Chapitre 02 : *Le système d'interruption*

1. Concept de processus

La notion de processus constitue un modèle simple et efficace pour représenter l'exécution concurrente de tâches au sein d'un système d'exploitation multitâches.

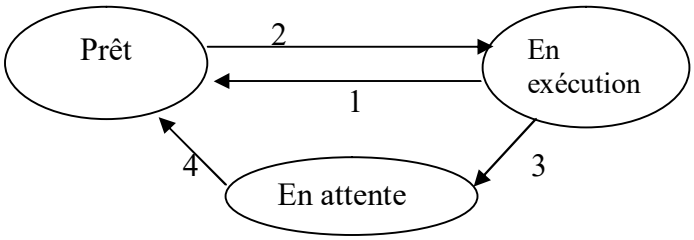
1.1. Définitions :

- un processus représente l'exécution d'un programme comportant des instructions et des séquences. C'est une entité dynamique (active) créée à un instant donné, qui disparaît en général au bout d'un temps fini.
- un processus est un programme en cours d'exécution, composé de : code, données, pile d'exécution, un compteur ordinal et autres informations caractérisant son état.

Différence entre processus et programme : le programme est une description statique (entité passive) ; le processus est une activité dynamique (il y a un début, un déroulement et un fin, il y a un état qui évolue au cours du temps).
Exemple de processus : Copier un fichier sur disque, Impression la fiche de paie

1.2. États d'un processus

Durant son séjour dans un SE, un processus peut changer d'état à plusieurs reprises. Ces états peuvent être résumés dans le schéma suivant :



Explication de chacun de ces états :

- Prêt : le processus est placé dans la file d'attente des processus prêts, en attente d'affectation du processeur.
- En exécution (Élu ou Actif) : le processus a été affecté à un processeur libre. Ses instructions sont en exécution.
- En attente (Bloqué): Le processus attend qu'un événement se produise, comme l'achèvement d'une opération d'E/S ou la réception d'un signal.

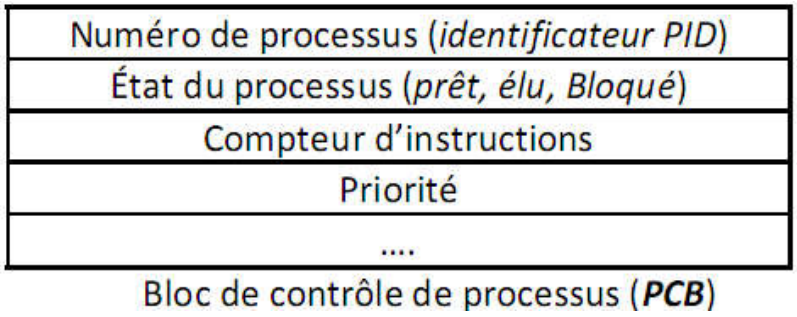
La signification des quatre transitions numérotées est la suivante :

1. Le processus a épuisé le quantum de temps qui lui a été attribué. L'ordonnanceur *(L'ordonnanceur est la partie de SE qui contrôle l'attribution de processeur aux processus)* élit un autre processus parmi les processus prêts,
2. L'ordonnanceur élit ce processus parmi les processus prêts,
3. Blocage du processus élu dans l'attente d'un évènement (E/S ou autre)
4. Réveil du processus bloqué après disponibilité de l'événement bloquant

1.3 Bloc de contrôle de processus :

La gestion des processus impose au système d'exploitation de garder trace de tout processus quel que soit son état. Pour cela, le S.E regroupe toutes les informations relatives à un processus dans une structure de données appelée « Descripteur de Processus » ou « Bloc de Contrôle de Processus » (PCB : Process Control Block). A la création d'un processus, le S.E crée son PCB correspondant qui l'identifiera durant sa durée de vie dans le système.

