INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

Rim EL IDRISSI-MOKDAD
Nicolas GRIPONT

Document de Réalisation TP Multitache

Rim EL IDRISSI-MOKDAD
Nicolas GRIPONT

Makefile

```
#VARIABLES

MAKEFILE = makefile
EXE = Parking
COMP = g++
RM = rm
EDL = g++
ECHO = echo
CONFIG = Config.h
INT = Mere.h Simulation.h Entree.h Sortie.h
REAL = $(INT:.h=.cpp)
OBJ = $(INT:.h=.o)
OUTPUT = -o
RMFLAGS = -f
CPPFLAGS = -std=c++11 -Wall
EDLFLAGS = $(LIBPATH)
LIBS = -ltp -lncurses -ltcl
INCLUDES = /share/public/tp/tp-multitache
LIBPATH = -L$(INCLUDES)
INCPATH = -I$(INCLUDES)
CLEAN = clean
SOS = backup
BACKUPFILE = backup.sh
RESSOURCES =


#RULES

$(EXE) : $(OBJ)
    $(ECHO) "Edition des liens"
    $(EDL) $(OUTPUT) $(EXE) $(EDLFLAGS) $(LIBPATH) $(OBJ) $(LIBS)

%.o : %.cpp
    $(ECHO) "Compilation de <$<>"
    $(COMP) $(CPPFLAGS) $(INCPATH) -c $<


$(CLEAN) :
    $(ECHO) "nettoyage du repertoire..."
    $(RM) $(RMFLAGS) $(EXE) $(OBJ) core


$(SOS) :
    ./$(BACKUPFILE) $(MAKEFILE) $(INT) $(REAL) $(RESSOURCES)
    $(BACKUPFILE) $(CONFIG)

Mere.o : Mere.h Simulation.h Entree.h Sortie.h $(CONFIG) $(MAKEFILE)
Simulation.o : Simulation.h $(CONFIG) $(MAKEFILE)
Entree.o : Entree.h $(CONFIG) $(MAKEFILE)
Sortie.o : Sortie.h $(CONFIG) $(MAKEFILE)

```

```
1  #!/bin/bash
2
3  #Script de sauvegarde c/cpp/h/sh ou fichiers donnés en parametètre.
4  #On enregistre dans le dossier ./Backups/date
5
6  backup="Backups" #nom du dossier où on enregistre les backups
7
8  if [ ! -e $backup ]; then  #si aucun fichier/dossier nommé $backup
   existe alors on créer le dossier $backup
9      mkdir "$backup"
10 else
11     if [ ! -d $backup ]; then #si un fichier $backup existe deja et
       que se n'est pas un dossier alors erreur
12         echo "Erreur: `pwd`/$backup existe déjà et n'est pas un
           dossier." >&2
13         exit 1
14     fi
15 fi
16
17 d=`date +"%m-%d-%y %H.%M.%S"`
18 dossier="$backup/$d" #Nom du dossier ou on enregistre les fichiers
19 mkdir "$dossier" # creation de ce dossier
20
21 if [ $# -gt 0 ]; then #si on donne des fichiers à sauvegarder en
   arguments alors on enregistre que ceux-la
22     for i in $(seq 1 $#)
23     do
24         cp ${!i} "$dossier"
25     done
26 else #sinon on enregistre ceux qui sont dans $fichiers
27     fichiers=(*.cpp *.h *.c *.sh ?akefile)
28     cp ${fichiers[*]} "$dossier"
29 fi
30
```

Config

```
1  /************************************************************************
   **
2                           Config  -  Fichier de Configuration
3                             ------------------
4  debut     : 19/03/16
5  copyright : (C) 2016 par Nicolas Gripont
6  e-mail    :  nicolas.gripont@insa-lyon.fr
7
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 ////////////////////////////////////////////////////////////////        
   INCLUDE
22 //-------------------------------------------------- Interfaces         
   utilisees
23 #include "Outils.h"
24
25 //------------------------------------------------------------------        
   Constantes
26
27 #define PARKING_EXE  "./Parking" // Fichier utilisé pour créer la clé    
   publique
28 #define TEMPO 1                 // Temps en seconde pour garer une      
   voiture
29 #define TYPE_TERMINAL TypeTerminal::XTERM // Terminal utilisé
30 #define DROITS_BOITE_AU_LETTRE 0660 // Droit d'accès aux boites aux      
   lettres de demandes d'entrée
31 #define DROITS_MEMOIRE_PARTAGEE 0660 // Droit d'accès aux boites aux     
   lettres de demandes de sortie
32 #define DROITS_SEMAPHORE 0660 // Droit d'accès aux sémaphores
33
34 // Indices des entrée dans le tableau de requete de la mémoire           
   partégée contenant les requetes
35 #define INDICE_ENTREE_BLAISE_PASCALE_PROF 0
36 #define INDICE_ENTREE_BLAISE_PASCALE_AUTRE 1
37 #define INDICE_ENTREE_GASTON_BERGER 2
38
39 //------------------------------------------------------------------      
   Types
40
41 // 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 messsageDemandeEntree {
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 messsageDemandeSortie {
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 ////////////////////////////////////////////////////////////////////      ↵
   PUBLIC
78 //------------------------------------------------- Fonctions          ↵
   publiques
79
80 #endif // CONFIG_H
81
82
```

