Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <stdbool.h>
#include <string.h>

#define MAX_NUMBERS 500
#define BUFFER_SIZE 128
#define NUM_VALUES 5

int x_values[NUM_VALUES] = {17, 29, 99, 777, 19};
int data[MAX_NUMBERS];
int buffer[BUFFER_SIZE];
int buffer_count = 0;

pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
bool produced = false;
int current_x;
FILE *output;
int result_counts[NUM_VALUES] = {0};

void *producer(void *arg) {
    FILE *fp = fopen("data.txt", "r");
    if (!fp) {
        perror("Error opening data.txt");
        exit(EXIT_FAILURE);
    }
    int i = 0;
    while (i < MAX_NUMBERS && fscanf(fp, "%d", &data[i]) == 1) {
        i++;
    }
    fclose(fp);

    pthread_mutex_lock(&mutex);
    for (int j = 0; j < MAX_NUMBERS; j++) {
        if (data[j] % current_x == 0) {
            if (buffer_count < BUFFER_SIZE) {
                buffer[buffer_count++] = data[j];
            }
        }
    }
    produced = true;
    pthread_mutex_unlock(&mutex);
```

```
    pthread_exit(NULL);
}

void *consumer(void *arg) {
    int id = *(int *)arg;
    int val = x_values[id];
    pthread_t prod;

    buffer_count = 0;
    produced = false;
    current_x = val;

    pthread_create(&prod, NULL, producer, NULL);
    while (!produced) {} // Busy wait until producer finishes

    pthread_mutex_lock(&mutex);
    for (int i = 0; i < buffer_count; i++) {
        fprintf(output, "%d ", buffer[i]);
        result_counts[id]++;
    }
    fprintf(output, "\n");
    pthread_mutex_unlock(&mutex);

    pthread_join(prod, NULL);
    pthread_exit(NULL);
}

int main() {
    pthread_t threads[NUM_VALUES];
    int ids[NUM_VALUES];

    output = fopen("result.txt", "w");
    if (!output) {
        perror("Error opening result.txt");
        return 1;
    }

    for (int i = 0; i < NUM_VALUES; i++) {
        ids[i] = i;
        pthread_create(&threads[i], NULL, consumer, &ids[i]);
        pthread_join(threads[i], NULL); // Wait sequentially
    }

    fclose(output);
```

```
    for (int i = 0; i < NUM_VALUES; i++) {
            printf("%d- %d multiple of %d are found.\n", i + 1,
result_counts[i], x_values[i]);
    }
    printf("all found results are stored in \"result.txt\".\n");

    return 0;
}
```

```
roy@LAPTOP-DLJ845NV:/mnt/a/GitHub/OS_codes/A4$ gcc prog.c -o prog -pthread
roy@LAPTOP-DLJ845NV:/mnt/a/GitHub/OS_codes/A4$ ./prog
1- 40 multiple of 17 are found.
2- 12 multiple of 29 are found.
3- 5 multiple of 99 are found.
4- 2 multiple of 777 are found.
5- 32 multiple of 19 are found.
all found results are stored in "result.txt".
```

result - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
765 663 357 357 527 765 714 442 459 391 238 884 697 561 51 714 663 663 901 442 493 374 595 85 578 782 85 799 969 595 612 714 663 187 680 680 408 442
319 609 232 464 841 493 232 725 957 261 841 493
594 396 198 891 594
777 777
912 361 475 931 399 133 665 95 209 304 931 418 722 247 589 912 874 209 361 779 589 684 931 969 133 836 513 285 608 380 760 665
```