Interfaz de Paso de Mensajes MPI

Christian Chinchilla Brizuela

Agenda

- Definición
- Objetivo principal MPI
- Historia
- Ventajas
- Desventajas
- Estructura MPI
  - Programa MPI
  - Llamadas de MPI
  - Funciones Principales MPI
  - Mensajes en MPI
  - Envoltura de un mensaje MPI
  - Tipos de datos MPI
  - Operaciones MPI

- Ejemplo

Definición

- MPI es una **interfaz** de paso de mensajes que representa un esfuerzo prometedor de mejorar la disponibilidad de un software **altamente eficiente y portable** para satisfacer las necesidades actuales en la **computación de alto rendimiento** a través de la definición de un estándar de paso de mensajes universal.

William D. Gropp et al.

Definición

- Interfaz de Paso de Mensajes MPI (Message Passing Interface)
- MPI es el estándar para la comunicación entre los nodos que ejecutan un programa en un sistema de memoria distribuida.
- Define un formato estándar de la sintaxis y la semántica para el paso de mensajes usado en programas con múltiples procesadores.

Objetivo principal

- El objetivo principal de MPI es logar la portabilidad a través de diferentes máquinas, tratando de obtener un lenguaje de programación que permita ejecutar de manera transparente, aplicaciones sobre sistemas heterogéneos.

Historia

- Diseño MPI = características más atractivas de los sistemas existentes para el paso de mensajes.
- Fuerte influencia de los trabajos realizados por IBM, INTEL, NX/, Express, nCUBE's Vernex, p4 y PARMACS.
- Otras contribuciones importantes provienen de Zipcode, Chimp, PVM, Chameleon y PICL.

Historia

- Estandarizar MPI involucró a cerca de 60 personas de 40 organizaciones diferentes principalmente de U.S.A. y Europa.

- La mayoría de los vendedores de computadoras concurrentes estaban involucrados con MPI, así como con investigadores de diferentes universidades, laboratorios del gobierno e industrias.

Historia

- El proceso de estandarización comenzó en el taller de estándares para el paso de mensajes en un ambiente con memoria distribuida, patrocinado por el Centro de Investigación en Computación Paralela en Williamsburg, Virginia,Estados Unidos (Abril 29-30 de 1992).

Historia

- Se llegó a una propuesta preliminar conocida como MPI1, enfocada principalmente en comunicaciones punto a punto sin incluir rutinas para comunicación colectiva y no presentaba tareas seguras.
- El estándar final por el MPI fue presentado en la conferencia de Supercómputo en Noviembre de 1993.

Ventajas

- La escalabilidad. Las computadoras con sistemas de memoria distribuida son fáciles de escalar, mientras que la demanda de los recursos crece, se puede agregar más memoria y procesadores.

Ventajas

- La ventaja de MPI sobre otras bibliotecas de paso de mensajes, es que los programas que utilizan la biblioteca son portables (dado que MPI ha sido implementado para casi toda arquitectura de memoria distribuida), y rápidos, (porque cada implementación de la librería ha sido optimizada para el hardware en la cual se ejecuta).

Desventajas

- El acceso remoto a memoria es lento.
- La programación puede ser complicada.

Programa MPI

- Con MPI el número de procesos requeridos se asigna antes de la ejecución del programa
- No se crean procesos adicionales mientras la aplicación se ejecuta
- A cada proceso se le asigna una variable que se denomina rank, la cual identifica a cada proceso. El control de la ejecución del programa se realiza mediante la variable rank

Programa MPI

- La variable rank permite determinar que proceso ejecuta determinada porción de código.
- En MPI se define un comunicator como una colección de procesos, los cuales pueden enviar mensajes el uno al otro. El comunicator básico se denomina MPI_COMM_WORLD.
- MPI_COMM_WORLD agrupa a todos los procesos activos durante la ejecución de una aplicación.

Llamadas de MPI

- Las llamadas de MPI se dividen en cuatro clases:
  - Llamadas utilizadas para inicializar, administrar y finalizar comunicaciones.
  - Llamadas utilizadas para transferir datos entre un par de procesos.
  - Llamadas para transferir datos entre varios procesos.
  - Llamadas utilizadas para crear tipos de datos definidos por el usuario.

