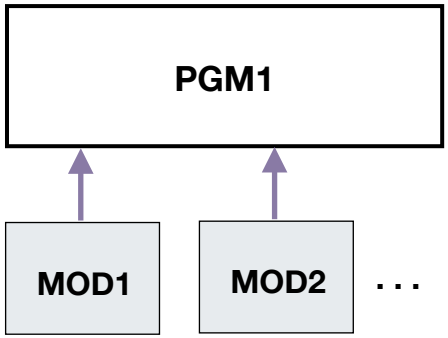
Vzniká kompilací:

CRTRPGMOD	Create RPG module	typ zdroje RPGLE
CRTCBLMOD	Create COBOL module	typ zdroje CBLLE
CRTCLMOD	Create CL module	typ zdroje CLLE
CRTCMOD	Create C module	typ zdroje C
CRTCPPMOD	Create C++ module	typ zdroje CPP

Může definovat

- Hlavní proceduru - UEP (user entry procedure). V RPG, COBOLu a CL se zvlášť neoznačuje, v C a C++ je to funkce main( ). 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.
- Importy - volání procedur z jiného modulu nebo čtení dat z jiného modulu.
- Debug data - údaje pro ladicí program.

ILE program složený z modulů



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

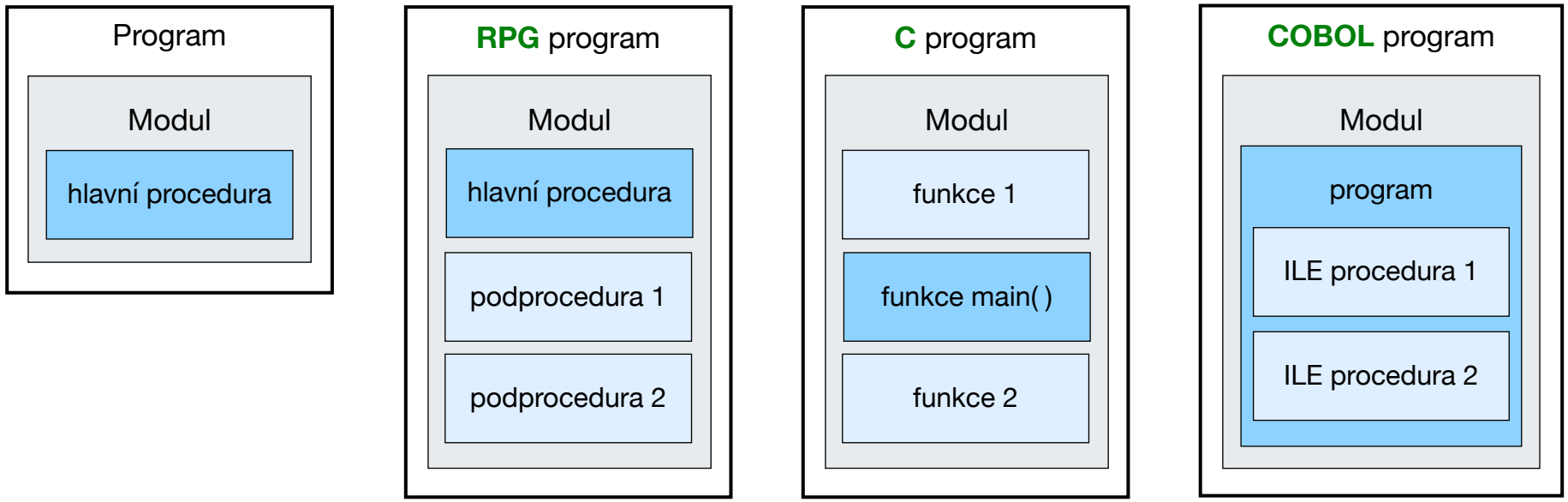
```
CRTPGM PGM(PGM1) MODULE(MOD1 MOD2 ...) ENTMOD(*FIRST)      vstupní modul je MOD1
CRTPGM PGM(PGM1) MODULE(MOD1 MOD2 ...) ENTMOD(MOD2)        vstupní modul je MOD2
```

Vstupní modul musí obsahovat hlavní proceduru.

