Oracle Solaris Studio

C, C++, and Fortran Compilers

- C, C++ and Fortran компиляторы обладающие возможностью расширенной оптимизации, для увеличения производительности приложений на Oracle системах.
- Генерируют код, который в 4.8 раза быстрее на Oracle системах по сравнению с открытыми системами.
- Поддерживают последние стандарты, включая C++ 2011 и OpenMP 4.0

Debugger

- Расширенный отладчик, особенности которого помогают разработчикам быстро определять местонахождение ошибок в одно и много-поточных приложениях.
- Помогает контролировать приложения построенные Oracle Solaris Studio или GNU компиляторами
- Доступ к нему обеспечивается посредством командной строки, интегрированной в IDE или посредством графической оболочки.

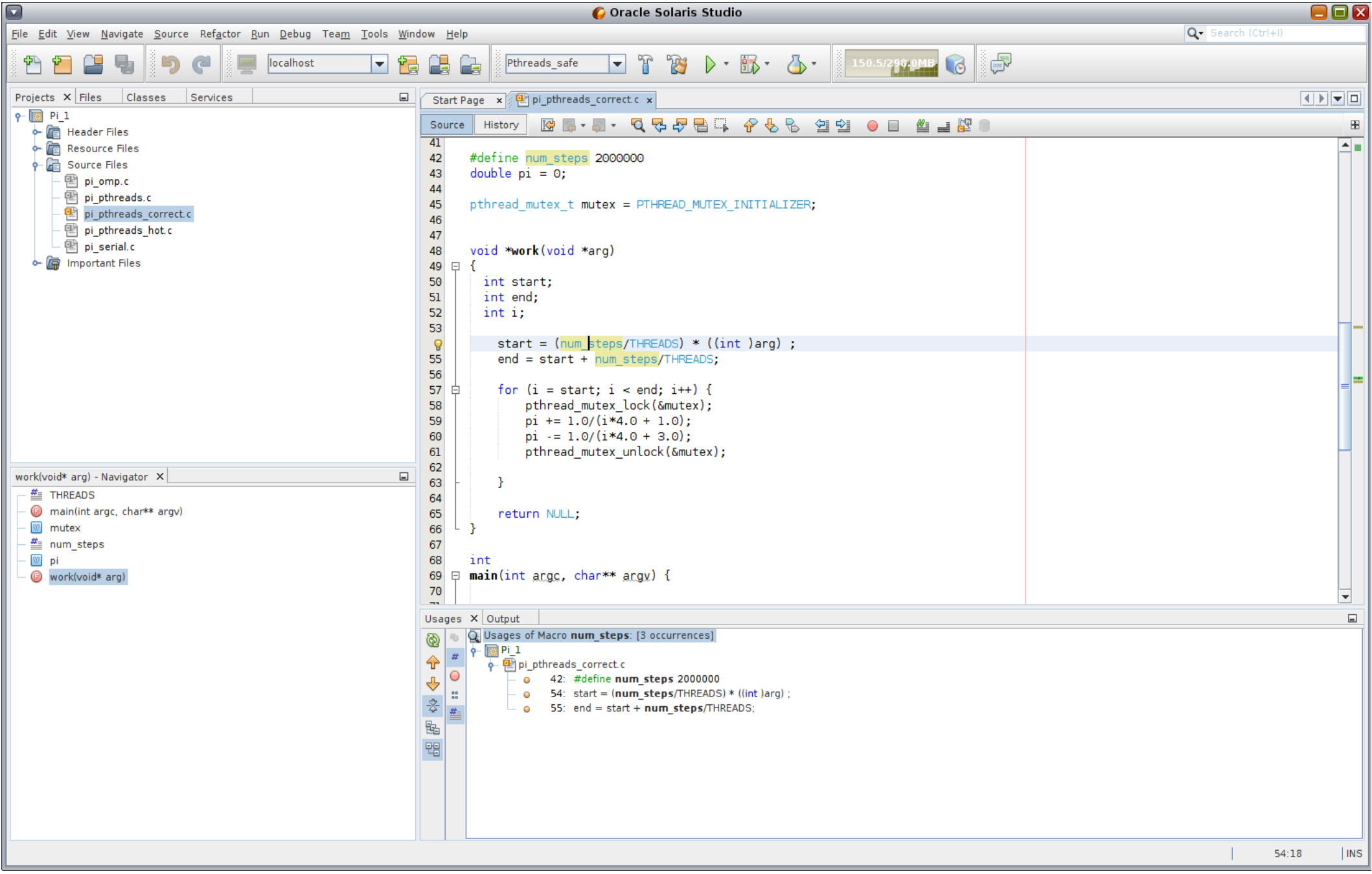
Performance Library

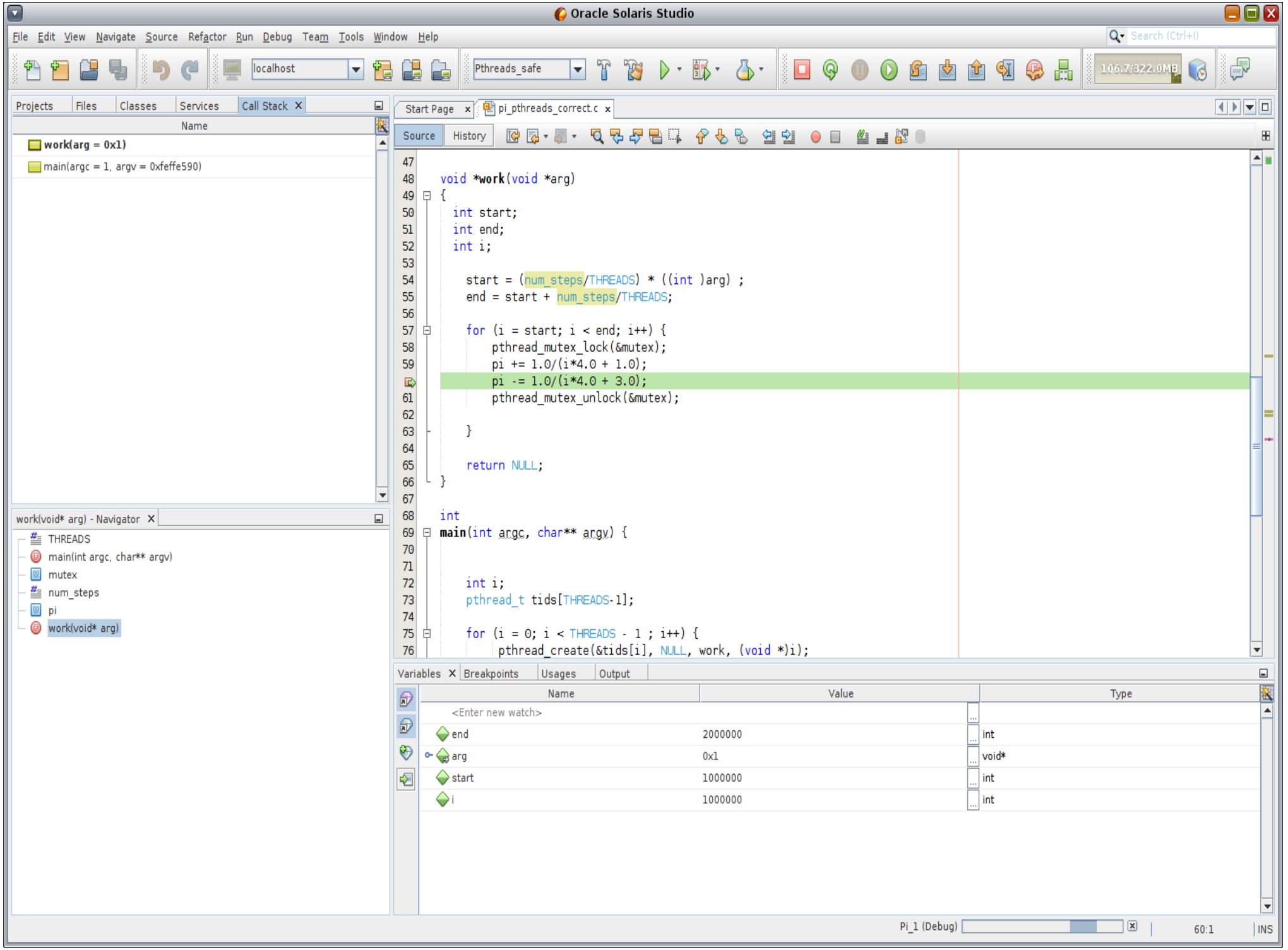
- Совокупность современных численных методов в библиотеках, которые максимизируют производительность объемных вычислительных задач.
- Данная библиотека отлажена и оптимизирована под последние Oracle системы (SPARC & x86)
- Позволяет использовать особенности параллелизации Oracle Solaris Studio C, C++ and Fortran компиляторов

