Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №3 по курсу «Операционные системы»

Управление потоками в ОС

Студент: О. В. Бабин
Преподаватель: А. А. Соколов
Группа: М8О-206Б-19
Дата: 25.12.2020
Оценка:
Подпись:

Москва, 2020

1 Постановка задачи

Цель работы:

Приобретение практических навыков в:

- Управление потоками в ОС
- Обеспечение синхронизации между потокам

Задание (вариант 10):

Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При обработки использовать стандартные средства создания потоков операционной системы (Windows/Unix). Ограничение потоков должно быть задано ключом запуска вашей программы. Так же необходимо уметь продемонстрировать количество потоков, используемое вашей программой с помощью стандартных средств операционной системы. В отчете привести исследование зависимости ускорения и эффективности алгоритма от входящих данных и количества потоков. Получившиеся результаты необходимо объяснить.

Наложить K раз медианный фильтр на матрицу, состоящую из целых чисел. Размер окна задается.

2 Общие сведения о программе

Для преобразования входного изображения в RGB-массив используется сторонняя библиотека stb_image в виде двух заголовочных файлов. На вход программе подаются имена входного и выходного изображений, количество раз, которое надо наложить фильтр, размер «окна» и, опционально, количество потоков (по умолчанию 4). Программа делит изображение горизонтально на количество частей, равное количестве потоков, и обрабатывает эти части параллельно. На выходе получаем сглаженное изображение.

3 Общий метод и алгоритм решения

Для реализации поставленной задачи необходимо:

1. Изучить принципы работы pthread.
2. Изучить работу с библиотекой stb_image.
3. Написать обработку аргументов запуска программы.
4. Написать функцию разбиения изображения на части.
5. Написать функцию запуска многопоточной обработки.
6. Написать функцию медианного фильтра.
7. Написать обработку ошибок
8. Написать тесты

4 Исходный код

Код библиотеки stb_image приводить не буду, так как это не относится к заданию лабораторной.

lab3.c

```
1
2  #include <pthread.h>
3  #include <stdio.h>
4  #include <stdlib.h>
5  #include <time.h>
6
7  #define STB_IMAGE_IMPLEMENTATION
8  #include "stb_image.h"
9  #define STB_IMAGE_WRITE_IMPLEMENTATION
10 #include "stb_image_write.h"
11
12 int min(int a, int b) {
13   if (a < b) {
14     return a;
15   } else {
16     return b;
17   }
18 }
19
20 int max(int a, int b) {
21   if (a > b) {
22     return a;
23   } else {
24     return b;
25   }
26 }
27
28 void swap(unsigned char **lhs, unsigned char **rhs) {
29   unsigned char *tmp = *lhs;
30   *lhs = *rhs;
31   *rhs = tmp;
32 }
33
34 unsigned char *data = NULL;
35 int width = 0;
36 int height = 0;
37 int comp = 0;
38 int window_size = 0;
39
40 int thread_count = 4;
41
42 int filter_by_average(int left_border, int right_border, int lower_border, int
       upper_border) {
```

