Инсрукция по запуску расчетного мудля на кластере "Сергей Королев"

Д.Понятовский <dponyatov@gmail.com>

18 января 2016 г.

Оглавление

Установка

Выполнить в домашнем каталоге[1]

```
git clone -o github https://github.com/ponyatov/pij.git
cd pij/pij2d
make EXE=
```

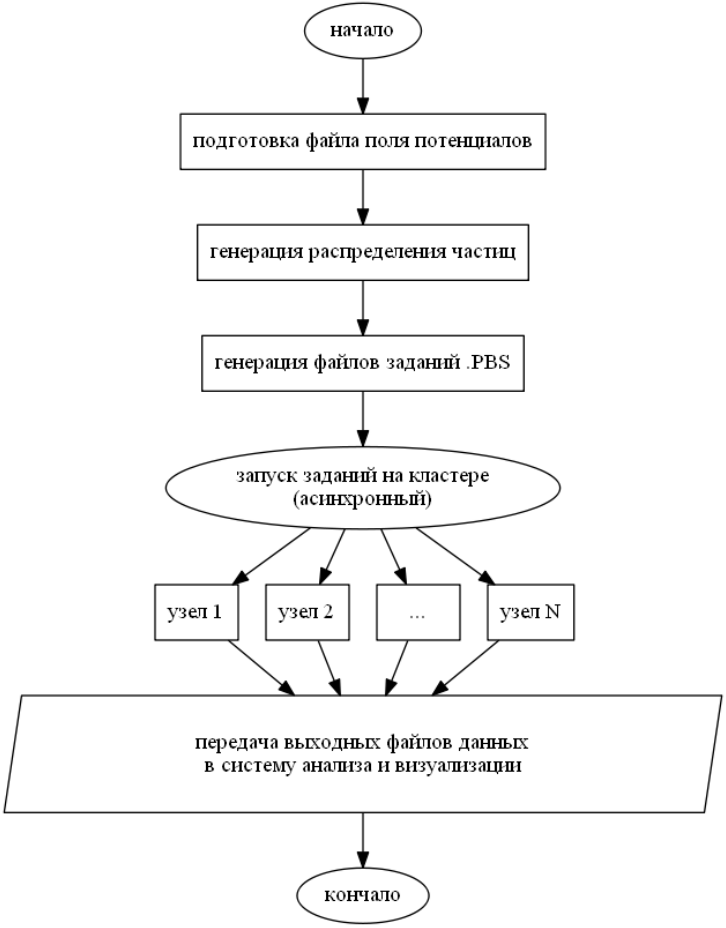
[1] в разделяемой между узлами файловой системе

Для сборки под $\boxplus Windows$[2]:

```
mingw32-make EXE=.exe
```

[2] для тестирования

Алгоритм



алгоритм на кластере "Сергей Королев"

Параметры командной строки

Параметры командной строки[3]:

```
./pij[.exe] [V] [Qm] [Alpha] [r]
```

`V`	V	модуль скорости
`Qm`	Q_m	отношние заряд/масса
`Alpha`	α	угол влета частицы
`r`	r	r

[3] см. исходный код функции main()

Глава 1

Исходный кот

1.1 Скрипт сборки

Makefile

```
1
2 #MPICH2 = C:\MinGW\MPICH2
3
4 MODULE = pij2d
5
6 .PHONY: exec
7 exec: Fi.txt
8 Fi.txt: ./$(MODULE)$(EXE)
9     ./$(MODULE)$(EXE) 1000 1 0.1 2 < Fi.txt > $(MODULE).log
10
11 C = cpp.cpp lex.yy.c
```

```
H = hpp.hpp
CXXFLAGS += -I. -std=gnu++11
#-I$(MPICH2)/include -L$(MPICH2)/lib
./$(MODULE)$(EXE): $(C) $(H)
	$(CXX) $(CXXFLAGS) -o $@ $(C)
#	-lmpi
lex.yy.c: lpp.lpp
	flex $<

NOW = $(shell date +%Y%m%d%H%M)
.PHONY: rar
rar: $(NOW).rar
$(NOW).rar: test.files Fi.txt
	rar a -m5 -o- -r $@ @test.files
```

1.2 Лексер: чтение файла потенциалов

lpp.lpp

```
%{
#include "hpp.hpp"

int item=0;
int R=0,Rlimit;
int X=0,Xlimit;

double Fi[Rmax][Xmax];
```

```
 9
10 %}
11 %option noyywrap
12 S [\+\-]?
13 N [0-9]+
14 %%
15 {S}{N}(\.{N})?       {
16     item++;
17     if (item==1) { Rlimit=atoi(yytext); cout << "R limit:\t" << Rlimit <<"\n"; }
18     if (item==2) { Xlimit=atoi(yytext); cout << "X limit:\t" << Xlimit <<"\n";
19         X=R=0; assert(Rlimit<Rmax); assert(Xlimit<Xmax);
20     cout << "\nFi[] phield data:"; }
21     if (item>2) { Fi[R][X++] = atof(yytext); }
22     }
23
24 [\r\n]+      { if (item>2) { X=0; R++; } }
25
26 <<EOF>> {
27     for (int r=0;r<=Rlimit;r++) {
28         cout << "\n\n" << r <<": ";
29         for (int x=0;x<Xlimit;x++) {
30             cout << Fi[r][x] << " ";
31     }}
32     yyterminate();
33 }
34
35 .    {}
36 %%
```

