Interoperabilita Fortranu s C

Programovací jazyky Fortran a C jsou užívané pro řešení numerických problémů. V jednom nebo druhém píšou programy uživatelé a jsou v nich psány také (paralelizační, numerické, vizualizační) knihovny, vzniká tak potřeba funkce psané v C připojovat k fortranskému kódu, nebo naopak. Programátor tak při řešení interoperability dvojjazyčných kódů naráží na témata jako specifikace rozhraní procedur, kompatibilita a způsob předávání argumentů, indexování polí, sdílení globálních proměnných. Kontakt s nefortranským světem zahrnuje i práci s binárními soubory ve fortranských programech. Předložené postupy jsou standardizovány normou Fortranu 2003 a implementovány v aktuálních překladačích; spolupracují nejen překladače téhož výrobce, ale i různé kombinace.

Modul iso_c_binding

Jeden z pěti vnitřních (intrinsic) modulů definovaných ve Fortranu 2003. Připojuje se k programové jednotce pomocí

```
USE iso_c_binding
```

nebo precizněji

```
USE,INTRINSIC :: iso_c_binding, ONLY : c_int, ...
```

a zahrnuje specifikace relevantních konstant, datových typů a rozhraní funkcí, včetně jejich implementace. Obsah modulu se překladač od překladače liší, soubor s modulem nemusí být v souborovém systému patrný.

Interoperabilní datové typy

Standardní datové typy jsou obvykle interoperabilní (tj. lze je popsat v obou jazycích). Vazbu fortranských datových typů na datové typy jazyka C (C99) vytvářejí následující konstanty pro specifikaci fortranského podtypu (KIND):

INTEGER(...)	c_int, c_short, c_long, c_long_long ad. c_signed_char (pro signed i unsigned char) c_size_t, c_intptr_t (pro ukazatele)	typické hodnoty: 4, 2, 4 (win, 32b) nebo 8, 8 1 4 (32b) nebo 8 (64b)
REAL(...)	c_float, c_double, c_long_double	typické hodnoty: 4, 8, 8|10|16|-1 hodnoty c_long_double: gfortran 10, g95 -1, ifort 16, pgfortran 8
COMPLEX(...)	c_float_complex, c_double_complex, c_long_double_complex	hodnoty stejné jako u REAL
LOGICAL(...)	c_bool (pro _Bool)	typická hodnota: 1
CHARACTER(1,...)	c_char (pro char)	typická hodnota: 1

Není-li interoperabilita některého podtypu implementována, nese příslušná konstanta zápornou hodnotu.

Př. Specifikace interoperabilních proměnných celočíselného, reálného, logického a znakového typu (též 1const.f90)

```
INTEGER(c_int) :: i ; REAL(c_float) :: x ; COMPLEX(c_float_complex) :: c ; LOGICAL(c_bool) :: b
CHARACTER(LEN=1,KIND=c_char) :: ch                    ! délka LEN=1 je pro interoperabilitu povinná
int i, float x, float _Complex c, _Bool b, char ch;
```

Požadavek interoperability odvozených datových typů s datovým typem struct jazyka C se vyjadřuje pomocí atributu BIND(C). Jména typů i položek se mohou na straně Fortranu a C lišit. Interoperabilitu vylučuje mj. výskyt položek s fortranskými atributy ALLOCATABLE a POINTER, C bitových a flexibilních polí a typových rozšíření pro objektově orientované programování. Proměnné neinteroperabilních odvozených typů lze sdílet pomocí funkcí c_loc a c_f_pointer z iso_c_binding pro vytvoření C a fortranských ukazatelů (ukázka v Metcalf et al., 2011, obr. 12.5).

Př. Specifikace interoperabilních odvozených typů a příslušných proměnných (též 2vars.f90)

```
TYPE,BIND(C) :: typeF ; INTEGER(c_int) i ; REAL(c_float) x ; END TYPE ; TYPE(typeF) s
typedef struct {int m; float r;} typeC; typeC s;
```

Př. Specifikace odvozeného typu c_ptr interoperabilního s obecným (tedy libovolným) C ukazatelem (void *)

```
TYPE,BIND(C) :: c_ptr ; PRIVATE ; INTEGER(c_intptr_t) ptr ; END TYPE          (podle ifort)
```

Specifikace odvozeného typu c_funptr pro ukazatel na funkce bývá totožná.

Výčtový typ (enumeration) je interoperabilním datovým typem sdružujícím několik pojmenovaných INTEGER konstant. Nejsou-li uvedeny inicializační výrazy, přiřazují se od poslední přiřazené konstanty po řadě celá čísla (0 do první).