Funciones principales MPI

- Las funciones principales de MPI son:
- MPI_Init
- MPI_Finalize
- MPI_Comm_size
- MPI_Comm_rank

Mensajes en MPI

- Un mensaje está conformado por el cuerpo del mensaje, el cual contiene los datos a ser enviados, y su envoltura, que indica el proceso fuente y el destino.

- El cuerpo del mensaje en MPI se conforma por tres piezas de información: buffer, tipo de dato y count.

Mensajes en MPI

- El buffer, es la localidad de memoria donde se encuentran los datos de salida o donde se almacenan los datos de entrada.

- El tipo de dato, indica el tipo de los datos que se envían en el mensaje. En casos simples, éste es un tipo básico o primitivo, por ejemplo, un número entero, y que en aplicaciones más avanzadas puede ser un tipo de dato construido a través de datos primitivos.

Mensajes en MPI

- El count es un número de secuencia que junto al tipo de datos permiten al usuario agrupar ítems de datos de un mismo tipo en un solo mensaje.

Envoltura de un mensaje MPI

- La envoltura de un mensaje en MPI, consta de cuatro partes:
  - la fuente
  - el destino
  - el comunicador
  - y una etiqueta.

Envoltura de un mensaje MPI

- La fuente identifica al proceso transmisor.
- El destino identifica al proceso receptor.
- El comunicator especifica el grupo de procesos a los cuales pertenecen la fuente y el destino.
- La etiqueta (tag) permite clasificar el mensaje, es un entero definido por el usuario que puede ser utilizado para distinguir los mensajes que recibe un proceso.

Envoltura de un mensaje MPI

- Por ejemplo:
  - Se tienen dos procesos A y B. El proceso A envía dos mensajes al proceso B, ambos mensajes contienen un dato.
  - Uno de los datos es utilizado para realizar un cálculo, mientras el otro es utilizado para imprimirlo en pantalla.
  - El proceso A utiliza diferentes etiquetas para los mensajes.
  - El proceso B utiliza los valores de etiquetas definidos en el proceso A e identifica que operación deberá realizar con el dato de cada mensaje.

Tipos de Datos en MPI

• MPI_CHAR	signed char
• MPI_SHORT	signed short int
• MPI_INT	signed int
• MPI_LONG	signed long int
• MPI_UNSIGNED_CHAR	unsigned char

Tipos de Datos en MPI

• MPI_UNSIGNED_SHOT	unsigned short int
• MPI_UNSIGNED	unsigned int
• MPI_UNSIGNED_LONG	unsigned long int
• MPI_FLOAT	float
• MPI_DOUBLE	double
• MPI_LONG_DOUBLE	long double
• MPI_BYTE	
• MPI_PACKED	

Tipos de Datos en MPI

- MPI, nos permite crear nuevos tipos de datos (DataType):
  - n elementos de un vector con stride s
  - MPI_Type_vector Stride s fijo
  - MPI_Type_contiguous Strides s==1
  - MPI_Type_indexed Stride s variable
  - generales (struct)
  - Empaquetamiento
- MPI_Type_commit

Hay que llamar a esta rutina antes de poder usar el nuevo tipo de dados que hemos creado.

Operadores MPI

Operador	Tipo de operación
MPI_MAX	maximum
MPI_MIN	minimum
MPI_SUM	sum
MPI_PROD	product
MPI_LAND	logical and
MPI_BAND	bitwise and

Operadores MPI

Operador	**Tipo de operación**
MPI_LOR	logical or
MPI_BOR	bitwise or
MPI_LXOR	logical exclusive or
MPI_BXOR	bitwise exclusive or
MPI_MAXLOC	max value and location
MPI_MINLOC	min value and location

Ejemplo

- Imaginemos que queremos ejecutar el siguiente programa en 4 máquinas distintas de forma paralela.

```
int main(){
printf("Hola mundo\n");
}
```

```
gcc –Wall hola.c –o hola
./hola
```

- Una opción, es ir a las 4 máquinas, y ejecutar el programa en ellas (**POCO EFICIENTE**).

