一个C++写的windows service程序模板

最近在写一个windows下文件传输服务,把涉及到windows service的部分摘出来分享一下  
如果把头部的 #define RUN\_AS\_SERVICE 1 注释掉就可以当做普通程序一样调试  
参考了下面两个地址  
https://www.codeproject.com/Articles/499465/Simple-Windows-Service-in-Cplusplus  
https://github.com/tromgy/service-base

|  |
| --- |
| #include "def.h"  #include "util.h"  #include "lev\_con.h"  #include <tchar.h>  #include <chrono>  #include <thread>  #include "server.h"    //define marco RUN\_AS\_SERVICE to run as service  #define RUN\_AS\_SERVICE  1    SERVICE\_STATUS\_HANDLE g\_StatusHandle = NULL;  SERVICE\_STATUS svcstatus;    VOID WINAPI ServiceMain (DWORD argc, LPTSTR \*argv);  VOID WINAPI ServiceCtrlHandler (DWORD);    void UpdateServiceStatus( int status, int exitcode, bool accept\_ctrl );    void usage();    bool InstallService();  bool UninstallService();  void ChangeWorkingDirToExePath();    #define SERVICE\_NAME     \_T("WinFileTrans")  #define SERVIER\_DISPLAY  \_T("Windows File Transfer Service")  #define SERVICE\_DESC     \_T("Windows File Transfer Service")    int main (int argc, char\* argv[])  {      if( argc == 2 && !strcmp( argv[1], "--help" ) )      {          usage();          return 0;      }        if( argc == 2 && !strcmp( argv[1], "InstallService" ) )      {          if( InstallService() )          {              printf( "install service ok!\n");              return 0;          }          else          {              printf( "install service failed!\n");              return 1;          }      }        if( argc == 2 && !strcmp( argv[1], "UninstallService" ) )      {          if( UninstallService() )          {              printf( "uninstall service ok!\n");              return 0;          }          else          {              printf( "uninstall service failed!\n");              return 1;          }      }        strcpy( log\_file, LOG\_FILE );          ChangeWorkingDirToExePath();      LevInitEnvironment();    #ifdef RUN\_AS\_SERVICE      SERVICE\_TABLE\_ENTRY ServiceTable[] =      {          {SERVICE\_NAME, (LPSERVICE\_MAIN\_FUNCTION) ServiceMain},          {NULL, NULL}      };        if (StartServiceCtrlDispatcher (ServiceTable) == FALSE)      {         write\_log( "[ERR] StartServiceCtrlDispatcher returned error!");         return GetLastError ();      }  #else        write\_log( "winfiletrans init.. " );      int port;      sock\_t lsnfd;      if( cfg\_get\_int( CFG\_FILE, "config", "port", &port ) )          return 1;        lsnfd = tcp\_listen( NULL, port );      if( lsnfd == ESOCK )      {          printf( "tcp\_listen failed on port %d!", port );          write\_log( "[ERR] tcp\_listen failed on port %d !", port );          return 1;      }        write\_log( "WinFileTrans startup ok!");      Server srv( lsnfd );      srv.Run();        #endif          return 0;  }      VOID WINAPI ServiceMain (DWORD argc, LPTSTR \*argv)  {      DWORD Status = E\_FAIL;      sock\_t lsnfd;      int port;        write\_log( "enter ServiceMain..");        g\_StatusHandle = RegisterServiceCtrlHandler (SERVICE\_NAME, ServiceCtrlHandler);        if (g\_StatusHandle == NULL)      {          write\_log("[ERR] RegisterServiceCtrlHandler failed!");          return;      }        UpdateServiceStatus( SERVICE\_START\_PENDING, 0, false );        if( cfg\_get\_int( CFG\_FILE, "config", "port", &port ) )      {          UpdateServiceStatus( SERVICE\_STOPPED, 1, false );          return;      }        lsnfd = tcp\_listen( NULL, port );      if( lsnfd == ESOCK )      {          write\_log( "[ERR] tcp listen on port %d failed!", port );          UpdateServiceStatus( SERVICE\_STOPPED, 1, false );          return;      }        UpdateServiceStatus( SERVICE\_RUNNING, 0, true );        Server srv( lsnfd );      srv.Run();          UpdateServiceStatus( SERVICE\_STOPPED, 0, false );        