Suppose we would like to write a program called monitor which allows two other programs to communicate with each other through the monitor while inspecting the send messages. It can be executed as follows:  

`> monitor prog1 prog2`

To simplify the implementation, we assume that prog1 (e.g., `ls`) will be printing some messages on the standard output while the prog2 (e.g., `sort`) reads from standard input and writes something into standard output.

Our monitor can run these two programs as children and let the first one send messages to the second one through our monitor. So our monitor can inspect the incoming messages. After inspection, it sends the message to the next program as is.

Again to simplify the implementation, we assume that our monitor simply looks for digits. When it detects a digit in the message, our monitor simply prints it on the stderr (this monitoring task can be complicated for other purposes).

Logically these programs will have the following relationship:

```
prog1(child1)[1] --> Pipe1 -->[0] Monitor (Parent)[1] --> Pipe2 -->[0] prog2(child2)
```

You are asked to create the necessary pipes, child processes and connect them as explained in the above scenario. You can ignore most of the error checking to make your solution clear, but you need to close all unnecessary file descriptors and check what read-write return.

```
> monitor /bin/ls  /usr/bin/sort
```

```c
#include <fcntl.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/stat.h>

int main (int argc, char *argv[]) {
    int Pipe1[2];
    int Pipe2[2];
    int ch1pid, ch2pid, numread, numwrite;
    char buf;
    ....
    if(ch1pid == 0){ /* child 1 */
        ....
        exec1(argv[1],"progl",NULL);
    }
    ....
    if(ch2pid == 0){ /* child 1 */
        ....
        exec1(argv[2],"prog2",NULL);
    }
    ....
```