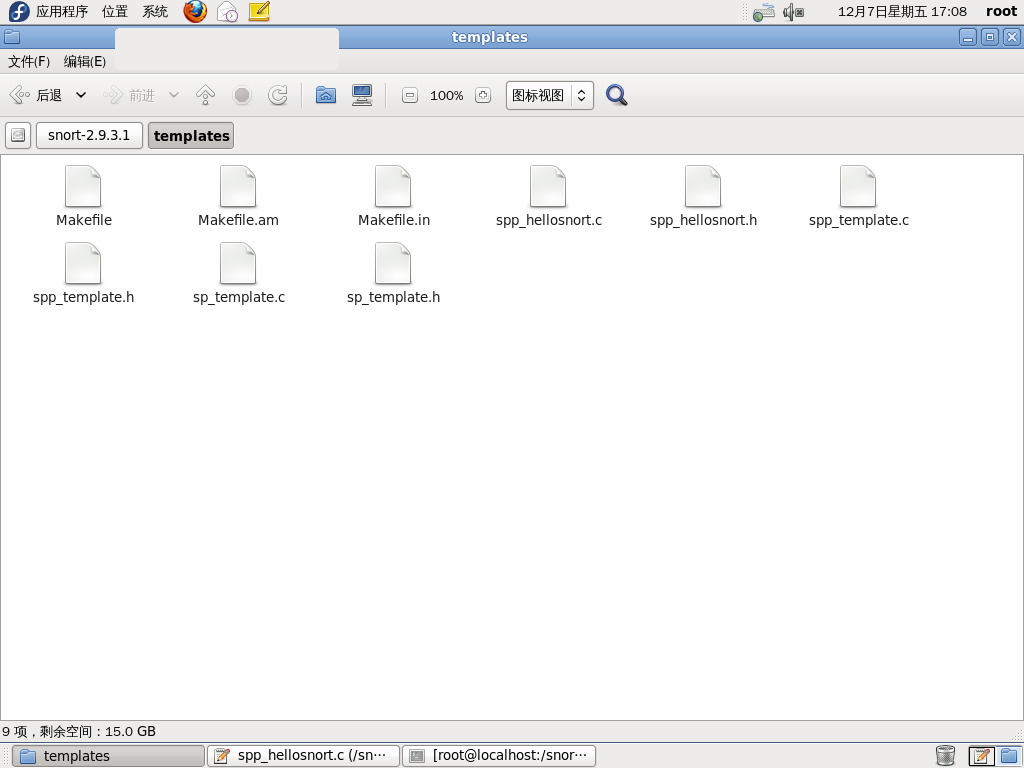
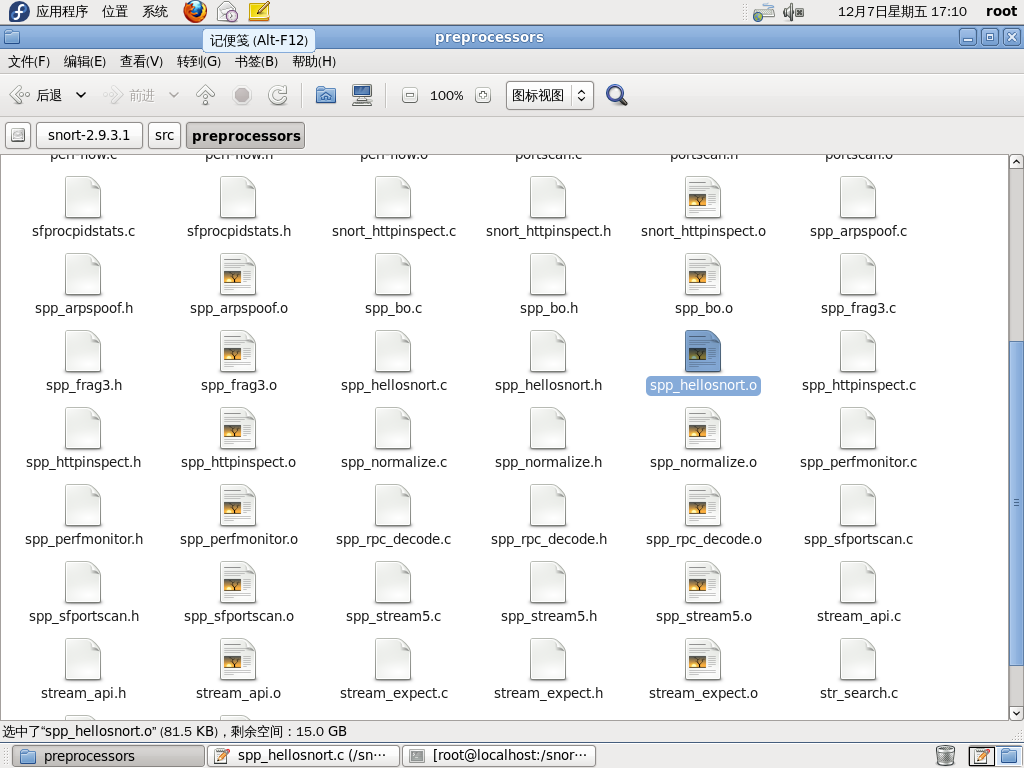
## Snort 2 .9 . 3 预处理器开发步骤

