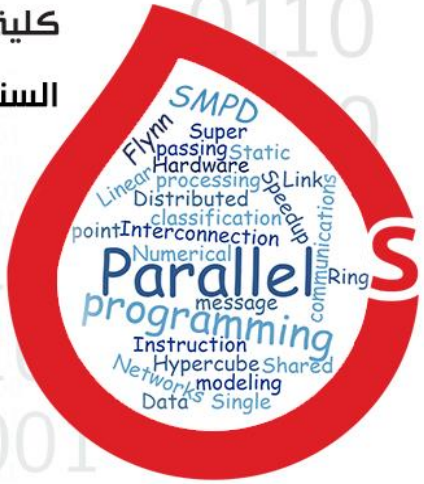
Fork



م. عمار المصري

09/03/2024

RB Informatics;

البرمجة التفرعية

في المحاضرات السابقة قمنا بكتابة برنامجين منفصلين

واحد للأب (hello) وواحد للابن (hello other) كل منهما يقوم بعمله بعيد عن الآخر، أما في هذه المحاضرة فسنرى أنه بالإمكان أن نكتب الأب والابن في برنامج واحد ونتحكم متى سيكون أب ومتى سيكون ابن وبالتالي تكون هيكلية البرنامج if/else وللمتحولات المشتركة.

أي أنّ البرنامج سيعمل كأب وابن في نفس الوقت و؟؟؟ كود الابن فسنضع كود الأب ضمن if وكود الابن سيكون ضمن else والمتحولات التي نحددها بالطرفين نرفعها لكي تصبح مشتركة.

حيث برنامج الأب يشغّل الكود المشترك وكود برنامج الابن يشغل الكود المشترك وكوده

برمجياً:

سيكون لدينا بناء على شرط معين إما سيدخل على الشرط وينفذ الأب وينفذ التعليمة spwan ونقوم بإعطائها اسم الملف نفسه " الأب " حتى تستطيع تحقيق مبدأ عمل fork وبالتالي عند تنفيذ تعليمة إنشاء الابن يتم تشغيل البرنامج عوديا فيجد نفسه الابن فيقوم بتنفيذ كود الابن

```
If( )
{
Father code
}
Else
{
Sons code
}
```

ملاحظات:

أي تابع من توابع ال PVM عندما يقوم برد قيمة أصغر من الصفر (سالبة) فهذا يدل على أن التابع فشل وهنا كمشكلة في تنفيذ التعليمة

في البرنامج لا يتم المرور على الأبناء التي تم إنشاؤها وفحصها

الكود المشترك مع الشرح:

أولا قمنا بتعريف المتحولات في الكود المتشرك

Define MAXCHID20: متحول ثابت يدل على العدد الأكبر للأبناء

Define JOWTAG11: تاغ الرسالة

Ntask=3: عدد المهام المطلوب إنشاؤها

Info: متحول نسند إليه عند المهام المنشأة

Mytid: id المهمة

myParent: id الأب

Child: مصفوفة لتخزين الأبناء

Mydata,bof,len,tag,tid: متحوات ستعرف في كود الأب.

PVM-mytid: تجلب id المهمة الحالية في حال كانت سالبة => error

هل من الأفضل تخزين متحولات الأب في القسم الخاص بالأب ومتحولات الابن في القسم الخاص بمتحولات الابن أم نضعها في القسم المشترك للبرنامجين

نضع المتحولات في القسم المشترك لعدم وجود تضارب لأن address space لكل task معزولة عن باقي tasks.

ضمن الشروط السابقة:

إن التابع mytid= pvm_mytid() هو تابعيرد قيمة ال *id* للمهمة الحالية (سواء كان أب أو ابن) فإن كانت القيمة الصادرة عن هذا التابع *mytid<()* فهذا يدل على وجود خطأ في تسجيل ال *PVM Task* وعندها يتم طباعة رسالة خطأ ومغادرة البرنامج.

