交叉编译各平台下GDB

cross compile 交叉编译

所谓的交叉编译，就是：在一种平台上编译，编译出来的程序，是放到别的平台上运行即编译的环境，和运行的环境，不一样，属于交叉的，此所谓cross compile。交叉编译，这个概念，主要和嵌入式开发有关。

之所以要有交叉编译，主要原因是：嵌入式系统中的资源太少。具体的解释就是：交叉编译出来的程序，所要运行的目标环境中，各种资源，都相对有限，所以很难进行直接的本地编译

。最常见的情况是：在进行嵌入式开发时，目标平台，即嵌入式开发板，比如是最大主频200MHz的ARM的CPU，加上32M的RAM，加上1G的Nand Flash等等。

在如此相对比较紧张的硬件资源的前提下，在已经运行了嵌入式Linux的前提下，是没法很方便的，直接在嵌入式Linux下，去本地编译，去在ARM的CPU下，编译出来，供ARM的CPU可以运行的程序的。

因为编译，开发，都需要相对比较多的CPU，内存，硬盘等资源，而嵌入式开发上的那点资源，只够嵌入式（Linux）系统运行的，没太多剩余的资源，供你本地编译。

为什么要编译GDB呢？为了方便在嵌入式设备上直接调试、分析程序。（同tcpdump）

网上对于如何构建交叉编译环境、如何一步步编译有很多文章。但，这些文章都是零散的、琐碎的、而且是不会出现其他问题的。本文主要记录笔者在搭建交叉编译环境以及对各平台下gdb进行编译的问题和解决方法。

本文最后会将编译好的一些平台下的工具的binary文件分享出来。

搭建交叉编译的环境

其实整个交叉编译环境的搭建就是编译好了能编译出特定平台下的gcc、ar、as等工具。比如我想搭建能交叉编译mips平台下程序的环境，我最终需要用到mipsel-linux-gcc等。所以，这个过程，我们是需要先编译出对应平台的gcc等工具。

先说下我本机环境吧，系统是BackBox（基于ubuntu14.04）：

```
tracy@Tracker:~$ uname -a
Linux Tracker 3.13.0-63-generic #103-Ubuntu SMP Fri Aug 14
21:42:59 UTC 2015 x86_64 x86_64 x86_64 GNU/Linux
tracy@Tracker:~$ cat /proc/version
Linux version 3.13.0-63-generic (buildd@lgw01-18) (gcc version
4.8.2 (Ubuntu 4.8.2-19ubuntu1) ) #103-Ubuntu SMP Fri Aug 14
21:42:59 UTC 2015
```

1. 先安装需要安装的东西，包括： build-essential bison flex 之类，通过下面命令可以安装好：

```
sudo apt-get install build-essential bison flex
```