1. 打开模板snort2.9.3.1/templates ,查看文件：sp\_template.h和sp\_template.c是开发预处理插件使用的模板，spp\_template.h和spp\_template.c是开发预处理器使用的模板。这里使用spp\_template.h和spp\_template.c。
2. 复制spp\_template.h和spp\_template.c重命名，得到新的插件

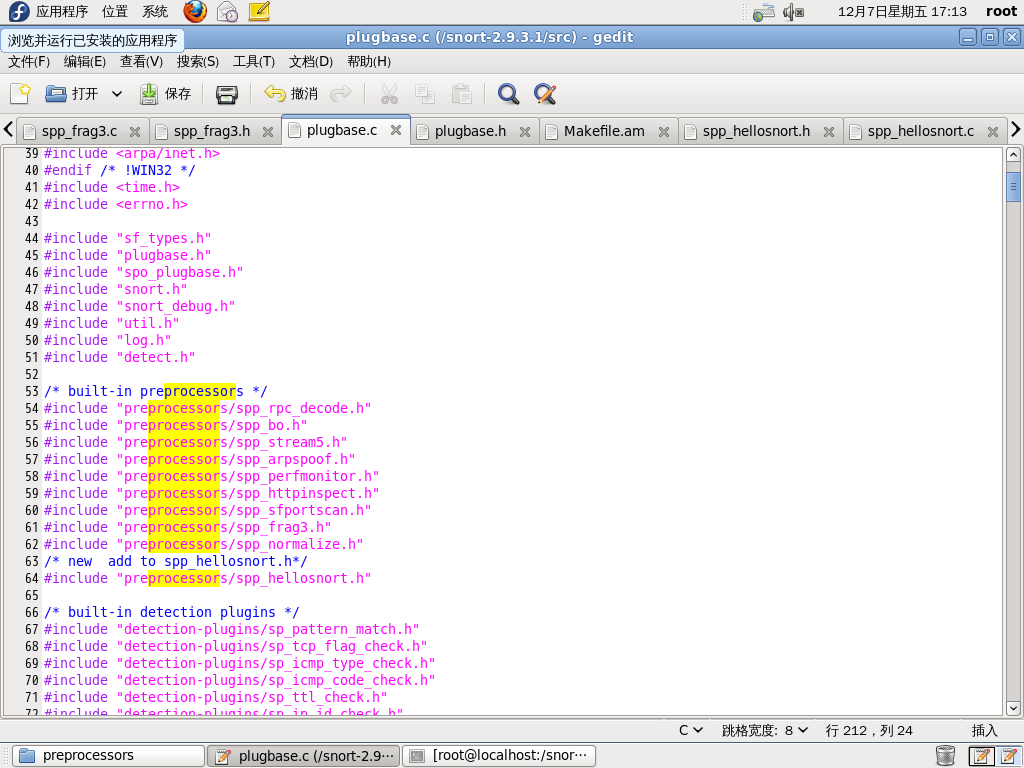


1. 复制spp\_hellosnort.c 和spp\_hellosnort.h 到snort2.9.3.1/src/preprocessors



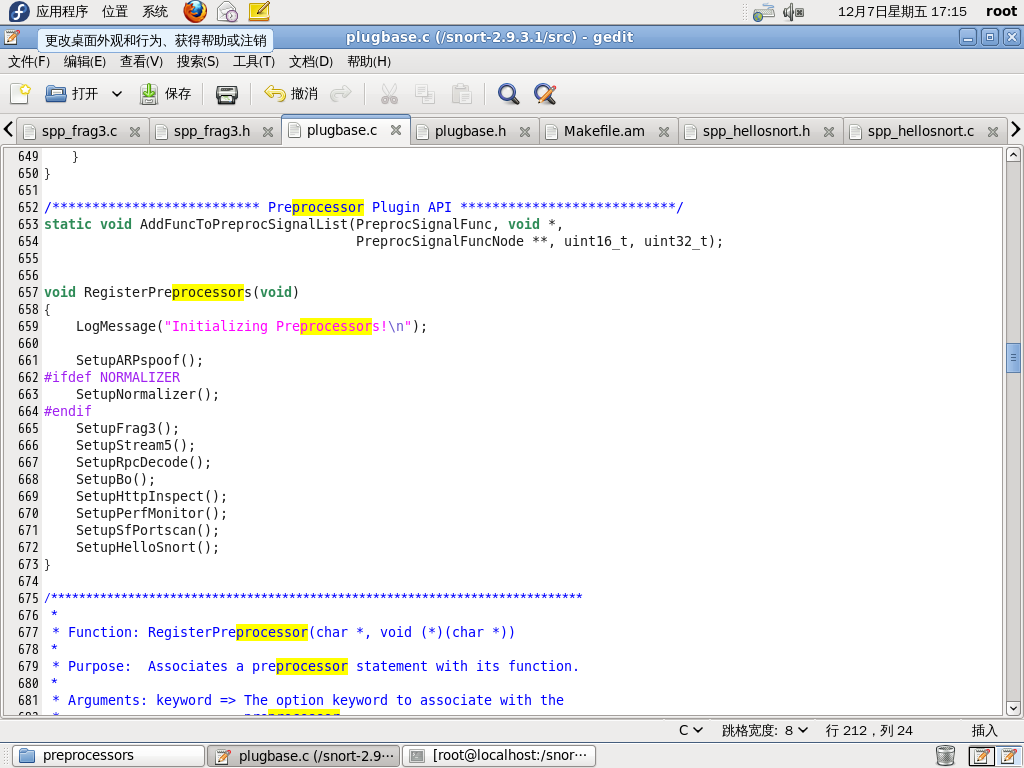
