البرمجة التفرعية باستخدام الـ Multithreading ضمن بيئة الـ VS2010

1 مفردات الجلسة:

- ✓ الأدوات البرمجية
- ✓ مقدمة عن Multithreading
- ✓ تدريب عملي

2 الأدوات البرمجية:

- ✓ تثبيت المكتبة pthreads ضمن بيئة VS2010:
- ✓ بالنسبة للنسخ الحديثة من بيئة الـ VS لا يوجد ضرورة لتثبيت مكتبة جديدة حيث أنها تدعم العمل بـ Multihtreading اعتماداً على المكتبة المعيارية thread.
- ✓ من أجل استخدام الـ Multithreading ضمن بيئة الـ VS2010 يجب اتباع الخطوات التالية:

1. تنزيل المكتبة عبر الرابط: ftp://sourceware.org/pub/pthreadswin32/dlllatest
2. نسخ محتويات المجلد include والتي تحتوي على ملفات من نوع h. إلى مجلد الـ include ضمن بيئة الـ VS2010 المجلد VC
3. نسخ محتويات المجلد lib والتي تحتوي على ملفات من نوع lib. إلى مجلد الـ lib ضمن بيئة الـ VS2010 المجلد VC
4. نسخ محتويات المجلد dll والتي تحتوي على ملفات من نوع dll. إلى مجلد الـ bin ضمن بيئة الـ VS2010 المجلد VC
5. إضافة الاعدادات التالية إلى Properties>Linker>Input ضمن الخاصية Additional Dependencies:

- ✓ pthreadVC2.lib

3 مقدمة عن متعدد الخيوط Multithreading:

يوجد ثلاث أصناف من إجراءات Pthreads

- ✓ إدارة الخيوط (Thread management):

تتضمن إجرائيات الإنشاء وفصلها والانضمام إليها (creating, detaching, and joining)، وما إلى ذلك. وهي تتضمن وظائف لتعيين/الاستعلام عن صفات الخيوط (قابلة للانضمام، والجدولة، وما إلى ذلك)

- ✓ كائنات المزامنة (Mutexes):

توفر إمكانية إنشاء كائنات المزامنة وتدميرها وقفلها وفتحها (creating, destroying, locking and unlocking). تتضمن ايضاً إجرائيات صفات كائن المزامنة (mutex) التي تقوم بتعيين أو تعديل الصفات المرتبطة بكائنات المزامنة

- ✓ متغيرات الحالة (Condition variables):

تتناول وظائف الاتصالات بين الخيوط التي تشترك في كائن المزامنة (mutex). وهي تستند إلى الشروط المحددة للمبرمج. تتضمن هذه الفئة وظائف للإنشاء والتدمير والانتظار والإشارة بناءً على قيم متغيرة محددة (create, destroy, wait). يتم أيضًا تضمين وظائف تعيين/الاستعلام عن صفات متغير الشرط.

1.3 إنشاء الخيوط

يستخدم التابع التالي:

pthread_create (thread, attr, start_routine, arg)

يقوم هذا التابع بإنشاء مؤشر ترابط جديد ويجعله قابلاً للتنفيذ. عادة، يتم إنشاء الخيوط أولاً من داخل التابع الرئيسي main() داخل عملية واحدة.

- ✓ يمكن إنشاء العديد من الخيوط
- ✓ يقوم التابع pthread_create بإرجاع معرف الخيط الجديد عبر بارامتر. يجب التحقق من هذا المعرف للتأكد من إنشاء الخيط (مؤشر الترابط) بنجاح
- ✓ يتم استخدام البارامتر attr لتعيين صفات الخيط. يمكن أن يكون كائنًا، أو NULL للقيم الافتراضية
- ✓ البارامتر start_routine هو التابع الذي سينفذه الخيط بمجرد إنشائه. قد يتم تمرير بارامتر واحدة إلى start_routine من النوع void pointer.
- ✓ الحد الأقصى لعدد الخيوط التي يمكن إنشاؤها بواسطة العملية يعتمد على التنفيذ.

