说明：

1.以下文章中的 osip 版本为 3.0.1

2.eXosip 版本为 3.0.1

3.编译环境为：Windows XP 专业版本+VS 2005

4.示例程序是在 http://blog.csdn.net/bat603/中下载，修改而成（原来是在 linux 下的示例）

5.如要转载此文章，请说明出处

6.本人也是才接触 SIP（呵呵，不到一周），其中一定有很多不对之处，请指正。

第一步，下载：

到 http://www.gnu.org/software/osip/下载最新的 osip 库并解压

到 http://savannah.gnu.org/projects/exosip/下载最新的 eXsoip 库并解压

第二步，编译 osip3.0.1：

1.用 VS2005 打开 libosip2-3.0.1/platform/vsnet/osip.sln

2.对 osip2 和 osipparser2 项目生成 Release DLL

3.其生成的 LIB/DLL 位于：libosip2-3.0.1/platform/vsnet/Release DLL/下

第三步，编译 eXosip3.0.1：

1.用 VS2005 打开 libeXosip2-3.0.1/platform/vsnet/eXosip.sln

2.将 libosip2-3.0.1/include/osip2 目录 COPY 到 libeXosip2-3.0.1/include/下

3.将 libosip2-3.0.1/include/osipparser2 目录 COPY 到 libeXosip2-3.0.1/include/下

4.在 libeXosip2-3.0.1 目录下新建 lib 目录

5.将“第二步”中生成的 libosip2-3.0.1/platform/vsnet/Release DLL/osip2.lib 文件 COPY 到 libeXosip2-3.0.1/lib 目录下

6.将“第二步”中生成的 libosip2-3.0.1/platform/vsnet/Release DLL/osipparser2.lib 文件 COPY 到 libeXosip2-3.0.1/lib 目录下

7.修改项目属性，新增 Dnsapi.lib Iphlpapi.lib Ws2_32.lib osip2.lib osipparser2.lib 库输入

8.修改项目属性，新增库目录../../lib

9.修改项目的输出为.DLL，默认为.lib

10.编译 Relase DLL

11.其生成的 LIB/DLL 位于：libeXosip2-3.0.1/platform/vsnet/Release/下

第四步，编译示例程序：

将以下程序作为 UAS.CPP 保存

[Copy to clipboard]

