<https://github.com/thaispkt/java_course_demo/tree/master/src/Baitapbuoi10_bai1>

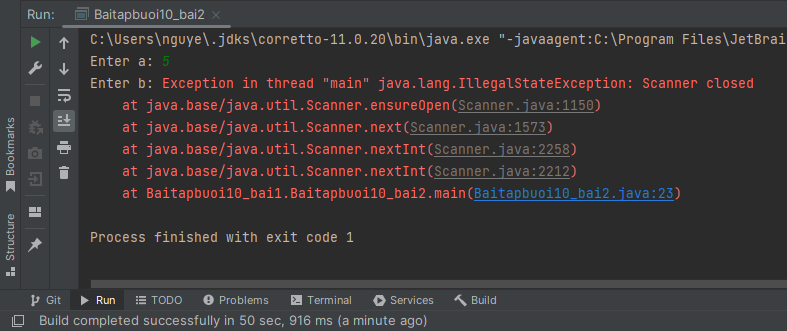
Fraction fraction1 = new Fraction();

fraction1.importData(1);

Fraction fraction2 = new Fraction();

fraction2.importData(2);

import java.util.Scanner;  
  
public class Baitapbuoi10\_bai2 {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.print("Enter a: ");  
 int a = scanner.nextInt();  
 scanner.close();  
 System.*out*.print("Enter b: ");  
 int b = scanner.nextInt();  
 scanner.close();  
 }  
}



|  |
| --- |
| Enter a: 5  Enter b: Exception in thread "main" java.lang.IllegalStateException: Scanner closed  at java.base/java.util.Scanner.ensureOpen(Scanner.java:1150)  at java.base/java.util.Scanner.next(Scanner.java:1573)  at java.base/java.util.Scanner.nextInt(Scanner.java:2258)  at java.base/java.util.Scanner.nextInt(Scanner.java:2212)  at Baitapbuoi10\_bai1.Baitapbuoi10\_bai2.main(Baitapbuoi10\_bai2.java:23)  Process finished with exit code 1 |

Cause:

Khi ta close scanner() nghĩa là ta đã close terminal đang chạy -> 1 khi đã đóng terminal thì ko open lại được

scanner.close();

You cannot reopen a closed Scanner. Once a Scanner object is closed, it cannot be used again. If you need to read input from the user multiple times, you should create a new Scanner object for each time you need to read input.

The exception occurs because you are trying to read input from the **Scanner** after you have closed it. Once the **Scanner** is closed using the **close()** method, it cannot be reopened for reading input.

Remember to close the **Scanner** when you are done reading all the required input from it.

Sure, here is the C program that corresponds to the Java program you provided:

C

#include <stdio.h>

int main() {

  int a, b;

  // Create a Scanner object to read input from the user.

  // This is equivalent to `Scanner scanner = new Scanner(System.in);` in Java.

  FILE\* input = stdin;

  // Read the value of `a` from the user.

  printf("Enter a: ");

  fscanf(input, "%d", &a);

  // Close the Scanner object.

  // This is equivalent to `scanner.close();` in Java.

  fclose(input);

  // Read the value of `b` from the user.

  printf("Enter b: ");

  fscanf(input, "%d", &b);

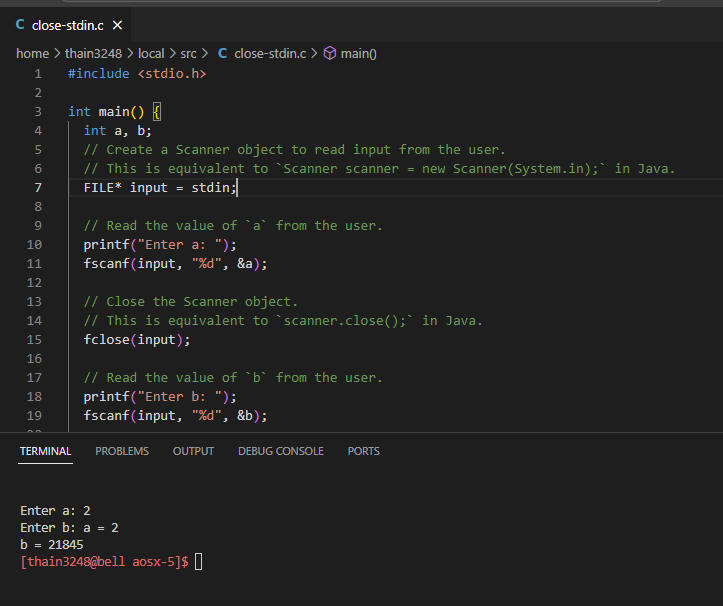
  // Print the values of `a` and `b`.

  printf("a = %d\n", a);

  printf("b = %d\n", b);

  return 0;

}



Nhập được a = 2 những ko nhập được b. b in ra là giá trị rác !!!

