Chapitre 2

Système d'Exploitation : mécanismes de base

- ressources
- ordonnancement
- commutation d'état
- interruptions

2.1 Quelques notions de base

Ressources :

⊡ Une ressource : élément qui contribue à la progression des processus.

⊡ Il y a deux types de ressouces : matérielle et logicielle :

⇒ Ressource matérielle (hardware) : UC, mémoire, réseau, ...

⇒ Ressource Logicielle (software) : les applications, compilateurs, ...

Remarques :

- Le dispatcheur / chargeur / ordonnanceur / ... sont des ressources logicielles.

- Un **processus** est une ressource logicielle (peut être appelé).

– Une ressource est décrite par un **TDA** : classe **ressource.**

– Un **objet ressouce** est instance de cette classe : e.g. une case mémoire.

– Opérateurs du TDA ressource : *demander, allouer, utiliser, libérer*

- La demande (via une commande du SE), l'allocation et la libération sont faites par le SE.
- Ressouce **réutilisable** : on peut demander, libérer, redemander, ... (e.g. mémoire)
- Certaines ressources **non réutilisables**
  $\Rightarrow$ disparaissent après utilisation (e.g. un message).
- Gestion des ressources par des tables de ressources
  $\Rightarrow$ états : disponible , allouée (à qui ?), ..
- une file d'attente (de PCB) par ressource ,
- **Réquisition** de ressource : ressource demandée par un processus prioritaire.
  $\Rightarrow$ save de l'état pour la restoration.

Ordonnancement :

⊡ Dans un système multi programmé mono processeur, l'ordonnanceur définit l'ordre d'attribution d'une ressource.

→ e.g. l'UC + sa durée d'utilisation à un processus .

⇒ Bon algo → bon taux d'utilisation → performances ↗.

⊡ Un ordonnanceur par ressource (e.g. pour la mémoire, les périphériques, ..).

⊡ Le plus important des ordonnanceurs

⟹ l'ordonnanceur de l'UC : **Scheduleur** (UC).

⊡ **Files d'attente :**

- Demande d'une ressource non disponible ⇒ PCB en file d'attente

- Lorsque la ressource est/devient disponible :

→ sélection d'un processus par l'ordonnanceur.

→ le processus va dans la file "prêt" (éligibles).

⊡ Types d'ordonnanceurs

– Rappel : un ordonnanceur par ressource.

– • Ordonnanceur à **long terme** (job scheduler) :

- gère les demandes d'entrée dans le système

- la demande peut être refuseé/différée (trop de processus présents, contraintes de performance, ..)

– • Ordonnanceur à **court terme** (de l'UC) :

- Choisit dans "prêt" le prochain processus

- Cet ordonnanceur est activé après la phase de calcul du processus en cours (env. 10 ms.), peut-être plutôt ;

- l'ordonnanceur à long terme est appelé moins souvent (env. 10 s.)

– • Ordonnanceur à **moyen terme** (medium term scheduler) : arbitre des accès en mémoire centrale

- Si swapping (disque ↔ mémoire), le **medium term scheduler** décide quel processus (rangé sur le disque ou en mémoire) peut accéder à la mémoire centrale (MC).

- Les accès aux disques sont effectués par un autre ordonnanceur.

– • Une fois le processus élu connu, l'ordonnanceur de l'UC (à court terme) appelle un autre processus système nommé **distributeur (dispatcher)** dont le rôle est de donner effectivement l'UC au processus élu.

⊡ Algorithmes d'ordonnancement :

– But : augmenter les performances et le taux d'utilisation

– Dépend de la ressource

– Différents types d'algorithme

– Seront détaillés plus loin

2.2 Mécanismes de base du SE

– Deux mécanismes de base implantés par matériel :

- La commutation du mot d'état (partie du contexte)
- Les interruptions

Commutation du mot d'état :

⊡ Pour être exécuté, un programme et ses données sont chargés en MC puis les instructions sont transférées de la MC vers l'UC.

⊡ Les registres de l'UC :

– L'UC contient des circuits (UAL, U. Cmd) et des registres

– Utilisation juducieuse pour diminuer les accès mémoire (cf. optimisation dans les compilateurs).

– Registre d'instruction : instruction en cours

– Compteur Ordinal (PC) : l'adresse de la prochaine instruction

$\Rightarrow$ la valeur du PC est modifiée pendant l'excution de l'instruction en cours pour pointer la prochaine .

$\Rightarrow$ L'exécution d'une instruction provoque des transferts UC $\leftrightarrow$ MC.

⇒ registre d'adresse : l'adresse de la cellule mémoire impliquée

⇒ registre de données : l'informations à transférer (cf. *Accumulateur*)

– Autres registres importants : registres d'état de l'UC.

⇒ ils indiquent un état : actif / inactif , mode super / user, codes des commandes (infos sur l'UC et sur le processus en cours : zone mémoire accessible, droits d'accès, priorité, ...).

⇒ L'ensemble de ces registres = **mot d'état** = **PSW** = Process Status Word

⇒ Le registre d'état fait partie du mot d'état.