CODE:

```
// UAS.cpp : 定义控制台应用程序的入口点。
//
#i nclude <eXosip2/eXosip.h>
```

```
＃i nclude <stdio.h>
＃i nclude <stdlib.h>
＃i nclude <Winsock2.h>
/*
＃i nclude <netinet/in.h>
＃i nclude <sys/socket.h>
＃i nclude <sys/types.h>*/
#pragma comment(lib, "osip2.lib")
#pragma comment(lib, "osipparser2.lib")
#pragma comment(lib, "eXosip.lib")
#pragma comment(lib, "Iphlpapi.lib")
#pragma comment(lib, "Dnsapi.lib")
#pragma comment(lib, "ws2_32.lib")
int
main (int argc, char *argv[])
{
  eXosip_event_t *je = NULL;
  osip_message_t *ack = NULL;
  osip_message_t *invite = NULL;
  osip_message_t *answer = NULL;
  sdp_message_t *remote_sdp = NULL;
  int call_id, dialog_id;
  int i,j;
  int id;
  char *sour_call = "sip:24@10.16.79.24";
  char *dest_call = "sip:24@10.16.79.24:5061";//client ip
  char command;
  char tmp[4096];
  char localip[128];
  int pos = 0;
  //初始化 sip
  i = eXosip_init ();
  if (i != 0)
     {
       printf ("Can't initialize eXosip!/n");
```

```
      return -1;
    }
else
    {
      printf ("eXosip_init successfully!/n");
    }
i = eXosip_listen_addr (IPPROTO_UDP, NULL, 5060, AF_INET, 0);
if (i != 0)
    {
      eXosip_quit ();
      fprintf (stderr, "eXosip_listen_addr error!/nCouldn't initialize transport layer!/n");
    }
for(;;)
    {
      //侦听是否有消息到来
      je = eXosip_event_wait (0,50);
      //协议栈带有此语句,具体作用未知
      eXosip_lock ();
      eXosip_default_action (je);
      eXosip_automatic_refresh ();
      eXosip_unlock ();
      if (je == NULL)//没有接收到消息
    continue;
      // printf ("the cid is %s, did is %s/n", je->did, je->cid);
      switch (je->type)
    {
    case EXOSIP_MESSAGE_NEW://新的消息到来
      printf (" EXOSIP_MESSAGE_NEW!/n");
      if (MSG_IS_MESSAGE (je->request))//如果接受到的消息类型是 MESSAGE
        {
      {
        osip_body_t *body;
        osip_message_get_body (je->request, 0, &body);
        printf ("I get the msg is: %s/n", body->body);
        //printf ("the cid is %s, did is %s/n", je->did, je->cid);
        }
        //按照规则，需要回复 OK 信息
        eXosip_message_build_answer (je->tid, 200,&answer);
        eXosip_message_send_answer (je->tid, 200,answer);
          }
        break;
      case EXOSIP_CALL_INVITE:
        //得到接收到消息的具体信息
                printf ("Received a INVITE msg from %s:%s, UserName is %s, password
is %s/n",je->request->req_uri->host,
              je->request->req_uri->port, je->request->req_uri->username, je->request->req_uri->password);
        //得到消息体,认为该消息就是 SDP 格式.
        remote_sdp = eXosip_get_remote_sdp (je->did);
        call_id = je->cid;
        dialog_id = je->did;

        eXosip_lock ();
        eXosip_call_send_answer (je->tid, 180, NULL);
        i = eXosip_call_build_answer (je->tid, 200, &answer);
        if (i != 0)
          {
            printf ("This request msg is invalid!Cann't response!/n");
            eXosip_call_send_answer (je->tid, 400, NULL);
          }
        else
          {
