

5. Заключительный этап подготовки

Содержание

- [5.1. Введение](#)
- [5.2. Binutils-2.42 - Проход 1](#)
- [5.3. GCC-13.2.0 - Проход 1](#)
- [5.4. Заголовочные файлы Linux-6.7.4 API](#)
- [5.5. Glibc-2.39](#)
- [5.6. Libstdc++ из GCC-13.2.0](#)

5.1. Введение

В этой главе дано описание, как создать кросс-компилятор и связанные с ним инструменты. Несмотря на то, что на данном этапе кросс-компиляция имитируется, принципы его работы те же, что и для настоящего кросс-тулчейна.

Программы, скомпилированные в этой главе, будут установлены в каталог **\$LFS/tools**, чтобы они были отделены от файлов, установленных в следующих главах. Библиотеки, же, устанавливаются на свое постоянное место, поскольку они относятся к системе, которую мы хотим создать.

5.2. Binutils-2.42 - Проход 1

Пакет Binutils содержит компоновщик, ассемблер и другие инструменты для работы с объектными файлами.

Приблизительное время сборки:	1 SBU
Требуемое дисковое пространство:	663 MB



Примечание

Вернитесь назад и перечитайте примечания в разделе Общие инструкции по компиляции. Понимание информации, помеченной как важная, может впоследствии избавить вас от многих проблем.

Очень важно, чтобы Binutils был скомпилированным первым, потому что и Glibc, и GCC выполняют различные тесты на доступных компоновщике и ассемблере, чтобы определить, какие из их функций следует включить.

В документации пакета Binutils рекомендуется выполнять сборку в отдельном каталоге, создадим его:

```
mkdir -v build
cd      build
```



Примечание

Для того, чтобы значения SBU, перечисленные в остальной части книги, были вам полезны, измерьте время, необходимое для сборки этого пакета, начиная с настройки и заканчивая установкой. Чтобы добиться этого, оберните команды сборки командой `time`: **`time { ../configure ... && make && make install; }`**.

Теперь подготовьте Binutils к компиляции:

```
../configure --prefix=$LFS/tools \
             --with-sysroot=$LFS \
             --target=$LFS_TGT \
             --disable-nls \
             --enable-gprofng=no \
             --disable-werror \
             --enable-default-hash-style=gnu
```

Значение параметров настройки:

- **-prefix=\$LFS/tools**

Указывает сценарию `configure` подготовить к установке пакет Binutils в каталог `$LFS/tools`.

- **-with-sysroot=\$LFS**

Для кросс-компиляции указывает системе сборки искать в `$LFS` библиотеки целевой системы, если необходимо.

- **-target=\$LFS_TGT**

Поскольку название машины в значении переменной `LFS_TGT` может отличаться от значения, которое возвращает сценарий `config.guess`, этот аргумент укажет сценарию `configure` как настроить систему сборки пакета Binutils для создания кросс-компоновщика.

- **-disable-nls**

Этот параметр отключает интернационализацию, так как `i18n` не требуется для временных инструментов.

- **-enable-gprofng=no**

Этот параметр отключает сборку `gprofng`, который не нужен для временного инструментария.

- **-disable-werror**

Этот параметр предотвращает остановку сборки в случае появления предупреждений от компилятора хоста.

- **-enable-default-hash-style=gnu**

По умолчанию компоновщик генерирует как хеш-таблицу в стиле GNU, так и классическую хеш-таблицу ELF для общих библиотек и динамически связанных исполняемых файлов. Хеш-таблицы необходимы только для динамического компоновщика, выполняющего поиск символов. В LFS динамический компоновщик (предоставляемый пакетом Glibc) всегда будет использовать хеш-таблицу в стиле GNU, к которой запросы выполняются быстрее. Так что классическая хеш-таблица ELF совершенно бесполезна. Этот параметр указывает компоновщику по умолчанию генерировать только хеш-таблицу в стиле GNU, поэтому мы можем избежать траты времени на создание классической хеш-таблицы ELF при сборке пакетов или не тратить дисковое пространство для ее хранения.

Скомпилируйте пакет:

```
make
```

Установите пакет:

```
make install
```

Подробная информация об этом пакете находится в [Разделе 8.19.2, «Содержимое пакета Binutils.»](#)

5.3. GCC-13.2.0 - Проход 1

Пакет GCC содержит коллекцию компиляторов GNU, которая включает компиляторы C и C++.	
Приблизительное время сборки:	3.8 SBU
Требуемое дисковое пространство:	4.1 GB

Примечание



В этой главе часто возникают недоразумения, хотя применяются те же процедуры, что и в любой другой главе, следуйте инструкции которую получили ранее ([Инструкции по сборке пакетов](#)). Сначала распакуйте пакет gcc-13.2.0 из архива, а затем перейдите в созданный каталог. Только после этого следует приступить к приведенным ниже инструкциям.

```
tar -xf ../mpfr-4.2.1.tar.xz
mv -v mpfr-4.2.1 mpfr
tar -xf ../gmp-6.3.0.tar.xz
mv -v gmp-6.3.0 gmp
tar -xf ../mpc-1.3.1.tar.gz
mv -v mpc-1.3.1 mpc
```

