

Assignment 3. IPC-Pipe and Multithreads

Due at midnight February 16, 2018

You have to deliver one program for each problem. Please provide a make file and README file to explain how to compile and run your programs.

1. Design a file-copying program named FileCopy using ordinary pipes. This program will be passed two parameters: the first is the name of the file to be copied, and the second is the name of the copied file. The program will then create an ordinary pipe and write the contents of the file to be copied to the pipe. The child process will read this file from the pipe and write it to the destination file. For example, if we invoke the program as follows: FileCopy input.txt copy.txt the file input.txt will be written to the pipe. The child process will read the contents of this file and write it to the destination file copy.txt.

2. Matrix Multiplication Project. Given two matrices, A and B, where matrix A contains M rows and K columns and Matrix B contains K rows and N columns, the matrix product of A and B is matrix C contains M rows and N columns. The entry in matrix C for row i column j ($C_{i,j}$) is the sum of the products of the elements for the row i in matrix A and column j in matrix B. That is

$$C_{i,j} = \sum_{n=1}^K A_{i,n} * B_{n,j}$$

For example, if A is a 3-by-2 matrix and B is a 2-by-3 matrix, element $C_{3,1}$ is the sum of $A_{3,1} \times B_{1,1}$ and $A_{3,2} \times B_{2,1}$.

For this project calculate each element $C_{i,j}$ in a separate worker thread. This will involve creating $M \times N$ worker threads. The main ----or parent ----thread will initialize the matrices A and B. These matrices will be declared as global data so that each worker thread has access to A, B, and C.

Matrices A and B can be initialized statically, as shown below:

```
# define M 3

# define K 2

# define N 3

int A [M][K] ={{1,4},{2,5},{3,6}};

int B [K][N] ={{8,7,6},{5,4,3}}

int C [M][N];
```

Passing Parameters to Each Thread The parent thread will create $M \times N$ worker threads, passing each worker the values of row i and column j that it is to use in calculating the matrix product. This requires passing two parameters to each thread. The easiest approach with Pthreads is to create a data structure using a struct. The members of this structure are i and j , and the structure appears as follows:

```
/* structure for passing data to threads */  
  
struct v  
{  
    int i; /* row */  
    int j; /* column */  
};
```

Pthreads programs will create the worker threads using a strategy similar to that shown below.

```
/* We have to create M * N worker threads */  
  
for (i = 0; i < M; i++) {  
    for (j = 0; j < N; j++){  
        struct v *data = (struct v *) malloc (sizeof(struct v));  
  
        data->i = i;  
        data->j = j;  
  
        /* Now create the thread passing it data as a parameter */  
    }  
}
```

The data pointer will be passed to the `pthread_create()` function, which in turn will pass it as a parameter to the function that is to run as a separate thread.

Waiting for Threads to Complete

Once all worker threads have completed, the main thread will output the product contained in matrix C . This requires the main thread to wait for all worker threads to finish before it can output the value of the matrix product.

A simple strategy for waiting on several threads using the Pthreads `pthread_join()` is to enclose the join operation within a simple for loop. For example, you could join on ten threads using the Pthread code below:

```
for (i = 0; i<10 ; i++)
```

```
    pthread_join(workers[i], NULL);
```