Introduction to MPI: Lecture 3

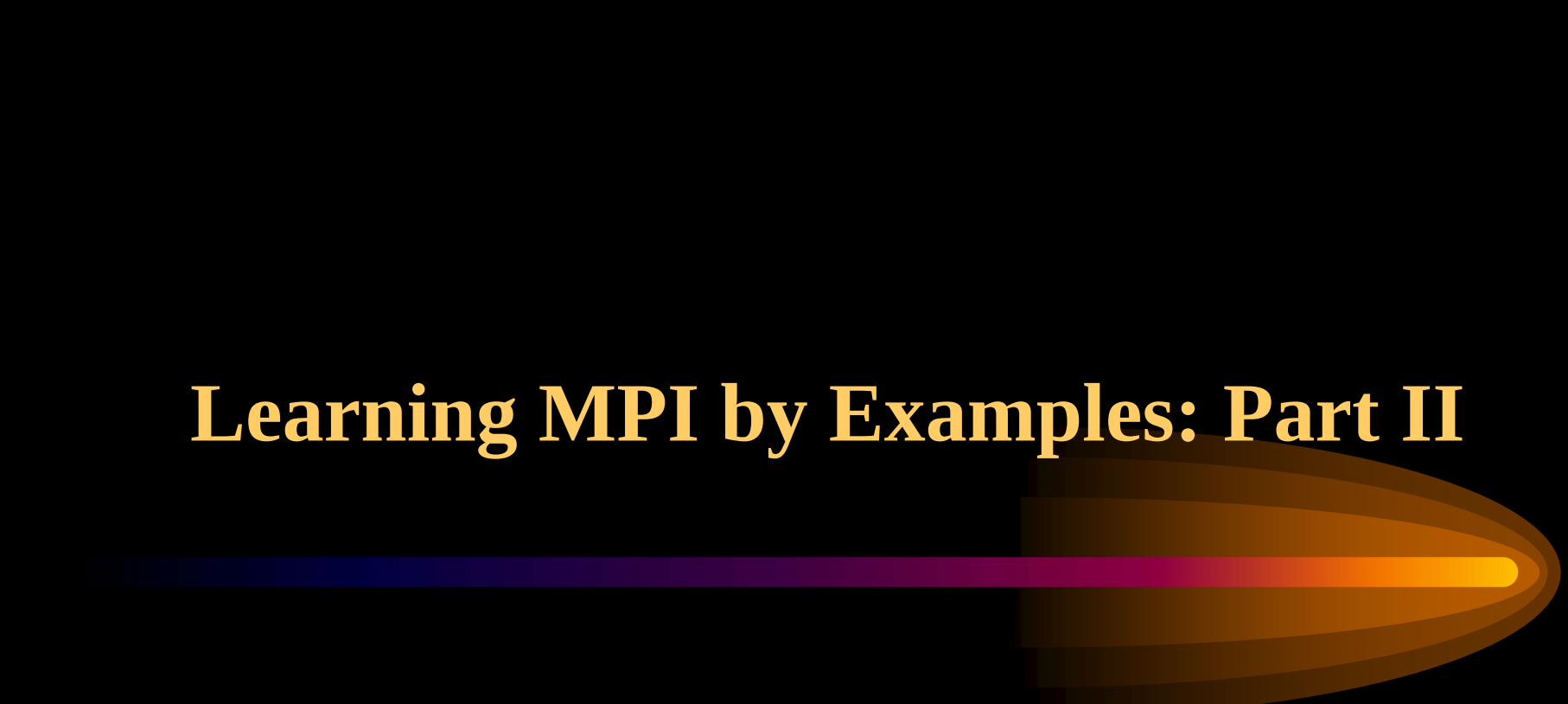
Jun Ni, Ph.D. M.E.

Associate Professor
Department of Radiology
Carver College of Medicine

Information Technology Services

The University of Iowa

Learning MPI by Examples: Part II



Parallel Programming with MPI
blocking sending/receiving

I/O on Parallel System and

Numerical Integration

Learning MPI by Examples: Part II

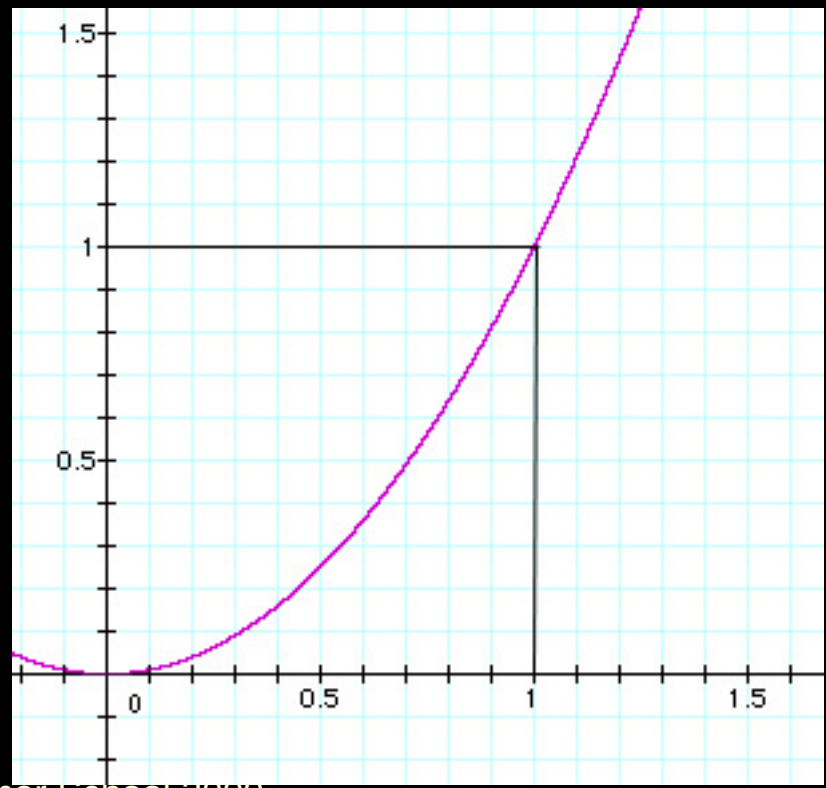
- Objective:
  - Learn how to use MPI blocking communication
  - Learn how to program I/O in parallel system
  - Use message passing to numerically solve a problem: numerical integration

