从 Windows 向 Linux 的 C/C++代码移植

一 准备移植

1 熟悉 linux 编程环境

⑴ linux 版本：
redhat 系列： redhat (7.2 , 8.0, 9.0, AS *, Fedora Core *)
debian 系列：ubuntu (6.06 , …)
suse 系列： suse (9.0 , …)
turbo linux
红旗 linux
…….

⑵ linux 内核：老版本多是 kernel2.4 或者更早，新版本多是 kernel2.6（内核更新主要是提高了系统性能及稳定性）

⑶ 典型 linux 开发环境：
① Shell 命令控制台：bash（最常用，大多讲解 shell 编程的资料都基于 bash），csh，ksh， 。。。
常用命令：
man 查看联机手册（包括命令参数，函数返回值等等）
ps 显示进程状态 （ps -ax）
top 显示实时 CPU 及内存状态 （shift + p，按 CPU 状态排序；shift + m，按内存状态排序）

ls 列出符合条件的文件或目录 （-R 递归输出； -l 详细输出，ls –l 在部分 linux 上可简写为 ll ）
pwd 列出当前路径
cd 切换到某个路径 （cd 切换到用户目录, cd . 切换到当前目录 , cd .. 切换到上一级目录, cd – 切换到上一次操作所在目录）
mkdir 创建目录 （rmdir 删除目录）
find 查找符合条件的文件或目录 （例 find . –name “*”，其中 “.” 代表在当前目录下查找，若 “/” 则代表在根目录下查找）
rm 删除文件或目录 （-r 递归，-f 非交互式）
cp 拷贝文件或目录 （-r 递归，-f 非交互式 ）
mv 移动文件或目录
cat 显示文本内容 （-A 显示所有内容，包括回车换行制表符等空白字符）
ln 建立链接文件 （例 ln –s source dist）
nm 列出目标文件中的符号 （例 nm lib***.a | grep）
ldd 列出共享库依赖
awk 按格式分割文本并输出 （例 awk –F : “{print $1}”）
grep 列出符合条件的行 （例 ps –ax | grep sshd ; grep –n –r “pattern” *）
sed 列出符合条件的行或替换行 （sed –n “/123/p” 123.txt ; sed –n “s/123/321/p” 123.txt）

② gcc，g++编译器：gcc 可以对 c/c++代码进行编译，g++可以对 c++代码进行编译，gcc 和 g++在编译 c++代码时，在编译阶段效果是相同的，在链接阶段 g++会自动链接上标准 c++库 libstdc++***.so，而 gcc 则需要手动加上 -lstdc++。

③ gdb 调试器：可对由编译器生成的程序进行调试，如何调试以及常用命令请参见附录 3：如何使用 gdb 进行调试及常用命令。

④ 可选开发工具及编译环境：
Shell 脚本，perl 脚本，icc 编译器（ihmmparam.dsp：source/share/ihmmparam.cpp）

2 了解 windows 与 linux 的 C/C++代码的差异

⑴ 关于路径分割符“/”和“\”：

“\”路径分隔符在 linux 上不支持，需要都改为“/”

⑵ 文件名大小写：

Windows 下文件名大小写不敏感，而在 linux 下文件名大小写敏感，代码中需严格遵照文件名大小写，否则编译报错“No Such file or directory”。

⑶ for 作用范围：

Windows 下 for 中变量定义可以应用在所在函数体接下来的部分，而在 linux 下 for 中变量定义只应用在 for 循环体内部，即在 windows 下，如下代码编译不报错：

```
for (int i=0; i<10; i++)
for (i=0; i<100; i++)
```

而在 linux 下，如上代码报错，需做如下修改：

```
int i;
for ( i=0; i<10; i++)
for (i=0; i<100; i++)
```

⑷ 另外 gcc/g++编译中语法判断较 VC 更严格，早期的 gcc2.96 相对不很严格，但编译生成程序的性能及优化程度都比高版本（目前最高版本 4.*）的要弱，所以建议都按较严格语法来编译代码，

例：在类中声明友类时，

Class

```
{
    friend myclass1;
 }
```
在 windows 下编译没有错，在 linux 下就会报错，
改为如下编译通过：
```
Class
{
    friend class myclass1;
 }
```

二 着手移植

1 撰写 Makefile 并着手编译

参考 windows 下的 dsp 文件，来创建对应的 Makefile 文件。

⑴ 明确 Makefile 内容：
生成程序文件名，.cpp 或.c 文件列表及其搜索路径，头文件包含路径，库文件链接路径，编译参数等。例子可参见附录 1：编译 TSR_API 工程所用的 Makefile

