

《Fortran多核并行计算》

期末考查作业

姓 名：

学 号：

专 业：

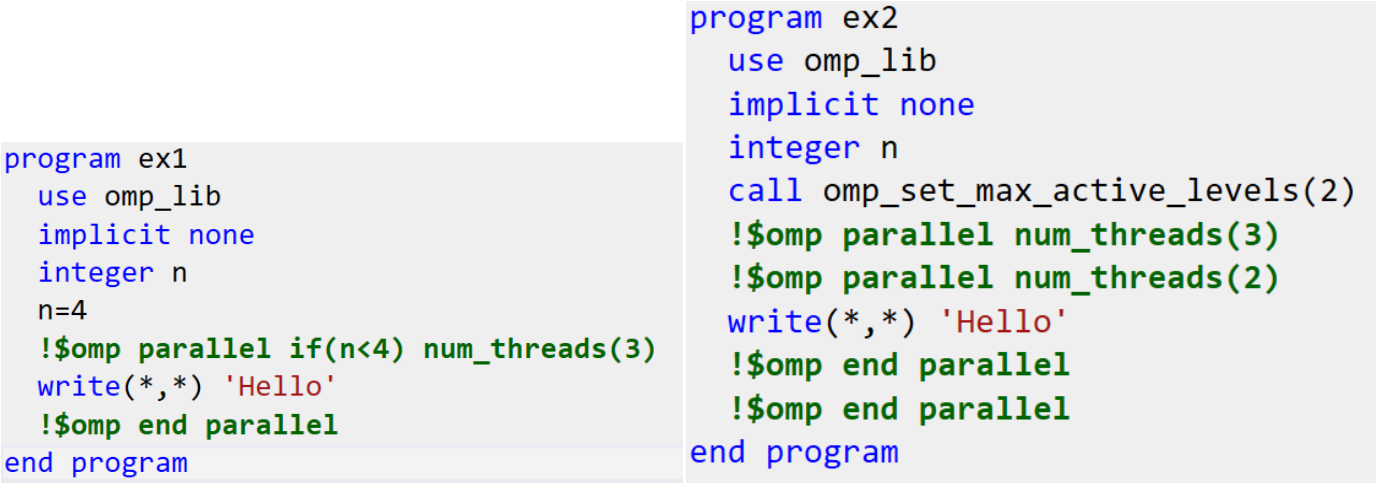
授课教师：

完成时间：2020年12月

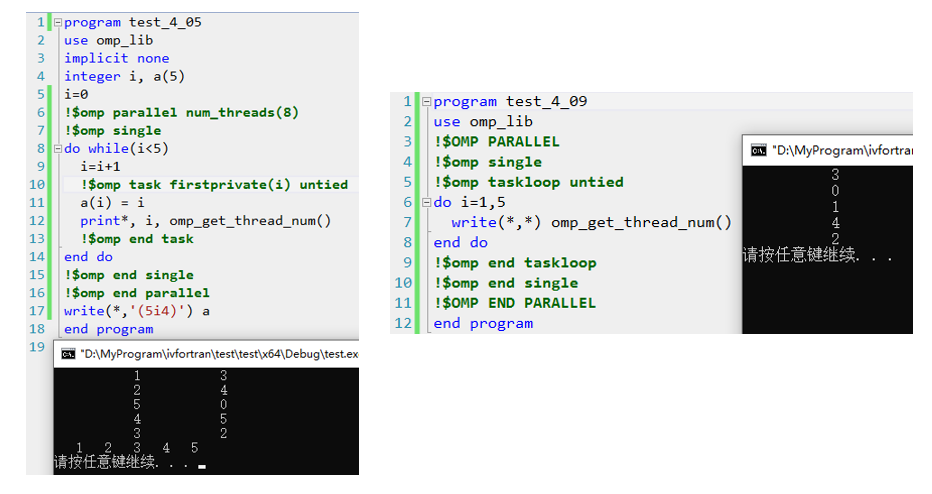
1. **写出常用的三种并行计算方式。（5分）**
2. 单机多核并行：OpenMP
3. 多节点并行：MPI
4. 设备(GPU)加速（内存要求不大，计算量大。针对大量的数据进行相同的操作）：CUDA/OpenACC
5. **说明进程和线程的区别。（10分）**
6. 包含关系：如果一个进程内有多个线程，则执行过程不是一条线的，而是多条线程共同完成的；线程是进程的一部分，所以线程也被称为轻权进程或者轻量级进程。
7. 内存分配：同一进程的线程共享本进程的地址空间和资源，而进程之间的地址空间和资源是相互独立的。
8. 进程和线程的根本区别是进程是操作系统资源分配的基本单位，而线程是处理器任务调度和执行的基本单位。
9. **OpenMP并行的特点。（5分）**