Mere

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

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

Rim EL IDRISSI-MOKDAD
Nicolas GRIPONT

Simulation

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

```
1  /*************************************************************************
   **
2                            Simulation  -  description
3                              -------------------
4  debut     : 16/03/16
5  copyright : (C) 2016 par Nicolas Gripont
6  e-mail    :  nicolas.gripont@insa-lyon.fr
7
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  ////////////////////////////////////////////////////////////////  PUBLIC
47  //---------------------------------------------------- Fonctions  publiques
48
49  static int msgid_FileDemandeEntree_Prof_BlaisePacal;
50  static int msgid_FileDemandeEntree_Autre_BlaisePacal;
51  static int msgid_FileDemandeEntree_GastonBerger;
52  static int msgid_FileDemandeSortie_GastonBerger;
53
54  void GestionClavier(int msgid_FDE_P_BP, int msgid_FDE_A_BP, int  msgid_FDE_GB, int msgid_FDS_GB)
55  // Algorithme :
56  //
57  {
58      msgid_FileDemandeEntree_Prof_BlaisePacal = msgid_FDE_P_BP;
59      msgid_FileDemandeEntree_Autre_BlaisePacal = msgid_FDE_A_BP;
60      msgid_FileDemandeEntree_GastonBerger = msgid_FDE_GB;
61      msgid_FileDemandeSortie_GastonBerger = msgid_FDS_GB;
62
63      struct sigaction action;
64      action.sa_handler = SIG_IGN ;
65      sigemptyset(&action.sa_mask);
66      action.sa_flags = 0 ;
67      sigaction(SIGUSR1,&action,NULL);
68
69      struct sigaction actionFin;
70      actionFin.sa_handler = SIG_IGN;
71      sigemptyset(&actionFin.sa_mask);
72      actionFin.sa_flags = 0 ;
73      sigaction(SIGUSR2,&actionFin,NULL);
74
75      for(;;)
76      {
77          Menu();
78      }
79
80  } //----- fin de GestionClavier
81
82
83  void Commande(char code, unsigned int valeur)
84  // Algorithme :
85  //
86  {
87      MessageDemandeEntree demandeEntree;
88      MessageDemandeSortie demandeSortie;
89
90      switch (code) {
91      case 'e':
92      case 'E':
93          exit(0);
```

```
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                 msgsnd(msgid_FileDemandeEntree_Prof_BlaisePacal,&demandeEntree,sizeof(MessageDemandeEntree),0);
105                 break;
106             case 2: // Gaston Berger
107                 msgsnd(msgid_FileDemandeEntree_GastonBerger,&demandeEntree,sizeof(MessageDemandeEntree),0);
108                 break;
109             default:
110                 break;
111             }
112             break;
113         case 'a':
114         case 'A':
115             demandeEntree.voiture.arrivee = time(NULL);
116             demandeEntree.voiture.numero = GetNumeroVoiture();
117             demandeEntree.voiture.typeUsager = TypeUsager::AUTRE;
118             demandeEntree.type = 1;
119
120             switch (valeur) {
121             case 1: // Blaise Pascal
122                 msgsnd(msgid_FileDemandeEntree_Autre_BlaisePacal,&demandeEntree,sizeof(MessageDemandeEntree),0);
123                 break;
124             case 2: // Gaston Berger
125                 msgsnd(msgid_FileDemandeEntree_GastonBerger,&demandeEntree,sizeof(MessageDemandeEntree),0);
126                 break;
127             default:
128                 break;
129             }
130             break;
131             break;
132         case 's':
133         case 'S':
134             demandeSortie.numeroPlace = valeur;
135             demandeSortie.type = 1;
136             msgsnd(msgid_FileDemandeSortie_GastonBerger,&demandeSortie,sizeof(MessageDemandeSortie),0);
137             break;
```