ILE program složený z jednoho modulu

Kompilační příkazy vytvoří program tak, že zkompilovaný modul uloží do knihovny QTEMP a hned jej "spojí" do programu.

CRTBNDRPG	Create Bound RPG Program	typ zdroje RPGLE
CRTBNDCBL	Create Bound COBOL Program	typ zdroje CBLLE
CRTBNDCL	Create Bound CL Program	typ zdroje CLLE
CRTBNDC	Create Bound C Program	typ zdroje C
CRTBNDCPP	Create Bound C++ Program	typ zdroje CPP



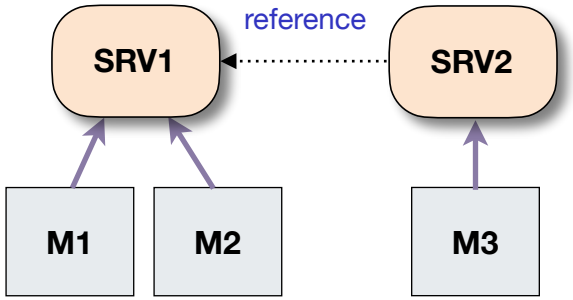
Servisní program

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(*CURLIB/SRV1) MODULE(*LIBL/M1 *LIBL/M2)
          EXPORT(*SRCFILE) SRCFILE(QSRVSRC) SRCMBR(SRV1)
          EXPORT(*ALL)
CRTSRVPGM  SRVPGM(SRV2) MODULE(M3) BNDSRVPGM(SRV1)
```



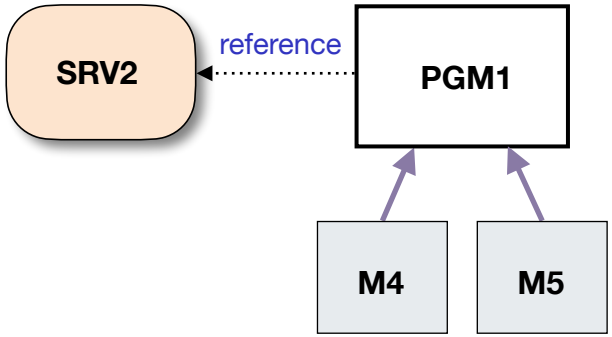
- obsahuje *tabulku* s adresami procedur a dat, tzv. *exportů*, které dává k dispozici navenek
- obsahuje jednu nebo více *signatur* pro kontrolu verzí
- není samostatně spustitelný

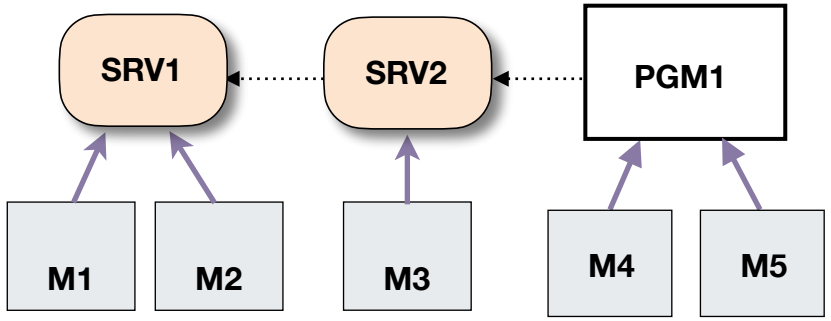
ILE program složený z modulů a servisních programů

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

