

Document de Réalisation TP Multitache

Config

```

1  /*****
2  **
3          Config - Fichier de Configuration
4          -----
5  debut      : 19/03/16
6  copyright  : (C) 2016 par Nicolas Gripont
7  e-mail     : nicolas.gripont@insa-lyon.fr
8  *****/
9
10 //----- Interface de la tache <Config> (fichier Config.h) -----
11 #if ! defined ( CONFIG_H )
12 #define CONFIG_H
13
14 //-----
15 // Rôle de la tache <Config>
16 // Fichier contenant differentes definitions utilisée dans les différents
17 // modules de l'application
18 //
19 //-----
20
21 ////////////////////////////////////////
22 INCLUDE
23 //----- Interfaces
24 utilisees
25 #include "Outils.h"
26
27 //-----
28 Constantes
29
30 #define PARKING_EXE  "./Parking" // Fichier utilisé pour créer la clé
31 publique
32 #define TEMPO 1 // Temps en seconde pour garer une
33 voiture
34 #define TYPE_TERMINAL TypeTerminal::XTERM // Terminal utilisé
35 #define DROITS_BOITE_AU_LETTRE 0660 // Droit d'accès aux boites aux
36 lettres de demandes d'entrée
37 #define DROITS_MEMOIRE_PARTAGEE 0660 // Droit d'accès aux boites aux
38 lettres de demandes de sortie
39 #define DROITS_SEMAPHORE 0660 // Droit d'accès aux sémaphores
40
41 // Indices des entrée dans le tableau de requete de la mémoire
42 partagée contenant les requetes
43 #define INDICE_ENTREE_BLAISE_PASCALE_PROF 0
44 #define INDICE_ENTREE_BLAISE_PASCALE_AUTRE 1
45 #define INDICE_ENTREE_GASTON_BERGER 2
46
47 //-----
48 Types
49
50 // structure permettant de stocker les informations relatives à une

```

```
voiture
42 typedef struct voiture {
43     TypeUsager typeUsager;
44     unsigned int numero;
45     time_t arrivee;
46 } Voiture;
47
48
49 // structure representant un message envoyé dans le boite aux lettres de
50 // demandes d'entrée
51 typedef struct messageDemandeEntree {
52     long type;
53     Voiture voiture;
54 } MessageDemandeEntree;
55
56 // structure representant un message envoyé dans le boite aux lettres de
57 // demandes de sortie
58 typedef struct messageDemandeSortie {
59     long type;
60     int numeroPlace;
61 } MessageDemandeSortie;
62
63 // structure de la mémoire partagée stockant les informations relatives
64 // aux voitures garées dans le parking
65 typedef struct memoirePartageeVoitures {
66     Voiture voitures[NB_PLACES];
67 } MemoirePartageeVoitures;
68
69
70 //// structure de la mémoire partagée stockant les informations relatives
71 //// aux requetes d'entrée
72 typedef struct memoirePartageeRequetes {
73     Voiture requetes[NB_BARRIERES_ENTREE];
74 } MemoirePartageeRequetes;
75
76
77 //////////////////////////////////////// ↵
78 PUBLIC
79 //----- Fonctions ↵
80 publiques
81
82 #endif // CONFIG_H
```

Mere

```

1  /*****
2  **
3      Mere - Application Parking
4      -----
5  debut      : 16/03/16
6  copyright  : (C) 2016 par Nicolas Gripont
7  e-mail     : nicolas.gripont@insa-lyon.fr
8  *****/
9
10 //----- Interface du module <Mere> (fichier Mere.h) -----
11 #if ! defined ( MERE_H )
12 #define MERE_H
13
14 //-----
15 --
16 // Rôle du module <Mere>
17 // Ce module permet de mettre en place les ressources et de lancer les
18 // différent processus qui interviennent dans la gestion du parking.
19 //-----
20 --
21 ////////////////////////////////////////
22 INCLUDE
23 //----- Interfaces
24 //-----
25 utilisees
26
27 //-----
28 Constantes
29
30 //-----
31 Types
32
33 ////////////////////////////////////////
34 PUBLIC
35 //----- Fonctions
36 //-----
37 publiques
38
39 int main ();
40 // Mode d'emploi :
41 // Fonction main permettant de créer mettre en place les différents
42 // processus permettant le fonctionnement de l'application Parking
43 // Contrat :
44 // Si l'application n'est pas quitter de manière classique (commande e/E
45 // de simulation), il faut supprimer les IPCs créés (4 boîtes aux
46 lettres,
47 // 5 sémaphores et 3 mémoires partagées)
48 //
49 #endif // MERE_H

```

```

1  /*****
2  **
3      Mere - Application Parking
4      -----
5  debut      : 16/03/16
6  copyright  : (C) 2016 par Nicolas Gripont
7  e-mail     : nicolas.gripont@insa-lyon.fr
8  *****/
9
10 //----- Realisation du module <Mere> (fichier Mere.cpp) -----
11
12 ////////////////////////////////////////
13 INCLUDE
14 //----- Include
15 systeme
16 #include <unistd.h>
17 #include <sys/types.h>
18 #include <sys/wait.h>
19 #include <sys/msg.h>
20 #include <sys/ipc.h>
21 #include <errno.h>
22 #include <stdio.h>
23 #include <sys/shm.h>
24 #include <sys/sem.h>
25 #include <signal.h>
26 //----- Include
27 personnel
28 #include "Mere.h"
29 #include "Outils.h"
30 #include "Simulation.h"
31 #include "Config.h"
32 #include "Heure.h"
33 #include "Entree.h"
34 #include "Sortie.h"
35
36 ////////////////////////////////////////
37 PRIVE
38 //-----
39 Constantes
40
41 //-----
42 Types
43
44 //----- Variables
45 statiques
46
47 //----- Fonctions
48 privees
49 //static type nom ( liste de parametres )
50 // Mode d'emploi :
51 //
52 // Contrat :

