

MPI

Irvyn xicale cabrera



MATRICULA: 201963582

**EJEMPLO 1**

Texto

Descripción generada automáticamente

Captura de pantalla de computadora

Descripción generada automáticamente

**Ejemplo 2**

este código demuestra la comunicación entre procesos MPI, donde los procesos envían y reciben datos entre sí dependiendo de su rango.

Captura de pantalla de computadora

Descripción generada automáticamente

Captura de pantalla de computadora

Descripción generada automáticamente

**MAPA MENTAL**

Diagrama

Descripción generada automáticamenteDiagrama

Descripción generada automáticamente

**¿en qué se parece el funcionamiento de las instrucciones mencionadas a los descriptores de archivos utilizados en la implementación de pipes o shared memory?**

* La comunicación de datos se realiza de manera punto a punto, lo que significa que los datos se envían desde un proceso (mediante send) a otro proceso (mediante recv).
* Cuando inicia el proceso de enviar se bloquea hasta terminar la transmisión y lo mismo paca cuando se recibe

**¿Cómo se puede evitar el interbloqueo?**

Implementando algún sistema de tiempo y errores para evitar que procesos se que queden en interbloqueo y una vez detectados tomar acciones. Pero lo mejor sería implementar una lógica de diseño para evitar estos casos

**EJEMPLO 3**

implementa una comunicación punto a punto entre dos procesos (con rango 1 y 5). Ambos procesos envían y reciben datos entre sí.

Texto

Descripción generada automáticamente

Captura de pantalla de computadora

Descripción generada automáticamente

**EJEMPLO 4**

este código divide la tarea de calcular la suma de números en múltiples procesos MPI, cada uno de los cuales calcula su propia suma parcial. Luego, las sumas parciales se envían al proceso raíz, donde se suman para obtener el resultado final de la suma.

**Texto

Descripción generada automáticamente**

Captura de pantalla de computadora

Descripción generada automáticamente

**EJEMPLO 5**

El proceso raíz inicializa la variable y luego la comparte con todos los demás procesos, lo que les permite acceder al mismo valor compartido.

Texto

Descripción generada automáticamente

Captura de pantalla de computadora

Descripción generada automáticamente

**EJEMPLO 6**

Cada proceso recibe un elemento de la lista y lo almacena en su propia variable recvbuf.

**Pantalla de computadora con letras

Descripción generada automáticamente con confianza media**

Captura de pantalla de computadora

Descripción generada automáticamente

**Comm.scatter(sendbuf, recvbuf, root)**

se utiliza para distribuir datos desde el proceso raíz (especificado por root) a todos los demás procesos en el comunicador.Es útil cuando deseas dividir una tarea en partes iguales entre los procesos.

**Parámetros:**

sendbuf: El arreglo de datos que se encuentra en el proceso raíz y se distribuirá a otros procesos.

recvbuf: El arreglo donde se almacenarán los datos distribuidos en cada proceso. En el proceso raíz, recvbuf puede ser None.

root: El rango del proceso raíz que posee los datos a distribuir. Los demás procesos deben especificar el mismo valor para root.

**Comm.scatterv(sendbuf, recvbuf, root)**

Permite distribuir segmentos de diferentes tamaños desde un arreglo de datos (sendbuf) en el proceso raíz a los demás procesos en el comunicador. Cada proceso especifica cuántos elementos debe recibir, lo que permite distribuciones no uniformes. Esto puede ser útil en situaciones en las que las tareas distribuidas tienen requisitos no uniformes de datos.

**Parámetros:**

sendbuf: El arreglo de datos que se encuentra en el proceso raíz y se distribuirá a otros procesos.

recvbuf: El arreglo donde se almacenarán los datos distribuidos en cada proceso. En el proceso raíz, recvbuf puede ser None.

root: El rango del proceso raíz que posee los datos a distribuir. Los demás procesos deben especificar el mismo valor para root.

**EJEMPLO 7**

Recopila los datos de los procesos y los agrega al array del proceso raíz 0

**Texto

Descripción generada automáticamente**

Captura de pantalla de computadora

Descripción generada automáticamente

**comm.Gather(sendbuf, root=0)**

se utiliza para recopilar datos desde todos los procesos y reunirlos en un solo proceso, el proceso raíz (especificado por root). Cada proceso proporciona un dato que se almacena en un arreglo en el proceso raíz.