Objekt typu *PGM - vytváří se z modulů a servisních programů příkazem *CRTPGM (Create Program)*.

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

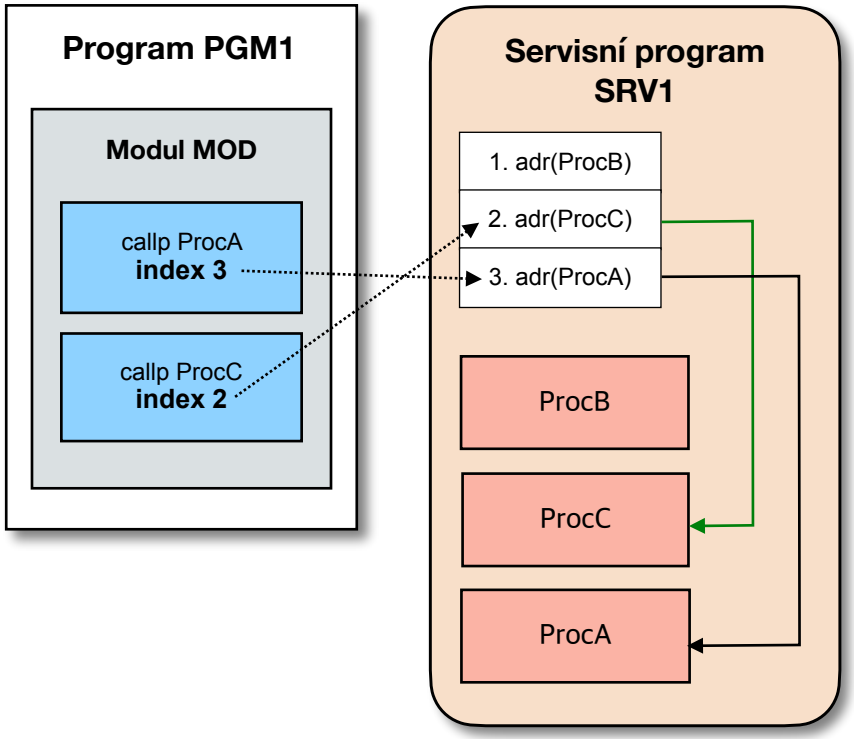




- Jeden z přímo připojených modulů musí být označen jako vstupní modul a musí tedy obsahovat hlavní proceduru.
- Ostatní procedury lze volat jen staticky (bound call) z hlavní procedury a z procedur v připojených modulech aservisních programech.

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 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.

Způsob vytvoření signatury určíme v parametru EXPORT příkazu CRTSRVPGM:

`CRTSRVPGM ... EXPORT(*ALL)` - signaturu generuje systém

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

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

Příklad: Spojovací text **SRV1** určuje tři podprocedury pro export a signaturu **VER01**:

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

Signatura odráží počet a pořadí exportů.

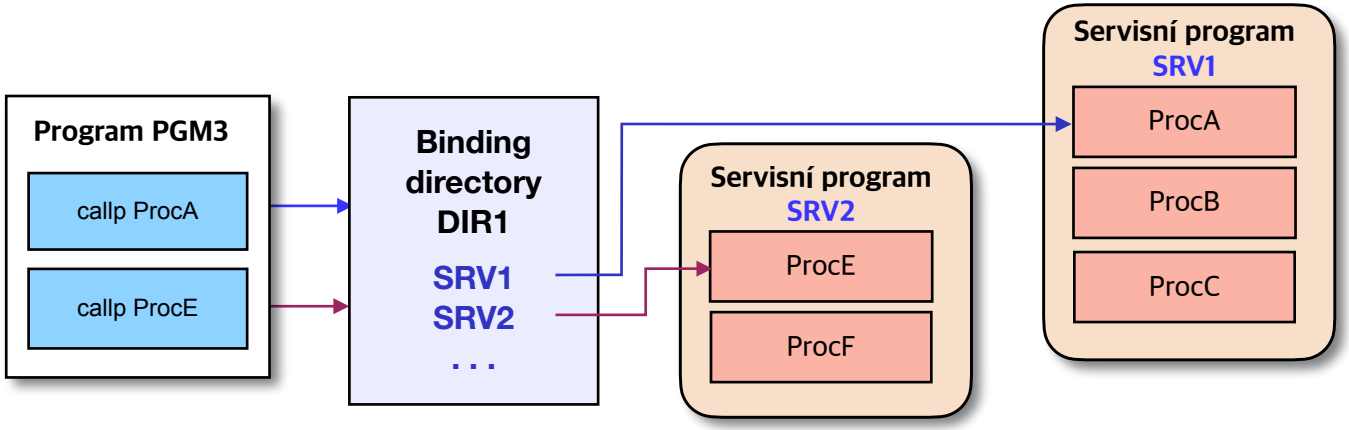
Spojovací seznam (binding directory)

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 ...)
```