```

```

45 //
46 // Algorithme :
47 //
48 //{
49 //} //----- fin de nom
50
51 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
52 PUBLIC
53 //----- Fonctions
54 publiques
55
56 int main ()
57 // Algorithme :
58 //
59 {
60     pid_t pid_GestionClavier;
61     int statut_GestionClavier;
62     pid_t pid_Heure;
63     int statut_Heure;
64     pid_t pid_EntreeGastonBerger;
65     int statut_EntreeGastonBerger;
66     pid_t pid_EntreeBlaisePascalProf;
67     int statut_EntreeBlaisePascalProf;
68     pid_t pid_EntreeBlaisePascalAutre;
69     int statut_EntreeBlaisePascalAutre;
70     pid_t pid_Sortie;
71     int statut_Sortie;
72
73     int msgid_FileDemandeEntree_Prof_BlaisePacal;
74     int msgid_FileDemandeEntree_Autre_BlaisePacal;
75     int msgid_FileDemandeEntree_GastonBerger;
76     int msgid_FileDemandeSortie_GastonBerger;
77
78     int mutex_MemoirePartageeVoitures;
79     int shmId_MemoirePartageeVoitures;
80     MemoirePartageeVoitures* memoirePartageeVoitures;
81
82     int mutex_MemoirePartageeRequetes;
83     int semSys_MemoirePartageeRequetes;
84     int shmId_MemoirePartageeRequetes;
85     MemoirePartageeRequetes* memoirePartageeRequetes;
86
87     struct sigaction action;
88
89     // PHASE INITIALISATION :
90
91     //masquage signaux SIGUSR1 et SIGUSR2
92     action.sa_handler = SIG_IGN;
93     sigemptyset(&action.sa_mask);
94     action.sa_flags = 0;
95     sigaction(SIGUSR1, &action, NULL);
96     sigaction(SIGUSR2, &action, NULL);

```



```

97
98
99 //MISE EN PLACE DES RESSOURCES
100 // /\ TODO : tester erreur (-1) pour chaque allocation de
ressource -> pour chaque appelle systeme
101 InitialiserApplication(TYPE_TERMINAL);
102
103 //Mise en place des boites aux lettres
104 msgid_FileDemandeEntree_Prof_BlaisePacal =
msgget(ftok(PARKING_EXE,0),IPC_CREAT | DROITS_BOITE_AU_LETTRE);
105 msgid_FileDemandeEntree_Autre_BlaisePacal =
msgget(ftok(PARKING_EXE,1),IPC_CREAT | DROITS_BOITE_AU_LETTRE);
106 msgid_FileDemandeEntree_GastonBerger =
msgget(ftok(PARKING_EXE,2),IPC_CREAT | DROITS_BOITE_AU_LETTRE);
107
108 msgid_FileDemandeSortie_GastonBerger =
msgget(ftok(PARKING_EXE,3),IPC_CREAT | DROITS_BOITE_AU_LETTRE);
109
110 //Mise en place des semaphores
111 mutex_MemoirePartageeVoitures =
semget(ftok(PARKING_EXE,4),1,IPC_CREAT | DROITS_SEMAPHORE);
112 semctl(mutex_MemoirePartageeVoitures,0,SETVAL,1);
113
114 mutex_MemoirePartageeRequetes =
semget(ftok(PARKING_EXE,5),1,IPC_CREAT | DROITS_SEMAPHORE);
115 semctl(mutex_MemoirePartageeRequetes,0,SETVAL,1);
116
117 semSys_MemoirePartageeRequetes =
semget(ftok(PARKING_EXE,6),3,IPC_CREAT | DROITS_SEMAPHORE);
118 semctl(semSys_MemoirePartageeRequetes,3,SETALL,0);
119
120 //Mise en place des mémoires partagées
121 shmId_MemoirePartageeVoitures =
shmget(ftok(PARKING_EXE,7),sizeof(MemoirePartageeVoitures),
IPC_CREAT | DROITS_MEMOIRE_PARTAGEE);
122 memoirePartageeVoitures = (MemoirePartageeVoitures*)
shmat(shmId_MemoirePartageeVoitures,NULL,0);
123 for(int i=0; i < (int) NB_PLACES ; i++)
124 {
125     memoirePartageeVoitures->voitures[i] = {TypeUsager::AUCUN,0,0};
126 }
127 shmdt(memoirePartageeVoitures);
128
129 shmId_MemoirePartageeRequetes =
shmget(ftok(PARKING_EXE,8),sizeof(MemoirePartageeRequetes),
IPC_CREAT | DROITS_MEMOIRE_PARTAGEE);
130 memoirePartageeRequetes = (MemoirePartageeRequetes*)
shmat(shmId_MemoirePartageeRequetes,NULL,0);
131 for(int i=0; i < (int) NB_BARRIERES_ENTREE ; i++)
132 {
133     memoirePartageeRequetes->requetes[i] = {TypeUsager::AUCUN,0,0};
134 }
135 shmdt(memoirePartageeRequetes);
136