In C, **stdin** is a predefined file pointer associated with the standard input stream, which is typically used to read input from the terminal. It is equivalent to the **System.in** stream in Java.

In the provided C program **close-stdin.c**, the code attempts to close the **stdin** file pointer using **fclose(input);**. This is incorrect because **stdin** is a predefined file pointer managed by the system, and you should not close it explicitly. Closing **stdin** can lead to unexpected behavior, as it will prevent further input reading from the terminal.

The **stdin** file pointer is opened and managed by the operating system, and once it is closed, there is no way to reopen it within the context of the same program. The reason is that **stdin** represents the standard input stream of the current process (in this case, the terminal). Once closed, it cannot be reopened within the same process.

Regarding your question about whether it is related to the terminal, yes, **stdin** is typically connected to the terminal (console) by default. When you run the program in the terminal, the standard input stream (**stdin**) is used to read input from the user. If you close it, you lose the ability to read input from the terminal, and there is no standard way to reopen it within the same program.

To fix the issue in the provided C program, you should not close the **stdin** file pointer, and instead, directly use **stdin** for input reading without explicitly opening or closing it:

Ans2:

Sure, I can explain FILE\* input = stdin and why we can't reopen input (stdin) after closing it.

The stdin variable is a global constant that points to the standard input stream. **This stream is usually the keyboard, but it can also be a pipe or a file**. When we close the stdin stream, we are essentially closing the connection to the keyboard or other input device. This means that we can no longer read input from the device.

The fclose() function closes the stdin stream and flushes any buffered data. Once the stdin stream is closed, it cannot be reopened. **This is because the operating system does not allow multiple processes to access the same input device at the same time.**

Cách 2:

Sure! Below is the equivalent C program based on the provided Java program:

c

Copy code

#include <stdio.h>

int main() {

int a, b;

printf("Enter a: ");

scanf("%d", &a);

printf("Enter b: ");

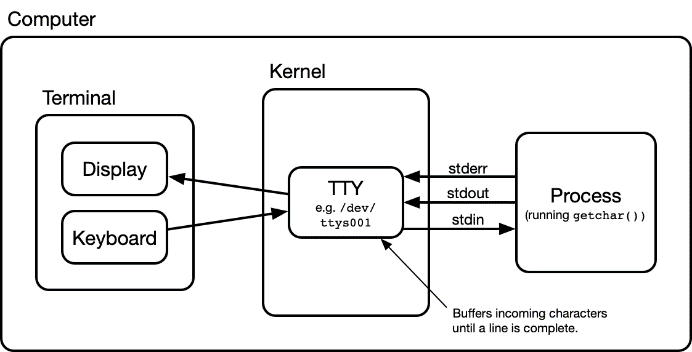
scanf("%d", &b);

return 0;

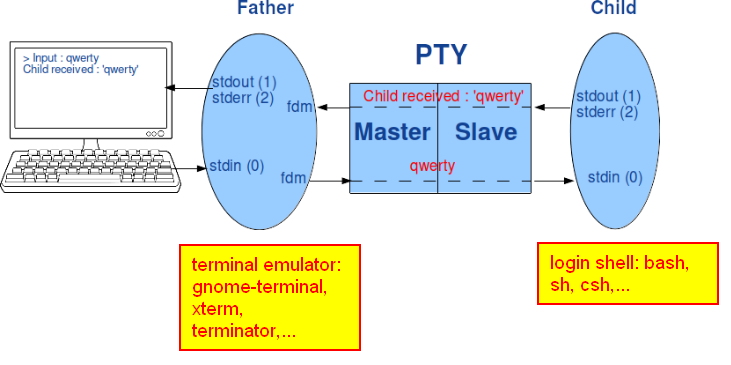
}

ssh, stdin stdout, terminal, pseudo terminal, pty, ptty, bash shell sh shell option -t ssh hangup.docx

<https://stackoverflow.com/questions/7741930/getchar-and-stdin>

[](https://i.stack.imgur.com/DXjqx.png)

The box entitled "Terminal" in above diagram is your terminal window, e.g. xterm, iTerm, or Terminal.app.



