Université Yahia Farès de Médéa
Département: GEI
2ème année Informatique (L.M.D)
Option : RSI/SIIA

Les Systèmes d'Interruption

A. KHELDOUN
Émail: univ.medea@gmail.com

Année Universitaire: 2013-2014

Plan

- **Concept de processus**
  - ❑ Définition
  - ❑ États d'un processus
  - ❑ Bloc de contrôle de processus (PCB)
  - ❑ Processus sous UNIX (fork())
  - ❑ Changement de contexte d'un processus
- **Le système d'Interruption**

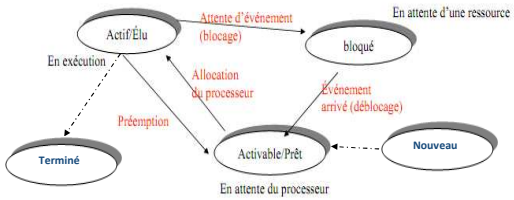
Concept de processus

❑ **Définition**

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.

Processus = instructions +ressources

❑ **États d'un processus**



Concept de processus

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

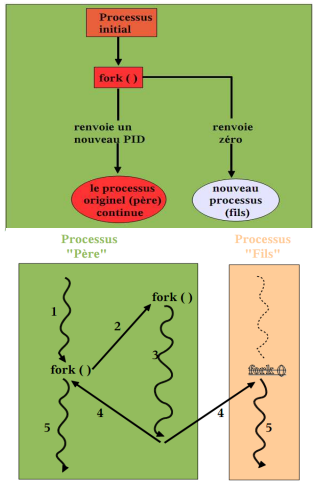
- ➢ **Objectif**: Permettre au SE de garder trace de tous les processus
- ➢ À chaque création d'un processus, le SE lui associé un PCB
- ➢ **Contenu**:
  - ✓ Numéro de processus (***identificateur PID***),
  - ✓ État du processus (***prêt, élu, Bloqué***),
  - ✓ Compteur d'instructions,
  - ✓ Priorité
  - ✓ … etc.

Concept de processus

❑ **Processus sous UNIX : fork()**

➢ Cet appel système permet de créer dynamiquement (en cours d'exécution) un nouveau processus qui s'exécute de façon concurrente avec le processus qui l'a créé.

➢ Le nouveau processus exécute donc le même code que le processus parent et démarre comme lui au retour du fork().



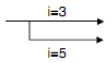
Concept de processus

1 programme, 2 processus
donc 2 mémoires virtuelles, 2 jeux de données

testfork.c

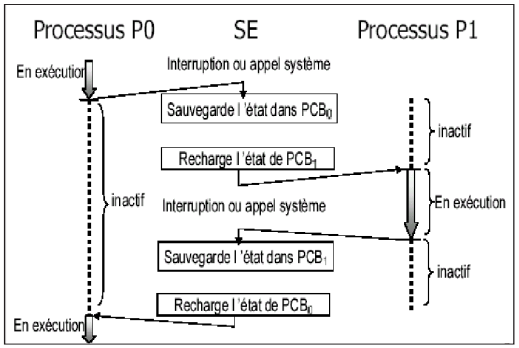
```
int i ;
if (fork() != 0) {
    printf("je suis le père, mon PID est %d\n", getpid());
    i = 3;
  } else {
    printf("je suis le fils, mon PID est %d\n", getpid());
    i = 5;
  }
printf("pour %d, i = %d\n", getpid(), i);
}
```

```
>gcc -o testfork testfork.c
>./testfork
je suis le fils, mon PID est 10271
pour 10271, i = 5
je suis le père, mon PID est 10270
pour 10270, i = 3
```



Concept de processus

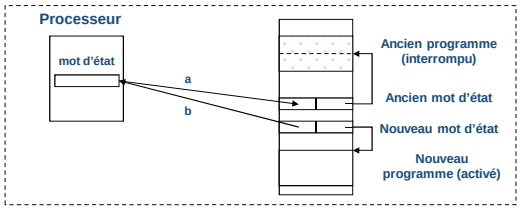
❑ **Changement de contexte d'un processus**