```

```

137 //Création des processus fils
138 if( (pid_GestionClavier = fork()) == 0 )
139 {
140     GestionClavier(msgid_FileDemandeEntree_Prof_BlaisePacal,msgid_Fi
        leDemandeEntree_Autre_BlaisePacal,msgid_FileDemandeEntree_Gaston
        Berger,msgid_FileDemandeSortie_GastonBerger);
141 }
142 else if( (pid_EntreeBlaisePascalProf = fork()) == 0 )
143 {
144     Entree(TypeBarriere::PROF_BLAISE_PASCAL,INDICE_ENTREE_BLAISE_PAS
        CALE_PROF,msgid_FileDemandeEntree_Prof_BlaisePacal,mutex_Memoire
        PartageeRequetes,semSync_MemoirePartageeRequetes,shmId_MemoirePar
        tageeRequetes,mutex_MemoirePartageeVoitures,shmId_MemoirePartage
        eVoitures);
145 }
146 else if( (pid_EntreeBlaisePascalAutre = fork()) == 0 )
147 {
148     Entree(TypeBarriere::AUTRE_BLAISE_PASCAL,INDICE_ENTREE_BLAISE_PA
        SCALE_AUTRE,msgid_FileDemandeEntree_Autre_BlaisePacal,mutex_Memo
        irePartageeRequetes,semSync_MemoirePartageeRequetes,shmId_Memoire
        PartageeRequetes,mutex_MemoirePartageeVoitures,shmId_MemoirePart
        ageeVoitures);
149 }
150 else if( (pid_EntreeGastonBerger = fork()) == 0 )
151 {
152     Entree(TypeBarriere::ENTREE_GASTON_BERGER,INDICE_ENTREE_GASTON_B
        ERGER,msgid_FileDemandeEntree_GastonBerger,mutex_MemoirePartagee
        Requetes,semSync_MemoirePartageeRequetes,shmId_MemoirePartageeReq
        uetes,mutex_MemoirePartageeVoitures,shmId_MemoirePartageeVoiture
        s);
153 }
154 else if( (pid_Sortie = fork()) == 0 )
155 {
156     Sortie(msgid_FileDemandeSortie_GastonBerger,mutex_MemoirePartage
        eRequetes,semSync_MemoirePartageeRequetes,shmId_MemoirePartageeRe
        quetes,mutex_MemoirePartageeVoitures,shmId_MemoirePartageeVoitur
        es);
157 }
158 else
159 {
160     pid_Heure = ActiverHeure();
161
162     // PHASE MOTEUR
163     waitpid(pid_GestionClavier,&statut_GestionClavier,0);
164
165     // PHASE DESTRUCTION
166     kill(pid_Heure,SIGUSR2); //test valeur retour -1 error
167     kill(pid_EntreeBlaisePascalProf,SIGUSR2); //test valeur retour
        -1 error
    
```

```
168      kill(pid_EntreeBlaisePascalAutre,SIGUSR2);//test valeur retour -1 error
169      kill(pid_EntreeGastonBerger,SIGUSR2);//test valeur retour -1 error
170      kill(pid_Sortie,SIGUSR2);//test valeur retour -1 error
171
172      waitpid(pid_Heure,&statut_Heure,0);
173
174      waitpid(pid_EntreeBlaisePascalProf,&statut_EntreeBlaisePascalProf,0);
175
176      waitpid(pid_EntreeBlaisePascalAutre,&statut_EntreeBlaisePascalAutre,0);
177
178      waitpid(pid_EntreeGastonBerger,&statut_EntreeGastonBerger,0);
179      waitpid(pid_Sortie,&statut_Sortie,0);
180
181      //liberation ressources
182      TerminerApplication();
183      //liberation boites aux lettres
184      msgctl(msgid_FileDemandeEntree_Prof_BlaisePascal,IPC_RMID,0);
185      msgctl(msgid_FileDemandeEntree_Autre_BlaisePascal,IPC_RMID,0);
186      msgctl(msgid_FileDemandeEntree_GastonBerger,IPC_RMID,0);
187      msgctl(msgid_FileDemandeSortie_GastonBerger,IPC_RMID,0);
188      //liberation memoires partages
189      shmctl(shmId_MemoirePartageeVoitures, IPC_RMID, 0);
190      shmctl(shmId_MemoirePartageeRequetes, IPC_RMID, 0);
191      //liberation semaphores
192      semctl(mutex_MemoirePartageeVoitures, IPC_RMID, 0);
193      semctl(semSyc_MemoirePartageeRequetes, IPC_RMID, 0);
194      semctl(mutex_MemoirePartageeRequetes, IPC_RMID, 0);
195      exit(0);
196
197  }
```

```
} //----- fin de main
```

Simulation

```

1  /*****
2  **
3      Simulation - description
4      -----
5  debut      : 16/03/16
6  copyright  : (C) 2016 par Nicolas Gripont
7  e-mail     : nicolas.gripont@insa-lyon.fr
8  *****/
9
10 //----- Interface de la tache <Simulation> (fichier Simulation.h)
11 //-----
12 #if ! defined ( SIMULATION_H )
13 #define SIMULATION_H
14 //-----
15 // Rôle de la tache <Simulation>
16 //
17 //
18 //-----
19
20 ////////////////////////////////////////
21 INCLUDE
22 //----- Interfaces
23 utilisees
24
25 //-----
26 Constantes
27
28 //-----
29 Types
30
31 ////////////////////////////////////////
32 PUBLIC
33 //----- Fonctions
34 publiques
35
36 void GestionClavier(int msgid_FDE_P_BP, int msgid_FDE_A_BP, int
37 msgid_FDE_GB, int msgid_FDS_GB);
38 // Mode d'emploi :
39 //
40 // Contrat :
41 //
42 void Commande(char code, unsigned int valeur);
43 // Mode d'emploi :
44 //
45 // Contrat :
46 //
47 #endif // SIMULATION_H