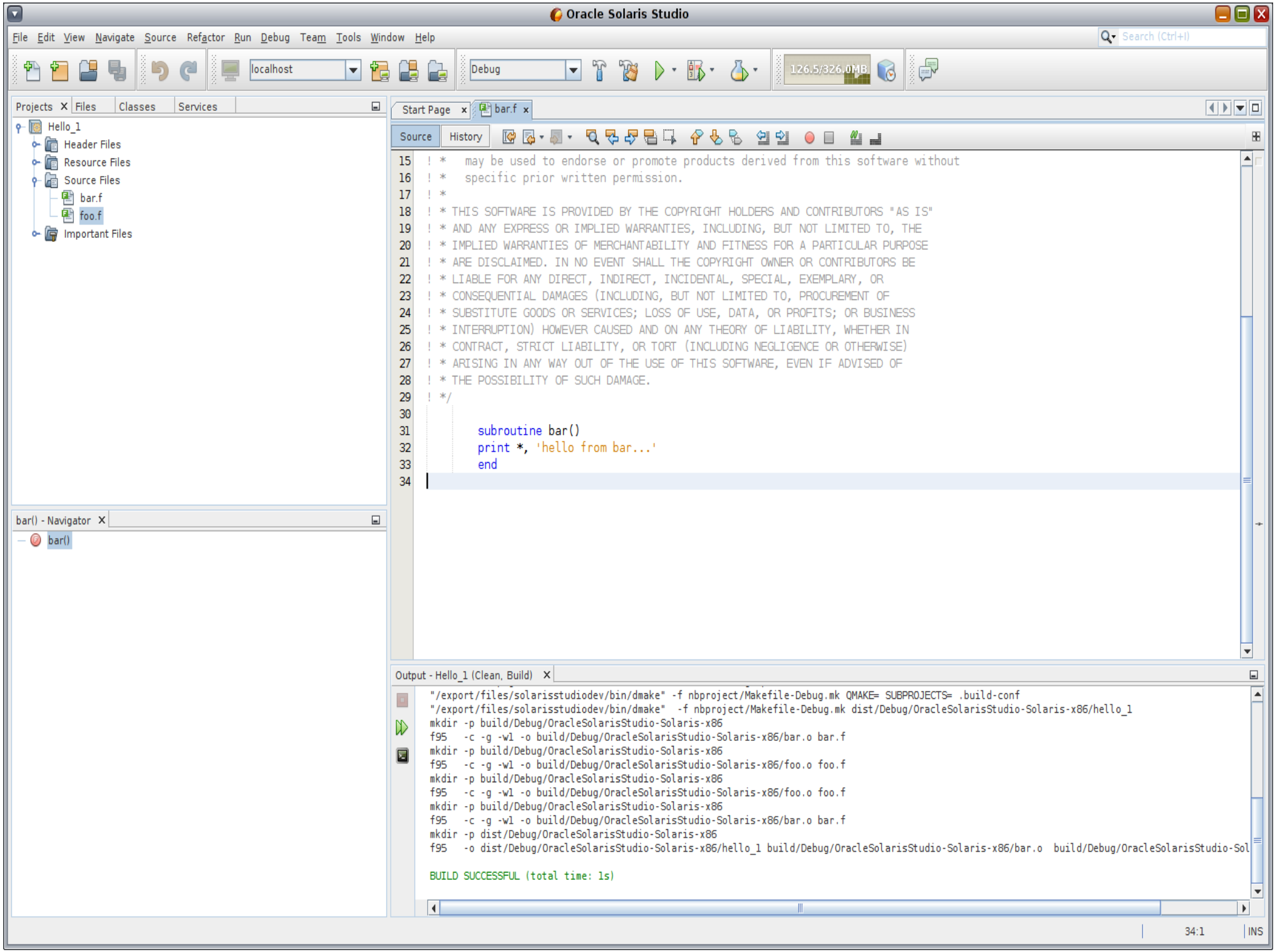
- Performance Analyzer
  - Инструмент профилирования C, C++, Java and Fortran приложения для быстрого определения узких мест в производительности оптимизированного и параллельного кода.
  - Его богатый пользовательский интерфейс дает всестороннее отображение данных и возможности расширенной фильтрации, сортировки, навигации и отображение хода выполнения приложения.
  - Увеличить эффективность поддержки удаленного анализа, который позволяет Вам профилировать приложения и видеть запуски на удаленном сервере с любого виртуального окружения.
- Code Analyzer
  - Позволяет определять общие ошибки кодирования, включая утечки памяти и улучшает надежность, безопасность и качество приложения.
  - Синтезировать данные собранные посредством статического анализа вовремя сборки приложения, динамический анализ при выполнении и анализ покрытия кода.
  - Его результаты могут отображаться посредством взаимодействия с пользовательским интерфейсом или интерфейсом командной строкой.
- Thread Analyzer
  - Определяет тяжело обнаруживаемые ошибки типа race conditions и deadlocks в многопоточных, параллельных приложениях.
  - Может запускать бинарные файлы, исключая необходимость пересборки с инструментированием
  - Поддерживает приложения написанные с использование Oracle Solaris threads, P-threads, или OpenMP

- IDE

- Основанная на NetBeans IDE для работы на C, C++ и Fortran языках




Oracle Solaris Studio
File Edit View Navigate Source Refactor Run Debug Team Tools Window Help
Search (Ctrl+I)
localhost
Pthreads_safe
106.7/322.0MB
Projects Files Classes Services Call Stack
Name
work(arg = 0x1)
main(argc = 1, argv = 0xfeffe590)
work(void* arg) - Navigator
THREADS
main(int argc, char** argv)
mutex
num_steps
pi
work(void* arg)
Start Page
pi_pthreads_correct.c
Source History
47
48 void *work(void *arg)
49 {
50 int start;
51 int end;
52 int i;
53
54 start = (num_steps/THREADS) * ((int)arg) ;
55 end = start + num_steps/THREADS;
56
57 for (i = start; i < end; i++) {
58 pthread_mutex_lock(&mutex);
59 pi += 1.0/(i*4.0 + 1.0);
60 pi -= 1.0/(i*4.0 + 3.0);
61 pthread_mutex_unlock(&mutex);
62
63 }
64
65 return NULL;
66 }
67
68 int
69 main(int argc, char** argv) {
70
71
72 int i;
73 pthread_t tids[THREADS-1];
74
75 for (i = 0; i < THREADS - 1 ; i++) {
76 pthread_create(&tids[i], NULL, work, (void *)i);
Variables Breakpoints Usages Output
Name Value Type
<Enter new watch>
end 2000000 int
arg 0x1 void*
start 1000000 int
i 1000000 int
Pi_1 (Debug)
60:1
INS


