--build=编译该软件所使用的平台

--host=该软件将运行的平台

--target=该软件所处理的目标平台

比如"本机编译","在本机运行"，但"为embedded linux服务"的binutils package。

for example,

1. download the latest binutils, binutils-2.25.tar.gz

2. ./configure --prefix=/home/walterzh/work2/temp/bin --target=**arm-linux-gnueabi**

这里“arm-linux-gnueabi”表示生成的binutils能处理“linux” “arm” “abi” file。

3. make install

--build

--host

都省略，表示"本机编译"和生成的binutils"在本机运行"

用生成的objdump disassemble vmlinux

```
$ ./arm-linux-gnueabi-objdump -d vmlinux-3.18.7-yocto-standard
OK
```

生成的tools都是x86_64 PC Linux环境下的可执行文件，但他们处理的是ARM Linux环境下的文件。原因就是

“--target=arm-linux-gnueabi”

使用x86_64 PC上的objdump来disassemble vmlinux

```
$ objdump -d vmlinux-3.18.7-yocto-standard
vmlinux-3.18.7-yocto-standard:     file format elf32-little

objdump: can't disassemble for architecture UNKNOWN!
```

假如build的平台是x86_64 PC(一般都是)，但build生成的binutils将安装运行在ARM linux平台上，并且该binutils就是为这ARM linux平台服务的，那么

--build可以忽略，这样configure就用当前运行的平台

--host=arm-linux-gnueabi

--target=arm-linux-gnueabi

==

```
$ ./configure --prefix=/home/walterzh/work2/temp/bin --host=arm-linux-gnueabi --target=arm-linux-gnueabi
$ make
$ make install
```

看一下生成的tools的类型

```
walterzh$ file ar
ar: ELF 32-bit LSB executable, ARM, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.31,
BuildID[sha1]=0x3d772e188210ecd80045338fb792a231ec8d05b9, not stripped
```

生成的是ARM可行性文件。

```
walterzh$ ./ar
bash: ./ar: cannot execute binary file
```

在x86_64 PC Linux环境下无法运行的。但复制到ARM linux环境下应该可以运行。

原因是“--host=arm-linux-gnueabi”

==

```
$ ./configure --prefix=/home/walterzh/work2/temp/bin --host=arm-linux-gnueabi --target=x86_64-linux-gnueabi
$ make
$ make install
```

```
walterzh$ ls -l
total 65400
-rwxr-xr-x 1 walterzh walterzh 4064686 12月  2 17:29 x86_64-linux-gnueabi-addr2line
-rwxr-xr-x 2 walterzh walterzh 4201853 12月  2 17:29 x86_64-linux-gnueabi-ar
-rwxr-xr-x 2 walterzh walterzh 5864383 12月  2 17:29 x86_64-linux-gnueabi-as
-rwxr-xr-x 1 walterzh walterzh 4022452 12月  2 17:29 x86_64-linux-gnueabi-c++filt
-rwxr-xr-x 1 walterzh walterzh   73181 12月  2 17:29 x86_64-linux-gnueabi-elfedit
-rwxr-xr-x 1 walterzh walterzh 4553590 12月  2 17:29 x86_64-linux-gnueabi-gprof
-rwxr-xr-x 4 walterzh walterzh 5920485 12月  2 17:29 x86_64-linux-gnueabi-ld
-rwxr-xr-x 4 walterzh walterzh 5920485 12月  2 17:29 x86_64-linux-gnueabi-ld.bfd
-rwxr-xr-x 2 walterzh walterzh 4090171 12月  2 17:29 x86_64-linux-gnueabi-nm
```

```
-rwxr-xr-x 2 walterzh walterzh 4660669 12月  2 17:29 x86_64-linux-gnueabi-objcopy
-rwxr-xr-x 2 walterzh walterzh 5597410 12月  2 17:29 x86_64-linux-gnueabi-objdump
-rwxr-xr-x 2 walterzh walterzh 4201860 12月  2 17:29 x86_64-linux-gnueabi-ranlib
-rwxr-xr-x 1 walterzh walterzh 1007176 12月  2 17:29 x86_64-linux-gnueabi-readelf
-rwxr-xr-x 1 walterzh walterzh 4053738 12月  2 17:29 x86_64-linux-gnueabi-size
-rwxr-xr-x 1 walterzh walterzh 4051721 12月  2 17:29 x86_64-linux-gnueabi-strings
-rwxr-xr-x 2 walterzh walterzh 4660672 12月  2 17:29 x86_64-linux-gnueabi-strip
```

```
walterzh$ file x86_64-linux-gnueabi-ar
x86_64-linux-gnueabi-ar: ELF 32-bit LSB executable, ARM, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.31, BuildID[sha1]=0xa67ed3ca53eec82ba439df5ed271fa1526ec8107, not stripped
```

x86_64-linux-gnueabi-ar是ARM executable file.