Learning MPI by Examples: Part II

- Example 1: numerical integration using various numerical method
  - Mathematical problem:
    - definite integral from a to b
  - Numerical methods:
    - rectangle (one-point), trapezoid (two-point), Simpson(three-point) methods
  - Serial programming and parallel programming

Learning MPI by Examples: Part II

- Problem: integration of x^2 from 0 to 1
  - trapezoid method
  - exact solution:
    1/3=0.33333333



Learning MPI by Examples: Part II

- i^{th} trapezoid sub-integral:
  - $[f(x_{i-1})+f(x_i)]\ h/2$
- The accumulated total integral:

$[f(x_0)+f(x_1)]\ h/2 + [f(x_1)+f(x_2)]\ h/2 \ldots$

$\ldots [f(x_{i-1})+f(x_i)]\ h/2 \ldots [f(x_{n-1})+f(x_n)]\ h/2$

$= [f(x_0)+f(x_n)]\ h/2 + [f(x_1)+f(x_2)+ \ldots f(x_{n-1})]\ h$

Learning MPI by Examples: Part II

```
/* serial.c -- serial trapezoidal rule
 *
 * Calculate definite integral using trapezoidal rule.
 * The function f(x) is hardwired.
 *  Input: a, b, n.
 * Output: estimate of integral from a to b of f(x)
*   using n trapezoids.
 *
 */
```

Learning MPI by Examples: Part II

```
#include <stdio.h>
float f(float x);                   /* function prototype  */
main()
{
    float  integral;        /* Store result in integral      */
    float  a, b;            /* Left and right endpoints    */
    int    n;               /* Number of trapezoids        */
    float  h;               /* Trapezoid base width        */
    float  x;
    int    i;
```

Learning MPI by Examples: Part II

```
printf("Enter a, b, and n\n");
scanf("%f %f %d", &a, &b, &n);
h = (b-a)/n;
integral = (f(a) + f(b))/2.0;
x = a;
for (i = 1; i <= n-1; i++)
{
    x = x + h;
    integral = integral + f(x);
 }
```

Learning MPI by Examples: Part II

```
    integral = integral*h;
    printf("With n = %d trapezoids, our estimate\n", n);
    printf("of the integral from %f to %f = %f\n",   a, b, integral);
}

float f(float x)
{
    /* Calculate f(x).       calculation f(x), here the function is x*x */
    return x*x;
}
```

Learning MPI by Examples: Part II

```
% cc -o serial serial.c
% serial
Enter a, b, and n
0 1 200
With n = 200 trapezoids, our estimation
of the integral from 0.000000 to 1.000000 = 0.333337
```

Learning MPI by Examples: Part II

serial code in Fortran:

```
C serial.f -- calculate definite integral using trapezoidal rule.
C
C The function f(x) is hardwired.
C Input: a, b, n.
C Output: estimate of integral from a to b of f(x)
C     using n trapezoids.
C
C See Chapter 4, pp. 53 & ff. in PPMPI.
C
```

Learning MPI by Examples: Part II

```
      PROGRAM serial
      INCLUDE 'mpif.h'
      real  integral
      real  a
      real  b
      integer  n
      real  h
      real  x
      integer i
C
      real f
C
```

Learning MPI by Examples: Part II

```
      print *, 'Enter a, b, and n'
      read *,  a,  b,  n
C
      h = (b-a)/n
      integral = (f(a) + f(b))/2.0
      x = a
      do 100 i = 1 , n-1
         x = x + h
         integral = integral + f(x)
 100  continue
      integral = integral*h
C
```

Learning MPI by Examples: Part II

```
      print *,'With n =', n,' trapezoids, our estimate'
      print *,'of the integral from ', a, ' to ',b, ' = ' , integral
      end
C
C*************************************************************
      real function f(x)
      real x
C  Calculate f(x).
C
      f = x*x
      return
```

Learning MPI by Examples: Part II

```
Program Example
implicit none
integer n, p, i, j
real h, result, a, b, integral, pi

pi = acos(-1.0)     !!  = 3.14159...
a = 0.0             !! lower limit of integration
b = pi*1./2.        !! upper limit of integration
p = 4               !! number of processes (partitions)
n = 500             !! number of increment within each process
h = (b-a)/n/p       !! length of increment
```

Learning MPI by Examples: Part II

```
result = 0.0        !! stores answer to the integral
do i=1,p            !! sum of integrals over all processes
  result = result + integral(a,i,h,n)
enddo
print *,'The result =',result
stop
end

real function integral(a,i,h,n)
 implicit none
 integer n, i, j
 real h, h2, aij, a
 real fct, x
```

Learning MPI by Examples: Part II

```
fct(x) = cos(x)                        !! kernel of the integral

integral = 0.0                          !! initialize integral
h2 = h/2.
do j=1,n                                !! sum over all "j" integrals
  aij = a + ((i-1)*n +(j-1))*h          !! lower limit of "j" integral
  integral = integral + fct(aij+h2)*h
enddo

return
end
```

Learning MPI by Examples: Part II

- To compile and execute example.f

```
% f77 serial.f -lmpi
% a.out
```

- Result:

```
Enter a, b, and n
0 1 200
With n =   200 trapezoids, our estimate
 of the integral from   0.0000000E+00 to    1.000000     =
0.3333370
```

Learning MPI by Examples: Part II

- Parallel programming with MPI blocking Send/Receive
  - implement-dependent because using assignment of inputs
  - Using the following MPI functions
    - MPI_Init and MPI_Finalize
    - MPI_Comm_rank
    - MPI_Comm_size
    - MPI_Recv
    - MPI_Send

Learning MPI by Examples: Part II

- Parallel programming with MPI blocking Send/Receive
  - master process receives each partial result, based on subinterval integration from other process
  - master sum all of the sub-result together
  - other processes are idle during master's performance (due to blocking communication)

