# OpenMP (Open Specification for Multi Processing)

OpenMP je rozhraní (application programming interface, API) pro (explicitní) programování paralelních aplikací na (jednom) počítači s více procesory (jádry) a sdílenou pamětí. Realizuje vláknovou paralelizaci (multithreading), totiž jeden spuštěný hlavní proces (master thread) na vstupu do paralelní oblasti (parallel region) vytváří skupinu vláken (team), která se dělí o práci a na konci paralelní oblasti po synchronizaci zaniká (fork-join model); dál pokračuje samotný hlavní proces. Vlákna se mohou podělit jak o provádění týchž instrukcí na různých datech (data parallelism), tak i o nezávislé větve programu (functional parallelism). O softwarová vlákna se dělí hardwarové procesory; je obvyklé, že počet vláken odpovídá počtu (volných) procesorů.
OpenMP programování spočívá především v doplnění (vhodně připraveného) sekvenčního zdrojového kódu o direktivy (interpretovatelné komentáře), které navedou překladač k vytvoření vláken a rozdělení práce mezi ně; někdy vyvstává potřeba přidat do zdrojového kódu i volání dotazovacích a konfiguračních funkcí a procedur; typické je nastavování některých vlastností spouštěných programů z vnějšku pomocí proměnných prostředí. Je možno dosáhnout stavu, kdy zdrojový kód OpenMP programu lze bez úprav přeložit jak pro sekvenční, tak pro paralelní běh; specifikace umožňuje (nezaručuje) získání shodných výsledků v obou případech. Úkolem programátora je nalézt paralelizovatelné oblasti sekvenčního kódu, správně rozvrhnout proměnné v paralelní oblasti na sdílené (originály přístupné všem vláknům) a soukromé (kopie rozmnožené pro každé vlákno) a zajistit synchronizaci vláken i přístupu do paměti; jinak se mohou vyskytnout datové dostihy (data race), nejčastější chyba v OpenMP programech.
Specifikace OpenMP (dostupná v PDF na www.openmp.org) je formulována pro jazyky C/C++ a Fortran, jednotlivé verze se datují roky 1.0 1997, 2.0 2000, 2.5 2005, 3.0 2008 a 3.1 2011, překladače se specifikacemi drží krok. OpenMP ve Fortranu implementují např. Intel ifort, GNU gfortran, Portland pgfortran aj., nikoliv však g95. Předběžně publikovaná rozšíření v návrhu specifikace 4.0 (2013) zatím nezahrnují např. představy pro použití OpenMP k programování výpočetních akcelerátorů; to bylo mezitím realizováno rozhraním OpenACC, implementovaným pro Nvidia GPU např. v Portland překladačích. Podkladem a zároveň nízkoúrovňovým konkurentem OpenMP je standard Pthreads (POSIX vlákna).

Příklad. OpenMP paralelizace sekvenčního kódu ve Fortranu

Sekvenční kód s paralelizovatelným cyklem

```
integer,parameter :: nmax=10; real a(nmax)
do i=1,nmax ; a(i)=i ; enddo
end program
```

OpenMP kód s paralelizovaným cyklem

```
integer,parameter :: nmax=10; real a(nmax)   sekvenční oblast
!$OMP PARALLEL                               direktiva pro vytvoření paralelní oblasti
!$OMP DO                                     direktiva pro rozdělení cyklu mezi vlákna
do i=1,nmax ; a(i)=i ; enddo                 paralelní oblast
!$OMP END DO                                 (volitelná) koncová direktiva cyklu
!$OMP END PARALLEL                           koncová direktiva oblasti, synchronizační bariéra
end program                                  sekvenční oblast
```

OpenMP kód s dotazovacími funkcemi v paralelní oblasti

```
use omp_lib                                  připojení modulu s rozhraním OpenMP procedur
!$OMP PARALLEL PRIVATE (nmax,n)              klauzule pro specifikaci soukromých proměnných
nmax=OMP_GET_NUM_THREADS()                   zjištění počtu vláken
n=OMP_GET_THREAD_NUM()                       zjištění čísla vlákna (master thread: 0)
!$OMP END PARALLEL
end program
```

| Překlad | pro paralelní běh | pro sekvenční běh (jsou-li volány OpenMP funkce) |
|---|---|---|
| | ifort -openmp files.f90 | ifort -openmp-stubs |
| | gfortran -fopenmp files.f90 | gfortran -lopenmp_stubs, je-li knihovna instalována |
| | pgfortran -mp files.f90 | pgfortran (bez -mp) |

Nastavení počtu vláken v paralelní oblasti pomocí proměnné prostředí

```
bash: export OMP_NUM_THREADS=4          windows: set OMP_NUM_THREADS=4
tcsh:  setenv OMP_NUM_THREADS 4
```

## Vytvoření paralelní oblasti

Hlavní proces v místě direktivy PARALLEL vytvoří předepsaný počet vláken a pro každé vlákno vlastní kopie proměnných soukromých v oblasti (PRIVATE; příbuzné klauzule FIRSTPRIVATE a REDUCTION zajišťují i inicializaci). Existují-li proměnné soukromé ve vláknu (specifikované direktivou THREADPRIVATE), jsou vláknům zpřístupněny (klauzulí COPYIN i inicializovány). Ostatní proměnné jsou sdílené (SHARED), jejich originální místa v paměti se vstupem do paralelní oblasti nemění a jsou zpřístupněna všem vláknům. Po dosažení direktivy END PARALLEL všemi vlákny (implicitní synchronizační bariéra) je v případě REDUCTION provedena redukční operace (např. sumace soukromých kopií do sdíleného originálu) a není-li oblast vnořena v jiné paralelní oblasti, dál pokračuje pouze hlavní proces s definovanými hodnotami SHARED, REDUCTION a THREADPRIVATE proměnných.

