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

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

//---------- Realisation du module <Mere> (fichier Mere.cpp) -----

////////////////////////////////////////////////////////////////  INCLUDE
//-------------------------------------------------------- Include systeme
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <sys/msg.h>
#include <sys/ipc.h>
#include <errno.h>
#include <stdio.h>
#include <sys/shm.h>
#include <sys/sem.h>
#include <signal.h>
//------------------------------------------------------ Include personnel
#include "Mere.h"
#include "Outils.h"
#include "Simulation.h"
#include "Config.h"
#include "Heure.h"
#include "Entree.h"
#include "Sortie.h"

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

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

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

//------------------------------------------------------ Fonctions privees
//static type nom ( liste de parametres )
// Mode d'emploi :
//
// Contrat :
```

```
45  //
46  // Algorithme :
47  //
48  //{
49  //} //----- fin de nom
50
51  ////////////////////////////////////////////////////////////////  PUBLIC
52  //---------------------------------------------------- Fonctions  publiques
53
54
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 semSyc_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         semSyc_MemoirePartageeRequetes = semget(ftok(PARKING_EXE,6),3,IPC_CREAT | DROITS_SEMAPHORE);
118         semctl(semSyc_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,semSyc_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,semSyc_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,semSyc_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,semSyc_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                                                                            
                waitpid(pid_EntreeBlaisePascalProf,&statut_EntreeBlaisePascalPro
                f,0);
174                                                                            
                waitpid(pid_EntreeBlaisePascalAutre,&statut_EntreeBlaisePascalAu
                tre,0);
175             waitpid(pid_EntreeGastonBerger,&statut_EntreeGastonBerger,0);
176             waitpid(pid_Sortie,&statut_Sortie,0);
177
178             //liberation ressources
179             TerminerApplication();
180             //liberation boites aux lettres
181             msgctl(msgid_FileDemandeEntree_Prof_BlaisePacal,IPC_RMID,0);
182             msgctl(msgid_FileDemandeEntree_Autre_BlaisePacal,IPC_RMID,0);
183             msgctl(msgid_FileDemandeEntree_GastonBerger,IPC_RMID,0);
184             msgctl(msgid_FileDemandeSortie_GastonBerger,IPC_RMID,0);
185             //liberation memoires partages
186             shmctl(shmId_MemoirePartageeVoitures, IPC_RMID, 0);
187             shmctl(shmId_MemoirePartageeRequetes, IPC_RMID, 0);
188             //liberation semaphores
189             semctl(mutex_MemoirePartageeVoitures, IPC_RMID, 0);
190             semctl(semSyc_MemoirePartageeRequetes, IPC_RMID, 0);
191             semctl(mutex_MemoirePartageeRequetes, IPC_RMID, 0);
192             exit(0);
193         }
194
195     } //----- fin de main
196
197
```