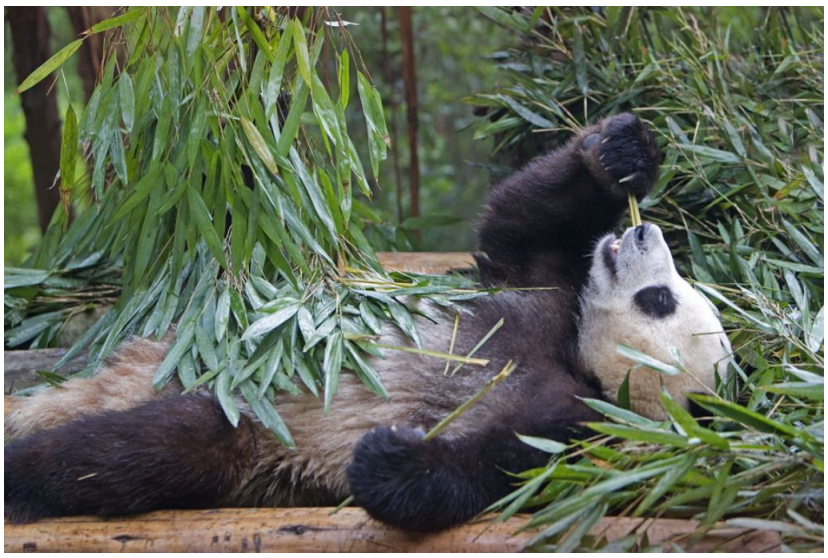
```
43    int sum = 0;
44    int count = 0;
45    for (int i = lower_border; i <= upper_border; ++i) {
46      for (int j = left_border; j <= right_border; j += comp) {
47        sum += data[i * width * comp + j];
48        ++count;
49      }
50    }
51    return sum / count;
52  }
53
54  int compare(const void *x1, const void *x2) {
55    return (*(int *)x1 - *(int *)x2);
56  }
57
58  int filter_by_median(int left_border, int right_border, int lower_border, int
        upper_border) {
59    int size_gor = (right_border - left_border) / comp + 1;
60    int size = (upper_border - lower_border + 1) * size_gor;
61    int arr[size];
62    int arr_idx = 0;
63    for (int i = lower_border; i <= upper_border; ++i) {
64      for (int j = left_border; j <= right_border; j += comp) {
65        arr[arr_idx] = data[i * width * comp + j];
66        ++arr_idx;
67      }
68    }
69    qsort(arr, size, sizeof(int), compare);
70    return arr[size / 2];
71  }
72
73  int filter(int idx) {
74    int real_width = width * comp;
75    int left_border = max((idx % real_width) % comp, idx % real_width - window_size *
          comp);
76    int right_border = min(real_width - comp + (idx % real_width) % comp, idx %
          real_width + window_size * comp);
77    int lower_border = max(0, idx / real_width - window_size);
78    int upper_border = min(height - 1, idx / real_width + window_size);
79
80    //return filter_by_average(left_border, right_border, lower_border, upper_border);
81    return filter_by_median(left_border, right_border, lower_border, upper_border);
82  }
83
84  struct process_buf_args {
85    unsigned char *buf;
86    int from;
87    int to;
88  };
```

```
89  typedef struct process_buf_args process_buf_args;
90
91  void *process_lines(void *args) {
92    process_buf_args *ft = (process_buf_args *)args;
93    for (int i = ft->from; i < ft->to; ++i) {
94      for (int j = 0; j < width * comp; ++j) {
95        int idx = i * width * comp + j;
96        ft->buf[idx] = filter(idx);
97      }
98    }
99    return NULL;
100 }
101
102 void process_image_to(unsigned char *result) {
103   process_buf_args args[thread_count];
104   int step = height / thread_count;
105   if (step == 0) {
106     fprintf(stderr, "The height of the image cannot be greater than the number of
          threads\n");
107     return;
108   }
109   for (int i = 0; i < thread_count; ++i) {
110     args[i].buf = result;
111     args[i].from = step * i;
112     args[i].to = step * (i + 1);
113   }
114   args[thread_count - 1].to = height;
115   pthread_t ids[thread_count];
116   for (int i = 0; i < thread_count; ++i) {
117     pthread_create(&ids[i], NULL, process_lines, &args[i]);
118   }
119   for (int i = 0; i < thread_count; ++i) {
120     pthread_join(ids[i], NULL);
121   }
122 }
123
124 int main(int argc, char **argv) {
125   if (argc < 5 || argc == 6 || argc > 7) {
126     fprintf(stderr, "USAGE: %s <input_file> <output_file.png> <count> <window_size>\n",
          argv[0]);
127     fprintf(stderr, "or\nUSAGE: %s <input_file> <output_file.png> <count> <window_size>
          -t <thread_count>\n", argv[0]);
128     return 1;
129   }
130   if (argc == 7) {
131     if (strcmp(argv[5], "-t") == 0 && atoi(argv[6]) > 0) {
132       thread_count = atoi(argv[6]);
133     } else {
134       fprintf(stderr, "Bad thread arrgument\n");
```

```
135       return 2;
136     }
137   }
138   if (atoi(argv[3]) < 0 || atoi(argv[4]) < 0) {
139     fprintf(stderr, "Bad argument 3 or 4\n");
140     stbi_image_free(data);
141     return 3;
142   }
143
144   data = stbi_load(argv[1], &width, &height, &comp, 0);
145   if (!data) {
146     fprintf(stderr, "Something was wrong during reading\n");
147     stbi_image_free(data);
148     return 4;
149   }
150   window_size = atoi(argv[4]);
151   fprintf(stderr, "width:%d height:%d channels:%d threads:%d\n", width, height, comp,
          thread_count);
152
153   time_t start = time(NULL);
154   unsigned char *buf = (unsigned char *)malloc(width * height * comp * sizeof(unsigned
          char));
155   for (int k = 0; k < atoi(argv[3]); ++k) {
156     process_image_to(buf);
157     swap(&data, &buf);
158   }
159   free(buf);
160   time_t finish = time(NULL);
161   fprintf(stderr, "time:%lds\n", finish - start);
162
163   int res = stbi_write_png(argv[2], width, height, comp, data, width * comp);
164   //int res = stbi_write_bmp(argv[2], width, height, comp, data);
165   stbi_image_free(data);
166
167   if (!res) {
168     fprintf(stderr, "Something was wrong during writing\n");
169     return 5;
170   }
171   return !res;
172 }
```

5 Пример работы

Продемонстрирую работу программы на примере изображения панды. Оригинал:



Используем два раза медианный фильтр с размером окна 5 (11*11 пикселей):

```
windicor@Lina-HP:~$ ./median_filter pictures/panda.jpg res.png 2 5
width:1024 height:683 channels:3 threads:4
time:12s
```

6 Вывод

В процессе работы над лабораторной я научился основам работы потоками в Си. К сожалению, в процессе работы не возникло потребности использования мьютексов, семафоров и т. п., так что нельзя сказать, что материал был освоен полностью на практике. Также было полезным найти библиотеку для работы с изображениями на Си.