1.3 Расчетный модуль (C^{++})

hpp.hpp

```
1 #ifndef _H_PIJ2D
2 #define _H_PIJ2D
3                                                   // log file name
4 #define TE "te.log"
5
6 //#include <mpi.h>
7                                                   // std. includes
8 #include <iostream>
9 #include <fstream>
10 #include <iomanip>
11 #include <cmath>
12 #include <cfloat>
13 #include <cstdlib>
14 #include <cstdio>
15 #include <cassert>
16 using namespace std;
17
18 extern int doit();                                // calculation procedure
19
20 extern int yylex();                               // lexer for phield file load
21 #define Rmax 20+1
22 #define Xmax 90+1
23 extern double Fi[Rmax][Xmax];                     // phield data
24
25 #endif // _H_PIJ2D
```

cpp.cpp

```cpp
#include "hpp.hpp"                              // === adaptation from Pascal prototype ===

double V  ;                                     // V:=1000;
double Qm ;                                     // Qm:=StrToFloat(Form1.Edit1.Text);
double Alpha ;                                  // Alpha:=0;
double r ;                                      // r:=StrToFloat(Form1.Edit3.Text)/1000;

int doit() {

ofstream te(TE);                                // append(te);

double Vz = V*cos(Alpha);                       // Vz:=V*Cos(Alpha);
double Vr = V*sin(Alpha);                       // Vr:=V*Sin(Alpha);

double dt = 1E-6;                               // dt:=1E-6;
double t=0;                                     // t:=0;
double z=0;                                     // z:=0;
double R=0;
double Z0=0;
double Ez,Er;

int i,j,n,m,x,y;

do {                                            // repeat

Z0=z;                                           // Z0:=Z;
```

```
27
28 do {                                          // Repeat
29     if (Z0*1000>90)                          // If Z0*1000>90 then
30         Z0=Z0-0.090;                         // Z0:=Z0-0.090;
31 } while (!(Z0<0.09));                        // Until Z0<0.09;
32
33 y = trunc(Z0*1000);                          // Y:=Trunc(Z0*1000);
34 x = 10-trunc(R*1000);                        //  X:=10-Trunc(R*1000);//X:=11;
35
36 Ez = (Fi[x][y]-Fi[x][y+1])*8000/0.001;      // Ez:=(Fi[x,y]-Fi[x,y+1])*8000/0.001;
37 Er = -1*(Fi[x][y]-Fi[x+1][y])*8000/0.001;   // Er:=-1*(Fi[x,y]-Fi[x+1,y])*8000/0.001;
38 z =z+Vz*dt+Ez*Qm*dt*dt/2;                    // Z:=Z+Vz*dt+Ez*qm*dt*dt/2;
39 Vz =Vz+Ez*Qm*dt;                            // Vz:=Vz+Ez*qm*dt;
40 R =R+Vr*dt+Er*Qm*dt*dt/2;                   // R:=R+Vr*dt+Er*qm*dt*dt/2;
41 Vr =Vr+Er*Qm*dt;                            // Vr:=Vr+Er*qm*dt;
42 t =t+dt;                                    // t:=t+dt;
43
44 te << z;                                    // write(te,z:8:6);
45 te << "␣";                                  // write(te,' ');
46 te << r;                                    // writeln(te,r:8:6);
47 te << "\n";                                 // closefile(te);
48
49 } while (!(                                 // until
50             (z>2.54)||(R>=0.0099)           // (z>2.54)or(R>=0.0099);
51             ));
52
53 return 0;
54 }
```

```
55
56 int main (int argc, char *argv[]) {
57     // command line processing
58     assert (argc==4+1); // pij.exe [V] [Qm] [Alpha] [r]
59     V     = atof(argv[1]); assert(V    >0); cout << "V:\t\t\t" << V << "\n";
60     Qm    = atof(argv[2]); assert(Qm   >0); cout << "Qm:\t\t\t" << Qm << "\n";
61     Alpha = atof(argv[3]); assert(Alpha>0); cout << "Alpha:\t\t" << Alpha << "\n";
62     r     = atof(argv[4])/1000; assert(r    >0); cout << "r:\t\t\t" << r << "\n";
63     // Fi.txt fielf data parsing
64     while (yylex());    // Fi.txt parser loop from stdin
65     // compute tracks
66     return doit();
67 }
```