Assignments

Assignment 2 - In progress

Honor Pledge Accepted
Draft - In progress
Submitted
Returned

Assignment Details

Title
Assignment 2
Due
Feb 14, 2023 11:55 PM
Number of resubmissions allowed
Unlimited
Accept Resubmission Until
Feb 16, 2023 11:55 PM
Status
Honor Pledge Accepted
Grade Scale
Points (max 100.00)

Instructions

Assignment overview

Large integer multiplication can be achieved by decomposing the two factors and performing multiplication and addition on the simplified terms. For example,

`1234*5678`

can also be written as

$[\ (12*56) * 10^4\ ] + [\ (12*78 + 34*56) * 10^2\ ] + [\ (34*78) * 10^0\ ]$

More generally,

1. assume n = length(a) = length(b), (can pad 0's for shorter number)
2. if length(a) <= 1 then return a * b
3. Partition a,b into a = a1 * $10^{n/2}$ + a2 and b = b1 * $10^{n/2}$ + b2
4. A = Multiply(a1, b1) 672
5. B = Multiply(a2, b1) 936
6. C = Multiply(a1, b2) 1904
7. D = Multiply(a2, b2) 2652
8. X = A * 10^n 6720000
9. Y = (B + C) * $10^{n/2}$ 284000
10. Z = D * 10^0 2652
11. Return X + Y + Z 7006652

For this assignment, write a C program which will:

1. Accept two 4-digit command-line parameters. (For simplicity, we will not worry about step 1 and 2 above - just deal with integers 1000-9999)
2. Partition the two numbers in to a1, a2, b1, b2
3. Establish a pipe
4. Fork a child process
    1. The parent will send a1 and b1 to the child through a pipe
    2. The child will multiply a1 and b1 and send A to the parent through a pipe. The parent will calculate X
    3. The parent will send a2 and b1 to the child through a pipe
    4. The child will multiply a2 and b1 and send B to the parent through a pipe
    5. The parent will send a1 and b2 to the child through a pipe

6. The child will multiply a1 and b2 and send C to the parent through a pipe. The parent will calculate Y
7. The parent will send a2 and b2 to the child through a pipe
8. The child will multiply a2 and b2 and send D to the parent through a pipe. The parent will calculate Z
9. (The child will exit)

5. The program will then calculate X + Y + Z and print the sum to the screen

Purpose

The goals of this assignment are the following:

- Learn how to use pipe for bi-directional communication between parent and child process.
- Get experience with the `pipe()` system function
- Gain more experience with the C programming language from an OS perspective

Computing platform

You are welcome to develop your program on your own workstation if you wish, but you are responsible for ensuring that your program compiles and runs without error on the Gaul computing platform. Marks will be deducted if your program fails to compile, or your program runs into errors on Gaul.

- https://wiki.sci.uwo.ca/sts/computer-science/gaul

Instructions

Attached to this assignment is a tarball with the following files in it. **None of these files should be modified**:

```
Makefile          <--- A pre-packaged Makefile. This tells you how your program should be structured
run-assignment.sh <--- A shell script that will automatically run your program
```

Download this tarball and upload it to Gaul. Extract the tarball (`tar -xvf Assignment-2.tar`). Change to the `Assignment-2` directory.

You will write a program called `assignment-2.c`. This program:

- Accepts two command-line parameters which are both 4-digit integers
- Prints out the communication between parent and child with identifying process ids
- Prints the product of the two integers using the method described above

Output

Executing `./assignment-2 1234 5678` should produce the following output:

```
Your integers are 1234 5678
Parent (PID 26580): created child (PID 26581)

###
# Calculating X
###
Parent (PID 26580): Sending 12 to child
Parent (PID 26580): Sending 56 to child
        Child (PID 26581): Received 12 from parent
        Child (PID 26581): Received 56 from parent
        Child (PID 26581): Sending 672 to parent
Parent (PID 26580): Received 672 from child

###
# Calculating Y
###
Parent (PID 26580): Sending 12 to child
Parent (PID 26580): Sending 78 to child
        Child (PID 26581): Received 12 from parent
        Child (PID 26581): Received 78 from parent
        Child (PID 26581): Sending 936 to parent
Parent (PID 26580): Received 936 from child
```

```
Parent (PID 26580): Sending 34 to child
Parent (PID 26580): Sending 56 to child
        Child (PID 26581): Received 34 from parent
        Child (PID 26581): Received 56 from parent
        Child (PID 26581): Sending 1904 to parent
Parent (PID 26580): Received 1904 from child

###
# Calculating Z
###
Parent (PID 26580): Sending 34 to child
Parent (PID 26580): Sending 78 to child
        Child (PID 26581): Received 34 from parent
        Child (PID 26581): Received 78 from parent
        Child (PID 26581): Sending 2652 to parent
Parent (PID 26580): Received 2652 from child

1234*5678 == 6720000 + 284000 + 2652 == 7006652
```