OpenMP是一种用于共享内存系统(单机)的多线程并行方案，支持C/C++和Fortran语言。它通过在串行程序的适当位置添加并行指令，以Fork-Join模式并行执行特定部分的代码。

1. **按照优先级(从高到低)列出设置并行域中线程数的三种方式。（5分）**
2. 子句 num\_threads：设置当前并行域的线程数。
3. 子程序 omp\_set\_num\_threads：设置后续所有并行域的线程数。
4. 环境变量 OMP\_NUM\_THREADS：缺省(默认)线程数，一般等于CPU核数。
5. **以下两组代码中“Hello”会输出几次，为什么？（10分）**



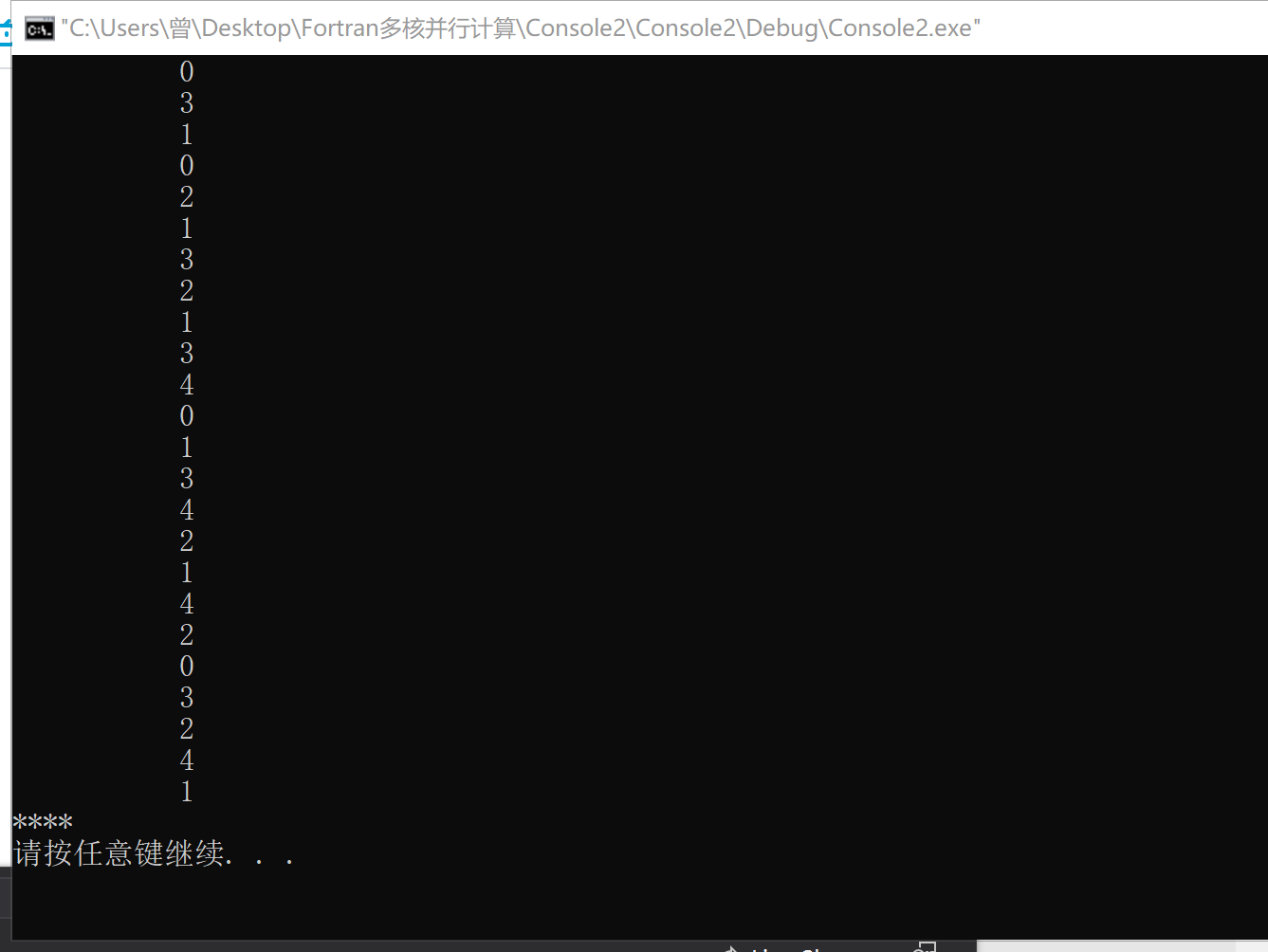
1. Program ex1 输出1次 ‘Hello’。因为n=4，if条件不满足，为串形代码；而当n<4时，线程数才为3，输出3次‘Hello’；
2. Program ex2 输出6次 ‘Hello’。该程序omp\_set\_max\_active\_levels(2)设置了最大嵌套并行层数为2，而该程序嵌套并行层数同样为2。故第一层num\_threads(3)设置后续所有并行域的线程数为3个，第二层num\_threads(2)设置此前每个并行域的后续所有并行域的线程数为2，故总共有3×2=6个线程。
3. **查找资料，结合例子test\_4\_05、test\_4\_09，对比讨论 taskloop和do的异同点，并说明各自的适用情形。假设test\_4\_09中的并行域有5线程，如果没有single，执行结果如何？（10分）**



