```c
/* Demo by Chris Cotter */

typedef float mdata_t;
struct Matrix
{
    int nrows, ncols;
    mdata_t *data;
};

static int matrix_initialize(struct Matrix *matrix, int nrows, int ncols)
{
    mdata_t *data;
    data = malloc(sizeof(mdata_t) * nrows * ncols);
    if (!data)
        return -ENOMEM;
    matrix->data = data;
    matrix->nrows = nrows;
    matrix->ncols = ncols;
    return 0;
}

static void matrix_free(struct Matrix *matrix)
{
    free(matrix->data);
    matrix->data = NULL;
    matrix->nrows = matrix->ncols = -1;
}

#define matrix_val(matrix, i, j) \
    (((matrix)->data)[(matrix)->ncols * (i-1) + j-1])
#define matrix_nrows(matrix) ((matrix)->nrows)
#define matrix_ncols(matrix) ((matrix)->ncols)

static int matrix_copy(struct Matrix *dst, struct Matrix *src)
{
    int nrows, ncols;
    nrows = matrix_nrows(src);
    ncols = matrix_ncols(src);
    if (matrix_initialize(dst, nrows, ncols))
        return -1;
    memcpy(dst->data, src->data, sizeof(mdata_t) * ncols * nrows);
    dst->nrows = nrows;
    src->ncols = ncols;
    return 0;
}

static int matrix_compare(struct Matrix *m1, struct Matrix *m2)
{
    int nrows, ncols;
    nrows = matrix_nrows(m1);
    ncols = matrix_ncols(m1);
    if (matrix_nrows(m2) != nrows || matrix_ncols(m2) != ncols)
        return -1;
    return memcmp(m1->data, m2->data, sizeof(mdata_t) * ncols * nrows);
}

static void matrix_print(struct Matrix *matrix)
{
    int nrows, ncols;
    int i, j;
    nrows = matrix_nrows(matrix);
    ncols = matrix_ncols(matrix);
    for (i = 1; i <= nrows; ++i)
    {
        for (j = 1; j <= ncols; ++j)
        {
            printf("%10.3f", matrix_val(matrix, i, j));
        }
        printf("\n");
    }
}

```



```c

#include <assert.h>
#include <time.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <errno.h>

#include <inc/fork_nondet.h>
#include <inc/determinism.h>

#include "/root/matrix.h"

/*
 * Gaussian elimination
 * http://engineering.ucsb.edu/~hpscicom/projects/gauss/introge.pdf
 */
#define NTHREADS 10
struct reduce_thread_data
{
    struct Matrix *matrix;
    int k, i;
};
static pthread_t reduce_threads[NTHREADS];
static struct reduce_thread_data reduce_thread_data[NTHREADS];
void *reduce_worker(void *_data)
{
    int i, j, k, nrows;
    struct reduce_thread_data *data;
    struct Matrix *matrix;
    data = _data;
    matrix = data->matrix;
    k = data->k;
    i = data->i;
    nrows = matrix_nrows(matrix);
    matrix_val(matrix, i, k) =
        matrix_val(matrix, i, k) / matrix_val(matrix, k, k);
    for (j = k + 1; j <= nrows + 1; ++j)
    {
        matrix_val(matrix, i, j) =
            matrix_val(matrix, i, j) -
            matrix_val(matrix, i, k) * matrix_val(matrix, k, j);
    }
    matrix_val(matrix, i, k) = 0;
    return NULL;
}
int parallel_reduce(struct Matrix *matrix)
{
    int nrows, ncols;
    int i, k;
    nrows = matrix_nrows(matrix);
    ncols = matrix_ncols(matrix);
    for (k = 1; k <= nrows - 1; ++k)
    {
        for (i = k + 1; i <= nrows; ++i)
        {
            reduce_thread_data[i].matrix = matrix;
            reduce_thread_data[i].k = k;
            reduce_thread_data[i].i = i;
            /* Create and start threads. */
            if (!dfork(i, DETERMINE_SNAP | DETERMINE_START))
            {
                reduce_worker(&reduce_thread_data[i]);
                dret();
            }
        }
        /* Wait for threads to finish, merge their changes into the parent. */
        for (i = k + 1; i < /* bug here - should be <= */ nrows; ++i)
            dget(i, DETERMINE_MERGE, matrix->data, sizeof(mdata_t) * ncols *
                    nrows, NULL);
    }
    return 0;
}
```

```
74
75  static float M1[4][5] = {
76      {3, 0, 6, -3, 1},
77      {0, 2, 3, 0, 1},
78      {-4, -7, 2, 0, 1},
79      {2, 0, 1, 10, 1}
80  };
81  static void use1(struct Matrix *matrix)
82  {
83      int i, j;
84      matrix_initialize(matrix, 4, 5);
85      for (i = 1; i <= 4; ++i)
86          for (j = 1; j <= 5; ++j)
87              matrix_val(matrix, i, j) = M1[i-1][j-1];
88  }
89  int main(void)
90  {
91      struct Matrix matrix, pmatrix;
92      become_deterministic();
93      use1(&pmatrix);
94      parallel_reduce(&pmatrix);
95      matrix_copy(&matrix, &pmatrix);
96      matrix_print(&matrix);
97      matrix_free(&matrix);
98      matrix_free(&pmatrix);
99      return 0;
100 }
101
```



```
1
2   #include <assert.h>
3   #include <time.h>
4   #include <stdio.h>
5   #include <string.h>
6   #include <pthread.h>
7   #include <stdlib.h>
8   #include <errno.h>
9
10  #include <matrix.h>
11
12  /*
13   * Gaussian elimination
