lesson 83 Thread

يقوم الكمبيوتر بعمل أكثر من مليار عملية فى الثانية الواحدة، كيف تتم هذه العمليات فى وقت واحد؟
كمبرمجين نحن نقوم بكتابة البرنامج ليتم تنفيذه خطوة تلو الخطوة على سبيل المثال:

نطلب من المستخدم إدخال رقم، ثم يدخل المستخدم الرقم ويأخذ بعض الثوانى لكتابته، لكن فى هذه الثوانى مالذى يفعله الكمبيوتر ؟!

الكمبيوتر يقوم بأشياء كثيرة، طالما الكمبيوتر يعمل فهناك أكثر من برنامج يعمل فى نفس اللحظة ، كبرنامج الفأرة، أو إذا كان برنامج google chrome أو media player أو هناك لعبة تعمل أو مقطع موسيقى، كل هذا يعمل فى وقت واحد، إذا توقف أحد البرامج فهناك برامج مازالت تعمل،و كل برنامج يعمل فى **processes** .

وبالنسبة للبرنامج الخاص بنا إذا توقف لكى ينتظر أن يدخل المستخدم الرقم، فى هذا الوقت هناك برامج أخرى تعمل، وبالتالي يستطيع الكمبيوتر عمل مليار عملية فى الثانية أو أقل على حسب البرامج الموجودة إذا كانت لا تحتاج لهذا الكم من العمليات .

نعود للبرنامج الخاص بنا، يعمل البرنامج فى **process** واحدة و فى صورة خطوات متتالية كل خطوة تتبع الى تسبقها.

هنا السؤال هل يمكننا جعل البرنامج الخاص بنا يقوم بعمل أكثر من عملية فى وقت واحد؟

- نعم واسمها **thread**

الكمبيوتر يقوم بتشغيل أكثر من برنامج فى وقت واحد وكل برنامج يعمل فى **processes** خاص به

على سبيل المثال :

قمنا بفتح فيديو على الكمبيوتر ومتصفح كروم فى نفس الوقت

كل برنامج منهم يعمل فى **processes** والاثنان يعملان مع بعض فى وقت واحد

لكن نريد داخل البرنامج أن يقوم بعمل أكثر من عملية فى نفس الوقت

فى لغة السى عند كتابة برنامج نقوم بكتابة الاوامر خلف بعضها بالترتيب

لكن هنا نريد هذه الأوامر ان تعمل معاً فى وقت واحد

أو نقوم بعمل أكثر من function فى وقت واحد ويعملوا فى وقت واحد

لكن ما هي الميزة من ذلك؟

إختصار الوقت، بدلاً من عمل جزء من البرنامج ويأخذ وقت وبحاجة إلى عمل **function** أخرى فى نفس الوقت

على سبيل المثال:

فى برنامج الورد **word** أثناء الكتابة هل ينتظر البرنامج حتى انتهى من الكتابة واضغط على **Enter** ثم يقوم بفحص الأخطاء الموجودة فيما كتبته ؟!

أم يقوم بإظهارها للتعديل عليها ويقوم بإظهار الخط الأحمر تحت الكلام الخطأ المراد تصحيحه أو القواعد خطأ

لكن هو أثناء الكتابة يقوم بإظهار الأخطاء، ولكن نحن نكتب يقوم بتنفيذ هذه الأوامر

وبالتالي ال **thread** سيجعل البرنامج الخاص بنا يعمل بصورة اسرع لأنه يقوم بتنفيذ الكثير من الأوامر فى وقت واحد

عند استخدام البرنامج من قبل المستخدمين، سيكونوا سعداء بسرعة البرنامج

لكن كيف نقوم بعمل **Thread**؟

سنأخذ مثال لبرنامج يعمل بال thread :