Sdílení proměnných je předurčené, explicitně určené nebo implicitně určené, DEFAULT mění implicitní určení. Předurčené sdílení PRIVATE mají řídicí proměnné (všech) cyklů v paralelní oblasti a THREADPRIVATE proměnné, nelze jej změnit a není třeba ho klauzulemi popisovat. Implicitně určené sdílení ostatních proměnných (tedy nejčastější případ) je SHARED nebo je dáno klauzulí DEFAULT, případně zrušeno variantou DEFAULT (NONE). Explicitně se sdílení určuje klauzulemi PRIVATE (FIRSTPRIVATE včetně inicializace), SHARED a REDUCTION; REDUCTION implikuje vytvoření soukromých proměnných, s jejichž hodnotami je na konci paralelní oblasti provedena redukční (kumulativní) operace (např. sumace, součin, nalezení extrému) a výsledek je zkombinován s hodnotou stejnojmenné vnější proměnné.

Vytvoření paralelní oblasti lze podmíněně potlačit (klauzule IF). Počet vláken v paralelní oblasti je dán výrazem v klauzuli NUM_THREADS, pokud není, tak proměnnou prostředí OMP_NUM_THREADS nebo implicitními pravidly; dynamická změna počtu vláken uvnitř paralelní oblasti je také možná. Vnořování paralelních oblastí bývá defaultně neúčinné, ve vnořených oblastech se pak vlákna dále nedělí (proměnná prostředí OMP_NESTED). Skoky do a z paralelní oblasti nejsou žádoucí.

Syntaxe direktivy:

```
!$OMP PARALLEL [IF NUM_THREADS DEFAULT PRIVATE FIRSTPRIVATE SHARED COPYIN REDUCTION]
paralelní oblast
!$OMP END PARALLEL
```

Lze psát !$omp, řádek direktivy vyžadující pokračování se ukončuje znakem &, direktivy pro pevný formát Fortranu 77 se uvozují C$OMP nebo *$OMP.

Přehled klauzulí:

IF (logical-scalar) pro podmíněné potlačení paralelní oblasti (default je .true. pro vytvoření)
NUM_THREADS (integer-scalar) pro předepsání počtu paralelních vláken
DEFAULT (PRIVATE|FIRSTPRIVATE|SHARED|NONE) pro přiřazení implicitního atributu sdílení proměnných
PRIVATE (list) proměnné soukromé v oblasti, bez inicializace a vracení jejich hodnot mimo oblast
FIRSTPRIVATE (list) proměnné soukromé v oblasti inicializované hodnotou stejnojmenné vnější proměnné
SHARED (list) sdílené proměnné
COPYIN (list) proměnné soukromé ve vláknu (THREADPRIVATE) inicializované hodnotou z hlavního procesu
REDUCTION (operator|procedure:list) redukční operace na soukromých proměnných s přenosem výsledku vně

OpenMP verze 4.0 slibuje klauzuli PROC_BIND (MASTER|CLOSE|SPREAD) pro nastavení afinity vláken neboli rozložení vláken mezi procesory. (Afinita je zatím řešena překladači individuálně, viz níže.)

Příklad. Sdílení proměnných. THREADPRIVATE proměnné jsou soukromé a při opakovaných vstupech do paralelní oblasti vázané na své vlákno; v klauzulích direktivy DO se neuvádějí. Proměnné mohou mít v různých oblastech různou sdílitelnost. Proměnná nmax je inicializovaná v sekvenční oblasti, sdílená v první oblasti, soukromá s inicializací v druhé oblasti. Proměnná n není inicializovaná v sekvenční oblasti, její soukromé kopie v první oblasti jsou inicializované, ale zanikají, soukromé kopie v druhé oblasti jsou inicializované (nedefinovanou) hodnotou ze sekvenční oblasti. Proměnná i existuje v počtu kopií totožném s počtem vláken, jejich hodnoty se přenášejí mezi paralelními oblastmi, v sekvenční oblasti je dostupná kopie nultého vlákna.

```
use omp_lib  ! bez modulu nebo popisu integer OMP_GET_THREAD_NUM je výsledek real a nesprávný
integer :: nmax=4
integer,save :: i  ! nemodulové THREADPRIVATE proměnné musí mít atribut save
!$OMP THREADPRIVATE (i)
!$OMP PARALLEL NUM_THREADS (nmax) DEFAULT (NONE) PRIVATE (n) SHARED (nmax)
n=OMP_GET_THREAD_NUM()
i=n
!$OMP END PARALLEL
!$OMP PARALLEL NUM_THREADS (nmax) FIRSTPRIVATE (n,nmax)
print *,n,i,nmax  ! n je inicializované nedefinovanou vnější hodnotou, i si drží hodnotu z předchozí oblasti
!$OMP END PARALLEL
end program
```