⑵ 执行 Makefile，开始编译：
make -f ***.makefile (如果 Makefile 以“makefile”或“Makefile”命名，可不带 –f 参数)
⑶ 查看并分析编译错误：

① 提示文件没找到：

首先排除一 2（1）（2）两种情况，然后检查 Makefile(VPATH , INC_***_PATH)，看文件路径是否可搜索到，如果是 windows 下的头文件，可采用条件编译 ifdef WIN32 方式注释掉，或者在可搜索路径下生成一个同名空文件替代。

② for 循环中变量定义报错：

采用前述 一 2（3）的方法解决。

③ 出现无定义的 windows 系统函数：

这种情况大多可以找到一个 linux 下相对应的函数，例如 _access -> access , stricmp -> strcasecmp ，（在 linux 下查看系统函数手册，使用 man ，有时需要加参数 2 或 3 以避免和命令混淆，例 man 2 mkdir ；查找系统函数，使用 man –k keyword；还找不到的话，或者确实没有对应函数，或者还是用 google 搜吧），如果找不到对应的函数的话，一种办法就是将几个函数组合实现其对应功能，另一种办法就是得修改功能实现的机制了。

我在以前的移植工作中归纳了一些 windows 和 linux 函数的对应转换，可参见附录 2：portingconfig.h 。

⑷ 编译成功，开始链接：

若链接失败，查看所需链接库文件路径（LIB_***_PATH）是否在 Makefile 文件中已指明，并确认库文件的正确性（系统库文件如 stl，ipp 需要保证安装配置正确；自己编译生成的库文件则需验证其内部符号定义是否完备）。

⑸ 链接成功，运行程序，看有无异常：

常见 segment fault 错误，多是内存越界，linux 默认不生成 core 文件，需修改系统配置：ulimit –c unlimited，然后就需要使用 gdb 来调试程序，具体如何调试和常用命令可参见附录 3：如何使用 gdb 进行调试及常用命令。

2 ACE 的使用

ACE 的使用可以消除 windows 和 linux 之间函数实现的差异，目前主要用于 dsr 的网络架构上，我在以前的移植工作中，对有关线程，互斥锁，事件的代码都做了替换：

```
#ifdef __USE_ACE_THREADS__
#include "ace/Synch.h"
#endif
```

⑴ 创建线程：

```
#ifndef __USE_ACE_THREADS__
        if ( (TestAgentHandle[i]=CreateThread(NULL, 0, TestAgent, (LPVOID)&TestAgentId[i], 0, &TestAgentThreadId[i]))==NULL )
#else
        if                  (ACE_Thread::spawn((ACE_THR_FUNC)TestAgent,(LPVOID)&TestAgentId[i],THR_NEW_LWP                  |
THR_JOINABLE,&TestAgentThreadId[i],&(ACE_hthread_t)TestAgentHandle[i]) == -1)
#endif
```

⑵ 获取当前线程 id：

```
#ifndef __USE_ACE_THREADS__
    TestAgentThreadId[id] = GetCurrentThreadId();
#else
    TestAgentThreadId[id] = ACE_Thread::self();
#endif
```

⑶ 结束线程：

```
#ifndef __USE_ACE_THREADS__
    WaitForMultipleObjects(opt_DecoderNum, TestAgentHandle, true, INFINITE);
#else
    for (i=0; i<opt_DecoderNum; i++)
```

```
    {
        int ret_val = ACE_Thread::join(TestAgentHandle[i]);
        if (ret_val == -1)
        exit(-1);
    }
#endif
```

⑷ 挂起线程：

```
#ifndef __USE_ACE_THREADS__
                int count = SuspendThread((HANDLE)CTSRInstance::instanceList[j]->GetThreadHandle());
#else
                int count = ACE_Thread::suspend((HANDLE)CTSRInstance::instanceList[j]->GetThreadHandle());
#endif
```

⑸ 继续线程：

```
#ifndef __USE_ACE_THREADS__
                int count = ResumeThread((HANDLE)CTSRInstance::instanceList[j]->GetThreadHandle());
#else
                int count = ACE_Thread::resume((HANDLE)CTSRInstance::instanceList[j]->GetThreadHandle());
#endif
```