```

```

1  /*****
2  **
3          Simulation - description
4          -----
5  debut      : 16/03/16
6  copyright  : (C) 2016 par Nicolas Gripont
7  e-mail     : nicolas.gripont@insa-lyon.fr
8  *****/
9
10 //----- Realisation de la tache <Simulation> (fichier
Simulation.cpp) ---
11
12 ///////////////////////////////////////////////////
INCLUDE
13 //----- Include
systeme
14 #include <stdlib.h>
15 #include <time.h>
16 #include <sys/msg.h>
17 #include <signal.h>
18
19 //----- Include
personnel
20 #include "Simulation.h"
21 #include "Menu.h"
22 #include "Config.h"
23
24 ///////////////////////////////////////////////////
PRIVE
25 //-----
Constantes
26
27 //-----
Types
28
29 //----- Variables
statiques
30
31 //----- Fonctions
privees
32 static unsigned int GetNumeroVoiture()
33 // Mode d'emploi :
34 //
35 // Contrat :
36 //
37 // Algorithme :
38 //
39 {
40     static unsigned int numeroVoiture = 0; // variable remanente,
initialisée au premier appel
41
42     return (numeroVoiture = (numeroVoiture % 999 + 1));

```

```

43 } //----- fin de GetNumeroVoiture
44
45
46 ///////////////////////////////////////////////////////////////////
47 PUBLIC
48 //----- Fonctions
49 publiques
50
51 static int msgid_FileDemandeEntree_Prof_BlaisePacal;
52 static int msgid_FileDemandeEntree_Autre_BlaisePacal;
53 static int msgid_FileDemandeEntree_GastonBerger;
54 static int msgid_FileDemandeSortie_GastonBerger;
55
56 void GestionClavier(int msgid_FDE_P_BP, int msgid_FDE_A_BP, int
57 msgid_FDE_GB, int msgid_FDS_GB)
58 // Algorithme :
59 //
60 {
61     msgid_FileDemandeEntree_Prof_BlaisePacal = msgid_FDE_P_BP;
62     msgid_FileDemandeEntree_Autre_BlaisePacal = msgid_FDE_A_BP;
63     msgid_FileDemandeEntree_GastonBerger = msgid_FDE_GB;
64     msgid_FileDemandeSortie_GastonBerger = msgid_FDS_GB;
65
66     struct sigaction action;
67     action.sa_handler = SIG_IGN ;
68     sigemptyset(&action.sa_mask);
69     action.sa_flags = 0 ;
70     sigaction(SIGUSR1,&action,NULL);
71
72     struct sigaction actionFin;
73     actionFin.sa_handler = SIG_IGN;
74     sigemptyset(&actionFin.sa_mask);
75     actionFin.sa_flags = 0 ;
76     sigaction(SIGUSR2,&actionFin,NULL);
77
78     for(;;)
79     {
80         Menu();
81     }
82 } //----- fin de GestionClavier
83
84 void Commande(char code, unsigned int valeur)
85 // Algorithme :
86 //
87 {
88     MessageDemandeEntree demandeEntree;
89     MessageDemandeSortie demandeSortie;
90
91     switch (code) {
92     case 'e':
93     case 'E':
94         exit(0);
95     }
96 }

```

```

94         break;
95     case 'p':
96     case 'P':
97         demandeEntree.voiture.arrivee = time(NULL);
98         demandeEntree.voiture.numero = GetNumeroVoiture();
99         demandeEntree.voiture.typeUsager = TypeUsager::PROF;
100        demandeEntree.type = 1;
101
102        switch (valeur) {
103        case 1: // Blaise Pascal
104
105            msgsnd(msgid_FileDemandeEntree_Prof_BlaisePacal,&demandeEntree,
106                sizeof(MessageDemandeEntree),0);
107            break;
108        case 2: // Gaston Berger
109
110            msgsnd(msgid_FileDemandeEntree_GastonBerger,&demandeEntree,
111                sizeof(MessageDemandeEntree),0);
112            break;
113        default:
114            break;
115        }
116        break;
117     case 'a':
118     case 'A':
119         demandeEntree.voiture.arrivee = time(NULL);
120         demandeEntree.voiture.numero = GetNumeroVoiture();
121         demandeEntree.voiture.typeUsager = TypeUsager::AUTRE;
122         demandeEntree.type = 1;
123
124        switch (valeur) {
125        case 1: // Blaise Pascal
126
127            msgsnd(msgid_FileDemandeEntree_Autre_BlaisePacal,&demandeEntree,
128                sizeof(MessageDemandeEntree),0);
129            break;
130        case 2: // Gaston Berger
131
132            msgsnd(msgid_FileDemandeEntree_GastonBerger,&demandeEntree,
133                sizeof(MessageDemandeEntree),0);
134            break;
135        default:
136            break;
137        }
138        break;
139     case 's':
140     case 'S':
141         demandeSortie.numeroPlace = valeur;
142         demandeSortie.type = 1;
143
144        msgsnd(msgid_FileDemandeSortie_GastonBerger,&demandeSortie,
145            sizeof(MessageDemandeSortie),0);
146        break;

```



```
138     default:
139         break;
140     }
141 } //----- fin de GestionClavier
142
143
144
145 //file d'attente:
146 //envoyer msg
147 //
148 msgsnd(msgid_FileDemandeEntree_ProfBlaisePacal,&snd,sizeof(MessageDemandeEntree),0);
```

