OpenMP (Open Specification for Multi Processing)

– rozhraní pro (explicitní) programování paralelních aplikací (API, Application Programming Interface)
– multithreadový multitasking: hlavní proces (master thread) vytváří paralelní vlákna (threads)
– omezení: systémy se sdílenou pamětí (paměť procesu fyzicky přístupná všem vláknům)
– verze: pro Fortran a C/C++
– http://www.openmp.org, dokumentace: OpenMP API Specification ver. 2.5 (PDF 250 stran)
– součásti: direktivy (metapříkazy) pro překladač
funkce a podprogramy
proměnné prostředí
– zdrojový kód s voláním OpenMP lze beze změn překládat i pro sériový běh (ifort: -openmp-stubs)

Použití OpenMP ve Fortranu

– vložení direktiv (!$OMP ...) do zdrojového kódu:

```
př.   sériový kód                                          (master thread only)
      !$OMP PARALLEL PRIVATE (i,j) SHARED (a)              (parallel region, podmnožina podprogramu)
      kód prováděný všemi vlákny                           (fork-join model)
      !$OMP DO                                             (těsně před cyklem)
      do i=1,nmax ; a(i)=i ; enddo                         (pouze cyklus s řídicí proměnnou, nelze skok ven)
      !$OMP END DO                                         (těsně za cyklem)
      u následující direktivy na sebe všechna vlákna počkají, krach jednoho vlákna je krachem všech
      !$OMP END PARALLEL                                   (synchronize and terminate)
      sériový kód                                          (master thread only)
```

– volání funkcí a podprogramů:

```
      print *, OMP_GET_NUM_THREADS()                       pro počet vláken
      print *, OMP_GET_THREAD_NUM()                        pro číslo vlákna, 0 pro master thread
      call OMP_SET_NUM_THREADS(integer)                    pro změnu počtu vláken
```

vhodné připojit moduly z omp_lib.f90 nebo include-soubor omp_lib.h s konstantami a INTERFACE bloky

– nastavení počtu vláken pomocí proměnné prostředí:

```
      OMP_NUM_THREADS                                      pro počet vláken
pozn. bash: export var=value;  tcsh: setenv var value;  Windows: set var=value
```

– překlad:

```
ifort -O3 -openmp -o a.out files.f90          nebo  -openmp-stubs  pro sériový běh
gfortran -O3 -fopenmp -o a.out files.f90
```

Přehled direktiv, funkcí, podprogramů a proměnných

– vytvoření paralelní oblasti (pokud IF .true., o daném počtu vláken, s lokálními a sdílenými jmény)

```
!$OMP PARALLEL IF (scalar-logical-expression) NUM_THREADS (scalar-integer-expression)
!$OMP&      PRIVATE (list) SHARED (list) DEFAULT (PRIVATE|SHARED|NONE) FIRSTPRIVATE (list)
!$OMP&      REDUCTION (operator:list) COPYIN (list)
!$OMP END PARALLEL
```

– sdílení práce

!$OMP DO, WORKSHARE, SECTIONS, SINGLE s párovými !$OMP END

1. vlákna se dělí o DO cyklus s řídicí proměnnou nebo o strukturované příkazy Fortranu 90 (data parallelism)

```
!$OMP DO SCHEDULE.. ORDERED PRIVATE.. FIRSTPRIVATE.. LASTPRIVATE.. REDUCTION..
způsob průchodu SCHEDULE (type,chunk), type: STATIC, DYNAMIC, GUIDED, RUNTIME, chunk: integer
!$OMP END DO [NOWAIT]                                  při NOWAIT bez synchronizace vláken
```

nebo

```
!$OMP WORKSHARE
blok: přiřazovací příkazy se skaláry i poli, cyklus forall, cyklus where, atomic, critical a parallel constructs
!$OMP END WORKSHARE [NOWAIT]
```

2. vlákna provádějí různé bloky (functional parallelism)

```
!$OMP SECTIONS PRIVATE.. FIRSTPRIVATE.. LASTPRIVATE.. REDUCTION..
!$OMP SECTION ...
!$OMP END SECTIONS [NOWAIT]
```

3. jedno vlákno blok provede, ostatní vlákna blok přeskočí (např. výpis)

```
!$OMP SINGLE PRIVATE.. FIRSTPRIVATE..
!$OMP END SINGLE [NOWAIT]
```

– kombinace předchozích
  !$OMP PARALLEL DO ... blok !$OMP END PARALLEL DO [NOWAIT]
  !$OMP PARALLEL WORKSHARE blok !$OMP END PARALLEL WORKSHARE [NOWAIT]
  !$OMP PARALLEL SECTIONS ... blok !$OMP END PARALLEL SECTIONS [NOWAIT]
– synchronizace
  !$OMP MASTER, CRITICAL, ATOMIC, BARRIER, FLUSH, ORDERED
  MASTER pro blok, který provede jen master thread, ostatní jej přeskočí
  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
– oblast platnosti
  THREADPRIVATE,
  PRIVATE, SHARED, DEFAULT, FIRSTPRIVATE, LASTPRIVATE, REDUCTION, COPYIN,
  SCHEDULE, ORDERED, NOWAIT
– funkce a podprogramy
  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
  OMP_: SCHEDULE, NUM_THREADS, DYNAMIC, NESTED

Poznámky
– implementace se mohou lišit (př. následující dva řádky)
– 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 umisťována do zásobníku, který může přetéci (Segmentation fault)
– vstup/výstup v paralelních oblastech vyžaduje pečlivost