2、Get distance information_Linux

1、Start

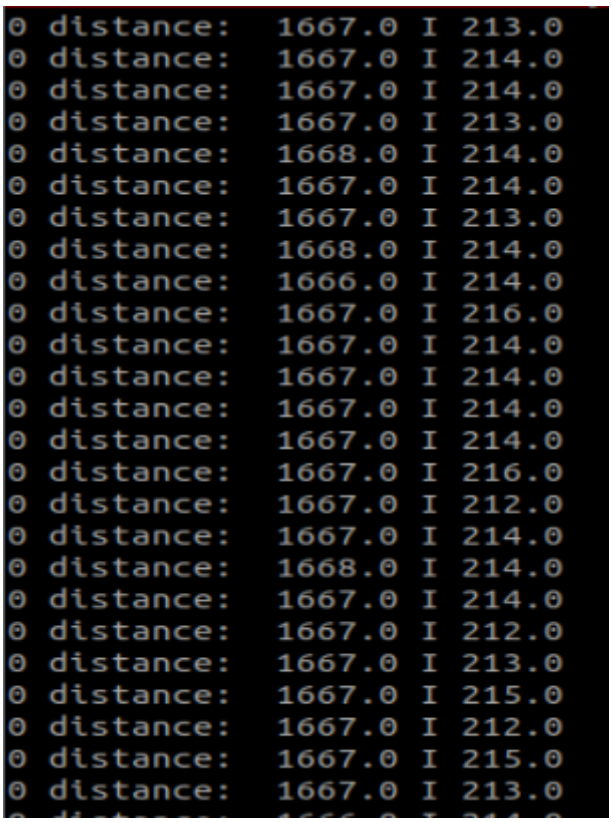
When compiling the SDK, we will generate an SDM_ The program of test is located in the YDLidar-SDK/build directory.

The executable file will be generated if it is compiled normally.

Enter following commands in terminal in this directory.

```
./sdm_test
```

Enter "1", press "Enter", and the following will appear.



Distance is the distance measured by the module, unit is mm.

2、Code

Code path（SDK installation path：~/software）：~/software/YDLidar-SDK/samples/sdm_test.cpp