Learning MPI by Examples: Part II

- Numerical Algorithms
  - The global variables:
    - a: global left endpoint, input variable
    - b: global right end point, input variable
    - p: total number of process, input variable
    - n: total number of trapezoids for each sub-integral
    - h: trapezoid base length while p=1 (single process)
  - The local variables for each process
    - local_a: local left endpoint
    - local_b: local right end point
    - local_h: local trapezoid base length

Learning MPI by Examples: Part II

- The expressions of local variables for process i (rank of process)

local_a=a+i(b-a)/p

local_b= a + (i+1) (b-a)/p
= a+i(b-a)/p + (b-a)/p
= local_a + local_h *n

local_h = (local_b - local_a)/n
= [(b-a)/p]/n = h/p

where h=(b-a)/n

Learning MPI by Examples: Part II

- assignment of sub-integrals to processes

[a,	a + (b-a)/p]
[a + (b-a)/p,	a + 2 (b-a)/p]
[a + 2(b-a)/p,	a + 3 (b-a)/p]
…	
[a+i(b-a)/p,	a + (i+1) (b-a)/p]
…	
[a + (p-1) (b-a)/p,	a + p (b-a)/p=b]

(i=0, 1, 2, … p-1)

Learning MPI by Examples: Part II

Example of parallel programming in C:

```
/* trap.c -- Parallel Trapezoidal Rule, first version
*
* Input: None.
* Output:  Estimate of the integral from a to b of f(x)
*    using the trapezoidal rule and n trapezoids.
*
* Algorithm:
*    1.  Each process calculates "its" interval of
*        integration.
*    2.  Each process estimates the integral of f(x)
*        over its interval using the trapezoidal rule.
```

Learning MPI by Examples: Part II

```
*    3a. Each process != 0 sends its integral to process 0.
*    3b. Process 0 sums the calculations received from
*        the individual processes and prints the result.
*
* Notes:
*    1.  f(x), a, b, and n are all hardwired.
*    2.  The number of processes (p) should evenly divide
*        the number of trapezoids (n = 1024)
*
*/
#include <stdio.h>
```

Learning MPI by Examples: Part II

```
 /* We'll be using MPI routines, definitions, etc. */
#include "mpi.h"

main(int argc, char** argv)
{
    int         my_rank;   /* My process rank           */
    int         p;         /* The number of processes   */
    float       a = 0.0;   /* Left endpoint             */
    float       b = 1.0;   /* Right endpoint            */
    int         n = 1024;  /* Number of trapezoidsi
                              in each subintegrals      */
    float       h;         /* Trapezoid base length     */
```

Learning MPI by Examples: Part II

```
/* local_a and local_b are the bounds
for each integration performed in individual process */

float       local_a;   /* Left endpoint my process  */
float       local_b;   /* Right endpoint my process */
float       local_h;   /* trapezoid base length for
                          each sub-integral */
float       integral;  /* Integral over my interval */
float       total;     /* Total integral            */
int         source;    /* Process sending integral  */
int         dest = 0;  /* All messages go to 0      */
int         tag = 0;
MPI_Status  status;
```

Learning MPI by Examples: Part II

```
/* Trap function prototype. Trap function is used to calculate
  local integral   */

   float Trap(float local_a, float local_b, int local_n);

   /* Let the system do what it needs to start up MPI */
   MPI_Init(&argc, &argv);

   /* Get my process rank */
   MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
```

Learning MPI by Examples: Part II

```
/* Find out how many processes are being used */
   MPI_Comm_size(MPI_COMM_WORLD, &p);

   h = (b-a)/n;    /* h is the same for all processes */
   local_h = h/p;  /* So is the number of trapezoids */

   local_a = a + my_rank*local_h*n;
   local_b = local_a + local_h*n;
   integral = Trap(local_a, local_b, n);
```

Learning MPI by Examples: Part II

```
if (my_rank == 0)
   {
   /* Add up the integrals calculated by each process */
     total = integral;   /* this is the intergal calculated by process 0 */
     for (source = 1; source < p; source++)
     {
        MPI_Recv(&integral, 1, MPI_FLOAT, source, tag,
              MPI_COMM_WORLD, &status);
        total = total + integral;
     }
   }
```

Learning MPI by Examples: Part II

```
else
   {
    printf("The intergal calculated from  process %d is %f\n",
              my_rank,integral);
    MPI_Send(&integral, 1, MPI_FLOAT, dest, tag,
                  MPI_COMM_WORLD);
   }

   /* Print the result */
   if (my_rank == 0)
   {
    printf("With n = %d trapezoids, our estimate\n", n);
    printf("of the integral from %f to %f = %f\n",a,b,total);
   }
```

Learning MPI by Examples: Part II

```
  /* Shut down MPI */
    MPI_Finalize();
}


float Trap (
          float  local_a   /* in */,
          float  local_b   /* in */,
          int    local_n   /* in */)
```

Learning MPI by Examples: Part II

```
{

  float integral;   /* Store result in integral  */
  float x;
  int i;
  float local_h;

  float f(float x); /* function we're integrating */
  local_h=(local_b-local_a)/local_n;
  integral = (f(local_a) + f(local_b))/2.0;
  x = local_a;
```

Learning MPI by Examples: Part II

```
for (i = 1; i <= local_n-1; i++)
   {
      x = x + local_h;
      integral = integral + f(x);
   }
   integral = integral*local_h;
   return integral;
}
```

Learning MPI by Examples: Part II

```
 float f(float x)
{
    float return_val;
    /* Calculate f(x). */
    /* Store calculation in return_val. */
    return_val = x*x;
    return return_val;
} /* f */
```

Learning MPI by Examples: Part II

- To compile a C code with MPI library

```
cc -o trap trap_.c -lmpi
```

- To run job interactively using SGI's MPI implementation:

```
/bin/time mpirun -np 8 trap
```