and `./run-assignment.sh 2` should produce the following output:

```
ASSIGNMENT 2 STARTED - Dow Mon  ## ##:##:## AM/PM EST 2023

Cleaning environment
----------------------------
rm -f assignment-2

Checking environment
----------------------------
846ec3b283736bf8a0c1d7065e6d8de3  ./run-assignment.sh
***********UNIQUE ID************  assignment-1.c
Makefile: OK

Building environment
----------------------------
make all
make[1]: Entering directory '/home/wbeldman/3305/Projects/Assignment 2/Assignment-2'
gcc -o assignment-2 assignment-2.c -Wall -Wpedantic -Wextra -std=gnu17
make[1]: Leaving directory '/home/wbeldman/3305/Projects/Assignment 2/Assignment-2'

Assignment 2
----------------------------
Proper usage is ./assignment-2 <4 digit integer> <4 digit integer>
Proper usage is ./assignment-2 <4 digit integer> <4 digit integer>}
... Output as above for 1234*5678 ...
... Output as above for 9999*9999 ...
... Output as above for 1000*1000 ...

Cleaning environment
----------------------------
rm -f assignment-2

ASSIGNMENT 2 COMPLETED - Dow Mon ## ##:##:## AM/PM EST 2023
```

Helpful hints

- Since command-line parameters are read in as strings, you may want to take advantage of the `atoi()` function which will convert any string that looks like an integer into an integer. For example `atoi("1234")` will return the integer `1234`.
- Decomposing a 4 digit number can be done as follows:

```
int i = 1234;
int a = i%100;
int b = i/100;
```

Submitting

When you are finished your assignment, follow these steps

1. From inside the Assignment-2 directory, run the following command: `script -c './run-assignment.sh 2' assignment-2.out`
   Your directory should now contain the following files:

```
assignment-2.c     <--- Your program
assignment-2.out   <--- The output produced by running the script command above
Makefile           <--- A pre-packaged Makefile. This tells you how your program should be structured
run-assignment.sh <--- A shell script that will automatically run your program and put the results in assignment-2.out
```

2. Assuming the command was successful, run the follow command to get out of the `Assignment-2` directory: `cd ..`
3. Package your assignment into a tarball: `tar -cvf Assignment-2.tar Assignment-2`
4. Verify the contents of your tarball (`tar -tvf Assignment-2.tar`) (`du -sh Assignment-2.tar`). **If your tarball is 10kb in size** you have an empty tarball and you made an error on this step. Make sure you are properly creating your tarball with the right files in it.
5. Use an SFTP program to download the tarball and then upload it to OWL.

Additional resources for assignment

- Assignment-2.tar (10 KB; Jan 6, 2023 4:51 pm)

Grading Rubric

Preview Rubric

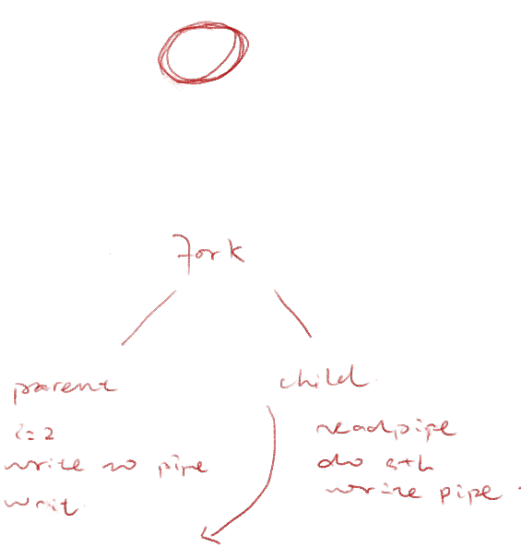
Submission

Attachments

No attachments yet

Select a file from computer Choose File No file chosen

Proceed | Preview | Save Draft | Cancel | Don't forget to save or proceed!


fork
parent
i=2
write to pipe
wait
child
readpipe
do sth
write pipe