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

Entree

```
1  /*************************************************************************
   **
2                            Entree  -  description
3                              -------------------
4  debut     : 16/03/16
5  copyright : (C) 2016 par Nicolas Gripont
6  e-mail    :  nicolas.gripont@insa-lyon.fr
7
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 ////////////////////////////////////////////////////////////////       
   INCLUDE
21 //-------------------------------------------------- Interfaces        
   utilisees
22 #include "Outils.h"
23
24 //------------------------------------------------------------------        
   Constantes
25
26 //------------------------------------------------------------------    
   Types
27
28 ////////////////////////////////////////////////////////////////       
   PUBLIC
29 //-------------------------------------------------- Fonctions        
   publiques
30
31
32 void Entree(TypeBarriere type, int indiceBarriere, int msgid_BAL, int    
   mutex_MPR, int semSyc_MPR, int shmId_MPR, int mutex_MPV, int shmId_MPV);
33 // Mode d'emploi :
34 //
35 // Contrat :
36 //
37
38
39 #endif // ENTREE_H
40
41
```

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

```
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,voit ↵
          ure.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
```

Sortie

```
1  /************************************************************************
   **
2                            Sortie  -  description
3                              -------------------
4  debut     : 16/03/16
5  copyright : (C) 2016 par Nicolas Gripont
6  e-mail    :  nicolas.gripont@insa-lyon.fr
7
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 ////////////////////////////////////////////////////////////////       
   INCLUDE
21 //-------------------------------------------------- Interfaces        
   utilisees
22
23 //------------------------------------------------------------------      
   Constantes
24
25 //------------------------------------------------------------------      
   Types
26
27 ////////////////////////////////////////////////////////////////       
   PUBLIC
28 //-------------------------------------------------- Fonctions         
   publiques
29
30
31 void Sortie(int msgid_BAL, int mutex_MPR, int semSyc_MPR, int shmId_MPR,
32             int mutex_MPV, int shmId_MPV);
33 // Mode d'emploi : Fonction permettant de gérer es sortie du parking
34 //
35 // Contrat : les différents paramètres doivent être correctes.
36 // (les id des IPCs)
37 //
38
39 #endif // SORTIE_H
40
41
```

