

Réalisation des processus et des mécanismes de synchronisation

Plan du chapitre

- Machine à processus
- Les sémaphores
- Réalisation de l'exclusion mutuelle
- Fonction et structure d'un noyau de synchronisation. Réalisation du noyau

Rappels : un processus =

- Un espace mémoire : code, données, pile
 - On suppose ici que programme et données sont chargés en mémoire
- Un mot d'état de programme (compteur ordinal + registre d'état)
- Des (images des) registres généraux

Activation - Arrêt

- Lancer l'exécution d'un processus
 - Charger le MEP et les registres
- Arrêter un processus
 - Ranger les valeurs courantes du MEP et des registres
- Table des processus, pid
 - Contient les valeurs des registres des différents processus
- Comment passer d'un processus à un autre ?

Passage de l'unité centrale d'un processus à un autre

- Attendre le blocage ou la terminaison du processus qui s'exécute
- Provoquer le changement
 - Ranger les registres physiques dans la table des processus
 - Choisir un nouveau processus à exécuter
 - Charger les registres physiques à partir de la table des processus
- Comment forcer l'exécution de ces trois étapes ?
- Utilisation d'une horloge externe et du mécanisme des interruptions

Le principe des interruptions

- Mécanisme câblé
- Déclenché par un événement extérieur à l'UC
- Sauvegarde du mot d'état de programme (dans une pile ou un emplacement fixe)
- Charge le mot d'état de programme par une valeur associée à la nature de l'événement externe
 - a pour effet de lancer l'exécution d'un programme associé à l'événement : le traitant de l'événement
- Possibilité de différer la prise en compte d'un événement (masquage)

Utilisation des interruptions pour la gestion des processus

- Existence d'une horloge
 - Dispositif qui permet de décompter un intervalle de temps
 - Qui envoie une interruption à la fin de cet intervalle
- Le traitement de l'interruption horloge a pour rôle l'exécution des trois opérations mentionnées plus haut
 - Rangement des registres
 - Choix d'un nouveau processus
 - Lancement du nouveau processus

Quelques remarques sur les interruptions

- Mécanisme plus général que celui décrit ici
 - Entrées sorties
 - Appels systèmes
 - Traitement des erreurs à l'exécution
- Il joue un rôle pour le passage du mode esclave au mode maître
- On l'étudiera de façon plus complète au chapitre suivant

Plan du chapitre

- Rappels
- Machine à processus
- **Les sémaphores**
- Réalisation de l'exclusion mutuelle
- Fonction et structure d'un noyau de synchronisation. Réalisation du noyau

Sémaphore : définition

- Un sémaphore s est l'ensemble d'un compteur entier noté $s.c$ et d'une file d'attente $s.f$
- Deux opérations qui sont exécutées en exclusion mutuelle
- $P(s) \ s.c--$; si $s.c < 0$ alors le processus qui exécute P est bloqué dans $s.f$
- $V(s) \ s.c++$; si $s.c \leq 0$ alors activer un processus de $s.f$

Propriétés des sémaphores

- Soit c_0 la valeur initiale d'un sémaphore s , $c_0 \geq 0$
- Si la valeur $s.c > 0$, elle représente le nb de processus qui peuvent exécuter P sans se bloquer (en l'absence de tout nouveau V)
- Si la valeur $s.c < 0$, elle représente le nombre de processus de $s.f$

Lancement d'un processus

s sémaphore de v , i , 0

Processus P1	Processus P2
Calcul ;	$P(s)$;
$V(s)$;	Traitement ;

Rendez vous de deux processus

s1 et s2 deux sémaphores de valeur initiale 0

```
V(s2);      V(s1);  
P(s1);      P(s2);
```

Exclusion mutuelle par sémaphore

```
Sémaphore mutex vi :1  
P(mutex);  
Section critique  
V(mutex);
```

Interblocage

```
mutex1,mutex2 valeur initiale 1  
  
P(mutex1); P(mutex2);  
P(mutex2); P(mutex1);  
V(mutex1); V(mutex1);  
V(mutex2); V(mutex2);
```

