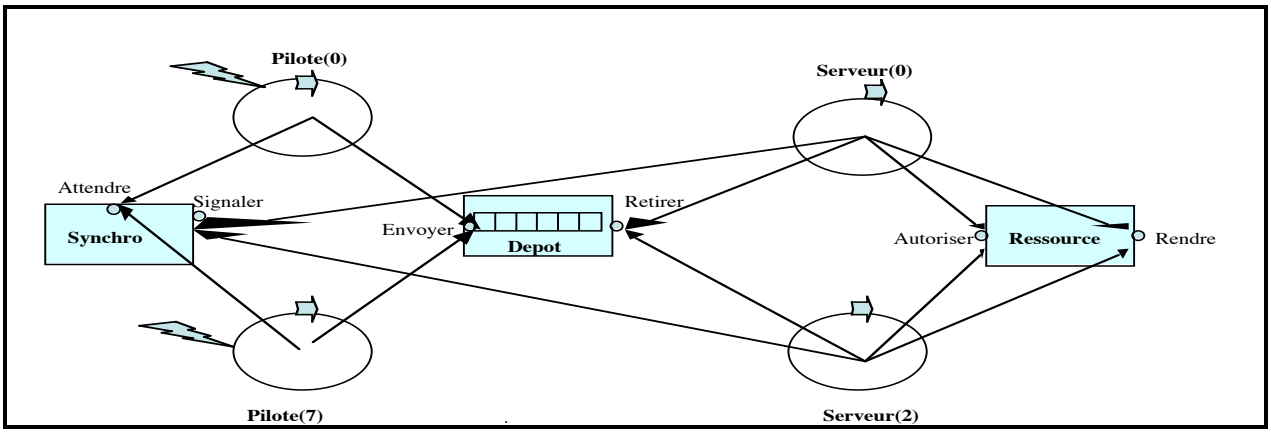
SURVEILLANCE PAR WEBCAMS

Un site Web utilise 8 appareils webcam de surveillance et récupère des images avec un lot de serveurs connectés à des clients. On a l'architecture suivante :




```
type Id_Cam is mod 8 ;

type Id_S is mod 3 ;

type Image is
  record
  Cam : Id_Cam ;
  Date : Time ;
  Paysage : Bitmap ;
end record ;

-- les processus coopérants

task type Un_Pilote ;
    Pilote : array(Id_Cam) of Un_Pilote ;

task type Un_Serveur ;
   Serveur : array(Id_S) of Un_Serveur ;
```

```
-- les paquetages de coopération

package Synchro is
  procedure Attendre(X : in Id_Cam);
  procedure Signaler(X : in Id_Cam);
end Synchro;

package Ressource is
  procedure Demander(X : in Id_Cam);
  procedure Rendre(X : in Id_Cam);
end Ressource;

package Depot is
  procedure Envoyer(X : in Id_Cam ; Y : in Image);
  procedure Retirer(X : in Id_Cam ; Y : out Image);
end Depot;
```

Chacun des trois processus Serveur est cyclique et est attaché à un client externe. Chaque Serveur commence par demander à son client le nom des caméras qu'il veut consulter soit A, B et C, (Ici on programme seulement le cas où la demande porte sur trois caméras). Après ce dialogue, le Serveur réserve chaque caméra demandée (c'est à dire A, B et C), puis il déclenche les pilotes associés et enfin retire du dépôt les images envoyées par ces pilotes pour les présenter au client.

Chacun des huit processus Pilote est cyclique et est attaché à une webcam. Chaque Pilote commence par attendre le signal de déclenchement, puis il lit la caméra et dépose l'image dans le tampon du dépôt. Clock est une fonction Ada qui retourne la date (et l'heure) courante.

```
task body Un_Pilote is
   X : Id_Cam := Nom.UniqueCam; Y : Image; Z : Bitmap ;
begin
   loop
        Synchro.Attendre(X) ; -- synchronisation pour attendre le signal envoyé à X par un serveur
        Lecture_de_la_Camera (Z) ;  -- acquisition de l'image Bitmap par un accès réseau local
        Y.Cam := X ; Y.Date : = Clock ; Y.Paysage := Z ;     -- préparation de Y
        Depot.Envoyer(X, Y) ; -- envoi vers les Serveurs de l'image Y prise par X
   end loop ;
end Un_Pilote ;
```

```
task body Un_serveur is
    I : Id_S := Nom.UniqueServeur ; A, B, C : Id_Cam; Y, Z, T : Image ;
                -- consultation de 3 webcam
begin
   loop
        Negociation_Avec_Le_Client(A, B, C) ; -- le client donne 3 noms de webcam
        Ressource.Demander(A) ; Ressource.Demander(B) ; Ressource.Demander(C) ;
        Synchro.Signaler(A) ; Synchro.Signaler(B) ; Synchro.Signaler(C) ;      -- réveil des pilotes
        Depot.Retirer(A, Y) ;    --  retrait de l'image de A qui est dans le dépôt et copie dans Y
        Depot.Retirer(B, Z) ;    --  retrait de l'image de B qui est dans le dépôt et copie dans Z
        Depot.Retirer(C, T) ;    --  retrait de l'image de C qui est dans le dépôt et copie dans T
        Ressource.Rendre(A) ; Ressource.Rendre(B) ; Ressource.Rendre(C) ;
        Envoi_Des_Images_Au_Client(Y, Z, T) ;
   end loop ;
end Un_Serveur ;
```