Le PCB peut être vu comme un enregistrement représenté par la figure suivante :



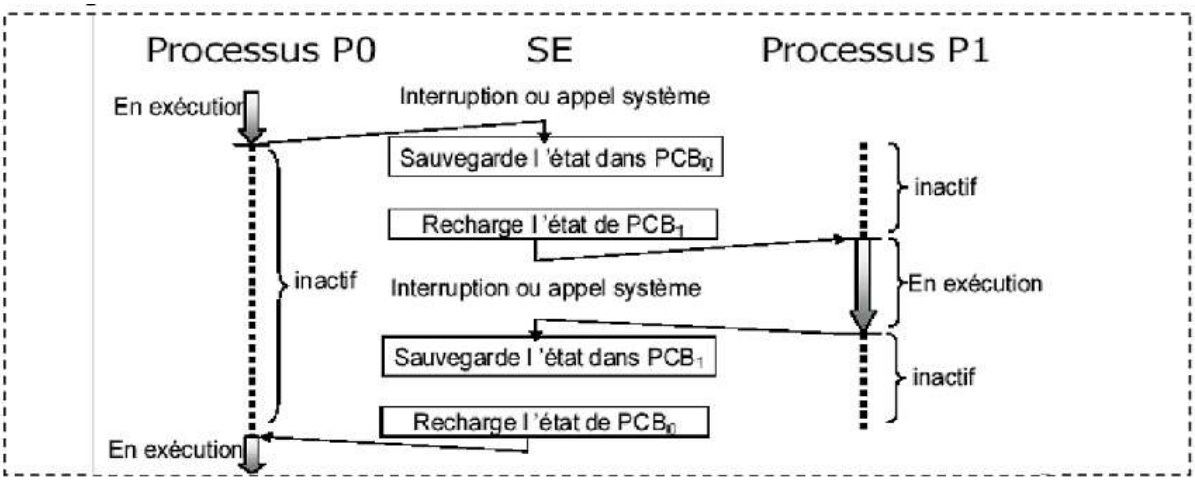
Bloc de contrôle de processus (***PCB***)

NB : Le compteur d'instructions indique l'adresse de la prochaine instruction à exécuter par le processus.

Le noyau de SE maintient une table pour gérer l'ensemble des processus. Cette table contient la liste de tous les processus avec des informations concernant chaque processus.

1.4 Changement de contexte :

Le passage dans l'exécution d'un processus à un autre nécessite une opération de sauvegarde de contexte du ***processus bloqué***, et de chargement de celui du ***nouveau processus***. En effet, le noyau doit arrêter l'exécution du processus en cours, copier les valeurs des registres dans le PCB, et mettre à jour les registres avec les valeurs du nouveau processus. Le schéma suivant montre les étapes de cette opération de changement de contexte.



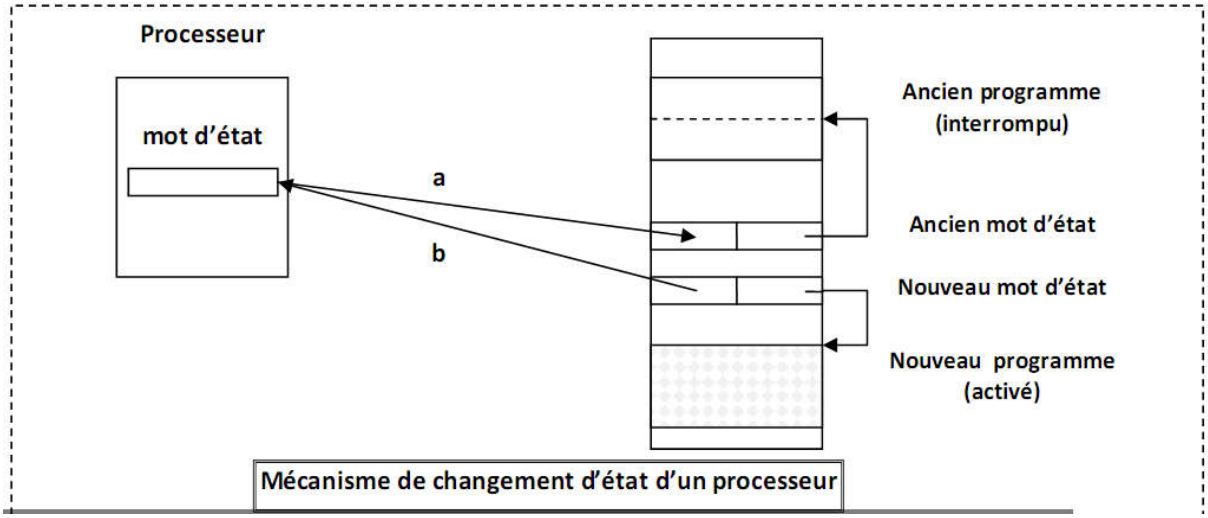
2. Les systèmes d'interruption

Le système d'interruption est la partie de système d'exploitation qui détecte et prendre en charge les interruptions (signaux matérielles et /ou appels systèmes).

