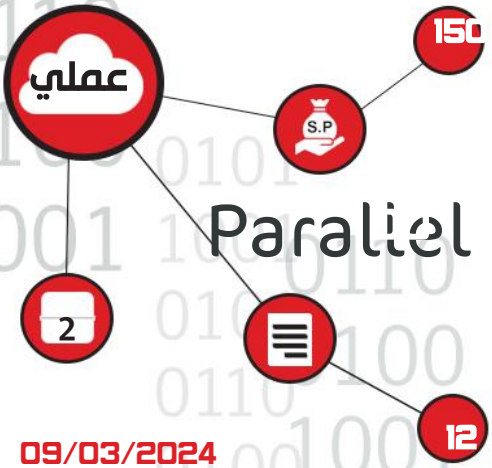
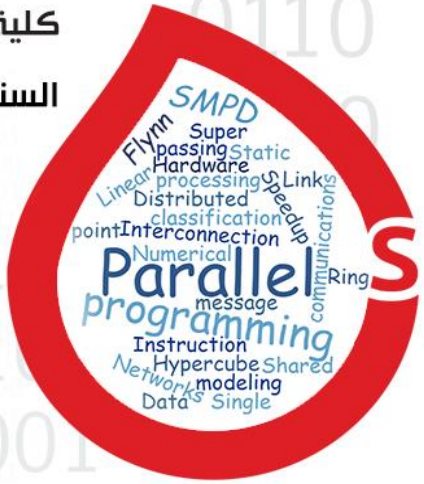


Paralel virtual machine



م. عمار المصري

09/03/2024

RB Informatics;

البرمجة التفرعية

لمحة بسيطة عن المحاضرة السابقة

- ذكرنا أن الهدف الرئيسي من مادة التفرعية السماح ل process 2 للتخاطب مع بعضهم البعض وبالتالي السماح ل process بالوصول ل memory space الخاصة بال process الأخرى وذكرنا أنه لدينا طريقتين أما عن طريق ال pipeline أو TCP \ IP لتسهيل التعامل بالنسبة للمبرمج

- بدانا بمكتبة PVM المبنية باستخدام برتوكول مهمتها الرئيسية تبادل البيانات من خلال عملية message passing
ميزة PVM أنها تسمح ل process 2 بالتخاطب على نظامي تشغيل مختلفين سواء windows او Linux و ذكرنا انها تعمل بنظام client / server من خلال PVM/PVMd
حيث PVM هو البرنامج المكتوب بتعليمات PVM من قبل المبرمج وPVMD هو local Deamon المسؤول عن تشغيل هذا البرنامج

- قلنا ان process الخاصة بال PVM نطلق عليها PVMtask

- لكل local Deamon موجود لدينا على مستوى الشبكة له address book فيه المواصفات الشبكية لكل PVM tasks الموجودة لدينا على مستوى الشبكة (في حالة نشاط)
حيث من اهم المواصفات الشبكية هي اسم الجهاز الذي تعمل عليه task , اسم task , port number

- ذكرنا انه اذا كان هناك task تريد التخاطب مع task أخرى يتم عن طريق local Deamon اذ تقوم بأخذ المواصفات الشبكية للتخاطب بشكل مباشر (في حال الجهازين مختلفين)

Parallel Virtual Machine (PVM)

- PVMمن اقدم المكتبات التفرعية مبنية على c/ c++ ، تدعم c/ c++ و Fortran ، تشغل ضمن شبكات هجينة أي انها محمولة على Linuxو Windows.
- من حيث نظام Windows، تعتمد PVM على مكتبة wSock32، و هي مكتبة قديمة لم تعد موجودة في إصدارات Windows الحديثة، لذلك تحتاج تشغيل PVM الى نظام Windows 7 او Windows XP.
- كبيئة تطوير تربط PVM مع Visual Studio 6.0 الذي يوفر C Compilerالذي تحتاجه PVM ويتوافق مع مكتبة wSock32.
- اما في Linux يكفي العمل على editor كون Linux تملك افتراضيا Standard C/C++.

Parallel Task Registration

- أي برنامج يعمل ضمن النظام هو by default يعتبر os task (os process).
- و من قبل نظام التشغيل يتم إعطائها process id (by default).
- لكل os processعلى مستوى الجهاز لها process id.

