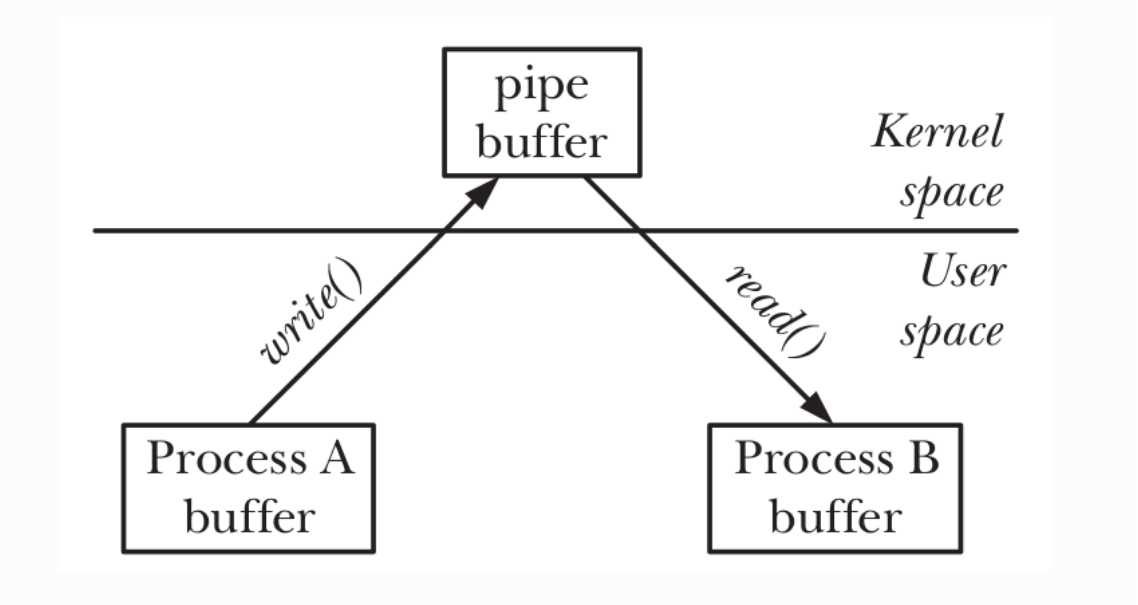
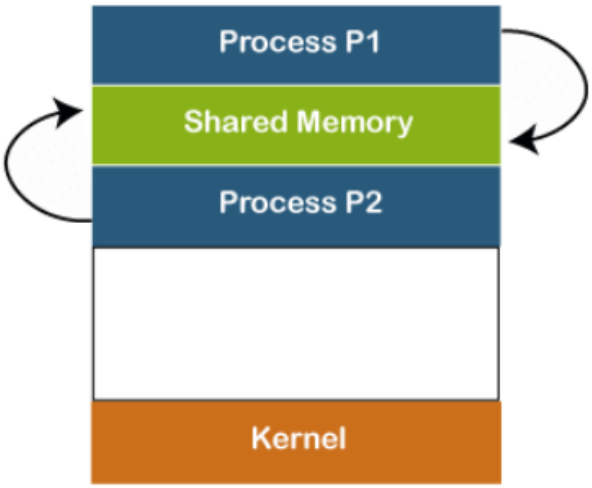
Como vimos anteriormente con los scripts de Windows, la manera que tiene el OS de conectar procesos es mediante tuberías (pipes)



Un pipe no deja de ser un espacio en memoria al cual tienen acceso ambos procesos. En la imagen anterior se puede ver como el proceso A escribe en el pipe. Por el contrario, el proceso B hace una llamada a read() para leer la información que le ha dejado el proceso A en el buffer del pipe.
Pero el pipe tiene sus limitaciones. Una de ellas es que es unidireccional. Ambos procesos comparten un espacio en memoria, pero solo uno puede escribir y solo uno puede leer.

Cuando se trata de comunicación de procesos el Java ocurre algo parecido pero bidireccional.



Algo como esto es lo que ocurre cuando lanzamos 2 procesos en Java y queremos que se comuniquen. Comparten una región de memoria que se usa como Buffer para que ambos procesos puedan leer y escribir.