1. 修改plugbase.c 文件，将插件的头文件spp\_hellosnort.h 包含到plugbase.c 中

如图（63、64行）



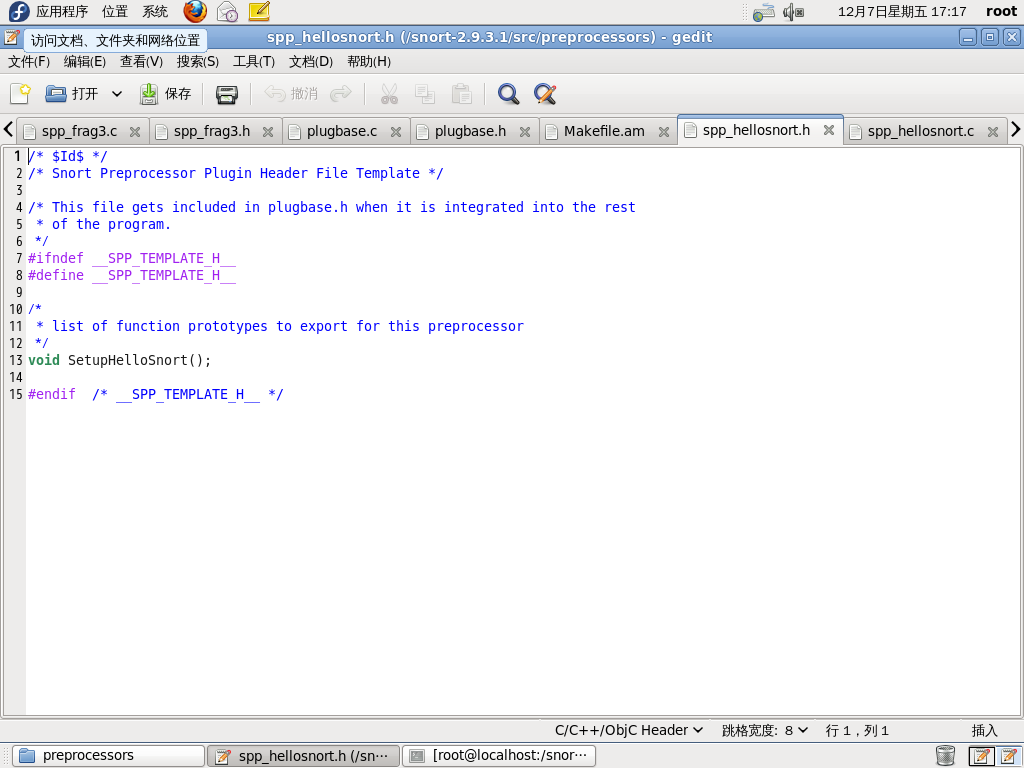
并且将插件的 SetupHelloSnort（）函数插入plugbase.c 的 RegisterPreprocessors()函数中

如图（672行）



1. 修改spp\_hellosnort.h文件，即定义初始化函数原形void SetupHelloSnort ();

如图（13行）



1. 修改spp\_hellosnort.c文件。

需注意地方:1、修改引用的头文件,删除#inlcude “spp\_template.h”，增加#inlcude”spp\_hellosnort.h”