1. !$omp do 结构可以很方便地将循环并行化，但其需要在执行并行之前确切知道循环次数，才能划分工作块；而当循环次数未知(或无穷)时，如do while、递归、链表、数据实时处理等，!$omp taskloop结构来显式定义任务，并交由某个线程执行。
2. 在test\_4\_05中8个线程，但是只有一个将起到作用。在每次循环会生成一个任务

task生成任务只能在single里面，否则8个线程都会生成任务。Taskloop 用于do循环的分配

1. 如果去掉single 则每个线程都将生成任务，一共输出25个数字。



1. **给出omp\_get\_wtime函数 与 cpu\_time子程序 的差异。（5分）**
2. omp\_get\_wtime函数

返回以秒为单位的绝对时间，调用两次函数就可计算出它们之间的代码的执行时间。

1. cpu\_time子程序

fortran标准函数，返回以秒为单位的近似的处理机时间(所有线程的任务量总和除以时钟频率)，该值仅反映任务量大小。对于串行代码，其值近似等于执行时间；对于并行代码，其值大于程序执行时间。

1. **某个任务，串行执行10秒，5线程并行执行2.5秒，求其加速比、效率。（5分）**
2. 加速比Sp = 10/2.5 = 4
3. 效率Ep = 4/5 = 0.8
4. **有一个整型数组定义为 integer a(4,3) 则 a(3,3) 是它的第几个元素？（5分）**

第11个元素。Fortran数组以列优先。故a(3,3)是第三列第三行。而该元素前面有两行，一列4个元素，该元素所在行前有2个元素，故a(3,3)前面有(j-1)\*m+(i-1)=(3-1)\*4+(3-1)=10个元素，则a(3,3)为第11个元素。

1. **写出一个求阶乘的函数，输入参数为整数n，。（10分）**

integer function fun(n)

implicit none

integer n,i

fun = 1

do i = n, 1, -1

fun = fun \* i

end do

end function

program test

integer a

interface

integer function fun(n)

integer n,i

end function

end interface

a = fun(4)

print\*,a

end program

1. **分别从文件a.txt及b.txt读取两个矩阵A(3,4)和B(4,2)，求矩阵相乘的结果C(3,2), 并将C存入文件c.txt 。代码中尽可能采用并行计算，并给出输入及输出结果的截图。（15分）**
2. 程序：

program test

use omp\_lib

implicit none

integer a(2,3),b(3,4),c(2,4)

integer i,j

**!$omp parallel num\_threads(2) private(i)**

**!$omp SECTIONS**

open(10,file = '1.txt')

do i = 1,2

read(10,\*) a(i,:)

end do

close(10)