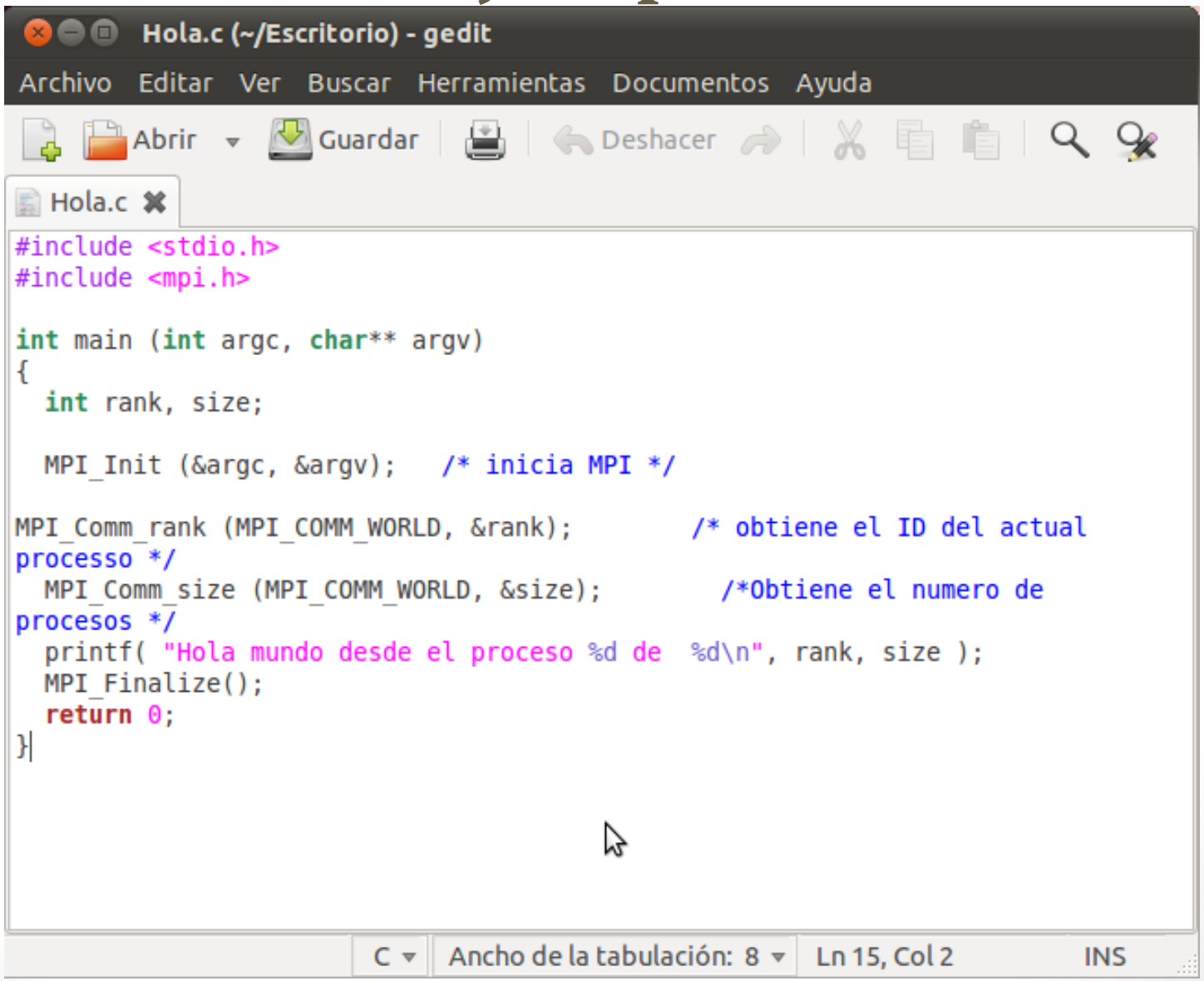
Ejemplo

- Otra opción es utilizar MPI, y desde una maquina, ejecutar el siguiente código (**MÁS EFICIENTE**)

