```cpp
  1 //-------------------------------------------------------------------------------------
  2 /*! ¥file
  3     ¥brief  4-legged robot simulator - client
  4     ¥author Akihiko Yamaguchi
  5     ¥date   Mar.13 2007  */
  6 //-------------------------------------------------------------------------------------
  7 #include <iostream> // TODO デバッグが終了しだい削除
  8 #include <cstdlib>
  9 #include <cstdio>
 10 #include <cstring>
 11 #include <unistd.h>
 12 #include <sys/types.h>
 13 #include <sys/socket.h>
 14 #include <sys/un.h>
 15 //-------------------------------------------------------------------------------------
 16 #include "protocol.h"
 17 //-------------------------------------------------------------------------------------
 18 #include <octave/config.h>
 19 #include <octave/Matrix.h>
 20 //-------------------------------------------------------------------------------------
 21 #ifdef OUTPUT_OCT
 22 #include <octave/oct.h>
 23 #endif
 24 //-------------------------------------------------------------------------------------
 25 static int  client_file_descriptor(-1);
 26 static int  JOINT_NUM(0);
 27 static int  JOINT_STATE_DIM (0);  // 関節状態の次元
 28 static int  BASE_STATE_DIM (0);  // ベース状態の次元
 29 //-------------------------------------------------------------------------------------
 30 static double *joint_state (NULL);
 31 static double *base_state (NULL);
 32 static ColumnVector  jState(0), bState(0);
 33 //-------------------------------------------------------------------------------------
 34 class __inner_destructor
 35 {
 36   __inner_destructor(void) {};
 37   ~__inner_destructor(void)
 38     {
 39       if (joint_state!=NULL) {delete[] joint_state; joint_state=NULL;}
 40       if (base_state!=NULL)  {delete[] base_state; base_state=NULL;}
 41     };
 42 };
 43 //-------------------------------------------------------------------------------------
 44
 45
 46 inline ColumnVector get_joint_state (void);
 47 inline ColumnVector get_base_state (void);
 48
 49 using namespace std;
 50
 51 //! check return
 52 void chret (int ret)
 53 {
 54   if (ret<0)
 55   {
 56     close (client_file_descriptor);
 57     client_file_descriptor = -1;
 58     exit(1);
 59   }
 60 }
 61
 62 //! check the client_file_descriptor
 63 void chfd (void)
 64 {
 65   if(client_file_descriptor<0)
 66   {
 67     cerr<<"error! the connection was already terminated."<<endl;
 68     exit(1);
 69   }
 70 }
 71
 72 bool setup_client (void)
 73   // ref. http://www.ueda.info.waseda.ac.jp/~toyama/network/example1.html
 74 {
 75   struct sockaddr_un    addr;
 76
 77   // ソケットを作成．  UNIX ドメイン，ストリーム型
 78   if ((client_file_descriptor = socket (PF_UNIX, SOCK_STREAM, 0)) < 0)
 79   {
 80     perror("socket");
 81     exit(1);
 82   }
 83
 84   bzero ((char *)&addr, sizeof(addr));
 85
 86   // ソケットの名前を代入
 87   addr.sun_family = AF_UNIX;
 88   strcpy (addr.sun_path, SOCK_NAME);
 89
 90   // サーバと接続を試みる．サーバ側で bind & listen の発行が終っている必要がある
 91   if (connect (client_file_descriptor, (struct sockaddr *)&addr,
 92        sizeof(addr.sun_family) + strlen(SOCK_NAME)) < 0)
 93   {
 94     perror("connect");
 95     cerr<<"-> maybe the server four-legged.exe is not running."<<endl;
 96     return false; //exit(1);
 97   }
 98   return true;
 99 }
100
101 ColumnVector get_torque (void)
102 {
103   static bool init(true);
104   static const double kp=100.0, kd=2.0;
105   static ColumnVector target(JOINT_NUM,0.0);
```

```
106    static const double MaxTorque(100.0);   // [Nm]
107    if(init)
108    {
109      const double q1=-0.25*M_PI, q2=0.5*M_PI;
110      for(int i(0);i<8;i+=2) target(i)=q1;
111      for(int i(1);i<8;i+=2) target(i)=q2;
112      init = false;
113    }
114    ColumnVector u (JOINT_NUM,0.0);   // 制御入力（トルク）
115    ColumnVector jstate (get_joint_state());   // 現在の関節状態
116    for (int i(0);i<8;++i)
117    {
118      u(i)=kp*(target(i)-jstate(i))-kd*jstate(8+i);
119      if (u(i)>MaxTorque)  u(i)=MaxTorque;
120      else if (u(i)<-MaxTorque)  u(i)=-MaxTorque;
121    }
122
123    return u;
124 }
125 //-----------------------------------------------------------------------------
126
127 inline void start_simulation (int window_width, int window_height)
128 {
129    chfd();
130    TXData data;
131    data.command = ORS_SET_WINDOWSIZE;
132    data.step = 0;
133    data.ivalue = window_width;
134    chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
135    data.step = 1;
136    data.ivalue = window_height;
137    chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
138    data.command = ORS_START_SIM;
139    chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
140 }
141
142 inline void stop_simulation (void)
143 {
144    chfd();
145    TXData data;
146    data.command = ORS_STOP_SIM;
147    chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
148    close(client_file_descriptor);
149    client_file_descriptor=-1;
150 }
151
152 inline void step_simulation (const ColumnVector &u, const double &time_step)
153 {
154    chfd();
155    TXData data;
156    data.command = ORS_SET_TORQUE;
157    for (int j(0); j<JOINT_NUM; ++j)
158    {
159      data.step = j;