⑹ 创建 mutex：

```
#ifndef __USE_ACE_THREADS__
    hTSRInstanceLock = CreateMutex(NULL, false, NULL);
#else
    hTSRInstanceLock = new ACE_Recursive_Thread_Mutex();
#endif
```

⑺ 获取锁，释放锁：

```
#ifndef __USE_ACE_THREADS__
```

```
    CTSRLockMutex(HANDLE hLock)
    {
        h = hLock;
        WaitForSingleObject(h, INFINITE);
    }
    ~CTSRLockMutex()
    {
        Release();
    }
    inline int Release()
    {
        if (h!=NULL)
        {
            ReleaseMutex(h);
            h = NULL;
        }
        return 0;
    }
    inline void Clear()
    {
        h = NULL;
    }
#else
    CTSRLockMutex(ACE_Recursive_Thread_Mutex *pLock)
    {
        p = pLock;
```

```
            p->acquire();
        }
        ~CTSRLockMutex()
        {
            Release();
        }
        inline int Release()
        {
            if (p!=NULL)
            {
                p->release();
                p = NULL;
            }
            return 0;
        }
        inline void Clear()
        {
            p = NULL;
        }
#endif
```

⑻ 创建事件：

```
#ifndef __USE_ACE_THREADS__
    eNewDataComing =    CreateEvent(NULL, false, false, NULL);
#else
    eNewDataComing = new ACE_Auto_Event();
#endif
```

⑼ 设置事件：

```
#ifndef __USE_ACE_THREADS__
    SetEvent(eNewDataComing);
#else
    eNewDataComing->signal();
#endif
```

⑽ 等待事件：

```
#ifndef __USE_ACE_THREADS__
                retVal = WaitForSingleObject(eNewDataComing, 400);
                if (retVal == WAIT_OBJECT_0)
                {
                    watchDog.Refresh();
                    break;
                }
#else
                ACE_Time_Value time_wait(0,400*1000);
                retVal = eNewDataComing->wait(&time_wait,0);
                eNewDataComing->reset();
                if (retVal == 0)
                {
                    watchDog.Refresh();
                    break;
                }
#endif
```

三 验证移植正确性

初步运行程序无异常后，就要开始验证 windows 和 linux 程序运行结果的一致性了，这时候就要使用到 shell 命令，shell 脚本，perl 脚本来对日志进行分析比对，结果（识别率，打分）验证无误后，接着就是压力测试（多线测试，直到系统最大极限），然后循环测试多天，以验证程序的稳定性。

附录 1：编译 TSR_API 工程所用的 Makefile

```
#Please change root directory to your own setting
ROOT=$(PWD)/../..

#Place to copy the target
TARGETDIR=$(ROOT)/lib

#Compiler settings
CCACHEPATH=/usr/local/bin
CC=$(CCACHEPATH)/ccache gcc
ICCPATH=/opt/intel/cc/9.0/bin
ICC=$(ICCPATH)/icc

#Source code searching path
VPATH = ../../source/linux:../../source/common:../../source/grammar:\
../../source/mergedlib:../../source/PSAPI:../../source/RS:../../source/share:\
../../source/tools:../../source/cluster:../../source/detector:\
```

../../source/recognizer:../../source/irecognizer:../../source/log:../../source/system:\
../../source/tools/cache:../../source/tools/DB:\
../../source/tools/Network:../../source/tools/security:../../source/tools/telnet

#include directories
INC_ACE_PATH=/home/ftproot/tools/lib-source/ACE_wrappers
INC_GCC_PATH=/usr/include/c++/3.2.2
INC_STL_PATH=/usr/local/include/stlport
INC_ICC_PATH=/opt/intel/cc/9.0/include
INC_IPP_PATH=/opt/intel/ipp41/ia32_itanium/include

#For compiler setting with GCC
INCFLAGS=-include $(ROOT)/source/linux/portingconfig.h -I$(ROOT)/source/linux -I${INC_ACE_PATH} -I${INC_STL_PATH}

#For compiler setting with intel compiler
INCFLAGS2=-include $(ROOT)/source/linux/portingconfig.h -I$(ROOT)/source/linux -I${INC_ACE_PATH} -I${INC_STL_PATH} -I${INC_ICC_PATH} -I${INC_ICC_PATH}/c++

#Normal compilation flags
#CFLAGS = -w -g -D _MT -D_DEBUG -MP -MMD
CFLAGS = -w -O3 -D _MT -D__USE_ACE_THREADS__ -MP -MMD