2. 交叉编译环境搭建工具我用的是buildroot。

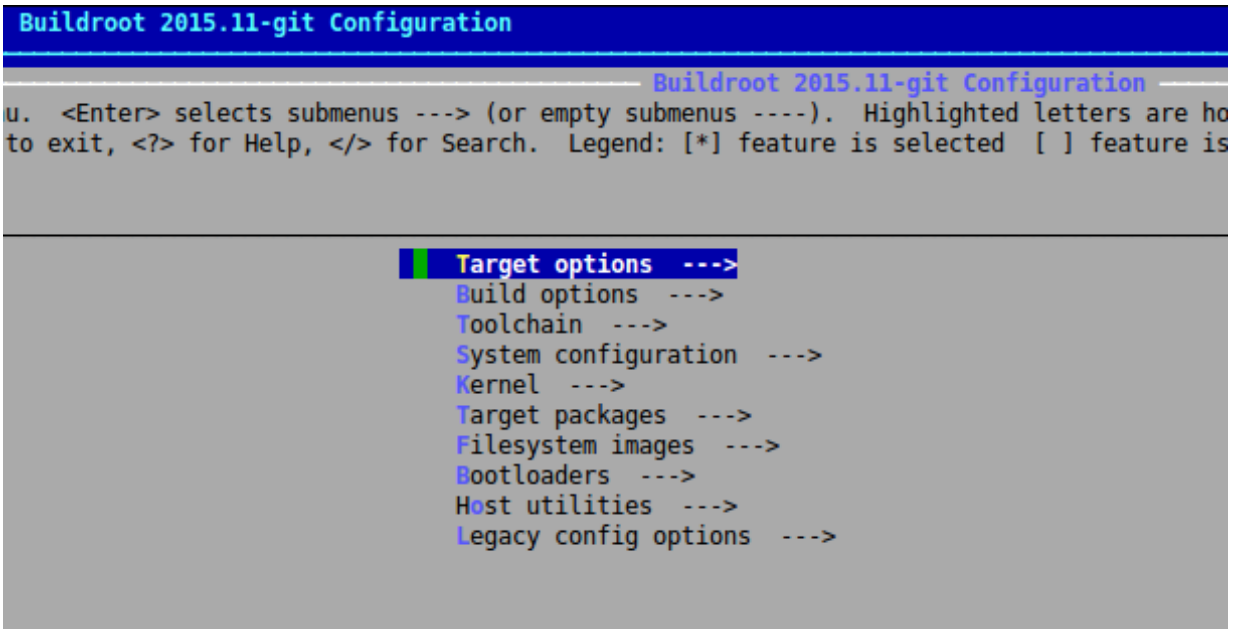
```
#下载
wget http://buildroot.uclibc.org/downloads/snapshots/buildroot-
snapshot.tar.bz2
#解压
bzip2 -d buildroot-snapshot.tar.bz2
tar -vxf buildroot-snapshot.tar
```

这样我们得到了一个buildroot的文件夹。

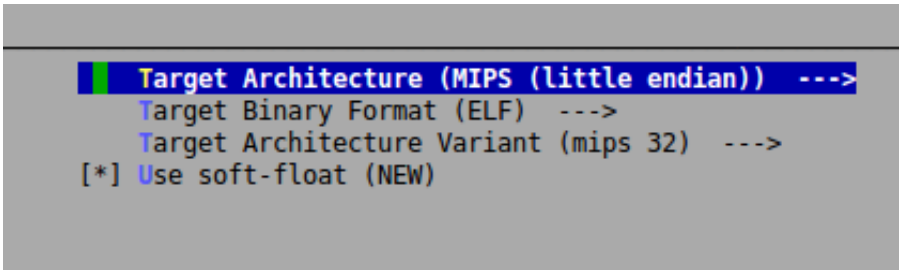
3.接下来就是设置交叉编译的环境了。

```
cd buildroot
make menuconfig
```

得到如下界面：



这里主要设置两个地方。先进去第一个选项（以mips32小端模式为例进行演示）：



> Target Architecture: 目标平台，可选项很多，这里选MIPS(little endian)。（PS：一定注意不要错，我就在这里错了次，然后～～大小端问题折腾了好久。）
> 后面的可以不要动，有兴趣也可以看看每个选项是干嘛的。

再进去第三个选项如下，这里需要注意的就是Kernel Header这里（PS：踩过坑的）：

```
    Toolchain type (Buildroot toolchain)  --->
(buildroot) custom toolchain vendor name
    *** Kernel Header Options ***
    Kernel Headers (Manually specified Linux version)  --->
(3.13) linux version
    Custom kernel headers series (3.13.x)  --->
    C library (uClibc)  --->
    *** uClibc Options ***
    uClibc C library Version (uClibc-ng)  --->
(package/uclibc/uClibc-ng.config) uClibc configuration file to use?
()  Additional uClibc configuration fragment files
[ ] Enable RPC support
[ ] Enable WCHAR support
[ ] Enable toolchain locale/i18n support
    Thread library implementation (Native POSIX Threading (NPTL))  --->
[ ] Thread library debugging
[ ] Enable stack protection support
[*] Compile and install uClibc utilities
[ ] Compile and install uClibc tests
    *** Binutils Options ***
    Binutils Version (binutils 2.24)  --->
()  Additional binutils options
    *** GCC Options ***
    GCC compiler Version (gcc 4.9.x)  --->
()  Additional gcc options
[ ] Enable C++ support
[ ] Enable Fortran support
[*] Enable compiler tls support
[ ] Enable compiler link-time-optimization support
[ ] Enable compiler OpenMP support
```