Příklad. Redukční operace. V paralelní oblasti se vytvoří soukromé kopie REDUCTION proměnné, vhodně inicializované (pro redukční operaci + nulou, pro * jedničkou). Na konci paralelní oblasti se hodnoty těchto soukromých proměnných spolu s hodnotou vnější proměnné předepsanou operací zkombinují a výsledek se přenese do vnější proměnné. Zde, díky volání OMP_GET_THREAD_NUM, vrací paralelní běh jiný výsledek než sekvenční běh nebo běh s IF (.false.).

```
use omp_lib
n=4
f=10.
!$OMP PARALLEL IF (.true.) NUM_THREADS (n) REDUCTION (*:f)
f=OMP_GET_THREAD_NUM()+1.
!$OMP END PARALLEL
print *,f  ! sekvenční běh: 1.; paralelní běh: 10. x n!
end program
```

**Direktivy DO, WORKSHARE a SECTIONS pro sdílení práce** (worksharing constructs)

Účelem OpenMP je rozdělit práci mezi paralelně běžící vlákna: indexované cykly se paralelizují direktivou DO, pro paralelizovatelné příkazy Fortranu 90 (přiřazování polí, forall, where) je určena direktiva WORKSHARE, různé větve zdrojového kódu lze rozložit mezi vlákna pomocí direktivy SECTIONS. Tyto direktivy a jejich koncové protějšky (nepovinné END DO, povinné END WORKSHARE, END SECTIONS) se aplikují v paralelní oblasti, tedy mezi direktivami PARALLEL a END PARALLEL, nebo se s nimi v elementárních případech spojují do kombinovaných direktiv, např. PARALLEL DO a END PARALLEL DO. Každá z uvedených direktiv musí být dosažena buď všemi vlákny nebo žádným z nich, koncové direktivy jsou implicitními synchronizačními bariérami, u kterých na sebe všechna vlákna vyčkají; synchronizaci vláken u koncových direktiv potlačuje klauzule NOWAIT. Řídicí proměnné všech cyklů (paralelizovaných i vnořených) jsou předurčeně soukromé, není možné jejich sdílitelnost měnit a není třeba ji specifikovat ani po DEFAULT (NONE) nebo (SHARED).

Direktiva DO: paralelizace indexovaných cyklů

```
!$OMP DO [PRIVATE FIRSTPRIVATE LASTPRIVATE REDUCTION SCHEDULE COLLAPSE ORDERED]
indexovaný cyklus DO (s možností vnořených cyklů)
[!$OMP END DO [NOWAIT]] (párová direktiva END DO není povinná)
```

Direktiva WORKSHARE: paralelizace konstrukcí Fortranu 90

```
!$OMP WORKSHARE (bez klauzulí)
blok s cykly forall, příkazy where a přiřazovacími příkazy se skaláry i poli
!$OMP END WORKSHARE [NOWAIT]
```

Direktiva SECTIONS: paralelní provádění různých bloků kódu

```
!$OMP SECTIONS [PRIVATE FIRSTPRIVATE LASTPRIVATE REDUCTION]
!$OMP SECTION // blok ... (více sekcí pro více vláken)
!$OMP END SECTIONS [NOWAIT]
```

Klauzule PRIVATE, FIRSTPRIVATE a REDUCTION jsou popsány u direktivy PARALLEL; uvádět klauzuli SHARED nelze (vnější soukromou proměnnou nelze změnit ve sdílenou). Klauzule LASTPRIVATE (list) zajišťuje vrácení hodnot soukromých proměnných ze sekvenčně poslední iterace cyklu nebo poslední sekce, proměnná může být současně v seznamech FIRSTPRIVATE i LASTPRIVATE. Způsob rozdělení indexovaného cyklu mezi vlákna řídí klauzule SCHEDULE (kind[,chunk]): kind může nabývat hodnot STATIC (pevné přidělení iterací vláknům po soustech chunk), DYNAMIC (průběžné přidělování volným vláknům), GUIDED (dynamické přidělování s průběžně se zmenšujícími porcemi), RUNTIME (volba kind za chodu programu), AUTO (volba překladačem). Klauzule COLLAPSE (n) umožňuje současně paralelizovat n vnořených cyklů, klauzule ORDERED vynuceje sekvenční průchod cyklem.