            snprintf (tmp, 4096,
                  "v=0/r/n"
                  "o=anonymous 0 0 IN IP4 0.0.0.0/r/n"
                  "t=1 10/r/n"
                  "a=username:rainfish/r/n"
                  "a=password:123/r/n");

            //设置回复的 SDP 消息体,下一步计划分析消息体
            //没有分析消息体，直接回复原来的消息，这一块做的不好。
```

```
              osip_message_set_body (answer, tmp, strlen(tmp));
              osip_message_set_content_type (answer, "application/sdp");

              eXosip_call_send_answer (je->tid, 200, answer);
              printf ("send 200 over!/n");
          }
        eXosip_unlock ();

        //显示出在 sdp 消息体中的 attribute  的内容,里面计划存放我们的信息
        printf ("the INFO is :/n");
        while (!osip_list_eol ( &(remote_sdp->a_attributes), pos))
          {
              sdp_attribute_t *at;

              at = (sdp_attribute_t *) osip_list_get ( &remote_sdp->a_attributes, pos);
              printf ("%s : %s/n", at->a_att_field, at->a_att_value);//这里解释了为什么在 SDP 消息体中属性 a
里面存放必须是两列

              pos ++;
          }
        break;
      case EXOSIP_CALL_ACK:
        printf ("ACK recieved!/n");
        // printf ("the cid is %s, did is %s/n", je->did, je->cid);
        break;
      case EXOSIP_CALL_CLOSED:
        printf ("the remote hold the session!/n");
        // eXosip_call_build_ack(dialog_id, &ack);
        //eXosip_call_send_ack(dialog_id, ack);
        i = eXosip_call_build_answer (je->tid, 200, &answer);
        if (i != 0)
          {
              printf ("This request msg is invalid!Cann't response!/n");
              eXosip_call_send_answer (je->tid, 400, NULL);
          }
       else
         {
          eXosip_call_send_answer (je->tid, 200, answer);
          printf ("bye send 200 over!/n");
         }
       break;
     case EXOSIP_CALL_MESSAGE_NEW://至于该类型和 EXOSIP_MESSAGE_NEW 的区别，源代码这
么解释的
     /*
     // request related events within calls (except INVITE)
          EXOSIP_CALL_MESSAGE_NEW,              < announce new incoming request.
     // response received for request outside calls
           EXOSIP_MESSAGE_NEW,              < announce new incoming request.
           我也不是很明白，理解是：EXOSIP_CALL_MESSAGE_NEW 是一个呼叫中的新的消息到来，
比如 ring trying 都算，所以在接受到后必须判断
           该消息类型，EXOSIP_MESSAGE_NEW 而是表示不是呼叫内的消息到来。
           该解释有不妥地方，仅供参考。
     */
       printf(" EXOSIP_CALL_MESSAGE_NEW/n");
       if (MSG_IS_INFO(je->request) ) //如果传输的是 INFO 方法
       {
                 eXosip_lock ();
            i = eXosip_call_build_answer (je->tid, 200, &answer);
            if (i == 0)
              {
               eXosip_call_send_answer (je->tid, 200, answer);
              }
            eXosip_unlock ();
          {
        osip_body_t *body;
        osip_message_get_body (je->request, 0, &body);
        printf ("the body is %s/n", body->body);
          }
       }
```

```
    break;
  default:
    printf ("Could not parse the msg!/n");
  }
  }
}
```