متى اجعل os task لتكون SPVM task

ذكرنا ان اغلب تعاملنا سيكون مع PVM task
سنقوم بالتعرف على مجموعة من التوابع الخاصة بال PVM
عند استدعاء احد تابعي التسجيل PVM-mytid() , PVM-parent () يتم الاتصال ب local Deamon
و استدعاء PVMD الذي يقوم بتسجيل البرنامج ك PVM task و إعطائه task ID و حفظ معلومات (Task ID- Parent ID) للمهمة و بالتالي يعمل البرنامج ك PVM task+ يتم انهاء المهام الحالية

ملاحظة: يتم استخراج توابع ال PVMضمن المجال المحدد من تسجيل الPVM task بأحد تابعي التسجيل وحتى انهاء التسجيلة و أي استدعاء خارج هذا ال scopeيتم إعطاء خطا Runtime ERROR لان كل تابع يحتاج ان يكون مرتبطا بمهمة تتصل مع ال PVMD لتنفيذه.

PVM-mytid(): يقوم بإرجاع id المهمة الحالية

PVM-parent(): يقوم بإرجاع id الاب للمهمة الحالية

ملاحظة: من خلال الثنائية (Task ID, Parent ID) يتم التواصل حيث التواصل يكون بين الاب ومجموعة الأبناء التي تم انشائها

عند تشغيل PVM task جديدة يتم إعطائها task id جديدة غير معطى من قبل لئلا يحصل تضارب بين المهمة قيد التشغيل (الحالية) والمهام الموجودة والتي تم الانتهاء من تنفيذ كود البرمجة الخاص بها (انتهى دورها).

- **عندما يتم التخاطب بين مهمتين على نفس الجهاز لسنا بحاجة الى local Deamon لان معلومات الاتصال لدينا (ال id الحاص بـparent محفوظ و كذلك جميع id الأبناء محفوظة على نفس الجهاز و يتم التخاطب بين process 2 بشكل مباشر).**
- **عملنا بشكل دائم خلال هذه المادة سيكون (التخاطب سيتم على نفس الجهاز) فانا بحاجة ل PVMD لتشغيل البرنامج ك PVM task فقط وإعطائها task id وليس للتخاطب بين process 2.**
- **إذا سيتم تشغيل VMware واحد على الجهاز خلال هذه المادة وانشاء parent task وانطلاقا منها سيتم انشاء ابن واحد او عدة أبناء حيث الابن هو PVM task وكذلك الاب ولكل منهما resource خاصة و كل منهما سيقوم بالعمل على التفرع و يتم التخاطب فيما بينهما من خلال عملية message passing و سيكون مسموح لكل منهما بالوصول ل memory space الخاصة بال task الأخرى.**

1. في حال تم استدعاء ??? PVM pare في مهمة الأب يقوم بإرجاع قيمة سالبة
2. مهمة الأب يتم إنشاؤها ضمن ال terminal
3. المهام الأبناء تشغّل في الذاكرة ولا ترتبط بال terminal (أي ليس لها وحدة خرج) وبالتالي تقوم بإرسال النتائج إلى المهمة الأب لوصول إلى الناتج النهائي حيث أن كل مهمة أب تملك ID s المهام الأبناء ترد لها من خلال التابع PVM Spawn()
4. كل مهمة أب تملك ID المهمة الأب لها
5. الابن مهمته بشكل رئيسي تخفيف وحمل مهام عن الآباء (send \ Receive) PVM Communication

كيف تتم عملية الاستقبال والارسال (message passing) بين الأب والأبناء

يتم عملية الأب والابن بنفس اللحظة الزمنية (على التفرع) لكل منها resource خاصة بها وفي لحظة زمنية يتم إرسال رسالة من طرف لآخر وفي أي لحظة زمنية يتم استلام الرسالة من قبل الطرف الآخر

فما هي خطوات الإرسال وخطوات الاستقبال؟

يتم التواصل بين tasks عن طريق عمليات الإرسال والاستلام حيث كل خطوة من الخطوات لها تابع سيتم التعرف عليه

عملية الإرسال

1. تهيئة الرسالة باستخدام أنماط معيارية معينة لنحقق عملية الإرسال مثلا التأكد من نمط البيانيات المرسلة وقابلية فهمها لدى المستقبل المرسلة إليه
2. تحزيم الرسالة تحديد مواصفات الرسالة وتحديد من المستقبل (اللصاقة)
3. إرسال الرسالة

عملية الاستقبال:

1. استقبال الرسالة
2. قراءة معلومات اللصاقة buffer مثل معرفة حجمها ؟؟؟؟ إذا كان المستقبل المراد
3. فك تحزيم الرسالة unpacking