2.1 Mécanisme de changement de mot d'état :

Dans une entité logique (généralement un mot), le SE regroupe des informations clés sur le fonctionnement du processeur : c'est le mot (registre) d'état du processus (Processus Status Word, PSW). Ce dernier comporte généralement, la valeur du compteur ordinal, des informations sur les interruptions (masquées ou non), le privilège du processeur (mode maître ou esclave),…etc. Le mécanisme de changement d'état permet en une seule opération indivisible de :

a - Ranger dans un emplacement spécifié de la mémoire, le contenu courant de mot d'état du Processus (PSW).
b - Charger dans le mot d'état un nouveau contenu préparé à l'avance dans un emplacement spécifié de la mémoire.



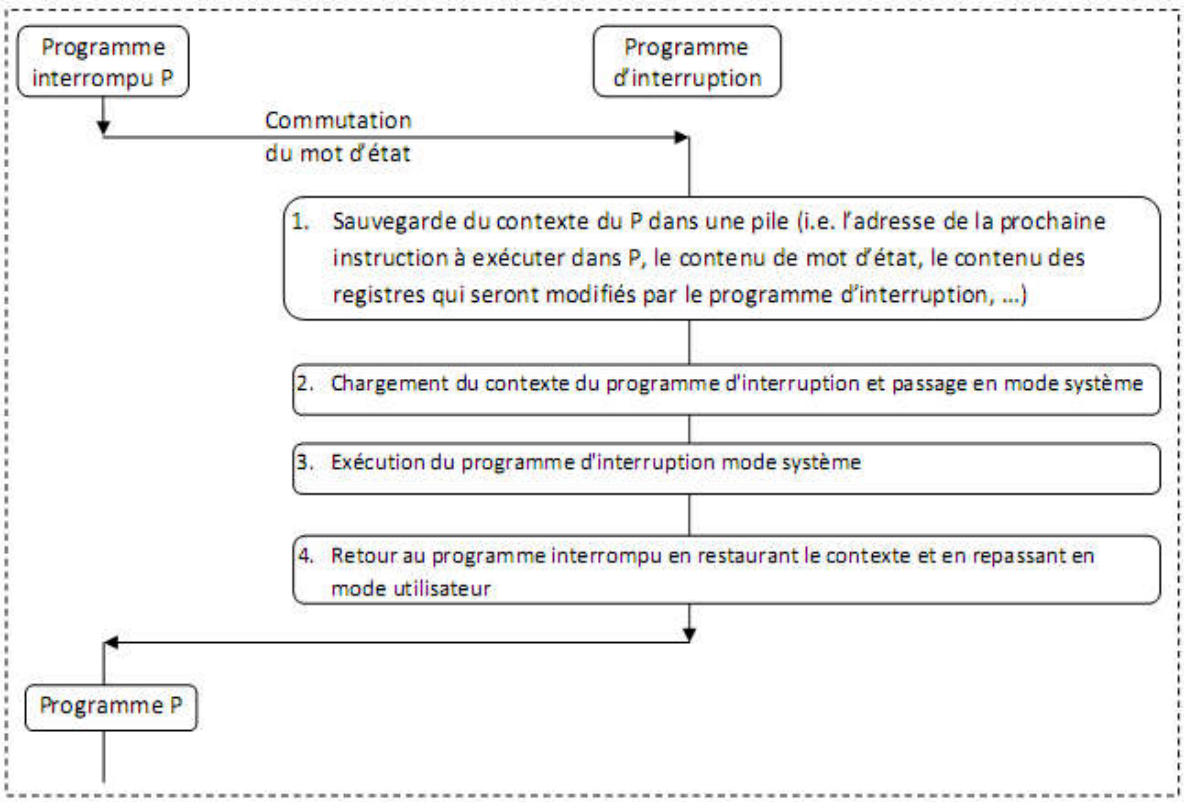
Remarque :

- Ce changement d'état n'est possible que si le processus se trouve dans un état bien défini s'appelle un point observable (point interruptible), c'est dire à un instant séparant l'exécution de deux instructions consécutives.
- Le changement d'état est déclenché suivant l'état d'un indicateur consulté par le processeur après l'exécution de chaque instruction.