#library searching path
LIB_ACE_PATH = /home/ftproot/tools/lib-source/ACE_wrappers/lib
LIB_STL_PATH = /usr/local/lib
LIB_IPP_PATH = /opt/intel/ipp41/ia32_itanium/sharedlib

```
LIB_IDECODER_PATH = $(ROOT)/lib
XML_LIB_PATH = /home/yili/mrcpstack/lib
LIB_FTP_PATH = /home/yili/LibFtp/lib_ftp
LIB_G2P_API_PATH = /home/yili/test_G2P/G2P_API

#objects for the targets
OBJECTS1 = DataCollection.o globalOpt.o modeltable.o pure_api.o TestPoint.o TSR_Registry.o \
        TSR_SpeechTag.o TSR_wrapper.o TSRDLL_Main.o
OBJECTS2 = CompileUI.o FSGparser.o gramformat.o GrammarFile.o Lexicon.o \
                PronDict.o RuleHash.o
OBJECTS3 = dumpmemusage.o getopt.o options.o
OBJECTS4 = CFeatExtract.o DecoderMain.o DecoderSchedulor.o lpc.o mfcc.o \
        onlinecms.o
OBJECTS5 = check_harmonic.o Detector.o Interpolation.o \
                postprocess.o
#OBJECTS6 = tprintf.o
OBJECTS6 =
#iplm.o
OBJECTS7 = algwg.o    aligndecoder.o alignnbest.o aligntrace.o chunk.o \
                cmmap.o graphnbest.o graphtrace.o ialgwg.o ialigndecoder.o \
                ilattnet.o iStateGraph.o lattnet.o mappedhmm.o \
                mngtrace.o nbestlist.o NBestScore.o treenbest.o treetrace.o \
                tstatephoneposterior.o
OBJECTS8 = chunkfile.o CodecConvert.o dcstree.o dirutil.o feat.o hmm98.o \
                hmmcd.o hmmcddcstree.o hmmcdonetoone.o hmmsdc.o hmmsdcdcstree.o hmmsdconetoone.o \
                HMMSimplified.o icache.o iexscan.o ifeat.o igmihys.o ihmmmap.o \
```

```
		ihmmparamcompressed.o ihmmphys.o ihmmsdcparam.o imem.o inamehash.o iqlpars.o iqlscan.o isdtexception.o mapfile.o monophone.o \
		phnenc.o question.o readwrite.o
OBJECTS9 = ihmmparam.o
OBJECTS10 = portingconfig.o
#mergedlib
#OBJECTS11 = ippmerged.o

#G2P
OBJECTS12 = monomap.o

#logfile.o

OBJECTS = $(OBJECTS1) $(OBJECTS2) $(OBJECTS3) $(OBJECTS4) $(OBJECTS5) $(OBJECTS6) $(OBJECTS7) $(OBJECTS8) $(OBJECTS9) $(OBJECTS10) $(OBJECTS12)

#For this project
TARGET=libTSR_API.so

all: $(TARGET)

#generate the dynamic library
$(TARGET) :: $(OBJECTS)
	$(CC) $(CFLAGS) -fPIC -shared -Wl,-soname,libTSR_API.so -L/home/ftproot/xml-xalan/c/lib -lxalanMsg -lxerces-c -lxalan-c -L$(LIB_IPP_PATH) -lipps -lippsr -L$(LIB_ACE_PATH) -lACE -L$(LIB_STL_PATH) -lstlport_gcc -L$(LIB_G2P_API_PATH) -lG2P_API -o $(TARGET) $(OBJECTS) $(LIB_IDECODER_PATH)/libidecoder.a $(XML_LIB_PATH)/libXML2ABNF.a
```

```
/home/yili/log4cxx/TITLog/liblog4cxx.a $(LIB_FTP_PATH)/libFtp.a
	cp -f $(TARGET) $(TARGETDIR)

#include dependencies generated automatically
-include $(OBJECTS:.o=.d)

#These two files must be built with icc
ihmmparam.o: ihmmparam.cpp
	$(ICC) $(INCFLAGS2) $(CFLAGS) -c $<

TSR_wrapper.o: TSR_wrapper.cpp
	$(ICC) $(INCFLAGS2) $(CFLAGS) -c $<

#implicit rules for source code compilation
%.o: %.cpp
	$(CC) $(INCFLAGS) $(CFLAGS) -c $<

%.o: %.c
	$(CC) $(INCFLAGS) $(CFLAGS) -c $<

.PHONY: clean cleanall rebuild
clean:
	rm -f *.o
	rm -f *.d
cleanall: clean
	rm -f $(TARGET)
```

```
rebuild: cleanall
	make
```