¿Como se crean procesos y como se comunican entre ellos en Java?

Vamos a utilizar las clases ProcessBuilder y Process de la librería java.io para crear, comunicar y destruir procesos.

ProcessBuilder se encarga de “preparar” y “definir” al futuro proceso que será lanzado. En otras palabras, se encarga de añadirle atributos como entorno de ejecución, en que directorio se ejecutará, que archivo ejecutará, etc.

A continuación, los métodos de ProcessBuilder:

MÉTODOS	MISIÓN
ProcessBuilder command (String argumentos ...)	Define el programa que se quiere ejecutar indicando sus argumentos como una lista de cadenas separadas por comas.
List < String > command ()	Devuelve todos los argumentos del objeto **ProcessBuilder**.
Map < String , String > environment ()	Devuelve en una estructura Map las variables de entorno del objeto **ProcessBuilder**.
ProcessBuilder redirectError (File file)	Redirige la salida de error estándar a un fichero.
ProcessBuilder redirectInput (File file)	Establece la fuente de entrada estándar en un fichero.
ProcessBuilder redirectOutput (File file)	Redirige la salida estándar a un fichero.
File directory()	Devuelve el directorio de trabajo del objeto **ProcessBuilder**.
ProcessBuilder directory(File directorio)	Establece el directorio de trabajo del objeto **ProcessBuilder**.
Process start ()	Inicia un nuevo proceso utilizando los atributos del objeto **ProcessBuilder**.

En cambio, la clase Process es la que va a recoger el proceso en sí, la clase con la que trabajaremos cuando queramos interactuar con el proceso.

MÉTODOS	MISIÓN
InputStream getInputStream ()	Devuelve el flujo de entrada conectado a la salida normal del subproceso. Nos permite leer el stream de salida del subproceso, es decir, podemos leer lo que el comando que ejecutamos escribió en la consola.
int waitFor ()	Provoca que el proceso actual espere hasta que el subproceso representado por el objeto **Process** finalice. Devuelve 0 si ha finalizado correctamente.
InputStream getErrorStream()	Devuelve el flujo de entrada conectado a la salida de error del subproceso. Nos va a permitir poder leer los posibles errores que se produzcan al lanzar el subproceso.
OutputStream getOutputStream()	Devuelve el flujo de salida conectado a la entrada normal del subproceso. Nos va a permitir escribir en el stream de entrada del subproceso, así podemos enviar datos al subproceso que se ejecute.
void destroy ()	Elimina el subproceso.
int exitValue ()	Devuelve el valor de salida del subproceso.
boolean isAlive ()	Comprueba si el subproceso representado por **Process** está vivo

Os dejo este código como ejemplo en el cual creamos un proceso que va a ejecutar el comando set para que nos devuelva las variables de entorno. ¿Cómo funciona?

- P1 (Proceso padre) crea el proceso hijo P2
- P2 se ejecuta y deja en la memoria compartida su salida (lo que devuelve el comando set)
- P1 accede a la memoria compartida en la cual su proceso hijo ha dejado el mensaje. Accedemos con p.getInputStream();
- Ya tenemos en la variable “is” el mensaje.
- Solo nos queda leerlo.

```java
//Preparamos el proceso ("comando", "argumento1", "argumento2") en este caso
ProcessBuilder pb = new ProcessBuilder("cmd.exe", "/C", "set");
//Creamos el proceso con start()
Process p2 = pb.start();

//p2 se ha ejecutado y nos ha dejado su salida en el buffer.
//Recogemos su salida y la pasamos como input a este proceso P1
InputStream is = p2.getInputStream();

//Leemos caracter por caracter
int c;
while((c = is.read()) != -1){
    System.out.print((char) c);
}

//Verificamos que el proceso se ha ejecutado correctamente
//Si se ha ejecutado correctamente, su valor de salida deberia ser 0
int exitVal;
try {
    exitVal = p2.waitFor();
    System.out.println("Valor de Salida: " + exitVal);
} catch (InterruptedException e) {
    e.printStackTrace();
}
```