Equivalence des moniteurs et des sémaphores

- La réalisation d'un sémaphore en termes de moniteur est triviale
- La réalisation inverse (implantation de moniteurs à l'aide de sémaphores) est plus délicate.
 - Laissée en exercice
 - Sera vue en TD

Plan du chapitre

- Rappels
- Machine à processus
- Les sémaphores
- **Réalisation de l'exclusion mutuelle**
- Fonction et structure d'un noyau de synchronisation. Réalisation du noyau

Réalisation de l'exclusion mutuelle

- Définitions
- Par masquage des interruptions
- Par attente active
- Cas du multiprocesseur : test & set

Exclusion mutuelle : définitions

- Soit un ensemble de processus, chacun traversant une **section critique**(SC)
- Un processus au plus en section critique
- Aucune hypothèse sur les vitesses
- Un processus bouclant hors SC ne doit rien perturber
- Pas d'attente infinie à l'entrée de SC

Masquage des interruptions

- Cas d'un système monoprocesseur
- Quand peut avoir lieu le passage d'un processus à un autre ?
- Lors d'une interruption
- Solution : toute section critique doit s'exécuter interruptions masquées

Attente active

- Opérations indivisibles :
 - Lecture d'un mot de mémoire
 - Écriture d'un mot de mémoire
- X++ se décompose en
$$R=X$$
$$R=R+1$$
$$X=R$$
(R étant un registre propre au processus)

Tentative de solution

tantque $x \neq 0$ faire ;
x=1;

Section critique
x=0 ;
Il existe une solution (algorithme de Dekker Peterson)

Algorithme de Dekker-Peterson

Soit moi et lui les pid des 2 processus
intention[moi] := vrai ;
tour := moi ;
tantque (tour =moi) et intention[lui] faire ;
section critique ;
intention[moi] := faux ;

Test and set

- Cas d'un système multiprocesseur
- Masquage insuffisant
- Solution de Peterson inefficace
- Instruction spéciale
 - Test&Set adr réalise de façon indivisible le positionnement d'un indicateur selon la valeur initiale de adr
 - la mise à 1 de adr

Solution avec Test and Set

Soit A de valeur initiale 0

```
Entrée  E: Test&Set  A
        jnz  E
```

section critique

```
Sortie  move #0,A
```

Plan du chapitre

- Rappels
- Machine à processus
- Les sémaphores
- Réalisation de l'exclusion mutuelle
- **Fonction et structure d'un noyau de synchronisation. Réalisation du noyau**

Le défi

- Objectif : description du fonctionnement d'un système complet appelé PedagogOS
- En quelques pages (10-20 ?)
- Sans détails techniques de programmation
- Sans cacher les difficultés

Généralités

- Machine fictive POP 1 introduite au fur et à mesure des besoins
- Quel langage utiliser ?
 - Pseudo C
 - Pas de types
- La machine et les notations seront utilisées dans tout le cours

La machine POP 1

Petit Ordinateur Personnel
UC, mémoire, clavier, écran, disque
Monoprocasseur
reg registres généraux
mep mot d'état de programme
Modes maître et esclave

L'Horloge

- Un emplacement de mémoire noté **timer**
- Mise en route de l'horloge = chargement dans **timer** d'une valeur positive
- Arrêt de l'horloge = **raz timer**
- Décrémenté chaque milliseconde
- Déclenche une interruption au passage de 1 à 0

Les interruptions

- Vecteur d'interruption
 - Contient les mots d'état des traitants
- Sauve le `mep` dans un emplacement fixe noté `ancien_mep`
- Charge dans `mep` l'entrée du vecteur d'interruption qui correspond à la cause de l'interruption
- Instruction de retour d'interruption `rti`
- Masquage-démasquage
- Appel système

Noyau de processus

- États des processus, files d'attente
- Table des processus
- Table des sémaphores
- Allocation de l'unité centrale
- Réalisation des primitives

États des processus

- Élu ce processus s'exécute
- Éligible attente de l'UC
- Bloqué attente dans une file de sémaphore

