CS 180A Assignment 4 (Threads) - Ryman Barnett

Generated by Doxygen 1.8.17

1 Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

matrix_data
Struct containing the data for the thread **2**

2 File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

main-thread.c
Source file for main-matrix functions to create a matrix from a file, print a matrix, and run a matrix. into child threads that calculate matrix squared **4**

3 Class Documentation

3.1 matrix_data Struct Reference

struct containing the data for the thread

```
#include <matrix-thread.h>
```

Public Attributes

- int * input
- int * output
- int width
- int row
- int col
- int pos

3.1.1 Detailed Description

struct containing the data for the thread

Definition at line 20 of file matrix-thread.h.

3.1.2 Member Data Documentation

3.1.2.1 col `int matrix_data::col`

column of the position

Definition at line 26 of file matrix-thread.h.

Referenced by main(), and matrix_thread().

3.1.2.2 input `int* matrix_data::input`

pointer to the input matrix

Definition at line 22 of file matrix-thread.h.

Referenced by main(), and matrix_thread().

3.1.2.3 output `int* matrix_data::output`

pointer to the output matrix

Definition at line 23 of file matrix-thread.h.

Referenced by main(), and matrix_thread().

3.1.2.4 pos `int matrix_data::pos`

position in the matrix

Definition at line 27 of file matrix-thread.h.

Referenced by main(), and matrix_thread().

3.1.2.5 row `int matrix_data::row`

row of the position

Definition at line 25 of file matrix-thread.h.

Referenced by main(), and matrix_thread().

3.1.2.6 width `int matrix_data::width`

width of the matrix

Definition at line 24 of file matrix-thread.h.

Referenced by main(), and matrix_thread().

The documentation for this struct was generated from the following file:

- matrix-thread.h

4 File Documentation

4.1 main-thread.c File Reference

Source file for main-matrix functions to create a matrix from a file, print a matrix, and run a matrix. into child threads that calculate matrix squared.

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include "matrix-thread.h"
```

Functions

- int ∗ get_matrix (const char ∗filename, int ∗width)

  creates a matrix from a file
- void print_matrix (int ∗matrix, int width)

  prints a matrix
- int main (int argc, char ∗∗argv)

  handles creating a matrix, setting up buffers, and creating a thread for each row of the matrix to calculate the square of the matrix

4.1.1 Detailed Description

Source file for main-matrix functions to create a matrix from a file, print a matrix, and run a matrix. into child threads that calculate matrix squared.

Author

Ryman Barnett

email: ryman.b@digipen.edu

DigiPen login: ryman.b

Course: CS180

Section: A

Assignment #3

Date

2022-10-14

4.1.2 Function Documentation

4.1.2.1 get_matrix()

```
int* get_matrix (
    const char * filename,
    int * width )
```

creates a matrix from a file

Parameters

filename	name of file to read from
width	width of the matrix

Returns

pointer to the matrix

Definition at line 34 of file main-thread.c.

```
{
  int value, *matrix, result;
  FILE *fp;

  /* Open the file */
  fp = fopen(filename, "rt");
  if (!fp)
  {
    printf("Can't open file: %s\n", filename);
    exit(-1);
  }

  /* Read the width */
  result = fscanf(fp, "%d", width);
  if (result == -1)
  {
    printf("Can't read from file: %s\n", filename);
    fclose(fp);
    exit(-1);
  }

  /* Allocate the matrix */
  matrix = malloc(*width * *width * sizeof(int));
  if (!matrix)
  {
    printf("Can't malloc matrix\n");
    fclose(fp);
    exit (-2);
  }

  /* Read the matrix */
  while (!feof(fp))
  {
    result = fscanf(fp, "%d", &value);
    if (result == -1)
      break;
    *matrix++ = value;
  }
  fclose(fp); /* close the file */
  return matrix - (*width * *width); /* return the matrix */
}
```

Referenced by main().

4.1.2.2 main()

```
int main (
    int argc,
    char ** argv )
```

handles creating a matrix, setting up buffers, and creating a thread for each row of the matrix to calculate the square of the matrix

Parameters

argc	number of arguments
argv	array of arguments

Returns

0 on success, -1 error code on failure

Definition at line 114 of file main-thread.c.

```c
{
  int i;
  int width;                  /* width of the matrix           */
  int matsize;                /* total matrix values           */
  int *input_matrix;          /* the matrix read in            */
  int *result_matrix;         /* threads will put results here */
  pthread_t* threads = NULL; /* array of threads */

  /* check for correct number of arguments */
  if (argc < 2)
  {
    printf("Insufficient parameters supplied\n");
    return -1;
  }

  /* Reading the input matrix from a file into it's memory. */
  input_matrix = get_matrix(argv[1], &width);

  /* calculate the size of the matrix */
  matsize = width * width;

  /* Allocating memory for the thread container. */
  threads = malloc(sizeof(pthread_t) * matsize);
  if (!threads)
  {
    printf("Can't malloc threads\n");
    free(input_matrix);
    exit(-1);
  }