حيث تتم هذه العملية بإحدى الطريقتين

Non-Blocking && Blocking

Blocki

عند الوصول إلى تعليمة الإرسال (آخر مرحلة بعملية الإرسال)

وتعليمة الاستقبال (أول مرحلة بعملية الاستقبال) في الكود البرمجي

يدخل البرنامج في حالة Block(أي يتم إيقاف تنفيذ البرنامج لبرهة من الزمن) حيث تتم عملية الاستقبال أو الإرسال ومن ثم يتابع التنفيذ

Non Blockin

بغض النظر إذا تم العمر أو لا ولكن هنا يجب الانتباه ألا تتم عملية قراءة معلومات اللصاقة طالما لم يتم استلام أي رسالة.

سيتم التعامل مع الآلية المتزامنة

ملاحظات

1. لا يتم إرسال من task1إلى task2 بشكل مباشر فعليا ضمن task1 (المرسل) يوجد لدينا send Buffer وضمن task2 (المستقبل) يوجد لدينا receive Buffer

2. Task1تدخل بحالة Blockحتى التأكد من خروج الرسالة لديها وتخزينها بال send Buffer بشكل صحيح وبالتالي send Buffer هو المسؤول عن تسليمها ل receive Buffer عندها يتم القرب من مرحلة Blockوتكمل تنفيذ التعليمات البرمجية الخاصة به
3. Task2 مرحلتها الأولى وبالآلية المتزامنة تدخل بمرحلة Blockحتى يتم التأكد من أن receive Buffer استلم الرسالة بشكل صحيح عندها يتم التمرير من عملية Block وتتم قراءة المعلومات وفك التحزيم
4. عند استخدام التابع PVM-recv تحفظ الرسالة الواردة ضمن system Buffer مخصص (receive Buffer) تكون القيمة cc المعادة من التابع هي عنوان ال buffer في الذاكرة
5. المهام الأبناء تشغل في الذاكرة ولا ترتبط ب terminalأي ليس لها واجهة خرج مباشرة ولطباعة خرج مهمة ابن يتم إرسال هذا الخرج للمهمة الأب التي تملك terminal لإظهار الخرج ضمنه
6. عند إنهاء تسجيل مهمة لإيعاد استخدام ال id الذي كانت تملكه في تسجيل مهمة جيدة حيث يستمد إلى PVMD في إعطاء قيم جيدة للمهام التي يسجلها حتى نهاية مجال الأرقام المتاح وذلك لتجنب حدوث أخطاء تداخل بين مهام مرتبطة بالمهمة المنتهية وللمهمة الجديدة.

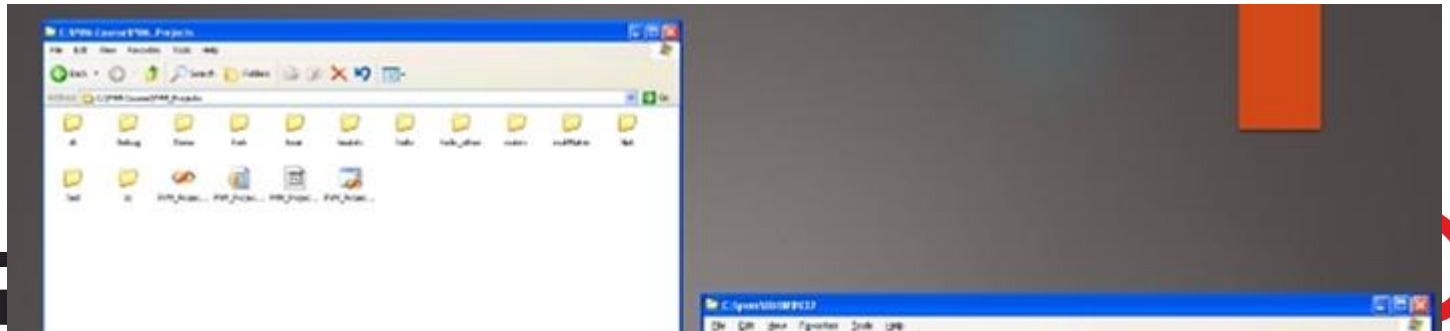
Parent\ child Relation

- ال PVMDيحفظ العلاقات بين المهام عند تسجيلها أي من أجل كل مهمة يقوم بتسجيل ال ID الخاص بها و IDالمهمة المنشأة لها parent ID
- المهمة الابن: في كل مهمة تنشأ برمجيا عبر التابع PVM_spawnونحصل على ال idالمهمة المنشاة لها عبر التابع () PVM_Parent
- المهمة الأب: كل مهمة تنشأ مباشرة على ال Terminalليس لها مهمة أب فقط استدعاء تابع () pvm_parent لها يرد قيمة سالبة

