Лекция 7

Библиотека ***thrust***

- Контейнеры *host_vector* и *device_vector.*
- Алгоритмы *thrust.*
- Преобразование указателей и комбинированный код.
- Алгоритм *transform* и функторы.
- Кортежи и *zip*-итератор.

```cpp
#include <thrust/host_vector.h>
#include <thrust/device_vector.h>
#include <thrust/generate.h>
#include <cstdlib>
int main(void){
  thrust::host_vector<int> h(1 << 24);
  thrust::generate(h.begin(), h.end(), rand);
  thrust::device_vector<int> d = h;

  return 0;
}
```

Line: 7 of 11 Col: 21 LINE INS 0.cu UTF

```
malkov@linux-5002:~/WORKSHOP/CUDA EXERCISE/THRUST/tests> nvcc 0.cu -o 0
malkov@linux-5002:~/WORKSHOP/CUDA EXERCISE/THRUST/tests> nvprof ./0
==11098== NVPROF is profiling process 11098, command: ./0
==11098== Profiling application: ./0
==11098== Profiling result:
Time(%)      Time     Calls       Avg       Min       Max  Name
100.00%  11.104ms         1  11.104ms  11.104ms  11.104ms  [CUDA memcpy HtoD]

==11098== API calls:
Time(%)      Time     Calls       Avg       Min       Max  Name
 86.14%  73.273ms         1  73.273ms  73.273ms  73.273ms  cudaMalloc
 13.09%  11.137ms         1  11.137ms  11.137ms  11.137ms  cudaMemcpyAsync
  0.36%  302.51us        83  3.6440us     291ns  128.30us  cuDeviceGetAttribute
  0.33%  278.22us         1  278.22us  278.22us  278.22us  cudaFree
  0.05%  40.411us         1  40.411us  40.411us  40.411us  cuDeviceTotalMem
  0.04%  32.067us         1  32.067us  32.067us  32.067us  cuDeviceGetName
  0.00%  1.8530us         2     926ns     441ns  1.4120us  cuDeviceGetCount
  0.00%     891ns         2     445ns     285ns     606ns  cuDeviceGet
malkov@linux-5002:~/WORKSHOP/CUDA EXERCISE/THRUST/tests>
```

```
#include <thrust/host_vector.h>
#include <thrust/device_vector.h>
#include <thrust/generate.h>
#include <thrust/sort.h>
#include <thrust/copy.h>
#include <cstdlib>

int main(void){
  thrust::host_vector<int> h(1 << 8);
  thrust::generate(h.begin(), h.end(), rand);

  thrust::device_vector<int> d = h;

  thrust::sort(d.begin(), d.end());

  thrust::copy(d.begin(), d.end(), h.begin()); //h=d;

  for(int i=0;i<(1<<8);i++)
    printf("%d\n",h[i]);

  return 0;
}
```

```
#include <thrust/host_vector.h>
#include <thrust/device_vector.h>
#include <thrust/fill.h>
#include <thrust/copy.h>
#include <cstdio>

__global__ void gTest(float* d){
  int idx=threadIdx.x+blockDim.x*blockIdx.x;
  d[idx]+=(float)idx;
}

int main(void){
  float *raw_ptr;

  thrust::host_vector<float> h(1 << 8);
  thrust::fill(h.begin(), h.end(), 3.1415f);
  thrust::device_vector<float> d = h;
  raw_ptr = thrust::raw_pointer_cast(&d[0]);//d.data());

  gTest<<<4,64>>>(raw_ptr);
  cudaDeviceSynchronize();

  thrust::copy(d.begin(), d.end(), h.begin());
  for(int i=0;i<(1<<8);i++)
    printf("%g\n",h[i]);

  return 0;
}
```

```
#include <thrust/host_vector.h>
#include <thrust/device_vector.h>
//#include <thrust/fill.h>
#include <thrust/transform.h>
#include <thrust/sequence.h>
//#include <thrust/generate.h>
#include <thrust/execution_policy.h>
#include <cstdio>
//#include <thrust/for_each.h>
#include <cmath>

struct range_functor
{
  float h;
  range_functor(float _h):h(_h){}
  __host__ __device__
  float operator()(float x){
    return h*x;
  }
};
struct sin_functor
{
  __host__ __device__
  float operator()(float x){
    return sin(x);
  }
};
```