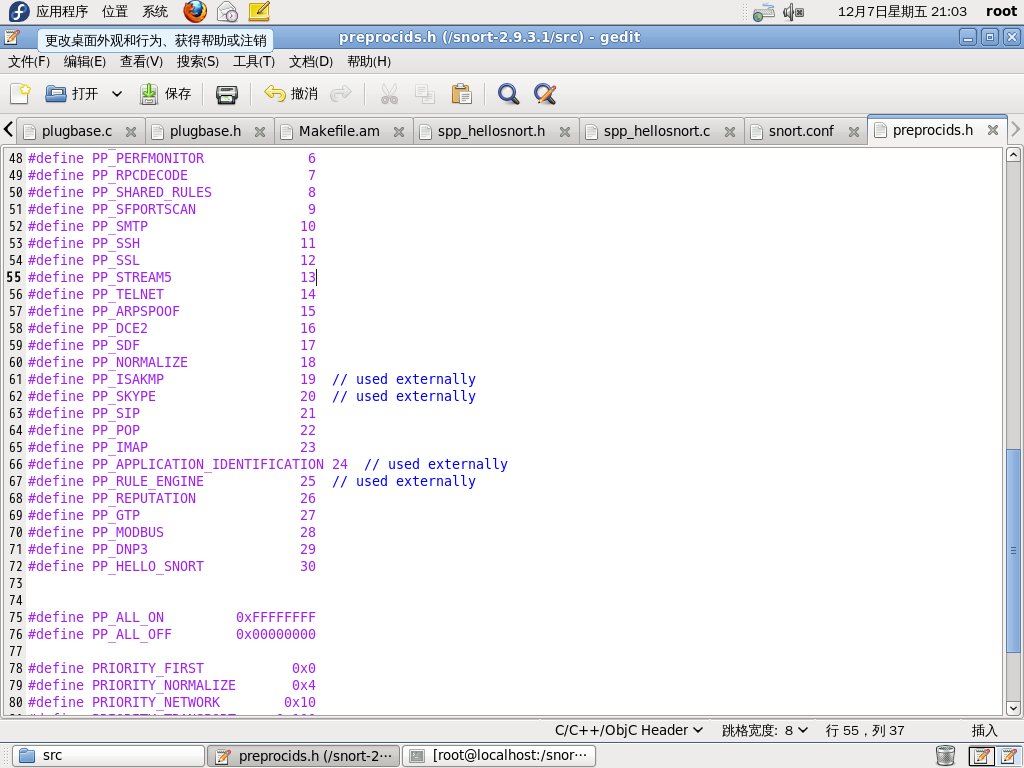
2、增加#define PROTO\_MASK 0x0001（添加函数到预处理器链表是用到）

3、AddFuncToPreprocList(HelloSnortFunct, PRIORITY\_NETWORK, PP\_HELLO\_SNORT, PROTO\_MASK);

此信息中的PRIORITY\_NETWORK为源文件/src/ preprocids.h中引用（图80行）

此信息中的PP\_HELLO\_SNORT 需在源文件 /src/ preprocids.h中定义的最后添加

如图：添加#define PP\_HELLO\_SNORT 30（图72行）



spp\_hellosnort.c文件源码如下（只写出打印信息做调用参考，没有具体功能函数）：

/\* $Id$ \*/

/\* Snort Preprocessor Plugin Source File Template \*/

/\*

\* Purpose:

\*

\* Preprocessors perform some function \*once\* for \*each\* packet. This is

\* different from detection plugins, which are accessed depending on the

\* standard rules. When adding a plugin to the system, be sure to

\* add the "Setup" function to the InitPreprocessors() function call in

\* plugbase.c!

\*

\* Arguments:

\*

\* This is the list of arguements that the plugin can take at the

\* "preprocessor" line in the rules file

\*

\* Effect:

\*

\* What the preprocessor does. Check out some of the default ones

\* (e.g. spp\_frag2) for a good example of this description.

\*

\* Comments:

\*

\* Any comments?

\*

\*/

#include <sys/types.h>

#include <stdlib.h>

#include <ctype.h>

#include <rpc/types.h>

#include "snort\_debug.h"

/\*

\* If you're going to issue any alerts from this preproc you

\* should include generators.h and event\_wrapper.h

\*/

#include "generators.h"

#include "event\_wrapper.h"

#include "util.h"

#include "plugbase.h"

#include "parser.h"

/\*

\* put in other inculdes as necessary

\*/

/\*

\* your preprocessor header file goes here if necessary, don't forget

\* to include the header file in plugbase.h too!