- الاتصال بين الآباء والأبناء مباشر ولا يحتاج ل local daemonلأن المواصفات الشبكية معروفة لدى الطرفين لإنشاء اتصال مباشر حيث أن:
  - كل مهمة أب تملك ID s المهام الأبناء لها ترد لها من خلال التابع PVM dpawn
  - كل مهمة ابن تملك idالمهمة الأب لها.
- إن المهام الأبناء تشغل في الذاكرة ولا ترتبط ب Terminal (أي ليس وحدة خرج مباشرة)

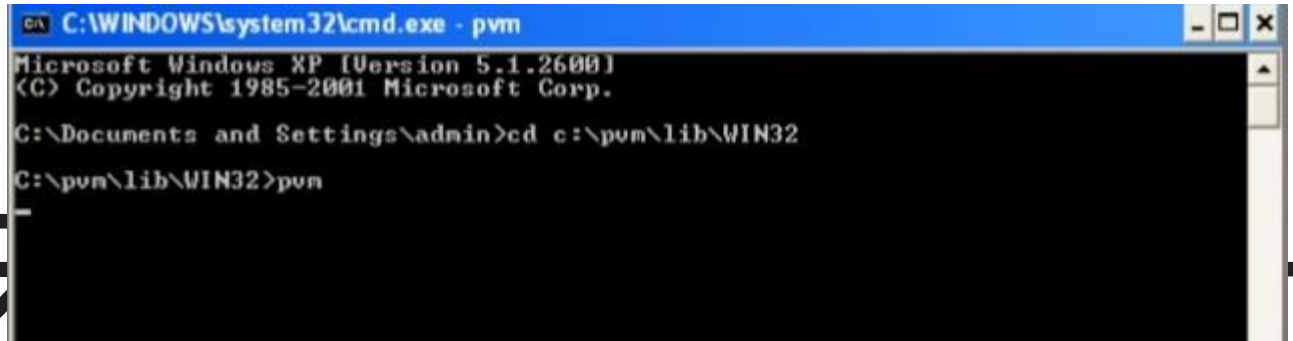
PVM Files

- لدينا ضمن القرص :C
  - كافة المشاريع التي سنعمل عليها متواجدة ضمن المجلد C:\PVM-PVM\Projects_Course
  - المكتبات البرمجية الخاصة بال PVMموجودة ضمن المسار C:\PVM-Cource\PVM_Projects
  - الملفات التنفيذية للبرامج متواجدة ضمن المسار C:\PVM\bin\WIN32