Kombinované direktivy pro paralelní oblast obsahující jedinou direktivu pro sdílení práce

| | | |
|---|---|---|
| !$OMP PARALLEL DO ... | cyklus | !$OMP END PARALLEL DO [NOWAIT] |
| !$OMP PARALLEL WORKSHARE ... | blok | !$OMP END PARALLEL WORKSHARE [NOWAIT] |
| !$OMP PARALLEL SECTIONS ... | sekce | !$OMP END PARALLEL SECTIONS [NOWAIT] |

Příklad. Kombinovaná direktiva pro paralelizaci indexovaného cyklu s vnořeným cyklem. Proměnné i, j jsou předurčeně soukromé a není třeba to uvádět explicitně ani při DEFAULT (NONE). Sdílení SHARED (x) by bylo nesprávné, pracovní skaláry v těle cyklu typicky mají být PRIVATE. (Bývá častým efektem optimalizace, že takové skaláry jsou umístěny mimo sdílitelnou paměť a opomenutí klauzule PRIVATE chybu nezpůsobí.) Pole zpracovávaná cyklem jsou SHARED.

```
!$OMP PARALLEL DO SCHEDULE (DYNAMIC,n/24) DEFAULT (NONE) PRIVATE (x) SHARED (a)
do i=1,n ; x=0 ; do j=1,i ; x=x+i ; enddo ; a(i)=x ; enddo
!$OMP END PARALLEL DO
```

Příklad. Redukční operace v cyklu. V paralelní oblasti se vytvoří soukromé kopie REDUCTION proměnné inicializované nulou. Na konci paralelní oblasti se hodnoty těchto soukromých proměnných spolu s hodnotou vnější proměnné sečtou a výsledek se přenese do vnější proměnné. Ukázka vrací stejný výsledek pro sekvenční i paralelní běh.

```
s=0.
!$OMP PARALLEL DO DEFAULT (NONE) SHARED (nmax) REDUCTION (+:s)
do i=1,nmax ; s=s+i ; enddo
!$OMP END PARALLEL DO
print *,s
```

Příklad. Direktiva SECTIONS

...

**Direktivy SINGLE a MASTER**

Některé části zdrojového kódu v paralelní oblasti je žádoucí provést jen jednou (jediným vláknem), např. výpis nebo rozdílení práce úlohám (direktiva TASK níže). Direktiva SINGLE vpustí do svého bloku první vlákno, které jí dosáhne, ostatní vlákna blok přeskočí a synchronizují se na koncové direktivě; klauzule NOWAIT synchronizaci vláken potlačí, klauzule COPYPRIVATE zajistí vynesení hodnoty soukromých proměnných vně bloku. Direktiva MASTER vpustí do svého bloku pouze vlákno indexované nulou (master thread), ostatní vlákna blok přeskočí; direktiva MASTER na rozdíl od SINGLE nemá žádné kaluzule a na její koncové direktivě se vlákna nesynchronizují.

Syntaxe direktiv:

```
!$OMP SINGLE [PRIVATE FIRSTPRIVATE]
blok prováděný jediným vláknem
!$OMP END SINGLE [NOWAIT COPYPRIVATE]
!$OMP MASTER
!$OMP END MASTER
```

Příklad. Výpis uvnitř paralelní oblasti

```
!$OMP PARALLEL
!$OMP SINGLE
print *,OMP_GET_NUM_THREADS()
!$OMP END SINGLE
!$OMP END PARALLEL
```

**Direktiva TASK pro definování úloh** (task construct)

Úlohy (tasks), zavedené až v OpenMP verze 3.0, umožňují paralelizovat obecnější konstrukce než starší direktivy pro sdílení práce (pro cykly nebo statické sekce). Obložením bloku zdrojového kódu direktivou TASK je definována (explicitní) úloha, která je zařazena do seznamu úloh plánovaných k pozdějšímu (odloženému) vykonání. (Ostatní direktivy pro sdílení práce vytvářejí implicitní úlohy.) Chopit se vykonání úlohy může kterékoliv vlákno; není-li úloha uvolněná (není UNTIED), musí ji toto vlákno i dokončit, jinak (klauzule UNTIED) může úlohu převzít i jiné vlákno; jsou definována místa, na kterých může vlákno zpracování úlohy přerušit a přejít k jiné práci. Úloha může žádat o pozastavení rodičovské úlohy a vykonání sebe dříve (klauzule IF), ne nutně ihned. Úloha (např. v dostatečně zanořené rekurzivní proceduře) může požádat i o okamžité vykonání týmž vláknem (klauzule FINAL), obvykle v kombinaci s požadavkem sdílet prostředí s rodičovskou úlohou (klauzule MERGEABLE); v takovém případě může (ale nemusí) překladač ušetřit na režii a vykonat kód sekvenčně bez vytváření nové úlohy. (Jistější je ukončit vytváření úloh přechodem do samostatné větve zdrojového kódu.) Default direktiva TASK vytvoří úlohu odložitelnou, IF (.true.) a FINAL (.false.), s vlastním prostředím (ne MERGEABLE) a vázanou na vlákno (ne UNTIED).

Synchronizace úloh: direktiva TASKWAIT je synchronizační bariérou pro úlohy vytvořené aktuální úlohou (child tasks), ostatní implicitní a explicitní synchronizační bariéry (END direktivy bez NOWAIT a direktiva BARRIER) vyžadují dokončení všech vytvořených úloh.