Entree

```

1  /*****
2  **
3          Entree - description
4          -----
5  debut      : 16/03/16
6  copyright  : (C) 2016 par Nicolas Gripont
7  e-mail     : nicolas.gripont@insa-lyon.fr
8  *****/
9
10 //----- Interface de la tache <Entree> (fichier Entree.h) -----
11 #if ! defined ( ENTREE_H )
12 #define ENTREE_H
13
14 //-----
15 // Rôle de la tache <Entree>
16 //
17 //
18 //-----
19
20 //////////////////////////////////////
21 INCLUDE
22 //----- Interfaces
23 utilisees
24 #include "Outils.h"
25
26 //-----
27 Constantes
28
29 //-----
30 Types
31
32 //////////////////////////////////////
33 PUBLIC
34 //----- Fonctions
35 publiques
36
37
38 void Entree(TypeBarriere type, int indiceBarriere, int msgid_BAL, int
39 mutex_MPR, int semSyc_MPR, int shmId_MPR, int mutex_MPV, int shmId_MPV);
40 // Mode d'emploi :
41 //
42 // Contrat :
43 //
44
45 #endif // ENTREE_H

```

```

1  /*****
2  **
3          Entree - description
4          -----
5  debut      : 16/03/16
6  copyright  : (C) 2016 par Nicolas Gripont
7  e-mail     : nicolas.gripont@insa-lyon.fr
8  *****/
9
10 //----- Realisation de la tache <Entree> (fichier Entree.cpp) ---
11
12 ////////////////////////////////////////
13 INCLUDE
14 //----- Include
15 systeme
16 #include <sys/types.h>
17 #include <sys/wait.h>
18 #include <sys/msg.h>
19 #include <errno.h>
20 #include <stdio.h>
21 #include <sys/shm.h>
22 #include <sys/sem.h>
23 #include <signal.h>
24 #include <map>
25 #include <unistd.h>
26 #include <time.h>
27
28 //----- Include
29 personnel
30 #include "Entree.h"
31 #include "Config.h"
32 #include "Outils.h"
33
34 ////////////////////////////////////////
35 PRIVE
36 //-----
37 Constantes
38
39 //-----
40 Types
41
42 //----- Variables
43 statiques
44
45 static int mutex_MemoirePartageeVoitures;
46 static int shmId_MemoirePartageeVoitures;
47 static int shmId_MemoirePartageeRequetes;
48 static MemoirePartageeVoitures* memoirePartageeVoitures;
49 static MemoirePartageeRequetes* memoirePartageeRequetes;
50 static map<pid_t,Voiture> voituriers;

```

```

46 //----- Fonctions
47 privees
48 static void finVoiturier(int numSignal);
49 // Mode d'emploi :
50 //
51 // Contrat :
52 //
53 static void fin(int numSignal);
54 // Mode d'emploi :
55 //
56 // Contrat :
57 //
58
59 static void finVoiturier(int numSignal)
60 // Algorithme :
61 //
62 {
63     //moteur
64     sembuf prendre = {(short unsigned int)0, (short)-1, (short)0};
65     sembuf vendre = {(short unsigned int)0, (short)1, (short)0};
66     pid_t pid_Voiturier;
67     int statut_Voiturier;
68     int numeroPlace;
69     if(numSignal == SIGCHLD)
70     {
71         pid_Voiturier = wait(&statut_Voiturier);
72
73         map<pid_t,Voiture>::iterator it = voitures.find(pid_Voiturier);
74         Voiture voiture = it->second;
75         voitures.erase(pid_Voiturier);
76         numeroPlace = WEXITSTATUS(statut_Voiturier);
77
78         AfficherPlace(numeroPlace,voiture.typeUsager,voiture.numero,voiture.arrivee);
79
80         //mettre voiture dans place memoire partagee
81         while(semop(mutex_MemoirePartageeVoitures,&prendre,1) == -1 &&
82             errno == EINTR);
83
84         memoirePartageeVoitures->voitures[numeroPlace-1] = voiture;
85
86         semop(mutex_MemoirePartageeVoitures,&vendre,1);
87     }
88 } //----- fin de finVoiturier
89
90 static void fin(int numSignal)
91 // Algorithme :
92 //
93 {
94     //destruction
95     if(numSignal == SIGUSR2)

```

```

96     {
97         struct sigaction action;
98         action.sa_handler = SIG_IGN ;
99         sigemptyset(&action.sa_mask);
100        action.sa_flags = 0 ;
101        sigaction(SIGCHLD,&action,NULL);
102
103
104        for(map<pid_t,Voiture>::iterator it = voitures.begin(); it != voitures.end(); it++)
105        {
106            kill(it->first,SIGUSR2);
107        }
108        for(map<pid_t,Voiture>::iterator it = voitures.begin(); it != voitures.end(); it++)
109        {
110            waitpid(it->first,NULL,0);
111        }
112
113        shmdt(memoirePartageeVoitures);
114        shmdt(memoirePartageeRequetes);
115
116        exit(0);
117    }
118 } //----- fin de fin
119
120
121 ////////////////////////////////////////
122 PUBLIC
123 //----- Fonctions
124 publiques
125
126 void Entree(TypeBarriere type, int indiceBarriere, int msgid_BAL, int
127 mutex_MPR, int semSync_MPR, int shmId_MPR, int mutex_MPV, int shmId_MPV)
128 // Algorithme :
129 //
130 {
131     //initialisation
132     pid_t pid_Voiturier;
133     TypeBarriere typeBarriere = type;
134     int msgid_BoiteAuxLettres = msgid_BAL;
135     int mutex_MemoirePartageeRequetes = mutex_MPR;
136     int semSync_MemoirePartageeRequetes = semSync_MPR;
137     shmId_MemoirePartageeRequetes = shmId_MPR;
138     mutex_MemoirePartageeVoitures = mutex_MPV;
139     shmId_MemoirePartageeVoitures = shmId_MPV;
140
141     memoirePartageeVoitures = (MemoirePartageeVoitures*)
142     shmat(shmId_MemoirePartageeVoitures,NULL,0);
143     memoirePartageeRequetes = (MemoirePartageeRequetes*)
144     shmat(shmId_MemoirePartageeRequetes,NULL,0);
145
146     struct sigaction action;

