ضرب المصفوفات

ضرب المصفوفات

في هذه المحاضرة سنعمل على ضرب مصفوفتين بحجم 3×2 , 2×3

الفكرة معقدة قليلا من حيث الرياضيات فيها لكن كفكرة تفرعية هي بسيطة.

نلاحظ لدينا المصفوفتين A,B وناتج ضربهما هو C على الشكل التالي

1	2	2
3	1	4

A=3 × 2
$n \times m_1$

×

1	3
2	1
4	3

B=2 × 3
$m_2 \times b$

=

13	11
21	22

C =2 × 2
$n \times b$

كيف سنمثل ذلك تفرعيا ؟؟

باستعمال spawn ننشئ أبناء بعدد عناصر C وفي مثالنا يكون عدد المهام 4

نقوم بإعطاء كل ابن سطر وعمود ليقوم بإيجاد الناتج وإرساله للأب بحيث تكون المهام كالتالي:

المهمة الأولى (السطر الأول العمود الأول.)

المهمة الثانية (السطر الأول العمود الثاني.)

المهمة الثالثة (السطر الثاني العمود الأول.)

المهمة الرابعة (السطر الثاني العمود الثاني.)

تذكرة :

شرط ضرب مصفوفتين هو عدد أعمدة المصفوفة الأولى يساوي عدد أسطر المصفوفة الثانية.

ويكون لكل رسالة tag يمثل مكان النتيجة في المصفوفة C فمثلا مجموع ناتج ضرب عناصر السطر الأول من A مع ما يقابلها من عناصر العمود الأول من B يكون في الخانة (1,1) من C فيكون tag المهمة الأولى (1,1) وهكذا باقي المهام

كي نستطيع إرسال الأسطر والأعمدة بين المهام عن طريق pvm يجب أن تكون المصفوفة أحادية وبالتالي سيتم وضع الأسطر خلف بعضها البعض فتكون المصفوفات كتالي :

A:

1	2	2	3	1	4

B:

1	3	2	1	4	3

C:

13	11	21	22

في المصفوفات الأحادية يكون لدينا مؤشر على أول عنصر من المصفوفة وللانتقال إلى عنصر ما z نضيف z إلى هذا العنصر, وفي المصفوفات الثنائية يتم التعامل مع الأعمدة بنفس الطريقة , أما بالنسبة للأسطر, نضيف i*m للانتقال إلى السطر i حيث m هو عدد الأعمدة , مثال:

لتكن لدينا المصفوفة التالية $h = \begin{bmatrix} 1 & 6 & 9 \\ 7 & 3 & 5 \\ 4 & 12 & 8 \end{bmatrix}$, نمثلها بمصفوفة أحادية فتصبح:

$$values = [1 \quad 6 \quad 9 \quad 7 \quad 3 \quad 5 \quad 4 \quad 12 \quad 8]$$
$$indices = [0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8]$$

ونريد الوصول للرقم 5 وهو في العمود 2 وفي السطر 1 (إذا بدأ العد من الصفر) , بالنسبة للأعمدة نضيف رقم العمود فقط أي 2 وبالنسبة للأسطر نضيف عدد الأعمدة * رقم العمود أي 3*1 = 3 فيكون المؤشر h+5 = h+3+2 وهو الindex المناسب.

مثال آخر , القيمة 12 وهي في السطر 2 وفي العمو 1 فتكون الخانة h+7= (2*3)+h+1

نعود إلى المسألة لنرى كيف سنرسل الأسطر والأعمدة .