⇒ A tout instant, un processus est caractérisé par :

- un programme et des données (= contexte en mémoire) ;

- un mot d'état (= PSW = contexte de l'UC)

Le PCB contient ces informations propres à chaque processus.

2.2.1 Commutation du mot d'état

- Scénario : plusieurs processus en méméoire, prêts à être exécutés
- A un instant donné de son exécution, un processus P est caractérisé par son contexte (état mémoire et UC)
- Si un autre processus Q reprenanit son exécution, il changerait le PSW.
- Pour pouvoir reprendre un processus, il faut sauvegarder son contexte ;
- La commutation de mot d'état permet de :

  $\hookrightarrow$ - sauvegarder l'état de P dans une zone précise (vers le PCB) ;

  $\hookrightarrow$ - charger un nouvel état (celui de Q) depuis son PCB
- La commutation même provoque l'exécution d'un autre processus (valeur du PC modifiée)
- Une commutation peut demander à Q de compléter, avant son prpore début, la sauvegarde de P.

Interruptions

– Une interruption (it° , déroutement)

⇒ c'est une commutation de mot d'état provoquée par un signal généré par le matériel (v. aussi **trap**)

– Ce signale est la conséquence d'un événement (evt) qui peut être :

- *interne* au processus en cours (e.g. demande d'E/S) et donc le résultantt de son exécution ;

- *externe* au processus, e.g. un evt venant d'un autre processus (un signal), d'un périphérique d'E/S, de l'opérateur, d'un capteur, ...

– Le signal modifie un indicateur (e.g. 1 bit/ 1 mot/ 1 vecteur) consulté régulièrement par le système (examen après chaque instruction, voire une micro instruction) ;

– On peut avoir plusieurs indicateurs (1 par type d'evt ⇒ plusieurs causes pour un type donné)

En général : plusieurs bits du PSW ⇒ vecteur d'it° :

– Le système détermine la cause de l'it° ;

à chaque cause est associé un niveau d'it°

– En général, il y a au moins 3 types (niveaux) d'it° :

- HardWare : les it° externes au processus (e.g. pannes, opérateur, ...)

- SoftWare : 2 types

* les déroutements (erreur, div/0, overflow) liées à l'instruction en cours

* les appels système = SVC (e.g. une demande d'E/S).

– Les it° sont traités selon leur niveau (= priorité, importance)

⇒ Exemple des niveaux sur IBM 360/370 (5 est le plus élevé) :

- niveau 5 : erreurs matérielles

- niveau 4 : déroutement (soft, trap)

- niveau 3 : appels systèmes

- niveau 2 : it externes

- niveau 1 : E / S

– Lors d'une it° ⇒ reconnaissance + commutation de contexte

⇒ commutation décidée selon le niveau de l'it°

– (1) le processus en cours est suspendu ou mis en attente (selon le type et la cause) ;

– (2) le chargement du mot d'état modifie PC et provoque l'exécution d'un **traitant** (it handler) ;

⇒ un traitant associé à chaque type d'it°

– Exemple des niveaux et les traitants sous Unix (0=↑) :

- niveau 0 : it Horloge (handler : clockintr)

- niveau 1 : it disque (handler : diskintr)

- niveau 3 : it console (ttyintr)

- niveau 4 : it autre périphérique (devintr)

- niveau 5 : appel system (softintr)

- niveau 6 : autre it (otherintr)