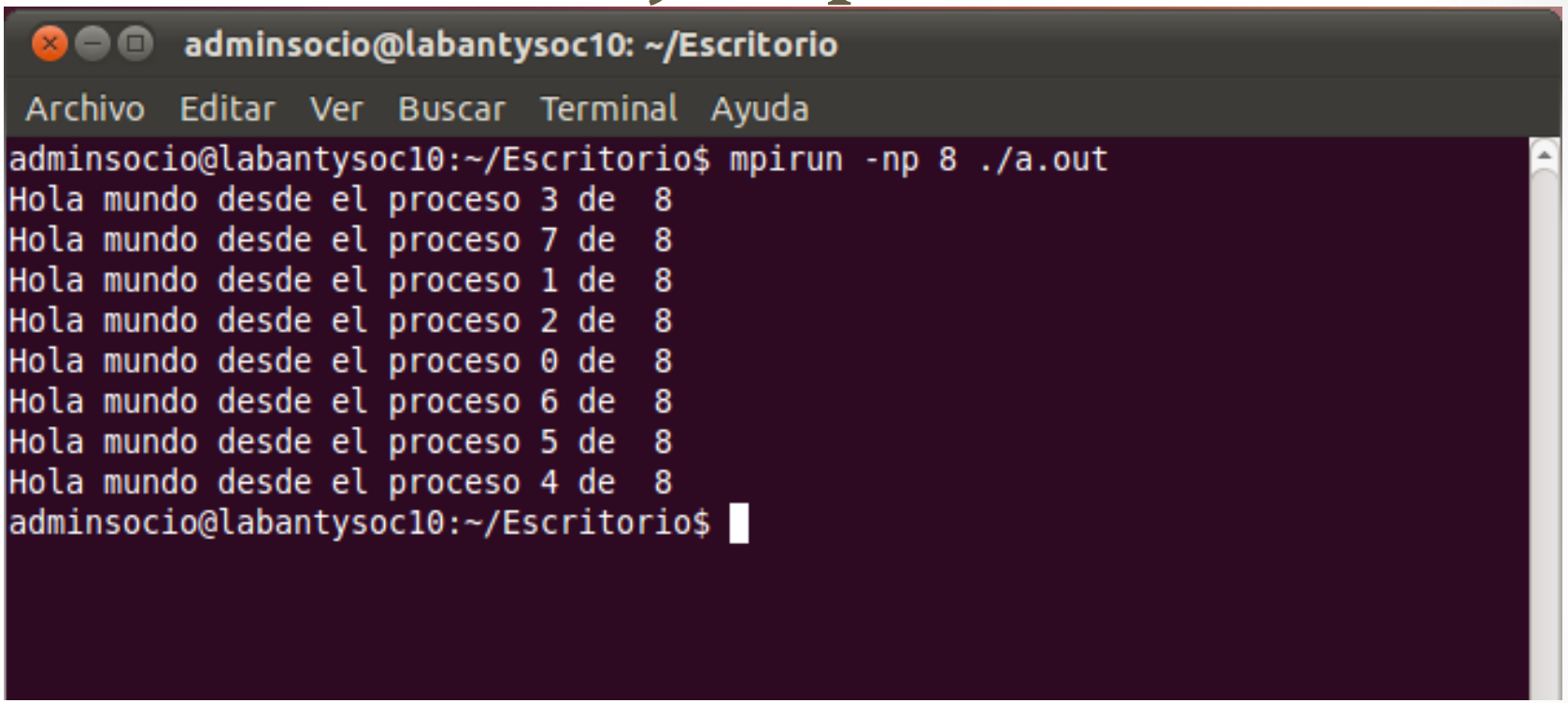
```
Int main() {
   MPI_Init(&argc,&argv);
   printf("Hola mundo\n");
   MPI_Finalice();
 }
```

```
mpicc –o hola hola.c
mpirun –n 4 hola
```