Př. Specifikace interoperabilních výčtových typů (atribut BIND(C) je povinný) a příslušných proměnných

```
ENUM,BIND(C) ; ENUMERATOR :: false=0,true=1 ; END ENUM ; INTEGER(KIND(false)) :: ibool
typedef enum {false=0,true=1} tBoolean; tBoolean ibool;
```

Modul iso_c_binding obvykle definuje i znakové konstanty pro speciální znaky: c_null_char (pro '\0'), c_new_line ('\n'), c_carriage_return ('\r') ad. a ukazatelové konstanty c_null_ptr a c_null_funptr (pro hodnotu NULL), např.

```
CHARACTER(1,c_char),PARAMETER :: c_null_char=achar(0), c_alert=achar(7), c_new_line=achar(10)
TYPE(c_ptr),PARAMETER :: c_null_ptr=c_ptr(0) ; TYPE(c_funptr),PARAMETER :: c_null_funptr=c_funptr(0)
```

Interoperabilita proměnných a polí

Interoperabilní proměnné (rozuměj: formální argumenty a modulové proměnné) musí mít interoperabilní datový typ a nesmí mít atributy ALLOCATABLE a POINTER.
Fortranské jednorozměrné pole explicitního tvaru může být interoperabilní s C jednorozměrným polem téže velikosti, fortranské pole předpokládané velikosti, např. fa(*), může interoperovat s C polem nespecifikované velikosti, např. ca[], pole předpokládaného tvaru (assumed-shape arrays) interoperabilní být nemůže.
Fortranská pole jsou implicitně indexována od 1 (explicitně však jakkoliv), zatímco C pole vždy od 0. V této souvislosti je užitečný obrat v C pro indexování pole od 1: po deklaraci a přiřazení `float a[n],*aa; aa=a-1;` lze pracovat s prvky aa[1] až aa[n]. Obdobně, C funkci `f(float a[],int n)` pracující s prvky a[1] až a[n] lze v rámci C programu předat nativní pole indexované od nuly voláním `f(a-1,n)`.
Dvourozměrná pole (matice) jsou v paměti ukládána ve Fortranu po sloupcích, v C po řádcích; pro vícerozměrná pole platí, že pro sekvenční průchod polem v paměti je třeba měnit nejrychleji ve Fortranu první index, v C poslední index. Odtud plyne, že vícerozměrná pole Fortranu a C jsou interoperabilní, odpovídají-li si jejich deklarace pozpátku, např. `REAL(c_float) :: fa(n1,n2), fb(n1,n2,n3)` je interoperabilní s `float ca[n2][n1], cb[n3,n2,n1];` a odpovídaly by si např. prvky fa(1,1) s ca[0][0] a fa(n1,n2) s ca[n2-1][n1-1]. V případě dvou dimenzí je tedy fortranská matice interoperabilní s C maticí deklarována a zpřístupňována jako transponovaná. (Příklad: 3arrays.f90.)
Fortranské řetězce i pole znaků jsou interoperabilní s C řetězci, a tím i poli znaků, např. `CHARACTER(10,c_char) ch` interoperuje s `char *ch` i `char ch[]`. C řetězce jsou ukončeny C znakem '\0' neboli fortranskou konstantou c_null_char. Délku řetězce vytvořeného v C lze ve Fortranu zjistit právě nalezením znaku c_null_char: `L=index(ch,c_null_char)-1 ; print *,ch(:L)`. (Fortranská funkce TRIM(ch) je neúčinná, odřezává zprava mezery, nikoliv smetí.)

Interoperabilita procedur a jejich volání

Interoperabilní procedury lze vytvářet jak v C, tak ve Fortranu, a lze je volat z obou částí dvojjazyčného programu. Jejich interoperabilita se vyznačuje fortranským atributem BIND(C) s volitelným uvedením názvu procedury v C (binding label), `BIND(C,NAME='BindingLabel')`. Názvy v C jsou citlivé na velikost písmen, fortranské názvy nikoliv. Bez uvedení C jména v LABEL je implicitně bráno fortranské jméno psané malými písmeny. Volat z Fortranu lze C funkce vytvořené uživatelem i C funkce ze standardních knihoven, z C lze volat fortranské procedury s atributem BIND, jak samostatné, tak modulové. Fortranské podprogramy (SUBROUTINE) jsou interoperabilní s C funkcemi s návratovou hodnotou void.