Learning MPI by Examples: Part II

- Result (first run):

```
% /bin/time mpirun -np 8 a.out
The integral calculated from  process 1 is 0.004557
The integral calculated from  process 2 is 0.012370
The integral calculated from  process 3 is 0.024089
The integral calculated from  process 5 is 0.059245
With n = 1024 trapezoids, our estimate
of the integral from 0.000000 to 1.000000 = 0.333333
The integral calculated from  process 4 is 0.039714
The integral calculated from  process 6 is 0.082682
The integral calculated from  process 7 is 0.110026
```

Learning MPI by Examples: Part II

- Result (second run):

```
mpirun -np 8 a.out
The integral calculated from  process 1 is 0.004557
The integral calculated from  process 7 is 0.110026
The integral calculated from  process 2 is 0.012370
The integral calculated from  process 3 is 0.024089
The integral calculated from  process 4 is 0.039714
The integral calculated from  process 5 is 0.059245
The integral calculated from  process 6 is 0.082682
With n = 1024 trapezoids, our estimate
of the integral from 0.000000 to 1.000000 = 0.333333
```

Learning MPI by Examples: Part II

- Result (thirf run):

```
mpirun -np 8 a.out
The integral calculated from  process 3 is 0.024089
The integral calculated from  process 2 is 0.012370
The integral calculated from  process 4 is 0.039714
The integral calculated from  process 5 is 0.059245
The integral calculated from  process 1 is 0.004557
The integral calculated from  process 6 is 0.082682
The integral calculated from  process 7 is 0.110026
With n = 1024 trapezoids, our estimate
of the integral from 0.000000 to 1.000000 = 0.333333
```

Learning MPI by Examples: Part II

- Result:

```
real 1.726
user 0.006
sys  0.050
```

Learning MPI by Examples: Part II

- Example of parallel programming in Fortran

```
c  trap.f -- Parallel Trapezoidal Rule, first version
c
c  Input: None.
c  Output:  Estimate of the integral from a to b of f(x)
c     using the trapezoidal rule and n trapezoids.
c
c  Algorithm:
c     1.  Each process calculates "its" interval of
c         integration.
```

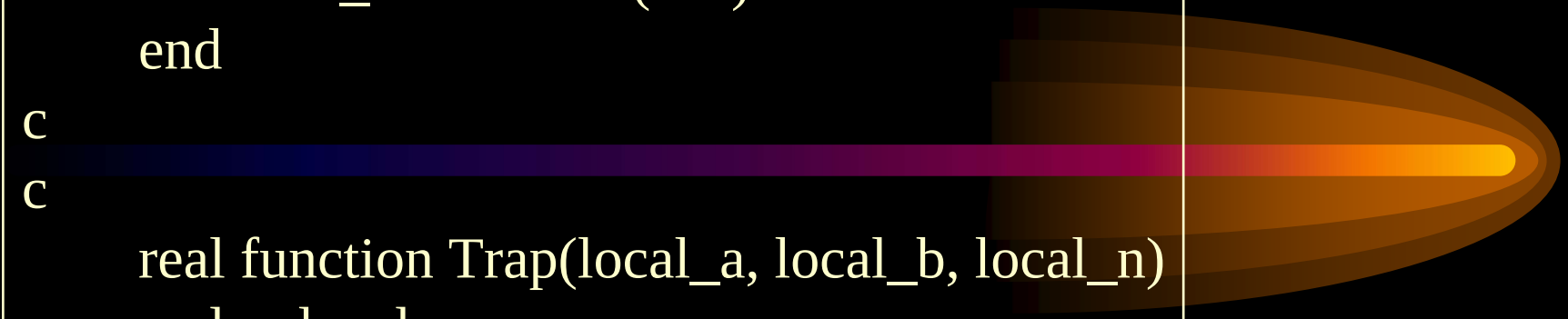
```
c     2.  Each process estimates the integral of f(x)
c         over its interval using the trapezoidal rule.
c     3a. Each process != 0 sends its integral to 0.
c     3b. Process 0 sums the calculations received from
c         the individual processes and prints the result.
c
c  Notes:
c     1.  f(x), a, b, and n are all hardwired.
c     2.  Assumes number of processes (p) evenly divides
c         number of trapezoids (n = 1024)
c
c
      program trapezoidal
c
      include 'mpif.h'
c
```

```
integer   my_rank
      integer   p
      real      a
      real      b
      integer   n
      real      h
      real      local_a
      real      local_b
      real      local_h
      integer   local_n
      real      integral
      real      total
      integer   source
      integer   dest
      integer   tag
      integer   status(MPI_STATUS_SIZE)
      integer   ierr
```

