

SURFEX

Suppression des variables globales



Gabriel Jonville, Andrea Piacentini, Isabelle d'Ast

5 juillet 2013

Objectif : supprimer les instructions `THREADPRIVATE` du parallélisme OpenMP

Etat des lieux du code :

Nombre d'instructions OMP `THREADPRIVATE` : 2789

Nombre de subroutines : 2255

Nombre de fichiers : 1542

Transformation du code se réalise en 2 phases :

Phase 1 : Pré-fixage des variables globales par le nom de la structure

Phase 2 : Passage par argument des structures

Phase 1 : Pré-fixage des variables globales par le nom de la structure

1. Transformation intermédiaire des modules avec

- ajout d'une variable POINTER de type TYPE défini dans le module, avec THREADPRIVATE
- suppression des POINTER des champs de la structure et des THREADPRIVATE associés
- simplification des sous-routines _GOTO_MODEL et _ALLOC

```
TYPE(CH_SEAFLUX_t), ALLOCATABLE, TARGET, SAVE :: CH_SEAFLUX_MODEL(:)
```

```
CHARACTER(LEN=6), POINTER :: CCH_DRY_DEP=>NULL()
!SOMP THREADPRIVATE(CCH_DRY_DEP)
REAL, DIMENSION(:,:), POINTER :: XDEP=>NULL()
!SOMP THREADPRIVATE(XDEP)
CHARACTER(LEN=6), DIMENSION(:), POINTER :: CSV=>NULL()
!SOMP THREADPRIVATE(CSV)
INTEGER, POINTER :: NSV_CHSBEG=>NULL(), NSV_CHSEND=>NULL()
!SOMP THREADPRIVATE(NSV_CHSBEG, NSV_CHSEND)
INTEGER, POINTER :: NSV_DSTBEG=>NULL(), NSV_DSTEND=>NULL()
!SOMP THREADPRIVATE(NSV_DSTBEG, NSV_DSTEND)
INTEGER, POINTER :: NSV_SLTBEG=>NULL(), NSV_SLTEND=>NULL()
!SOMP THREADPRIVATE(NSV_SLTBEG, NSV_SLTEND)
INTEGER, POINTER :: NSV_AERBEG=>NULL(), NSV_AEREND=>NULL()
!SOMP THREADPRIVATE(NSV_AERBEG, NSV_AEREND)
INTEGER, POINTER :: NBEO=>NULL()
!SOMP THREADPRIVATE(NBEO)
INTEGER, POINTER :: NDSTEO=>NULL()
!SOMP THREADPRIVATE(NDSTEO)
INTEGER, POINTER :: NSLTEO=>NULL()
!SOMP THREADPRIVATE(NSLTEO)
INTEGER, POINTER :: NAEREO=>NULL()
!SOMP THREADPRIVATE(NAEREO)
CHARACTER(LEN=6), DIMENSION(:), POINTER :: CCH_NAMES=>NULL()
!SOMP THREADPRIVATE(CCH_NAMES)
CHARACTER(LEN=6), DIMENSION(:), POINTER :: CDSTNAMES=>NULL()
!SOMP THREADPRIVATE(CDSTNAMES)
CHARACTER(LEN=6), DIMENSION(:), POINTER :: CSLTNAMES=>NULL()
!SOMP THREADPRIVATE(CSLTNAMES)
CHARACTER(LEN=6), DIMENSION(:), POINTER :: CAER_NAMES=>NULL()
!SOMP THREADPRIVATE(CAER_NAMES)
```