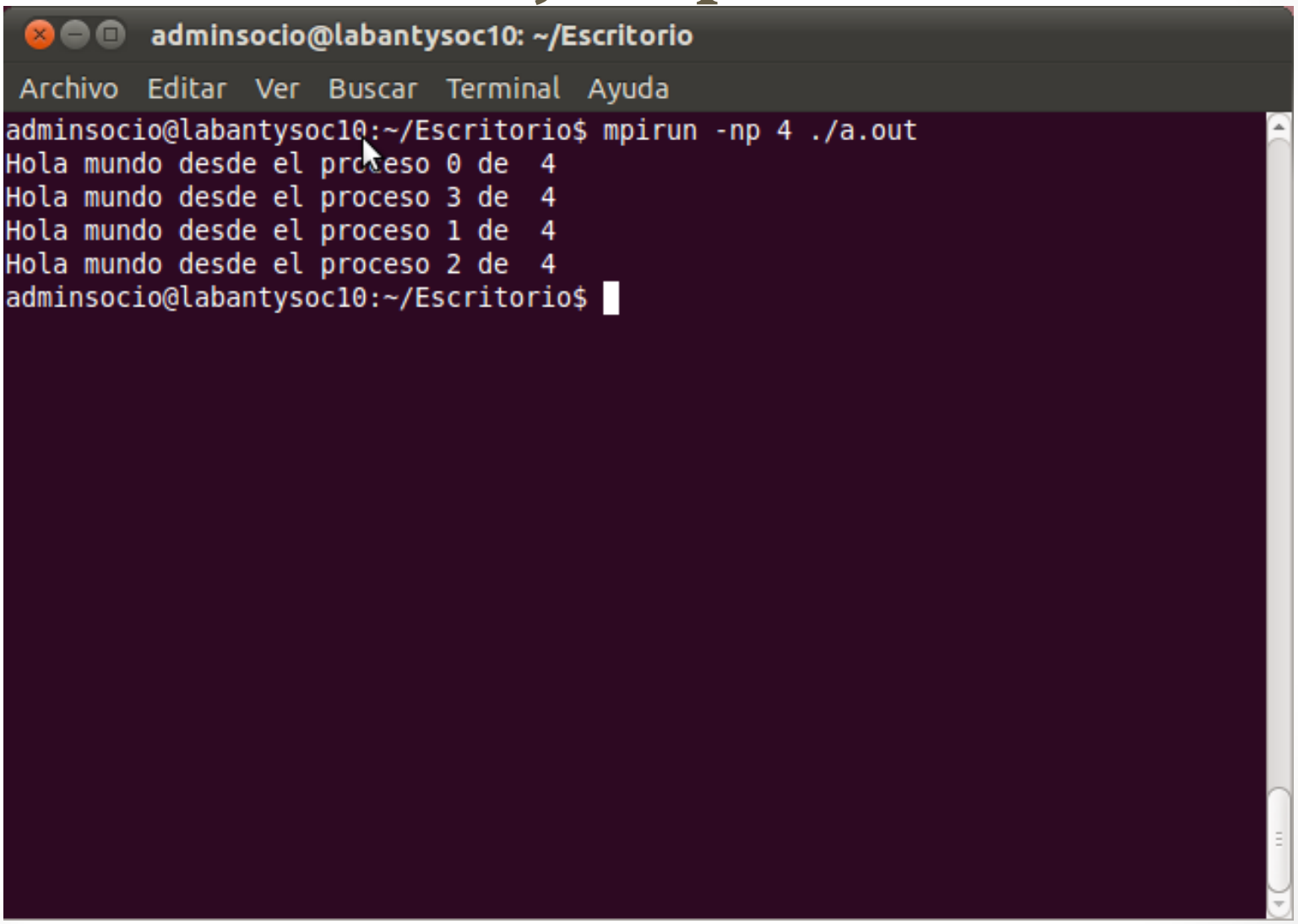
Ejemplo


Hola.c (~/Escritorio) - gedit
Archivo Editar Ver Buscar Herramientas Documentos Ayuda
Abrir
Guardar
Deshacer
Hola.c
#include <stdio.h>
#include <mpi.h>
int main (int argc, char** argv)
{
int rank, size;
MPI_Init (&argc, &argv); /* inicia MPI */
MPI_Comm_rank (MPI_COMM_WORLD, &rank); /* obtiene el ID del actual
processo */
MPI_Comm_size (MPI_COMM_WORLD, &size); /*Obtiene el numero de
procesos */
printf("Hola mundo desde el proceso %d de %d\n", rank, size);
MPI_Finalize();
return 0;
}
C
Ancho de la tabulación: 8
Ln 15, Col 2
INS