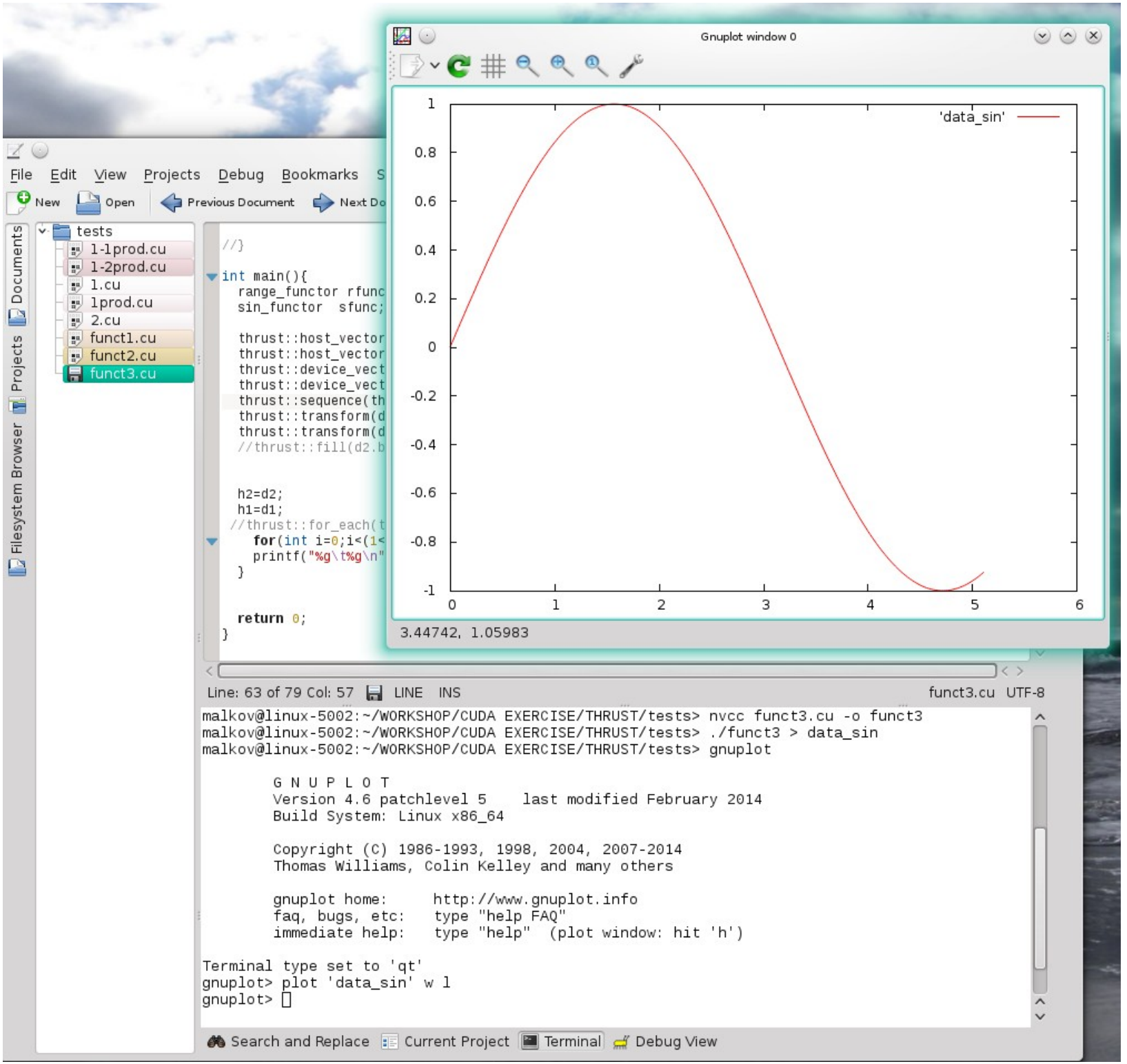
```
int main(){
  range_functor rfunc(0.02);
  sin_functor  sfunc;

  thrust::host_vector<float> h1(1 << 8);
  thrust::host_vector<float> h2(1 << 8);
  thrust::device_vector<float> d1(1 << 8);// = h1;
  thrust::device_vector<float> d2(1 << 8);// = h2;
  thrust::sequence(thrust::device,d1.begin(), d1.end());
  thrust::transform(d1.begin(), d1.end(), d1.begin(), rfunc);
  thrust::transform(d1.begin(), d1.end(), d2.begin(), sfunc);

//thrust::for_each(thrust::device, d2.begin(), d2.end(), printf_functor());
  h2=d2;
  h1=d1;

  for(int i=0;i<(1<<8);i++){
    printf("%g\t%g\n",h1[i], h2[i]);
  }


  return 0;
}
```