160      data.dvalue = u(j);
161      chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
162    }
163    data.command = ORS_STEP_SIM;
164    data.dvalue = time_step;
165    chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
166 }
167
168 inline void reset_simulation (void)
169 {
170    chfd();
171    TXData data;
172    data.command = ORS_RESET_SIM;
173    chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
174 }
175
176 inline ColumnVector get_joint_state (void)
177 {
178    chfd();
179    if (JOINT_STATE_DIM<=0)  return ColumnVector(0);
180    if (jState.dim1() != JOINT_STATE_DIM)  jState.resize(JOINT_STATE_DIM);
181    TXData data;
182    data.command = ORS_GET_JOINT_STATE;
183    chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
184    chret (read (client_file_descriptor, (char*)joint_state, sizeof(double)*JOINT_STATE_DIM));
185    // cerr<<"c-joint1= ";for(int j(0);j<JOINT_STATE_DIM;++j)cerr<<" "<<joint_state[j];cerr<<endl;
186    for (int j(0); j<JOINT_STATE_DIM; ++j)  jState(j)=joint_state[j];
187    // cerr<<"c-joint2= "<<jState.transpose()<<endl;
188    return jState;
189 }
190 inline ColumnVector get_base_state (void)
191 {
192    chfd();
193    if (BASE_STATE_DIM<=0)  return ColumnVector(0);
194    if (bState.dim1() != BASE_STATE_DIM)  bState.resize(BASE_STATE_DIM);
195    TXData data;
196    data.command = ORS_GET_BASE_STATE;
197    chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
198    chret (read (client_file_descriptor, (char*)base_state, sizeof(double)*BASE_STATE_DIM));
199    for (int j(0); j<BASE_STATE_DIM; ++j)  bState(j)=base_state[j];
200    return bState;
201 }
202
203 inline void draw_world (void)
204 {
205    chfd();
206    TXData data;
207    data.command = ORS_DRAW_WORLD;
208    chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
209 }
210
```

```cpp
211  inline int get_joint_num (void)
212  {
213    chfd();
214    TXData data;
215    data.command = ORS_GET_JOINT_NUM;
216    chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
217    chret (read (client_file_descriptor, (char*)&JOINT_NUM, sizeof(JOINT_NUM)));
218    cerr<<"joint-num = "<<JOINT_NUM<<endl;
219    return JOINT_NUM;
220  }
221
222  inline int get_joint_state_dim (void)
223  {
224    chfd();
225    TXData data;
226    data.command = ORS_GET_JSTATE_DIM;
227    chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
228    chret (read (client_file_descriptor, (char*)&JOINT_STATE_DIM, sizeof(JOINT_STATE_DIM)));
229    cerr<<"joint-state-dim = "<<JOINT_STATE_DIM<<endl;
230    if (joint_state!=NULL) {delete[] joint_state; joint_state=NULL;}
231    joint_state = new double[JOINT_STATE_DIM];
232    return JOINT_STATE_DIM;
233  }
234
235  inline int get_base_state_dim (void)
236  {
237    chfd();
238    TXData data;
239    data.command = ORS_GET_BSTATE_DIM;
240    chret (write (client_file_descriptor, (char*)&data, sizeof(data)));
241    chret (read (client_file_descriptor, (char*)&BASE_STATE_DIM, sizeof(BASE_STATE_DIM)));
242    cerr<<"base-state-dim = "<<BASE_STATE_DIM<<endl;
243    if (base_state!=NULL) {delete[] base_state; base_state=NULL;}
244    base_state = new double[BASE_STATE_DIM];
245    return BASE_STATE_DIM;
246  }
247
248  #ifdef OUTPUT_OCT
249  DEFUN_DLD (startSimulation, args, ,
250    "int startSimulation(int window_width, int window_height).")
251  {
252    if(!setup_client()) return octave_value(1);
253    start_simulation(args(0).double_value(), args(1).double_value());
254    get_joint_num();
255    get_joint_state_dim();
256    get_base_state_dim();
257    return octave_value(0);
258  }
259
260  DEFUN_DLD (stopSimulation, args, ,
261    "void stopSimulation(void).")
262  {
263    stop_simulation();
264    return octave_value();
265  }
266
267  DEFUN_DLD (stepSimulation, args, ,
268    "void stepSimulation(const ColumnVector &u, const dReal &time_step).")
269  {
270    ColumnVector u(args(0).vector_value());
271    step_simulation (u, args(1).double_value());
272    return octave_value();
273  }
274
275  DEFUN_DLD (resetSimulation, args, ,
276    "void resetSimulation(void).")
277  {
278    reset_simulation();
279    return octave_value();
280  }
281
282  DEFUN_DLD (drawWorld, args, ,
283    "void drawWorld(void).")
284  {
285    draw_world();
286    return octave_value();
287  }
288
289  DEFUN_DLD (getJointState, args, ,
290    "ColumnVector getJointState(void).")
291  {
292    ColumnVector state (get_joint_state());
293    return octave_value (state);
294  }
295
296  DEFUN_DLD (getBaseState, args, ,
297    "ColumnVector getBaseState(void).")
298  {
299    ColumnVector state (get_base_state());
300    return octave_value (state);
301  }
302  #endif
303
304
305  int main (int argc, char **argv)
306  {
307    if(!setup_client()) return 1;
308    start_simulation(400,400);
309    get_joint_num();
310    get_joint_state_dim();
311    get_base_state_dim();
312    while(1)
313    {
314      step_simulation (get_torque(), 0.001);
315      draw_world();
```

```
316    }
317
318    close (client_file_descriptor);
319    return 0;
320 }
321 //--------------------------------------------------------------------------------
322
323
```