Operation on thread

Firstly, study and try to understand what **POSIX threads** is and also how to write and compile the multithreaded C program using POSIX threads *(an example of related website, https://en.wikipedia.org/wiki/POSIX_Threads)*. Then, compiling and running the "question1.1_1.2.c" source code and then answer the question 1.1 and 1.2: *(In addition, press **Ctrl + c** to stop the program execution)*

P.S. Each thread in a process is identified by a **thread ID**. When referring to thread IDs in C or C++ programs, use the type **pthread_t**. The **pthread_create** function creates a new thread. You provide it with the following: *(In addition, please find more information about the **pthread_create** function by yourself)*

1. **A pointer to a pthread_t variable**, in which the thread ID of the new thread is stored.
2. **A pointer to a thread attribute object**. This object controls details of how the thread interacts with the rest of the program. If you pass NULL as the thread attribute, a thread will be created with the default thread attributes.
3. **A pointer to the thread function**. This is an ordinary function pointer, of this type:

   void* (*) (void*)

4. **A thread argument value of type void***. Whatever you pass is simply passed as the argument to the thread function when the thread begins executing.

```
#include <pthread.h>
#include <stdio.h>

/* Prints x's to stderr. The parameter is unused. Does not return. */
void* print_xs (void* unused)
{
      while (1)
            fputc ('x', stderr);
      return NULL;
}

/* The main program. */
int main ()
{
      pthread_t thread_id;
      /* Create a new thread. The new thread will run the print_xs
      function. */
      pthread_create (&thread_id, NULL, &print_xs, NULL);
      /* Print o's continuously to stderr. */
      while (1)
            fputc ('o', stderr);
      return 0;
}
```

question1.1_1.2.c

1.1 **How many threads** have been **concurrently running** in this program? In addition, **what does each thread do**?

1.2 Considering the **result of program execution shown on your monitor screen** and then please explain **why character 'o' and 'x' printed alternately**.

Compiling and **running** the "**question1.3.c**" source code and then **answer the question 1.3**:

```
#include <pthread.h>
#include <stdio.h>

/* Parameters to print_function. */

struct char_print_parms
{
      /* The character to print. */
      char character;
      /* The number of times to print it. */
      int count;
};

/* Prints a number of characters to stderr, as given by PARAMETERS,
which is a pointer to a struct char_print_parms. */

void* char_print (void* parameters)
{
      /* Cast the cookie pointer to the right type. */
      struct char_print_parms* p = (struct char_print_parms*)
parameters;
      int i;

      for (i = 0; i < p->count; ++i)
            fputc (p->character, stderr);
      return NULL;
}

/* The main program. */

int main ()
{
      pthread_t thread1_id;
      pthread_t thread2_id;
      struct char_print_parms thread1_args;
      struct char_print_parms thread2_args;

      /* Create a new thread to print 30,000 'x's. */
      thread1_args.character = 'x';
      thread1_args.count = 30000;
      pthread_create (&thread1_id, NULL, &char_print, &thread1_args);
```

```
      /* Create a new thread to print 20,000 o's. */
      thread2_args.character = 'o';
      thread2_args.count = 20000;
      pthread_create (&thread2_id, NULL, &char_print, &thread2_args);

      return 0;
}
```

Question1.3.c

1.3 **How many threads** have been **concurrently running** in this program? In addition, **what does each thread do**? Please, explain **why nothing printed on the monitor screen after the program executed**.

Compiling and **running** the "**question1.4.c**" source code and then **answer the question 1.4**:

P.S. Firstly, study and try to understand what the **pthread_join** function is. In addition, the **pthread_join** function takes two arguments:

1. The thread ID of the thread to wait for.
2. A pointer to a **void*** variable that will receive the finished thread's return value. If you don't care about the thread return value, pass **NULL** as the second argument.

```
#include <pthread.h>
#include <stdio.h>

/* Parameters to print_function. */

struct char_print_parms
{
      /* The character to print. */
      char character;
      /* The number of times to print it. */
      int count;
};

/* Prints a number of characters to stderr, as given by PARAMETERS,
which is a pointer to a struct char_print_parms. */

void* char_print (void* parameters)
{
      /* Cast the cookie pointer to the right type. */
      struct char_print_parms* p = (struct char_print_parms*)
parameters;
      int i;

      for (i = 0; i < p->count; ++i)
            fputc (p->character, stderr);
```

```
      return NULL;
}

/* The main program. */

int main ()
{
      pthread_t thread1_id;
      pthread_t thread2_id;
      struct char_print_parms thread1_args;
      struct char_print_parms thread2_args;

      /* Create a new thread to print 30,000 'x's. */
      thread1_args.character = 'x';
      thread1_args.count = 30000;
      pthread_create (&thread1_id, NULL, &char_print, &thread1_args);

      /* Create a new thread to print 20,000 o's. */
      thread2_args.character = 'o';
      thread2_args.count = 20000;
      pthread_create (&thread2_id, NULL, &char_print, &thread2_args);

      /* Make sure the first thread has finished. */
      pthread_join (thread1_id, NULL);
      /* Make sure the second thread has finished. */
      pthread_join (thread2_id, NULL);

      /* Now we can safely return. */
      return 0;
}
```

Question1.4.c

1.4 What is the **difference** of the **result of program execution** between the program using in **question 1.3** and **question 1.4**? Please, **give the reason** to support your answer.

1.5 Considering the below **INCOMPLETE** program source code ***(question1.5.c)*** and then **INSERT your code** to **complete** the program for **creating two threads** that a one thread executes the **add** function and another thread executes the **multiply** function, simultaneously. In addition, the **number** variable must be used as the argument of both function when they are called. Please, **explain the detail of your code inserted**, why the **second argument** of the **pthread_join** function of this program is not **NULL** as same as the program using in **question 1.4** and finally **show the result of your completed program execution on a monitor screen**.

```c
#include <pthread.h>
#include <stdio.h>

/* Add a number with its own number and return the result*/
void* add (void* arg)
{
      int n = *((int*) arg);
      int result;

      result = n + n;

      return (void*) result;
}

/* Multiply a number with its own number and return the result*/
void* multiply (void* arg)
{
      int n = *((int*) arg);
      int result;

      result = n * n;

      return (void*) result;
}

/* The main program. */
int main ()
{
      pthread_t thread1_id;
      pthread_t thread2_id;

      int number = 5;
      void *add_result, *mul_result;

      /* Create a new thread to add a number with its own number. */
      /* Insert Your Code #1 */

      /* Create a new thread to add a number with its own number. */
      /* Insert Your Code #2 */

      /* Make sure the first thread has finished. */
      pthread_join (thread1_id, &add_result);
      /* Make sure the second thread has finished. */
      pthread_join (thread2_id, &mul_result);

      printf("The result of add is %d\n", add_result);
      printf("The result of multiply is %d\n", mul_result);

      /* Now we can safely return. */
      return 0;
}
```

Question1.5.c