**!$omp end SECTIONS**

**!$omp SECTIONS**

open(11,file = '2.txt')

do i = 1,3

read(11,\*) b(i,:)

end do

close(11)

**!$omp end SECTIONS**

**!$omp MASTER**

write(\*,\*) "读取完毕"

**!$omp END MASTER**

**!$omp do collapse(2)**

do i=1,2

do j=1,4

c(i,j) = sum(a(i,:)\*b(:,j) )

end do

end do

**!$omp end do**

**!$omp end parallel**

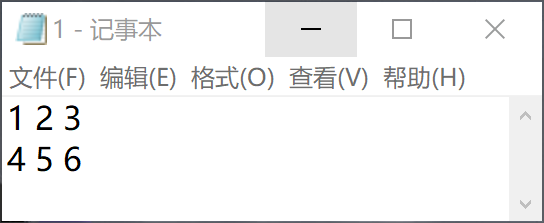
do i=1,2

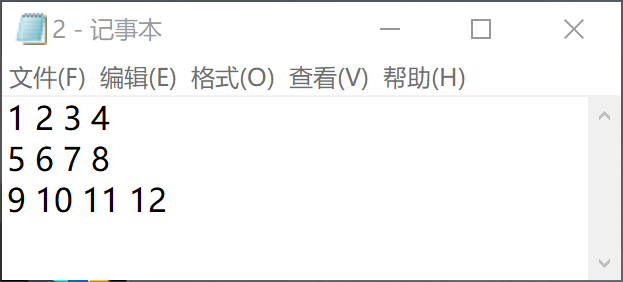
write(\*,'(4i5)') c(i,:)

end do

end program

1. 输入：





1. 输出：



1. **结合自身所学专业或者希望解决的实际问题，设计并实现一个并行计算的算法，并给出结果。（15分）**

program testQuickSort

! PURPOSE - to test the QuickSort

implicit none

integer :: n, m, i

real,allocatable, dimension(:) :: nums

real :: ran\_num

! Read in a number - N

write(\*,\*) 'Pls give the length (N) of the array (N should be the integer times of 5)'

read(\*,\*) n

allocate(nums(n))

! Generate the random array of the length N in the scope of (0 to N)

do i=1,n

call random\_number(ran\_num)

nums(i) = ran\_num\*n

end do

! Print the Original array before the sorting

write(\*,\*) '-------------------------------------------'

write(\*,\*) 'The original Array (left to right):'

m = 0

do while (m <n)

write(\*,'(5f11.4)') nums(m+1:m+5)

m = m + 5

end do

!Sort the array

call QuickSort(nums,n)

!Print the Array after the sorting

write(\*,\*) '-------------------------------------------'

write(\*,\*) 'The sorted Array (left to right):'

m = 0

do while (m <n)

write(\*,'(5f11.4)') nums(m+1:m+5)

m = m + 5

end do

pause

end program

!------------------------------------

recursive subroutine QuickSort(A, n)

implicit none

integer,intent(in) :: n

real, intent(in out), dimension(n) :: A

real:: V

integer :: i, n\_low, n\_high, n\_equal

real, dimension(n):: low, high, equal

n\_low = 0; n\_high=0; n\_equal=0

if (n>1) then

V = A(1)

do i=1, n

if (A(i)<V) then

n\_low = n\_low+1

low(n\_low) = A(i)

elseif (A(i) > V) then

n\_high = n\_high+1

high(n\_high) = A(i)

else

n\_equal = n\_equal+1

equal(n\_equal) = A(i)

end if

end do

if (n\_low>1) call QuickSort(low(1:n\_low),n\_low)

if (n\_high>1)call QuickSort(high(1:n\_high),n\_high)

if(n\_low>0) A(1:n\_low) = low(1:n\_low)

if(n\_equal>0) A(n\_low+1:n\_low+n\_equal) = equal(1:n\_equal)

if(n\_high>0) A(n\_low+n\_equal+1:n) = high(1:n\_high)

return

else

return

end if

end subroutine

