# Unixové funkce v programovacích jazycích ILE

Vladimír Župka, 2021

Úvod ..... 2
Jazyk C ..... 3
Poznámka k jazyku C++ ..... 4
Jazyk RPG ..... 5
Jazyk Cobol ..... 6
Jazyk CL ..... 7

# Úvod

Prostředí ILE dovoluje kromě jiného využít funkce unixových systémů (nebo i standardní funkce jazyka C), a to nejen v jazyku C nebo C++, ale i v jazycích RPG, Cobol a CL. V tomto článku se podíváme na to, jakým způsobem se to dělá.

Uvádíme co možná nejjednodušší příklad: použijeme funkci `sleep()`, která pozastaví výpočet na zadaný počet sekund. Účelem příkladu je pouze demonstrovat prostředky, které se k tomu v jednotlivých jazycích používají, nikoliv poskytnout praktickou aplikaci.

# Jazyk C

Funkce časového zpoždění `sleep()` v jazyku C je popsána v manuálu *Unix - Type APIs*, podkapitole "*Signal APIs*", kde najdeme vysvětlení této funkce zároveň s jejím prototypem. *Prototyp* funkce se v jazyku C také nazývá *deklarace* funkce.

Prototyp funkce `sleep()` je v dokumentaci popsán takto:

```
unsigned int sleep( unsigned int seconds );
```

Říká, že funkce `sleep()` přijímá jako argument číslo typu `unsigned int` a vrací hodnotu stejného typu.[1] Typ `unsigned int` představuje 4bajtové binární celé číslo bez znaménka. (Typ `int` je stejný, ale se znaménkem.)

Argument v závorce zadává čas v počtu sekund, po nějž je výpočet pozastaven. Slovo `seconds` je jen informační; může, ale nemusí být zapsáno nebo můžeme zvolit jiné slovo.

Vrácená hodnota bude `0`, jestliže funkce proběhne dobře, nebo bude `-1`, dojde-li během provádění funkce k chybě. Test na chybu je zaveden proto, že funkce může být sdílená několika procesy.

Následující příklad pozastaví výpočet o 5 sekund a poté vypíše vrácenou hodnotu, tedy návratový kód 0 (protože funkce se nejspíš provede správně).

V první variantě C programu je zařazen povel `#include <unistd.h>`, který začlení prototypy standardních unixových funkcí, mezi nimi také prototyp funkce `sleep()`. Tento povel je při kompilaci nasměrován na knihovnu `QSYSINC`, soubor `H`, člen `UNISTD`, kde je kromě jiných uveden zmíněný prototyp, který se vkopíruje do programu. Prototyp se použije při kompilaci programu ke statické kontrole volání funkce `sleep()`.

Zdrojový kód programu má typ `C` a kompiluje se příkazem `CRTBNDC` (volbou 14 v PDM). Předvolený zdrojový soubor je `QCSRC`. Po kompilaci, v okamžitě navazující fázi spojování, je programu automaticky přiřazena aktivační skupina `*NEW`. Kdybychom chtěli zadat jinou aktivační skupinu, museli bychom nejprve vytvořit modul příkazem `CRTCMOD` (volba 15 v PDM) a pak vytvořit program příkazem `CRTPGM` se zadáním zvolené aktivační skupiny.

```
#include <stdio.h>
#include <unistd.h>  // zde je obsažen prototyp funkce sleep()
int main(void)
{
   unsigned int sekundy = 5, rc;
   rc = sleep(sekundy);
   printf("Return code: %d", rc);
}
```

[1] V jazyku C se používá název argument, což odpovídá názvu parametr v některých jiných jazycích.

Ve druhé variantě napíšeme místo příkazu `#include` rovnou prototyp (beze slova `seconds` nebo i s ním):

```
#include <stdio.h>
unsigned int sleep( unsigned int );  // prototyp funkce sleep()
int main(void)
{
   unsigned int sekundy = 5, rc;
   rc = sleep(sekundy);
   printf("Return code: %d", rc);
}
```

V obou variantách program vypíše:

```
Return code: 0
Press ENTER to end terminal session.
```

### Poznámka k jazyku C++

V jazyku C++ by byl zdrojový text označen typem `CPP` a kompilace by se prováděla příkazem `CRTBNDCPP` (volbou 14 v PDM). Předvolený zdrojový soubor je `QCPPSRC`. Z obou příkladů funguje beze změny jen první varianta s příkazem `#include`, ve druhé variantě hlásí spojovací program chybu, že nenalezl k deklaraci (importu) `sleep__FUi` příslušnou definici (export). Dá se to napravit tak, že před prototyp předřadíme ještě symbol `QBFC_EXTERN`:

```
QBFC_EXTERN unsigned int sleep( unsigned int );
```

To zjistíme z protokolu o překladu první varianty, když v příkazu `CRTBNDCPP` zadáme parametry `OUTPUT( *PRINT )` a `OPTIONS( *SHOWINC )`.