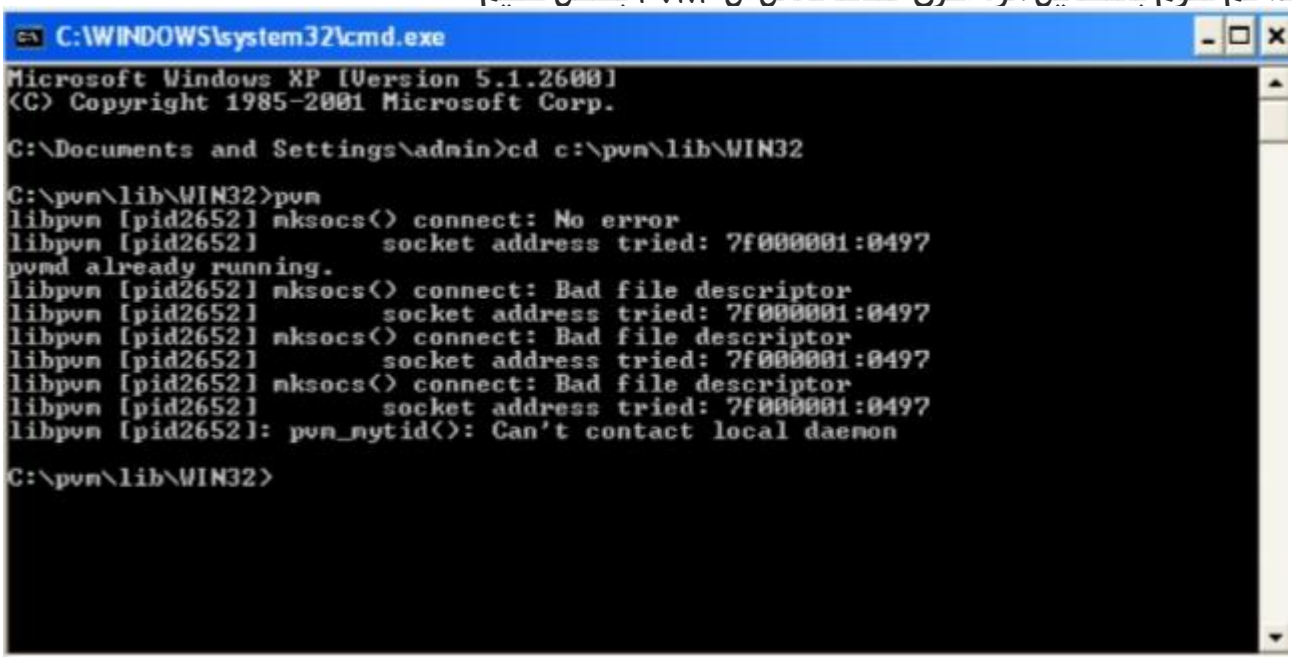
STEP ONE

- تشغيل ال PVM لتأمين جاهزية بيئة العمل التفرعية:
  - عن طريق ال CMD يتم التوجه إلى المسار الخاص بال PVM (c:\PVM\lib\win32)
  - نقوم بتنفيذ الأمر PVM.exe ونواجه حالتين عند التشغيل



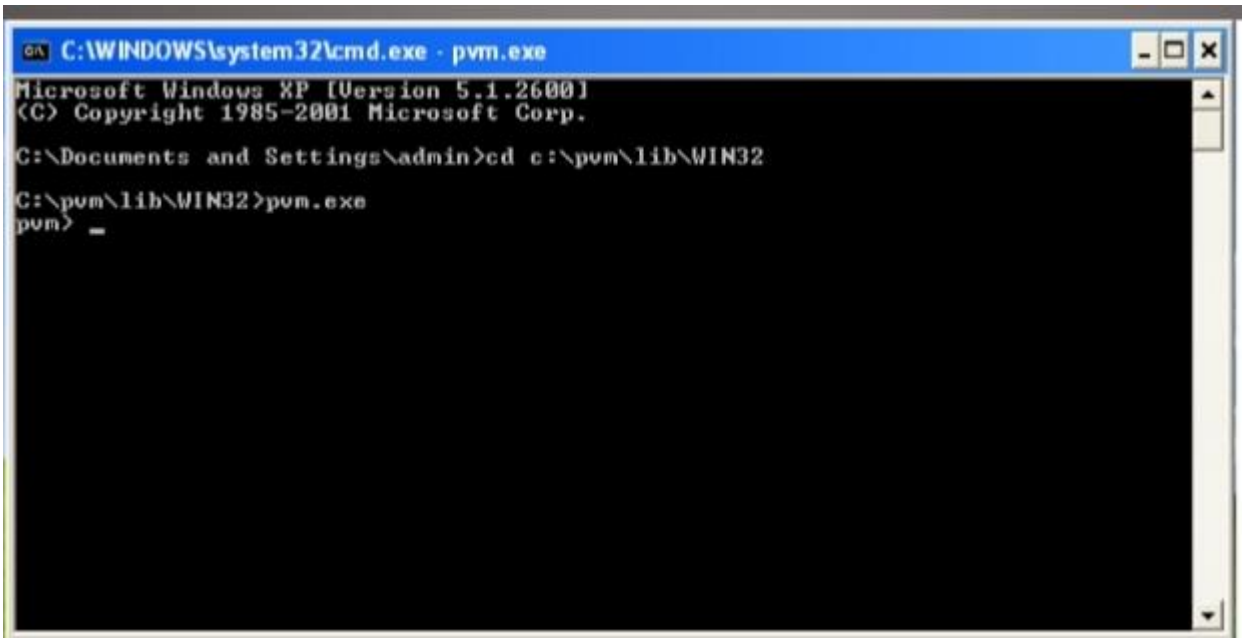
الحالة الأولى

أن يكون قد تم إغلاق ال PVM بشكل غير نظامي وفي هذ الحالة بن يعمل وستظهر الرسالة PVM already running ولحل هذه المشكلة نتوجه إلى مجلد PVM_temp (المتواجد على سطح المكتب)ونقوم بحذف كافة الملفات المتواجدة بداخله قم نقوم بالتشغيل مرة أخرى عندها تعمل ال PVM بشكل سليم