附录 2：portingconfig.h

```
/* head file for porting codes from windows to linux */

#ifndef _PORTINGCONFIG_H
#define _PORTINGCONFIG_H 1

#ifdef __linux__

#include <sys/timeb.h>
#include <time.h>
#include <unistd.h>
#include <stdio.h>
#include <limits.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <string.h>
#include <strings.h>
//for socket
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
```

```
#include <netdb.h>

#define Sleep(x) usleep((x)*1000)
#define _strdate strdate
#define _strtime strtime
#define _timeb timeb
#define _ftime ftime
#define _access access
#define _tempnam tempnam
#define _getcwd getcwd
#define _fullpath(x,y,z) realpath(y,x)      //linux: PATH_MAX = 4096
#define _mkdir(x) mkdir(x,0755)
#define CreateDirectory(x,y) mkdir(x,0755)
#define stricmp strcasecmp
#define strnicmp strncasecmp
#define _itoa(x,y,z) sprintf(y,"%d",x)
#define _vsnprintf vsnprintf
#define _snprintf snprintf
#define GetCurrentProcessId() getpid()

#undef LINKDLL
#undef __stdcall
#undef __declspec
#undef __cdecl
#undef dllimport
#undef WINBASEAPI
```

```
#undef WINAPI
#undef _cdecl

#define LINKDLL
#define __stdcall
#define __declspec(x)
#define __cdecl
#define dllimport
#define WINBASEAPI
#define WINAPI
#define _cdecl
#define LOG4CXX_EXPORT

//note: strlen(buf) must >= 8
char * strdate(char *buf);
char * strtime(char *buf);

int _kbhit();      //_kbhit for linux
char *strlwr(char *str);    //strlwr for linux
char *chfilesep(char *str);

#define HANDLE int
typedef unsigned long    DWORD;
typedef void *LPVOID;
typedef void *PVOID;
#define BOOL bool
```

```
typedef void *HMODULE;
typedef void *HINSTANCE;
typedef char *LPSTR;
typedef const char *LPCSTR;
typedef wchar_t WCHAR;
typedef WCHAR *LPWSTR;
typedef const WCHAR *LPCWSTR;
typedef DWORD *LPDWORD;
typedef unsigned long UINT_PTR;
typedef UINT_PTR SIZE_T;
#define _MAX_PATH      260 /* max. length of full pathname */
typedef unsigned short WORD;

#ifndef VOID
#define VOID void
typedef char CHAR;
typedef short SHORT;
typedef long LONG;
#endif

#define STATUS_TIMEOUT                          ((DWORD      )0x00000102L)
#define WAIT_TIMEOUT                              STATUS_TIMEOUT
#define STATUS_WAIT_0                          ((DWORD      )0x00000000L)
#define WAIT_OBJECT_0           ((STATUS_WAIT_0 ) + 0 )
#define STATUS_ABANDONED_WAIT_0                 ((DWORD      )0x00000080L)
#define WAIT_ABANDONED              ((STATUS_ABANDONED_WAIT_0 ) + 0 )
```

```
#ifndef _WAVEFORMATEX_
#define _WAVEFORMATEX_
typedef struct tWAVEFORMATEX
{
    WORD        wFormatTag;         /* format type */
    WORD        nChannels;          /* number of channels (i.e. mono, stereo...) */
    DWORD       nSamplesPerSec;     /* sample rate */
    DWORD       nAvgBytesPerSec;    /* for buffer estimation */
    WORD        nBlockAlign;        /* block size of data */
    WORD        wBitsPerSample;     /* Number of bits per sample of mono data */
    WORD        cbSize;             /* The count in bytes of the size of
                                        extra information (after cbSize) */

} WAVEFORMATEX;
typedef WAVEFORMATEX            *PWAVEFORMATEX;
#endif /* _WAVEFORMATEX_ */

#ifndef _DEBUG
#define _ASSERT(expr) ((void)0)
#else
#define _ASSERT(expr) do { if (!(expr)) exit(-1);} while (0) //???
#endif

#undef __try
```

```
#undef __finally

#define __try
#define __finally

//#define INFINITE                0xFFFFFFFF    // Infinite timeout
#define INFINITE          -1    // Infinite timeout

#ifndef FALSE
#define FALSE                   0
#endif

#ifndef TRUE
#define TRUE                    1
#endif

typedef unsigned int            UINT;
typedef UINT WPARAM;
typedef LONG LPARAM;
#define MAX_PATH                260

typedef struct _SYSTEM_INFO {
    union {
        DWORD dwOemId;              // Obsolete field...do not use
        struct {
            WORD wProcessorArchitecture;
```

```
            WORD wReserved;
        };
    };
    DWORD dwPageSize;
    LPVOID lpMinimumApplicationAddress;
    LPVOID lpMaximumApplicationAddress;
    DWORD dwActiveProcessorMask;
    DWORD dwNumberOfProcessors;
    DWORD dwProcessorType;
    DWORD dwAllocationGranularity;
    WORD wProcessorLevel;
    WORD wProcessorRevision;
} SYSTEM_INFO;

#define PostThreadMessage(a,b,c,d) 1
#define _CrtSetBreakAlloc(x) 1
#define _CrtDumpMemoryLeaks() 1
#define SetThreadPriority(x,y) 1
#define TerminateThread(x,y) 1
#define PeekMessage(a,b,c,d,e) 1
#define WaitMessage() 1
#define WM_DESTROY                      0x0002

typedef void *HWND;
typedef struct tagPOINT
{
```

```
    LONG    x;
    LONG    y;
} POINT;
typedef struct tagMSG {
    HWND            hwnd;
    UINT            message;
    WPARAM          wParam;
    LPARAM          lParam;
    DWORD           time;
    POINT           pt;
} MSG;

#define MAX_COMPUTERNAME_LENGTH 15
#define __int64 long long

#endif    //linux


#endif    /* portingconfig.h */
```

