UNIVERSITÉ TOULOUSE III PAUL SABATIER Université de Toulouse

Licence 3 Informatique

Université Toulouse III – Paul Sabatier
118 route de Narbonne
31062 Toulouse cedex 9

Programmation Système
Contrôle terminal – Session 1 – 17 décembre 2018

Durée : 1h30 **Aucun document autorisé**

Éléments de solution

Partie 1 : Processus UNIX

Partie 1

Programme principal

Vérifier les paramètres éventuels (x, y de la destination posDest)
setDestination(posDest)
Créer le processus Conducteur [fork]
en le protégeant de SIGUSR1 et SIGUSR2 [signal ou sigaction]
Créer le tube de communication [pipe]
Créer le processus Trajectoire [fork]
en le protégeant de SIGALRM [signal ou sigaction]
Créer le processus Commande [fork]
Fermer le tube en L/E [close] (Option 2 : Le main devient Commande)
Attendre les processus fils [wait] (Option 3 : Ne rien faire et se terminer)

Processus Conducteur

Variables : mode = DETENTE
Handler *de SIGUSR1 et SIGUSR2*
Réarmer protection [signal]
Si SIGUSR1 reçu : changer de mode de conduite (par exemple : (mode + 1)%2)
Si SIGUSR2 reçu : se terminer [exit]
Comportement

```
Tant que vrai faire
    Si mode == DETENTE alors
        detente()
    Sinon
        conduite()
    FinSi
FinTantQue
```

Processus Trajectoire – Version synchrone

Variables : posCrte, posDest, etat, arrive = false
Handler *de SIGALRM*
Réarmer protection contre SIGALRM [signal]
Si non arrive alors
Armer nouveau délai [alarm]
Fsi

Comportement

Fermer tube en lecture [close]
Armer délai [alarm]
Tant que posCrte != posDest faire
Attendre fin délai/période [pause ou sigsuspend]
etat = obtenirTrafic(posCrte)
calculerNewPosition(etat, &newPos)
Envoyer {etat, newPos} à Commande via tube[1] [write] (Option : envoyer newPos seule)
posCrte = newPos
FinTantQue
Fermer tube en écriture [close]

Processus Trajectoire – Version asynchrone

Variables : posCrte, posDest, etat, arrive = false

***Handler** de SIGALRM*

Réarmer protection contre SIGALRM [signal]
etat = obtenirTrafic(posCrte)
calculerNewPosition(etat, &newPos)
Envoyer {etat, newPos} à Commande via tube[1] [write] (Option : envoyer newPos seule)
posCrte = newPos
Si non arrive alors
Armer nouveau délai [alarm]
Fsi

Comportement

Fermer tube en lecture [close]
Armer délai [alarm]
Tant que posCrte != posDest faire
(rien)
FinTantQue
Fermer tube en écriture [close]

Processus Commande

Variables : modePrecedent = DETENTE
Fermer tube en écriture [close]
Tant que lecture sur tube[0] possible faire [read]
(Option : Récupérer posCrte et posDest + TantQue posCrte != posDest) faire)
Récupérer newPos et newEtat
(Option : Récupérer seulement posDest et estimer newEtat par obtenirEtatTrafic())
Si newEtat == CHARGE alors
mode = CONDUITE
Sinon
mode = DETENTE
Fsi
Si mode != modePrecedent alors
Envoyer SIGUSR1 au Conducteur [kill]
modePrecedent = mode
Fsi
piloter(newPos)
FinTantQue

Envoyer SIGUSR2 à Conducteur	[kill]

Partie 1 : Synchronisation par sémaphores

Question 1 : NBNT navettes individuelles

Sémaphores :

```
    Init(Aile, NBA) ;
    Init(Navette, NBNT) ;

DemanderAile() {
    P(Aile)
}
RendreAile() {
    V(Navette) ;
    V(Aile)
}
RemonterAvecNavette() {
    P(Navette) ;
}
```

Question 2 : Une navette à NBP places devant être remplie avant de pouvoir remonter

Partage :

```
    int NbParapentistes = 0 ;
```

Sémaphores :

```
    Init(Aile, NBA) ;
    Init(Navette, 0) ;
    Init(Mutex, 1) ;

DemanderAile () {
    P(Aile)
}
RendreAile() {
    V(Aile)
}
RemonterAvecNavette() {
    P(Mutex)
    nbParapentistes++
    if (nbParapentistes == NBP)  {
        for (i=0; i < NBP-1; i++)
            V(Navette);
        V(Mutex);
    }
    else {
        V(Mutex);
        P(Navette);
    }
}
```