将以下程序作为 UAC.CPP 保存

[Copy to clipboard]

CODE:

```
// UAC.cpp : 定义控制台应用程序的入口点。
//
＃i nclude <eXosip2/eXosip.h>
＃i nclude <stdio.h>
＃i nclude <stdlib.h>
＃i nclude <Winsock2.h>
/*
＃i nclude <netinet/in.h>
＃i nclude <sys/socket.h>
＃i nclude <sys/types.h>*/
#pragma comment(lib, "osip2.lib")
#pragma comment(lib, "osipparser2.lib")
#pragma comment(lib, "eXosip.lib")
#pragma comment(lib, "Iphlpapi.lib")
#pragma comment(lib, "Dnsapi.lib")

#pragma comment(lib, "ws2_32.lib")
int
main (int argc, char *argv[])
{
  eXosip_event_t *je;
  osip_message_t *reg = NULL;
  osip_message_t *invite = NULL;
  osip_message_t *ack = NULL;
  osip_message_t *info = NULL;
osip_message_t *message = NULL;

int call_id, dialog_id;

int i,flag;

int flag1 = 1;

int id;


char *identity = "sip:24@10.16.79.24";

char *registerer = "sip:10.16.79.24:5060";//server ip

char *source_call = "sip:24@10.16.79.24";

char *dest_call = "sip:24@10.16.79.24:5060";//server ip


char command;

char tmp[4096];

char localip[128];

printf("r      向服务器注册/n/n");

printf("c      取消注册/n/n");

printf("i      发起呼叫请求/n/n");

printf("h      挂断/n/n");

printf("q      退出程序/n/n");

printf("s      执行方法 INFO/n/n");

printf("m      执行方法 MESSAGE/n/n");

//初始化

i = eXosip_init ();

if (i != 0)

   {

     printf ("Couldn't initialize eXosip!/n");

     return -1;

   }

else

   {

     printf ("eXosip_init successfully!/n");

   }

i = eXosip_listen_addr (IPPROTO_UDP, NULL, 5061, AF_INET, 0);

if (i != 0)

   {
        eXosip_quit ();
        fprintf (stderr, "Couldn't initialize transport layer!/n");
        return -1;
      }
  flag = 1;
  while (flag)
      {
        printf ("please input the comand:/n");

        scanf ("%c", &command);
        getchar ();

        switch (command)
      {
      case 'r':
        printf ("This modal isn't commpleted!/n");
        break;
      case 'i':/* INVITE */
        i = eXosip_call_build_initial_invite (&invite, dest_call, source_call, NULL, "This si a call for a
conversation");
        if (i != 0)
          {
            printf ("Intial INVITE failed!/n");
            break;
          }
          //符合 SDP 格式,其中属性 a 是自定义格式,也就是说可以存放自己的信息,但是只能是两列,比如帐户
信息
          //但是经测试,格式:v o t 必不可少,原因未知,估计是协议栈在传输时需要检查的
        snprintf (tmp, 4096,
              "v=0/r/n"
              "o=anonymous 0 0 IN IP4 0.0.0.0/r/n"
              "t=1 10/r/n"
              "a=username:rainfish/r/n"
              "a=password:123/r/n");
        osip_message_set_body (invite, tmp, strlen(tmp));
```

```
osip_message_set_content_type (invite, "application/sdp");


eXosip_lock ();
i = eXosip_call_send_initial_invite (invite);
eXosip_unlock ();
flag1 = 1;
while (flag1)
  {
      je = eXosip_event_wait (0, 200);


      if (je == NULL)
  {
      printf ("No response or the time is over!/n");
      break;
  }


      switch (je->type)
  {
  case EXOSIP_CALL_INVITE:
      printf ("a new invite reveived!/n");
      break;
  case EXOSIP_CALL_PROCEEDING:
      printf ("proceeding!/n");
      break;
  case EXOSIP_CALL_RINGING:
      printf ("ringing!/n");
      // call_id = je->cid;
      // dialog_id = je->did;
      printf ("call_id is %d, dialog_id is %d /n", je->cid, je->did);
      break;
  case EXOSIP_CALL_ANSWERED:
      printf ("ok! connected!/n");
      call_id = je->cid;
      dialog_id = je->did;
      printf ("call_id is %d, dialog_id is %d /n", je->cid, je->did);
```

```
        eXosip_call_build_ack (je->did, &ack);
        eXosip_call_send_ack (je->did, ack);
        flag1 = 0;
        break;
    case EXOSIP_CALL_CLOSED:
        printf ("the other sid closed!/n");
        break;
    case EXOSIP_CALL_ACK:
        printf ("ACK received!/n");
        break;
    default:
        printf ("other response!/n");
        break;
    }
        eXosip_event_free (je);

    }
  break;
case 'h':
  printf ("Holded !/n");

  eXosip_lock ();
  eXosip_call_terminate (call_id, dialog_id);
  eXosip_unlock ();
  break;
case 'c':
  printf ("This modal isn't commpleted!/n");
  break;
case 's':
//传输 INFO 方法
  eXosip_call_build_info (dialog_id, &info);
  snprintf (tmp , 4096,
          "hello,rainfish");
  osip_message_set_body (info, tmp, strlen(tmp));
  //格式可以任意设定,text/plain 代表文本信息
```

```
      osip_message_set_content_type (info, "text/plain");
      eXosip_call_send_request (dialog_id, info);
      break;
    case 'm':
    //传输 MESSAGE 方法,也就是即时消息，和 INFO 方法相比，我认为主要区别，是 MESSAGE 不用建立连接，直接传输信息，而 INFO 必须
    //在建立 INVITE 的基础上传输。
      printf ("the mothed :MESSAGE/n");
      eXosip_message_build_request (&message, "MESSAGE", dest_call, source_call, NULL);
      snprintf (tmp, 4096,
              "hellor rainfish");
      osip_message_set_body (message, tmp, strlen(tmp));
      //假设格式是 xml
      osip_message_set_content_type (message, "text/xml");
      eXosip_message_send_request (message);
      break;
    case 'q':
      eXosip_quit ();
      printf ("Exit the setup!/n");
      flag = 0;
      break;
    }
    }
  return (0);
}
```