6/7

GROUPE ESIEE PARIS

CHAMBRE DE COMMERCE ET D'INDUSTRIE DE PARIS

Nom Stéphane REVEL

Classe I4SE

Le 27 octobre 2000

Composition de iN412

Programmation concurrente

(1)

- Acquisition : cette tâche est lancée périodiquement toutes les 10 ms.
- Commande : cette tâche attend un événement "signaler" venant de la tâche acquisition
- Mise à jour des sorties : il nous manque des informations (?). Disons que cette tâche est, comme acquisition, lancée toutes les 10 ms.
- Affichage : cette tâche est lancée périodiquement t les 40 ms.

1,5

(2)

- On place les données portant de la tâche d'acquisition dans 2 FIFO temps réel, l'une à destination de "Commande", l'autre à destination "Affichage".
- On place les données portant de "Commande" da 2 FIFO, l'une à destination de "Mise à jour des sorties", l'autre à destination de "Affichage".

1,5

③ Grandes lignes du programme (1 entrée entière, 1 sortie entière)

```
void * acquisition
{
    int acq_resu;
    pthread_make_periodic_np(self, 10ms);
    while (1)
    {
        pthread_wait_np();
        acq_resu = acquerir();
        rtl_fifo_write(fifo1, acq_resu);
        rtl_fifo_write(fifo2, acq_resu);
}}      signaler(evt1);

void * commande
{
    int acq_resu;
    int commande;
    while (1)
    {
        attendre(evt1);
        rtl_fifo_read(fifo1, acq_resu);
        commande = calcul(acq_resu);
        rtl_fifo_write(fifo3, commande);
        rtl_fifo_write(fifo4, commande);
        effacer(evt1);
}}
```

2

```
void * mise_à_jour
{
    int commande;
    pthread_make_periodic_np (self, 10ms);
    while (1)
    {
        pthread_wait_np();
        commande = fifo_read (fifo 3);   // on attend qqch dans fifo
        appliquer (commande);
}}

void * affichage
{
    int valeur1;
    int valeur2;
    pthread_make_periodic_np (self, 40ms);
    while (1)
    {
        pthread_wait_np();
        valeur1 = fifo_read (fifo2);
        // on a changé le retour pour ne lire qu'une
        // valeur sur les 4. Pareil ci-dessous.
        valeur2 = fifo_read (fifo4);
        afficher_entree (valeur1);
        afficher_consigne (valeur2);
    }
}
```

A ceci s'ajoute un "init_module" qui crée les threads et les FIFO, et un "cleanup_module" qui les détruit.

④

On crée un 5ème thread, "watchdog". On fait en sorte que "mise-à-jour" signale l'événement 3 après avoir mis à jour les sorties.

On rend watchdog périodique de période 10ms. Si il détecte l'événement 3, il se rendort*. Sinon il le signale en lançant une procédure quelconque et il incrémente son compteur d'erreurs.

Il teste le compteur et si il vaut 5, il se charge d'arrêter le système.

* et il remet le compteur à 0.