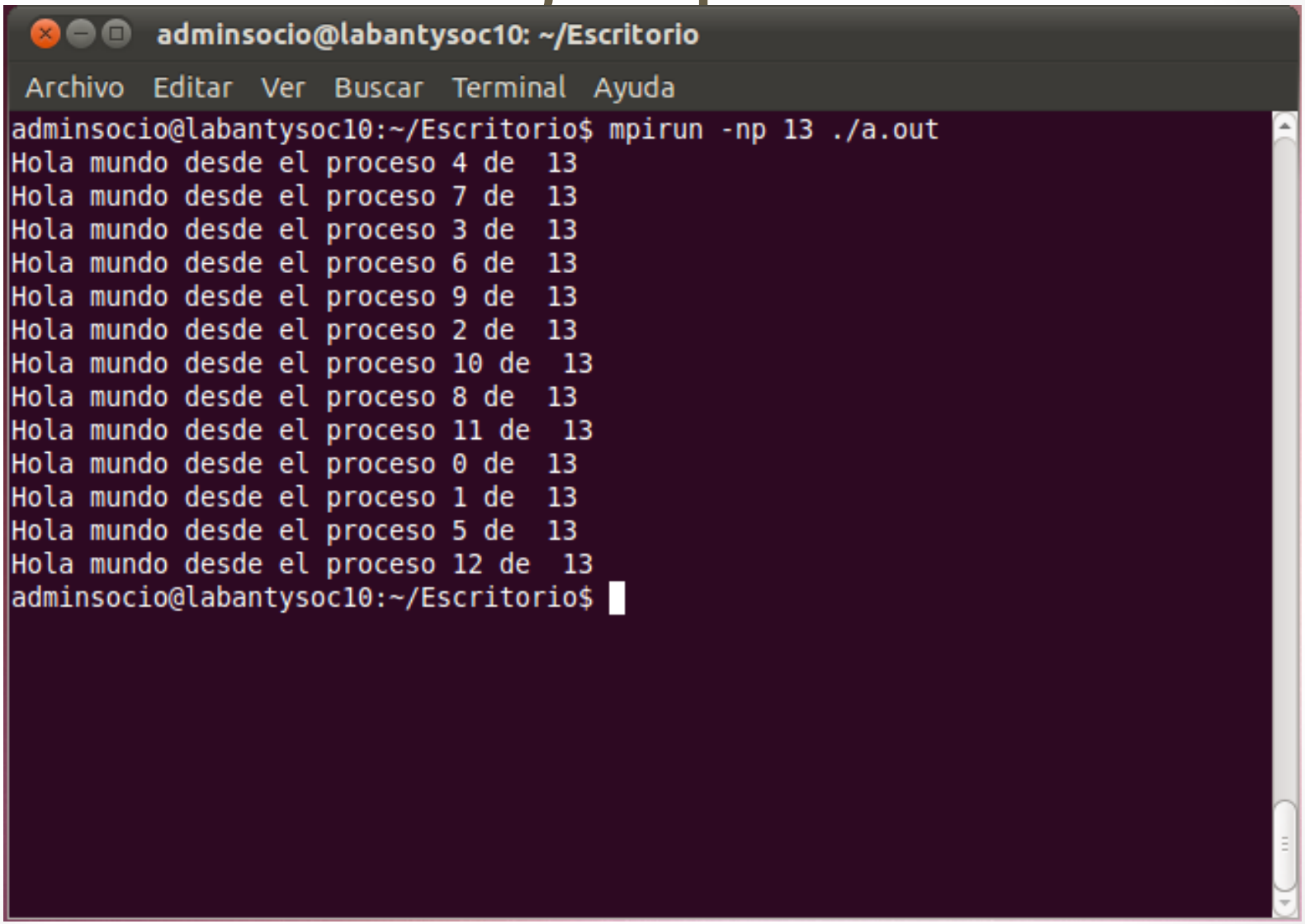
Ejemplo


adminsocio@labantysoc10: ~/Escritorio
Archivo Editar Ver Buscar Terminal Ayuda
adminsocio@labantysoc10:~/Escritorio$ mpirun -np 8 ./a.out
Hola mundo desde el proceso 3 de 8
Hola mundo desde el proceso 7 de 8
Hola mundo desde el proceso 1 de 8
Hola mundo desde el proceso 2 de 8
Hola mundo desde el proceso 0 de 8
Hola mundo desde el proceso 6 de 8
Hola mundo desde el proceso 5 de 8
Hola mundo desde el proceso 4 de 8
adminsocio@labantysoc10:~/Escritorio$