На хостах x86_64 измените имя каталога по умолчанию для 64-битных библиотек на «lib»:

```
case $(uname -m) in
  x86_64)
    sed -e '/m64=/s/lib64/lib/' \
        -i.orig gcc/config/i386/t-linux64
```

```
;;  
esac
```

В документации к GCC рекомендуется собирать GCC в отдельном каталоге:

```
mkdir -v build  
cd      build
```

Подготовьте GCC к компиляции:

```
../configure \\  
  --target=$LFS_TGT \\  
  --prefix=$LFS/tools \\  
  --with-glibc-version=2.39 \\  
  --with-sysroot=$LFS \\  
  --with-newlib \\  
  --without-headers \\  
  --enable-default-pie \\  
  --enable-default-ssp \\  
  --disable-nls \\  
  --disable-shared \\  
  --disable-multilib \\  
  --disable-threads \\  
  --disable-libatomic \\  
  --disable-libgomp \\  
  --disable-libquadmath \\  
  --disable-libssp \\  
  --disable-libvtv \\  
  --disable-libstdcxx \\  
  --enable-languages=c,c++
```

Значение параметров настройки:

- **-with-glibc-version=2.39**

Этот параметр указывает версию Glibc, которая будет использоваться на целевой системе. Он не имеет отношения к libc хост-дистрибутива, потому что все, скомпилированное в этом разделе, будет выполняться в среде chroot, которая изолирована от libc хост-дистрибутива.

- **-with-newlib**

Поскольку работающая библиотека C еще недоступна, это гарантирует, что константа `inhibit_libc` будет определена при сборке `libgcc`. Это предотвращает компиляцию любого кода, требующего поддержки libc.

- **-without-headers**

При создании полного кросс-компилятора GCC требует наличия стандартных заголовков, совместимых с целевой системой. Для наших целей эти заголовки не понадобятся. Этот параметр предотвращает их поиск GCC.

- **-enable-default-pie и -enable-default-ssp**

Эти параметры позволяют GCC по умолчанию компилировать программы с некоторыми функциями усиливающими безопасность (более подробная информация о них приведена в примечание о PIE и SSP в Главе 8). На данном этапе это не является строго обязательным, поскольку компилятор будет создавать только временные исполняемые файлы. Но лучше, чтобы временные пакеты были максимально приближены к тем, что будут в готовой системе LFS.

- **-disable-shared**

Этот параметр заставляет GCC статически связывать свои внутренние библиотеки. Он необходим потому что общие библиотеки требуют Glibc, который еще не установлен в целевой системе.

- **-disable-multilib**

На x86_64, LFS не поддерживает конфигурацию multilib. Этот аргумент никак не влияет на работу с архитектурой x86.

- **-disable-threads, -disable-libatomic, -disable-libgomp, -disable-libquadmath, -disable-libssp, -disable-libvtv, -disable-libstdcxx**

Эти аргументы отключают поддержку расширений для работы с многопоточностью, libatomic, libgomp, libquadmath, libssp, libvtv и стандартной библиотеки C++ соответственно. Эти функции могут не скомпилироваться при сборке кросс-компилятора и не нужны для задач кросс-компиляции временной libc

- **-enable-languages=c,c++**

Этот параметр обеспечивает сборку только компиляторов C и C++. Это единственные языки, которые нужны сейчас.

Скомпилируйте GCC, выполнив:

```
make
```

Установите пакет:

```
make install
```

Во время сборки GCC установил пару внутренних системных заголовочных файлов. Обычно один из файлов limits.h, включает соответствующие системные ограничения **limits.h**, в данном случае **\$LFS/usr/include/limits.h**. Однако во время сборки **GCC \$LFS/usr/include/limits.h** не существует, поэтому только что установленный внутренний заголовочный файл является частичным, автономным файлом и не включает расширенные функции системного файла. Этого достаточно для сборки Glibc, но полный внутренний заголовочный файл понадобится позже. Создайте полную версию внутреннего заголовочного файла с помощью команды, идентичной той, что система сборки GCC использует обычно:



Примечание



В приведенной ниже команде показан пример подстановки вложенных команд, используя два метода: обратные кавычки и конструкцию `$()`. Его можно было бы переписать, используя один и тот же метод для обеих замен, но сделано так, чтобы продемонстрировать, как их можно использовать одновременно. В целом метод `$()` предпочтительнее.

```
cd ..
cat gcc/limitx.h gcc/glimits.h gcc/limity.h > \
  `dirname $($LFS_TGT-gcc -print-libgcc-file-name)`/include/limits.h
```

Подробная информация об этом пакете находится в [Разделе 8.28.2. «Содержимое пакета GCC.»](#)

5.4. Заголовочные файлы Linux-6.7.4 API

From:
<http://vladpolskiy.ru/> - **book51.ru**

Permanent link:
http://vladpolskiy.ru/doku.php?id=software:linux_server:ifs:chapter05&rev=1719868298

Last update: **2024/07/02 00:11**