Oracle Solaris Studio
File Edit View Navigate Source Refactor Run Debug Team Tools Window Help
Search (Ctrl+I)
localhost
Debug
126.5/326.0MB
Projects
Files
Classes
Services
Hello_1
Header Files
Resource Files
Source Files
bar.f
foo.f
Important Files
bar() - Navigator
bar()
Start Page
bar.f
Source
History
15 ! *   may be used to endorse or promote products derived from this software without
16 ! *   specific prior written permission.
17 ! *
18 ! * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
19 ! * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
20 ! * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
21 ! * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
22 ! * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
23 ! * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
24 ! * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
25 ! * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
26 ! * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
27 ! * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
28 ! * THE POSSIBILITY OF SUCH DAMAGE.
29 ! */
30
31 subroutine bar()
32 print *, 'hello from bar...'
33 end
34
Output - Hello_1 (Clean, Build)
"/export/files/solarisstudiodev/bin/dmake" -f nbproject/Makefile-Debug.mk QMAKE= SUBPROJECTS= .build-conf
"/export/files/solarisstudiodev/bin/dmake" -f nbproject/Makefile-Debug.mk dist/Debug/OracleSolarisStudio-Solaris-x86/hello_1
mkdir -p build/Debug/OracleSolarisStudio-Solaris-x86
f95   -c -g -w1 -o build/Debug/OracleSolarisStudio-Solaris-x86/bar.o bar.f
mkdir -p build/Debug/OracleSolarisStudio-Solaris-x86
f95   -c -g -w1 -o build/Debug/OracleSolarisStudio-Solaris-x86/foo.o foo.f
mkdir -p build/Debug/OracleSolarisStudio-Solaris-x86
f95   -c -g -w1 -o build/Debug/OracleSolarisStudio-Solaris-x86/foo.o foo.f
mkdir -p build/Debug/OracleSolarisStudio-Solaris-x86
f95   -c -g -w1 -o build/Debug/OracleSolarisStudio-Solaris-x86/bar.o bar.f
mkdir -p dist/Debug/OracleSolarisStudio-Solaris-x86
f95   -o dist/Debug/OracleSolarisStudio-Solaris-x86/hello_1 build/Debug/OracleSolarisStudio-Solaris-x86/bar.o build/Debug/OracleSolarisStudio-Sol
BUILD SUCCESSFUL (total time: 1s)
34:1
INS

Fortran Compiler f77, f90, f95

- -C – проверка выхода за границы массива
- -O (O1,...,O5) – уровень оптимизации
- -U – различие верхнего и нижнего регистра
- -ansi – вывод сообщений об использовании расширений языка
- -autopar – разрешает автоматическую параллелизацию циклов
- -dryrun – отображает ключи передаваемые компилятору драйвером
- -e – включает поддержку исходного кода строки длиной 132 символа
- -f77 – fortran 77
- -fast — оптимизация выше -O5
- -fixed – фиксированный формат кода
- -free – свободный формат кода
- -ftrap – обработка исключительных ситуаций
- -g<n> – генерация отладочной информации для dbx

Fortran Compiler f77, f90, f95

- -keepmod – позволяет пересобирать или нет имеющиеся модули
- -library – подключать известные библиотеки
- -m32/-m64 – задает битность модели программы
- -openmp={none|noopt|parallel} – подключение OpenMP.
- -recl={out:N|error:N|all:N} – устанавливаеть длину вывода строки.
- -u – IMPLICIT NONE устанавливается по умолчанию.
- -xfilebyteorder – определяет представление и выравнивание для двоичных файлов.
- -xlibmil – разрешает встраивание выбранных libm функций для оптимизации.
- -xtypemap – определение типов данных по умолчанию (integer:64,real:32,double:128).
- -xipo – межпроцедуральная оптимизация.
- -xia – использование интервальной арифметики

Fortran 2003

- Производные типы данных (Type … End Type)
- Абстрактные типы данных (Abstract)
- Наследование (Extends)
- Права доступа (PUBLIC, PRIVATE, PROTECTED)
- Type–bound procedures – процедуры и функции производного типа данных (аналог процедур членов класса в C++)
- Процедуральные указатели (Procedure(type), Pointer :: Var_Name)
- Полиморфные типы данных (Class, Select Type, Extentds_Type_Of, Same_Type_As)
- Final – процедура финализации (аналог деструктора)
- Import – используется, когда тип данных недоступен в интерфейсе
- Allocate – динамическое выделение памяти (Allocatable, Pointer, Allocate, Deallocate, Allocated, Associated)
- Associate – конструкция, позволяющая сократить длинные выражения