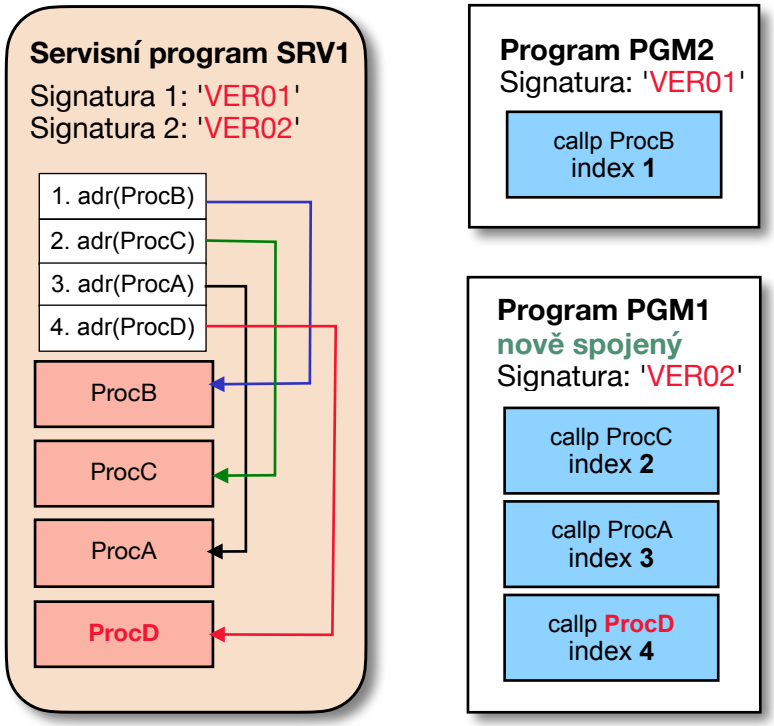


Spojovací seznamy lze také zadat v parametru kompilace **RPG** modulu nebo programu:

```
CRTRPGMOD ... BNDDIR(QC2LE {:...})
CRTBNDRPG ... BNDDIR(QC2LE {:...})
CTL-OPT BNDDIR('QC2LE' {:...})     // ve zdrojovém programu
```

Poznámka: 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 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

Aktivační skupina obsahuje všechny 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
  - aj.
- Programy pro zpracování výjimek a ukončovací procedury

Druhy aktivačních skupin

1. Předvolená aktivační skupina (default activation group) existuje stále.
   - Skupina číslo 1 - Hostí systémové programy, např. QCMD
   - Skupina číslo 2 - Hostí aplikační programy
2. Pojmenovaná aktivační skupina určená parametrem ACTGRP(name). Při ukončení programu nezaniká.
3. Systémová aktivační skupina určená parametrem ACTGRP(*NEW). Při každé aktivaci programu se vytvoří nová. Zaniká při ukončení programu.
4. Převzatá aktivační skupina určená parametrem ACTGRP(*CALLER) znamená, že program nebo servisní program je aktivován ve stejné aktivační skupině jako volající program.

Určení aktivační skupiny

Aktivační skupina je určena programu nebo servisnímu programu při spojování.

