```
1  static void * thread_func (void* arg)
2  {
3    input_data *input = (input_data *)arg;
4    for (unsigned int i = input->row_start; i < input
5           multiply_row_by_matrix(a, i, b, res);
6    }
7
8    return (void *)0;
9  }
10
11
12 void threads_create()
13 {
14   pthread_attr_t attr;
15   int status;
16
17   in = (input_data *) malloc (sizeof(input_data) *
18   threads = (pthread_t *) malloc (sizeof(pthread_t)
19
20   status = pthread_attr_init(&attr);
21   if (status != 0)
22     exit(1);
23
24   int chunk_size = dim / thread_count;
25   int count = 0;
26
27   for (unsigned int thread_ind = 0; thread_ind < th
28     int chunk = chunk_size + extra;
29     in[thread_ind].row_start = count;
30     in[thread_ind].row_end = count + chunk;
31     count += chunk;
32
33     status = pthread_create(&threads[thread_ind], &
34     if (status != 0)
35       exit(1);
36
37   }
38 }
```

```
->row_end; i++) {
```

```
thread_count);
 * thread_count);
```

```
read_count; thread_ind++) {
```

```
attr, &thread_func, &in[thread_ind]);
```