# OpenCL中的数据划分

OpenCL程序中，在对数据进行划分时，需要设置global size和local size，global size要为local size的整数倍，而local size最好为32的整数倍。如果要处理的数据长宽不是local size的整数倍时，在核函数中要对边界进行判定，超过边界时直接return返回，所以在核函数参数中，最好有一个指示数组大小的参数。

以下程序演示了数据划分的方法：

|  |
| --- |
| #include "stdafx.h"  #include <CL\cl.hpp>  #include <fstream>  #include <iostream>  using namespace std;  using namespace cl;  cl::Program CreateProgram(std::string filename)  {  vector<Platform> platforms;  Platform::get(&platforms);  auto platform = platforms[2];  vector<Device> devices;  platform.getDevices(CL\_DEVICE\_TYPE\_GPU, &devices);  Device device = devices.front();  Context context(devices);  ifstream helloWorldFile(filename.c\_str());  std::string src(istreambuf\_iterator<char>(helloWorldFile), (istreambuf\_iterator<char>()));  Program::Sources sources(1, std::make\_pair(src.c\_str(), src.length() + 1));  Program program(context, sources);  auto err = program.build("-cl-std=CL1.2");  return program;  }  #define NUM\_WORK\_ITEMS 16  #define BUFFER\_SIZE 20  int main()  {  auto program = CreateProgram("improc.cl");  auto context = program.getInfo<CL\_PROGRAM\_CONTEXT>();  auto devices = context.getInfo<CL\_CONTEXT\_DEVICES>();  auto& device = devices.front();  float host\_out[BUFFER\_SIZE];  Buffer outBuf(context, CL\_MEM\_WRITE\_ONLY | CL\_MEM\_USE\_HOST\_PTR, sizeof(float) \* BUFFER\_SIZE, (void\*)host\_out);  Kernel kernel(program, "idex\_test");  kernel.setArg(0, outBuf);  kernel.setArg(1, BUFFER\_SIZE);  CommandQueue queue(context, device);  //此处计算一个大于BUFFER\_SIZE的最小NUM\_WORK\_ITEMS的位数作为global\_size  int global\_size = (BUFFER\_SIZE + NUM\_WORK\_ITEMS - 1) / NUM\_WORK\_ITEMS \* NUM\_WORK\_ITEMS;  int err = queue.enqueueNDRangeKernel(kernel, NullRange, NDRange(global\_size), NDRange(NUM\_WORK\_ITEMS));  queue.enqueueReadBuffer(outBuf, CL\_TRUE, 0, sizeof(float) \* BUFFER\_SIZE, host\_out);  for (int i = 0;i < BUFFER\_SIZE;i++)  {  cout << host\_out[i] << endl;  }  cin.get();  return 0;  } |

核函数：

|  |
| --- |
| \_\_kernel void idex\_test(\_\_global float\* d\_out, int len)  {  int global\_idx = get\_global\_id(0);  if (global\_idx > len)return;  int global\_sz = get\_global\_size(0);  printf("idx = %d, size = %d \n", global\_idx, global\_sz);  d\_out[global\_idx] = global\_idx + ((float)global\_sz) / 1000;  } |