```cpp
/*************************************************************************
**
                            Entree  -  description
                              -------------------
debut     : 16/03/16
copyright : (C) 2016 par Nicolas Gripont
e-mail    :  nicolas.gripont@insa-lyon.fr

*************************************************************************
*/

//---------- Realisation de la tache <Entree> (fichier Entree.cpp) ---

////////////////////////////////////////////////////////////////  INCLUDE
//-------------------------------------------------------- Include systeme
#include <sys/types.h>
#include <sys/wait.h>
#include <sys/msg.h>
#include <errno.h>
#include <stdio.h>
#include <sys/shm.h>
#include <sys/sem.h>
#include <signal.h>
#include <map>
#include <unistd.h>
#include <time.h>

//------------------------------------------------------ Include personnel
#include "Entree.h"
#include "Config.h"
#include "Outils.h"

////////////////////////////////////////////////////////////////  PRIVE
//---------------------------------------------------------------- Constantes

//------------------------------------------------------------------ Types

//---------------------------------------------------- Variables statiques

static int mutex_MemoirePartageeVoitures;
static int shmId_MemoirePartageeVoitures;
static int shmId_MemoirePartageeRequetes;
static MemoirePartageeVoitures* memoirePartageeVoitures;
static MemoirePartageeRequetes* memoirePartageeRequetes;
static map<pid_t,Voiture> voituriers;


```

```
46  //------------------------------------------------------ Fonctions privees
47  static void finVoiturier(int numSignal);
48  // Mode d'emploi :
49  //
50  // Contrat :
51  //
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 = voituriers.find(pid_Voiturier);
74          Voiture voiture = it->second;
75          voituriers.erase(pid_Voiturier);
76          numeroPlace = WEXITSTATUS(statut_Voiturier);
77          AfficherPlace(numeroPlace,voiture.typeUsager,voiture.numero,voiture.arrivee);
78
79          //mettre voiture dans place memoire partagé
80          while(semop(mutex_MemoirePartageeVoitures,&prendre,1) == -1 && errno == EINTR);
81
82          memoirePartageeVoitures->voitures[numeroPlace-1] = voiture;
83
84          semop(mutex_MemoirePartageeVoitures,&vendre,1);
85
86      }
87  } //----- fin de finVoiturier
88
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 = voituriers.begin(); it
             != voituriers.end(); it++)
105          {
106              kill(it->first,SIGUSR2);
107          }
108          for(map<pid_t,Voiture>::iterator it = voituriers.begin(); it
             != voituriers.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  ////////////////////////////////////////////////////////////////
     PUBLIC
122  //---------------------------------------------------- Fonctions
     publiques
123
124
125  void Entree(TypeBarriere type, int indiceBarriere, int msgid_BAL, int
     mutex_MPR, int semSyc_MPR, int shmId_MPR, int mutex_MPV, int shmId_MPV)
126  // Algorithme :
127  //
128  {
129      //initialisation
130      pid_t pid_Voiturier;
131      TypeBarriere typeBarriere = type;
132      int msgid_BoiteAuxLettres = msgid_BAL;
133      int mutex_MemoirePartageeRequetes = mutex_MPR;
134      int semSyc_MemoirePartageeRequetes = semSyc_MPR;
135      shmId_MemoirePartageeRequetes = shmId_MPR;
136      mutex_MemoirePartageeVoitures = mutex_MPV;
137      shmId_MemoirePartageeVoitures = shmId_MPV;
138
139      memoirePartageeVoitures = (MemoirePartageeVoitures*)
         shmat(shmId_MemoirePartageeVoitures,NULL,0);
140      memoirePartageeRequetes = (MemoirePartageeRequetes*)
         shmat(shmId_MemoirePartageeRequetes,NULL,0);
141
142      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           while(msgrcv(msgid_BoiteAuxLettres,&demande,sizeof(MessageDemandeEntree),0,0) == -1 && errno == EINTR); //sans block
166
167           demande.voiture.arrivee = time(NULL);
168
169           if( (pid_Voiturier = GarerVoiture(typeBarriere)) == -1 )
170           {
171               //requete
172               sembuf prendreMutex = {(short unsigned int)0, (short)-1, (short)0};
173               sembuf vendreMutex = {(short unsigned int)0, (short)1, (short)0};
174               while(semop(mutex_MemoirePartageeRequetes,&prendreMutex,1) == -1 && errno == EINTR);
175               memoirePartageeRequetes->requetes[indiceBarriere] = demande.voiture;
176               semop(mutex_MemoirePartageeRequetes,&vendreMutex,1);
177               AfficherRequete (typeBarriere,demande.voiture.typeUsager,demande.voiture.arrivee);
178
                  DessinerVoitureBarriere(typeBarriere,demande.voiture.typeUsager);
179
180               sembuf prendreSemSync = {(short unsigned int)indiceBarriere, (short)-1, (short)0};
181
                  while(semop(semSyc_MemoirePartageeRequetes,&prendreSemSync,1) == -1 && errno == EINTR);
182               pid_Voiturier = GarerVoiture(typeBarriere);
183           }
```

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