### **13. e) STREAMS message/PIPEs/FIFO:pipe, popenand pcloseFunctions**

**Filter to add two numbers, using standard I/O.**

**Objectives:**

1. To learn about STREAMS message/PIPEs/FIFO:pipe, popenand pcloseFunctions.

**Theory:**

Coprocesses-

A UNIX system filter is a program that reads from standard input and writes to standard output. Filters are normally connected linearly in shell pipelines. A filter becomes a *coprocess* when the same program generates the filter's input and reads the filter's output.

The Korn shell provides coprocesses. The Bourne shell, the Bourne-again shell, and the C shell don't provide a way to connect processes together as coprocesses. A coprocess normally runs in the background from a shell, and its standard input and standard output are connected to another program using a pipe. Although the shell syntax required to initiate a coprocess and connect its input and output to other processes is quite contorted, coprocesses are also useful from a C program.

Whereas popen gives us a one-way pipe to the standard input or from the standard output of another process, with a coprocess, we have two one-way pipes to the other process: one to its standard input and one from its standard output. We want to write to its standard input, let it operate on the data, and then read from its standard output.

##### Example

Let's look at coprocesses with an example. The process creates two pipes: one is the standard input of the coprocess, and the other is the standard output of the coprocess. shows this arrangement.

The program in [Figure 15.17](http://poincare.matf.bg.ac.rs/~ivana/courses/ps/sistemi_knjige/pomocno/apue/APUE/0201433079/ch15lev1sec4.html" \l "ch15fig17) is a simple coprocess that reads two numbers from its standard input, computes their sum, and writes the sum to its standard output. (Coprocesses usually do more interesting work than we illustrate here. This example is admittedly contrived so that we can study the plumbing needed to connect the processes.)

We compile this program and leave the executable in the file add2.

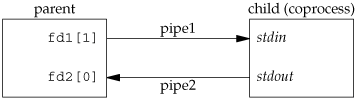
The program in [Figure 15.18](http://poincare.matf.bg.ac.rs/~ivana/courses/ps/sistemi_knjige/pomocno/apue/APUE/0201433079/ch15lev1sec4.html" \l "ch15fig18) invokes the add2 coprocess after reading two numbers from its standard input. The value from the coprocess is written to its standard output.

Here, we create two pipes, with the parent and the child closing the ends they don't need. We have to use two pipes: one for the standard input of the coprocess and one for its standard output. The child then calls dup2 to move the pipe descriptors onto its standard input and standard output, before calling execl.

If we compile and run the program in [Figure 15.18](http://poincare.matf.bg.ac.rs/~ivana/courses/ps/sistemi_knjige/pomocno/apue/APUE/0201433079/ch15lev1sec4.html" \l "ch15fig18), it works as expected. Furthermore, if we kill the add2 coprocess while the program in [Figure 15.18](http://poincare.matf.bg.ac.rs/~ivana/courses/ps/sistemi_knjige/pomocno/apue/APUE/0201433079/ch15lev1sec4.html" \l "ch15fig18) is waiting for our input and then enter two numbers, the signal handler is invoked when the program writes to the pipe that has no reader.

Recall from [Figure 15.1](http://poincare.matf.bg.ac.rs/~ivana/courses/ps/sistemi_knjige/pomocno/apue/APUE/0201433079/ch15lev1sec1.html" \l "ch15fig01) that not all systems provide full-duplex pipes using the pipe function. In [Figure 17.4](http://poincare.matf.bg.ac.rs/~ivana/courses/ps/sistemi_knjige/pomocno/apue/APUE/0201433079/ch17lev1sec2.html" \l "ch17fig04), we provide another version of this example using a single full-duplex pipe instead of two half-duplex pipes, for those systems that support full-duplex pipes.

##### **Figure Driving a coprocess by writing its standard input and reading its standard output**



#include <stdlib.h>

#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

#include<string.h>

#include <stdlib.h>

int main()

{

int status = 0;

int pfds[2];

int pfds2[2];

int pfds3[2];

int pfds4[2];

int val = 0;

int val2 = 0;

pipe(pfds);

pipe(pfds2);

pipe(pfds3);

pipe(pfds4);

printf("Enter FIRST Number: ");

scanf("%i", &val);

printf("Enter SECOND Number: ");

scanf("%i", &val2);

if (fork() == 0)

{

//Child

printf("I'm Child 1 and I'm Calculating Sum Of Numbers\n");

read(pfds[0], &val, sizeof(val));

read(pfds2[0], &val2, sizeof(val));

int sum = val + val2;

int sub = val - val2;

write(pfds3[1], &sum, sizeof(val));

write(pfds4[1], &sub, sizeof(val));

if (fork() == 0)

