Module STR
Examen session 1, 18 décembre 2014
Systèmes temps réel
Durée : 1h20
Tous documents autorisés

Singhoff Frank
singhoff@univ-brest.fr
C-202

NB : ce document contient 2 feuilles recto/verso.

Exercice n^o 1 : Questions de cours (3 points)

- Expliquez ce qu'est une inversion de priorité.
- Décrire le fonctionnement du procotole PIP.

Exercice n^o 2 : Ordonnancement temps réel (7 points)

Dans cet exercice, on étudie les tâches qui s'exécutent sur un téléphone portable. Le téléphone héberge 4 tâches périodiques :

- La tâche *Clavier* est en charge du clavier du téléphone.
- La tâche *Antenne* s'occupe du pilotage de l'antenne.
- La tâche $Video$ décode, stocke en mémoire puis affiche une séquence d'images. Une image est présentée à l'écran à chaque activation périodique de cette tâche.
- La tâche *Audio* décode, stocke en mémoire puis transmet à la carte son les échantillons audio. Un échantillon audio est présenté à l'écran à chaque activation périodique de cette tâche.

On suppose les tâches indépendantes. On suppose que $\forall i : Di = Pi$ et $Si = 0$. Ces tâches sont ordonnancées selon un algorithme à priorité fixe preemptif. On suppose un ordonnancement POSIX dont les priorités varient de 0 à 31 (0 est la priorité la plus forte). Le tableau ci-dessous présente les autres paramètres des tâches.

Nom	Capacité (en milliseconde)	Période (en milliseconde)
Antenne	4	30
Video	2	20
Audio	1	5
Clavier	1	10

Pour qu'un film soit correctement présenté, il est nécessaire que les tâches *Audio* et $Video$ respectent leurs échéances, on se propose donc d'étudier l'ordonnancement de ces tâches.

1. Proposez une priorité pour chaque tâche selon la politique Rate Monotonic.
2. Vérifiez que ce jeu de tâche est ordonnançable sans dessiner le chronogramme d'ordonnancement des tâches.
3. Confirmez le résultat de la question précédente en dessinant sur la période d'étude l'ordonnancement des tâches.
4. La capacité de la tâche $Video$ (2 millisecondes) permet à la tâche de traiter une image par activation. Pour traiter une image supplémentaire par activation, cette tâche requiert 2 millisecondes supplémentaires. Ainsi, une capacité de 4 millisecondes permet de traiter 2 images pendant une activation de la tâche, une capacité de 6 millisecondes permet de traiter 3 images, etc

   Quel est le nombre maximal d'image que cette tâche peut traiter tout en respectant les échéances de toutes les tâches ?

Exercice n^o 3 : Programmation concurrente avec Ada (4 points)

On se propose d'étudier le programme *cuisine.adb* donné en annexe. Ce programme simule le travail de deux cuisiniers modélisés sous la forme de deux tâches Ada. Ces cuisiniers se partagent l'accès à un four et à deux plats.

Chaque cuisinier réalise un gâteau. Pour ce faire, le premier cuisinier utilise un four et un plat. De son coté, le deuxième cuisinier utilise deux plats et un four.

Question 1

Avec le programme donné en annexe, un interblocage est possible. Décrivez comment cet interblocage peut se produire en donnant le scénario d'ordonnancement de ces quatre tâches qui conduit à l'interblocage.

Question 2

Modifier le programme afin d'éviter tout interblocage.

Annexe : le programme *cuisine.adb*

```
with text_io; use text_io;

procedure cuisine is

        task four is
                entry demander_le_four;
                entry liberer_le_four;
        end four;

        task plats is
                entry demander_un_plat;
                entry liberer_un_plat;
        end plats;

        task cuisinier1;
        task cuisinier2;

        task body plats is
        nb_plats : natural :=2;
        begin
          loop
                if(nb_plats = 0)
                        then accept  liberer_un_plat do
                                nb_plats:=nb_plats+1;
                             end liberer_un_plat;
                        else if(nb_plats = 2)
                                then accept  demander_un_plat do
                                        nb_plats:=nb_plats-1;
                                     end demander_un_plat;
                                else select
                                                accept demander_un_plat do
                                                        nb_plats:=nb_plats-1;
                                                end demander_un_plat;
                                           or       accept liberer_un_plat do
                                                        nb_plats:=nb_plats+1;
                                                end liberer_un_plat;
                                     end select;
                             end if;
                end if;
          end loop;
        end plats;
```

```
        task body four is
        begin
                loop
                        accept demander_le_four;
                        accept liberer_le_four;
                end loop;
        end four;

       task body cuisinier1  is
        begin
          loop
                four.demander_le_four;
                plats.demander_un_plat;

                put_line("cuisinier1 fait son gateau");
                delay 1.0;

                plats.liberer_un_plat;
                four.liberer_le_four;

                put_line("cuisinier1 a termine son gateau");
          end loop;
        end cuisinier1;

        task body cuisinier2 is
        begin
          loop
                plats.demander_un_plat;
                plats.demander_un_plat;
                four.demander_le_four;

                put_line("cuisinier2 fait son gateau");
                delay 1.0;

                plats.liberer_un_plat;
                plats.liberer_un_plat;
                four.liberer_le_four;

                put_line("cuisinier2 a termine son gateau");

          end loop;
        end cuisinier2;

begin
        null;
end cuisine;
```