write\_log( "ServiceMain exit" );        return;  }      VOID WINAPI ServiceCtrlHandler (DWORD CtrlCode)  {      write\_log( "recv service control: %d", CtrlCode );        switch (CtrlCode)      {          case SERVICE\_CONTROL\_SHUTDOWN:          case SERVICE\_CONTROL\_STOP:                write\_log("recv exit service control cmd.");              UpdateServiceStatus( SERVICE\_STOP\_PENDING, 0, false );              LevStopAllEventLoop();                break;         default:           break;      }  }        void usage()  {      printf(      "usage:winfiletrans InstallService   - install   windows service\n"      "                   UninstallService - uninstall windows service\n"      "config file:\n"      "winfiletrans.cfg  - config file\n"      "ip\_whitelist.txt  - ip white list file(optional)\n"      "user\_list.txt     - user list file\n"      );      }        bool InstallService()  {      TCHAR path[MAX\_PATH];      SC\_HANDLE mg\_handle;      SC\_HANDLE sc\_handle;      SERVICE\_DESCRIPTION sd;        GetModuleFileName(NULL, path, MAX\_PATH);        mg\_handle = OpenSCManager( NULL, NULL, SC\_MANAGER\_CREATE\_SERVICE );      if( !mg\_handle )      {          printf( "install service failed! OpenSCManager failed!\n");          return false;      }        sc\_handle = CreateService( mg\_handle,      SERVICE\_NAME,      SERVIER\_DISPLAY,      SERVICE\_ALL\_ACCESS,      SERVICE\_WIN32\_OWN\_PROCESS,      SERVICE\_AUTO\_START,      SERVICE\_ERROR\_NORMAL,      path,      NULL,      NULL,      NULL,      NULL,      NULL      );        if( !sc\_handle )      {          printf("install service failed! CreateService failed!\n");          CloseServiceHandle( mg\_handle );          return false;      }          sd.lpDescription = SERVICE\_DESC;      ChangeServiceConfig2(sc\_handle, SERVICE\_CONFIG\_DESCRIPTION, &sd);        CloseServiceHandle( mg\_handle );      CloseServiceHandle( sc\_handle );          return true;  }    bool UninstallService()  {      SC\_HANDLE mg\_handle;      SC\_HANDLE sc\_handle;      SERVICE\_STATUS ssSvcStatus;      bool ret;        mg\_handle = OpenSCManager( NULL, NULL, SC\_MANAGER\_CONNECT );      if( !mg\_handle )      {          printf( "uninstall service failed! OpenSCManager failed!\n");          return false;      }        sc\_handle = OpenService( mg\_handle, SERVICE\_NAME, SERVICE\_STOP | SERVICE\_QUERY\_STATUS | DELETE );      if( !sc\_handle)      {          printf( "uninstall service failed! OpenService failed!\n");          CloseServiceHandle( mg\_handle );          return false;      }        ControlService(sc\_handle, SERVICE\_CONTROL\_STOP, &ssSvcStatus);        std::this\_thread::sleep\_for(std::chrono::seconds(3));        if( DeleteService( sc\_handle ) )      {          ret = true;      }      else      {          ret = false;      }        CloseServiceHandle( mg\_handle );      CloseServiceHandle( sc\_handle );        return true;  }      void ChangeWorkingDirToExePath()  {      TCHAR path[MAX\_PATH];      int len;      int i;        GetModuleFileName(NULL, path, MAX\_PATH);        len = (int)strlen(path);        for (i = len - 1; i >= 0; i--)      {          if (path[i] == '\\')          {              path[i] = 0;              break;          }      }        SetCurrentDirectory(path);  }      void UpdateServiceStatus( int status, int exitcode, bool accept\_ctrl )  {          memset( &svcstatus, 0, sizeof(svcstatus) );      svcstatus.dwServiceType = SERVICE\_WIN32\_OWN\_PROCESS;      svcstatus.dwCurrentState = status;      svcstatus.dwControlsAccepted = accept\_ctrl ? ( SERVICE\_ACCEPT\_SHUTDOWN | SERVICE\_ACCEPT\_STOP ) : 0;      svcstatus.dwWin32ExitCode = exitcode;      svcstatus.dwServiceSpecificExitCode = 0;        SetServiceStatus (g\_StatusHandle, &svcstatus);      } |