|  |  |
| --- | --- |
| Bonnin Robin  Kuhner Pierre  3TC2 | 30 novembre 2015 |

SDE - Préparation TP IPC

Gestion d’un carrefour

# Conception

## Lancement de l’application

Le lancement de l’application se fait à l’aide d’un script bash qui lance Coordinateur et Générateur\_traffic. En effet, Générateur\_traffic\_prio et Feux sont des processus fils lancés par Coordinateur. Toutes les IPC sont instanciées par Coordinateur.

## Fermeture de l’application

La fermeture de l’application se fait par l’envoi d’un SIGKILL à Générateur\_traffi et à Coordinateur.

## Tests

* Vérifier la bonne initialisation des IPC
* Vérifier qu’il n’y a pas d’interblocage ni de famine (mutex)
* Vérifier que les IPC sont bien détruites à la fermeture

## Clés des IPC et variables globales

|  |  |
| --- | --- |
| 1 | #define cle\_bal 10 |
| 2 | #define cle\_shmem 20 |
| 3 | #define cle\_mutex 21 |
| 4 |  |
| 5 | #define NB\_VOITURES 100 |
| 6 |  |
| 7 | #define NORD 1 |
| 8 | #define EST 2 |
| 9 | #define SUD 3 |
| 10 | #define OUEST 4 |
| 11 |  |
| 12 | #define VERT 1 |
| 13 | #define ROUGE 0 |
| 14 |  |
| 15 | int depart\_prio = 0 |
| 16 | int arrivee\_prio = 0 |

## Macro-algorithmes

|  |  |
| --- | --- |
| 1 | Coordinateur { |
| 2 | créer la mémoire partagée qui contient : pid\_coord, pid\_feux, feu1, feu2, feu3, feu4, id\_prio, depart\_prio, arrivee\_prio |
| 3 | attacher la mémoire partagée en RW |
| 4 |  |
| 5 | créer la boîte aux lettres |
| 6 | attacher la boîte aux lettres |
| 7 |  |
| 8 | créer le sémaphore mutex |
| 9 |  |
| 10 | créer le processus fils Generateur\_traffic\_prio |
| 11 | créer le processus fils Feux |
| 12 |  |
| 13 | down(mutex) |
| 14 | stocker les PID de Coordinateur et de Feux dans la mémoire partagée |
| 15 | up(mutex) |
| 16 |  |
| 17 | rediriger le signal SIGUSR1 vers la fonction alerter\_feux |
| 18 |  |
| 19 | int[NB\_VOITURES][3] voitures |
| 20 | int nb |
| 21 |  |
| 22 | for(i=0;i<NB\_VOITURES;i++) { |
| 23 | voitures[i][0] = 0 |
| 24 | voitures[i][1] = 0 |
| 25 | voitures[i][2] = 0 |
| 26 | } |
| 27 |  |
| 28 | while(1) { |
| 29 | nb = 0 |
| 30 | lire état des feux |
| 31 |  |
| 32 | for(i=0;i<NB\_VOITURES;i++) { |
| 33 | if(voiture[i][2] == 0){ |
| 34 | if(il y a une voiture dans la file) { |
| 35 | voitures[i][0] = depart |
| 36 | voitures[i][1] = arrivee |
| 37 | voitures[i][2] = id |
| 38 | nb++ |
| 39 | } |
| 40 | } else { |
| 41 | nb++ |
| 42 | } |
| 43 | } |
| 44 |  |
| 45 | afficher l'état des voitures |
| 46 |  |
| 47 | for(i = 0; i < nb; i++) { |
| 48 | if(feu\_nord == VERT && feu\_sud == VERT && feu\_est == ROUGE && feu\_ouest == ROUGE) { |
| 49 | if((voitures[i][0] == NORD && (voitures[i][1] == SUD || voitures[i][1] == OUEST)) || (voitures[i][0] == SUD && (voitures[i][1] == NORD || voitures[i][1] == EST))) { |
| 50 | voitures[i][0] = voitures[i][1] = voitures[i][2] = 0 |
| 51 | afficher "Coordinateur : La voiture `voitures[i][2]` a pu franchir le carrefour." |
| 52 | } |
| 53 |  |
| 54 | if((voitures[i][0] == NORD && voitures[i][1] == EST) || (voitures[i][0] == SUD && voitures[i][1] == OUEST)) { |
| 55 | voitures[i][0] = voitures[i][1] = voitures[i][2] = 0 |
| 56 | afficher "Coordinateur : La voiture `voitures[i][2]` a pu franchir le carrefour." |
| 57 | } |
| 58 | } else if(feu\_est == VERT && feu\_ouest == VERT && feu\_nord = ROUGE && feu\_sud == ROUGE) { |
| 59 | if((voitures[i][0] == EST && (voitures[i][1] == OUEST || voitures[i][1] == NORD)) || (voitures[i][0] == OUEST && (voitures[i][1] == EST || voitures[i][1] == SUD))) { |
| 60 | voitures[i][0] = voitures[i][1] = voitures[i][2] = 0 |
| 61 | afficher "Coordinateur : La voiture `voitures[i][2]` a pu franchir le carrefour." |
| 62 | } |
| 63 |  |
| 64 | if((voitures[i][0] == EST && voitures[i][1] == SUD) || (voitures[i][0] == OUEST && voitures[i][1] == NORD)) { |
| 65 | voitures[i][0] = voitures[i][1] = voitures[i][2] = 0 |
| 66 | afficher "Coordinateur : La voiture `voitures[i][2]` a pu franchir le carrefour." |
| 67 | } |
| 68 | afficher état des voitures |
| 69 | } |
| 70 | } |
| 71 |  |
| 72 | mettre les voitures restantes au début du tableau pour qu'elles soient au début de la file du feu... |
| 73 | } |
| 74 |  |
| 75 | se détacher de la boite aux lettres |
| 76 | se détacher de la mémoire partagée |
| 77 | } |