\*/

#include "spp\_hellosnort.h"

#define PROTO\_MASK 0x0001

/\*

\* define any needed data structs for things like configuration

\*/

typedef struct \_TemplateData

{

/\* Your struct members here \*/

} TemplateData;

/\*

\* If you need to instantiate the preprocessor's

\* data structure, do it here

\*/

TemplateData SomeData;

/\*

\* function prototypes go here

\*/

static void HelloSnortInit(char \*);

static void ParseTemplateArgs(char \*);

static void HelloSnortFunct(Packet \*);

static void PreprocCleanExitFunction(int, void \*);

static void PreprocRestartFunction(int, void \*);

static void helloSnortreloadFuction(char \*args);

/\*

\* Function: SetupTemplate()

\*

\* Purpose: Registers the preprocessor keyword and initialization

\* function into the preprocessor list. This is the function that

\* gets called from InitPreprocessors() in plugbase.c.

\*

\* Arguments: None.

\*

\* Returns: void function

\*

\*/

void SetupHelloSnort()

{

/\*

\* link the preprocessor keyword to the init function in

\* the preproc list

\*/

#ifndef SNORT\_RELOAD

RegisterPreprocessor("Hello\_Snort", HelloSnortInit);

#else

RegisterPreprocessor("Hello\_Snort", HelloSnortInit,

helloSnortreloadFuction, NULL, NULL);

#endif

printf("now call the setupHelloSnort <><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>");

DEBUG\_WRAP(DebugMessage(DEBUG\_PLUGIN,"Preprocessor: HelloSnort is setup...\n"););

}

/\*

\* Function: TemplateInit(u\_char \*)

\*

\* Purpose: Calls the argument parsing function, performs final setup on data

\* structs, links the preproc function into the function list.

\*

\* Arguments: args => ptr to argument string

\*

\* Returns: void function

\*

\*/

static void HelloSnortInit(char \*args)

{

DEBUG\_WRAP(DebugMessage(DEBUG\_PLUGIN,"Preprocessor: HelloSnortInit Initialized\n"););

printf("HelloSnortInit ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^is setup");

/\*

\* parse the argument list from the rules file

\*/

//ParseTemplateArgs(args);

/\*

\* perform any other initialization functions that are required here

\*/

/\*

\* Set the preprocessor function into the function list

\*/

//example: AddFuncToPreprocList(PreprocEvalFunc pp\_eval\_func, uint16\_t priority,uint32\_t preproc\_id, uint32\_t proto\_mask)

AddFuncToPreprocList(HelloSnortFunct, PRIORITY\_NETWORK, PP\_HELLO\_SNORT, PROTO\_MASK);

// AddFuncToCleanExitList(PreprocCleanExitFunction, NULL);

// AddFuncToRestartList(PreprocRestartFunction, NULL);

}

/\*

\* Function: ParseTemplateArgs(char \*)

\*

\* Purpose: Process the preprocessor arguements from the rules file and

\* initialize the preprocessor's data struct. This function doesn't

\* have to exist if it makes sense to parse the args in the init

\* function.

\*

\* Arguments: args => argument list

\*

\* Returns: void function

\*

\*/

static void ParseTemplateArgs(char \*args)

{

/\* your parsing function goes here, check out the other spp files

for examples \*/

}

/\*

\* Function: PreprocFunction(Packet \*)

\*

\* Purpose: Perform the preprocessor's intended function. This can be

\* simple (statistics collection) or complex (IP defragmentation)

\* as you like. Try not to destroy the performance of the whole

\* system by trying to do too much....

\*

\* Arguments: p => pointer to the current packet data struct

\*

\* Returns: void function

\*

\*/

static void HelloSnortFunct(Packet \*p)

{

/\* your preproc function goes here.... \*/

/\*

\* if you need to issue an alert from your preprocessor, check out

\* event\_wrapper.h, there are some useful helper functions there

\*/

printf("the HelloSnort`s Main function HelloSnortFunct is here^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^");

}

/\*

\* Function: PreprocCleanExitFunction(int, void \*)

\*

\* Purpose: This function gets called when Snort is exiting, if there's

\* any cleanup that needs to be performed (e.g. closing files)

\* it should be done here.

\*

\* Arguments: signal => the code of the signal that was issued to Snort

\* data => any arguments or data structs linked to this

\* functioin when it was registered, may be

\* needed to properly exit

\*

\* Returns: void function

\*/

static void PreprocCleanExitFunction(int signal, void \*data)

{

/\* clean exit code goes here \*/

}

static void helloSnortreloadFuction(char \*args)

{

printf("call the reload hellosnort");

}

/\*

\* Function: PreprocRestartFunction(int, void \*)

\*

\* Purpose: This function gets called when Snort is restarting on a SIGHUP,

\* if there's any initialization or cleanup that needs to happen

\* it should be done here.