سنقوم بعمل **2functions**

```
#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
//exit(0); تضمين لأمر

void* fun1(void* v) {
   //pointer * لابد من وضع  thread تعمل مع  fun لجعل
   // funcation إلى  main لنقوم بتمرير متغيرات من
   // نقوم بوضع متغير داخل الاقواس بهذه الصيغة
   printf("Hello Gammal Tech\n");
   return NULL;
   //return  فلابد أن يكون هناك  pointer * بما اننا وضعنا
   // لكن لأننا لن نرجع رقم أو قيمة سنجعله يرجع شئ فارغ ليس له قيمة
   //null نستخدم
}
void* fun2(void* v) {
   // نقوم بوضع متغير داخل الاقواس بهذه الصيغة
   //pointer * لابد من وضع  thread تعمل مع  fun لجعل
   printf("Hi Gammal Tech\n");
   return NULL;
   //return  فلابد أن يكون هناك  pointer * بما اننا وضعنا
   // لكن لأننا لن نرجع رقم أو قيمة سنجعله يرجع شئ فارغ ليس له قيمة
```

```
    //null نستخدم
        // نقوم بوضع متغير داخل الاقواس بهذه الصيغة

}


int main(  ) {
    pthread_t th;
    // thread تعريف
    // th نقوم بتسميتها
    pthread_create(&th, NULL, fun1, NULL);
    // المراد عملها  funcation يشير إلى  &th
    // attrbute لعدم وجودى إى  NULL
    // المراد عملها function اسم fun1
    // function لعدم وجود متغيرات معرفة داخل  NULL
    pthread_create(&th, NULL, fun2, NULL);
    pthread_join(th, NULL);
    // th أن البرنامج لايقف الا بعد انتهاء
    //main  تعنى أن يتوقف  return 0;
    exit(0);
    // أن يتوقف البرنامج نهائياً
}
```

(قم بتجربة الكود بنفسك واضغط هنا)

مثال أخر :

```
#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
//exit(0); تضمين لأمر
#include <unistd.h>
//sleep(  ); تضمين أمر

void* fun1(void* v) {
   //pointer * لابد من وضع   thread  تعمل مع   fun لجعل
   // funcation إلى  main لنقوم بتمرير متغيرات من
   // نقوم بوضع متغير داخل الاقواس بهذه الصيغة
   int i,*n=(int *)v;
   for (i = 0; i < *n; i++) {
       printf("Hello Gammal Tech\n");
       sleep(1);
       // يتوقف ثانية بعد تنفيذ امر الطباعة
   }
   return NULL;
   //return  فلابد أن يكون هناك  pointer * بما اننا وضعنا
   // لكن لأننا لن نرجع رقم أو قيمة سنجعله يرجع شئ فارغ ليس له قيمة
   //null نستخدم
}
```

```
void* fun2(void* v) {
   // نقوم بوضع متغير داخل الاقواس بهذه الصيغة
   //pointer * لابد من وضع  thread تعمل مع  fun لجعل
   int i,*n=(int*)v;
   for (i = 0; i < *n; i++) {
       printf("Hi Gammal Tech\n");
       sleep(1);
       // يتوقف ثانية بعد تنفيذ امر الطباعة
   }
   return NULL;
   //return  فلابد أن يكون هناك  pointer * بما اننا وضعنا
   // لكن لأننا لن نرجع رقم أو قيمة سنجعله يرجع شئ فارغ ليس له قيمة
   //null نستخدم
   // نقوم بوضع متغير داخل الاقواس بهذه الصيغة
}


int main(  ) {
   int n = 5;
   pthread_t th;
   // thread تعريف
   // th نقوم بتسميتها
   pthread_create(&th, NULL, fun1, &n);
   // المراد عملها  funcation يشير إلى  &th
   // attrbute لعدم وجودى إى  NULL
   // المراد عملها function اسم fun1
```

```
    // function لعدم وجود متغيرات معرفة داخل  NULL

    pthread_create(&th, NULL, fun2, &n);
    pthread_join(th, NULL);
    //th ان الكمبيوتر لا يتوقف الا بعد انتهاء
    exit(0);
    // أن يتوقف البرنامج نهائياً
}
```

(قم بتجربة الكود بنفسك واضغط هنا)