Př. Specifikace rozhraní funkce ze standardní knihovny C pro možné použití z Fortranu

```
C:      int atoi(const char *s);
Fortran:INTERFACE ; INTEGER(c_int) FUNCTION atoi(s) BIND(C)
        IMPORT c_char,c_int ; CHARACTER(1,c_char),INTENT(IN) :: s(*)
        END FUNCTION ; END INTERFACE
```

Př. Specifikace rozhraní volající standardní fortranskou funkci pro možné použití z C

```
C:      int CountPositive(float a[],const int n);
Fortran:FUNCTION CountPositive(a,n) BIND(C,NAME='CountPositive')
        IMPORT c_int,c_float ; INTEGER(c_int) CountPos ; REAL(c_float),INTENT(IN) :: a(*)
        INTEGER(c_int),VALUE,INTENT(IN) :: n
        CountPos=count(a(1:n)>0.) ; END FUNCTION
```

Při asociaci skutečných a formálních argumentů procedur je ve Fortranu preferováno předání odkazem (předání adresy, ukazatele) a pokud nelze, předání hodnotou (jednosměrné kopírování hodnoty do lokální kopie v proceduře); v C se argumenty předávají hodnotou nebo se explicitně posílají ukazatele. Pro zaručené předání argumentu hodnotou poskytuje Fortran atribut VALUE pro formální argument procedury. Rozhraní volané procedury pak musí být ve volající jednotce explicitní; formální argument s atributem VALUE nesmí mít atributy ALLOCATABLE, POINTER a INTENT jiný než IN. Skutečný argument alokovatelným polem nebo ukazatelem být může.
Př. Argument procedury s atributem VALUE (výpis: 1. 2.; bez atributu VALUE: 2. 2.; též 4args.f90)

```
x=1.; y=f(x); print *,x,y; CONTAINS; FUNCTION f(x); REAL,VALUE :: x; x=x+x; f=x; END FUNCTION; END
```

Pole nelze předat s atributem VALUE. Při práci se sekcemi polí nebo ukazatelovými poli může dojít ke kopírování hodnot, ovšem obousměrnému, s účelem předat do volané procedury pole uložené v paměti spojitě.

Blok rozhraní nepřejímá automaticky jména z jednotky, v které se nachází. K tomu slouží specifikace IMPORT, která může přenést do bloku rozhraní buď všechna jména z hostitele nebo jen jména výslovně uvedená.
Př. USE iso_c_binding ; INTERFACE ; SUBROUTINE s(n) ; IMPORT c_int ; INTEGER(c_int) n ; ... END INTERFACE
Argumenty s C atributem const je vhodné deklarovat ve Fortranu s atributem INTENT(IN). Argumenty deklarované v C jako pole, a[], nebo jako řetězce, char *, se na straně Fortranu deklarují jako pole předpokládané velikosti, a(*). Generické argumenty deklarované v C jako obecné ukazatele, void *, je nutné ve Fortranu deklarovat jako TYPE(c_ptr) a před dalším použitím je konvertovat na fortranský ukazatel pomocí funkce c_f_pointer nebo přetypovat funkcí TRANSFER.
Př. Specifikace rozhraní C funkce sscanf a její použití pro přiřazení do INTEGER a REAL proměnných. Norma neupřesňuje interoperabilitu C funkcí s proměnným počtem argumentů (např. právě sscanf), překladače však uvedený příklad přeloží (gfortran v Linuxu ne). Překladač g95 nepřeloží příkaz x=TRANSFER(p,x), přeloží však sekvenci i=TRANSFER(p,i) ; x=TRANSFER(i,x). (Příklady: 5callC.f90 a 6callF.c.)

```
USE iso_c_binding ; INTERFACE ; SUBROUTINE sscanf(s,format,p) BIND(C) ; IMPORT c_char,c_ptr
CHARACTER(1,c_char),INTENT(IN) :: s(*),format(*)
TYPE(c_ptr),INTENT(OUT) :: p
END SUBROUTINE ; END INTERFACE
CHARACTER(100,c_char) :: ch='1' ; TYPE(c_ptr) p
call sscanf(ch,"%d",p) ; i=transfer(p,i) ; call sscanf(ch,"%f",p) ; x=transfer(p,x)
```

Tab. Korespondence specifikací C funkcí a fortranských bloků rozhraní

Návratové hodnoty	
void s(...);	INTERFACE; SUBROUTINE s(...); IMPORT; ...; END SUBROUTINE; END INTERFACE
int f(...);	INTERFACE; INTEGER(c_int) FUNCTION f(...); IMPORT; ...; END FUNCTION; END INTERFACE
char *f(...);	INTERFACE; TYPE(c_ptr) FUNCTION f(...); IMPORT; ...