إن تابع *myparent= pvm_parent()* هو تابع يرد قيمة ال *id* للأب الخاص بالمهمة الحالية فإن كانت القيمة الصادرة عن هذا التابع *myparent<()* فهذا يعني أن العملية فشلت ولا يوجد أب للمهمة الحالية (أي أنها المهمة الحالية لمثل الأب)

لدينا المتحولات *(PvmParentNotSet,PvmNotParent)* وهي متحولات معرفة مسبقا ضمن ال *PVM* وتشير إلى وجود الأب للمهمة الحالية أم لا والقيمة الخاصة بهذه المتحولات هي *id* الأب.

إذا كانت قيمة المتحول *myparent* مساوية للقيمة *PvmParentNotSet* أو القيمة *PvmNotParent* فهذا يعني أن المهمة الحالية لا تملك أب، وبالتالي فهي تمثل الأب عندها تنتقل إلى تنفيذ الجزء البرمجي الخاص بالأب وإلا فننتقل إلى تنفيذ لجزء الخاص بالابن

ملاحظة:

PvmParentNotSet && PvmNotParent متحولات محجوزة في الذاكرة بال *PVM* قيمتها الرئيسية مؤشرات ل *taskID* الخاص بالأب ضمن *memory*

يقوم التابع *pvm_spawn* بإنشاء الأبناء حيث اسم البرنامج الذي سينفذه هو ذات البرنامج *fork* عدد المهما التي سيتم إنشاؤها تساوي قيمة المتحول *ntask* أي سيتم إنشاء ثلاث أبناء *child* هي مصفوفة المؤشرات التي سيتم بها تخزين ال *id's* للأبناء الذين تم إنشاؤهم بنجاح

لدينا حلقة *for* نمر بها على عدد الأبناء حيث عندما *child<0* فالابن لم يتم إنشاؤه وسيطبع *id* بيمة سالبة

إذا كانت قيمة المتحول *info=0* وهو المتحول الذي يدل على عدد الأبناء التي تم إنشاؤها بنجاح فسيتم الخروج من البرنامج

سيتم استخدام حلقة *for* لاستقبال الرسائل من كافة الأبناء التي تم إنشاؤها

التابع *pvm_recv()* يقوم باستقبال الرسالة من الابن خرج التابع هي القيمة *buf* فإذا كانت القيمة سالبة *(buf<0)* عندها فإن عملية الاستقبال قد فشلت ويتم طباعة رسالة خطأ للمستخدم

التابع *pvm_bufinfo()* يقوم بقراءة معلومات ال *buffer* خرج التابع هو المتحول *info* فإذا كانت هذه القيمة سالبة *info<0* عندها فإن عملية فك التحزيم قد فشلت ويتم طباعة رسالة خطأ لمستخدم

أخيرا يتم طباعة المعلومات الرسالة من أجل كل ابن

متى يأخذ ال *Buffer* قيمة أقل من الصفر(قيمة سالبة)
انطلاقا من نقطة أنه عندما ينفذ التابع *PVM-recv()* يدخل الأب في حالة *Blocking* لينظر أول ابن ليرسل أول رسالة
فيكون لدينا ضمن مفهوم ال *PVM* أنن الرسالة إما أن:
تصل لرسالة بشكل صحيح
الرسالة لم تصل أبدا

ولا يكون لدينا حالة أن تصل نصف الرسالة صحيح والآخر لا يصل وبالتالي يأخذ ال *Buffer* قيمة أقل من الصفر عندما لا يصل بشكل صحيح

في حالة إذا وصلت الرسلة بشكل صحيح "قيمة ال *Buffer* موجب" وتكون البارامترات الخاصة *PVM-bufinfo()*

خرج عملية *receive*

حجم المعطيات

Tag الرسالة

TaskID لمرسل الرسالة

خرج التابع هو المتغير *Info*

خطوات تنفيذ كود الابن

PVM-initsend تهيئة الإرسال

PVM-Pkint يقوم بتحزيم الرسالة المراد إرسالها إلى الأب (قيمة *iD* الخاص بالابن)

()PVM-send يقوم بإرسال الرسالة حيث يتم تحديد *id* خاص بالأب وقيمة *tag* الرسالة إذا كنا نريد إرسال عنصر واحد فيكون البارامتر الثاني والثالث دائما هو واحد في التحزيم => أي قيمة *(1)*