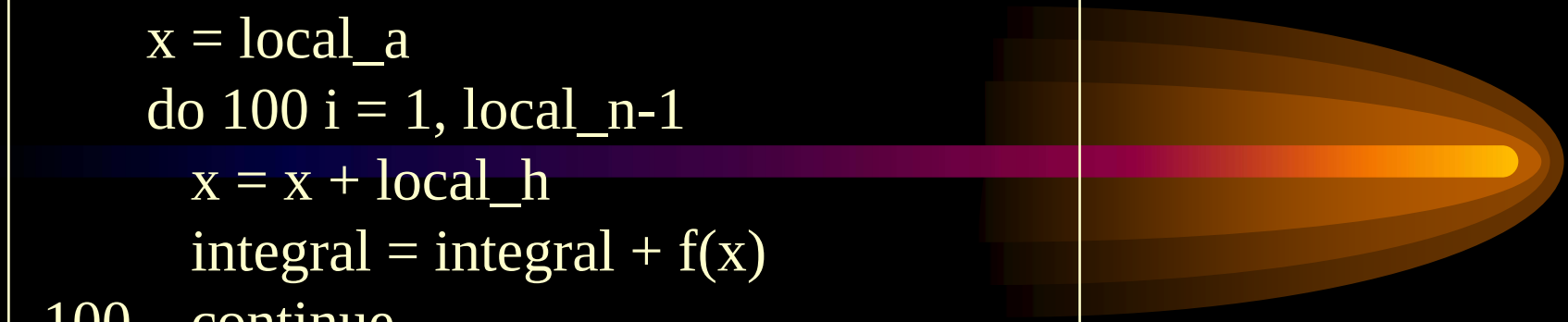
```
c
      real      Trap
c
      data a, b, n, dest, tag /0.0, 1.0, 1024, 0, 0/

      call MPI_INIT(ierr)
      call MPI_COMM_RANK(MPI_COMM_WORLD, my_rank, ierr)
      call MPI_COMM_SIZE(MPI_COMM_WORLD, p, ierr)

      h = (b-a)/n
      local_h=h/p

      local_a = a + my_rank*local_h*n
      local_b = local_a + local_h*n
      integral = Trap(local_a, local_b, n)
```

```
if (my_rank .EQ. 0) then
        total = integral
        do 100 source = 1, p-1
           call MPI_RECV(integral, 1, MPI_REAL, source, tag,
   +            MPI_COMM_WORLD, status, ierr)
           total = total + integral
100        continue
      else
        call MPI_SEND(integral, 1, MPI_REAL, dest,
   +         tag, MPI_COMM_WORLD, ierr)
      endif
      if (my_rank .EQ. 0) then
        write(6,200) n
200        format(' ','With n = ',I4,' trapezoids, our estimate')
        write(6,300) a, b, total
300        format(' ','of the integral from ',f6.2,' to ',f6.2,
   +             ' = ',f11.5)
      endif
```

```
      call MPI_FINALIZE(ierr)
      end
c
c
      real function Trap(local_a, local_b, local_n)
      real    local_a
      real    local_b
      integer  local_n
      real    local_h
c
      real    integral
      real    x
      real    i
c
      real    f
c
```



```
      local_h=(local_b-local_a)/local_n
      integral = (f(local_a) + f(local_b))/2.0
      x = local_a
      do 100 i = 1, local_n-1
        x = x + local_h
        integral = integral + f(x)
100   continue
      Trap = integral*local_h
      return
      end
c
      real function f(x)
      real x
      real return_val
      return_val = x*x
      f = return_val
      return
      end
```



Learning MPI by Examples: Part II

- To compile a f77 code with MPI library

```
f77 -o trap trap.f -lmpi
```

- To run job interactively using SGI's MPI implementation:

```
/bin/time mpirun -np 8 trap
```

- First run result:

```
With n = 1024 trapezoids, our estimate
of the integral from    0.00 to    1.00 =      0.33333
```

Learning MPI by Examples: Part II

- Notes:
  - Different process performs different part of computation based on branching statements
  - Distinguish between the variables whose contents were significant on all the processes, and the variables whose contents were only significant on individual processes.
  - global and local variables, respectively

Learning MPI by Examples: Part II

- Notes:
  - Clear documenting the global and local variables is very crucial to parallel programming
  - Partial results are over all identical
  - Problem: this code lacks of input/output generality. That means a, b, and n are hardwired.

Learning MPI by Examples: Part II

- I/O in parallel system
  - options:
    - let every process do I/O work
    - let process 0 (master process) do I/O work. In this case, we need for process 0 to send user's inputs to other processes, using MPI_Send() and MPI_Recv()

Learning MPI by Examples: Part II

- I/O in parallel system
  - Let process 0 send a, b, and n to each process.
  - use different tag for each data transferring
  - input/output is performed using separate function

Learning MPI by Examples: Part II

```
/* get_data.c -- Parallel Trapezoidal Rule,
 *         uses basic Get_data function for input.
*
* Input:
*    a, b: limits of integration.
*    n: number of trapezoids.
* Output:  Estimate of the integral from a to b of f(x)
*    using the trapezoidal rule and n trapezoids.
*
* Notes:
*    1.  f(x) is hardwired.
*    2.  Assumes number of processes (p) evenly divides
*        number of trapezoids (n)    */
```

Learning MPI by Examples: Part II

```
#include <stdio.h>

/* We'll be using MPI routines, definitions, etc. */
#include "mpi.h"

main(int argc, char** argv)
{
    int         my_rank;   /* My process rank           */
    int         p;         /* The number of processes   */
    float       a;         /* Left endpoint             */
    float       b;         /* Right endpoint            */
    int         n;         /* Number of trapezoids      */
    float       h;         /* Trapezoid base length     */
```

Learning MPI by Examples: Part II

```
float       local_a;   /* Left endpoint my process  */
   float       local_b;   /* Right endpoint my process */
   float       local_h;   /* trapezoid base length for  */
                          /*  each sub-integral         */
   float       integral;  /* Integral over my interval */
   float       total;     /* Total integral            */
   int         source;    /* Process sending integral  */
   int         dest = 0;  /* All messages go to 0      */
   int         tag = 0;
   MPI_Status  status;
```

Learning MPI by Examples: Part II

```
/* function prototypes */
   void Get_data(float* a_ptr, float* b_ptr,
                 int* n_ptr, int my_rank, int p);
   float Trap(float local_a, float local_b, int local_n);
                    /* Calculate local integral  */

   /* Let the system do what it needs to start up MPI */
   MPI_Init(&argc, &argv);

   /* Get my process rank */
   MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
```

Learning MPI by Examples: Part II

```
/* Find out how many processes are being used */
   MPI_Comm_size(MPI_COMM_WORLD, &p);


   Get_data(&a, &b, &n, my_rank, p);

   h = (b-a)/n;    /* h is the same for all processes */
   local_h = h/p;  /* So is the number of trapezoids */

   local_a = a + my_rank*local_h*n;
   local_b = local_a + local_h*n;

   integral = Trap(local_a, local_b, n);
```

Learning MPI by Examples: Part II

```
/* Add up the integrals calculated by each process */
  if (my_rank == 0)
  {
      total = integral;
      for (source = 1; source < p; source++)
      {
        MPI_Recv(&integral, 1, MPI_FLOAT, source, tag,
              MPI_COMM_WORLD, &status);
        total = total + integral;
      }
  }
  else
```

Learning MPI by Examples: Part II

```
{
    MPI_Send(&integral, 1, MPI_FLOAT,
                    dest, tag, MPI_COMM_WORLD);
   }
 /* Print the result */
   if (my_rank == 0)
   {
    printf("With n = %d trapezoids, our estimate\n", n);
    printf("of the integral from %f to %f = %f\n",
          a, b, total);
   }
   MPI_Finalize();
}
```