CRTPGM

ACTGRP(jméno)	pojmenovaná
ACTGRP(*NEW)	systémová
ACTGRP(*CALLER)	převzatá
ACTGRP(*ENTMOD)	pojmenovaná

- QILE pro jazyky RPG, Cobol, CL a STGMDL(*SNGLVL)
- QILES pro jazyky RPG, Cobol, CL a STGMDL(*TERASPACE)
- *NEW pro jazyk C

CRTSRVPGM

ACTGRP(jméno)	pojmenovaná
ACTGRP(*CALLER)	převzatá

Aktivační skupina se také určuje při kompilaci programu složeného z jednoho modulu v příkazech

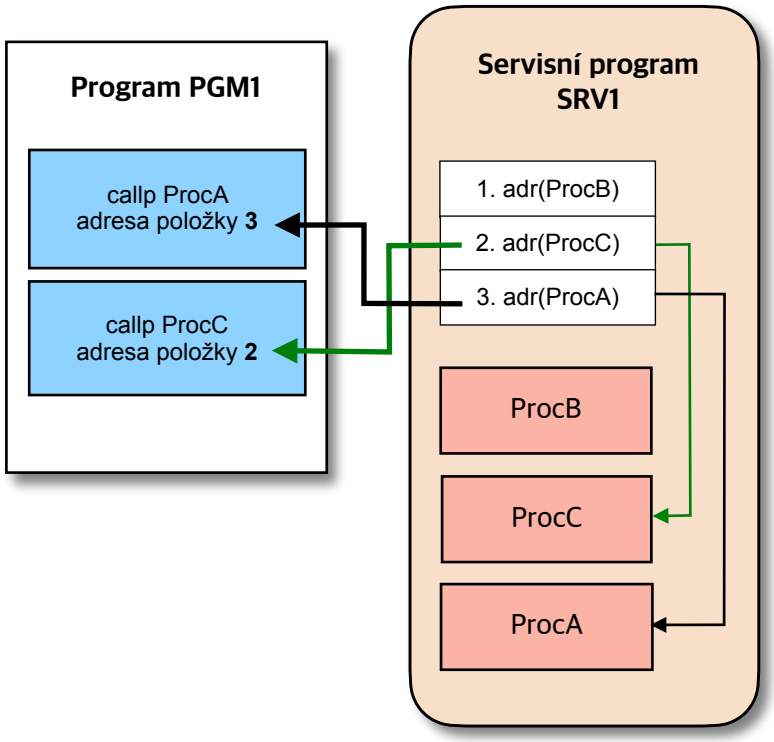
CRTBNDRPG, CRTBNDCBL, CRTBNDCL, CRTBNDC	jako CRTPGM
CRTSQLRPGI OBJECT(*PGM)	QILE / QILES
CRTSQLRPGI OBJECT(*SRVPGM)	*CALLER