- Enumerations and enumerators – аналог ENUM в языке C (перечисляемый тип данных, когда интуитивно-понятному имени сопоставляется целое число, тип совместим с C посредством BINB(C))
- – IEEE modules – IEEE_EXCEPTIONS, IEEE_ARITHMETIC и IEEE_FEATURES
- – ISO_FORTRAN_ENV module – определяет постоянные связанные с окружением Fortran-а (процессорно-зависимые значения для устройств ввода-вывода, размеры IO буферов)
- – ISO_C_BINDING module – содержит описание типов сопоставимых в языке C

==========

- Параметризованный Format
- Расширения VAX, включая структуры (Structure … End Structure, Record)
- Беззнаковый целый тип (unsigned*(1,2,4,8))
- OpenMP
- Интервальная арифметика
- Набор нестандартных intrinsic функций для real*4, real*8, real*16

```
MODULE POINT_2D_M
  TYPE POINT_2D
    REAL :: X, Y
   contains
    procedure :: asmt => Point_to_Point
    procedure :: add => Point_Point_Add
    generic :: operator(+) => add
    generic :: assignment(=) => asmt
  END TYPE POINT_2D

CONTAINS

  Subroutine Point_To_Point(A,B)
    Class(Point_2D), Intent(OUT) :: A
    Class(Point_2D), Intent(IN) :: B
    A%X=B%X
    A%Y=B%Y
    Return
  End Subroutine Point_To_Point

  Function Point_Point_Add(A,B)
    Class(Point_2D), Intent(IN) :: A
    Class(Point_2D), Intent(IN) :: B
    Class(Point_2D), allocatable :: Point_Point_Add
    allocate(point_point_add)
    Point_Point_Add = Point_2d(A%X+B%X, A%Y+B%Y)
    Return
  End Function Point_Point_Add
END MODULE POINT_2D_M
```

```
MODULE POINT_3D_M
  USE Point_2D_M
  Type,Extends (Point_2D) :: POINT_3D
    REAL :: Z
   contains
    procedure :: asmt => Point_To_Point_3
    procedure :: add => Point_Point_Add_3
  END TYPE POINT_3D

CONTAINS
  Subroutine Point_To_Point_3(A,B)
    Class(Point_3D), Intent(OUT) :: A
    Class(Point_2D), Intent(IN) :: B
    A%X=B%X;    A%Y=B%Y
    select type (B)
      type is (Point_2D) ;          A%Z=0
      class is (Point_3D);          A%Z=B%Z
    end select
    Return
  End Subroutine Point_To_Point_3

  Function Point_Point_Add_3(A,B)
    Class(Point_3D), Intent(IN) :: A
    Class(Point_2D), Intent(IN) :: B
    Class(Point_2D), allocatable :: Point_Point_Add_3
    REAL :: Z
    Z = A%Z
    select type (B)
      class is (Point_3D);          Z = Z + B%Z
    end select
    allocate(Point_3d::point_point_add_3)
    Point_Point_Add_3 = Point_3D(A%X+B%X, A%Y+B%Y, Z)
    Return
  End Function Point_Point_Add_3
END MODULE POINT_3D_M
```

```
Module Vector_3D_Class
  USE Point_3D_m

  Type :: Vector_3D
    Type(Point_3D) :: A, B
   contains
    procedure :: asmt => Vector_To_Vector_3
    procedure :: plus => Vector_To_Vector_Add_3
    generic :: operator(+) => plus
    generic :: assignment(=) => asmt
  End Type Vector_3D

CONTAINS

  Subroutine Vector_To_Vector_3(A,B)
    Class(Vector_3D), Intent(IN)  :: B
    Class(Vector_3D), Intent(OUT) :: A
    A%A=B%A
    A%B=B%B
    Return
  End Subroutine Vector_To_Vector_3

  Function  Vector_To_Vector_Add_3(A,B)
    Class(Vector_3D), allocatable :: Vector_To_Vector_Add_3
    Class(Vector_3D), Intent(IN) :: A,B

    Type(Point_3D) :: X, Y

    X = A%A+B%A
    Y = A%B+B%B
    allocate(Vector_3D::Vector_To_Vector_Add_3)
    Vector_To_Vector_Add_3 = Vector_3D(X, Y)
    Return
  End Function Vector_To_Vector_Add_3
End Module Vector_3D_Class
```

```
Program OOF
  USE Point_3d_m
  USE Vector_3d_Class

  Type(Vector_3D) :: V31=Vector_3d(Point_3d(1.,2.,3.),Point_3d(3.,2.,1.))
  Type(Vector_3D) :: V32=Vector_3d(Point_3d(3.,4.,5.),Point_3d(5.,6.,7.))
  Type(Vector_3D) :: V33
  V33=V31+V32
  write(*,*)V33

End Program OOF
```

```
4.0 6.0 8.0 8.0 8.0 8.0
```

