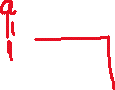
LAB 7

* Threads the last exam topic
* All what we need to know will be covered in this lab 😊
* Processes of type fork
* Child processes ->memory independent (copy of parents memory in the moment of the child creation)
* Var declared in the parent, child has a copy of the var
* Pipes ->unidirectional
* Pipe = data structure incorporated into a vector(with 2 components->indexes: 0->read,1->write)



Protocol:

Close(p[1])

Read(p[0],&a,1)

First write, then read

Exercices:

-define a={1,2,3,4} prg create a child process->result sum of the vector

Parent->sum of the first part of the vector

Child->the second sum

P={1,2}->3 C={3,4}->7

* 10

Scheleton:

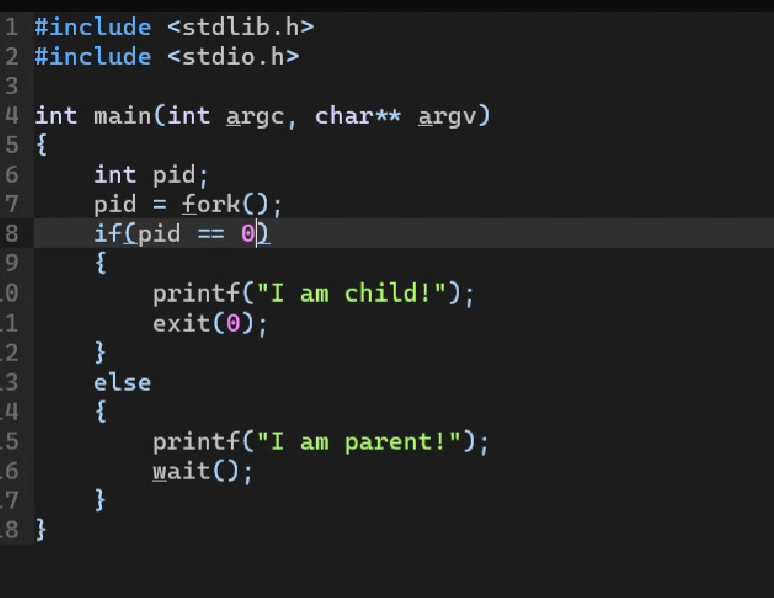
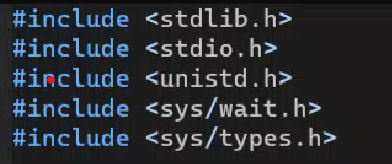
Create a new c program

Declare var pid -> used for fork

Have an if else for child + the parent

Child-> exit(0)

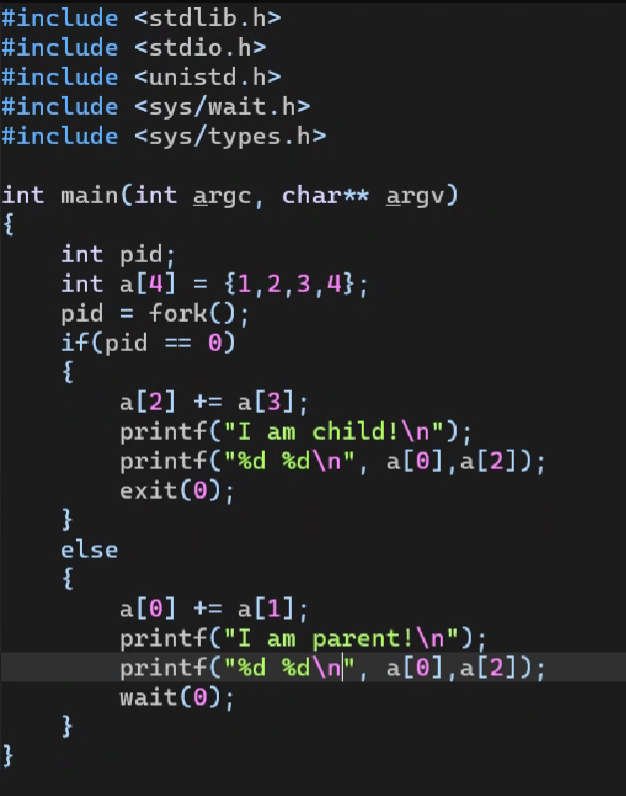
Parent-> wait()



Execute: Text

Description automatically generated

We divide the problem using the divide and conquer method:



Text

Description automatically generated

A={1,2,3,4}=> {3,2,7,4}

P={3} -> 3 3

C={7} -> 1 7

Send the child to the parent (use pipe!!)