2.3 إنهاء الخيوط

يتم انهاء الخيوط بإحدى الطرق التالية:

- ✓ يخرج الخيط من اجرائية التشغيل الخاصة به (التابع الرئيسي لعملية انشاء الخيط)
- ✓ يقوم الخيط باستدعاء الإجراء الفرعي pthread_exit
- ✓ يتم إلغاء الخيط بواسطة خيط آخر عبر الاجرائية pthread_cancel
- ✓ يمكن أن تظهر بعض المشاكل في تناسق البيانات
- ✓ يتم إنهاء العملية بأكملها نتيجة لاستدعاء إما exec أو exit من الإجراءات الفرعية.

الإجرائية pthread_exit(status)

- ✓ يتم استدعاء الإجرائية pthread_exit() بعد أن يكمل الخيط عمله ولم يعد مطلوبًا للوجود

- ✓ إذا انتهى التابع الرئيسي main() قبل الخيوط التي أنشأها، وخرج باستخدام pthread_exit()، فسوف تستمر الخيوط الأخرى في التنفيذ. -وإلا، فسيتم إنهاؤها تلقائيًا عند انتهاء الدالة main()
- ✓ قد يحدد المبرمج بشكل اختياري حالة الإنهاء، والتي يتم تخزينها كمؤشر فارغ لأي خيط قد ينضم إلى الخيط المتصل

Cleanup:

- ✓ لا تقوم الاجرائية pthread_exit() بإغلاق الملفات
- ✓ يوصى باستخدام pthread_exit() للخروج من كافة الخيوط...خاصة التابع main() والذي يعتبر إحدى الخيوط المنشأة

3.3 تمرير الوسائط إلى الخيوط

- ✓ بدء تشغيل الخيط غير حتمية وهو يعتمد على التنفيذ
- ✓ إذا كنا لا نعرف متى سيبدأ الخيط، فكيف يمكننا تمرير البيانات إلى الخيط مع العلم أنه سيكون له القيمة الصحيحة في وقت بدء التشغيل؟
  - لا تمرر البيانات كوسائط يمكن تغييرها بواسطة خيط آخر
  - بشكل عام، يتم استخدم كائن منفصل من بنية البيانات لكل خيط.

4 تدريب عملي:

1.4 تدريب 1

المطلوب انشاء 5 خيوط يقوم كل منها بتنفيذ تابع فرعي باسم PrintHello والذي بدوره يقوم بطباعة الجملة " Hello World! It's me, thread " مع رقم الخيط

الحل:

```
#include <iostream>
#include <pthread.h>
#include <stdio.h>
using namespace std;
#define NUM_THREADS 5
void *PrintHello(void *threadid) {
        long tid;
        tid = (long)threadid;
        cout<<"Hello World! It's me, thread "<< tid<<endl;
        pthread_exit(NULL);
        return NULL; }
```

```
int main (int argc, char *argv[]) {
        pthread_t threads[NUM_THREADS];
        int rc;
        long t;_
        for(i=0; i<NUM_THREADS; i++){
                cout<<"In main: creating thread "<< t<<endl;
                rc = pthread_create(&threads[i], NULL, PrintHello, (void *)i);
                if (rc){
                        printf("ERROR; return code from pthread_create() is %d\n", rc);
                        exit(-1);
                }
        }
        pthread_exit(NULL);
}
```

2.4 تدريب2

المطلوب تمرير عددين صحيحين إلى الـ Thread عند إنشاءه

الحل:

```
#include <stdio.h>
#include <pthread.h>
struct arg_struct {
  int arg1;
  int arg2;
};
void *print_the_arguments(void *arguments)
{
  struct arg_struct *args = (struct arg_struct *)args;
  printf("%d\n", args -> arg1);
  printf("%d\n", args -> arg2);
  pthread_exit(NULL);
  return NULL;
}
int main()
{
  pthread_t some_thread;
  struct arg_struct args;
  args.arg1 = 5;
```

```
  args.arg2 = 7;

  if (pthread_create(&some_thread, NULL, &print_the_arguments, (void *)&args) != 0) {
    printf("Uh-oh!\n");
    return -1;
  }
  return pthread_join(some_thread, NULL); /* Wait until thread is finished */
}
```