Sdílení proměnných v úlohách: proměnné bez předurčeného nebo explicitně určeného sdílení jsou SHARED, pokud jsou SHARED i vně úlohy, jinak jsou FIRSTPRIVATE. Z toho plyne pro úlohy vytvářené v procedurách s PRIVATE lokálními proměnnými, že tyto proměnné jsou uvnitř úlohy implicitně FIRSTPRIVATE; pokud má lokální PRIVATE proměnná vynést z úlohy hodnotu, musí být v úloze explicitně SHARED (byť je vně úlohy PRIVATE).

Syntaxe direktiv:

```
!$OMP TASK [IF FINAL UNTIED DEFAULT MERGEABLE PRIVATE FIRSTPRIVATE SHARED]
definiční blok úlohy
!$OMP END TASK
!$OMP TASKWAIT
```

Přehled klauzulí:

IF (logical-scalar) .false. pro přednostní vykonání úlohy před rodičovskou úlohou (undeferred task)
FINAL (logical-scalar) .true. pro okamžité vykonání úlohy (final/included task) vláknem, které na ni narazilo
UNTIED vytvoření úlohy, jejíž chod může být snadno přerušován a přejímán jinými vlákny (untied task)
DEFAULT (PRIVATE|FIRSTPRIVATE|SHARED|NONE) přiřazení implicitního atributu sdílení proměnných
MERGEABLE žádost o vytvoření úlohy sdílející prostředí s rodičovskou úlohou
PRIVATE (list) proměnné soukromé v úloze, bez inicializace a vracení jejich hodnot mimo úlohu
FIRSTPRIVATE (list) proměnné soukromé v úloze inicializované hodnotou stejnojmenné vnější proměnné
SHARED (list) proměnné úlohy sdílející paměťové místo s originálními proměnnými vně úlohy

OpenMP verze 4.0 doplňuje klauzuli DEPEND direktivy TASK pro vyjádření závislosti mezi úlohami, direktivu TASKYIELD pro explicitní vyznačení místa, kde lze pozastavit úlohu, a direktivu TASKGROUP pro vytvoření synchronizační bariéry pro skupinu úloh.

Poznámky. Úlohy jsou obvykle vytvářeny jedním vláknem (direktiva SINGLE) a obvykle provedeny později, obecně v nespecifikovaném pořadí, některým z připravených vláken. Úloha s klauzulí UNTIED není vázána na vlákno, kterým je zahájena, a může být kdekoliv přerušena (tj. užití THREADPRIVATE proměnných nemá smysl). Úloha (bez UNTIED) může být přerušena pouze za místem definice (jiné) explicitní úlohy, v místě direktivy TASKWAIT a v místech implicitních a explicitních bariér (tj. vlákno vytvářející úlohy může dočasně přerušit jejich vytváření a chopit se jejich vykonávání). Nová úloha (bez IF (.false.) a UNTIED) bude vláknem zahájena jen tehdy, je-li potomkem všech ostatních úloh vázaných na dané vlákno (nebo nejsou-li žádné úlohy vázané na dané vlákno). Pracuje-li úloha se SHARED proměnnou, musí být zajištěna její existence po celou dobu vykonatelnosti úlohy. Skoky do a z definičního bloku úlohy mají nespecifikovaný efekt. Nesynchronizované výpisy různých úloh na totéž zařízení mají nespecifikovaný efekt (mohou vést k zamrznutí).

Příklad. Zpracování spojového seznamu není paralelizovatelné pomocí direktiv DO a SECTIONS, pomocí TASK však ano. Pořadí zpracování úloh může programátor částečně ovlivnit. Klauzule IF (.false.) má vést k přerušení vytvářející úlohy po dobu, než bude provedena nově vytvořená úloha; aktuální verze tří testovaných překladačů to respektují, nad rámec specifikace omezují vykonání úloh s IF (.false.) na vlákno vytvářející úlohy. Toto by měla podle specifikace vynucovat klauzule FINAL (.true.), což se však v aktuálních překladačích zatím neděje. Proměnná p je SHARED v paralelní oblasti a defaultně by taková byla i v úlohách, každá úloha si proto musí vyžádat vlastní inicializovanou kopii pomocí FIRSTPRIVATE. Alternativně by proměnná p mohla být deklarována PRIVATE v direktivě PARALLEL a být tak v úlohách FIRSTPRIVATE implicitně; v paralelní oblasti by však přitom vzniklo tolik kopií, kolik vláken, a vzhledem k výhradnímu užití proměnné v bloku SINGLE by všechny kopie až na jednu byly nevyužity.

```
type tNode ; integer i ; type(tNode),pointer :: next ; end type
type(tNode),pointer :: head,p
!$OMP PARALLEL
!$OMP SINGLE
p=>head
do
  !$OMP TASK FIRSTPRIVATE (p) ! IF (.false.) ! FINAL (.true.) MERGEABLE
  print *,p%i
  !$OMP END TASK
  p=>p%next ; if (.not.associated(p)) exit
enddo
!$OMP END SINGLE
!$OMP END PARALLEL
```

Příklad. Zdvojující rekurze pro výpočet Fibonacciho čísel je učebnicovou ukázkou, nikoliv prakticky užitelným algoritmem. ...