\*

\* Arguments: signal => the code of the signal that was issued to Snort

\* data => any arguments or data structs linked to this

\* functioin when it was registered, may be

\* needed to properly exit

\*

\* Returns: void function

\*/

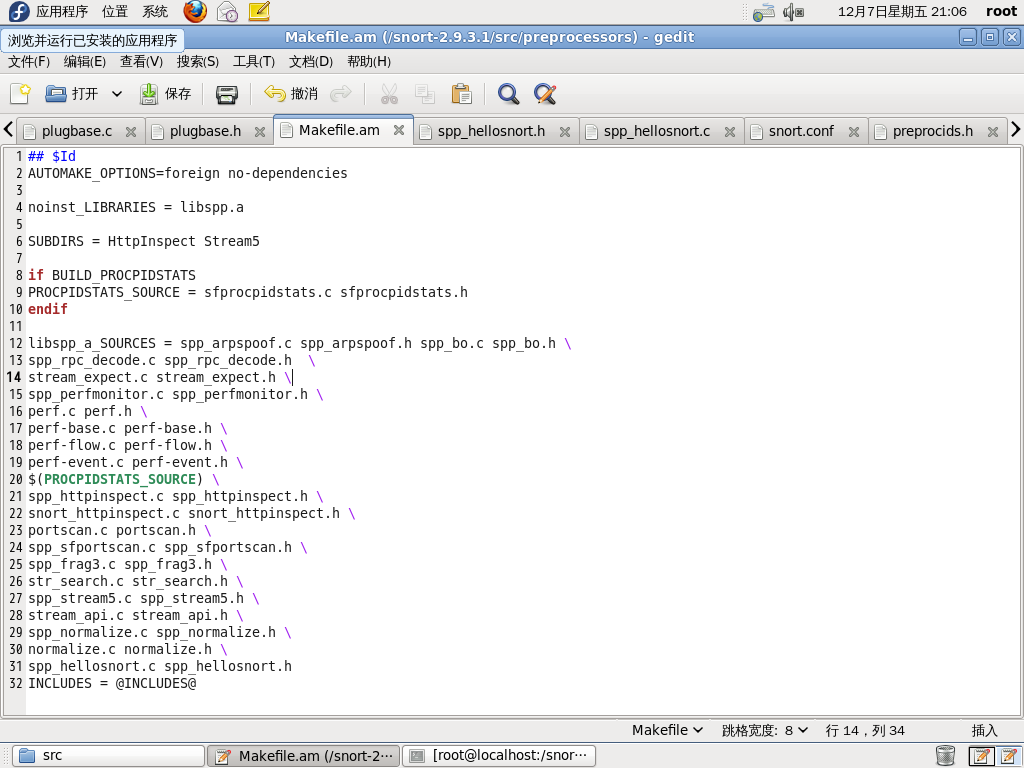
static void PreprocRestartFunction(int signal, void \*foo)

{

/\* restart code goes here \*/

}

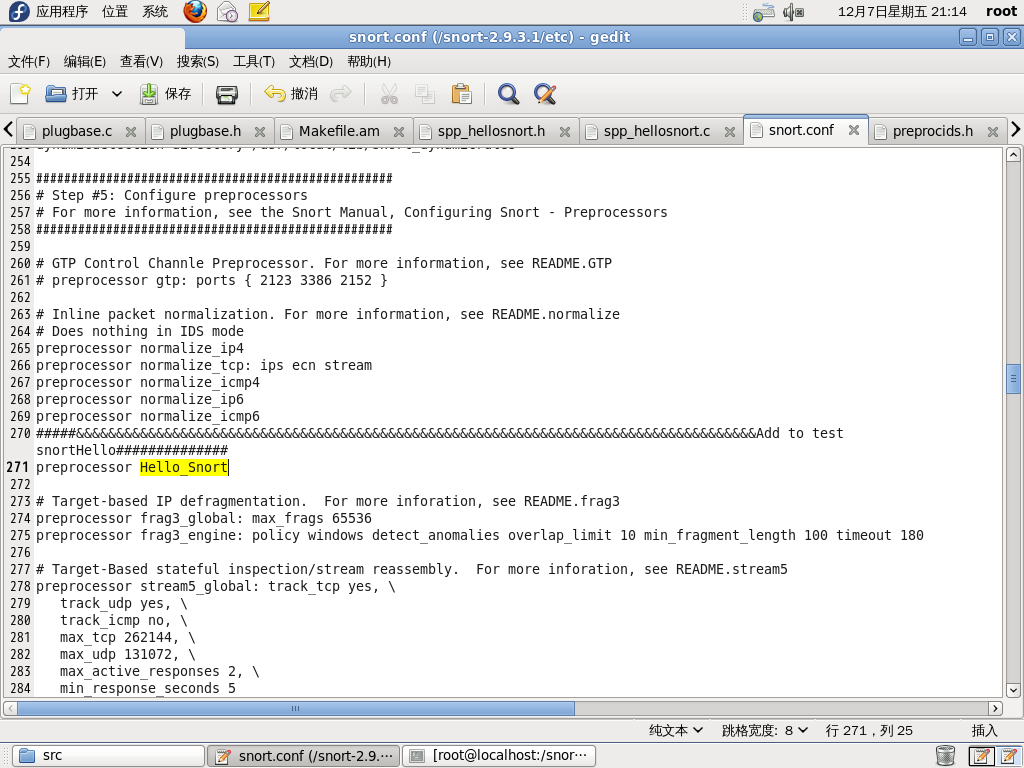
1. 修改snort2.9.3.1/src/preprocessors/Makefile.am文件如下图添加信息（图31行）