- Transitions entre états

Table des processus

- Pour chaque processus, on conserve un contexte
 - Registres généraux reg
 - Mot d'état psw
 - Indicateurs etat
 - Priorité prio
 - Liens de chaînage
- Chaque processus est identifié par un pid
- Table des processus pid->reg, pid->prio

Table des processus

```
struct descripteur_de_processus {  
    reg ; //registres généraux  
    psw ; // mot d'état de programme  
    etat ; // élu, éligible ou bloqué  
    prio ; // priorité  
    liens ;  
}
```

Table des sémaphores

- Pour chaque sémaphore
 - Valeur du sémaphore cpt
 - Pointeur vers la file d'attente file
- Chaque sémaphore est identifié par un sid
- Table des sémaphores sid->cpt, sid->file

Table des sémaphores

```
struct descripteur_de_semaphore {  
  cpt ;           // valeur du semaphore  
  file ;         // file du semaphore  
}
```

Files d'attente

- Une file d'attente par sémaphore
- Une file d'attente de l'UC
- Opérations standards sur une file f
 - entrer (pid,f)
 - premier (f) -> pid
 - sortir (f) -> pid
 - vide (f) -> booléen

Gestion de l'unité centrale

- Objectifs
- Avec ou sans réquisition
- Méthode du tourniquet : avantage aux programmes courts
 - Choix du quantum d'UC
- Autres idées : priorités, files multiples, échéances, etc.

Traitement de l'IT horloge

- Arrêt du processus élu
- Chargement horloge
- Lancement d'un nouveau processus
- Difficulté technique : le changement d'état

Changement d'état

```
ranger_proelu {
    proelu->reg = reg ;
    proelu->psw = ancien_mep ; /* le mot d'état du
    programme interrompu n'est plus dans mep, mais a
    été rangé dans ancien_mep par le matériel */
}
lancer_exec(proelu) {
    reg = proelu->reg ;
    ancien_mep = proelu->psw ;
    rti ; //relance l'exécution à partir de ancien_mep
}
```

Traitant de l'interruption horloge

```
traitant_it_horloge {
    ranger_proelu ; /* sauvegarde registres et
    mep */
    proelu->etat = eligible ;
    entrer (proelu, f_elistes) ;
    lancer_processus_suivant ;
}
```

Définition et lancement du prochain processus

```
lancer_processus_suivant {  
    proelu = sortir (f_elligibles) ;  
    proelu->etat = elu ;  
    lancer_horloge(quantum) ;  
    lancer_exec (proelu) ;  
}
```

Réalisation des opérations P et V

- Des appels systèmes pour le masquage
- Mouvements de files d'attente
- Se terminent par le changement du processus élu et son activation

Traitant de l'appel système

```
traitant_svc { // seuls P et V sont possibles  
    switch (r0) // r0 code de l'appel système  
    {  
        case CODE_P : executer_p (r1) ; break ;  
        case CODE_V : executer_v (r1) ; break ;  
        default : exec_erreur ;  
    }  
}
```

Traitement de P(s)

```
executer_p(s) {  
  s->cpt -- ;  
  if (s->cpt >=0 ) rti ;  
  else {  
    ranger_proelu ;  
    proelu->etat = bloque ;  
    entrer (proelu,s->file) ;  
    lancer_processus_suivant ;  
  }  
}
```

Traitement de V(s)

```
executer_v(s) {  
  ranger_proelu ;  
  proelu->etat = eligible ;  
  entrer (proelu,f_eligibles) ;  
  s->cpt ++ ;  
  if (s->cpt <=0 ) {  
    paux = sortir (s->file) ;  
    paux->etat = eligible ;  
    entrer (paux, f_eligibles) ;  
  } ;  
  lancer_processus_suivant ;  
}
```

Initialisation

```
vect_int[SVC]=(&traitant_svc,maitre, masque) ;  
timer = 0 ;  
vect_int[HORLOGE]=(&traitant_it_horloge, maitre,  
masque) ;
```

/* il faut initialiser aussi les files de processus */

Aspects dynamiques

- **Création**
 - de processus
 - de sémaphore
- **Destruction**
 - de processus
 - de sémaphores
- **Maintien de la cohérence lors d'une destruction**