كيف ستعمل وأين سيتم وضع النتائج

بداية عندما أقوم بـ spawn لـ 4 أبناء سيكون لدي 4 ملفات exe تعمل في الذاكرة ولكن ولا أي من الملفات يعلم شيئا عن الاخر

عندما أرسل سطر وعمود للابن لا يعلم أي سطر وعمود تم ارساله انما سيرى مصفوفتين احاديتين عليه ضربهما

الأربع أبناء سيعملون على 4 أجهزة مثلا ويمكن ان ينتهي الابن الرابع مثلا أولا قبل الأول فترسل النتائج بالشكل التالي: 22←13←11←21 فيتوجب علينا ان نعرف كل قيمة اين ستوضع في مكانها الصحيح لذا نستعمل الـ tags (وهو مهم جدا للحسابات التفرعية)

- أي سنرسل تاغ للابن الذي سيحسب القيمة في المصفوفة ويكون لدي 4 قيم للتاغ هنا :
  - الابن اللأول ارسل له قيمة التاغ 0.
  - الابن الثاني ارسل له قيمة التاغ 1.
  - الابن الثالث ارسل له قيمة التاغ 2.
  - الابن الرابع ارسل له قيمة التاغ 3.

فمثلا انتهى الابن الثالث وحين ينتهي سيخزن القيمة في C2 وهكذا ....

اللابن سيقرأ التاغ (يكون مع recv (اسم الاب)) ويعرف قيمة التاغ عن طريق اللصاقة وفي الrecv يوجد (-1) لأنه لا يعرف انه لدي 4 ملفات exe ولكنها لها نفس الكود, لذا لا يمكننا التقرير بان هذا الكود يأخذ هذه القيمة وهذا الكود يأخذ قيمة أخرى فالجميع يستقبل recv(-1,-1)

أي كل أبن سيأخذ التاغ من اللصاقة ويقوم بعملية الضرب ويرجع الناتج من التاغ نفسه الذي استقبله.

- عن طريق التاغ استطيع الربط بين الاباء والابناء.
- الid لا يمكننا الاستفادة منه لانه يتم تحديد outo من قبل pvm بينما التاغ نحن من يقوم بتحديده.

المعادلات التي سنتبعها على المصفوفة الأحادية $i * p + j$ حيث p تشير الى عدد الاعمدة (القفزة) والعمود الأول هو أول خانة :

- I=0 , J=0 ⟶ 0
- I=0 , J=1 ⟶ 1
- I=1 , J=0 ⟶ 2
- I=1 , J=1 ⟶ 3

هذا يعني انه إذا اردنا ان نرسل السطر الأول من A سنرسل اول خانات (1,2,2)

1	2	2	3	1	4

إذا اردنا ارسال العمود الثاني من B نقوم بتحديد p أي عدد الاعمدة التي سنتجاوزها للحصول على الخانات المطلوبة وهي هنا واحد أي سيتجاوز اول خانة وينتقل للاخرى ثم يتجاوز الخانة التي تليها وهكذا فيكون العمود (3,1,3)

1	3	2	1	4	3

الكود الخاص بالأب hello

في مثالنا سنقرأ أبعاد المصفوفات من ملف موجود على القرص C وسوف تكون به الأرقام التى ستتم ضربها حيث ابعاد اول مصفوفة 2*3 والثانية 1*2

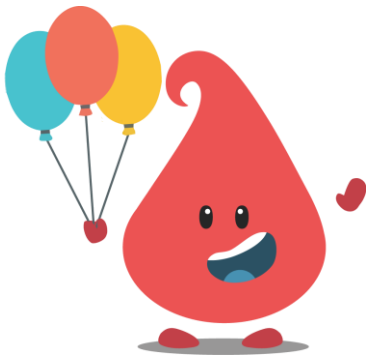
نشغل ال vmare كما في المحاضرة السابقة ثم نفعل ال pvm عن طريق cd c:\pvm\lib\win32 ثم pvm.exe.

read_matrix (int *T, int d1, int d2) تابع يقوم بقراءة مصفوفة أبعادها d1,d2

الملف يحتوي على :