Aktivace programu

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

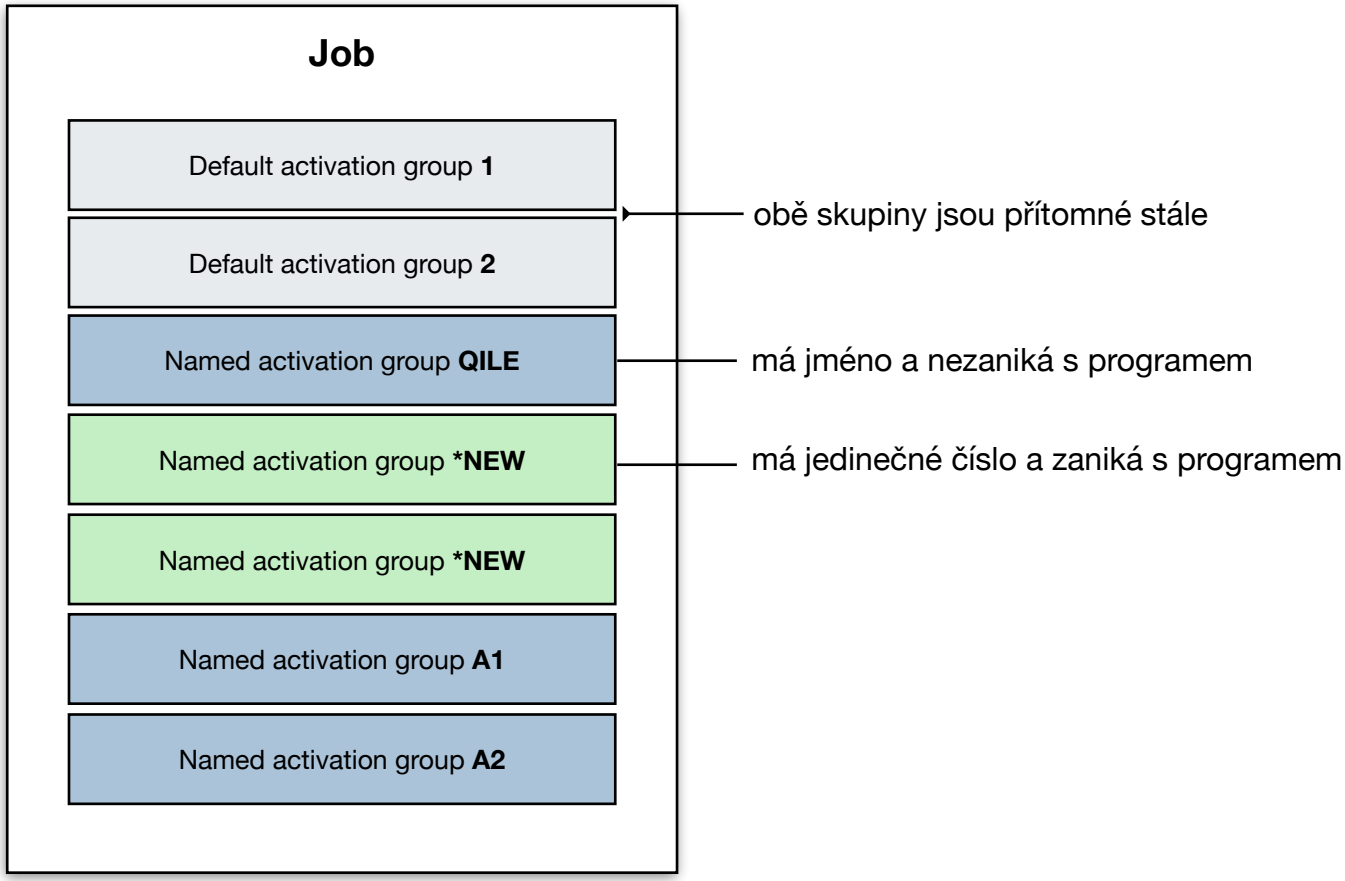
Jestliže aktivační skupina v jobu již existuje, použije ji. Jestliže příslušná aktivační skupina v jobu neexistuje, probíhá aktivace programu takto:

- vytvoří novou aktivační skupinu programu
- vytvoří paměť pro statické proměnné 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.



Statické volání (bound call)

Rozhraní procedury (procedure interface) - definice parametrů a návratové hodnoty.
Prototyp procedury - deklarace parametrů a návratové hodnoty.

Statické volání - volání připojené procedury. Procedura může být zapsána ve stejném nebo jiném (připojeném) modulu.

Statické volání má v ILE jazycích různou formu:

S prototypem

RPG jméno funkce ve výrazu
jméno procedury v příkazu CALLP
C, C++ jméno funkce

Bez prototypu - CL a COBOL neznají prototyp

CL příkaz CALLPRC
COBOL příkaz CALL LINKAGE TYPE PROCEDURE

Dva moduly v programu (RPG)

Modul **OBSAH_PRC** obsahuje 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);  // poloviční součet stran
    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;
```

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;
    // prototyp
    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;
```

Spojení obou modulů do programu

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

Jeden modul v programu (RPG)

Kompilace CRT**BND**RPG **OBSAH_B**

```
CTL-OPT MAIN (OBSAH_B);  // určuje lineární hlavní proceduru
CTL-OPT DFTACTGRP(*NO);  // nebo např. ACTGRP(*NEW)
CTL-OPT BndDir('QC2LE'); // potřebné pro funkce z jazyka C

// lineární hlavní procedura - hlavní program
DCL-PROC OBSAH_B;  // hlavní procedura má podobu podprocedury
    dsply OBSAH ( 3: 4: 5 );  // statické volání podprocedury bez prototypu
END-PROC;

// podprocedura
DCL-PROC  Obsah;  // převádí se do velkých písmen
    // rozhraní podprocedury
    DCL-PI *N   packed(15: 5);        // vracená hodnota
        a    packed(15: 5) value;     // tři parametry
        b    packed(15: 5) value;
        c    packed(15: 5) value;
    END-PI;
    // prototyp standardní funkce sqrt z jazyka C
    dcl-pr Sqrt float(8)  extproc('sqrt');
        *n   float(8)   value;
    end-pr;
    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)); // statické volání funkce Sqrt
END-PROC OBSAH;
```