Declare vector with 2 values->type int initially

Call function pipe->one arg->the vector in which we store the val of the vector

Pipe shared between the child and the parent->go to the child and send a[2] to the parent => write function used:

Reference &a[2] -> to get the value

Sizeof(int)

Operation of pipe->close() before using the write()

Parent:

Close()->right end

Read()->3 arg->read of the pipe, address of the value we want to store, how many bytes

Final sum a[0] and a[2]

Text

Description automatically generated

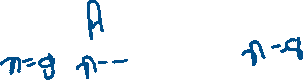
Close the pipes->when we do not use them

Text

Description automatically generated

Create 2 child processes -> one parent

Parent p creates A and B children -> 3 pipes



We have in P val (n) , decrement, sent to A, decrement, sent to B=> continue until one is 0

Solution:

Text

Description automatically generated

Run: Text

Description automatically generated

First write performed by the parent

Text

Description automatically generated

Work on the child now 😊

+ final read for the parent => have a cicle

Text

Description automatically generated

Text

Description automatically generated

We have to do this until we get to 0

Write a while in A -> reuse in the B, and P-> it will run until whatever, we do not know which process will kill the process

While(1) -> closing happening only at the end -> stop the condition of the while

If n==0 -> perform a break

If read(p2a[0],&n,sizeof(int)<0) -> perform a break

Take the while and replicate for all processes

#include <stdlib.h>  
#include <stdio.h>  
#include <unistd.h>  
#include <sys/wait.h>  
#include <sys/types.h>

int main(int argc, char\*\* argv)  
{  
        int pidA, pidB;  
        int n = 10;  
        int p2a[2];  
        pipe(p2a);  
        int a2b[2];  
        pipe(a2b);  
        int b2p[2];  
        pipe(b2p);

        pidA = fork();  
        if(pidA == 0)  
        {  
                close(a2b[0]);  
                close(b2p[0]);  
                close(b2p[1]);  
                close(p2a[1]);

                while(1)  
                {  
                        if(read(p2a[0], &n, sizeof(int)) < =0)  
                                break;

                        if(n == 0)  
                                break;

                        n--;  
                        printf("A: %d\n", n);  
                        write(a2b[1], &n, sizeof(int));  
                }  
                close(p2a[0]);  
                close(a2b[1]);  
                exit(0);  
        }

        pidB = fork();  
        if(pidB == 0)  
        {  
                close(p2a[0]);  
                close(p2a[1]);  
                close(b2p[0]);  
                close(a2b[1]);

                while(1)  
                {  
                        if(read(a2b[0], &n, sizeof(int)) < =0)  
                                break;

                        if(n == 0)  
                                break;

                        n--;  
                        printf("B: %d\n", n);  
                        write(b2p[1], &n, sizeof(int));  
                }  
                close(a2b[0]);  
                close(b2p[1]);  
                exit(0);  
        }

        printf("P: %d\n", n);

        close(a2b[0]);  
        close(a2b[1]);  
        close(b2p[1]);  
        close(p2a[0]);  
        write(p2a[1], &n, sizeof(int));

        while(1)  
        {  
                if(read(b2p[0], &n, sizeof(int)) < =0)  
                        break;

                if(n == 0)  
                        break;

                n--;  
                printf("P: %d\n", n);  
                write(p2a[1], &n, sizeof(int));  
        }  
        close(p2a[1]);  
        close(b2p[0]);  
        wait(0);  
        wait(0);  
}

Text

Description automatically generated

We have a pipe p2a->communicate from P to A

P2a[1]->3 processes ready to write in it

->used functionality

Read op on the p2a ->in process A

Reading from this pipe, A close down the right end of the pipe

Read will continue until we have writers

At this moment P is the only writer

P has a pipe b2a -> reads in the same manner as A

Same pipe=> break the code

Zero writers-> return 0 ->break condition from process A

->something like this on the exam -> process P to A

->section 6.7 pipes on the teaching notes

->next lab focused on fifos(!2 different processes ex: python and c)

->pipes stored in memory