Gnuplot window 0
'data_sin'
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5
6
3.44742, 1.05983
File Edit View Projects Debug Bookmarks
New Open Previous Document Next Do
Documents
Projects
Filesystem Browser
tests
1-1prod.cu
1-2prod.cu
1.cu
1prod.cu
2.cu
funct1.cu
funct2.cu
funct3.cu
//}
int main(){
range_functor rfunc
sin_functor sfunc;
thrust::host_vector
thrust::host_vector
thrust::device_vect
thrust::device_vect
thrust::sequence(th
thrust::transform(d
thrust::transform(d
//thrust::fill(d2.b
h2=d2;
h1=d1;
//thrust::for_each(t
for(int i=0;i<(1<
printf("%g\t%g\n"
}
return 0;
}
Line: 63 of 79 Col: 57 LINE INS
funct3.cu UTF-8
malkov@linux-5002:~/WORKSHOP/CUDA EXERCISE/THRUST/tests> nvcc funct3.cu -o funct3
malkov@linux-5002:~/WORKSHOP/CUDA EXERCISE/THRUST/tests> ./funct3 > data_sin
malkov@linux-5002:~/WORKSHOP/CUDA EXERCISE/THRUST/tests> gnuplot
G N U P L O T
Version 4.6 patchlevel 5 last modified February 2014
Build System: Linux x86_64
Copyright (C) 1986-1993, 1998, 2004, 2007-2014
Thomas Williams, Colin Kelley and many others
gnuplot home: http://www.gnuplot.info
faq, bugs, etc: type "help FAQ"
immediate help: type "help" (plot window: hit 'h')
Terminal type set to 'qt'
gnuplot> plot 'data_sin' w l
gnuplot>
Search and Replace
Current Project
Terminal
Debug View

```
#include <thrust/host_vector.h>
#include <thrust/device_vector.h>
#include <thrust/fill.h>
#include <thrust/transform.h>
#include <thrust/sequence.h>
#include <cstdio>

struct saxpy_functor
{
  const float a;
  saxpy_functor(float _a) : a(_a) {}
  __host__ __device__
  float operator()(float x, float y){
    return a*x+y;
  }
};

void saxpy(float a,
           thrust::device_vector<float>& x,
           thrust::device_vector<float>& y){
  saxpy_functor func(a);
  // call transform
  thrust::transform(x.begin(), x.end(), y.begin(), y.begin(), func);
}
```

```
int main(){
  thrust::host_vector<float> h1(1 << 24);
  thrust::host_vector<float> h2(1 << 24);
  thrust::sequence(h1.begin(), h1.end());
  thrust::fill(h2.begin(), h2.end(), 2.0);

  thrust::device_vector<float> d1 = h1;
  thrust::device_vector<float> d2 = h2;

  saxpy(3.0, d1, d2);

  h2=d2;

  for(int i=0;i<(1<<8);i++){
    printf("%g\n",h2[i]);
  }
  return 0;
}
```

КОРТЕЖИ

```
#include <thrust/tuple.h>
#include <cstdio>

int main(){
  thrust::tuple<int, float, const char *> test_tuple(23, 4.5, "thrust");
  printf("%d\t%g\t%s\n", thrust::get<0>(test_tuple),
                         thrust::get<1>(test_tuple),
                         thrust::get<2>(test_tuple));
  return 0;
}
```

```
malkov@linux-5002:~/WORKSHOP/CUDA EXERCISE/THRUST/tests> ./tuple1
23      4.5     thrust
```

```
#include <thrust/tuple.h>
#include <thrust/device_vector.h>
#include <thrust/host_vector.h>
#include <thrust/transform.h>
#include <thrust/fill.h>
#include <thrust/iterator/zip_iterator.h>
#include <cstdio>
#define N 32

struct rotate_tuple{
  __host__ __device__
  thrust::tuple<float,float,float> operator()(thrust::tuple<float&,float&,float&>& t){
    float x = thrust::get<0>(t);
    float y = thrust::get<1>(t);
    float z = thrust::get<2>(t);

    float rx=0.36*x+0.48*y-0.80*z;
    float ry=-0.80f*x+0.60*y+0.00f*z;
    float rz=0.48f*x+0.64f*y+0.60f*z;

    return thrust::make_tuple(rx,ry,rz);
  }
};
```

```
int main(){
  thrust::device_vector<float> x(N), y(N), z(N);

  thrust::fill(x.begin(), x.end(), 2.0);
  thrust::fill(y.begin(), y.end(), 3.0);
  thrust::fill(z.begin(), z.end(), 5.0);




  thrust::transform( thrust::make_zip_iterator( thrust::make_tuple(x.begin(), y.begin(),
z.begin() )),
            thrust::make_zip_iterator( thrust::make_tuple(x.end(),   y.end(),   z.end()   )),
            thrust::make_zip_iterator( thrust::make_tuple(x.begin(), y.begin(), z.begin() )),
                                                  rotate_tuple() );

  thrust::host_vector<float> hx(N), hy(N),hz(N);
  hx=x;  hy=y; hz=z;
      for(int i=0;i<N;i++)
         printf("%g\t%g\t%g\n",hx[i], hy[i], hz[i]);

  return 0;
}
```