14   * http://engineering.ucsb.edu/~hpscicom/projects/gauss/introge.pdf
15   */
16
17  #define NTHREADS 10
18  struct reduce_thread_data
19  {
20      struct Matrix *matrix;
21      int k, i;
22  };
23  static pthread_t reduce_threads[NTHREADS];
24  static struct reduce_thread_data reduce_thread_data[NTHREADS];
25  void *reduce_worker(void *_data)
26  {
27      int i, j, k, nrows;
28      struct reduce_thread_data *data;
29      struct Matrix *matrix;
30      data = _data;
31      matrix = data->matrix;
32      k = data->k;
33      i = data->i;
34      nrows = matrix_nrows(matrix);
35      matrix_val(matrix, i, k) =
36          matrix_val(matrix, i, k) / matrix_val(matrix, k, k);
37      for (j = k + 1; j <= nrows + 1; ++j)
38      {
39          matrix_val(matrix, i, j) =
40              matrix_val(matrix, i, j) -
41              matrix_val(matrix, i, k) * matrix_val(matrix, k, j);
42      }
43      matrix_val(matrix, i, k) = 0;
44      return NULL;
45  }
46  int parallel_reduce(struct Matrix *matrix)
47  {
48      int nrows;
49      int i, k;
50      nrows = matrix_nrows(matrix);
51      for (k = 1; k <= nrows - 1; ++k)
52      {
53          for (i = k + 1; i <= nrows; ++i)
54          {
55              reduce_thread_data[i].matrix = matrix;
56              reduce_thread_data[i].k = k;
57              reduce_thread_data[i].i = i;
58              pthread_create(&reduce_threads[i],
59                      NULL, reduce_worker, &reduce_thread_data[i]);
60          }
61          for (i = k + 1; i < /* bug here - should be <= */ nrows; ++i)
62              pthread_join(reduce_threads[i], NULL);
63      }
64      return 0;
65  }
66
67  static float M1[4][5] = {
68      {3, 0, 6, -3, 1},
69      {0, 2, 3, 0, 1},
70      {-4, -7, 2, 0, 1},
71      {2, 0, 1, 10, 1}
72  };
73  static void use1(struct Matrix *matrix)
```

```
74  {
75      int i, j;
76      matrix_initialize(matrix, 4, 5);
77      for (i = 1; i <= 4; ++i)
78          for (j = 1; j <= 5; ++j)
79              matrix_val(matrix, i, j) = M1[i-1][j-1];
80  }
81  int main(void)
82  {
83      struct Matrix pmatrix, matrix;
84      use1(&pmatrix);
85      parallel_reduce(&pmatrix);
86      matrix_copy(&matrix, &pmatrix);
87      matrix_print(&matrix);
88      matrix_free(&pmatrix);
89      matrix_free(&matrix);
90      return 0;
91  }
92
```



```
1   --- gauss_nondet.c      2012-04-19 11:50:41.000000000 -0500
2   +++ gauss_det.c 2012-04-19 11:50:41.000000000 -0500
3   @@ -3,17 +3,18 @@
4    #include <time.h>
5    #include <stdio.h>
6    #include <string.h>
7   -#include <pthread.h>
8    #include <stdlib.h>
9    #include <errno.h>
10
11  -#include <matrix.h>
12  +#include <inc/fork_nondet.h>
13  +#include <inc/determinism.h>
14  +
15  +#include "/root/matrix.h"
16
17   /*
18    * Gaussian elimination
19    * http://engineering.ucsb.edu/~hpscicom/projects/gauss/introge.pdf
20    */
21  -
22   #define NTHREADS 10
23   struct reduce_thread_data
24   {
25  @@ -45,9 +46,10 @@
26   }
27   int parallel_reduce(struct Matrix *matrix)
28   {
29  -    int nrows;
30  +    int nrows, ncols;
31       int i, k;
32       nrows = matrix_nrows(matrix);
33  +    ncols = matrix_ncols(matrix);
34       for (k = 1; k <= nrows - 1; ++k)
35       {
36           for (i = k + 1; i <= nrows; ++i)
37  @@ -55,11 +57,17 @@
38               reduce_thread_data[i].matrix = matrix;
39               reduce_thread_data[i].k = k;
40               reduce_thread_data[i].i = i;
41  -            pthread_create(&reduce_threads[i],
42  -                    NULL, reduce_worker, &reduce_thread_data[i]);
43  +            /* Create and start threads. */
44  +            if (!dfork(i, DETERMINE_SNAP | DETERMINE_START))
45  +            {
46  +                reduce_worker(&reduce_thread_data[i]);
47  +                dret();
48  +            }
49           }
50  +        /* Wait for threads to finish, merge their changes into the parent. */
51           for (i = k + 1; i < /* bug here - should be <= */ nrows; ++i)
52  -            pthread_join(reduce_threads[i], NULL);
53  +            dget(i, DETERMINE_MERGE, matrix->data, sizeof(mdata_t) * ncols *
54  +                    nrows, NULL);
55       }
56       return 0;
57   }
58  @@ -80,13 +88,14 @@
59    }
60    int main(void)
61    {
62  -    struct Matrix pmatrix, matrix;
63  +    struct Matrix matrix, pmatrix;
64  +    become_deterministic();
65       use1(&pmatrix);
66       parallel_reduce(&pmatrix);
67       matrix_copy(&matrix, &pmatrix);
68       matrix_print(&matrix);
69  -    matrix_free(&pmatrix);
70       matrix_free(&matrix);
71  +    matrix_free(&pmatrix);
72       return 0;
73   }
```

74