Dynamické volání v RPG a CL

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

RPG

```
dcl-s a char(10);
dcl-s b char(10);
DCL-PR PGM1 EXTPGM; // prototyp
   *n char(10);
   *n char(10);
END-PR;
...
CALLP  PGM1 (a: b); // volání s prototypem
```

CL

```
CALL  PGM(PGM1)  PARM(&A &B)  /* volání bez prototypu */
```

Program PGM1

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

Dynamické volání v dalších ILE jazycích

C

```
# pragma  linkage ( PGM1, OS )
void PGM1 (char*, char*);     // prototyp
PGM1 ( a, b );                // volání programu jako funkce
```

C++

```
extern "OS" void PGM1 (char*, char*);  // prototyp
PGM1 ( a, b );                // volání programu jako funkce
```

COBOL - bez prototypu

```
CALL PROGRAM "PGM1" USING A B.
```

CMD - bez prototypu

```
CMD         PROMPT('CALL_PGM')
PARM        KWD(A) TYPE(*CHAR) LEN(10) PROMPT('Parametr A')
PARM        KWD(B) TYPE(*CHAR) LEN(10) PROMPT('Parametr B')

CRTCMD CMD(CALL_PGM) PGM(PGM1)    Command Processing Program PGM1
CALL_PGM A(XXX) B(YYY)            Vyvolání příkazu s parametry
```

```
                          CMD_CALL (CALL_PGM)

  Type choices, press Enter.

  Parametr A . . . . . . . . . . . .  XXX          Character value
  Parametr B . . . . . . . . . . . .  YYY          Character value
```

Aktivační skupina programu s jedním modulem

RPG

- s ILE, tedy *s podprocedurami*, je třeba zadat DFTACTGRP(*NO) v kompilačním příkazu nebo ve zdrojovém programu ve specifikaci CTL-OPT.
  Také lze zadat ACTGRP s hodnotou *STGMDL (QILE, QILETS), *NEW, *CALLER, jméno.
- bez ILE, tedy *bez podprocedur*, je použita předvolená aktivační skupina - DFTACTGRP(*YES).

COBOL

- s ILE lze zadat ACTGRP s hodnotou *STGMDL (QILE, QILETS), *NEW, *CALLER, jméno.
- bez ILE je použita předvolená aktivační skupina.

CL

- s ILE je třeba zadat DFTACTGRP(*NO) v kompilačním příkazu.
  Také lze zadat ACTGRP s hodnotou *STGMDL (QILE, QILETS), *NEW, *CALLER, jméno.
- bez ILE je použita předvolená aktivační skupina DFTACTGRP(*YES).

C, C++

- s ILE je použita systémová aktivační skupina *NEW (kompilační příkaz neobsahuje parametr ACTGRP).
- bez ILE je použita předvolená aktivační skupina.

Statické volání v dalších ILE jazycích

CL

Volající modul - bez prototypu. Kompilace CRT**CL**MOD **OBSAH_CALL**

```
DCL         VAR(&A) TYPE(*DEC) LEN(15 5) VALUE(3)
DCL         VAR(&B) TYPE(*DEC) LEN(15 5) VALUE(4)
DCL         VAR(&C) TYPE(*DEC) LEN(15 5) VALUE(5)
DCL         VAR(&PLOCHA) TYPE(*DEC) LEN(15 5)
DCL         VAR(&PLOCHA_C) TYPE(*CHAR) LEN(16)
/*   volání podprocedury 'obsah' */
CALLPRC     PRC('obsah') PARM((&A *BYVAL) (&B *BYVAL) +
              (&C *BYVAL)) RTNVAL(&PLOCHA)
CHGVAR      VAR(&PLOCHA_C) VALUE(&PLOCHA)
SNDPGMMSG   MSG('plocha = ' *BCAT &PLOCHA_C) TOPGMQ(*EXT)
```

Poznámka: Modul v CL může obsahovat jen hlavní proceduru, ale nemůže definovat podproceduru.