> Kernel Headers：这里，先进去找找，看看有没有对应你本机系统内核版本的。怎么看本机系统内核？翻到文章最前面去。我的版本是Linux version 3.13.0-63-generic，然而，选项里面并没有3.13之类，于是选择手动设置，然后在后面的linux version这一栏里手动填写3.13。
> Custom kernel headers series：这里，大家就去找自己对应的版本就好了。再后面，我没改动过了。同上，有兴趣可以自行测试。

设置好这两个地方后，就保存，然后退出。之后，原地make一下。然后就是漫长的等待，因为buildroot会根据你设置的选项，去下载相应的文件进行编译。大概，一个小时的样子吧，就全部结束了。

结束后，你能看到下面文件夹及内容，就代表环境搭建成功了。

```
tracy@Tracker:~$~/buildroot# ls output/host/usr/bin/ -l
-rwxr-xr-x 1 root root   24763  9月  7 00:23 faked
-rwxr-xr-x 1 root root    3955  9月  7 00:23 fakeroot
-rwxr-xr-x 1 root root   24647  9月  7 00:23 ldconfig
-rwxr-xr-x 1 root root   18908  9月  7 00:23 ldd
-rwxr-xr-x 1 root root  174085  9月  7 00:23 m4
-rwxr-xr-x 1 root root   24126  9月  7 00:23 makedevs
```

```
-rwxr-xr-x 1 root root  983926  9月  7 00:23 mipsel-buildroot-
linux-uclibc-addr2line
-rwxr-xr-x 1 root root 1017749  9月  7 00:23 mipsel-buildroot-
linux-uclibc-ar
-rwxr-xr-x 1 root root 1755935  9月  7 00:23 mipsel-buildroot-
linux-uclibc-as
lrwxrwxrwx 1 root root      33  9月  7 00:23 mipsel-buildroot-
linux-uclibc-cc -> mipsel-buildroot-linux-uclibc-gcc
-rwxr-xr-x 1 root root  978706  9月  7 00:23 mipsel-buildroot-
linux-uclibc-c++filt
-rwxr-xr-x 1 root root  888320  9月  7 00:23 mipsel-buildroot-
linux-uclibc-cpp
-rwxr-xr-x 1 root root   38638  9月  7 00:23 mipsel-buildroot-
linux-uclibc-elfedit
-rwxr-xr-x 1 root root  888276  9月  7 00:23 mipsel-buildroot-
linux-uclibc-gcc
-rwxr-xr-x 1 root root  888276  9月  7 00:23 mipsel-buildroot-
linux-uclibc-gcc-4.9.3
-rwxr-xr-x 1 root root   35115  9月  7 00:23 mipsel-buildroot-
linux-uclibc-gcc-ar
-rwxr-xr-x 1 root root   35115  9月  7 00:23 mipsel-buildroot-
linux-uclibc-gcc-nm
-rwxr-xr-x 1 root root   35119  9月  7 00:23 mipsel-buildroot-
linux-uclibc-gcc-ranlib
-rwxr-xr-x 1 root root  518819  9月  7 00:23 mipsel-buildroot-
linux-uclibc-gcov
-rwxr-xr-x 1 root root 1055900  9月  7 00:23 mipsel-buildroot-
linux-uclibc-gprof
-rwxr-xr-x 1 root root 2078471  9月  7 00:23 mipsel-buildroot-
linux-uclibc-ld
-rwxr-xr-x 1 root root 2078471  9月  7 00:23 mipsel-buildroot-
linux-uclibc-ld.bfd
lrwxrwxrwx 1 root root       8  9月  7 00:23 mipsel-buildroot-
linux-uclibc-ldconfig -> ldconfig
lrwxrwxrwx 1 root root       3  9月  7 00:23 mipsel-buildroot-
linux-uclibc-ldd -> ldd
-rwxr-xr-x 1 root root  995011  9月  7 00:23 mipsel-buildroot-
linux-uclibc-nm
-rwxr-xr-x 1 root root 1188601  9月  7 00:23 mipsel-buildroot-
linux-uclibc-objcopy
-rwxr-xr-x 1 root root 1618020  9月  7 00:23 mipsel-buildroot-
linux-uclibc-objdump
-rwxr-xr-x 1 root root 1017748  9月  7 00:23 mipsel-buildroot-
linux-uclibc-ranlib
-rwxr-xr-x 1 root root  451588  9月  7 00:23 mipsel-buildroot-
linux-uclibc-readelf
-rwxr-xr-x 1 root root  984548  9月  7 00:23 mipsel-buildroot-
linux-uclibc-size
-rwxr-xr-x 1 root root  983692  9月  7 00:23 mipsel-buildroot-