Most GUI terminal emulators will start the program on the slave side with a TERM variable whose value matches their terminal emulation on the master side.

<http://manpages.ubuntu.com/manpages/bionic/man1/gnome-terminal.1.htm>

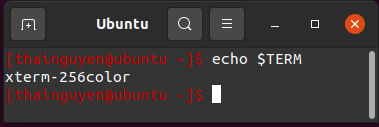
gnome-terminal — is a terminal emulation application.

**gnome-terminal** tương đương chức năng với chương trình **xterm**

[thainguyen@ubuntu ~]$ echo $TERM

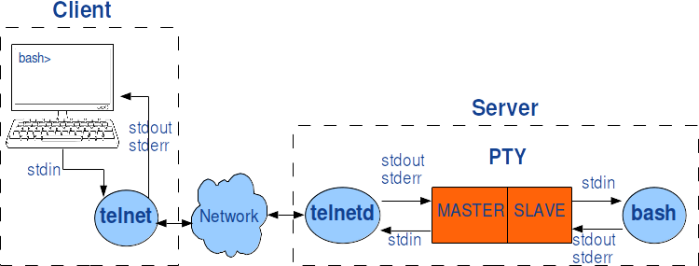
xterm-256color

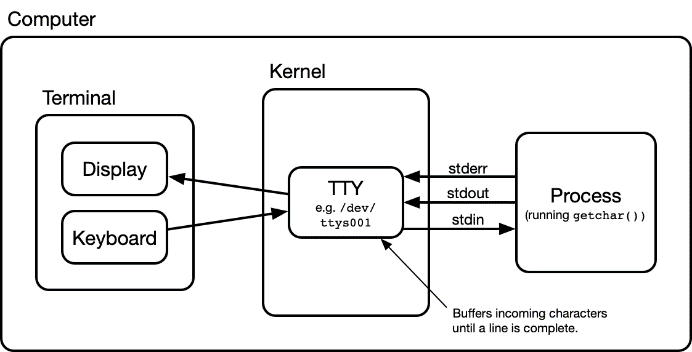
[thainguyen@ubuntu ~]$ gnome-terminal



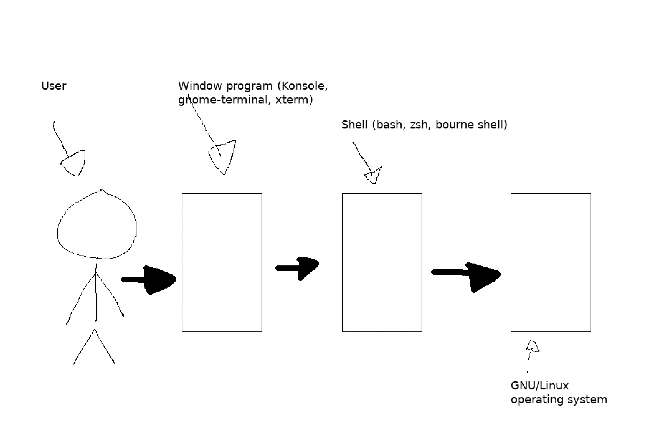
<https://superuser.com/questions/343729/what-is-the-difference-between-xterm-and-terminal-in-ubuntu>

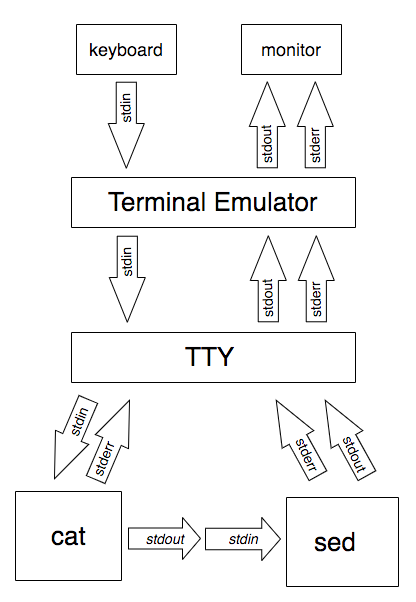
**Figure 4: Description of a telnet session**



[](https://i.stack.imgur.com/DXjqx.png)

The box entitled "Terminal" in above diagram is your terminal window, e.g. xterm, iTerm, or Terminal.app.







tty trong Linux chính là cái cổng COM (serial port)

[thainguyen@ubuntu ~]$ tty

/dev/pts/15

[thainguyen@ubuntu ~]$ stty -a

speed 38400 baud; rows 45; columns 203; line = 0;