## Direktiva THREADPRIVATE pro deklaraci proměnných soukromých ve vláknu

Proměnné THREADPRIVATE se replikují pro každé vlákno a jsou na něj vázané po celou dobu existence. Obvykle si udržují své hodnoty i mezi různými paralelními oblastmi. Hodí se tak např. pro modulové proměnné, u kterých se nehodí jejich implicitní atribut SHARED. Na vstupu do paralelní oblasti s klauzulí COPYIN jsou inicializovány hodnotou z hlavního procesu (ve vnořené oblasti hodnotou ze vstupního vlákna), na výstupu z bloku SINGLE s COPYPRIVATE jsou přepsány hodnotou z bloku SINGLE. Užití THREADPRIVATE proměnných v úlohách (tasks) vyžaduje opatrnost (úlohy se mohou přepínat, UNTIED úlohy mohou měnit vlákno). Nemodulové THREADPRIVATE proměnné musí mít atribut SAVE.

Syntaxe: !$OMP THREADPRIVATE (list)
Příklad

## Ostatní direktivy

CRITICAL [name] pro blok, který smí být v danou chvíli prováděn nejvýše jedním vláknem, ostatní čekají
ATOMIC pro řádek, který smí být v danou chvíli prováděn nejvýše jedním vláknem, ostatní čekají
BARRIER místo (povinného) setkání všech vláken
FLUSH (list) místo synchronizace uvedených proměnných mezi hlavní pamětí a (dočasnou) pamětí vlákna
(implicitně po BARRIER,PARALLEL,CRITICAL,ORDERED,END PARALLEL/DO/SECTIONS/SINGLE/CRITICAL/ORDERED)
ORDERED pro blok v oblasti DO ORDERED, který musí být proveden v předepsaném pořadí

## Funkce a podprogramy

Rozhraní procedur definováno v modulech z omp_lib.f90. Připojení modulu omp_lib (jeho prostřednictvím i ostatních modulů) je žádoucí, neboť funkce obvykle vracejí integer výsledek, jejich implicitní typ je však real. Do zdrojových kódů pro Fortran 77 se vkládá soubor s definicemi omp_lib.h.

OMP_GET_: THREAD_NUM, NUM_THREADS, MAX_THREADS, NUM_PROCS, DYNAMIC, NESTED, WTIME, WTICK
OMP_SET_: NUM_THREADS, DYNAMIC, NESTED (podprogramy) (funkce)
OMP_IN_: PARALLEL (funkce)
OMP_: INIT_LOCK, DESTROY_LOCK, SET_LOCK, UNSET_LOCK, TEST_LOCK (podprogramy)

## Proměnné prostředí

Proměnné prostředí jsou užívány pro (statické) nastavování různých vlastností spouštěného programu; jejich změna ovlivňuje jeho chování bez potřeby nového překladu. Tyto vlastnosti bývá možné konfigurovat i dynamicky, pomocí klauzulí direktiv nebo nastavováním tzv. interních proměnných (internal control variables, ICV) za chodu programu (tj. zápisem ve zdrojovém kódu). Linuxové shelly nastavují proměnné prostředí pomocí příkazů (bash) export var=value nebo (tcsh) setenv var value, interpret Windows pomocí set var=value.

OMP_NUM_THREADS pro počet vláken; default bývá počet procesorů, někdy (např. Portland) však jen 1
OMP_DYNAMIC pro možnost dynamické změny počtu vláken uvnitř paralelní oblasti (nastavením ICV)
OMP_SCHEDULE pro nastavení scheduleru paralelních DO cyklů (klauzule SCHEDULE direktivy DO má přednost)
OMP_NESTED pro možnost vnořování paralelních oblastí (default bývá vnořená paralelizace neúčinná)
OMP_STACKSIZE pro možnost zvětšení zásobníku vlákna (doporučení ifort pro OMP_STACKSIZE: 16M)
a řada dalších, včetně proprietárních proměnných překladačů (ifort: KMP_ proměnné, gfortran: GOMP_ proměnné)

## Afinita vláken: svázání vláken s procesory

Běžně dostupné procesory (CPU Intel/AMD) obsahují více jader (cores; 2, 4, 6, 8), jádra mohou obsluhovat více instrukčních front (hyperthreading; 2), základní desky mohou pojmout více CPU (2); je tak dobře možné dostat se ke stroji až s 32 (virtuálními, „OpenMP") procesory (2 CPU x 8 jader x 2 fronty), obecně nerovnocennými. Default nestanovuje žádnou vazbu mezi vlákny a procesory, vlákna tak mohou být natěsnána v rámci jednoho jádra nebo naopak být mezi jádry často přepínána. Může být proto vhodné konfigurovat afinitu vláken (thread affinity), neboli vlákna s procesory svázat. OpenMP verze 4.0 slibuje jemné nastavení pomocí proměnných OMP_PROC_BIND (false, true, spread, close, master) a OMP_PLACES (threads, cores, sockets nebo seznam), verze 3.1 se omezuje jen na vynucení afinity, OMP_PROC_BIND=true, verze 3.0 nenabídla nic. Aktuálně je tak jemná konfigurace afinity řešena překladači individuálně (proměnná ifort: KMP_AFFINITY, např. scatter,verbose pro pravidelné rozložení vláken a informaci o topologii procesorů, nebo explicit,proclist=[0,2,4,6]; gfortran i ifort: GOMP_CPU_AFFINITY, např. 0,2,4-7:2 pro osazení sudých procesorů; pgfortran MPBIND=yes, MP_BLIST=0,2,4,6).