```

```

143     action.sa_handler = SIG_IGN ;
144     sigemptyset(&action.sa_mask);
145     action.sa_flags = 0 ;
146     sigaction(SIGUSR1,&action,NULL);
147
148     struct sigaction actionFin;
149     actionFin.sa_handler = fin;
150     sigemptyset(&actionFin.sa_mask);
151     actionFin.sa_flags = 0 ;
152     sigaction(SIGUSR2,&actionFin,NULL);
153
154     struct sigaction actionFinVoiturier;
155     actionFinVoiturier.sa_handler = finVoiturier;
156     sigemptyset(&actionFinVoiturier.sa_mask);
157     actionFinVoiturier.sa_flags = 0 ;
158     sigaction(SIGCHLD,&actionFinVoiturier,NULL);
159
160     MessageDemandeEntree demande;
161
162     //moteur
163     for(;;)
164     {
165
166         while(msgrcv(msgid_BoiteAuxLettres,&demande,sizeof(MessageDemandeEntree),0,0) == -1 && errno == EINTR); //sans block
167
168         demande.voiture.arrivee = time(NULL);
169
170         if( (pid_Voiturier = GarerVoiture(typeBarriere)) == -1 )
171         {
172             //requete
173             sembuf prendreMutex = {(short unsigned int)0, (short)-1, (short)0};
174             sembuf vendreMutex = {(short unsigned int)0, (short)1, (short)0};
175             while(semop(mutex_MemoirePartageeRequetes,&prendreMutex,1) == -1 && errno == EINTR);
176             memoirePartageeRequetes->requetes[indiceBarriere] = demande.voiture;
177             semop(mutex_MemoirePartageeRequetes,&vendreMutex,1);
178             AfficherRequete(typeBarriere,demande.voiture.typeUsager,demande.voiture.arrivee);
179
180             DessinerVoitureBarriere(typeBarriere,demande.voiture.typeUsager);
181
182             sembuf prendreSemSync = {(short unsigned int)indiceBarriere, (short)-1, (short)0};
183             while(semop(semSync_MemoirePartageeRequetes,&prendreSemSync,1) == -1 && errno == EINTR);
184             pid_Voiturier = GarerVoiture(typeBarriere);
185         }

```

```
184         voituriers.insert(make_pair(pid_Voiturier,demande.voiture));
185
186         sleep(TEMPO);
187     }
188 } //----- fin de Entree
189
190
191
```


Sortie

```
1  /*****
2  **
3          Sortie - description
4          -----
5  debut      : 16/03/16
6  copyright  : (C) 2016 par Nicolas Gripont
7  e-mail     : nicolas.gripont@insa-lyon.fr
8  *****/
9
10 //----- Interface de la tache <Sortie> (fichier Sortie.h) -----
11 #if ! defined ( SORTIE_H )
12 #define SORTIE_H
13
14 //-----
15 // Rôle de la tache <Sortie>
16 // Module permettant de gérer la sortie de voitures du Parking
17 //
18 //-----
19
20 ////////////////////////////////////////
21 INCLUDE
22 //----- Interfaces
23 utilisees
24
25 //-----
26 Constantes
27
28 //-----
29 Types
30
31 ////////////////////////////////////////
32 PUBLIC
33 //----- Fonctions
34 publiques
35
36 void Sortie(int msgid_BAL, int mutex_MPR, int semSyc_MPR, int shmId_MPR,
37             int mutex_MPV, int shmId_MPV);
38 // Mode d'emploi : Fonction permettant de gérer es sortie du parking
39 //
40 // Contrat : les différents paramètres doivent être correctes.
41 // (les id des IPCs)
42 //
43 #endif // SORTIE_H
```

```

1  /*****
2  **
3      Sortie - description
4      -----
5  debut      : 16/03/16
6  copyright  : (C) 2016 par Nicolas Gripont
7  e-mail     : nicolas.gripont@insa-lyon.fr
8  *****/
9
10 //----- Realisation de la tache <Sortie> (fichier Sortie.cpp) ---
11
12 ///////////////////////////////////////////////////
13 INCLUDE
14 //----- Include
15 systeme
16 #include <sys/types.h>
17 #include <sys/wait.h>
18 #include <sys/msg.h>
19 #include <errno.h>
20 #include <stdio.h>
21 #include <sys/shm.h>
22 #include <sys/sem.h>
23 #include <signal.h>
24 #include <set>
25 #include <unistd.h>
26 #include <time.h>
27
28 //----- Include
29 personnel
30 #include "Config.h"
31 #include "Outils.h"
32 #include "Sortie.h"
33
34 ///////////////////////////////////////////////////
35 PRIVE
36 //-----
37 Constantes
38
39 //-----
40 Types
41
42 //----- Variables
43 statiques
44 static int mutex_MemoirePartageeVoitures;
45 static int shmId_MemoirePartageeVoitures;
46 static int mutex_MemoirePartageeRequetes;
47 static int shmId_MemoirePartageeRequetes;
48 static int semSyc_MemoirePartageeRequetes;
49 static MemoirePartageeVoitures* memoirePartageeVoitures;
50 static MemoirePartageeRequetes* memoirePartageeRequetes;
51 static set<pid_t> voituriers; // set contenant les pids des