CONTAINS

```
SUBROUTINE CH_SEAFLUX_GOTO_MODEL(KFROM, KTO, LKFROM)
LOGICAL, INTENT(IN) :: LKFROM
INTEGER, INTENT(IN) :: KFROM, KTO
REAL(KIND=JPRB) :: ZHOOK_HANDLE
```

```
! Save current state for allocated arrays
IF (LKFROM) THEN
  CH_SEAFLUX_MODEL(KFROM)%XDEP=>XDEP
  CH_SEAFLUX_MODEL(KFROM)%CSV=>CSV
  CH_SEAFLUX_MODEL(KFROM)%CCH_NAMES=>CCH_NAMES
  CH_SEAFLUX_MODEL(KFROM)%CDSTNAMES=>CDSTNAMES
  CH_SEAFLUX_MODEL(KFROM)%CSLTNAMES=>CSLTNAMES
  CH_SEAFLUX_MODEL(KFROM)%CAER_NAMES=>CAER_NAMES
ENDIF
```

```
! Current model is set to model KTO
IF (LHOOK) CALL DR_HOOK('MODD_CH_SEAFLUX_N:CH_SEAFLUX_GOTO_MODEL',0,ZHOOK_HANDLE)
```

```
CCH_DRY_DEP=>CH_SEAFLUX_MODEL(KTO)%CCH_DRY_DEP
XDEP=>CH_SEAFLUX_MODEL(KTO)%XDEP
CSV=>CH_SEAFLUX_MODEL(KTO)%CSV
NSV_CHSBEG=>CH_SEAFLUX_MODEL(KTO)%NSV_CHSBEG
NSV_CHSEND=>CH_SEAFLUX_MODEL(KTO)%NSV_CHSEND
NSV_DSTBEG=>CH_SEAFLUX_MODEL(KTO)%NSV_DSTBEG
NSV_DSTEND=>CH_SEAFLUX_MODEL(KTO)%NSV_DSTEND
NSV_SLTBEG=>CH_SEAFLUX_MODEL(KTO)%NSV_SLTBEG
NSV_SLTEND=>CH_SEAFLUX_MODEL(KTO)%NSV_SLTEND
NSV_AERBEG=>CH_SEAFLUX_MODEL(KTO)%NSV_AERBEG
NSV_AEREND=>CH_SEAFLUX_MODEL(KTO)%NSV_AEREND
```

```
USE PARKIND1 , ONLY : JPRB
!
IMPLICIT NONE
TYPE CH_SEAFLUX_t
!
  CHARACTER(LEN=6)           :: CCH_DRY_DEP           ! deposition scheme
  REAL, DIMENSION(:,:), POINTER :: XDEP =>NULL()     ! final dry deposition
  CHARACTER(LEN=6), DIMENSION(:), POINTER :: CSV =>NULL() ! velocity for sea
  INTEGER                     :: NSV_CHSBEG, NSV_CHSEND ! name of the scalar var.
  INTEGER                     :: NSV_DSTBEG, NSV_DSTEND ! chemical begin and ending
  INTEGER                     :: NSV_SLTBEG, NSV_SLTEND ! index of the HSV/CSV array
  INTEGER                     :: NSV_AERBEG, NSV_AEREND ! dust begin and ending
  INTEGER                     :: NSV_AEREND           ! sea salt begin and ending
  INTEGER                     :: NSV_AEREND           ! aerosol begin and ending
  INTEGER                     :: NSV_AEREND           ! number of chemical species
  INTEGER                     :: NSV_AEREND           ! number of dust species
  INTEGER                     :: NSV_AEREND           ! number of sea salt species
  INTEGER                     :: NSV_AEREND           ! number of aerosol species
  CHARACTER(LEN=6), DIMENSION(:), POINTER :: CCH_NAMES=>NULL() ! in the surface scheme
  CHARACTER(LEN=6), DIMENSION(:), POINTER :: CCH_NAMES=>NULL() ! NAME OF CHEMICAL SPECIES
  CHARACTER(LEN=6), DIMENSION(:), POINTER :: CDSTNAMES=>NULL() ! (FOR DIAG ONLY)
  CHARACTER(LEN=6), DIMENSION(:), POINTER :: CSLTNAMES=>NULL()
  CHARACTER(LEN=6), DIMENSION(:), POINTER :: CAER_NAMES=>NULL()
!
END TYPE CH_SEAFLUX_t
```

```
TYPE(CH_SEAFLUX_t), ALLOCATABLE, TARGET, SAVE :: CH_SEAFLUX_MODEL(:)
```

```
TYPE(CH_SEAFLUX_t), POINTER :: CH_SEAFLUX => NULL()
!SOMP THREADPRIVATE(CH_SEAFLUX)
```