```

```
linux-uclibc-strings
-rwxr-xr-x 1 root root 1188632  9月  7 00:23 mipsel-buildroot-
linux-uclibc-strip
lrwxrwxrwx 1 root root      39  9月  7 00:23 mipsel-linux-addr2line
-> mipsel-buildroot-linux-uclibc-addr2line
lrwxrwxrwx 1 root root      32  9月  7 00:23 mipsel-linux-ar ->
mipsel-buildroot-linux-uclibc-ar
lrwxrwxrwx 1 root root      32  9月  7 00:23 mipsel-linux-as ->
mipsel-buildroot-linux-uclibc-as
lrwxrwxrwx 1 root root      32  9月  7 00:23 mipsel-linux-cc ->
mipsel-buildroot-linux-uclibc-cc
lrwxrwxrwx 1 root root      37  9月  7 00:23 mipsel-linux-c++filt -
> mipsel-buildroot-linux-uclibc-c++filt
lrwxrwxrwx 1 root root      33  9月  7 00:23 mipsel-linux-cpp ->
mipsel-buildroot-linux-uclibc-cpp
lrwxrwxrwx 1 root root      37  9月  7 00:23 mipsel-linux-elfedit -
> mipsel-buildroot-linux-uclibc-elfedit
lrwxrwxrwx 1 root root      33  9月  7 00:23 mipsel-linux-gcc ->
mipsel-buildroot-linux-uclibc-gcc
lrwxrwxrwx 1 root root      39  9月  7 00:23 mipsel-linux-gcc-4.9.3
-> mipsel-buildroot-linux-uclibc-gcc-4.9.3
lrwxrwxrwx 1 root root      36  9月  7 00:23 mipsel-linux-gcc-ar ->
mipsel-buildroot-linux-uclibc-gcc-ar
lrwxrwxrwx 1 root root      36  9月  7 00:23 mipsel-linux-gcc-nm ->
mipsel-buildroot-linux-uclibc-gcc-nm
lrwxrwxrwx 1 root root      40  9月  7 00:23 mipsel-linux-gcc-
ranlib -> mipsel-buildroot-linux-uclibc-gcc-ranlib
lrwxrwxrwx 1 root root      34  9月  7 00:23 mipsel-linux-gcov ->
mipsel-buildroot-linux-uclibc-gcov
lrwxrwxrwx 1 root root      35  9月  7 00:23 mipsel-linux-gprof ->
mipsel-buildroot-linux-uclibc-gprof
lrwxrwxrwx 1 root root      32  9月  7 00:23 mipsel-linux-ld ->
mipsel-buildroot-linux-uclibc-ld
lrwxrwxrwx 1 root root      36  9月  7 00:23 mipsel-linux-ld.bfd ->
mipsel-buildroot-linux-uclibc-ld.bfd
lrwxrwxrwx 1 root root      38  9月  7 00:23 mipsel-linux-ldconfig
-> mipsel-buildroot-linux-uclibc-ldconfig
lrwxrwxrwx 1 root root      33  9月  7 00:23 mipsel-linux-ldd ->
mipsel-buildroot-linux-uclibc-ldd
lrwxrwxrwx 1 root root      32  9月  7 00:23 mipsel-linux-nm ->
mipsel-buildroot-linux-uclibc-nm
lrwxrwxrwx 1 root root      37  9月  7 00:23 mipsel-linux-objcopy -
> mipsel-buildroot-linux-uclibc-objcopy
lrwxrwxrwx 1 root root      37  9月  7 00:23 mipsel-linux-objdump -
> mipsel-buildroot-linux-uclibc-objdump
lrwxrwxrwx 1 root root      36  9月  7 00:23 mipsel-linux-ranlib ->
mipsel-buildroot-linux-uclibc-ranlib
lrwxrwxrwx 1 root root      37  9月  7 00:23 mipsel-linux-readelf -
> mipsel-buildroot-linux-uclibc-readelf
```

```
lrwxrwxrwx 1 root root      34  9月  7 00:23 mipsel-linux-size ->
mipsel-buildroot-linux-uclibc-size
lrwxrwxrwx 1 root root      37  9月  7 00:23 mipsel-linux-strings -
> mipsel-buildroot-linux-uclibc-strings
lrwxrwxrwx 1 root root      35  9月  7 00:23 mipsel-linux-strip ->
mipsel-buildroot-linux-uclibc-strip
```