- TYPE matrix (k, b)
- INTEGER, KIND :: k = 4
- INTEGER (8), LEN :: b
- REAL (k) :: element (b,b)
- END TYPE matrix
- TYPE (matrix (8, 10)) :: square
- TYPE (matrix (10)) :: square
- ----------
- TYPE matrix (k, d1, d2)
- INTEGER, KIND :: k = kind (0.0)
- INTEGER (selected_int_kind (12)), LEN :: d1, d2
- REAL (k) :: element (d1 ,d2)
- END TYPE
- TYPE(matrix(k = KIND(0d0), d1=200+5, d2=dim)) :: my_matrix1
- TYPE(matrix(d1=2*dim, d2=dim)) :: my_matrix2
- TYPE(matrix(KIND(0d0), :, :)), pointer :: my_deferred_matrix
- TYPE(matrix(KIND(0d0), *, *)) :: my_assumed_matrix

- Matrix (kind (0.0),1,3) :: my_matrix
- my_matrix = matrix (kind (0.0),1,3) ([1.0,2.0,3.0])
- ----------
- Type :: t1 (k1,k2)
- Integer, kind :: k1,k2
- Real(k1) :: a(k2)
- End type
-
- Type, extends (t1) :: t2(k3)
- Integer, kind :: k3
- Logical(k3) flag
- End type
-

- TYPE(matrix(k=KIND(0.0), d1= :, d2= :)), pointer :: my_mtrx_ptr, my_mtrx_alloc
- TYPE(matrix(KIND(0.0), 100, 200)), target :: my_mtrx_tgt
- TYPE(matrix(KIND(0.0), 1, 2)) :: my_mtrx_src
-
- my_mtrx_ptr => my_mtrx_tgt ! Gets values from target
- ! my_mtrx_ptr has d1= 100 and d2 = 200.
-
- ALLOCATE(matrix(KIND(0.0), 10, 20) :: my_mtrx_alloc) ! Gets values from allocation
- ! my_mtrx_alloc has d1=10 and d2=20
- DEALLOCATE(my_mtrx_alloc)
-
- ALLOCATE(my_mtrx_alloc, source=my_mtrx_src) ! Gets values from allocation
- ! my_mtrx_alloc has d1=1 and d2=2

- Type(matrix(KIND(0d0), 10, :)), pointer :: y(:)
- Call print_matrix(y)
- ...
- Subroutine print_matrix(x)
-
- ! d1 here is assumed and its value '10' is obtained from the actual argument
- Type(matrix(k= KIND(0d0), d1=*, d2=:), pointer :: x(:)
- ALLOCATE(matrix(KIND(0.0), *, 10) :: x(10))
- ...
- End Subroutine

```
Type, Abstract :: One
Integer, Public, Allocatable :: I
End Type One

Type, Extends(One) :: Two
Integer, Allocatable, Public :: J
End Type Two

Type(Two), Target :: t_1, t_2
Class(Two), Pointer :: ct
Class(One), Pointer :: co

t_1%i=11; t_1%j=12
t_2%i=-11; t_2%j=-12

ct=>t_1
co=>t_2

write(*,*)'-= t_1 =-',t_1%i,t_1%j
write(*,*)'-= t_2 =-',t_2%i,t_2%j

write(*,*)ct%i,ct%j
write(*,*)co%i
end
```

```
-= t_1 =- 11 12
-= t_2 =- -11 -12
11 12
-11
```

```
module M
    implicit real(a)
    type t
    private
     integer n
    end type
contains

subroutine process(x,y,result,monitor)
type(t), intent(in) :: x(:,:), y(:,:)
type(t), intent(out) :: result(:,:)
interface
    subroutine monitor(iteration_number, current_estimate)
Import  t
integer, intent(in) :: iteration_number
type(t), intent(in) :: current_estimate(:,:)
    end subroutine
end interface

do i=1,10
   do j = 1,10
       result(i,j)%n = (j-1)*10 + i
   enddo
enddo

call monitor( 5, result )
end subroutine
```

```
subroutine print_arr( arr )
type(t), intent(in) :: arr(:,:)
  print 10, arr
10 format(20i3)
end subroutine
end module

subroutine add_iter( n, est )
use M
type (t) est(:,:)

do i = 1,n
end do
end subroutine

 use M
interface
    subroutine add_iter(iteration_number, current_estimate)
import
integer, intent(in) :: iteration_number
type(t), intent(in) :: current_estimate(:,:)
    end subroutine
end interface
type (t) a(10,10), b(10,10), r(10,10)

call process( a, b, r, add_iter )
call print_arr(r)
end

```

```
Module A
Type Type_A
 Integer, Allocatable :: IA
 Integer, Allocatable :: IB
Contains
 FINAL :: DE_A
End Type Type_A

Contains

Subroutine DE_A(X)
Type(Type_A) :: X
Deallocate(X%IA)
Deallocate(X%IB)
write(*,*)'IA&IB components have been deallocated'
Return
End Subroutine DE_A

End Module A

Use A
Type(Type_A) :: XA

Allocate( XA%IA, SOURCE = 15 )
Allocate( XA%IB, SOURCE = 35 )

Write(*,*)XA%IA, XA%IB

End
```

```
bash-4.1$ ./a.out
 15 35
 IA&IB components have been deallocated
```

```
REAL :: MYREAL, X, Y, THETA, A

Real :: z=-1., v=-2.

X = 0.42

Y = 0.35

MYREAL = 9.1

THETA = 1.5

A = 0.4


ASSOCIATE ( Z => EXP(-(X**2+Y**2)) * COS(THETA), V
=> MYREAL)

        PRINT *, A+Z, A-Z, V

        V = V * 4.6

END ASSOCIATE


      PRINT *, MYREAL

      Print *, Z, V

End
```

```
 0.4524611      0.3475389      9.100000
  41.86000
-1.000000      -2.000000
```