## Poznámky

Implementace OpenMP v překladačích se mohou lišit (př. v následujících větách). Paralelní oblasti lze vnořovat (může být neúčinné, ve vnořených oblastech se vlákna nemusí dále dělit). Počet vláken může být dynamicky měněn (může být neúčinné). Vlákna mohou udržovat vlastní datovou cache, jejíž obsah lze explicitně vylévat (FLUSH) do sdílené paměti. Lokální pole v paralelních oblastech jsou umisťována do zásobníku, který může přetéci (Segmentation fault); je možno ho zvětšit (OMP_STACKSIZE) nebo se mu vyhýbat (např. ifort -heap-arrays). Některé direktivy v některých překladačích paralelizují hůře, než by se čekalo (týká se zvláště fortranské direktivy WORKSHARE). Může dojít k souboji paralelizující a optimalizující části překladače (paralelizace může omezit optimalizaci). Vstup a výstup dat v paralelních oblastech vyžaduje pečlivost (direktivy SINGLE a MASTER, synchronizační bariéry).
zápis do polí pomocí ATOMIC, př. s. 204
kdy nutný explicitní FLUSH
data dependency, data conflict, deadlock s. 134

# OpenMP a sdílení paměti

## Fyzické a logické sdílení paměti

OpenMP je určeno pro hardware umožňující fyzicky sdílet paměť každým procesorem/jádrem. Jednotlivá vlákna programu logicky sdílet proměnné v paměti mohou, ale nemusí, často ani nesmí.
Vlákna mohou udržovat vlastní dočasný obraz dat ze sdílené paměti (v registrech, v cache paměti), data ve sdílené paměti tak nemusí být aktuální (relaxed-consistency shared-memory model). Je nezbytná frekventovaná (implicitní i explicitní) synchronizace sdílené paměti a jejího obrazu ve vláknech.

## Způsob sdílení (a nesdílení) proměnných

sdílené proměnné (SHARED)
- typické použití: pole v paralelizovaných cyklech, modulová pole
- ke sdíleným proměnným mohou přistupovat všechna vlákna (s možnou potřebou synchronizace)
- klauzule SHARED (list) a DEFAULT (SHARED) direktiv PARALLEL a TASK
- implicitně SHARED proměnné v oblasti PARALLEL, modulové proměnné a lokální proměnné procedur se SAVE
- př. sdílené pole
  !$OMP PARALLEL DO ; DO i=1,n ; a(i)=… ; ENDDO ; !$OMP END PARALLEL DO

proměnné soukromé v oblasti (PRIVATE)
- typické použití: pomocné skaláry v paralelizovaných cyklech a lokální proměnné volaných procedur
- vlákno si pro soukromé proměnné vytvoří dočasnou paměť, jejich hodnoty se neinicializují ani nevracejí
- klauzule PRIVATE (list) direktiv PARALLEL, TASK, DO, SECTIONS a SINGLE
- varianty FIRSTPRIVATE (list) pro převzetí počáteční hodnoty do soukromé proměnné
  LASTPRIVATE (list) pro vrácení hodnoty proměnné z poslední iterace cyklu nebo poslední sekce
  proměnná se může vyskytovat současně pouze v klauzulích FIRSTPRIVATE a LASTPRIVATE
- př. pomocný skalár
  !$OMP PARALLEL DO PRIVATE (x) ; DO i=1,n ; x=… ; a(i)=x ; ENDDO ; !$OMP END PARALLEL DO
- klauzule REDUCTION (operator|procedure:list) direktiv PARALLEL, DO a SECTIONS
  pro kumulativní operaci na (rozumně inicializovaných) soukromých proměnných jednotlivých vláken a kombinaci výsledku s aktuální hodnotou vnější sdílené proměnné
  operator: +, *, −, .AND., .OR., .EQV., .NEQV., procedure: MAX, MIN, IAND, IOR, IEOR
- klauzule DEFAULT (PRIVATE) a DEFAULT (FIRSTPRIVATE) direktiv PARALLEL a TASK

proměnné soukromé ve vláknu (THREADPRIVATE)
- typické použití: pomocné modulové proměnné modifikované v paralelní oblasti
- (deklarativní) direktiva !$OMP THREADPRIVATE (list) v oblasti platnosti proměnných
- prostor THREADPRIVATE proměnných ze sekvenční oblasti (i hodnoty) přebírá master thread, následně je do sekvenční oblasti i vrací, každé další vlákno pracuje s vlastními replikovanými proměnnými s možností jejich inicializace a uchování hodnot pro totéž vlákno v následující paralelní oblasti
- klauzule COPYIN (list) direktivy PARALLEL pro inicializaci soukromých proměnných hodnotami z master threadu
- nemodulové THREADPRIVATE proměnné musí mít atribut SAVE