Learning MPI by Examples: Part II

```
/*********************************************************************/
/* Function Get_data
 * Reads in the user input a, b, and n.
 * Input parameters:
 *     1.  int my_rank:  rank of current process.
 *     2.  int p:  number of processes.
 * Output parameters:
 *     1.  float* a_ptr:  pointer to left endpoint a.
 *     2.  float* b_ptr:  pointer to right endpoint b.
 *     3.  int* n_ptr:  pointer to number of trapezoids.
```

Learning MPI by Examples: Part II

```
 * Algorithm:
 *     1.  Process 0 prompts user for input and
 *         reads in the values.
 *     2.  Process 0 sends input values to other
 *         processes.
 */
void Get_data(
         float*  a_ptr    /* out */,
         float*  b_ptr    /* out */,
         int*    n_ptr    /* out */,
         int     my_rank  /* in  */,
         int     p        /* in  */)
{
```

Learning MPI by Examples: Part II

```
int source = 0;    /* All local variables used by */
int dest;          /* MPI_Send and MPI_Recv       */
int tag;
MPI_Status status;

if (my_rank == 0)
{
    printf("Enter a, b, and n\n");
    scanf("%f %f %d", a_ptr, b_ptr, n_ptr);
```

Learning MPI by Examples: Part II

```
for (dest = 1; dest < p; dest++)
      {
       tag = 0;
       MPI_Send(a_ptr, 1, MPI_FLOAT, dest, tag,
                                          MPI_COMM_WORLD);
       tag = 1;
       MPI_Send(b_ptr, 1, MPI_FLOAT, dest, tag,
                                          MPI_COMM_WORLD);
       tag = 2;
       MPI_Send(n_ptr, 1, MPI_INT, dest, tag,
                                          MPI_COMM_WORLD);
      }
  }
else
```

Learning MPI by Examples: Part II

```
{
    tag = 0;
    MPI_Recv(a_ptr, 1, MPI_FLOAT, source, tag,
MPI_COMM_WORLD, &status);
    tag = 1;
    MPI_Recv(b_ptr, 1, MPI_FLOAT, source, tag,
MPI_COMM_WORLD, &status);
    tag = 2;
    MPI_Recv(n_ptr, 1, MPI_INT, source, tag,
MPI_COMM_WORLD, &status);
  }
} /* Get_data */
```

Learning MPI by Examples: Part II

```
/*********************************************************************
*****************/
float Trap( float  local_a   /* in */,
            float  local_b   /* in */,
            int    local_n   /* in */)
{

   float integral;   /* Store result in integral  */
   float x;
   int i;
   float local_h;
```

Learning MPI by Examples: Part II

```
float f(float x); /* function we're integrating */

   local_h=(local_b-local_a)/local_n;
   integral = (f(local_a) + f(local_b))/2.0;
   x = local_a;
   for (i = 1; i <= local_n-1; i++) {
       x = x + local_h;
       integral = integral + f(x);
   }
   integral = integral*local_h;
   return integral;
} /*  Trap  */
```

Learning MPI by Examples: Part II

```
/*********************************************************
*****************/
float f(float x)
{
    float return_val;
    /* Calculate f(x). */
    /* Store calculation in return_val. */
    return_val = x*x;
    return return_val;
} /* f */
```

Learning MPI by Examples: Part II

```
% cc get_data.c -lmpi
% mpirun -np 8 a.out
Enter a, b, and n
0 1 1024
With n = 1024 trapezoids, our estimate
of the integral from 0.000000 to 1.000000 = 0.333333
```

Learning MPI by Examples: Part II

```
% cc get_data.c -lmpi
% mpirun -np 8 a.out
Enter a, b, and n
0 1 1024
With n = 1024 trapezoids, our estimate
of the integral from 0.000000 to 1.000000 = 0.333333
```

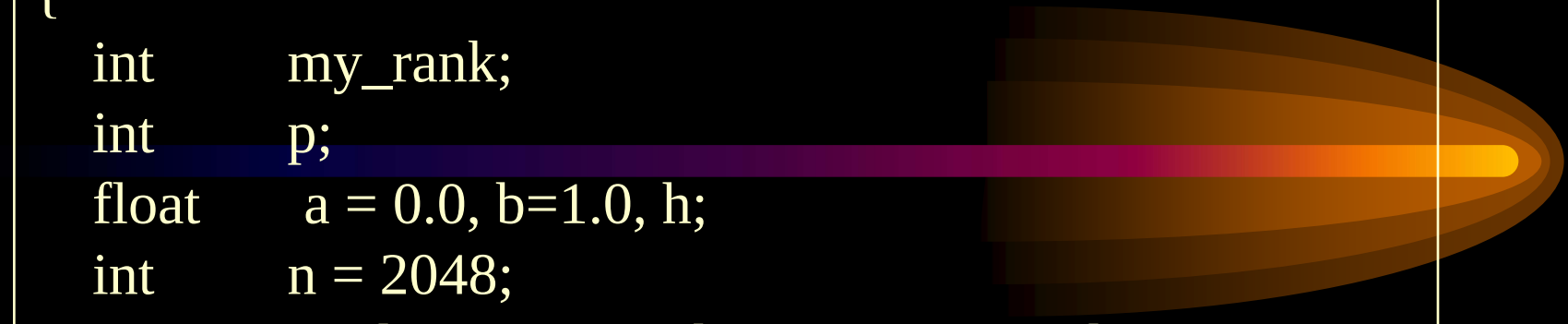
Learning MPI by Examples: Part II

- Parallel programming for numerical integration with various numerical methods

Learning MPI by Examples: Part II

```
silicon.weeg.uiowa.edu% more seosl_intNCb.c
/* seosl_intNC -- Parallel version of numerical integration
   with Newton-Cotes methods, which includes
     rectangle rule (one-point rule),
     tprapezoidal rule (two-point rule),
     Simpson rule(three-point rule)
*/

#include <stdio.h>
#include "mpi.h"
#include <math.h>
```

```
main(int argc, char** argv)
{
    int         my_rank;
    int         p;
    float       a = 0.0, b=1.0, h;
    int         n = 2048;
    int         mode=3;   /* mode=1,2,3  rectangle,
trapezoidal, and Simpson */

    float       local_a, local_b, local_h;

    float       local_integral, integral;
    int         source;
    int         dest = 0;
    int         tag = 0;
    MPI_Status  status;
```