- Module Describe
-
- enum, bind(c)
- enumerator :: sunday, monday, tuesday, wednesday, thursday, friday, saturday
- end enum
-
- Type :: Object
- Integer :: i = -1
- End Type Object
- End Module Describe
-
- use Describe
-
- Type(Object), Parameter :: Def_Const_O=Object(sunday)
- Type(Object), Parameter :: Const_O_1=Object(wednesday)
-
- Type(Object) :: O1, O2=Object(friday), O3, O4
-
- O1=Def_Const_O
- O2=Const_O_1
- O3=Object(saturday)
- O4%i=monday
-
- write(*,*)O1
- write(*,*)O2
- write(*,*)O3
- write(*,*)O4
- end

- ./a.out
- 0
- 3
- 6
- 1

```
#include <stdio.h>
extern void vecref( float _Complex[2][3][3], float _Complex *, int );
void main ()
{
  int i, k, j;
  float f[2][3][6] = {
                { {1.0, -2.0, 3.0,  -1.0, 2.0, -3.0},  {4.0, -5.0, 6.0,  -4.0, 5.0, -6.0},
                {7.0, -8.0, 9.02, -7.0, 8.0, -9.01} } , { {-1.0, 2.0, -3.0,  1.0, -2.0, 3.0},
                {-4.0, 5.0, -6.0,  4.0, -5.0, 6.0}, {-7.0, 8.0, -9.01, 7.0, -8.0, 9.02} } } ;
  float _Complex v[2][3][3];
  float *sijk = (float*) &v ;
  float _Complex sum ;
  float *s = (float*) &sum ;

  for (k=0; k<2; k++)
   for (i=0; i<3; i++)
    for (j=0; j<6; j++)
     *sijk++ = f[k][i][j];

  sijk = (float*) &v ;
  *s = -5.0f; *(s+1) = 0.0f;
  for (k=0; k<2; k++)
   for (i=0; i<3; i++)
   {
    for (j=0; j<3; j++)
     printf ( " (%.2f, %.2f)", *sijk++, *sijk++ );
    printf ( "\n") ;
   }

  vecref( v, &sum, k );
  printf ( " sum=(%.2f, %.2f)\n", *s++, *s ) ;
}
```

```
      subroutine VecRef( v, total, n), bind(c)
      use, intrinsic :: iso_c_binding, only : C_INT, C_FLOAT_COMPLEX
      integer(C_INT) :: i
      integer(C_INT),value :: n
      complex(C_FLOAT_COMPLEX) :: total, v(3,3,n)
      total = 0.0
      write(66,*) 'v=',v
      write(66,*) total
      write(66,*) n
      do 1 i = 1,n
      do 1 j = 1,3
      do 1 k = 1,3
1           total = total + conjg(v(k,j,i))
      write(66,*) total,'=total'
      End
./a.out
 (1.00, -2.00) (3.00, -1.00) (2.00, -3.00)
 (4.00, -5.00) (6.00, -4.00) (5.00, -6.00)
 (7.00, -8.00) (9.02, -7.00) (8.00, -9.01)
 (-1.00, 2.00) (-3.00, 1.00) (-2.00, 3.00)
 (-4.00, 5.00) (-6.00, 4.00) (-5.00, 6.00)
 (-7.00, 8.00) (-9.01, 7.00) (-8.00, 9.02)
 sum=(0.01, -0.01)
```

- cat ce1-2.f95
- INTERVAL :: X = [2, 3], Y = [4, 5] ! Line 1
- PRINT *, "[2, 3] + [4, 5] = ", X+Y ! Line 2
- END
- f95 -xia ce1-2.f95
- ./a.out
- [2, 3] + [4, 5] = [6.0,8.0]
-
-

- Interval-Specific Operators
- Operator Name Mathematical Symbol
- .IH. Interval Hull
- .IX. Intersection
- .DJ. Disjoint
- .IN. Element
- .SB. Subset
- .SP. Superset
- Interval-Specific Intrinsics
- Infimum inf([a, b]) = a INF
- Supremum sup([a, b]) = b SUP
- Width w([a, b]) = b — a WID
- Midpoint mid([a, b]) = (a + b)/2 MID
- Magnitude max(|a|) ∈A MAG
- Mignitude min(|a|) ∈A MIG
- Test for empty true if A is empty ISEMPTY
- Division with intersection (A/B)∩C DIVIX
- Number of digits NDIGITS