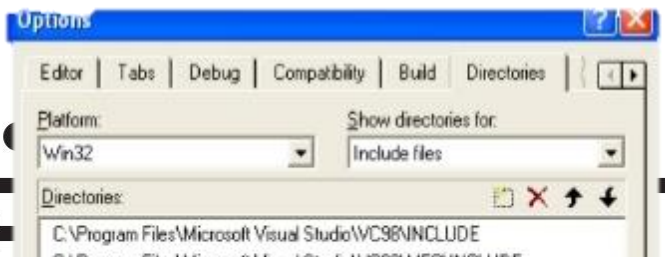
الحالة الثانية:

أن يكون قد تم إغلاق ال PVM بشكل سليم عندها سيظهر لدينا مباشرة على الشاشة PVM ولإغلاق ال PVM بشكل سليم نستعمل التعليمة halt

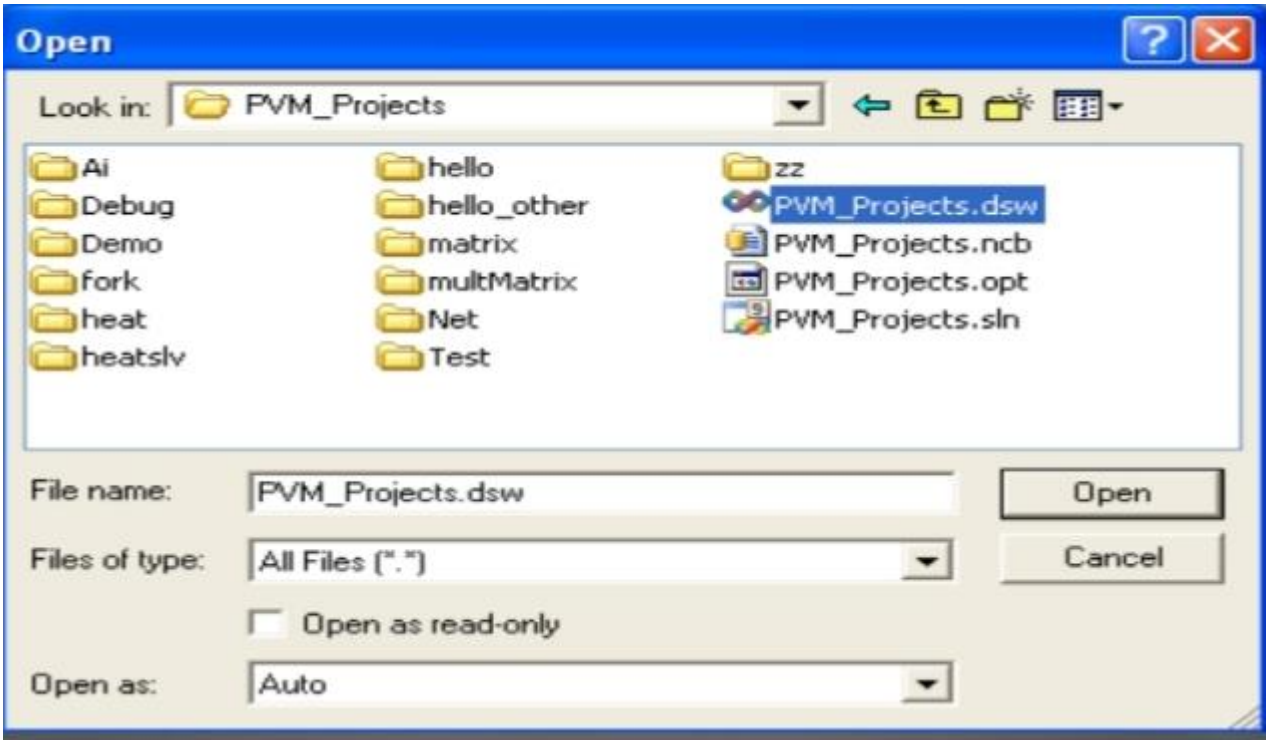


STEP TWO

- ربط ال PVM مع Microsoft visual c++ 6.0
  - من خيار tools ثم optionعند فتح النافذة نختار Directoriesثم نقوم بتحديد مسار c:\PVM\include من خيار include files
  -
  - ونكرر نفس العملية لأجل ال Library Filesعبر تحديد المسار c:\PVM\lib\win32



- للقيام بفتح المشاريع نعرض كافة الملفات ضمن المجلد PVM_project .ونختار الملف ذو اللاحقة dsw من أجل عرض كافة المشاريع ضمن ال worckspace