On vous demande de programmer l'application de surveillance.
Pour le paquetage Depot, on utilise les données persistantes

```
type Id_Depot is mod 6 ;
Tete, Queue : mod 6 := 0 ;
T : array (Id_Depot) of Image ;
```

La procédure Depot.Envoyer(X : in Id_Cam ; Y : in Image) permet aux pilotes de déposer des images à l'ancienneté dans le tampon du Depot. Le tampon peut contenir jusqu'à 6 images.

La procédure Depot.Retirer(X : in Id_Cam ; Y : out Image) doit permettre de retirer du tampon de Depot une image Y déposée par X, c'est à dire une image Y dont le champ Y.Cam est égal à X. Ce n'est pas un retrait à l'ancienneté. On veut cependant conserver l'utilisation du tampon sous forme de tampon circulaire habituel. Quand on est sûr que l'image Y (dont le champ Y.Cam est égal à X) est présente dans le tampon T, on l'extrait du tampon avec la procedure Extraire(X : in Id_Cam ; Y : out Image), qui est donnée ci-après.

Pour être sûr qu'une image de X est bien présente dans le tampon quand Extraire est appelée, on complète les procédures Depot.Envoyer(X : in Id_Cam ; Y : in Image) et Depot.Retirer(X : in Id_Cam ; Y : out Image). La procédure Depot.Envoyer(X : in Id_Cam ; Y : in Image) signale classiquement à l'ensemble des consommateurs l'arrivée d'un nouveau message, mais elle notifie aussi que l'envoi est fait par X. Cette notification est alors attendue dans Depot.Retirer(X : in

Id_Cam ; Y : out Image) avant l'attente de la présence d'une image et la lecture par Extrairece. Pour le dépôt,

procedure **Extraire**(X : in Id_Cam ; Y : out Image) is

```
-- on doit être certain qu'il y a dans le tampon T une image déposée par X
-- on va parcourir le tampon T à la recherche de T( ).Cam = X
-- puis compacter le tampon si nécessaire
-- au retour de cette procedure, il y a une case libre de plus et elle est située à (Queue – 1) mod 6

        I : Integer ;
begin
        I := Tete ;                                  -- on commence la recherche en Tete
        while I /= Queue loop
            if T(I).Cam = X then
                        Y := T(I) ;
                        if (I + 1) mod 6 = Queue then return; end if ; -- on a retiré la dernière image
                        -- on a retiré une image et cela laisse un trou ; on remplit ce trou
                        while (I + 1) mod 6 /= Queue loop
                                T(I)  := T((I + 1) mod 6) ;      -- on déplace l'image suivante
                                I := I + 1 ;
                        end loop ;
                        return; -- on a déplacé toutes les images qui étaient placées après Y
                end if ;
            I := (I + 1) modulo 6 ;   -- on n'a pas trouvé, on essaie la suivante
        end loop ;

-- Après ce retrait et cette gestion, il y a maintenant une case vide dans le tampon avant Queue
-- Depot.retirer doit donc mettre à jour Queue par l'instruction : Queue := (Queue – 1) mod 6 ;

end Extraire;
```

Nota. Un exemple de réalisation avec des sémaphores est donné dans l'énoncé et la solution de l'examen de septembre 2004 de la partie systèmes du cours Systèmes réseaux B .
voir
http://deptinfo.cnam.fr/Enseignement/CycleProbatoire/SRI/Systemes/ExamenSeptembre2004.pdf
et
http://deptinfo.cnam.fr/Enseignement/CycleProbatoire/SRI/Systemes/CorrigeSeptembre2004.pdf