Производительность кода компиляторов

```
#include <stdio.h>

unsigned long fib( int n ) {
    return n < 2 ? 1 : fib( n - 1 ) + fib( n - 2 );
}

int main( int argc, char **argv ) {
    unsigned num = 30;
    printf( "%ld\n", fib( num ) );
    return 0;
}
```

```
recursive function fib(n) result(f)
unsigned*8 :: f
unsigned*8, intent(in) :: n
  if ( n<2 ) then
   f = 1
   return
  else
   f = fib( n - 1 ) + fib( n - 2 )
  end if
end function fib

unsigned*8 n, fib
 n=30U_8
 write(*,*)fib(n)
End
```

- bash-4.1$ gcc c_fib.c
- bash-4.1$ time ./a.out
- 1346269
- real 0m0.019s user 0m0.017s sys 0m0.001s
-
- cc c_fib.c
- bash-4.1$ time ./a.out
- 1346269
- real 0m0.023s user 0m0.020s sys 0m0.000s
-
- bash-4.1$ gcc c_fib.c -O
- bash-4.1$ time ./a.out
- 1346269
- real 0m0.011s user 0m0.010s sys 0m0.000s
-
- cc c_fib.c -O
- bash-4.1$ time ./a.out
- 1346269
- real 0m0.010s user 0m0.007s sys 0m0.000s

- bash-4.1$ gfortran f_fib.f90
- bash-4.1$ time ./a.out
- 1346269
- real 0m0.029s user 0m0.019s sys 0m0.001s
-
- f90 f_fib.f90
- bash-4.1$ time ./a.out
- 1346269
- real 0m0.029s user 0m0.023s sys 0m0.003s
-
- bash-4.1$ gfortran f_fib.f90 -O
- bash-4.1$ time ./a.out
- 1346269
- real 0m0.013s user 0m0.010s sys 0m0.000s
-
- f90 f_fib.f90 -O
- bash-4.1$ time ./a.out
- 1346269
- real 0m0.014s user 0m0.007s sys 0m0.004s

```
bash-4.1$ cat c1_fib.c
#include <stdio.h>

unsigned long fib( int n ) {
  unsigned long f1=1, f2=1, f3;
   for( int i=1; i<n; i++ )
    {
      f3= f1+f2;
      f1=f2;
      f2=f3;
    }
  return(f2);
}

int main( int argc, char **argv ) {
   unsigned num = 30;
   printf( "%ld\n", fib( num ) );
   return 0;
}
```

```
bash-4.1$ cat f1_fib.f90
function fib(n)
integer*8 :: fib, f1, f2
integer*8, intent(in) :: n
 f1=1
 fib=f1
 do i= 3, n+1
  f2= fib+f1
  f1=fib
  fib=f2
 end do
end function fib

integer*8 n, fib
  n=30_8
  write(*,*)fib(n)
End
```

- bash-4.1$ gcc c1_fib.c -std=c99
- bash-4.1$ time ./a.out
- 1346269
- real 0m0.003s user 0m0.000s sys 0m0.000s
-
- bash-4.1$ cc c1_fib.c
- bash-4.1$ time ./a.out
- 1346269
- real 0m0.003s user 0m0.000s sys 0m0.000s
-
- bash-4.1$ gcc c1_fib.c -std=gnu99 -O
- bash-4.1$ time ./a.out
- 1346269
- real 0m0.005s user 0m0.000s sys 0m0.000s
-
- bash-4.1$ cc c1_fib.c -O
- bash-4.1$ time ./a.out
- 1346269
- real 0m0.003s user 0m0.000s sys 0m0.000s

- bash-4.1$ gfortran f1_fib.f90
- bash-4.1$ time ./a.out
- 1346269
- real 0m0.003s user 0m0.000s sys 0m0.000s
-
- bash-4.1$ f90 f1_fib.f90
- bash-4.1$ time ./a.out
- 1346269
- real 0m0.007s user 0m0.000s sys 0m0.004s
-
- bash-4.1$ gfortran f1_fib.f90 -O
- bash-4.1$ time ./a.out
- 1346269
- real 0m0.002s user 0m0.000s sys 0m0.000s
-
- bash-4.1$ f90 f1_fib.f90 -O
- bash-4.1$ time ./a.out
- 1346269
- real 0m0.006s user 0m0.001s sys 0m0.002s

```
#include <stdio.h>
int binary_data[1024];

/* Create a file with 1024 32-bit integers */
int
main(void)
{
    int i;
    FILE *fp;

    for (i = 0; i < 1024; ++i)
        binary_data[i] = i;
    fp = fopen("test", "w");
    fwrite(binary_data, sizeof(binary_data), 1, fp);
    fclose(fp);
}
```

```
program reader
 integer:: a(1024), i, result
 open(file="test", unit=8, access="stream",form="unformatted")
! read all of a
 read(8) a
 do i = 1,1024
   if (a(i) .ne. i-1) print *,'error at ', i
 enddo
! read the file backward
 do i = 1024,1,-1
   read(8, pos=(i-1)*4+1) result
   if (result .ne. i-1) print *,'error at ', i
 enddo
 close(8)
end
```