附录 3：如何使用 gdb 进行调试及常用命令

一 进入 gdb 调试界面：

gdb program 或 gdb program core.* (linux 默认不生成 core 文件，需要运行命令 ulimit –c unlimited)

二 常用命令：

1 在 gdb 中运行程序： run （简写为“r”），相当于 VCdebug 的 F5，可带参数，格式 r argv1 argv2 argv3 ……
例 r –licsvr 127.0.0.1:3000 –rmsvr 127.0.0.1:2000 –loglevel 4
2 有关断点：
1）设置断点：break（b），相当于 VCdebug 的 F9，格式 b filename:linenumber ;
b filename:function-name ； b line-or-function if condition ； b routine-name
例 b trec.cpp:109
b trec.cpp:TestAgent
2）删除断点：delete，格式 delete b （删除所有断点）delete b number1 number2 …… （删除指定断点）
例 delete b 2 3
3）禁用断点：disable，格式 disable b (禁用所有断点) disable b number1 number2 …… （禁用指定断点）
例 disable b 2 3
4）启用断点：enable，格式 enable b (启用所有断点) enable b number1 number2 …… （启用指定断点）
例 enable b 2 3
5）显示断点信息：info，格式 info b (显示所有断点信息) info b number1 number2 …… （显示指定断点信息）
例 info b 1
6）清除指定行上的所有断点：clear，格式 clear linenumber
例 clear 8
3 显示 N 行源代码： list，格式 list filename:linenumber
例 list trec.cpp:109 （列出 trec.cpp109 行前后 10 行，即打印 104－113 行）
list 5,25 （列出当前文件 5－25 行）
4 进入的单步调试：step（s），相当于 VCdebug 的 F11，返回到调用函数：finish ，相当于
VCdebug 的 Shift＋F11
5 不进入的单步调试：next（n），相当于 VCdebug 的 F10

6 从断点开始继续执行：continue（c）
7 结束当前循环：until （u）
8 打印变量或表达式的值：print （p）
9 设置参数：set args
10 显示参数：show args
11 显示变量类型：whatis，ptype
12 将值赋予变量：set variable = value
13 调用函数：call ，格式 call function-name
例 call printf(“abcd\n”)
14 有关信号：
1）捕获信号并处理：handle ，格式 handle 信号名或信号编号 处理动作
例 handle SIGPIPE stop print
2）发送信号：signal，格式 signal 信号名或信号编号
例 signal 2
15 显示程序中的当前位置和表示如何到达当前位置的栈跟踪：backtrace (bt) ，where
16 显示当前工作目录：pwd
17 在源文件中反向搜索正规表达式：reverse-search
18 在源文件中搜索正规表达式：search
19 在程序中设置一个监测点：watch
20 退出 gdb：quit（q）