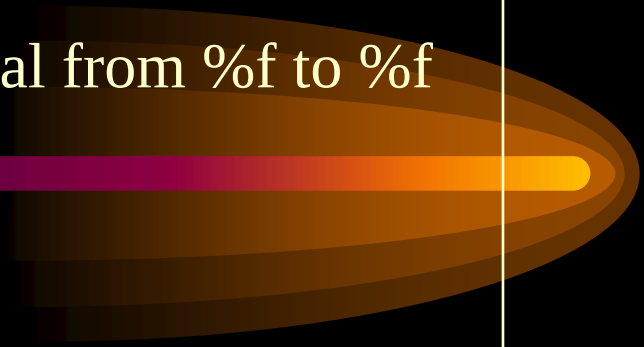
```
/* function prototypes */
    void Get_data(float* a_ptr, float* b_ptr,
                  int* n_ptr, int my_rank, int p, int *mode_ptr);
    float rect(float local_a, float local_b, int local_n);
    float trap(float local_a, float local_b, int local_n);
    float simp(float local_a, float local_b, int local_n);
/* MPI starts   */
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD,
&my_rank);
    MPI_Comm_size(MPI_COMM_WORLD, &p);
    Get_data(&a, &b, &n, my_rank, p, &mode);
    h = (b-a)/n;
    local_h=h/p;
    local_a = a + my_rank*local_h*n;
    local_b = local_a + local_h*n;
```

```
switch(mode)
 {
 case(1):
   local_integral = rect(local_a, local_b, n);
   break;
 case(2):
   local_integral = trap(local_a, local_b, n);
   break;
 case(3):
   local_integral = simp(local_a, local_b, n);
 }
```

```
if(my_rank==0)
{
 if (mode==1)
   printf("Rectangle rule (0-point rule) is selected\n");
 else if (mode==2)
   printf("Trapezodial rule (2-point rule) is selected\n");
 else    /* defaulted */
   printf("Simpson rule (3-point rule) is selected\n");
}

if (my_rank == 0)
{
  integral = local_integral;
  for (source = 1; source < p; source++)
  {
```

```
MPI_Recv(&local_integral,1,MPI_FLOAT,source,tag,MP
I_COMM_WORLD, &status)
;
        integral += local_integral;
      }
    }
    else
    {
      printf("The intergal calculated from  process %d is
%f\n",my_rank,local_in
tegral);
      MPI_Send(&local_integral, 1, MPI_FLOAT, dest, tag,
MPI_COMM_WORLD);
    }
```

```
  if (my_rank == 0)
    {
      printf("With n = %d, the total integral from %f to %f
= %f\n",n, a,b,integ
ral);
    }

/* MPI finished */
    MPI_Finalize();
}

/*********************************************************
**********************/
/* Function Get_data
 * Reads in the user input a, b, and n.
 * Input parameters:
 *     1.  int my_rank:  rank of current process.
```



```
 *     2.  int p:  number of processes.
 * Output parameters:
 *     1.  float* a_ptr:  pointer to left endpoint a.
 *     2.  float* b_ptr:  pointer to right endpoint b.
 *     3.  int* n_ptr:    pointer to number of trapezoids.
       3.  int* mode_ptr: pointer to mode of rule of Newton-
Cotes methods
 * Algorithm:
 *     1.  Process 0 prompts user for input and
 *         reads in the values.
 *     2.  Process 0 sends input values to other
 *         processes.
 */
void Get_data(
         float*  a_ptr    /* out */,
         float*  b_ptr    /* out */,
         int*    n_ptr    /* out */,
```

```
int     my_rank  /* in  */,
        int     p        /* in  */,
        int*    mode_ptr /* out */)
{
    int source = 0;    /* All local variables used by */
    int dest;          /* MPI_Send and MPI_Recv       */
    int tag;
    MPI_Status status;

    if (my_rank == 0)
    {
       do
       {
       printf("Enter a, b, n(1024), and mode(1--rect, 2-- trap,
3-- simp):\n");
       scanf("%f %f %d %d", a_ptr, b_ptr, n_ptr, mode_ptr);
       } while (*mode_ptr<1 || *mode_ptr>3);
```

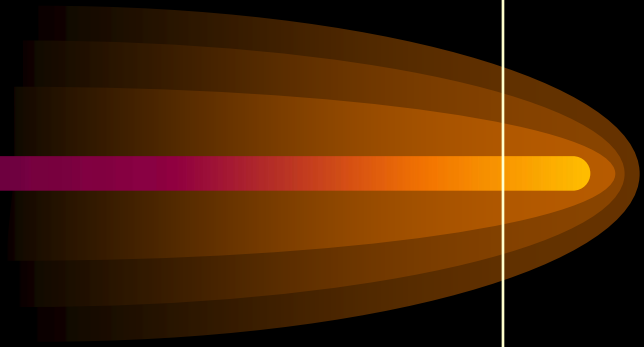
```
 for (dest = 1; dest < p; dest++)
      {
        tag = 0;
        MPI_Send(a_ptr, 1, MPI_FLOAT, dest, tag,
MPI_COMM_WORLD);
        tag = 1;
        MPI_Send(b_ptr, 1, MPI_FLOAT, dest, tag,
MPI_COMM_WORLD);
        tag = 2;
        MPI_Send(n_ptr, 1, MPI_INT, dest, tag,
MPI_COMM_WORLD);
        tag = 3;
        MPI_Send(mode_ptr, 1, MPI_INT, dest, tag,
MPI_COMM_WORLD);
      }
   }
   else
```

```
 {
      tag = 0;
      MPI_Recv(a_ptr, 1, MPI_FLOAT, source, tag,
MPI_COMM_WORLD, &status);
      tag = 1;
      MPI_Recv(b_ptr, 1, MPI_FLOAT, source, tag,
MPI_COMM_WORLD, &status);
      tag = 2;
      MPI_Recv(n_ptr, 1, MPI_INT, source, tag,
MPI_COMM_WORLD, &status);
      tag = 3;
      MPI_Recv(mode_ptr, 1, MPI_INT, source, tag,
MPI_COMM_WORLD, &status);
    }
} /* Get_data */
```