# Jazyk RPG

Zdrojový kód programu má typ RPGLE a kompiluje se příkazem CRTBNDRPG (volbou 14 v PDM). Předvolený zdrojový soubor je QRPGLESRC.

```
**free
// aktivační skupina QILE
ctl-opt DftActGrp(*no);

// prototyp funkce sleep()
dcl-pr  sleep uns(10) extproc('sleep');
    *n  uns(10) value;
end-pr;

dcl-s  sekundy uns(10) inz(5);
dcl-s  rc      int(10);

rc = sleep(sekundy);

dsply ('Return code: ' + %char(rc));
return;
```

Program vypíše:

```
DSPLY  Return code: 0
```

Jak vidíme, v jazyku RPG již není zápis zdrojového programu tak jednoduchý jako v jazyku C. Abychom mohli použít vlastnosti ILE (volání funkce), musíme v programu uvést několik dodatečných zápisů:

V příkazu CTL-OPT je zadán parametr DftActGrp(*no), který říká, že program bude běžet v aktivační skupině QILE. Jinou aktivační skupinu můžeme zadat třemi způsoby:

- do příkazu CTL-OPT zapíšeme parametr ActGrp se zvolenou hodnotou, nebo
- při kompilaci zadáme v příkazu CRTBNDRPG parametr ActGrp, nebo
- vytvoříme modul příkazem CRTRPGMOD (volba 15 v PDM) a pak vytvoříme program příkazem CRTPGM s parametrem ActGrp.

Oba parametry lze alternativně zadat už při zahájení kompilace příkazem CRTBNDRPG (volbou 14 v PDM). Jejich zadání ve zdrojovém programu je však většinou praktičtější.

Dále musíme zapsat *prototyp funkce* sleep() tak, aby přesně odpovídal popisu z jazyka C. Prototyp se použije při kompilaci programu ke statické kontrole volání funkce sleep().

První řádek prototypu uvádí jméno sleep, které je po kompilaci převedeno na velká písmena: SLEEP. Proto je v parametru extproc zadáno jméno funkce v doslovném znění (malými písmeny a uzavřené v apostrofech). V jazyku C a v systémech typu Unix se totiž striktně rozlišují velká a malá písmena. Kromě jména funkce je v řádku uveden typ návratové hodnoty uns(10), což znamená celé číslo bez znaménka (přesně odpovídá typu unsigned int jazyka C). Jde o 4bajtové binární číslo, které obsáhne maximálně 10 dekadických číslic, když se převede do dekadického tvaru.

Druhý řádek prototypu uvádí parametr funkce sleep jako celé číslo bez znaménka uns(10). Slovo value znamená, že parametr se funkci předává hodnotou. V jazyku C se tak totiž předávají parametry jednoduchých typů.

# Jazyk Cobol

Zdrojový kód programu má typ CBLLE a kompiluje se příkazem CRTBNDCBL (volbou 14 v PDM). Předvolený zdrojový soubor je QCBLLESRC. Při spojování je programu přiřazena aktivační skupina QILE, aniž ji zadáme. Jinou aktivační skupinu můžeme zadat buď při kompilaci parametrem ACTGRP nebo tak, že nejprve vytvoříme modul příkazem CRTCBL–MOD (volba 15 v PDM) a pak vytvoříme program příkazem CRTPGM se zadáním zvolené aktivační skupiny.

```
*---Kompilovat s OPTION(*NOMONOPRC) nebo
 PROCESS OPTIONS NOMONOPRC.
*---(nepřevádět jména procedur do velkých písmen)
 WORKING-STORAGE SECTION.
 01  sekundy      PIC   S9(10)  BINARY VALUE 5.
 01  rc           PIC   S9(10)  BINARY.
 PROCEDURE DIVISION.
     CALL PROCEDURE "sleep", USING BY VALUE sekundy, RETURNING rc.
     DISPLAY "Return code: " rc.
```

Program vypíše:

```
Return code: 0000000000
```

Chceme-li v jazyku Cobol využít unixové funkce, musíme použít volání procedury. Jde o příkaz CALL PROCEDURE, v němž zadáme jméno funkce, parametry a návratovou hodnotu. Musíme ovšem zařídit, aby jméno funkce vystupovalo v doslovném znění, tj. s malými písmeny. Toho dosáhneme zápisem příkazu PROCESS OPTIONS NOMONOPRC na začátku zdrojového programu nebo při kompilaci zadáním volby *NOMONOPRC v parametru OP–TION. Tento parametr způsobí, že se jméno procedury nepřevede do velkých písmen, k čemuž by jinak došlo.[2]

V jazyku Cobol se žádný prototyp neuvádí. Popis parametru a návratové hodnoty však musí odpovídat prototypu z jazyka C. V našem příkladu jde o 4bajtové binární proměnné s maximálním počtem 10 dekadických číslic. Jsou to sice čísla se znaménkem, ale při volání funkce sleep() vyhovují.