而后，问了方便下次再搭建其他环境，你可以把整个host目录copy到其他位置，再添加环境变量。如下：

```
#复制
tracy@Tracker:~$ cp output/host /opt/embedded/mipsel/ -R
#修改环境变量
tracy@Tracker:~$ echo "export
PATH=\$PATH:/opt/embedded/mipsel/usr/bin/" >> ~/.bashrc
#使修改生效
tracy@Tracker:~$ source ~/.bashrc
```

现在，在终端下输入mips按table键应该是可以直接补全了。
之后，我们来测试一下编译出来的环境是否可用。

```
#生成hello.c
tracy@Tracker:~$ echo "int main() {puts(\"Hello world\");}" >
hello.c
#编译成mips下的执行文件
tracy@Tracker:~$ mipsel-linux-gcc hello.c -o hello
```

没有报错，我们看下文件类型：

```
tracy@Tracker:~$ file hello
hello: ELF 32-bit LSB  executable, MIPS, MIPS32 version 1 (SYSV),
dynamically linked (uses shared libs), not stripped
```

Nice，这样子，我们就可以在自己电脑上编译出可在mipsel上运行的程序了。也就以为这，我们的mips交叉编译环境就搭建好了。

同理，如果想再搭建mips大端模式、mips64、arm的，记得先把output目录下的host文件夹复制到其他地方，便于下次使用。而后执行：

```
make clean
```

并再次执行：

```
make menuconfig
```

更改Target Option后再make就行了。

编译GDB

这是一个悲伤的话题，很多坑，多的来不及跳开，那就一步步来说说吧。可能，你会碰到的坑比我的还多，但无所谓了，总会解决的。

1. 下载gdb源码，我先前gnu官网下载的，后来，觉得麻烦，就直接：

```
tracy@Tracker:~$ sudo  apt-get source gdb
```

然后，自动解压出来了一个gdb-7.7.1的文件夹，下面就开始折腾吧。

```
#进入文件夹
tracy@Tracker:~$ cd gdb-7.7.1/
#配置编译选项 这编译选项是我摸索了好久的得到的可用的，顺便解释下：
#--target：代表编译出来的程序要处理的平台
#--host：代表编译出来的程序要运行的平台。这两项都填mipsel-linux是因为，configure脚本在配置的时候，默认往在`mipsel-linux`后添加`-gcc`，然后检测其存不存在。其实这里的设置，也就是为了设置使用哪个gcc。不设置的话，默认就用本机的gcc了。
#--prefix：这里配置的路径，就是在make install后，编译好的文件保存的位置
#--enable-static：允许静态链接库
#CC：其实可以不用设置，为了保险起见，还是手动设置一下，并且给他加个编译选项，这样就不会出现编译好的程序复制到mips运行提示缺少xxxx.so了。
tracy@Tracker:~/gdb-7.7.1$ ./configure --target=mipsel-linux --host=mipsel-linux --prefix=/home/tracy/mips/ --enable-static CC="mipsel-linux-gcc -static"
#编译，为了保险起见，加CC，加编译选项。
tracy@Tracker:~/gdb-7.7.1$ make CC="mipsel-linux-gcc -static"
```