|  |  |
| --- | --- |
|  | |
| |  |  | | --- | --- | | 1 | Feux { | | 2 | attacher le mutex | | 3 | attacher la mémoire partagée en RW | | 4 | armer redirection signal SIGUSR2 vers fonction gestion\_prio | | 5 | int k = 0 | | 6 |  | | 7 | while(1) { | | 8 | sleep(5) | | 9 | k++ | | 10 |  | | 11 | if(k%2 == 0) { // On passe les feux nord/sud au vert et les feux est/ouest au rouge | | 12 | feu\_nord = feu\_sud = VERT | | 13 | feu\_est = feu\_ouest = ROUGE | | 14 | afficher "Feux : passage des feux Nord et Sud au vert." | | 15 | } else { // On fait l'inverse | | 16 | feu\_nord = feu\_sud = VERT | | 17 | feu\_est = fest\_ouest = ROUGE | | 18 | afficher "Feux : passage des feux Est et Ouest au vert." | | 19 | } | | 20 |  | | 21 | down(mutex) | | 22 | envoyer l'état des feux dans la mémoire partagée | | 23 | up(mutex) | | 24 | } | | 25 |  | | 26 | se détacher de la mémoire partagée | | 27 | } | |  |

|  |  |
| --- | --- |
| 1 | Generateur\_traffic\_prio { |
| 2 | attacher le mutex |
| 3 | attacher la mémoire partagée en RW |
| 4 | récupérer le PID de Coordinateur |
| 5 |  |
| 6 | int id\_prio = 0; |
| 7 |  |
| 8 | while(1) { |
| 9 | int t = random(15..25) |
| 10 | sleep(t) |
| 11 |  |
| 12 | depart\_prio = random(1..4) |
| 13 | arrivee\_prio = random(1..4) |
| 14 |  |
| 15 | while(depart\_prio == arrivee\_prio) { |
| 16 | depart\_prio = random(1..4) |
| 17 | } |
| 18 |  |
| 19 | down(mutex) |
| 20 | envoyer id\_prio, depart\_prio et arrivee\_prio dans la mémoire partagée |
| 21 | up(mutex) |
| 22 |  |
| 23 | afficher "!!! PRIORITAIRE Génération traffic prio : La voiture `id\_prio` vient de `depart` et veut aller vers `arrivee`. Je préviens le coordinateur." |
| 24 | id\_prio++ |
| 25 | envoyer un signal SIGUSR1 au Coordinateur |
| 26 | } |
| 27 |  |
| 28 | se détacher de la mémoire partagée |
| 29 | } |

|  |  |
| --- | --- |
| 1 | Generateur\_traffic { |
| 2 | attacher la boite aux lettres |
| 3 |  |
| 4 | int id, depart, arrive |
| 5 |  |
| 6 | while(1) { |
| 7 | depart = random(1..4) |
| 8 | arrivee = random(1..4) |
| 9 |  |
| 10 | while(depart == arrivee) { |
| 11 | depart = random(1..4) |
| 12 | } |
| 13 |  |
| 14 | afficher "Générateur traffic : La voiture `id` vient de `depart` et veut aller vers `arrivee`" |
| 15 | écrire id, depart et arrivee dans la boite aux lettres |
|  |  |
| 16 | id++ |
| 17 | sleep(1) |
| 18 | } |
| 18 |  |
| 19 | se détacher de la boite aux lettres |
| 20 | } |

|  |  |
| --- | --- |
|  | |
| |  |  | | --- | --- | | 1 | alerter\_feux { | | 2 | attacher la mémoire partagée et récupérer le PID de Feux | | 3 | afficher "!!! PRIORITAIRE Coordinateur : un véhicule prioritaire arrive. Je préviens les feux." | | 4 | envoyer signal SIGUSR2 à Feux | | 5 | se détacher de la mémoire partagée | | 6 | } | | 7 |  | | 8 | gestion\_prio() { | | 9 | attacher la mémoire partagée en RW | | 10 |  | | 11 | afficher "!!! PRIORITAIRE Feux : passage de tous les feux au rouge" | | 12 | feu\_nord = feu\_est = feu\_sud = feu\_ouest = ROUGE | | 13 |  | | 14 | afficher "!!! PRIORITAIRE Feux : passage du feu " | | 15 | switch(depart\_prio) { | | 16 | case NORD: | | 17 | feu\_nord = VERT | | 18 | afficher "Nord" | | 19 | break | | 20 | case EST: | | 21 | feu\_est = VERT | | 22 | afficher "Est" | | 23 | break | | 24 | case SUD: | | 25 | feu\_sud = VERT | | 26 | afficher "Sud" | | 27 | break | | 28 | case OUEST: | | 29 | feu\_ouest = VERT | | 30 | afficher "Ouest" | | 31 | break | | 32 | } | | 33 | afficher " au vert." | | 34 |  | | 35 | envoyer l'état des feux dans la mémoire partagée | | 36 | se détacher de la mémoire partagée | | 37 | } | |  |