أبعاد المصفوفة الاولى 2 3

المصفوفة الأولى 1 2 2



3 1 4

أبعاد المصفوفة الثانية 3 2

المصفوفة الثانية 1 3

2 1

4 3

```
int main(int argc, char* argv[])
{
    int n_child, pr;
    int i, j, k;
    int info, bufid, bytes, tag, tid;
    int n, m, m1, m2, p;
    int s;
    int *childID;
    int *A, *B, *C;

    freopen("c:\\m.txt", "r", stdin);
    scanf("%d %d", &n, &m1);
    A = (int*) malloc(n * m1 * sizeof(int));
    //creat_matrix(A, n, m1);
    read_matrix(A, n, m1);
    scanf("%d %d", &m2, &p);
    if (m1 != m2) {
        printf("error in matrix dimension\n");
        return 1;
    }
    B = (int*) malloc(m2 * p * sizeof(int));
    //creat_matrix(&B, m2, p);
    read_matrix(B, m2, p);

    C = (int*) malloc(n * p * sizeof(int));
    //creat_matrix(&C, n, p);
    m = m1 & m2;
    pr = n * p;
```

Freopen
يغير مصدر الدخل لمؤشر الملفات , وفي مثالنا تغير مصدر stdin من الconsole إلى الملف m.txt

ثم قمنا بقراءة أبعاد المصفوفة من الملف وحجز ذاكرة لها ثم قراءتها باستعمال التابع read_matrix

Pvm_mytid() للتذكرة فقط , ليس له علاقة بالمسألة

```
childID = (int*) malloc(pr * sizeof(int));

print_matrix(A, n, m, "A");
print_matrix(B, m, p, "B");

pvm_mytid();
n_child = pvm_spawn("hello_other", (char**)0, 0, "", pr, childID);

if (n_child == pr) {
    for (i=0; i<n; i++) {
        for (j=0; j<p; j++){
            pvm_initsend(PvmDataDefault);
            pvm_pkint(&m, 1, 1);
            pvm_pkint(A+i*m, m, 1);
            pvm_pkint(B+j, m, p);
            pvm_send(childID[i*p+j], i*p+j);
        }
    }
    for (k=0; k<pr; k++) {
        bufid = pvm_recv(-1, -1);
        info = pvm_bufinfo(bufid, &bytes, &tag, &tid);
        pvm_upkint(&s, 1, 1);
        C[tag] = s;
    }
```

- لدينا المتحولات:
  (n ,m1) أبعاد المصفوفة الأولى, (m2,p) أبعاد المصفوفة الثانية
- مصفوفة childID عبارة عن مصفوفات ال id التي ستمرر لل spon
- *A ,*B,*C ثلاث مصفوفات سنقبوم بتعبئتهم
- Read_matrix : استدعينا التابع لنقوم بتعبئة القيم في المصفوفة
- Pr : عبارة عن عدد الابناء وهي p * n
- نحجز مصفوفة للابناء childID أبعادها يساوي pr عدد الأبناء المنشئين
- pvm_spawn : ينشئ الأبناء hello_other ونمرر للتابع pr وهو عبارة عن عدد المهام , childID المصفوفة التي سيخزن النتائج بها ثم يتم تخزين الناتج في n_child للتاكد من عدد الأبناء المنشأ
- نتحقق اذا كان عدد الأبناء المنشأ يساوي pr (تم إنشاء 4 أبناء الآن **لدينا مرحلتين** : مرحلة ارسال سطر وعمود ...., ومرحلة استقبال)

↙ مرحلة الارسال :

لدينا حلقتين for للمرور على جميع الأسطر والأعمدة

- Pvm_initsend(pvmdatadefault) : نقوم بالتهيئة للارسال
- Pvm_pkint(&m,1,1) : نمرر أولا m وهي أبعاد المصفوفة, المتحول الثاني عدد العناصر المحزمة, المتحول الثالث: القفزة
- Pvm_pkint(A+i*m,m,1) : تحزيم الأسطر حسب المعادلة في أول بارامتر.
- Pvm_pkint(b+j,m,p) : تحزيم العمود حسب المعادلة نلاحظ أيضا تم التحزيم بشكل متتالي
- Pvm_send(childID[$i*p+j$], $i*p+j$) : تابع الارسال وفيه اول بارمتر يشير الى ال id الذي سنرسل اليه حيث ترسل الى childID حسب قيمة $i*p+j$, والبارمتر الثاني هو التاغ الخاص بالرسالة ويتم إرسال التاغ مع الرسالة من اجل عند إرسال النتيجة نعرف أين سيتم وضع القيمة في مصفوفة النتائج

