OpenCL

OpenCL: получение списка драйверов, поддерживающих OpenCL

```
#include <CL/cl.h>
int main()
{
cl_platform_id platform[10];
cl_uint num_platforms;
clGetPlatformIDs(10, platform, &num_platforms);
```

```
cl_int clGetPlatformIDs (cl_uint numEntries, cl_platform_id *platforms, cl_uint  *numPlatforms);
```

numEntries	размер массива, на который указывает platforms (0, если *platforms* == NULL);
platforms	массив для возврата информации о драйверах (NULL - не возвращать);
numPlatforms	возвращаемое количество драйверов OpenCL (NULL - не возвращать);

OpenCL: обнаружение устройств

```
cl_device_id device;
cl_uint num_gpgpu;
clGetDeviceIDs(platform[0], CL_DEVICE_TYPE_GPU, 1, &device, &num_gpgpu);
if (num_gpgpu == 0){
   clGetDeviceIDs( platform[0], CL_DEVICE_TYPE_CPU, 1, &device, NULL);
}
```

```
cl_int clGetDeviceIDs (cl_platform_id platformID, cl_device_type  nDeviceType,
        cl_uint nNumEntries, cl_device_id *pDevices, cl_uint *pnNumDevices);
```

nDeviceType :

CL_DEVICE_TYPE_CPU	центральный процессор
CL_DEVICE_TYPE_GPU	видеокарта
CL_DEVICE_TYPE_ACCELERATOR	специализированный ускоритель
CL_DEVICE_TYPE_DEFAULT	устройство по молчанию в системе
CL_DEVICE_TYPE_ALL	все доступные устройства OpenCL

OpenCL: создание контекста и очереди на устройстве

```
cl_context context = clCreateContext(NULL, 1, &device, NULL, NULL, NULL);
cl_command_queue queue = clCreateCommandQueue(context, device, 0, NULL);
```

```
cl_context clCreateContext(const cl_context_properties * properties,
                cl_uint num_devices,    const cl_device_id *devices,
                void (CL_CALLBACK *pfnNotify),   void *userData,   cl_int * errcode_ret);

CL_CALLBACK  *pfnNotify(const char *errinfo,   const void *private_info,   size_t cb, void
*user_data) – функция обратного вызова
```

```
cl_command_queue clCreateCommandQueue (cl_context context,
cl_device_id device, cl_command_queue_properties properties, cl_int *errcode_ret)

properties - список свойств очереди команд: можно ли выполнять команды не
последовательно и разрешено ли профилирование команд
```

OpenCL: компиляция программы

```
char *source = /* читаем из файла код kernel-функции */;
cl_program program = clCreateProgramWithSource(context, 1,
                (const char **) &source, NULL, NULL);
```

```
cl_program clCreateProgramWithSource (cl_context context,   cl_uint count,
                const char **strings,   const size_t *lengths,   cl_int *errcode_ret)
```

```
cl_int clBuildProgram (cl_program program, cl_uint num_devices,
                              const cl_device_id *device_list,   const char *options,
                              void CL_CALLBACK *pfn_notify),   void *user_data)
```

CL_CALLBACK *pfn_notify(cl_program program, void *user_data) – функция обратного вызова

OpenCL: определение точки входа

```
cl_kernel kernel = clCreateKernel(program, "kernel_function_name", NULL);
```

```
cl_kernel clCreateKernel (cl_program program, const char *kernel_name,
                          cl_int *errcode_ret)
```

```
__kernel void sum(__global int *x, int n, __global int *res)
{
    uint i,j, id=get_global_id(0), sz = get_global_size(0);
    int s = 0;
    for (int i = id; i<n; i += sz)
        s += x[i];
    atomic_add(&res[0], s);
}
```

OpenCL: выделение памяти на ускорителе

```
cl_mem x_gpu = clCreateBuffer( context, CL_MEM_READ_WRITE |
                CL_MEM_COPY_HOST_PTR, n * sizeof(x[0]), x, NULL );
```

```
cl_mem clCreateBuffer (cl_context context, cl_mem_flags flags, size_t size,
            void *host_ptr,   cl_int *errcode_ret)
```

cl_mem_flags:

CL_MEM_READ_WRITE	чтение / запись
CL_MEM_WRITE_ONLY	только запись
CL_MEM_READ_ONLY	только запись
CL_MEM_USE_HOST_PTR	использование памяти в CPU напрямую
CL_MEM_ALLOC_HOST_PTR	выделение памяти в CPU
CL_MEM_COPY_HOST_PTR	копирование данных из CPU в GPU

OpenCL: инициализация аргументов kernel-функции

```
clSetKernelArg (kernel, 0, sizeof(x_gpu), (void*) &x_gpu);
clSetKernelArg (kernel, 1, sizeof(n), (void*) &n);
```

```
cl_int clSetKernelArg (cl_kernel kernel, cl_uint arg_index, size_t arg_size,
        const void *arg_value)
```

arg_index – порядковый номер параметра

OpenCL: добавление задачи в очередь

```
size_t gridSize = 10;
clEnqueueNDRangeKernel(queue, kernel, 1, NULL, &gridSize, NULL, 0, NULL, NULL);
clFinish(queue);

//clEnqueueNDRangeKernel(queue, kernel, 1, NULL, &gridSize, NULL, 0, NULL,&event);
//clWaitForEvents(1, &event);
```

```
cl_int clEnqueueNDRangeKernel (cl_command_queue command_queue,
        cl_kernel kernel, cl_uint work_dim, const size_t *global_work_offset, const
        size_t *global_work_size, const size_t *local_work_size, cl_uint
         num_events_in_wait_list, const cl_event *event_wait_list, cl_event *event)
```

work_dim – размерность пространства Work-item (рабочих)

OpenCL: копирование данных из памяти ускорителя

clEnqueueReadBuffer*(queue, res_gpu, CL_TRUE, 0, res_size, res_cpu, 0, NULL, NULL);*

```
cl_int clEnqueueReadBuffer (cl_command_queue command_queue,
        cl_mem buffer, cl_bool blocking_read, size_t offset, size_t cb, void *ptr,
        cl_uint num_events_in_wait_list, const cl_event *event_wait_list, cl_event *event)
```

Проверка ошибок

```
#define checkError(func) \
if (errcode != CL_SUCCESS){\
printf("Error in " #func "\nError code = %d\n", errcode);\
exit(1);\
}
#define checkErrorEx(command) \
command; \
checkError(command);
```

OpenCL: замеры времени

clEnqueueNDRangeKernel(*queue*, *kernel*, 1, NULL, *&gridSize*, NULL, 0, NULL, &event);
clWaitForEvents(1, &event);
checkErrorEx(*errcode* = clGetEventProfilingInfo(*event,*
CL_PROFILING_COMMAND_START, sizeof(time_start), &time_start, NULL));
checkErrorEx(*errcode* = clGetEventProfilingInfo(*event,*
CL_PROFILING_COMMAND_END, sizeof(time_end), &time_end, NULL));
tgpu = (double)(time_end - time_start)/1e9;

cl_int clGetEventProfilingInfo (cl_event *event*, cl_profiling_info *param_name*, size_t *param_value_size*, void **param_value*, size_t param_value_size_ret)

Сборка программы

Если CUDA SDK устанавливалась с сайта NVIDIA:

```
gcc -std=c99 -L/usr/local/cuda/lib64 -I/usr/local/cuda/include
main.c -lOpenCL -O2 -o proga
```

Если из репозитория:

```
gcc -std=c99 main.c -lOpenCL -O2 -o proga
g++ -std=c++11 main.cpp -lOpenCL -O2 -o proga
```