8、打开/etc/snort.conf

开启Hello\_Snort（此关键字对应spp\_helloSnort.c中setupHelloSnort函数中的RegisterPreprocessor("Hello\_Snort", HelloSnortInit);注册的Hello\_Snort）预处理器

如图（271行）



返回snort根目录 命令行输入automake 自动修改makefile（/src/preprocessors/目录下）文件

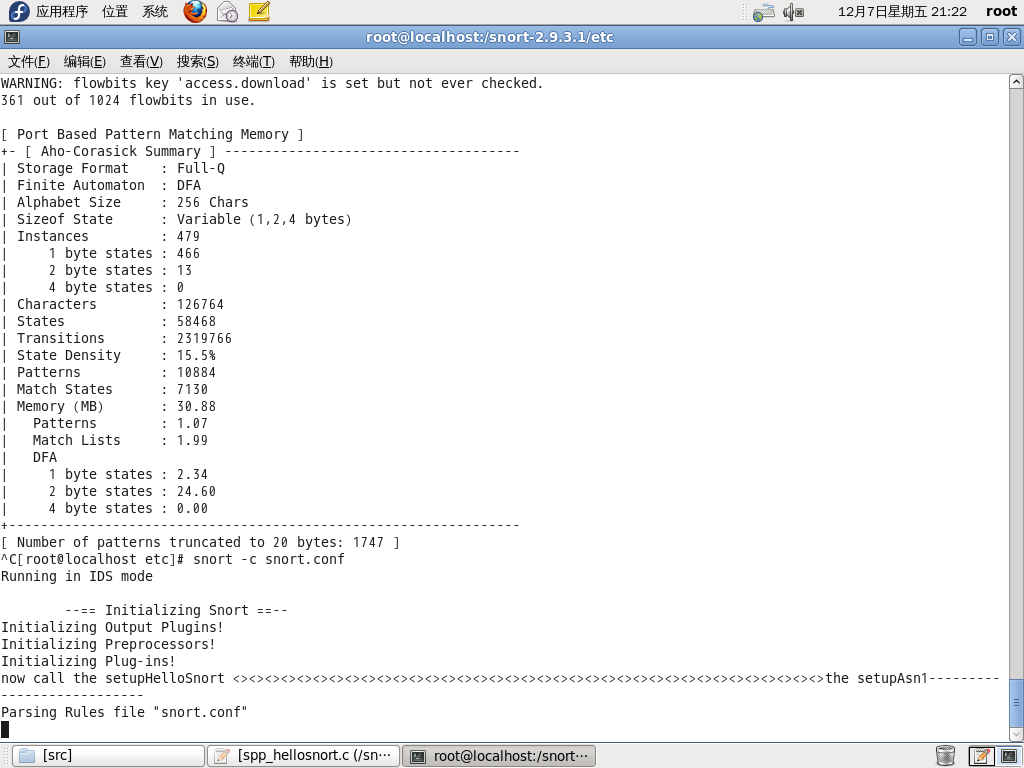
（

或者手动修该/src/preprocessors/makefile文件

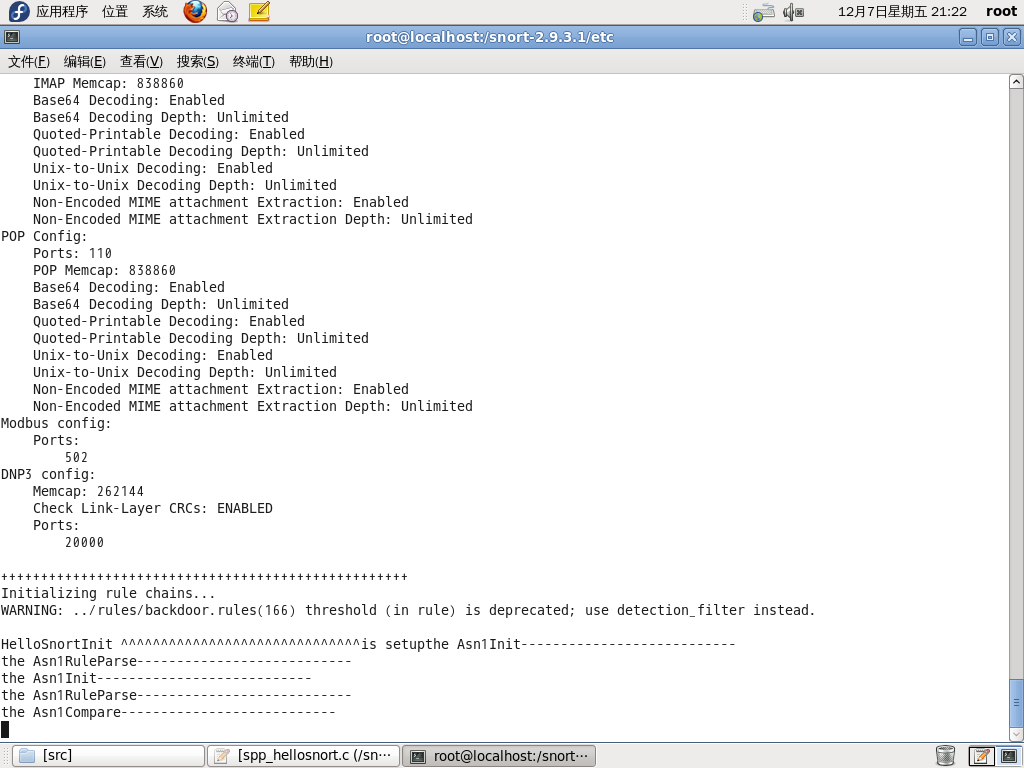