↙ مرحلة الاستقبال:

لدينا حلقة for للمرور على عدد الأبناء

- Pvm_recv(-1,-1): للاستقبال و ال (-1,-1) تدل على الاستقبال من أي ابن أنهى مهمته
- Pvm_bufinfo(bufid, &bytes, &tag, &tid) : قراءة معلومات ال bufid buffer نمرر له خرج عملية الاستقبال
  - &tag : تاغ الرسالة (أهم وسيط , لأنه يمثل الخانة المناسبة للنتيجة)
  - &bytes: حجم الرسالة
  - &tid: id المرسل
- Pvm_upkint(&s, 1, 1): لفك التحزيم
  - &s: ناتج ضرب السطر بالعمود نفك تحزيمها ونضعها في مصفوفة ال C في خانة ال tag ([tag]C)

شرح كود Hello_other

فكرة عن الكود : الابن سيستقبل سطر وعمود ويرد الناتج للأب

```
int main()
{
    int *A, *B;
    int i, s, m;
    int mytid, master;
    int info, bufid, tag;

    //mytid = pvm_mytid();
    master = pvm_parent();

    bufid = pvm_recv(master, -1);
    info = pvm_bufinfo(bufid, 0, &tag, 0);

    pvm_upkint(&m, 1, 1);

    A = (int*) malloc(m * sizeof(int));
    B = (int*) malloc(m * sizeof(int));

    pvm_upkint(A, m, 1);
    pvm_upkint(B, m, 1);

    s = 0;
    for (i=0; i<m; i++)
        s += A[i] * B[i];

    pvm_initsend(PvmDataDefault);
    pvm_pkint(&s, 1, 1);
    pvm_send(master, tag);

    pvm_exit();
    return 0;
}
```

يعرف الابن مصفوفتين A,B

Master: تعطينا id الأب

لدينا 3 خطوات

- Pvm_recv : استقبال من الأب أياً كان التاغ
- Pvm_bufinfo(bufid, 0, &tag, 0): قراءة معلومات الرسالة والتاغ
- Pvm_upkint(&m, 1, 1): نفك تحزيم عدد العناصر m ,1 عدد العناصر 1 , القفزة

حجزنا مصفوفتين A, B بطول m ,لقراءة المصفوفتين نقوم بفك تحزيمهم مع ملاحظة أن القفزة تساوي 1 لأن المصفوفتين احاديتين ولأن عملية القفز تمت في الأب فهنا تصل للابن جاهزة لعملية الضرب مع الاخرى

وأخيرا مرحلة ارسال النتيجة الى الاب :

تهيئة , تحزيم , ارسال

- لدينا حلقتين for من $0 \rightarrow m$ من اجل عملية الضرب
- Sهو متحول التخزين
- Pvm_initsend تهيئة الارسال
- Pvm_pkint تحزيم الرسالة
- Pvm_send إرسال الرسالة مع التاغ نفسه الى الأب.

نلاحظ أن الكود يتم تنفيذه أربع مرات بعدد الابناء.

ملاحظة: Pvm_upkint , Pvm_pkint كلاهما يتعامل مع عناوين في الذاكرة وحجم البيانات المراد إرسالها أو استقبالها لذلك عندما نريد إرسال متحول واحد نقوم بتمرير عنوانه (عن طريق العملية &) ونضع عدد العناصر 1 لأنه متحول وحيد , ولا يهم طول القفزة لأنه لن يقوم بالقفز أساساً إذ لدينا عنصر واحد فقط

انتهت المحاضرة