2.2 Définition (Interruption) :

C'est une ***commutation de mot d'état*** provoqué par un ***évènement*** qui peut être ***interne*** au ***processus*** et résultant de son exécution, ou ***extérieur*** et indépendant de cette exécution. En effet, l'interruption permet de forcer un processeur à suspendre l'exécution du programme (processus) en cours et à exécuter un programme spécifié : c'est ***le programme d'interruption.***

2.3 Schéma général d'un programme d'interruption :



Une interruption a pour fonction de forcer un processeur à réagir à un évènement. L'exécution du programme en cours est suspendue et, un programme d'interruption est exécuté. Le programme interrompu repris ensuite son exécution.

2.4 Types d'interruptions :

Les interruptions peuvent être classées en deux classes :

- ***Interruptions logicielles*** : Elles sont appelées par une instruction à l'intérieur d'un programme. Ces interruptions ont une fonction définie, par exemple la lecture et l'écriture sur le disque, l'écriture des données à l'écran, etc.

Exemple (Programme Pascal):

Fonction DOS (Fct 09h) : permet d'afficher, par appel (interruption) système de fonctin MsDos, un message sur l'écran.

```
Program DOSWrite ;
Uses DOS ;
Var  regs : registers ; Message : string[100] ;
Begin
  Message := 'Ce message est affiché par la primitive DOS' + '$' ;
  Regs.AH := $09 ;
  Regs.DS := seg(Message[1]) ;   Regs.DX := ofs(Message[1]) ;
  MsDos( Regs ) ;
End.
```

- ***Interruptions matérielles*** : Elles sont déclenchées par une unité électronique (lecteur, clavier, contrôleur de périphérique, panne de courant,....) ou par l'horloge.

2.4 Priorité et masquage des interruptions

A chaque interruption, est associée une priorité (système d'interruptions hiérarchisées) qui permet de regrouper les interruptions en classes.

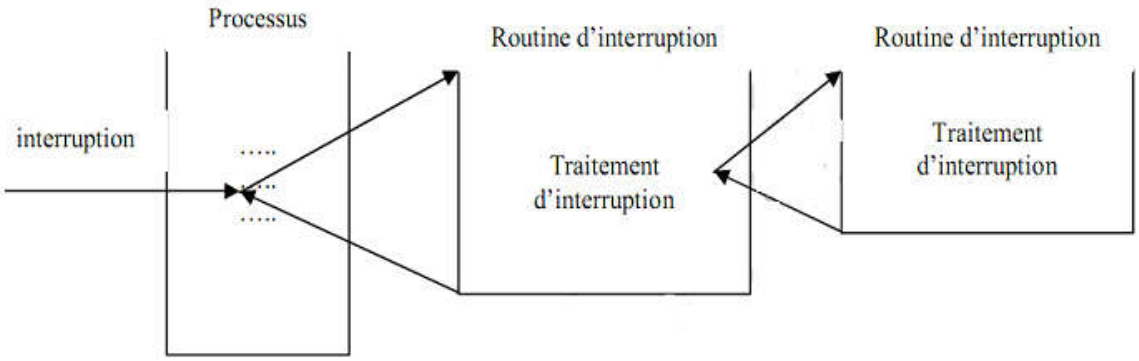
Règle : Une interruption de priorité **j** est plus prioritaire qu'une interruption de niveau **i** si **j > i**.

L'intérêt de ce système est la solution de problèmes tels que :

- Arrivée de plusieurs signaux d'interruption pendant l'exécution d'une instruction.

- Arrivée d'un signal d'interruption pendant l'exécution du signal de traitement d'une interruption précédente.

Selon cette hiérarchie de priorité, on peut envisager une interruption en cascade dans le cas où la routine d'interruption elle-même peut être interrompue par une interruption plus prioritaire.



Interruptions en cascade

Il est possible à un processus superviseur de masquer une interruption. Masquer (inhiber) une interruption revient à retarder son effet temporairement. Elle peut être démasquée après. À un instant donné, les interruptions sont soit masquées soit autorisées, suivant l'état d'un indicateur spécial du registre d'état, IF (Interrupt Flag).