Note: Search according to your SDK installation location, which is in the samples under the SDK directory

```
#include "CYdLidar.h"
#include <iostream>
#include <string>
#include <algorithm>
#include <cctype>
using namespace std;
using namespace ydlidar;

#if defined(_MSC_VER)
#pragma comment(lib, "ydlidar_sdk.lib")
#endif
int main(int argc, char *argv[])
{
  printf("__   ______  _     ___ ____     _    ____  \n");
  printf("\\ \\ / /  _ \\| |   |_ _|  _ \\  / \\  |  _ \\ \n");
  printf(" \\ V /| | | | |    | || | | |/ _ \\ | |_) | \n");
  printf("  | | | |_| | |___ | || |_| / ___ \\|  _ <  \n");
  printf("  |_| |____/|_____|___|____/_/   \\_\\_| \\_\\ \n");
  printf("\n");
  fflush(stdout);

  //Initialization
  ydlidar::os_init();
  //Initialize serial port
  std::string port;
  std::map<std::string, std::string> ports = ydlidar::lidarPortList();
  std::map<std::string, std::string>::iterator it;
  if (ports.size() == 1)
  {
    port = ports.begin()->second;
  }
  else
  {
    int id = 0;
    for (it = ports.begin(); it != ports.end(); it++)
    {
      printf("[%d] %s %s\n",
        id, it->first.c_str(), it->second.c_str());
      id++;
    }

    if (ports.empty())
    {
      printf("Not Lidar was detected. Please enter the lidar serial port:");
      std::cin >> port;
    }
    else
    {
```

```
    while (ydlidar::os_isOk())
    {
      printf("Please select the lidar port:");
      std::string number;
      std::cin >> number;

      if ((size_t)atoi(number.c_str()) >= ports.size())
      {
        continue;
      }

      it = ports.begin();
      id = atoi(number.c_str());

      while (id)
      {
        id--;
        it++;
      }

      port = it->second;
      break;
    }
  }
}

int baudrate = 460800; //Default serial port number

bool isSingleChannel = false;

CYdLidar laser;
//////////////////////string property/////////////////
/// lidar port
laser.setlidaropt(LidarPropSerialPort, port.c_str(), port.size());
/// ignore array
std::string ignore_array;
ignore_array.clear();
laser.setlidaropt(LidarPropIgnoreArray, ignore_array.c_str(),
                  ignore_array.size());
//////////////////////int property/////////////////
/// lidar baudrate
laser.setlidaropt(LidarPropSerialBaudrate, &baudrate, sizeof(int));
/// sdm lidar
int optval = TYPE_SDM;
laser.setlidaropt(LidarPropLidarType, &optval, sizeof(int));
/// device type
optval = YDLIDAR_TYPE_SERIAL;
laser.setlidaropt(LidarPropDeviceType, &optval, sizeof(int));
/// sample rate
optval = 4;
laser.setlidaropt(LidarPropSampleRate, &optval, sizeof(int));
/// abnormal count
optval = 3;
laser.setlidaropt(LidarPropAbnormalCheckCount, &optval, sizeof(int));
/// Intenstiy bit count
```

```
optval = 4;
laser.setlidaropt(LidarPropIntenstiyBit, &optval, sizeof(int));

//////////////////////bool property/////////////////
/// fixed angle resolution
bool b_optvalue = false;
laser.setlidaropt(LidarPropFixedResolution, &b_optvalue, sizeof(bool));
/// rotate 180
laser.setlidaropt(LidarPropReversion, &b_optvalue, sizeof(bool));
/// Counterclockwise
laser.setlidaropt(LidarPropInverted, &b_optvalue, sizeof(bool));
b_optvalue = true;
laser.setlidaropt(LidarPropAutoReconnect, &b_optvalue, sizeof(bool));
/// one-way communication
laser.setlidaropt(LidarPropSingleChannel, &isSingleChannel, sizeof(bool));
/// intensity
b_optvalue = true;
laser.setlidaropt(LidarPropIntenstiy, &b_optvalue, sizeof(bool));
/// Motor DTR
b_optvalue = true;
laser.setlidaropt(LidarPropSupportMotorDtrCtrl, &b_optvalue, sizeof(bool));
/// HeartBeat
b_optvalue = false;
laser.setlidaropt(LidarPropSupportHeartBeat, &b_optvalue, sizeof(bool));

//////////////////////float property/////////////////
/// unit: °
float f_optvalue = 180.0f;
laser.setlidaropt(LidarPropMaxAngle, &f_optvalue, sizeof(float));
f_optvalue = -180.0f;
laser.setlidaropt(LidarPropMinAngle, &f_optvalue, sizeof(float));
/// unit: m
f_optvalue = 20.f;
laser.setlidaropt(LidarPropMaxRange, &f_optvalue, sizeof(float));
f_optvalue = 0.025f;
laser.setlidaropt(LidarPropMinRange, &f_optvalue, sizeof(float));
/// unit: Hz
float frequency = 100.0;
laser.setlidaropt(LidarPropScanFrequency, &frequency, sizeof(float));

//Lidar initialization
bool ret = laser.initialize();
if (!ret)
{
  fprintf(stderr, "[YDLIDAR] Fail to initialize %s\n", laser.DescribeError());
  return -1;
}
//Start scan
ret = laser.turnOn();
if (!ret)
{
  fprintf(stderr, "[YDLIDAR] Fail to turn on %s\n", laser.DescribeError());
  return -1;
}
```

```
  LaserScan scan;

  while (ret && ydlidar::os_isOk())
  {
    if (laser.doProcessSimple(scan))
    {
      for (size_t i = 0; i < scan.points.size(); ++i)
      {
        const LaserPoint &p = scan.points.at(i);
        printf("%lu distance:  %.01f I %.01f\n", i,
          p.range * 1000.0f, p.intensity);
      }
      fflush(stdout);
    }
    else
    {
      fprintf(stderr, "[YDLIDAR] Failed to get lidar data\n");
      fflush(stderr);
    }
  }

  laser.turnOff();
  laser.disconnecting();

  return 0;
}
```

while function.

```
for (size_t i = 0; i < scan.points.size(); ++i)
      {
        const LaserPoint &p = scan.points.at(i);
        printf("%lu distance:  %.01f I %.01f\n", i,
          p.range * 1000.0f, p.intensity);
      }
```

Get the distance and print it. The actual code logic is to select the serial port, initialize the serial port, initialize the radar, start scanning, and print the result.