{

read(pfds3[0], &val, sizeof(val));

printf("Sum Of Numbers Is : %i\n", val);

read(pfds4[0], &val, sizeof(val));

exit(1);

}

else

{

wait(&status);

}

exit(1);

}

else

{

//wait(&status);

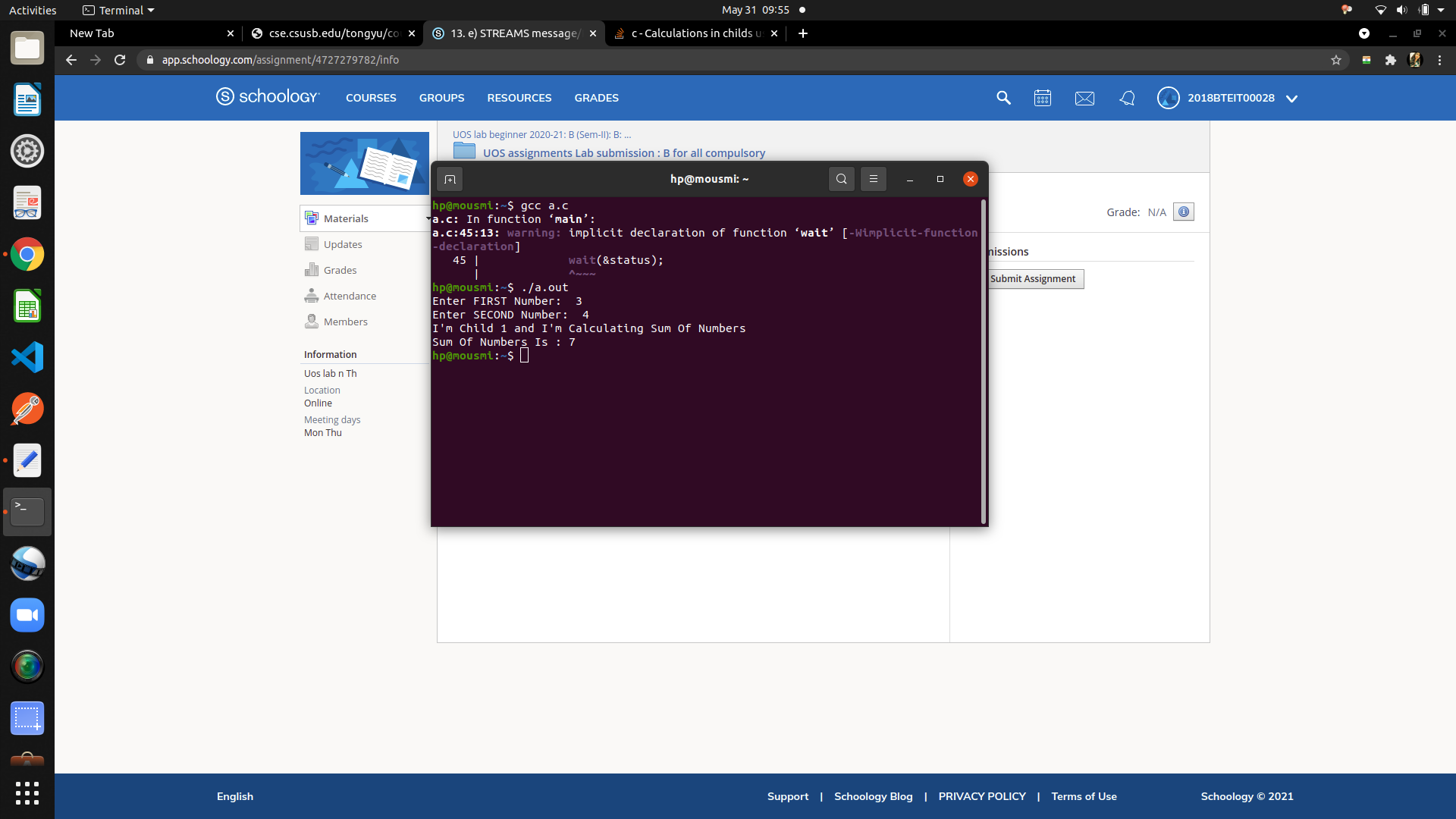
write(pfds[1], &val, sizeof(val));

write(pfds2[1], &val2, sizeof(val));

wait(&status);

}

}



#### Conclusion:

Filter to add two numbers, using standard I/O using STREAMS message/PIPEs/FIFO:pipe, popenand pcloseFunctions implemented

#### References :

http://cse.csusb.edu/tongyu/courses/cs460/labs/lab4.php