1. 3GPP协议文档转C文件方法

1.1将3GPP协议文档另存为txt文档（在Word中另存为，不要直接重命名）

1.2编译运行3GPP2asn.cpp程序（代码见附录），将上一步的txt文档转换成asn文件：

编译：g++ 3GPP2asn.cpp -o 3GPP2asn

运行：./ 3GPP2asn xxxx.txt （xxxx.txt为文件要转换的文件）

编译运行成功则会生成对应的xxxx.asn文件

1.3使用ASN1C工具，将转换的asn文件生成C文件

工具下载地址：<http://lionet.info/asn1c/download.html>（本文使用版本为0.9.28，Linux）

工具安装：解压缩安装包，编译并安装

工具检测：输入asn1c -h ，若终端打印版本及帮助等信息则安装成功

1.4生成C文件：

在xxxx.asn文件所在目录输入命令：

asn1c  -S /usr/local/share/asn1c -fcompound-names -fskeletons-copy -gen-PER -pdu=auto xxxx.asn （xxxx.asn为需要转换的asn1描述文件）

命令说明：

-S：支持目录 默认为：/usr/local/share/asn1c

-fcompound-names：不加此项编译会报枚举重定义错误信息

-fskeletons-copy：从-S指定目录拷贝asn基础类解码文件，避免依赖必须加此项

-gen-PER：生成PER支持代码

-pdu=auto：生成PDU 数组，定义了所有的PDU消息，不加此选项，不会产生

pdu\_colletion.c文件

运行成功即可在该目录下生成C代码，并包含了示例程序converter-sample.c及

Makefile.am.sample文件，在converter-sample.c中添加需解码的PDU的类型，

运行make -f Makefile.am.sample 即会生成progname的可执行文件。

附录：

#include <iostream>

#include <fstream>

#include <string>

#include <vector>

int main(int argc, char\* argv[])

{

if (argc != 2)

{

return 1;

}

std::string output\_file;

std::string input\_file = argv[1];

int pos = input\_file.find('.');

if (pos == std::string::npos )

{

output\_file = input\_file + ".asn";

}

else

{

output\_file = input\_file.substr(0,pos) + ".asn";

}

std::fstream input;

input.open(input\_file.c\_str(), std::fstream::in );

if ( input.fail() == true)

{

std::cout<<"Please check input file is correct !"<<std::endl;

return 1;

}

std::fstream output;

output.open(output\_file.c\_str(), std::fstream::out );

if ( output.fail() == true)

{

std::cout<<"The output file can not be created here !"<<std::endl

;

return 1;

}

std::string input\_line;

std::vector<std::string > vec\_asn;

std::vector<std::string >::iterator itr;

const unsigned long cul\_asn\_idle = 0x0;

const unsigned long cul\_asn\_start = 0x1;

unsigned long asn\_state = cul\_asn\_idle;

while ( std::getline(input, input\_line) )

{

if ( cul\_asn\_idle == asn\_state )

{

if ( input\_line.find("-- ASN1START") != std::string::npos )

{

asn\_state |= cul\_asn\_start;

}

continue;

}

if ( 0 != (cul\_asn\_start & asn\_state) )

{

if ( input\_line.find("-- ASN1STOP") != std::string::npos )

{

asn\_state = cul\_asn\_idle;

}

else

{

vec\_asn.push\_back(input\_line);

}

}

}

for ( itr = vec\_asn.begin(); itr != vec\_asn.end(); ++itr )

{

output<<\*itr<<std::endl;

}

input.close();

output.close();

return 0;

}