然后，对，报错了。报错如下：

```
checking for iconv... no, consider installing GNU libiconv
checking for library containing waddstr... no
configure: WARNING: no enhanced curses library found; disabling
TUI
checking for library containing tgetent... no
configure: error: no termcap library found
make[1]: *** [configure-gdb] 错误 1
make[1]:正在离开目录 `/home/tracy/gdb-7.7.1'
make: *** [all] 错误 2
```

gdb编译需要termcap库，然而，我们没有，那就去下载并编译一个吧。

编译termcap

```
#回到home目录下下载termcap-1.3.1.tar.gz
tracy@Tracker:~$ wget http://ftp.gnu.org/gnu/termcap/termcap-
1.3.1.tar.gz
#解压
tracy@Tracker:~$ tar -zvxf termcap-1.3.1.tar.gz
tracy@Tracker:~$ cd termcap-1.3.1/
#配置编译选项，各选项含义如上，`-EL`是编译出小端
tracy@Tracker:~/termcap-1.3.1$ ./configure --host=mipsel-linux --
target=mipsel-linux --prefix=/home/tracy/mips/termcap/ CC="mipsel-
linux-gcc -EL -static" --disable-shared --enable-static
#编译
tracy@Tracker:~/termcap-1.3.1$ make
gcc -c  -DHAVE_STRING_H=1 -DHAVE_UNISTD_H=1 -DSTDC_HEADERS=1  -
DTERMCAP_FILE=\"/etc/termcap\" -I. -I. -g termcap.c
gcc -c  -DHAVE_STRING_H=1 -DHAVE_UNISTD_H=1 -DSTDC_HEADERS=1  -
DTERMCAP_FILE=\"/etc/termcap\" -I. -I. -g tparam.c
gcc -c  -DHAVE_STRING_H=1 -DHAVE_UNISTD_H=1 -DSTDC_HEADERS=1  -
DTERMCAP_FILE=\"/etc/termcap\" -I. -I. -g version.c
ar rc libtermcap.a termcap.o tparam.o version.o
ranlib libtermcap.a
```

看起来是对了，但是，我们file一下termcap.o。

```
tracy@Tracker:~/termcap-1.3.1$ file termcap.o
termcap.o: ELF 64-bit LSB  relocatable, x86-64, version 1 (SYSV),
not stripped
```

居然，还是x86_64的，也就意味这我们刚configure设置的host、target之类的并没有起作用。

所以，这里还是在make的时候带上CC选项。

```
#清理已编译内容
tracy@Tracker:~/termcap-1.3.1$ make clean
rm -f *.a *.o core
#编译
tracy@Tracker:~/termcap-1.3.1$ make CC="mipsel-linux-gcc -static"
mipsel-linux-gcc -static -c  -DHAVE_STRING_H=1 -DHAVE_UNISTD_H=1 -
DSTDC_HEADERS=1  -DTERMCAP_FILE=\"/etc/termcap\" -I. -I. -g
termcap.c
mipsel-linux-gcc -static -c  -DHAVE_STRING_H=1 -DHAVE_UNISTD_H=1 -
DSTDC_HEADERS=1  -DTERMCAP_FILE=\"/etc/termcap\" -I. -I. -g
tparam.c
mipsel-linux-gcc -static -c  -DHAVE_STRING_H=1 -DHAVE_UNISTD_H=1 -
DSTDC_HEADERS=1  -DTERMCAP_FILE=\"/etc/termcap\" -I. -I. -g
version.c
ar rc libtermcap.a termcap.o tparam.o version.o
ranlib libtermcap.a
#查看termcap.o类型
tracy@Tracker:~/termcap-1.3.1$ file termcap.o
termcap.o: ELF 32-bit LSB  relocatable, MIPS, MIPS32 version 1
(SYSV), not stripped
#install
tracy@Tracker:~/termcap-1.3.1$ make install
```