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

```
    voiturierSortie
46                                  // en cours d'exécution
47
48  //------------------------------------------------------ Fonctions privees
49
50  static void finVoiturier(int numSignal);
51  // Mode d'emploi : fonction permettant de gérer la fin d'un processus
52  // SortieVoiture.
53  // A connecter au signal SIGCHLD
54  //
55  // Contrat : l'accès à la mémoire partagés requêtes et voitures
56  // doit être possible (memoirePartageeRequetes et memoirePartageeVoitures)
57  // L'accès aux différents sémaphores doit être possible
58  // (mutex_MemoirePartageeVoitures,mutex_MemoirePartageeRequetes
59  // semSyc_MemoirePartageeRequetes)
60  //
61
62  static void fin(int numSignal);
63  // Mode d'emploi : fonction permettant de Terminer le processus.
64  // A connecter au signal SIGUSR2
65  //
66  // Contrat : l'accès à la mémoire partagés requêtes et voitures
67  // doit être possible (memoirePartageeRequetes et memoirePartageeVoitures)
68
69
70  static int choixRequete();
71  // Mode d'emploi : Fonction permettant de choisir la requete d'entrée
72  // à traiter. Renvoie l'indice de la requete à traiter (dans la
73  // memoirePartageeRequetes) ou -1 si aucune requete.
74  //
75  // Contrat : l'accès à la mémoire partagés requêtes doit être possible
76  // (memoirePartageeRequetes)
77
78  static bool isV1Prio(Voiture v1, Voiture v2);
79  // Mode d'emploi : Renvoie true si v1 est prioritaire par rapport à v2,
80  // false sinon.
81  //
82  // Contrat : aucun
83
84
85
86  static void finVoiturier(int numSignal)
87  // Algorithme :
88  //
89  {
90      // PHASE MOTEUR :
91      sembuf prendreMutex = {(short unsigned int)0, (short)-1, (short)0};
92      sembuf vendreMutex = {(short unsigned int)0, (short)1, (short)0};
93      pid_t pid_Voiturier;
94      int statut_Voiturier;
95      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
             accéder (être seul à y accéder)
106          while(semop(mutex_MemoirePartageeVoitures,&prendreMutex,1) ==
             -1 && errno == EINTR);
107
108          //recupération de la voiture sortie
109          voiture = memoirePartageeVoitures->voitures[numeroPlace-1];
110          //on efface la voiture sortie de la mémoire partagée
111          memoirePartageeVoitures->voitures[numeroPlace-1] =
             {TypeUsager::AUCUN,0,0};
112          //on rend l'accès à la mémoire partagée voitures
113          semop(mutex_MemoirePartageeVoitures,&vendreMutex,1);
114
115          //on efface les infos de la voiture sortie de la zone etat
116          Effacer((TypeZone)numeroPlace);
117          //on affiche les informations de la voiture sortie dans la
             zone sortie
118
             AfficherSortie(voiture.typeUsager,voiture.numero,voiture.arrivee
             ,time(NULL));
119          //on enleve le pid de du voiturierSortie terminé du set
120          voituriers.erase(pid_Voiturier);
121
122          //on prend le mutex memoire partagée requetes pour pouvoir y
             accéder (être seul à y accéder)
123          while(semop(mutex_MemoirePartageeRequetes,&prendreMutex,1) ==
             -1 && errno == EINTR);
124
125          //s'il y a une requete a traiter
126          if( (indiceRequete = choixRequete()) != -1 )
127          {
128              //on efface la requete à traiter de la mémoire partagée
                 requetes
129              memoirePartageeRequetes->requetes[indiceRequete] =
                 {TypeUsager::AUCUN,0,0};
130              //on efface la requete à traiter de la zone requetes
131              Effacer((TypeZone)(TypeZone::REQUETE_R1 + indiceRequete));
132              //on libere le processus entrée en donnant un jeton au
                 semSync correspondant à la requête
133              sembuf vendreSemSync = {(short unsigned int)indiceRequete,
                 (short)1, (short)0};
134
                 while(semop(semSyc_MemoirePartageeRequetes,&vendreSemSync,1)
                  == -1 && errno == EINTR);
135          }
```

```
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
                la fin
150             //d'un voiturierSortie
151             struct sigaction action;
152             action.sa_handler = SIG_IGN ;
153             sigemptyset(&action.sa_mask);
154             action.sa_flags = 0 ;
155             sigaction(SIGCHLD,&action,NULL);
156
157             //on envoie le signal SIGUSR2 a tous les voiturierSortie en
                cours d'exécution
158             for(set<pid_t>::iterator it = voituriers.begin(); it !=
                voituriers.end(); it++)
159             {
160                 kill(*it,SIGUSR2);
161             }
162             //on attend la fin des voiturierSortie à qui on a envoyé SIGUSR2
163             for(set<pid_t>::iterator it = voituriers.begin(); it !=
                voituriers.end(); it++)
164             {
165                 waitpid(*it,NULL,0);
166             }
167
168             //On se détache des mémoires partagées
169             shmdt(memoirePartageeVoitures);
170             shmdt(memoirePartageeRequetes);
171
172             //On quitte le processus
173             exit(0);
174         }
175     } //----- fin de fin
176
177
178     static bool isV1Prio(Voiture v1, Voiture v2)
179     {
180         if(v1.typeUsager == v2.typeUsager) { return (v1.arrivee <=
            v2.arrivee); }
181         if(v1.typeUsager == TypeUsager::PROF && v2.typeUsager ==
            TypeUsager::AUTRE) { return true; }
182         if(v1.typeUsager == TypeUsager::AUTRE && v2.typeUsager ==
            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 = memoirePartageeRequetes->requetes[INDICE_ENTREE_BLAISE_PASCALE_PROF];
193         Voiture v2 = memoirePartageeRequetes->requetes[INDICE_ENTREE_BLAISE_PASCALE_AUTRE];
194         Voiture v3 = memoirePartageeRequetes->requetes[INDICE_ENTREE_GASTON_BERGER];
195
196         if(isV1Prio(v1,v2) && isV1Prio(v1,v3) && v1.typeUsager != TypeUsager::AUCUN)
197         {
198             return 0;
199         }
200         if(isV1Prio(v2,v1) && isV1Prio(v2,v3) && v2.typeUsager != TypeUsager::AUCUN)
201         {
202             return 1;
203         }
204         if(isV1Prio(v3,v2) && isV1Prio(v3,v1) && v3.typeUsager != TypeUsager::AUCUN)
205         {
206             return 2;
207         }
208
209         return -1;
210     } //----- fin de choixRequete
211
212     ////////////////////////////////////////////////////////////////  PUBLIC
213     //---------------------------------------------------- Fonctions publiques
214
215
216     void Sortie(int msgid_BAL, int mutex_MPR, int semSyc_MPR, int shmId_MPR, int mutex_MPV, int shmId_MPV)
217     // Algorithme :
218     //
219     {
220         pid_t pid_Voiturier;
221         int msgid_BoiteAuxLettres;
222         MessageDemandeSortie demande;
223
224         // PHASE INITIALISATION
225         //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         semSyc_MemoirePartageeRequetes = semSyc_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 fonctionnée
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 
```