1. 第61行 加入spp\_hellosnort.c spp\_hellosnort.h
2. 第71行 加入 spp\_hellosnort.$(OBJEXT)
3. 第282行 make all 的前面即可 加入 spp\_hellosnort.c spp\_hellosnort.h

）

9、（重新编译snort 后）启动snortIDS模式启动插件的时候可以看到Hello\_Snort预处理的启动过程中输入启动信息（spp\_helloSnort.c中的输入打印信息）

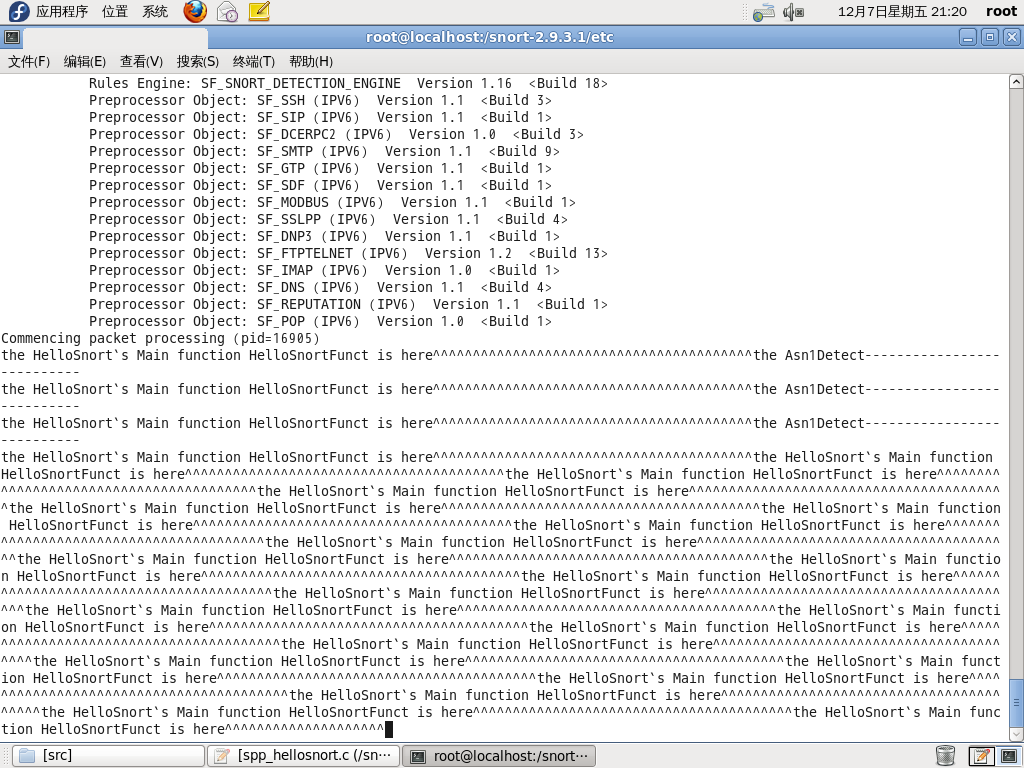
如下图 

生成规则链表的时候的启动信息如下图：



启动成功后每通过一个数据包，就会调用预处理插件并输出信息

如下图



每Ping一次该主机，snort预处理器都会产生输入报文如下图：