  /* Allocating memory for the result matrix. */
  result_matrix = malloc(matsize * sizeof(int));
  if (!result_matrix)
  {
    printf("Can't malloc result matrix\n");
    free(input_matrix);
    free(threads);
    exit(-1);
  }

    /* Printing the input matrix. */
  print_matrix(input_matrix, width);

    /* Creating all of the other threads and supplying them with */
    /* their parameters                                          */
  for (i = 0; i < matsize; i++)
  {
    /* create the data struct for the thread */
    matrix_data *ds = malloc(sizeof(matrix_data));
    ds->input = input_matrix;
    ds->output = result_matrix;
    ds->width = width;
    ds->row = i / width;
    ds->col = i % width;
    ds->pos = i;

    /* create the thread */
    pthread_create(&threads[i], NULL, matrix_thread, (void *)ds);
  }

    /* Waiting for all of the threads to finish. */
  for (i = 0; i < matsize; i++)
    pthread_join(threads[i], NULL);

    /* Printing the resulting squared matrix. */
  print_matrix(result_matrix, width);

    /* Cleaning up any memory or resources the main thread created. */
  free(input_matrix);
  free(result_matrix);
  free(threads);

  return 0;
}
```

References matrix_data::col, get_matrix(), matrix_data::input, matrix_thread(), matrix_data::output, matrix_data::pos, print_matrix(), matrix_data::row, and matrix_data::width.

4.1.2.3 print_matrix() `void print_matrix ( int * matrix, int width )`

prints a matrix

Parameters

matrix	pointer to the matrix
width	width of the matrix

Definition at line 87 of file main-thread.c.

```
{
  int i, size = width * width;
  for (i = 0; i < size; i++)
  {
    printf("%8i", matrix[i]);
    if ( (i + 1) % width == 0)
      printf("\n");
  }
  printf("\n");
}
```

Referenced by main().

4.2 matrix-thread.c File Reference

Source file for matrix_thread function to calculate the value of a single position in a matrix squared.

```
#include <stdlib.h>
#include "matrix-thread.h"
```

Functions

- void ∗ matrix_thread (void ∗data)
  calculates the value of a single position in a matrix squared

4.2.1 Detailed Description

Source file for matrix_thread function to calculate the value of a single position in a matrix squared.

Author

Ryman Barnett

email: ryman.b@digipen.edu

DigiPen login: ryman.b

Course: CS180

Section: A

Assignment #3

Date

2022-10-14

4.2.2 Function Documentation

4.2.2.1 matrix_thread()

```
void* matrix_thread (
        void * data )
```

calculates the value of a single position in a matrix squared

Parameters

data	struct containing the data for the thread

Returns

null

Definition at line 28 of file matrix-thread.c.

```
{
  matrix_data *ds = (matrix_data *)data; /* cast the data to the struct */
  int sum = 0;                           /* sum of the row              */
  int i;                                 /* loop index                  */

  /* calculate the sum of the row */
  for (i = 0; i < ds->width; i++)
  {

    sum += (ds->input)[(ds->row * ds->width) + i] * (ds->input)[ds->col + (i * ds->width)];
  }

  (ds->output)[ds->pos] = sum; /* set position in mem to answer */

  /* clean up */
  free(ds);

  return NULL;
}
```

References matrix_data::col, matrix_data::input, matrix_data::output, matrix_data::pos, matrix_data::row, and matrix↩_data::width.

Referenced by main().

4.3 matrix-thread.h File Reference

header file for matrix_thread function to calculate the value of a single position in a matrix squared and data struct for the thread.

Classes

- struct matrix_data

  struct containing the data for the thread

Functions

- void ∗ matrix_thread (void ∗data)

  calculates the value of a single position in a matrix squared

4.3.1 Detailed Description

header file for matrix_thread function to calculate the value of a single position in a matrix squared and data struct for the thread.

Author

Ryman Barnett

email: ryman.b@digipen.edu

DigiPen login: ryman.b

Course: CS180

Section: A

Assignment #3

Date

2022-10-14

4.3.2 Function Documentation

4.3.2.1 matrix_thread()

```
void* matrix_thread (
        void * data )
```

calculates the value of a single position in a matrix squared

Parameters

data	struct containing the data for the thread

Returns

null

Definition at line 28 of file matrix-thread.c.

```
{
  matrix_data *ds = (matrix_data *)data; /* cast the data to the struct */
  int sum = 0;                           /* sum of the row              */
  int i;                                 /* loop index                  */

  /* calculate the sum of the row */
  for (i = 0; i < ds->width; i++)
  {

    sum += (ds->input)[(ds->row * ds->width) + i] * (ds->input)[ds->col + (i * ds->width)];
  }

  (ds->output)[ds->pos] = sum; /* set position in mem to answer */

  /* clean up */
  free(ds);

  return NULL;
}
```

References matrix_data::col, matrix_data::input, matrix_data::output, matrix_data::pos, matrix_data::row, and matrix↵_data::width.

Referenced by main().

Index