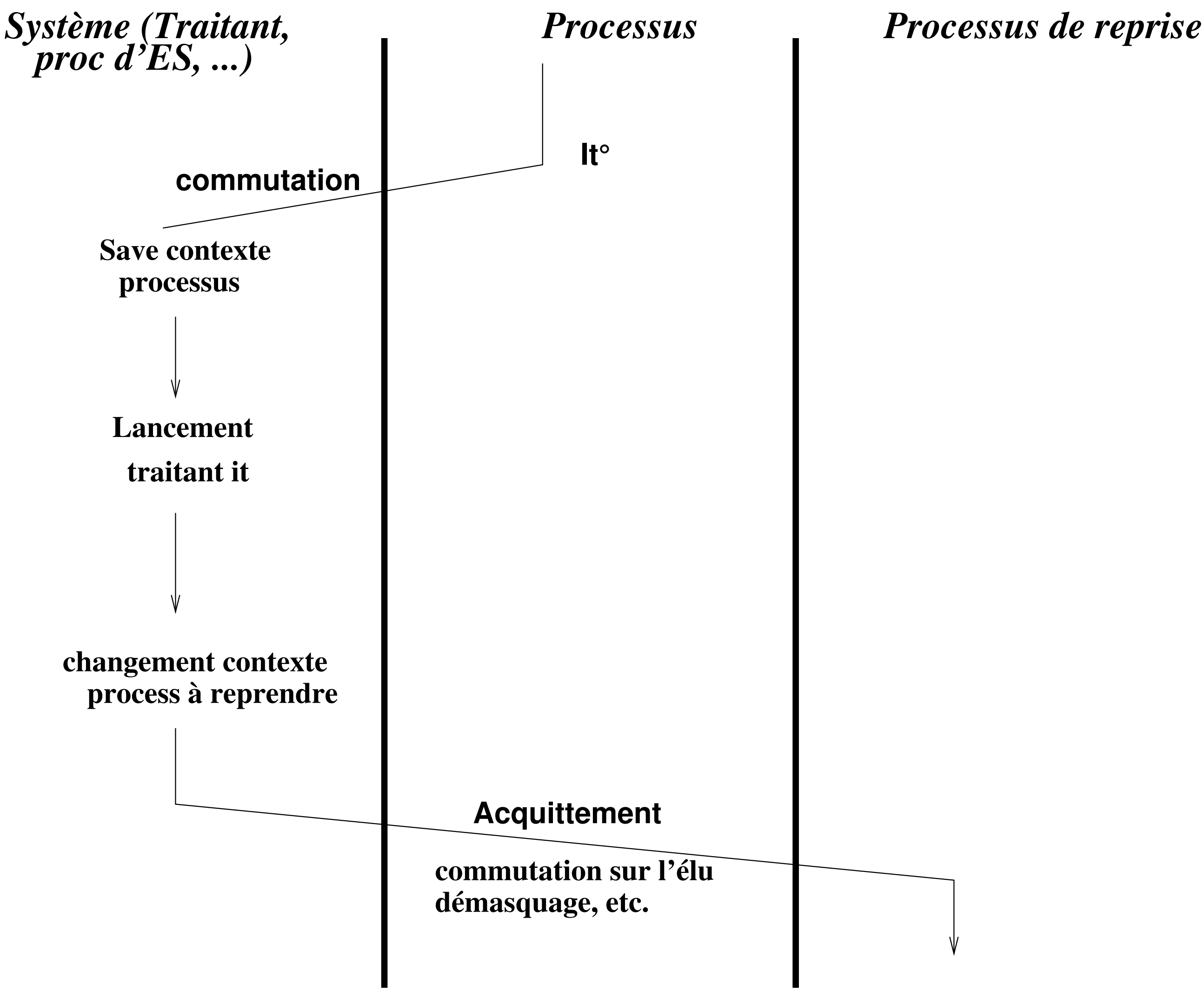
– Le traitant peut compléter la sauvegarde du contexte de l'ancien processus ;

- si le même processus doit reprendre (e.g. cas d'appel de l'opérateur), on ne sauvegarde pas le contexte mémoire.

- le minimum à sauvegarder (par matériel) est le PSW, le reste peut être fait par programme (e.g. par le handler) ;
- le traitant s'exécute puis l'ordonnanceur charge le contexte d'un processus (peut être le process interompu ?) ;

$\Rightarrow$ Cette opération sera **l'aquittement** de l'it°.

– Donc : it° $\Rightarrow$ traitant $\Rightarrow$ chargement de contextes + commutation (= aquittement).



Priorité, masquage et désarmement

– Une/plusieurs it° ⇒ plusieurs indicateurs positionnés

- pour une it° : le niveau ⇒ la priorité

- le SE traite la plus prioritaire (niveau le +élevé d'abord)

– Pendant l'exécution d'un handler, une it° peut arriver !

si ceci se répète, le SE ne progresse plus !

– → Il faut retarder / annuler la prise en compte d'une it°

- Technique : **masquage** et **désarmement** de niveau

⇒ On retarde la prise en compte des it° d'un/plusieurs niveaux → masquage de ce(s) niveau(x).

⇒ Si l'on ne veut pas les traiter → désarmement

– Le masquage se fait par un indicateur du PSW qui est consulté à l'arrivée d'une it° ⇒ **peut-on me déranger** ? ?

⇒ Un masque d'it° : la suite de valeurs des indicateurs.

– L'indicateur du niveau mis à 1 ⇒ niveau masqué

– Pour le démasquage : on remet l'indicateur à sa valeur d'origine.

– Le masque se modifie avec les commutations (why ?).

– Certaines it° ne sont pas masquables (car trop déterminantes)

– Pendant l'exécution d'un traitant d'un niveau N, on peut masquer les it° de niveau inférieur.
– Quand le traitant fait l'aquittement (appel de l'ordonnanceur), la nouvelle commutation restore (démasque) les niveaux masqués.
$\Rightarrow$ les it° survenues pendant le traitement peuvent être sauvegardées et traitées + tard (Pooling)
Elles sont sauvegardées (au niveau hard possible) sauf si désarmement (i.e. *on n'en veut pas* !)
– On peut complètement supprimer la prise en compte d'un niveau $\Rightarrow$ désarmement du niveau (+ réarmement ultérieur)
– Pour désarmer : un indicateur de PSW $\Rightarrow$ désarmer le niveau
– Certains déroutements (= it° logicielles), e.g div/0, ne peuvent pas être masqués (car lié à l'instruction en cours)
$\Rightarrow$ si on les traite plus tard $\rightarrow$ on plante pour sûr !
– d'autres (non masquables) peuvent être désarmés
– Remaque : désarmement = retour immédiat dans le traitant $\rightarrow$ pas de save ni traitement