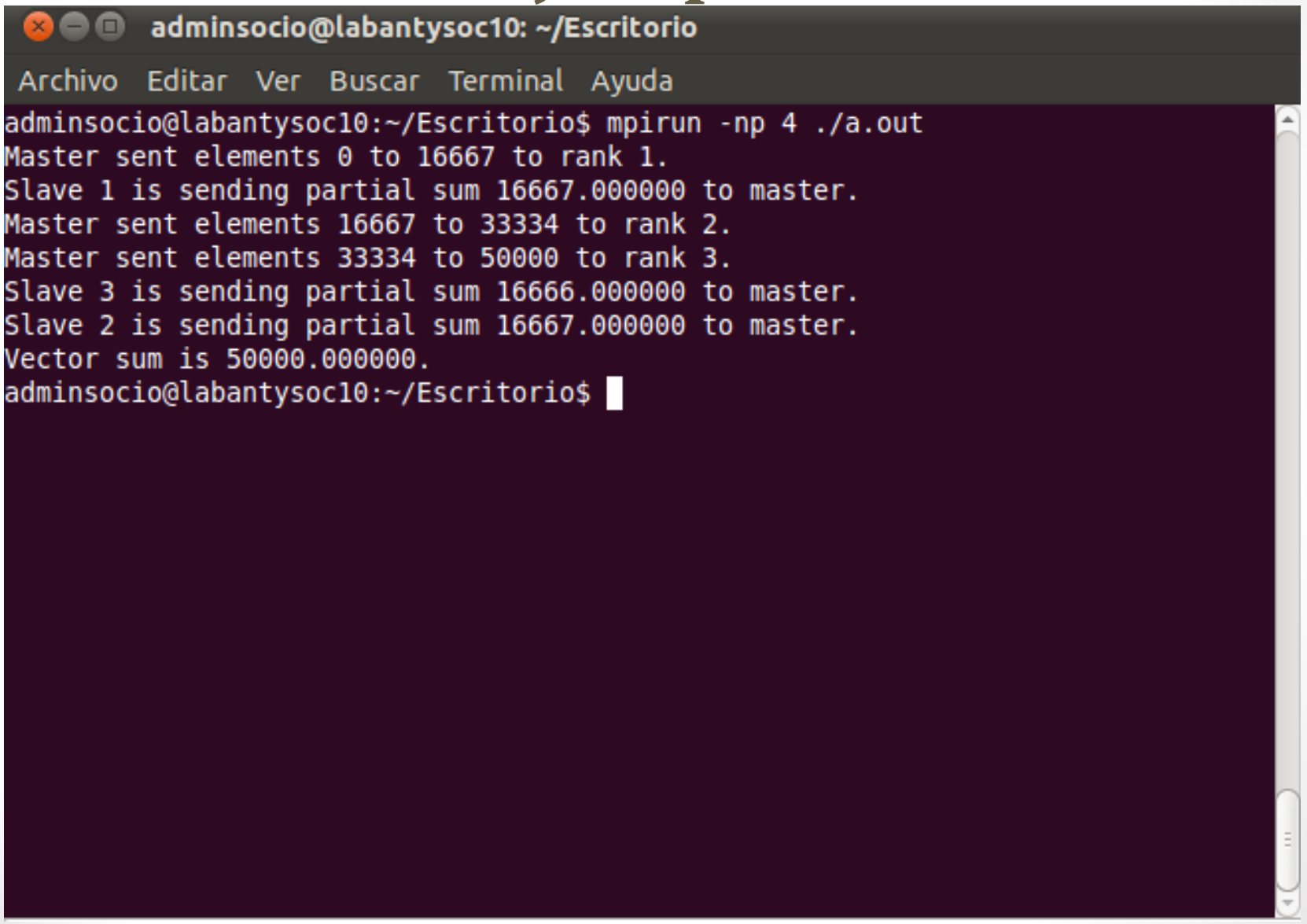
Ejemplo


adminsocio@labantysoc10: ~/Escritorio
Archivo Editar Ver Buscar Terminal Ayuda
adminsocio@labantysoc10:~/Escritorio$ mpirun -np 4 ./a.out
Hola mundo desde el proceso 0 de 4
Hola mundo desde el proceso 3 de 4
Hola mundo desde el proceso 1 de 4
Hola mundo desde el proceso 2 de 4
adminsocio@labantysoc10:~/Escritorio$

Ejemplo


adminsocio@labantysoc10: ~/Escritorio
Archivo Editar Ver Buscar Terminal Ayuda
adminsocio@labantysoc10:~/Escritorio$ mpirun -np 13 ./a.out
Hola mundo desde el proceso 4 de 13
Hola mundo desde el proceso 7 de 13
Hola mundo desde el proceso 3 de 13
Hola mundo desde el proceso 6 de 13
Hola mundo desde el proceso 9 de 13
Hola mundo desde el proceso 2 de 13
Hola mundo desde el proceso 10 de 13
Hola mundo desde el proceso 8 de 13
Hola mundo desde el proceso 11 de 13
Hola mundo desde el proceso 0 de 13
Hola mundo desde el proceso 1 de 13
Hola mundo desde el proceso 5 de 13
Hola mundo desde el proceso 12 de 13
adminsocio@labantysoc10:~/Escritorio$

Ejemplo


adminsocio@labantysoc10: ~/Escritorio
Archivo Editar Ver Buscar Terminal Ayuda
adminsocio@labantysoc10:~/Escritorio$ mpirun -np 4 ./a.out
Master sent elements 0 to 16667 to rank 1.