复制x86_64-linux-gnueabi-readelf到embedded Linux system

```
root@granite2:~# ./x86_64-linux-gnueabi-readelf
Usage: readelf <option(s)> elf-file(s)
 Display information about the contents of ELF format files
 Options are:
  -a --all               Equivalent to: -h -l -S -s -r -d -V -A -I
  -h --file-header       Display the ELF file header
  -l --program-headers   Display the program headers
     --segments          An alias for --program-headers
  -S --section-headers   Display the sections' header
     --sections          An alias for --section-headers
  -g --section-groups    Display the section groups
  -t --section-details   Display the section details
  -e --headers           Equivalent to: -h -l -S
  -s --syms              Display the symbol table
     --symbols           An alias for --syms
  --dyn-syms             Display the dynamic symbol table
  -n --notes             Display the core notes (if present)
  -r --relocs            Display the relocations (if present)
  -u --unwind            Display the unwind info (if present)
  -d --dynamic           Display the dynamic section (if present)
  -V --version-info      Display the version sections (if present)
  -A --arch-specific     Display architecture specific information (if any)
  -c --archive-index     Display the symbol/file index in an archive
  -D --use-dynamic       Use the dynamic section info when displaying symbols
  -x --hex-dump=<number|name>
                         Dump the contents of section <number|name> as bytes
  -p --string-dump=<number|name>
                         Dump the contents of section <number|name> as strings
  -R --relocated-dump=<number|name>
                         Dump the contents of section <number|name> as relocated bytes
  -w[lLiaprmfFsoRt] or
  --debug-dump[=rawline,=decodedline,=info,=abbrev,=pubnames,=aranges,=macro,=frames,
               =frames-interp,=str,=loc,=Ranges,=pubtypes,
               =gdb_index,=trace_info,=trace_abbrev,=trace_aranges,
               =addr,=cu_index]
                         Display the contents of DWARF2 debug sections
  --dwarf-depth=N        Do not display DIEs at depth N or greater
  --dwarf-start=N        Display DIEs starting with N, at the same depth
                         or deeper
  -I --histogram         Display histogram of bucket list lengths
  -W --wide              Allow output width to exceed 80 characters
  @<file>                Read options from <file>
  -H --help              Display this information
  -v --version           Display the version number of readelf
```

下面的echo是从x86_64 PC Linux系统上copy到ARM embedded Linux系统上。

```
root@granite2:~# ./x86_64-linux-gnueabi-readelf -S echo

Elf file type is EXEC (Executable file)
Entry point 0x401688
There are 9 program headers, starting at offset 64

Program Headers:
  Type           Offset             VirtAddr           PhysAddr
                 FileSiz            MemSiz              Flags  Align
  PHDR           0x0000000000000040 0x0000000000400040 0x0000000000400040
                 0x00000000000001f8 0x00000000000001f8  R E    8
  INTERP         0x0000000000000238 0x0000000000400238 0x0000000000400238
                 0x000000000000001c 0x000000000000001c  R      1
      [Requesting program interpreter: /lib64/ld-linux-x86-64.so.2]
  LOAD           0x0000000000000000 0x0000000000400000 0x0000000000400000
                 0x0000000000005784 0x0000000000005784  R E    200000
  LOAD           0x0000000000005e20 0x0000000000605e20 0x0000000000605e20
                 0x0000000000000370 0x0000000000000520  RW     200000
  DYNAMIC        0x0000000000005e48 0x0000000000605e48 0x0000000000605e48
                 0x0000000000000190 0x0000000000000190  RW     8
  NOTE           0x0000000000000254 0x0000000000400254 0x0000000000400254
                 0x0000000000000044 0x0000000000000044  R      4
  GNU_EH_FRAME   0x0000000000004e20 0x0000000000404e20 0x0000000000404e20
                 0x00000000000001e4 0x00000000000001e4  R      4
  GNU_STACK      0x0000000000000000 0x0000000000000000 0x0000000000000000
                 0x0000000000000000 0x0000000000000000  RW     8
  GNU_RELRO      0x0000000000005e20 0x0000000000605e20 0x0000000000605e20
                 0x00000000000001e0 0x00000000000001e0  R      1

 Section to Segment mapping:
  Segment Sections...
   00
   01     .interp
   02     .interp .note.ABI-tag .note.gnu.build-id .gnu.hash .dynsym .dynstr .gnu.version .gnu.version_r .rela.dyn .rela.plt .init .plt .text .fini .rodata .eh_frame_hdr .eh_frame
   03     .ctors .dtors .jcr .dynamic .got .got.plt .data .bss
   04     .dynamic
   05     .note.ABI-tag .note.gnu.build-id
   06     .eh_frame_hdr
   07
   08     .ctors .dtors .jcr .dynamic .got
root@granite2:~#
```

```
--host=arm-linux-gnueabi
--target=arm-linux-gnueabi
```

在x86_64 Linux system上build binutils,生成的tools在ARM Linux system上安装并运行，但tools处理的确是x86_64 Linux system上的executable file。这个实验纯粹为了验证"--host"a and "--target"，实际上应该不会有这种开发场景。