Le système d'Interruption

❑ **Mécanisme de changement d'état : Opération indivisible qui permet de:**

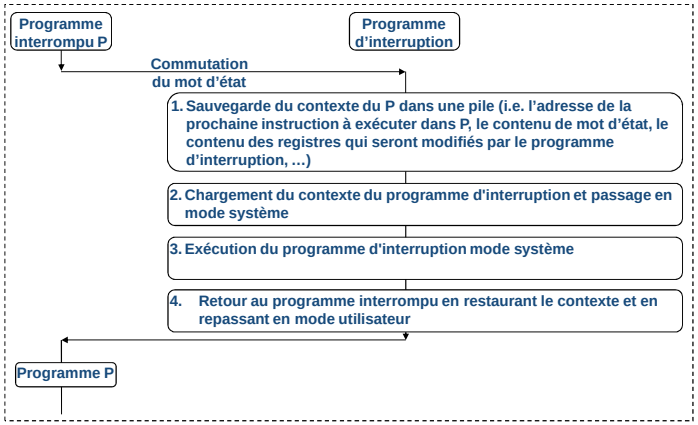
a. Ranger dans un emplacement spécifié de la mémoire, le contenu courant de mot d'état du Processus (PSW- *Processus Status Word*).
b. Charger dans le mot d'état un nouveau contenu préparé à l'avance dans un emplacement spécifié de la mémoire



Une Interruption 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.

Le système d'Interruption

❑ Schéma général d'un programme d'interruption



Le système d'Interruption

❑ Types d'interruptions

➢ Interruptions logicielles

✓ Appelées par une instruction à l'intérieur d'un programme

✓ Exemple: Lecture/Écriture sur le disque, etc.

```
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

✓ déclenchées par une unité électronique (lecteur, clavier, panne de courant,....) ou par l'horloge.

Le système d'Interruption

❑ Déroutement (*Trap*)

➢ Interruption logicielle

➢ Traiter une anomalie d'exécution d'une instruction

✓ Données incorrectes : division par zéro, …

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

❑ Masquage des interruptions

➢ Retarder l'effet d'une interruption temporairement

➢ Indiquer par un indicateur (*registre d'état*- **Interrupt Flag (IF)**)

✓ **IF = 1** ***(interruptions autorisées)***

✓ **IF = 0** ***(interruptions masquées)***

➢ ***CLI : IF=0 et STI: IF=1***

Le système d'Interruption

❑ Le système d'interruptions dans les PCs

➢ Le vecteur d'interruption

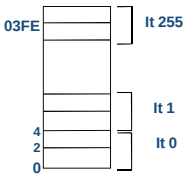
✓ Table de 256 entrées

→ 256 causes numérotées de 0-255

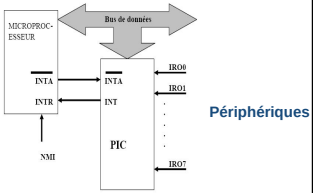
✓ Lien entre l'interruption (***cause***) et la routine d'interruption

→ Chaque entrée désigne l'adresse début de la routine d'interruption

→ Ex: entrée 0 (0h à 3h) : adresse de la routine d'interruption 0.



➢ **Contrôleur d'interruption** (PIC : *Programmable Interruption Controler*)



Le système d'Interruption

❑ **Contrôleur d'interruption**

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** puis il 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.
   - IF= 0: Signal ignoré, sinon la demande acceptée
4. Le processeur met sa sortie **INTA** au niveau 0 pendant deux cycles d'horloge
   → indiquer au contrôleur qu'il prend en compte sa demande
5. Le contrôleur d'interruption place le numéro de l'interruption associé à la borne **IRQi** sur le bus de données.

Le système d'Interruption

❑ **Contrôleur d'interruption**

6. Le processeur lit le numéro de l'interruption sur le bus de données et l'utilise pour trouver le vecteur d'interruption. Ensuite:
   - ✓ Sauvegarder les indicateurs du registre d'état sur la pile
   - ✓ Mettre **IF** à **0** pour masquer les interruptions suivantes
   - ✓ Chercher dans la table des vecteurs d'interruptions l'adresse du traitant d'interruption
7. Reprendre 'exécution du programme interrompu (**IRET**).