

## Corrigé Série 3

Série rendue le: 2 mai 2005

### Exercice 1 : Sémantique des Moniteurs

Les scénarios donnent les diagrammes temps-processus suivant:

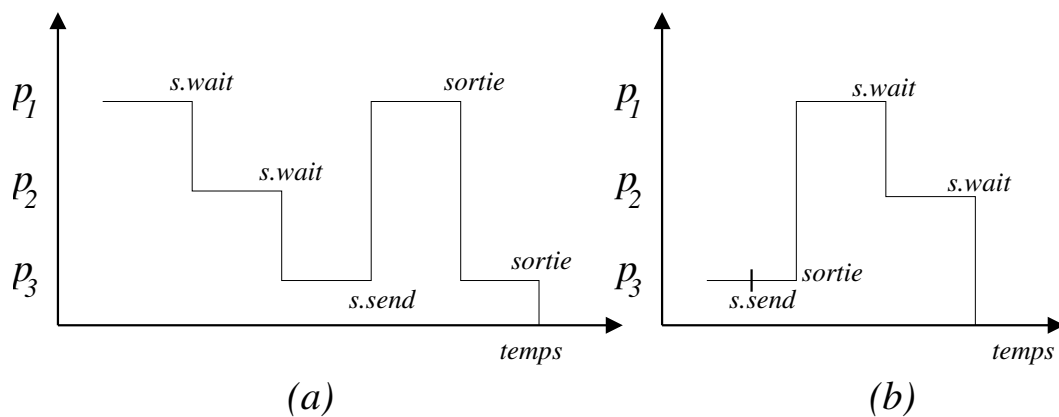


Figure 1: Diagramme temps-processus

### Exercice 2 : Lecteurs/rédacteurs avec des sémaphores

Une solution possible au problème des lecteurs-rédacteurs avec priorité aux rédacteurs est présentée dans l'Algorithme 1.

Le sémaphore  $r\_sem$  garantit la sémantique pour les lecteurs: un lecteur peut être autorisé à accéder à la section critique uniquement si aucun autre rédacteur n'y est présent ou n'attend pour y accéder. Ainsi, le premier rédacteur à se présenter dans `debut_écriture` appelle  $P(r\_sem)$ , qui a pour effet de bloquer tous les futurs lecteurs au début de `debut_lecture`. Tant qu'il y a un rédacteur en attente ( $w\_number \neq 0$ ), l'opération  $V(r\_sem)$  n'est pas appelée.

Si un lecteur est dans la section critique et qu'aucun rédacteur n'est en attente, d'autres lecteurs peuvent entrer dans la section critique car l'appel à  $P(r\_sem)$  ne sera pas bloquant.

Le sémaphore  $w\_sem$  garantit l'accès exclusif en écriture à la section critique. Un rédacteur à la fois aura accès à la section critique et un lecteur déjà dans la section critique pourra empêcher l'accès aux rédacteurs tant qu'il n'en est pas ressorti.

Finalement, les sémaphores  $r\_mutex$  et  $w\_mutex$  servent à l'exclusion mutuelle entre lecteurs, respectivement rédacteurs, lors de l'accès à  $r\_number$  et  $w\_number$  respectivement.

---

**Algorithme 1** Solution au problème des lecteurs-rédacteurs avec priorité aux rédacteurs

---

```

module priorite_redacteurs;
defines debut_lecture, fin_lecture, debut_ecriture, fin_ecriture;
var
  r_number: integer  $\leftarrow$  0;           {nombre de lecteurs}
  w_number: integer  $\leftarrow$  0;           {nombre de rédacteurs}
  r_mutex: semaphore  $\leftarrow$  1;         {exclusion mutuelle pour lecteurs}
  w_mutex: semaphore  $\leftarrow$  1;         {exclusion mutuelle pour rédacteurs}
  r_sem: semaphore  $\leftarrow$  1;           {garantit la sémantique pour les lecteurs}
  w_sem: semaphore  $\leftarrow$  1;           {garantit la sémantique pour les rédacteurs}

procedure debut_lecture
  P(r_sem);           {attente sur les rédacteurs}
  P(r_mutex);
  r_number  $\leftarrow$  r_number + 1;
  if r_number = 1 then
    P(w_sem); {bloque l'accès aux rédacteurs}
  end if
  V(r_mutex);
  V(r_sem);
end

procedure fin_lecture
  P(r_mutex);
  r_number  $\leftarrow$  r_number - 1;
  if r_number = 0 then
    V(w_sem); {permet l'accès aux rédacteurs}
  end if
  V(r_mutex);
end

procedure debut_ecriture
  P(w_mutex);
  w_number  $\leftarrow$  w_number + 1;
  if w_number = 1 then
    P(r_sem);           {bloque futurs lecteurs}
  end if
  V(w_mutex);
  P(w_sem);           {attend l'accès}
end

procedure fin_ecriture
  V(w_sem);           {permet l'accès aux lecteurs}
  P(w_mutex);
  w_number  $\leftarrow$  w_number - 1;
  if w_number = 0 then
    V(r_sem);           {permet l'accès aux lecteurs}
  end if
  V(w_mutex);
end

```

---

### Exercice 3 : Lecteurs/Rédacteurs avec des moniteurs

```
Monitor priorite_lecteurs;
  Defines debut_lecture,
         fin_lecture,
         debut_ecriture,
         fin_ecriture;

  Signal ecrireOK, lireOK;

  nb_lecteurs : integer;
  redacteur : boolean;

  Procedure debut_lecture;
  Code
    nb_lecteurs := nb_lecteurs + 1;
    if redacteur then lireOK.wait;
    lireOK.send;
  End debut_lecture;

  Procedure fin_lecture;
  Code
    nb_lecteurs := nb_lecteurs - 1;
    if nb_lecteurs = 0 then
      ecrireOK.send;
    endif
  End fin_lecture;

  Procedure debut_ecriture;
  Code
    if redacteur or
      nb_lecteurs != 0 then
      ecrireOK.wait;
    endif
    redacteur := true;
  End debut_ecriture;

  Procedure fin_ecriture;
  Code
    redacteur := false;
    if nb_lecteurs > 0 then
      lireOK.send;
    else
      ecrireOK.send;
    endif
  End fin_ecriture;

  Code (* init *)
    nb_lecteurs := 0;
    redacteur := false;
  End priorite_lecteurs;
```

Les signaux *lireOK* et *ecrireOK* sont utilisés pour attendre qu'une lecture, respectivement une écriture, soient possibles. La procédure *debut\_lecture* consiste à attendre qu'une lecture soit possible, augmenter le nombre de lecteurs et signaler les (éventuels) lecteurs en attente. Dans *fin\_lecture*, l'algorithme libère l'accès aux rédacteurs lorsque le dernier lecteur a arrêté de lire.

Dans *debut\_ecriture*, le rédacteur attend qu'il ait un accès exclusif (pas d'autre rédacteur et aucun lecteur). Il signale le début de sa rédaction grâce à la variable *redacteur*. Lors de la fin de l'écriture, le rédacteur signale les lecteurs s'il y en a en attente et, sinon, signale un éventuel rédacteur en attente.