C řetězec musí být na fortranské straně předán do C ukazatele a z něj konvertován na fortranský ukazatel na řetězec, TYPE(c_ptr) pc ; CHARACTER(len,c_char),POINTER :: pf ; pc=f(...) ; CALL c_f_pointer(pc,pf). Jak je uvedeno výše, dynamická délka C řetězce je dána polohou znaku '\0' a např. pro výpis pf je třeba print *,pf(:index(pf,c_null_char)-1). Tento obrat aktuálně nepřeloží pgfortran.

Argumenty předávané hodnotou	
int i,float x	INTEGER(c_int),VALUE,INTENT(IN) :: i ; REAL(c_float),VALUE,INTENT(IN) :: x
char ch	CHARACTER(1,c_char),VALUE,INTENT(IN) :: ch

S předáním znaku hodnotou (char ch) mají aktuálně potíž ifort a pgfortran; ifort nepředá správně jako skutečný argument proměnnou, CALL s(ch), předá však konstantu, CALL s('A'), nebo výraz, CALL s((ch)); v pgfortranu je třeba na fortranské straně deklarovat INTEGER(c_signed_char),VALUE,INTENT(IN) a na místě skutečného argumentu uvést TRANSFER(ch,0_1). (Příklad: 7cula.f90.)

Argumenty předávané odkazem	
int *i,float *x	INTEGER(c_int) i ; REAL(c_float) x
int ia[],float xa[]	INTEGER(c_int) ia(*) ; REAL(c_float) xa(*)
char *ch,char ch[]	CHARACTER(1,c_char) ch(*)

Fortranské popisy mohou obsahovat, podle situace, i atribut INTENT (IN, OUT nebo INOUT).

Interoperabilita globálních proměnných

Interoperabilní mohou být nejen argumenty procedur, ale i modulové proměnné a také modulové COMMON bloky. Jsou-li specifikovány s atributem BIND(C), jsou interoperabilní s globálními proměnnými v C. COMMON bloky interoperují s proměnnými typu struct. Nemodulové proměnné interoperovat s globálními proměnnými nemohou.
Př. Specifikace interoperabilní modulové proměnné a interoperabilního COMMON bloku (též 3arrays.f90)

```
INTEGER(c_int),BIND(C,NAME='cvar') :: fvar
REAL(c_float) a,b ; COMMON /cb/ a,b ; BIND(C,NAME='s') /cb/
```

Interoperující globální proměnné na straně C

```
int cvar;
typedef struct {float a; float b;} typeC; typeC s;
```

Binární soubory

Pro zápis binárních souborů jsou (nejen ve Fortranu) vhodné bezformátové proudy (streams). V takto vytvořených souborech (form='unformatted', access='stream') se na rozdíl od bezformátových souborů (form='unformatted', access='sequential' nebo 'direct') neobjevují nežádoucí znaky navíc. Do existujícího souboru proud zapíše požadovaný

počet znaků a zbytek souboru nezmění; je-li úmyslem existující soubor přepsat, je třeba to uvést explicitně (status='replace'). Rychlost čtení i zápisu souborů je s různými překladači výrazně jiná, a to i v závislosti na operačním systému. Např. ve Windows je g95 pomalý při čtení a gfortran při zápisu a g95 je rychlý při zápisu a gfortran při čtení; na Linuxu pracuje gfortran se soubory velmi rychle, pgfortran rovněž a ifort přijatelně rychle.
Př. Příprava binárního souboru s ASCII hlavičkou a binárními daty (se změnou endianity – pořadí bytů)

```
OPEN (id,file=trim(filename),form='unformatted',access='stream',status='replace')
do i=1,nheader; write (id) trim(header_line(i))//new_line(' '); enddo      ! hlavička s ASCII řádky
do i=1,npoints; write (id) swapEndian(s(i)); enddo                         ! binární data se změnou pořadí bytů
CLOSE (id)
FUNCTION swapEndian(x) RESULT (result)
REAL(4),INTENT(IN) :: x
CHARACTER(4) :: result,z; CHARACTER(1),DIMENSION(4) :: za; EQUIVALENCE (z,za)
z=transfer(real(x,4),z); za(1:4)=za(4:1:-1); result=z
END FUNCTION
```

Překladače

Přiložené příklady byly testovány na dvojicích překladačů GNU gfortran a gcc, g95 a gcc (gcc není součástí balíčku g95), Intel ifort a icc a Portland Group pgfortran a pgcc. Testy byly většinou úspěšné i s překladači z různých dvojic, na Linuxu vždy, ve Windows nespolupracoval pgfortran s gcc. Překladače gfortran a pgfortran zprostředkují i překlad zdrojových textů v C, př. gfortran p1.f90 p2.c, jinak jsou třeba dva kroky, př. icc -c p2.c a ifort p1.f90 p2.o. Pro spojení C kódu s fortranskou procedurou nelze užít pgcc, ale pgfortran s přepínačem, pgfortran -Mnomain main.c sub.f90. Použité verze překladačů: gfortran 4.4.3, g95 0.93, ifort 11.1 a pgfortran 12.2.