CONTAINS

```
SUBROUTINE CH_SEAFLUX_GOTO_MODEL(KFROM, KTO, LKFROM)
LOGICAL, INTENT(IN) :: LKFROM
INTEGER, INTENT(IN) :: KFROM, KTO
REAL(KIND=JPRB) :: ZHOOK_HANDLE
```

```
! Current model is set to model KTO
IF (LHOOK) CALL DR_HOOK('MODD_CH_SEAFLUX_N:CH_SEAFLUX_GOTO_MODEL',0,ZHOOK_HANDLE)
```

```
CH_SEAFLUX => CH_SEAFLUX_MODEL(KTO)
```

```
IF (LHOOK) CALL DR_HOOK('MODD_CH_SEAFLUX_N:CH_SEAFLUX_GOTO_MODEL',1,ZHOOK_HANDLE)
```

```
END SUBROUTINE CH_SEAFLUX_GOTO_MODEL
```

```
SUBROUTINE CH_SEAFLUX_ALLOC(KMODEL)
```

```
INTEGER, INTENT(IN) :: KMODEL
REAL(KIND=JPRB) :: ZHOOK_HANDLE
IF (LHOOK) CALL DR_HOOK('MODD_CH_SEAFLUX_N:CH_SEAFLUX_ALLOC',0,ZHOOK_HANDLE)
ALLOCATE(CH_SEAFLUX_MODEL(KMODEL))
CH_SEAFLUX_MODEL(:)%CCH_DRY_DEP=' '
CH_SEAFLUX_MODEL(:)%NSV_CHSBEG=0
CH_SEAFLUX_MODEL(:)%NSV_CHSEND=0
CH_SEAFLUX_MODEL(:)%NSV_DSTBEG=0
CH_SEAFLUX_MODEL(:)%NSV_DSTEND=0
CH_SEAFLUX_MODEL(:)%NSV_SLTBEG=0
CH_SEAFLUX_MODEL(:)%NSV_SLTEND=0
CH_SEAFLUX_MODEL(:)%NSV_AERBEG=0
CH_SEAFLUX_MODEL(:)%NSV_AEREND=0
CH_SEAFLUX_MODEL(:)%NSV_AEREND=0
CH_SEAFLUX_MODEL(:)%NSV_AEREND=0
```

2. Transformation des sous-routines utilisant ces modules

- sur la ligne « USE MODD_ », remplacement des variables globales par la structure
- pré-fixage des variables globales par le nom de la structure