- Si IF = 1 (interruptions autorisées), alors le processeur accepte les demandes d'interruptions masquables, c'est-à-dire qu'il les traite immédiatement ;

- si IF = 0 (interruptions masquées), alors le processeur ignore ces interruptions.

L'état de l'indicateur IF peut être modifié à l'aide de deux instructions, CLI (CLear IF pour la mise à 0 de IF), et STI (SeT IF, pour la mise à 1 de IF).

2.5. Déroutement : Quand un indicateur de changement d'état est modifié par une cause liée à l'exécution d'une instruction en cours, on dit qu'il y a déroutement (trap). Le mécanisme de déroutement a essentiellement pour rôle de traiter une anomalie dans le déroulement d'une instruction :

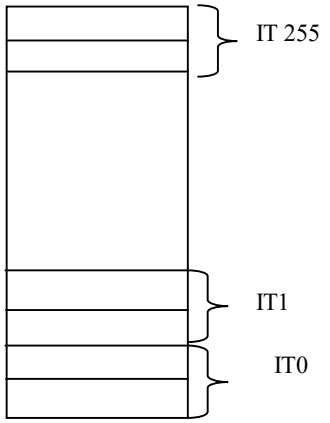
- Données incorrectes : division par zéro, …

- Tentative d'exécution d'une opération interdite par un dispositif de protection (violation de la mémoire,…).

3. Le système d'interruptions dans les PCs :

Les programmes associés aux interruptions sont appelés les routines d'interruptions. Cette routine est un programme qui peut accéder à toutes les instructions du langage machine du processeur. La fin de cette routine est signalée par l'instruction IRET (Interrupt RETurn) qui ordonne au processeur de reprendre l'exécution du programme interrompu exactement là où il avait été abandonné.

3.1 Le vecteur d'interruption : Le PC comprend 256 causes d'interruption numérotés de 0 à 255. Chaque interruption correspond à une routine d'interruption qui sera exécutée lors de l'appel d'interruption correspondante. Le vecteur d'interruption fait le lien entre l'interrutpion (cause) et la routine d'interruption. Cest une table qui contient 256 entrées ; une pour chaque interruption.