Knihovny

Mnohé knihovny poskytují rozhraní pro oba jazyky, některé však jen pro C a interoperabilitu s Fortranem řeší buď pomocí modulů s bloky rozhraní nebo nijak. Přeložené C knihovny s rozhraním pro starší Fortran (bez iso_c_binding) mohou spolupracovat jen s některými překladači a jen v některém z operačních systémů, linkování nativních C knihoven k Fortranu pomocí iso_c_binding se zdá být robustnější cestou. Při linkování přeložených knihoven uživatel naráží také na různé potřeby různých překladačů (přepínač překladačů -L pro cestu ke knihovnám, proměnné prostředí ve Windows: PATH, LIB aj., v Linuxu: LIB, LIBRARY_PATH, LD_LIBRARY_PATH aj., jména knihoven: pgfortran ve Windows může vyžadovat přejmenování *.lib na lib*.lib, atd.) a samozřejmě kompatibilitu, např. 32- a 64bitových režimů překladačů a verzí knihoven.

Paralelizační knihovny

Knihovna MPI je psána v C a poskytuje rozhraní pro oba jazyky, přenos argumentů fortranských volání do C zajišťuje sama knihovna. K ruce jsou dva typy fortranského vázání (binding): vázání pro Fortran 77 definované v MPI-1, nezajišťující kontrolu shody skutečných a formálních argumentů, a vázání pro Fortran 90 přidané v MPI-2, typové kontroly zajišťující. Náklady typové kontroly jsou však neúměrné vzhledem k potřebě uvádět rozhraní MPI procedur pro všechny očekávatelné datové typy (a podtypy a počty dimenzí) posílaných zpráv (buferů). Ošetření typové kontroly fortranských kódů tak může řešit vázání pro C (MPI pro C je ovšem tvořeno funkcemi, nikoliv podprogramy jako MPI pro Fortran).
Př.

```
INTERFACE ; FUNCTION MPI_Send(buf,count,...) BIND(C,name='MPI_Send') ; IMPORT c_ptr,c_int
TYPE(c_ptr),INTENT(IN) :: buf ; INTEGER(c_int),VALUE,INTENT(IN) :: count ; ...
INTEGER(c_int) :: MPI_Send ; END INTERFACE
```

OpenMP není knihovna, ale součást samotného překladače a interoperabilita tak nehraje roli.

Numerické knihovny

Knihovny BLAS a LAPACK vznikly jako fortranské knihovny, jejich volání z C je dokumentováno; C programátor může respektovat fortranské řazení prvků matic po sloupcích nebo má možnost nastavit příznak pro transponování matic. Komerčně udržované knihovny výrobců hardwaru MKL (Intel Math Kernel Library) a ACML (AMD Core Math Library), stejně jako vůdčí numerické knihovny NAG a IMSL, poskytují rozhraní pro oba jazyky. Knihovna CULA Tools s implementací LAPACKu pro GPU je vzorovou ukázkou moderní knihovny se zastaralým řešením interoperability: verzí pro Fortran je poskytnuto několik a ne všechny se zdají být funkční, linkování verzí pro C k Fortranu pracuje, na uživateli však je vytvoření bloků rozhraní. (Příklad: 7cula.f90)

Vizualizační knihovny

Ve světě vizualizace je Fortran neznámým jazykem a potřebné C knihovny si fortranský uživatel váže vlastnoručně.
Ukázky: LodePNG, FreeImage

Odkazy

Metcalf M., Reid J., Cohen M., Modern Fortran Explained, Oxford Science, 2011
Press W. H., Teukolsky S. A., Vetterling W. T., Flannery B. P., Numerical Recipes in C: The Art of Scientific Computing, Second Edition, Cambridge University Press, 1992, www.nrbook.com/a/bookcpdf.php
Dokumentace normy a překladačů
Fortran 2003 Committee Draft: j3-fortran.org/doc/2003_Committee_Draft/04-007.pdf
Intel: software.intel.com/sites/products/documentation/hpc/compilerpro/en-us/fortran/lin/compiler_f/index.htm
GNU: gcc.gnu.org/onlinedocs

L. H., 22. 2. 2012