The Code:

- لدينا برنامجان هما:
  - Hello :وسيكون البرنامج الأب
  - Hello_ other :وسيمثل برنامج الابن

HELLO

```
1)  #include "pvm3.h"
2)  main()
    {
4)     int cc, tid;
5)     char buf[100];
6)     printf("i'm t%x\n", pvm_mytid());
7)     cc = pvm_spawn("hello_other", (char**)0, 0, "", 1, &tid);
8)     if (cc == 1)
       {
9)         cc = pvm_recv(-1, -1);
10)        pvm_bufinfo(cc, (int*)0, (int*)0, &tid);
11)        pvm_upkstr(buf);
12)        printf("from t%x: %s\n", tid, buf);
       }
13)        else
14)        Printf("can't start hello_other\n");
15)    pvm_exit();
16)    exit(0); }
```

- في السطر السادس يتم تسجيل البرنامج ك PVM Task عبر استدعاء التابع PVM_mytid() والذي يقوم بطباعة ال Task ID لمهمة الحالية (المهمة الأب)

- في السطر السابع التابع PVM_spawn() هو المسؤول عن إنشاء الأبناء والبارمترات الخاصة بهذا هي:
  - الملف التنفيذ للابن المراد إنشاء مهمة نسخة منه
  - البارامتران الثاني والثالث نحتفظ بقيمتها كما هي حيث 0(* int) يعبر عن القيمة null
  - البارامتر الرابع يعبر عن عدد المهام الأبناء التي نرغب بإنشائها
  - البارامتر الخامس يعبر عن قيمة ال Task ID للمهمة الابن حيث ترد القيمة By Reference
  - Cc : القيمة المعادة من التابع PVM_Spawn وتمثل العدد الحقيقي لمهام التي تم إنشاؤها بناء على القيمة السالبة في حال عدم نجاح التابع في إنشاء أي نسخة

- في السطر الثامن يتم اختبار نجاح إنشاء مهمة hello _other حيث إذا تم إنشاء الابن بنجاح سيقوم بنفس اللحظة بعمل run للابنhello _other
- في السطر التاسع :تابع استلام الرسائل PVM recv (int task_id , int msg_tag)
  - Task_id :يحدد id مهمة معينة لاستلام الرسالة منها القيمة 1 تمثل الاستلام من أي مهمة دون الاهتمام بالجهة المرسلة
  - Msg_tag :يحدد tag معين للرسالة لاستقبالها والقيمة -1 تعني قبول أي tag أي يتم استقبال دون الاهتما بال tag الخاص بالرسالة

- التابع pvm_recيبقى ال task قيد الانتظار إلى أن يستلم الرسالة المطلوبة فعند استلام أول رسالة موافقة لشروط ستتوقف عملية الاستلام وينتقل للتعليمة التالية.

- في السطر العاشر التابع PVM-bufinfo وهو تابع قراءة معلومات اللصاقة حيث بارمتراته هي:
  - البارامتر الأول هو دخل عملية الاستقبال حتى يعلم لصاقة أي رسالى مرسلة ستتم قراءتها (بحال استقبال أكثر من رسالة)
  - البارامتر الثاني يعبر عن حجم الرسالة والبارمترات الثالث يعبر عن tag الرسالة
  - البارامتر الرابع يحدد id المهمة التي أرسلت الرسالة

- في السطر الحادي عشر التابع PVM_upkstr وهو التابع المسؤول عن فك تحزيم الرسالة

- في السطر الثاني عشر يتم طباعة قيمة ال id الخاص بالابن المنشا وطباعة محتوى الرسالة التي استلامها المخزن ضمن المتحول buf .

Hello Other

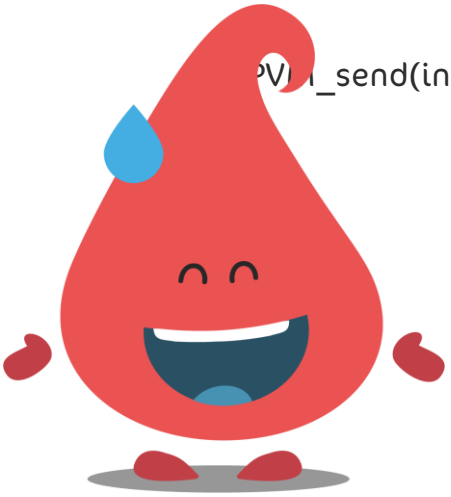
```
1)    #include "pvm3.h"
2)    main() {
3)     int ptid;
4)     char buf[100];
5)     ptid = pvm_parent();
6)     strcpy( buf,"hello , world from ");
7)     gethostname( buf + strlen(buf) , 64 );
8)     pvm_initsend( PvmDataDefault );
9)     pvm_pkstr( buf );
10)    pvm_send(ptid, 1);
11)    pvm_exit();
12)    exit(0); }
```