<pre> !* 0. DECLARATIONS !----- USE MODD TYPE DATE_SURF USE MODD SEAFLOW n, ONLY : XSST_INI, TTIME, JSX USE MODD DATA SEAFLOW n, ONLY : NTIME, XDATA_SST, TDATA_SST USE MODI_TEMPORAL_DIST USE MODI_TEMPORAL_LTS ! ! USE YOMHOOK ,ONLY : LHOOK, DR_HOOK USE PARKIND1 ,ONLY : JPRB ! IMPLICIT NONE ! !* 0.1 declarations of arguments ! REAL, DIMENSION(:), INTENT(INOUT) :: PSST ! sst ! !* 0.2 declarations of local variables ! INTEGER :: IDECADE ! decade of simulation INTEGER :: JTIME ! decade of simulation TNTPGFR, SAVF :: JT INTEGER :: JXP REAL, DIMENSION(SIZE(PSST)) :: ZSST REAL, SAVE :: ZSDTJX REAL :: ZDT, ZALPHA REAL(KIND=JPRB) :: ZHOOK_HANDLE !----- ! IF (LHOOK) CALL DR_HOOK('PREP_SST_INIT',0,ZHOOK_HANDLE) LOOP: DO JI = NTIME-1,1,-1 JSX = JI IF (.NOT. TEMPORAL_LTS(TTIME,TDATA_SST(JSX))) EXIT LOOP ENDDO LOOP IF (TEMPORAL_LTS (TTIME, TDATA_SST(JSX))) THEN ZSST(:) = XDATA_SST(:,JSX) ELSE IF (.NOT. TEMPORAL_LTS (TTIME, TDATA_SST(NTIME))) THEN ZSST(:) = XDATA_SST(:,NTIME) ELSE CALL TEMPORAL_DIST (TDATA_SST(JSX+1)%DATE%YEAR,TDATA_SST(JSX+1)%DATE%MONTH, & TDATA_SST(JSX+1)%DATE%DAY ,TDATA_SST(JSX+1)%TIME, & TDATA_SST(JSX)%DATE%YEAR, TDATA_SST(JSX)%DATE%MONTH, & TDATA_SST(JSX)%DATE%DAY ,TDATA_SST(JSX)%TIME, & ZSDTJX) CALL TEMPORAL_DIST (TTIME%DATE%YEAR ,TTIME%DATE%MONTH, & TTIME%DATE%DAY , TTIME%TIME, & TDATA_SST(JSX)%DATE%YEAR,TDATA_SST(JSX)%DATE%MONTH, & TDATA_SST(JSX)%DATE%DAY ,TDATA_SST(JSX)%TIME, & ZDT) ! ZALPHA = ZDT / ZSDTJX ! ZSST(:) = XDATA_SST(:,JSX)+(XDATA_SST(:,JSX+1)-XDATA_SST(:,JSX))*ZALPHA </pre>	<p>→ ←</p>	<pre> !* 0. DECLARATIONS !----- USE MODD TYPE DATE_SURF USE MODD SEAFLOW n, ONLY : S => SEAFLOW USE MODD DATA SEAFLOW n, ONLY : DTS => DATA_SEAFLOW USE MODI_TEMPORAL_DIST USE MODI_TEMPORAL_LTS ! ! USE YOMHOOK ,ONLY : LHOOK, DR_HOOK USE PARKIND1 ,ONLY : JPRB ! IMPLICIT NONE ! !* 0.1 declarations of arguments ! REAL, DIMENSION(:), INTENT(INOUT) :: PSST ! sst ! !* 0.2 declarations of local variables ! INTEGER :: IDECADE ! decade of simulation INTEGER :: JTIME ! decade of simulation TNTPGFR, SAVF :: JT INTEGER :: JXP REAL, DIMENSION(SIZE(PSST)) :: ZSST REAL, SAVE :: ZSDTJX REAL :: ZDT, ZALPHA REAL(KIND=JPRB) :: ZHOOK_HANDLE !----- ! IF (LHOOK) CALL DR_HOOK('PREP_SST_INIT',0,ZHOOK_HANDLE) LOOP: DO JI = DTS%NTIME-1,1,-1 S%JSX = JI IF (.NOT. TEMPORAL_LTS(S%TTIME,DTS%TDATA_SST(S%JSX))) EXIT LOOP ENDDO LOOP IF (TEMPORAL_LTS (S%TTIME, DTS%TDATA_SST(S%JSX))) THEN ZSST(:) = DTS%XDATA_SST(:,S%JSX) ELSE IF (.NOT. TEMPORAL_LTS (S%TTIME, DTS%TDATA_SST(DTS%NTIME))) THEN ZSST(:) = DTS%XDATA_SST(:,DTS%NTIME) ELSE CALL TEMPORAL_DIST (DTS%TDATA_SST(S%JSX+1)%DATE%YEAR,DTS%TDATA_SST(S%JSX+1)%DATE%MONTH, & DTS%TDATA_SST(S%JSX+1)%DATE%DAY ,DTS%TDATA_SST(S%JSX+1)%TIME, & DTS%TDATA_SST(S%JSX)%DATE%YEAR, DTS%TDATA_SST(S%JSX)%DATE%MONTH, & DTS%TDATA_SST(S%JSX)%DATE%DAY ,DTS%TDATA_SST(S%JSX)%TIME, & ZSDTJX) CALL TEMPORAL_DIST (S%TTIME%DATE%YEAR ,S%TTIME%DATE%MONTH, & S%TTIME%DATE%DAY , S%TTIME%TIME, & DTS%TDATA_SST(S%JSX)%DATE%YEAR, DTS%TDATA_SST(S%JSX)%DATE%MONTH, & DTS%TDATA_SST(S%JSX)%DATE%DAY ,DTS%TDATA_SST(S%JSX)%TIME, & ZDT) ! ZALPHA = ZDT / ZSDTJX ! ZSST(:) = DTS%XDATA_SST(:,S%JSX)+(DTS%XDATA_SST(:,S%JSX+1)-DTS%XDATA_SST(:,S%JSX))*ZALPHA </pre>
---	------------	--

Résultats de cette première phase :

- le nombre d'instructions OMP THREADPRIVATE restantes est de 473 (sur 2789)
- le nombre de fichiers impactés est de 598 (sur 1542)