## Stanovení způsobu sdílení proměnných

předurčené sdílení
- proměnné paralelizovaných cyklů (DO, PARALLEL DO), cyklických seznamů a FORALL cyklů jsou PRIVATE
- proměnné cyklů v oblasti PARALLEL nebo TASK jsou PRIVATE
- proměnné uvedené v direktivě THREADPRIVATE jsou THREADPRIVATE
- předurčenost sdílení nelze měnit pomocí klauzulí direktiv, až na výjimky (např. proměnná paralelizovaného cyklu může být uvedena v klauzulích PRIVATE nebo LASTPRIVATE)

explicitně určené sdílení
- vyjmenováním proměnných v seznamu klauzulí direktiv
- direktivy PARALLEL, TASK: klauzule SHARED, PRIVATE, FIRSTPRIVATE
- direktivy DO, SECTIONS, SINGLE: klauzule PRIVATE, FIRSTPRIVATE, LASTPRIVATE
- direktiva SINGLE: klauzule PRIVATE, FIRSTPRIVATE, direktiva END SINGLE: COPYPRIVATE
- klauzule DEFAULT (NONE) pro vynucení explicitního určení všech (kromě proměnných s předurčeným sdílením)

implicitně určené sdílení
- týká se proměnných bez předurčeného sdílení a neuvedených v seznamech klauzulí
- atribut sdílení podle klauzule DEFAULT (SHARED | PRIVATE | FIRSTPRIVATE) direktiv PARALLEL a TASK

jinak proměnné v oblasti PARALLEL jsou SHARED a v oblasti TASK jsou SHARED (jsou-li SHARED vně oblasti)
nebo FIRSTPRIVATE
implicitně určené sdílení ve volaných procedurách
formální argumenty procedur předané odkazem přebírají atribut sdílení od skutečných argumentů
lokální a modulové THREADPRIVATE proměnné jsou THREADPRIVATE
lokální proměnné s inicializací nebo atributem SAVE jsou SHARED, ostatní PRIVATE
modulové proměnné ve volaných procedurách jsou SHARED

Synchronizace sdílené paměti
každé vlákno si může udržovat vlastní dočasný obraz sdílené paměti, je nezbytná synchronizace (flush operation)
kroky: …
explicitně direktivou FLUSH (list) nebo FLUSH bez seznamu pro všechna data sdílená vláknem
implicitně FLUSH bez seznamu na vstupu i výstupu bloků PARALLEL, CRITICAL a ORDERED,
na výstupu bloků DO, WORKSHARE, SECTIONS a SINGLE (pokud nemají klauzuli NOWAIT),
před a za direktivou BARRIER a ostatními místy setkání vláken,
FLUSH se seznamem před a za oblastí direktivy ATOMIC
synchronizace se neprovádí na vstupu DO, WORKSHARE, SECTIONS a SINGLE a kolem MASTER
FLUSH pro alokovatelné pole synchronizuje celé pole, pro ukazatele (i ukazatelové pole) pouze ukazatel

„Životní situace"
paralelizovaný cyklus DO bez volání procedur
proměnná cyklu (vždy) PRIVATE, ostatní proměnné explicitně nebo podle DEFAULT nebo jsou SHARED
paralelizovaný cyklus DO s voláním procedury bez modulových dat
viz výše a lokální proměnné procedury jsou buď THREADPRIVATE (povinné SAVE), SHARED (mají-li SAVE
nebo jsou-li inicializované) nebo jsou PRIVATE
paralelizovaný cyklus DO s voláním procedury s modulovými daty
viz výše a modulové proměnné jsou buď THREADPRIVATE nebo SHARED
direktiva TASK v proceduře
lokální proměnné procedury bývají PRIVATE a úlohy vytvářené v proceduře s nimi zacházejí jako
s FIRSTPRIVATE; proměnné, které mají z úloh vynášet hodnoty, musí být explicitně SHARED
datové dostihy (data races): do sdílené proměnné zapisuje více vláken bez synchronizace
do sdílené proměnné jedno vlákno zapisuje, jiné z ní bez synchronizace čte
předávání sekcí pole jako argumentů do procedur může vyvolat zpětný (nesynchronizovaný) zápis
FIRSTPRIVATE soukromé proměnné inicializované aktuální hodnotou stejnojmenné vnější proměnné
mohou obsahovat v různých vláknech různé hodnoty
pozor s NOWAIT, př. s. 158, atomic př. s. 202
společné bloky (COMMON)
mohou se vyskytovat v direktivě THREADPRIVATE (??) i klauzulích PRIVATE/SHARED, proměnné ze
společných bloků se samostatně v klauzulích vyskytovat mohou, ale ztrácejí tím na společný blok vazbu

18. 4. 2013