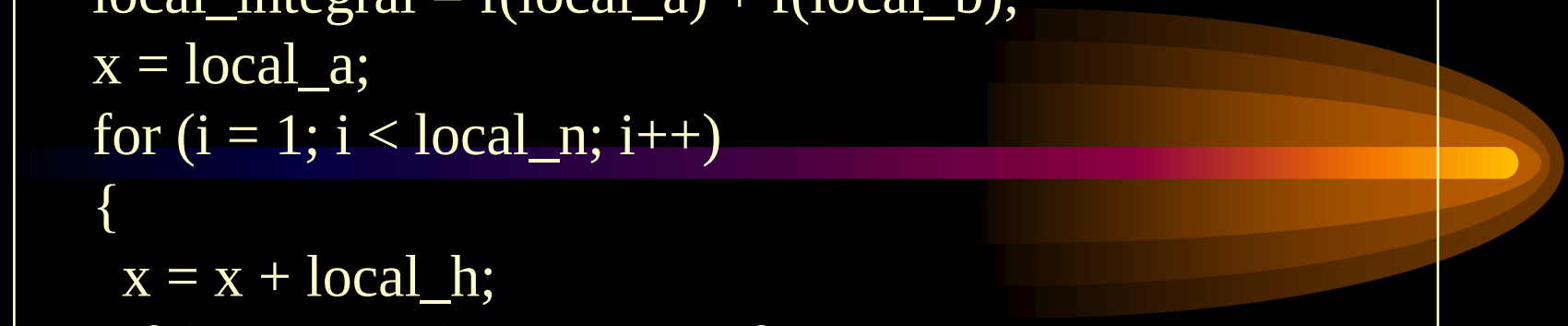
```
float rect( float local_a, float local_b, int local_n)
{
    float local_integral;
    float x;
    int i;
    float local_h;

    float f(float x);
    local_h=(local_b-local_a)/local_n;
    local_integral = f(local_a);
    x = local_a;
    for (i = 1; i <= local_n-1; i++)
    {
        x = x + local_h;
        local_integral += f(x);
    }
```

```
local_integral *=local_h;
    return local_integral;
}

float trap( float local_a, float local_b, int local_n)
{
    float local_integral;
    float x;
    int i;
    float local_h;

    float f(float x);

    local_h=(local_b-local_a)/local_n;
    local_integral = f(local_a) + f(local_b);
    x = local_a;
```

```
 for (i = 1; i <= local_n-1; i++)
    {
        x = x + local_h;
        local_integral += 2.0*f(x);
    }
    local_integral *=local_h/2.0;
    return local_integral;
}

float simp( float local_a, float local_b, int local_n )
{
    float local_integral;
    float x;
    int i;
    float local_h;

    float f(float x);
```



```
local_h=(local_b-local_a)/local_n;
    local_integral = f(local_a) + f(local_b);
    x = local_a;
    for (i = 1; i < local_n; i++)
    {
      x = x + local_h;
      if (i % 2 == 0)              /* if i is even */
         local_integral = local_integral + 2 * f(x);
      else                         /* if i is odd  */
         local_integral = local_integral + 4 * f(x);
      }
    local_integral *=local_h/3.0;
    return local_integral;
}

float f(float x)
{  return x*x; }
```



Learning MPI by Examples: Part II

```
mpirun -np 8 a.out
Enter a, b, n(1024), and mode(1--rect, 2-- trap, 3-- simp):
0 1 1024 1
Rectangle rule (0-point rule) is selected
The intergal calculated from  process 6 is 0.082670
The intergal calculated from  process 1 is 0.004554
The intergal calculated from  process 3 is 0.024082
The intergal calculated from  process 4 is 0.039705
The intergal calculated from  process 5 is 0.059234
The intergal calculated from  process 2 is 0.012365
The intergal calculated from  process 7 is 0.110012
With n = 1024, the total integral from 0.000000 to 1.000000
   = 0.333272
```

Learning MPI by Examples: Part II

```
% mpirun -np 8 a.out
Enter a, b, n(1024), and mode(1--rect, 2-- trap, 3-- simp):
0 1 1024 2
Trapezodial rule (2-point rule) is selected
The intergal calculated from  process 1 is 0.004557
The intergal calculated from  process 2 is 0.012370
The intergal calculated from  process 3 is 0.024089
The intergal calculated from  process 4 is 0.039714
The intergal calculated from  process 5 is 0.059245
The intergal calculated from  process 6 is 0.082682
The intergal calculated from  process 7 is 0.110026
With n = 1024, the total integral from 0.000000 to 1.000000 =
 0.333333
```

Learning MPI by Examples: Part II

```
%mpirun -np 8 a.out
Enter a, b, n(1024), and mode(1--rect, 2-- trap, 3-- simp):
0 1 1024 3
Simpson rule (3-point rule) is selected
The intergal calculated from  process 3 is 0.024089
The intergal calculated from  process 6 is 0.082682
The intergal calculated from  process 1 is 0.004557
The intergal calculated from  process 4 is 0.039714
The intergal calculated from  process 5 is 0.059245
The intergal calculated from  process 7 is 0.110026
The intergal calculated from  process 2 is 0.012370
With n = 1024, the total integral from 0.000000 to 1.000000
= 0.333333
```

Learning MPI by Examples: Part II

- Exercise:
  - The last example is a parallel code for numerical integration using the Newton-Cotes methods. Write a parallel code for numerical integration using the Gaussian-rule. Reference can be found at (http://www.engineering.uiowa.edu/~ncalc/dni/dni_03.html)