Slave 1 is sending partial sum 16667.000000 to master.
Master sent elements 16667 to 33334 to rank 2.
Master sent elements 33334 to 50000 to rank 3.
Slave 3 is sending partial sum 16666.000000 to master.
Slave 2 is sending partial sum 16667.000000 to master.
Vector sum is 50000.000000.
adminsocio@labantysoc10:~/Escritorio$

Referencias

- www-unix.mcs.anl.gov/mpi/mpich/
- www.lam-mpi.org/
- www-unix.mcs.anl.gov/mpi/
- www.mpi-forum.org
- http://mpi.deino.net/mpi_functions/
- http://www.mpi-forum.org
- http://www.mcs.anl.gov/mpi/mpich/
- http://www.lam-mpi.org/
- http://www.arcos.inf.uc3m.es/~ii_ac2/03-04/ejemplos_mpi.zip
- http://www.mhpcc.edu/training/workshop/mpi/MAIN.html

Referencias

- http://www.urjc.es/cat/hidra/manuales/MPI/tutorial_MPI.pdf
- http://www.cnb.uam.es/~carazo/practica_mpi.html
- http://numerix.us.es/pers/denk/_parallel/
- http://www.ldc.usb.ve/~ibanez/docencia/MPI/
- http://www.tc.cornell.edu/services/edu/topics/parallelvw.asp
- http://web.tiscali.it/Moncada/documenti/mpi.ppt
- http://www-copa.dsic.upv.es/docencia/iblanque/lcp/
- http://www.arcos.inf.uc3m.es/~mimpi/
- http://www.df.uba.ar/users/marcelo/
- http://www.dirinfo.unsl.edu.ar/~prgparal/Teorias/
- http://www.cecalc.ula.ve/documentacion/tutoriales/mpi/mpi.ps