```

```

voiturierSortie
// en cours d'exécution

//----- Fonctions
privees

static void finVoiturier(int numSignal);
// Mode d'emploi : fonction permettant de gérer la fin d'un processus
// SortieVoiture.
// A connecter au signal SIGCHLD
//
// Contrat : l'accès à la mémoire partagés requêtes et voitures
// doit être possible (memoirePartageeRequetes et
memoirePartageeVoitures)
// L'accès aux différents sémaphores doit être possible
// (mutex_MemoirePartageeVoitures,mutex_MemoirePartageeRequetes
// semSyc_MemoirePartageeRequetes)
//
static void fin(int numSignal);
// Mode d'emploi : fonction permettant de Terminer le processus.
// A connecter au signal SIGUSR2
//
// Contrat : l'accès à la mémoire partagés requêtes et voitures
// doit être possible (memoirePartageeRequetes et
memoirePartageeVoitures)

static int choixRequete();
// Mode d'emploi : Fonction permettant de choisir la requete d'entrée
// à traiter. Renvoie l'indice de la requete à traiter (dans la
// memoirePartageeRequetes) ou -1 si aucune requete.
//
// Contrat : l'accès à la mémoire partagés requêtes doit être possible
// (memoirePartageeRequetes)

static bool isV1Prio(Voiture v1, Voiture v2);
// Mode d'emploi : Renvoie true si v1 est prioritaire par rapport à v2,
// false sinon.
//
// Contrat : aucun

static void finVoiturier(int numSignal)
// Algorithme :
//
{
    // PHASE MOTEUR :
    sembuf prendreMutex = {(short unsigned int)0, (short)-1, (short)0};
    sembuf vendreMutex = {(short unsigned int)0, (short)1, (short)0};
    pid_t pid_Voiturier;
    int statut_Voiturier;
    int numeroPlace;

```

```

96  Voiture voiture;
97  int indiceRequete;
98
99  if(numSignal == SIGCHLD)
100 {
101     //recuperation du pid du voiturier terminé
102     pid_Voiturier = wait(&statut_Voiturier);
103     numeroPlace = WEXITSTATUS(statut_Voiturier);
104
105     //on prend le mutex memoire partagée voitures pour pouvoir y
106     //accéder (être seul à y accéder)
107     while(semop(mutex_MemoirePartageeVoitures,&prendreMutex,1) ==
108     -1 && errno == EINTR);
109
110     //recupération de la voiture sortie
111     voiture = memoirePartageeVoitures->voitures[numeroPlace-1];
112     //on efface la voiture sortie de la mémoire partagée
113     memoirePartageeVoitures->voitures[numeroPlace-1] =
114     {TypeUsager::AUCUN,0,0};
115     //on rend l'accès à la mémoire partagée voitures
116     semop(mutex_MemoirePartageeVoitures,&vendreMutex,1);
117
118     //on efface les infos de la voiture sortie de la zone etat
119     Effacer((TypeZone)numeroPlace);
120     //on affiche les informations de la voiture sortie dans la
121     //zone sortie
122
123     AfficherSortie(voiture.typeUsager,voiture.numero,voiture.arrivee
124     ,time(NULL));
125     //on enleve le pid de du voiturierSortie terminé du set
126     voituriers.erase(pid_Voiturier);
127
128     //on prend le mutex memoire partagée requetes pour pouvoir y
129     //accéder (être seul à y accéder)
130     while(semop(mutex_MemoirePartageeRequetes,&prendreMutex,1) ==
131     -1 && errno == EINTR);
132
133     //s'il y a une requete a traiter
134     if( (indiceRequete = choixRequete()) != -1 )
135     {
136         //on efface la requete à traiter de la mémoire partagée
137         //requetes
138         memoirePartageeRequetes->requetes[indiceRequete] =
139         {TypeUsager::AUCUN,0,0};
140         //on efface la requete à traiter de la zone requetes
141         Effacer((TypeZone)(TypeZone::REQUETE_R1 + indiceRequete));
142         //on libere le processus entrée en donnant un jeton au
143         //semSync correspondant à la requête
144         sembuf vendreSemSync = {(short unsigned int)indiceRequete,
145         (short)1, (short)0};
146
147         while(semop(semSync_MemoirePartageeRequetes,&vendreSemSync,1)
148         == -1 && errno == EINTR);
149     }