之后，在设置的prefix目录下会有一个lib目录，其中便有我们编译好的所需要的libtermcap.a文件。

再次编译GDB

有了libtermcap.a，该放到哪里呢？本来想写折腾经过，后来想了想，不挖坑了，直接告诉大家我怎么改的吧，其实问题应该是处在gdb目录下的configure文件中。其中有段代码如下：

```
7090
7091 case $host_os in
7092   cygwin*)
7093     if test -d $srcdir/libtermcap; then
7094       LIBS="../libtermcap/libtermcap.a $LIBS"
7095       ac_cv_search_tgetent="../libtermcap/libtermcap.a"
7096     fi ;;
7097   go32* | *djgpp*)
7098     ac_cv_search_tgetent="none required"
7099     ;;
7100   *mingw32*)
7101     ac_cv_search_tgetent="none required"
7102     CONFIG_OBS="$CONFIG_OBS windows-termcap.o"
7103     ;;
7104 esac
```

在这段的前面加上一行`ac_cv_search_tgetent="../libtermcap.a"`，改为：

```
7090 ac_cv_search_tgetent="../libtermcap.a"
7091 case $host_os in
7092   cygwin*)
7093     if test -d $srcdir/libtermcap; then
7094       LIBS="../libtermcap/libtermcap.a $LIBS"
7095       ac_cv_search_tgetent="../libtermcap/libtermcap.a"
7096     fi ;;
7097   go32* | *djgpp*)
7098     ac_cv_search_tgetent="none required"
7099     ;;
7100   *mingw32*)
7101     ac_cv_search_tgetent="none required"
7102     CONFIG_OBS="$CONFIG_OBS windows-termcap.o"
7103     ;;
7104 esac
```

再次编译：

```
tracy@Tracker:~/gdb-7.7.1$ make
tracy@Tracker:~/gdb-7.7.1$ file gdb/gdb
gdb/gdb: ELF 32-bit LSB  executable, MIPS, MIPS32 version 1
(SYSV), statically linked, not stripped
```

艾玛，居然过了，我先前编译是还会报两个地方的错的。先把刚才编译好的gdb放到mips下试试。

```
root@XiaoQiang:/tmp# ./gdb
GNU gdb (GDB) 7.7.1
Copyright (C) 2014 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later
<http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show
copying"
and "show warranty" for details.
This GDB was configured as "mipsel-linux".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word".
(gdb) quit
```

亲测可用。另外，要是想编译ARM平台下的，步骤也是一样的，先buildroot生成ARM交叉编译环境，然后gdb中configure参数设置好，并且对应的termcap也必须重新编译为arm版的。当然，ARM平台似乎种类有点多，自行探索吧。

