\*README  
\*ELIAS ENNEBT 19391679

\*Parent.c child1, child2.c, child3.c

Solution Logic:

NAME

Parent.c executes 3 child processes, child 1, child 2 and child 3

DESCRIPTION:

This File creates 3 child processes using a fork(), all 3 children are sibling and are initiated at the same time, the parent waits until all 3 processes are finished then returns exit statuses of each child process. The parent also checks before initialising the 3 child processes

Known limitations: was unable to

NOTES

Parent creates all 3 children and accepts two text files as inputs ( all 3 children are siblings so piping is not necessary)

Child1: prompts user to enter in 10 numbers

Child2: counts number of words in afile.txt using wc and exec()

Child3: opens afile.txt and copies and replaces the words “run” and “study”

Exit status printed for 3 children when completed

Limitations:

As all children run at the same time the output can be abit messy, can only be fixed by using wait()

Buffersize limited

Compiling:

To compile and run use unix systems, cannot use windows enter the following commands

cc parent.c

./a.out afile.txt empty.txt

//PSEUDOCODE

Parent.c

IF (number of arguments passed in is 2 )

Print( file 1 and file 2)

ELSE:

return EXIT FAILURE

------------------

IF (EXIT STATUS FOR ALL 3 CHILDREN ARE 0)

PRINT “ EXIT STATUS FOR CHILD1, CHILD2 AND CHILD3 EXIT STATUS”

ELSE

PRINT “ERROR AND EXIT STATUS”

ENDIF

Child1

Create child process pid1:

If failed

Print error to stderr

Else:

Print ( pid and parent pid)

Execute Child1() function:

FOR 1 – 10:

“Ask user for input 10 numbers “

Sum += number

If n > biggestnumber:

Biggestnumber = number;

If n < smallestnumber:

Smallestnumber = number;

Averagenum = Sum/10.0

Print(SUM and AVG);

Print(Biggest number and Smallest Number);

ENDIF

EXIT(SUCCESSFUL)

-------------------------------------------------------------------------------------------------------

Prompts user for an input(stdin)

User enters in 10 numbers

The highest, lowest and average grade of the 10 numbers are calculated

Print results to stdin(terminal)

-------------------------------------------------------------------------------------------------------

Child2

Create child process pid2:

If failed

Print error to stderr

exit

Else:

Execute Child2() function:

Read through afile.txt and get a word count using “wc” unix command

Print the word count of afile.txt

ENDIF

EXIT(SUCCESSFUL)

---------------------------------------------------------------------------------------------------------------------------------

Reads from the afile.txt

Using exec() read through afile.txt and count for number of words

Print results to stdin (terminal)

----------------------------------------------------------------------------------------------------------------------------------

Child3

Create child process pid2:

If failed

Print error to stderr

exit

Else:

Open( afile.txt as read only)

Opne(empty.txt as write file)

Write( “UPDATE VERSION OF FILE” in empty.txt)

WHILE LOOP (through afile.txt):

Replace all instances of ( “execute” to “run”)

Replace all instances of (“study” to “examine”)

Move to next word

END WHILE

Print to stdin( CONTENST COPIED AND UPDATED to empty.txt)

Close file1

Close file2

Exit(status 1)

---------------------------------------------------------------------------------------------------------

Open afile.txt and open empty.txt

Write “updated version of File” to empty.txt

Copy afile.txt -> empty.txt

Read through empty.txt and update every phrase “execute” to “run” and “study” to “examine”

Close empty.txt and afile.txt

--------------------------------------------------------------------------------------------------------------------------------

TEST PLAN:

1. Test multiple files
2. Test each child alone
3. Compile from parent.c file
4. Test if each file works independently
5. Check for errors