**Parámetros:**

sendbuf: El dato que cada proceso desea recopilar. En el proceso raíz, este argumento puede ser None.

root (opcional): El rango del proceso raíz que recopilará los datos. Si no se especifica, se asume que el proceso raíz es el proceso con rango 0.

**comm.Gatherv(sendbuf, recvbuf, root=0)**

es similar a Gather, pero permite a cada proceso enviar diferentes cantidades de datos al proceso raíz. Cada proceso especifica cuántos elementos desea enviar, lo que permite una recopilación no uniforme.

**Parámetros:**

sendbuf: El arreglo de datos que cada proceso desea recopilar. En el proceso raíz, este argumento puede ser None.

recvbuf: El arreglo donde se almacenarán los datos recopilados en el proceso raíz. En los procesos no raíz, este argumento debe ser None.

root (opcional): El rango del proceso raíz que recopilará los datos. Si no se especifica, se asume que el proceso raíz es el proceso con rango 0.

**EJEMPLO 8**

Enviamos mediante Alltoall la data del proceso y recibimos la data de los otros procesos.

**Pantalla de computadora con letras

Descripción generada automáticamente con confianza media**

Captura de pantalla de computadora

Descripción generada automáticamente

**EJEMPLO 9**

Cada proceso contribuye con datos específicos, que se suman en el proceso raíz para obtener un resultado final. La suma se realiza con la operación de reducción MPI.SUM. Esto es útil para calcular sumas globales de datos distribuidos en un entorno paralelo.

Texto

Descripción generada automáticamente

Captura de pantalla de computadora

Descripción generada automáticamente

* **MPI.SUM:** Realiza una suma de todos los valores distribuidos entre los procesos y devuelve la suma total.
* **MPI.PROD:** Calcula el producto de todos los valores distribuidos entre los procesos y devuelve el producto total.
* **MPI.MAX:** Encuentra el valor máximo entre todos los valores distribuidos y devuelve el valor máximo.
* **MPI.MIN**: Encuentra el valor mínimo entre todos los valores distribuidos y devuelve el valor mínimo.
* **MPI.LAND:** Realiza una operación lógica "AND" entre todos los valores booleanos distribuidos y devuelve "VERDADERO" si todos los valores son verdaderos, de lo contrario, devuelve "FALSO".
* **MPI.LOR:** Realiza una operación lógica "OR" entre todos los valores booleanos distribuidos y devuelve "VERDADERO" si al menos un valor es verdadero, de lo contrario, devuelve "FALSO".
* **MPI.LXOR:** Realiza una operación lógica "XOR" entre todos los valores booleanos distribuidos y devuelve "VERDADERO" si un número impar de valores son verdaderos, de lo contrario, devuelve "FALSO".
* **MPI.BAND:** Realiza una operación "AND" a nivel de bits entre todos los valores enteros distribuidos y devuelve el resultado.
* **MPI.BOR:** Realiza una operación "OR" a nivel de bits entre todos los valores enteros distribuidos y devuelve el resultado.
* **MPI.BXOR:** Realiza una operación "XOR" a nivel de bits entre todos los valores enteros distribuidos y devuelve el resultado.
* **MPI.MINLOC:** Encuentra el valor mínimo entre todos los valores distribuidos y también devuelve la ubicación (rango) del proceso que contiene el valor mínimo.
* **MPI.MAXLOC:** Encuentra el valor máximo entre todos los valores distribuidos y también devuelve la ubicación (rango) del proceso que contiene el valor máximo.

**Ventajas de la unificación de long e int en Python**

* La unificación de long e int en Python 3 elimina la necesidad de preocuparse por la diferencia entre estos tipos de datos.
* La unificación permite realizar operaciones aritméticas con enteros grandes de manera más sencilla y coherente, ya que no es necesario cambiar entre tipos int y long.
* hace que el código sea más claro y fácil de entender, ya que no es necesario considerar las diferencias de tipo en la mayoría de los casos.