```

```

136         //on rend l'accès à la mémoire partagée requetes
137         semop(mutex_MemoirePartageeRequetes,&vendreMutex,1);
138     }
139 } //----- fin de finVoiturier
140
141
142 static void fin(int numSignal)
143 // Algorithme :
144 //
145 {
146     // PHASE DESTRUCTION :
147     if(numSignal == SIGUSR2)
148     {
149         //on masque le signal SIGCHLD pour ne pas être interrompu par
150         //la fin
151         //d'un voiturierSortie
152         struct sigaction action;
153         action.sa_handler = SIG_IGN ;
154         sigemptyset(&action.sa_mask);
155         action.sa_flags = 0 ;
156         sigaction(SIGCHLD,&action,NULL);
157
158         //on envoie le signal SIGUSR2 a tous les voiturierSortie en
159         //cours d'exécution
160         for(set<pid_t>::iterator it = voituriers.begin(); it !=
161             voituriers.end(); it++)
162         {
163             kill(*it,SIGUSR2);
164         }
165         //on attend la fin des voiturierSortie à qui on a envoyé SIGUSR2
166         for(set<pid_t>::iterator it = voituriers.begin(); it !=
167             voituriers.end(); it++)
168         {
169             waitpid(*it,NULL,0);
170         }
171
172         //On se détache des mémoires partagées
173         shmdt(memoirePartageeVoitures);
174         shmdt(memoirePartageeRequetes);
175
176         //On quitte le processus
177         exit(0);
178     }
179 } //----- fin de fin
180
181 static bool isV1Prio(Voiture v1, Voiture v2)
182 {
183     if(v1.typeUsager == v2.typeUsager) { return (v1.arrivee <=
184         v2.arrivee); }
185     if(v1.typeUsager == TypeUsager::PROF && v2.typeUsager ==
186         TypeUsager::AUTRE) { return true; }
187     if(v1.typeUsager == TypeUsager::AUTRE && v2.typeUsager ==
188         TypeUsager::PROF) { return false; }

```

```

183     if(v1.typeUsager > v2.typeUsager) { return true; }
184     //     if(v1.typeUsager < v2.typeUsager) { return false; }
185     return false;
186 }
187
188 static int choixRequete()
189 // Algorithme :
190 //
191 {
192     Voiture v1 =
193     memoirePartageeRequetes->requetes[INDICE_ENTREE_BLAISE_PASCALE_PROF]
194     ;
195     Voiture v2 =
196     memoirePartageeRequetes->requetes[INDICE_ENTREE_BLAISE_PASCALE_AUTRE]
197     ];
198     Voiture v3 =
199     memoirePartageeRequetes->requetes[INDICE_ENTREE_GASTON_BERGER];
200
201     if(isV1Prio(v1,v2) && isV1Prio(v1,v3) && v1.typeUsager !=
202     TypeUsager::AUCUN)
203     {
204         return 0;
205     }
206     if(isV1Prio(v2,v1) && isV1Prio(v2,v3) && v2.typeUsager !=
207     TypeUsager::AUCUN)
208     {
209         return 1;
210     }
211     if(isV1Prio(v3,v2) && isV1Prio(v3,v1) && v3.typeUsager !=
212     TypeUsager::AUCUN)
213     {
214         return 2;
215     }
216
217     return -1;
218 } //----- fin de choixRequete
219
220 //////////////////////////////////////
221 PUBLIC
222 //----- Fonctions
223 publiques
224
225 void Sortie(int msgid_BAL, int mutex_MPR, int semSyc_MPR, int
226 shmId_MPR, int mutex_MPV, int shmId_MPV)
227 // Algorithme :
228 //
229 {
230     pid_t pid_Voiturier;
231     int msgid_BoiteAuxLettres;
232     MessageDemandeSortie demande;
233
234     // PHASE INITIALISATION
235     //masquage des signaux SIGUSR1, SIGUSR2, SIGCHLD

```

```

226 struct sigaction action;
227 action.sa_handler = SIG_IGN ;
228 sigemptyset(&action.sa_mask);
229 action.sa_flags = 0 ;
230 sigaction(SIGUSR1,&action,NULL);
231 sigaction(SIGUSR2,&action,NULL);
232 sigaction(SIGCHLD,&action,NULL);
233
234 //Positionnement du handler fin sur SIGUSR2
235 struct sigaction actionFin;
236 actionFin.sa_handler = fin;
237 sigemptyset(&actionFin.sa_mask);
238 actionFin.sa_flags = 0 ;
239 sigaction(SIGUSR2,&actionFin,NULL);
240
241 //Positionnement du handler finVoiturier sur SIGUSR2
242 struct sigaction actionFinVoiturier;
243 actionFinVoiturier.sa_handler = finVoiturier;
244 sigemptyset(&actionFinVoiturier.sa_mask);
245 actionFinVoiturier.sa_flags = 0 ;
246 sigaction(SIGCHLD,&actionFinVoiturier,NULL);
247
248 //récupération des paramètres
249 msgid_BoiteAuxLettres = msgid_BAL;
250 semSys_MemoirePartageeRequetes = semSys_MPR;
251 mutex_MemoirePartageeRequetes = mutex_MPR;
252 shmId_MemoirePartageeRequetes = shmId_MPR;
253 mutex_MemoirePartageeVoitures = mutex_MPV;
254 shmId_MemoirePartageeVoitures = shmId_MPV;
255
256 //Attachement des mémoires partagées
257 memoirePartageeVoitures = (MemoirePartageeVoitures*) ↵
    shmat(shmId_MemoirePartageeVoitures,NULL,0);
258 memoirePartageeRequetes = (MemoirePartageeRequetes*) ↵
    shmat(shmId_MemoirePartageeRequetes,NULL,0);
259
260 // PHASE MOTEUR
261 for(;;)
262 {
263     //lecture de messages dans la boite aux lettres
264     while(msgrcv(msgid_BoiteAuxLettres,&demande,sizeof(MessageDemand ↵
    eSortie),0,0) == -1 && errno == EINTR); //sans block
265
266     // si la création d'un voiturierSortie a fonctionné
267     if( (pid_Voiturier = SortirVoiture(demande.numeroPlace)) != -1 )
268     {
269         // ajout du pid au set
270         voituriers.insert(pid_Voiturier);
271     }
272 }
273 } //----- fin de Sortie
274
275

```