Comunicación mediante un hilo con tuberías en Kotlin

[crayon-6a9d57f38931e605799007/]

Aquí está la explicación paso a paso:

1. Importaciones:kotlinCopy codeimport java.io.IOException import java.io.PipedInputStream import java.io.PipedOutputStream Estas importaciones son necesarias para trabajar con las clases relacionadas con tuberías (pipes) y excepciones en Kotlin.
2. Función main:kotlinCopy codefun main() { Aquí comienza la función principal del programa. Es el punto de entrada para la ejecución del programa Kotlin.
3. Creación de tuberías de entrada y salida:kotlinCopy codeval pipeIn = PipedInputStream() val pipeOut = PipedOutputStream(pipeIn) Se crean dos objetos: pipeIn que es una tubería de entrada y pipeOut que es una tubería de salida. La tubería de salida se conecta a la tubería de entrada, de modo que lo que se escriba en pipeOut se pueda leer en pipeIn.
4. Creación de un hilo para el proceso hijo:kotlinCopy codeval childThread = Thread { // Código del proceso hijo aquí } Se crea un hilo (childThread) que ejecutará el código del proceso hijo. Este hilo es necesario porque el proceso hijo debe ejecutarse en paralelo al proceso padre.
5. Código del proceso hijo dentro del hilo:kotlinCopy codetry { val mensaje = "Hola, soy el hijo" pipeOut.write(mensaje.toByteArray()) pipeOut.close() } catch (e: IOException) { e.printStackTrace() } Dentro del hilo del proceso hijo, se escribe un mensaje en la tubería de salida pipeOut. Primero, se convierte el mensaje en un arreglo de bytes (mensaje.toByteArray()) y

se escribe en la tubería. Luego, se cierra la tubería de salida para indicar que no se enviarán más datos.

6. Inicio del hilo del proceso hijo:kotlinCopy codechildThread.start() Se inicia el hilo del proceso hijo para que comience a ejecutar su código en paralelo al proceso padre.
7. Lectura del mensaje en el proceso padre:kotlinCopy codeval buffer = ByteArray(100) val bytesRead = pipeIn.read(buffer) val mensajeRecibido = String(buffer, 0, bytesRead) En el proceso padre, se crea un búfer (buffer) para almacenar los datos que se leerán de la tubería de entrada. Se utiliza pipeIn.read(buffer) para leer los datos de la tubería de entrada en el búfer y se registra la cantidad de bytes leídos en bytesRead. Luego, se convierte el búfer en una cadena de texto (mensajeRecibido) para mostrarlo por pantalla.
8. Impresión del mensaje recibido en el proceso padre:kotlinCopy codeprintln("El padre recibió el mensaje:") println(mensajeRecibido)
9. Cierre de la tubería de entrada:kotlinCopy codepipeIn.close() Finalmente, se cierra la tubería de entrada para liberar recursos.