C, C++

Volající modul. Kompilace CRT**C**MOD **OBSAH_CALL**

```
#include <decimal.h>
#include <stdio.h>
// prototyp (deklarace)
decimal(15, 5) obsah (decimal(15, 5), decimal(15, 5), decimal(15, 5));
int main () // hlavní procedura
{  // volání funkce obsah
   decimal (15, 5) plocha = obsah (3.0D, 4.0D, 5.0D);
   printf ("plocha = %D(15,5)", plocha);
}
```

Modul s procedurou 'obsah'. Kompilace CRT**C**MOD **OBSAH_PRC**

```
// procedura (funkce)
#include <decimal.h>
#include <math.h>
decimal(15, 5) obsah                    rozhraní procedury
               (decimal(15, 5) a, decimal(15, 5) b, decimal(15, 5) c)
{  decimal(15, 5) s;
   if (a >= b + c || b >= a + c || c >= a + b) {
      return -1.0D;
   }
   s = (a + b + c) / 2;
   return sqrt(s * (s - a) * (s - b) * (s - c));
}
```

Poznámka: Jazyky C, C++ rozlišují velká a malá písmena.

COBOL

Volající modul - bez prototypu. Kompilace CRT**CBL**MOD **OBSAH_CALL**

```
PROCESS OPTIONS NOMONOPRC.
 IDENTIFICATION DIVISION. PROGRAM-ID. OBSAH_CALL.
 WORKING-STORAGE SECTION.
 01  a       PICTURE S9(10)V9(5) VALUE 3.0.
 01  b       PICTURE S9(10)V9(5) VALUE 4.0.
 01  c       PICTURE S9(10)V9(5) VALUE 5.0.
 01  plocha  PICTURE S9(10)V9(5).
 PROCEDURE DIVISION.
*   volání ILE procedury 'obsah'
     CALL LINKAGE TYPE IS PROCEDURE "obsah",
        USING  BY VALUE a BY VALUE b BY VALUE c
        RETURNING plocha.
     DISPLAY "plocha = " plocha.
 END PROGRAM OBSAH_CALL.
```

Modul s procedurou 'obsah'. Kompilace CRT**CBL**MOD **OBSAH_PRC**

```
 PROCESS OPTIONS NOMONOPRC.
*   ILE procedura "obsah"
 IDENTIFICATION DIVISION. PROGRAM-ID. "obsah".
 WORKING-STORAGE SECTION.
 01  s              PACKED-DECIMAL PICTURE S9(10)V9(5).
 01  soucin-dec     PACKED-DECIMAL PICTURE S9(10)V9(5).
 01  plocha-dec     PACKED-DECIMAL PICTURE S9(10)V9(5).
 01  soucin-double COMP-2.
 01  plocha-double COMP-2.

 LINKAGE SECTION.
 01  a        PACKED-DECIMAL PICTURE S9(10)V9(5).
 01  b        PACKED-DECIMAL PICTURE S9(10)V9(5).
 01  c        PACKED-DECIMAL PICTURE S9(10)V9(5).
```

rozhraní podprocedury

```
 PROCEDURE DIVISION USING BY VALUE a BY VALUE b BY VALUE c
        RETURNING plocha-dec.
     IF a >= b + c OR b >= a + c OR c >= a + b
        MOVE -1.0 TO plocha-dec
        GOBACK
     END-IF.
     COMPUTE  s = (a + b + c) / 2.
     COMPUTE  soucin-double = s * (s - a) * (s - b) * (s - c).
     CALL PROCEDURE "sqrt" USING BY VALUE soucin-double
        RETURNING plocha-double.
     MOVE plocha-double TO plocha-dec.
 END PROGRAM "obsah".
```