Mangesh Raut (mbr63)

HW-6-Project-5-Basic-Processes

CS503-001

Part-1

Problem 8.11

It prints the 4 times the hello on the output. As you can see in program use for loop less than 2 and before that fork that’s why it prints 4 times hello. Fork makes two process which run same program after this instruction.

hello

hello

hello

hello

Problem 8.12

It prints 8 times the hello on the output. As first fork print 2 times second fork print 4 times then printf and printf so total is 8 times it will print the hello.

hello

hello

hello

hello

hello

hello

hello

hello

Problem 8.13

It prints x=4, x=3, x=2 as output on screen after running this program. Output maybe print in any sequence for me first sequence is 4,3,2. Second time it print 4,2,3.

Parent process

x=4

x=3

child process

x=2

Problem 8.14

It will print 3 times hello.

hello

hello

hello

Problem 8.16

Output of the counter is 2.

counter = 2

Part-2

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <sys/wait.h>

#include <unistd.h>

#include <fcntl.h>

#include <string.h>

void runchild(int); //child prototype takes int

void runparent(int); //parent prototype takes int

int

main(int argc, char \*argv[])

{

int chpid; // child process id number

if(argc != 2) { //it checks if argc is not 2 then print the usage and then exit

fprintf(stderr, "Usage: chat n\n"); //debug print msg usage

exit(1);

}

chpid = fork();

if(chpid < 0) { // if its negative then problem and exit

perror("fork");

exit(2);

}

if(chpid == 0) { //if child process id equal to zero

runchild(atoi(argv[1])); // run the child pass it to argcv 1

}

else {

runparent(atoi(argv[1])); // otherwise run the parent pass it to argcv 1

}

exit(0); //exit

}

void

runcommand(char \*buf) //it takes char buf

{

int i, status; //int i and status

char \*tokens[32]; //char tokens with 32 size

if(fork() > 0) { //fork is greater than 0

wait(&status); //wait for status

}

else {

tokens[0] = strtok(buf + 1, " \t\n"); //address of second char in buffer

for(i = 1; i < 32; i++) { //for loop from tokens 1 to 32

tokens[i] = strtok(NULL, " \t\n"); //tokens i get other calls with NULL has to be first argv

if(tokens[i] == NULL) //if tokens i is null break it

break;

}

execvp(tokens[0], tokens); //tokens 0 from tokens array basically argv

perror("exec"); //unable to run the program

exit(4); //exit

}

}

void

dooutredir(char \*buf, int fdw)

{

int i, status; //integer I and status

char \*tokens[32]; //list of tokens 32 tokens

if(fork() > 0) { //if fork is greater than 0

wait(&status); //wait for status

}

else { //else

tokens[0] = strtok(buf + 1, " \t\n"); //address of second char in buffer

for(i = 1; i < 32; i++) { //for loop fro tokens 1 to 32

tokens[i] = strtok(NULL, " \t\n"); //tokens i get other calls with NULL has to be first argv

if(tokens[i] == NULL) //if tokens i is null break it

break;

}

close(1); //close

dup(fdw); //dup write

execvp(tokens[0], tokens); //tokens 0 from tokens array basically argv

perror("exec"); //unable to run the program

exit(4); //exit

}

}

void

runparent(int which)

{

int fdw, n; //fdw is for write and n is number for how many times we read.

char buf[128]; //buffer of 128 char

if(which == 1) { //if it is 1 it is for writing

fdw = open("chatpipe1", O\_WRONLY); // it will open chatpipe1

}

else { //chatpipe 2 will open

fdw = open("chatpipe2", O\_WRONLY);

}

if(fdw < 0) { //if write is less than 0

perror("parent open"); //parent open

exit(3); //exit

}

while((n = read(0, buf, 127)) > 0) { //if it is greater than 0 it will out from loop.

buf[n] = '\0'; //no byte

switch(buf[0]) { //switch till no byte

case '!':

runcommand(buf); //runcommand buffer

break;

case '>':

dooutredir(buf, fdw); //dooutredir buffer and write

break;

default:

write(fdw, buf, n); //write write and n

break; //break

}

}

close(fdw); //close the pipe return and exit

}

void

runchild(int which)

{

int fdr, n; // pipe is for reading and n if for number

char buf[128]; //buffer of 128 char

if(which == 1) { //if it is 1 it is for reading

fdr = open("chatpipe2", O\_RDONLY); //chatpipe2 will open for reading

}

else {

fdr = open("chatpipe1", O\_RDONLY); //chatpipe1 will open for reading

}

if(fdr < 0) {

perror("child open");

exit(3);

}

while((n = read(fdr, buf, 128)) > 0) { //reading from fdr into 128 buffer

write(1, ">>> ", 4); //write 4 char

write(1, buf, n); //write n bytes there

}

close(fdr); //close and then return

}

Output of the program.

(base) mangeshraut@n3-89-82 Desktop % ./chat 1

Hello

>>> This is testing

cs503 chat program

>>> exit

(base) mangeshraut@n3-89-82 Desktop % ./chat 2

>>> Hello

This is testing

>>> cs503 chat program

exit