Phase 2 : Passage par argument des structures

1. Transformation des sous-routines utilisant ces modules
 - mise en argument des structures
 - sur la ligne « USE MODD_ », remplacement de la structure par le nom du TYPE
 - déclaration des structures

```
! #####
SUBROUTINE PREP_SST_INIT(PSST)
! #####
!!** *SST_UPDATE*
!!
!! PURPOSE
!! -----
!!
!! performs the time evolution of sst
!!
!!** METHOD
!! -----
!!
!! EXTERNAL
!! -----
!!
!! none
!!
!! IMPLICIT ARGUMENTS
!! -----
!!
!! none
!!
!! REFERENCE
!! -----
!!
!!
!! AUTHOR
!! -----
!!
!! P. Le Moigne      * Meteo-France *
!!
!! MODIFICATIONS
!! -----
!!
!! Original    09/2007
!!
!! -----
!!
!! 0.  DECLARATIONS
!! -----
!!
!!
!! USE MODD_TYPE_DATE_SURF
!! USE MODD_SEAFLUX_n, ONLY : S => SEAFLUX
!! USE MODD_DATA_SEAFLUX_n, ONLY : DTS => DATA_SEAFLUX
!! USE MODI_TEMPORAL_DISTs
!! USE MODI_TEMPORAL_LTS
!!
!!
!! USE YOMHOOK ,ONLY : LHOOK, DR_HOOK
!! USE PARKIND1 ,ONLY : JPRB
!!
!! IMPLICIT NONE
!!
!!* 0.1  declarations of arguments
!!
!!
!! REAL, DIMENSION(:), INTENT(INOUT) :: PSST ! sst
!!
!!* 0.2  declarations of local variables
!!
!!
!! INTEGER          :: IDECADE ! decade of simulation
!! INTEGER          :: JTIME   ! decade of simulation
! #####

SUBROUTINE PREP_SST_INIT (DTS, S, &
PSST)
! #####
!!** *SST_UPDATE*
!!
!! PURPOSE
!! -----
!!
!! performs the time evolution of sst
!!
!!** METHOD
!! -----
!!
!! EXTERNAL
!! -----
!!
!! none
!!
!! IMPLICIT ARGUMENTS
!! -----
!!
!! none
!!
!! REFERENCE
!! -----
!!
!!
!! AUTHOR
!! -----
!!
!! P. Le Moigne      * Meteo-France *
!!
!! MODIFICATIONS
!! -----
!!
!! Original    09/2007
!!
!! -----
!!
!! 0.  DECLARATIONS
!! -----
!!
!!
!! USE MODD_DATA_SEAFLUX_n, ONLY : DATA_SEAFLUX_t
!! USE MODD_SEAFLUX_n, ONLY : SEAFLUX_t
!!
!!
!! USE MODD_TYPE_DATE_SURF
!! USE MODI_TEMPORAL_DISTs
!! USE MODI_TEMPORAL_LTS
!!
!!
!! USE YOMHOOK ,ONLY : LHOOK, DR_HOOK
!! USE PARKIND1 ,ONLY : JPRB
!!
!! IMPLICIT NONE
!!
!!* 0.1  Declaration of arguments
!!
!!
!! TYPE(DATA_SEAFLUX_t), INTENT(INOUT) :: DTS
!! TYPE(SEAFLUX_t), INTENT(INOUT) :: S
!!
!!
!! REAL, DIMENSION(:), INTENT(INOUT) :: PSST ! sst
! #####
```

2. Transformation des modules

- suppression du pointer de structure introduit précédemment
- suppression de la subroutine `_GOTO_MODEL`

Modules particuliers avec **THREADPRIVATE**

- les modules `MODD_PACK_*` : absence de TYPE => à introduire et traiter comme les autres
- les modules `MODD_IO_SURF_*`
`MODD_MASK` => à traiter à la fin
`MODD_SURFEX_MPI` et `_OMP`
- les « nouveaux » modules (version 7.3) définissant plusieurs TYPE :
`MODD_BEM_n`
`MODD_DIAG_MISC_TEB_n`
`MODD_TEB_GARDEN_n` => splités en 2 ou 4 pour définir un seul TYPE par module
`MODD_TEB_GREENROOF_n`
`MODD_TEB_n`

mais il apparait dans ces modules des tableaux 2D de TYPE :

```
TYPE(BEM_t), ALLOCATABLE, TARGET, SAVE :: BEM_MODEL(:,:)
```

avec

```
SUBROUTINE BEM_GOTO_MODEL(KFROM, KTO, LKFROM, KFROM_PATCH, KTO_PATCH)
```