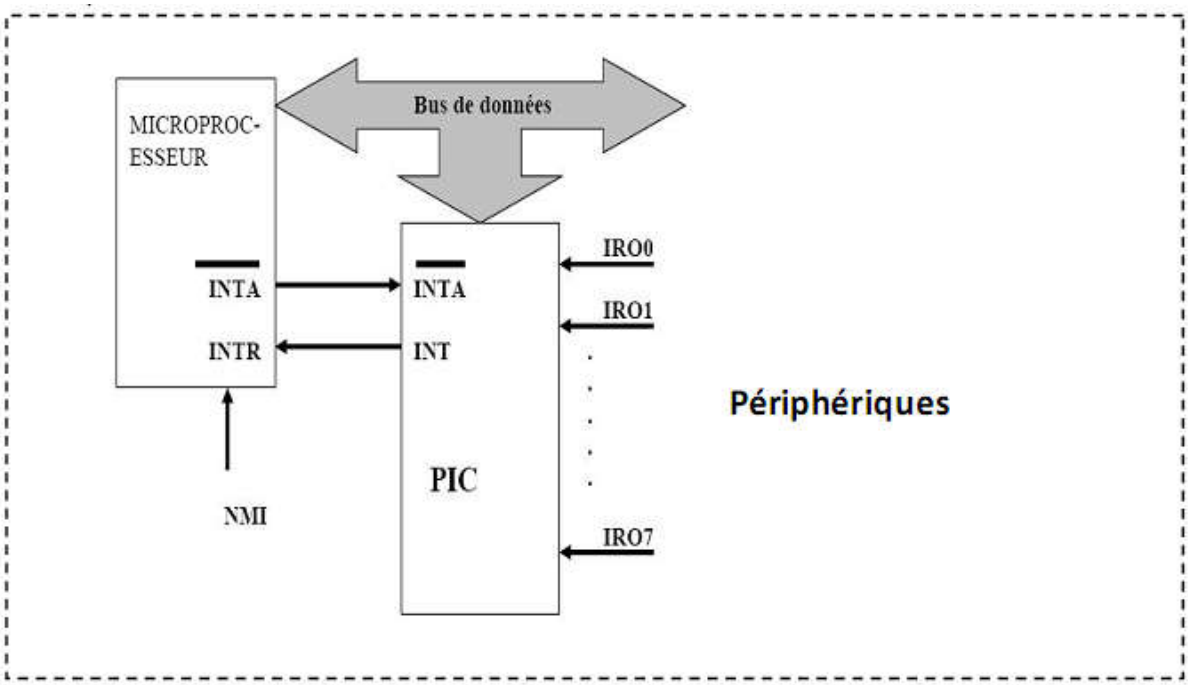


Chaque entrée désigne l'adresse début de la routine d'interruption correspondante en mémoire. Quand une interruption est appelée ou provoquée, le processeur va automatiquement chercher l'adresse début de la routine d'interruption dans le vecteur d'interruption et il commence l'exécution de la routine.

Exemple : l'adresse de la routine d'interruption 0 figure dans l'entrée 0 de la table i.e. 0h à 3h.

Remarque : pour calculer la cellule mémoire dans laquelle est stockée l'adresse début de la routine d'interruption, il suffit de multiplier le numéro de l'interruption par 4.

3.1 Contrôleur d'interruption (PIC : Programmable Interruption Controler):
L'ordinateur est relié à plusieurs périphériques, mais nous venons de voir qu'il n'y avait qu'un seul signal de demande d'interruption, à savoir INTR. Le contrôleur d'interruptions est un circuit spécial, extérieur au processeur, dont le rôle est de distribuer et de mettre en attente les demandes d'interruptions provenant des différents périphériques
La figure suivante indique les connexions entre le microprocesseur et le contrôleur d'interruptions.



Le contrôleur est relié aux interfaces gérant les périphériques par les bornes IRQ (InteRrupt reQuest). Il gère les demandes d'interruption envoyées par les périphériques, de façon à les envoyer une par une au processeur (via INTR). Il est possible de programmer le contrôleur pour affecter des priorités différentes à chaque périphérique. Avant d'envoyer l'interruption suivante, le contrôleur attend d'avoir reçu le signal $\overline{\text{INTA}}$, indiquant que le processeur a bien traité l'interruption en cours.

Reprenons les différents évènements liés à la réception et le traitement d'une interruption masquable :

1. Un signal INT est émis par un périphérique (ou plutôt par l'interface gérant celui-ci).

2. Le contrôleur d'interruptions reçoit ce signal sur une de ses bornes IRQi. Dès que cela est possible (suivant les autres interruptions en attente de traitement), le contrôleur envoie un signal au processeur sur sa borne INT.

3. Le processeur prend en compte le signal sur sa borne INTR après avoir achevé l'exécution de l'instruction en cours (ce qui peut prendre quelques cycles d'horloge). Si l'indicateur IF=0, le signal est ignoré, sinon, la demande d'interruption est acceptée.

4. Si la demande est acceptée, le processeur met sa sortie $\overline{\text{INTA}}$ au niveau 0 pendant deux cycles d'horloge, pour indiquer au contrôleur qu'il prend en compte sa demande.

5. En réponse, le contrôleur d'interruption place le numéro de l'interruption associé à la borne IRQi sur le bus de données.

6. Le processeur lit le numéro de l'interruption sur le bus de données et l'utilise pour trouver le vecteur d'interruption (afin de traiter l'interruption). Ensuite, tout se passe comme pour un appel système, c'est-à-dire que le processeur :

- Sauvegarde les indicateurs du registre d'état sur la pile (en d'autres termes il conserve l'état de la mémoire concernant le programme en cours d'exécution) ;

- Met l'indicateur IF à 0 pour masquer les interruptions suivantes ;

- Cherche dans la table des vecteurs d'interruptions l'adresse du traitant d'interruption.

7. La procédure traitant l'interruption se déroule. Pendant ce temps, les interruptions sont masquées (IF=0).

8. La procédure se termine par l'instruction IRET qui permet de reprendre le programme qui avait été interrompu.

3.1 Appel des interruptions à partir du langage C :

Toutes les interruptions ainsi que les structures de données correspondantes sont déclarées dans le fichier dos.h.

Exemple d'interruptions :

N° interruption	cause
0	division par zéro
1	CPU pas à pas
9	clavier
19	reset du système (ALT + Ctrl+ Del)
20h	fin normale d'un programme