- في السطر الخامس يتم تسجيل البرنامج ك PVM Task عبر استدعاء التابع PVM_parent والذي يقوم بإعادة الأب وتخزينها في المنحول ptid

- في السطر السادس التابع strcpy() هو تابع للنسخ سلسلة محارف حيث يقوم بنسخ محتويات البارامتر الثاني إلى الأول
- في السطر الرابع التابع grthostname يتم استخدامه لأنه يمكن أن نشغل الأب على جهاز والابن على ..يقوم بتحديد مكان تواجد المؤشر حيث سيقوم بتخزين اسم الجهاز المضيف بعد كلمة from وبطول أعظمي 64 محرف (تم تحديد الطول في البارامتر الثاني)

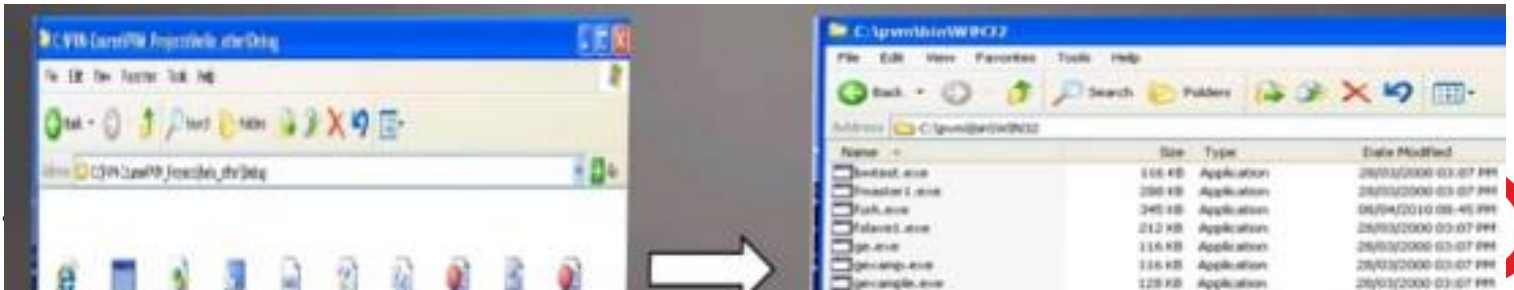
تتم عملية الإرسال وفق الخطوات الثلاث التالية:

- في السطر الثامن تتم تهيئة العرسال عبر التابع PVM_initsend حيث PVMDataDefault هو نمط معياري لإرسال يستخدم لعمل encoding للبيانيات بطريقة تناسب تبادل الرسائل لتقرأ ضمن أي نظام تشغيل
- تستخدم التابع PVM_pkstr() في السطر التاسع لتحزيم البيانيات المراد إرسالها إلى الطرف الآخر (المصفوفة buf)
- في السطر العاشر لدينا تابع إرسال الرسالة (int task_id , nt msg_tag)_send
  - Task_id: يحدد id المهمة المراد إرسال الرسالة لها
  - Msg_tag: يحدد tag مميز للرسالة



Execute The Code

- الخطوة الأولى: نقوم بعمل Build للابن hello other وذلك عبر الضغط على ملف البرنامج واختيار Build
- الخطوة الثانية:نقوم بالتوجه إلى المسار c:\PVM-Cource\PVM Projects-hello othrt\Debuf ونقوم بنسخ الملف التنفيذي للابن hello_other.exe ونضعه ضمن المسار التالي c:\PVM\bin



الهدف من الخطوة السابقة:

عندما يتم عمل spwan بغرض إنشاء ال hello_other عندما يتم البحث عن الملف ...للابن ضمن مسار متفق عليه عوضاً عن البحث عنه ضمن كامل الجهاز

- **الخطوة الثالثة:تقوم بتنفيذ المشروع المشروع الأب** hello **عن طريق الضغط عليه بالزر اليميني وتعيينه كالمشروع ...**(Set as Active Project) **ثم نقوم بعمل** Execute **للمشروع** (Ctrl + F5)