Parametr SEKUNDY je zde předáván hodnotou (BY VALUE), protože to vyžaduje funkce sleep(). V jazyku C se totiž jednoduché proměnné vždy předávají přímo jako hodnota, na rozdíl od jazyka Cobol, kde se normálně předávají prostřednictvím adresy (reference).

Číslo 5 typu BINARY je v paměti zobrazeno jako X'00000005', tedy pětka v posledním bajtu 4bajtové paměťové oblasti, což je přesně to, co potřebuje funkce sleep(), aby zdržela výpočet na 5 vteřin.

Poněvadž chybí prototyp, nekontroluje se volání při spojování programu, a proto se nesprávně zadané typy parametrů nebo návratové hodnoty *projeví až při výpočtu*! Zadáme-li například u proměnné SEKUNDY typ COMP místo BINARY, bude výpočet pozastaven na 95 sekund místo 5 sekund. V podobě COMP je číslo zobrazeno jako X'000000005F', tedy 5bajtové dekadické číslo 5 v pakovaném tvaru. Funkce sleep() si z toho vezme (kupodivu) poslední 4 bajty, které představují číslo 95 (5 x 16 + 15) a skutečně zastaví výpočet na 95 sekund.

Z toho je vidět, že v jazyku Cobol je nutné velmi pečlivě dbát na dodržení správného *typu* parametrů, jinak budou výsledky nepředvídatelné. Na druhé straně není nutné naprosto

---

[2] Pro funkce jazyka C a unixové funkce je obecně třeba zadat v příkazu CRTBNDCBL parametr BndDir(QC2LE), aby se daná funkce mohla připojit. Zajímavé je, že pro funkci sleep() to nutné není.

přesně dodržet prototyp z jazyka C. V našem případě bychom u typu `BINARY` měli správně zadat obraz `S9(10)`, ale stačí zadat `S9(9)` nebo třeba jen `S9(1)`. Kompilátor si proměnnou upraví na správnou velikost. Situace zde není tak jednoznačná jako v jazycích C a RPG.

# Jazyk CL

V jazyku CL lze použít jen velmi omezený výběr funkcí z jazyka C a z unixu. Je to dáno omezeným počtem datových typů. Nelze například použít čísla v pohyblivé řádové čárce (float, double), pole nebo struktury, kteréžto typy jsou v C a unixu hojně využívány.

Zdrojový kód programu má typ `CLLE` a kompiluje se příkazem CRTBNDCL (volbou 14 v PDM). Po při spojování je programu přiřazena aktivační skupina `*DFTACTGRP`, aniž ji zadáme. Kdybychom chtěli zadat jinou aktivační skupinu, museli bychom nejprve vytvořit modul příkazem `CRTCLMOD` (volba 15 v PDM) a pak vytvořit program příkazem CRTPGM se zadáním zvolené aktivační skupiny.

```
DCL         VAR(&SEKUNDY) TYPE(*UINT) VALUE(5)
DCL         VAR(&RC_UINT) TYPE(*UINT)
DCL         VAR(&RC_CHAR) TYPE(*CHAR) LEN(1)
CALLPRC     PRC('sleep') PARM((&SEKUNDY *BYVAL)) +
              RTNVAL(&RC_UINT)
CHGVAR      VAR(&RC_CHAR) VALUE(&RC_UINT)
SNDPGMMSG     MSG('Return code: ' *BCAT &RC_CHAR) +
              TOUSR(*REQUESTER)
```

Volání funkce provádíme příkazem `CALLPRC` (Call Procedure). Jméno funkce musíme uvést malými písmeny v apostrofech, parametr `&SEKUNDY` předáme hodnotou (`*BYVAL`) a návratovou hodnotu zadáme v parametru `RTNVAL`.

V jazyku CL rovněž neuvádíme žádný prototyp. Ani zde tedy neprobíhá při spojování programu žádná (statická) kontrola volání, nesprávně zadané parametry nebo návratová proměnná se projeví *až při výpočtu*!