这里也写一下报错的内容和解决办法吧。

常见报错

- 报错一：
  缺少`sgidefs.h`
  解决方法：
  下载一个吧，或者把下面内容复制，保存为`sgidefs.h`。

```
/*
 *  * This file is subject to the terms and conditions of the GNU
General Public
 *   * License.  See the file "COPYING" in the main directory of
this archive
 *    * for more details.
 *     *
 *      * Copyright (C) 1996 by Ralf Baechle
 *       *
 *        * Definitions commonly used in SGI style code.
 *         */
#ifndef __ASM_SGIDEFS_H
```

```
#define __ASM_SGIDEFS_H

/*
 *  * There are compilers out there that don't define _MIPS_ISA,
_MIPS_SIM,
 *   * _MIPS_SZINT, _MIPS_SZLONG, _MIPS_SZPTR.  So we notify the
user about this
 *    * problem.  The kernel sources are aware of this problem, so
we don't warn
 *     * when compiling the kernel.
 *      */
#if !defined(_MIPS_ISA) && !defined(__KERNEL__)
#warning "Macro _MIPS_ISA has not been defined by specs file"
#endif

#if !defined(_MIPS_SIM) && !defined(__KERNEL__)
#warning "Macro _MIPS_SIM has not been defined by specs file"
#endif

#if !defined(_MIPS_SZINT) && !defined(__KERNEL__)
#warning "Macro _MIPS_SZINT has not been defined by specs file"
#endif

#if !defined(_MIPS_SZLONG) && !defined(__KERNEL__)
#warning "Macro _MIPS_SZLONG has not been defined by specs file"
#endif

#if !defined(_MIPS_SZPTR) && !defined(__KERNEL__)
#warning "Macro _MIPS_SZPTR has not been defined by specs file"
#endif

#if (!defined(_MIPS_ISA) || \
             !defined(_MIPS_SIM) || \
             !defined(_MIPS_SZINT) || \
             !defined(_MIPS_SZLONG) || \
             !defined(_MIPS_SZPTR)) && !defined(__KERNEL__)
#warning "Please update your GCC to GCC 2.7.2-4 or newer"
#endif

/*
 *  * Now let's try our best to supply some reasonable default
values for
 *   * whatever defines GCC didn't supply.  This cannot be done
correct for
 *    * all possible combinations of options, so be careful with
your options
 *     * to GCC.  Best bet is to keep your fingers off the a.out
GCC and use
 *      * ELF GCC 2.7.2-3 where possible.
```

```
 *       */
#ifndef _MIPS_ISA
#if __mips == 1
#define _MIPS_ISA   _MIPS_ISA_MIPS1
/* It is impossible to handle the -mips2 case correct.  */
#elif __mips == 3
#define _MIPS_ISA   _MIPS_ISA_MIPS3
#elif __mips == 4
#define _MIPS_ISA   _MIPS_ISA_MIPS4
#else /* __mips must be 5 */
#define _MIPS_ISA   _MIPS_ISA_MIPS5
#endif
#endif
#ifndef _MIPS_SIM
#define _MIPS_SIM   _MIPS_SIM_ABI32
#endif
#ifndef _MIPS_SZINT
#define _MIPS_SZINT 32
#endif
#ifndef _MIPS_SZLONG
#define _MIPS_SZLONG    32
#endif
#ifndef _MIPS_SZPTR
#define _MIPS_SZPTR 32
#endif
/*
 *  * Definitions for the ISA level
 *   */
#define _MIPS_ISA_MIPS1 1
#define _MIPS_ISA_MIPS2 2
#define _MIPS_ISA_MIPS3 3
#define _MIPS_ISA_MIPS4 4
#define _MIPS_ISA_MIPS5 5
/*
 *  * Subprogram calling convention
 *   *
 *    * At the moment only _MIPS_SIM_ABI32 is in use.  This will
change rsn.
 *     * Until GCC 2.8.0 is released don't rely on this
definitions because the
 *      * 64bit code is essentially using the 32bit interface
model just with
 *       * 64bit registers.
 *        */
#define _MIPS_SIM_ABI32     1
#define _MIPS_SIM_NABI32    2
#define _MIPS_SIM_ABI64     3
#endif /* __ASM_SGIDEFS_H */
#ifndef _ABIO32
```

```
#define _ABIO32 1
#endif
#define _MIPS_SIM_ABI32 _ABIO32
#ifndef _ABIO32
#define _ABIO32 1
#endif
#define _MIPS_SIM_ABI32 _ABIO32
#ifndef _ABIN32
#define _ABIN32 2
#endif
#define _MIPS_SIM_NABI32 _ABIN32
#ifndef _ABI64
#define _ABI64 3
#endif
#define _MIPS_SIM_ABI64 _ABI64
```

- 报错二：
  linux-mips-low.c中MMLO、FPC_CSR等未定义。
  解决办法：
  将下面代码加到linux-mips-low.c最前面。

```
#define FPR_BASE    32
#define PC      64
#define CAUSE       65
#define BADVADDR    66
#define MMHI        67
#define MMLO        68
#define FPC_CSR     69
#define FPC_EIR     70
```

好吧，就写到这里吧。

Binary download

下载地址：