4.4. Настройка окружения

Настроим хорошо работающее окружение, создав два новых файла запуска для оболочки bash. Войдя в систему как пользователь **lfs**, введите следующую команду, чтобы создать новый **.bash_profile**:

```
cat > ~/.bash_profile << "EOF"
exec env -i HOME=$HOME TERM=$TERM PS1='\u:\w\$ ' /bin/bash
EOF
```

При входе в систему под учетной записью пользователя *lfs* или при переключении на *lfs*, используя команду **su** с опцией «-», начальная оболочка представляет собой оболочку *login*, которая читает данные из `/etc/profile` хоста (который, вероятно, содержит некоторые настройки и переменные среды), а затем `.bash_profile`. Команда `exec env -i…/bin/bash` в файле `.bash_profile` заменяет запущенную оболочку новой, не содержащей переменные среды, за исключением переменных HOME, TERM, и PS1. Это гарантирует, что никакие нежелательные и потенциально опасные переменные среды из хост-системы не попадут в среду сборки.

Новый экземпляр оболочки представляет собой *non-login* оболочку, которая не считывает и не выполняет содержимое файлов `/etc/profile` и `.bash_profile`, а вместо этого выполняет чтение из файла .bashrc. Создайте файл .bashrc:

```
cat > ~/.bashrc << "EOF"
set +h
umask 022
LFS=/mnt/lfs
LC_ALL=POSIX
LFS_TGT=$(uname -m)-lfs-linux-gnu
PATH=/usr/bin
if [ ! -L /bin ]; then PATH=/bin:$PATH; fi
PATH=$LFS/tools/bin:$PATH
CONFIG_SITE=$LFS/usr/share/config.site
export LFS LC_ALL LFS_TGT PATH CONFIG_SITE
EOF
```

Значение настроек в .bashrc

- **set +h**

Команда `set +h` отключает хэш-функцию bash. Хеширование является полезной функцией –bash использует хеш-таблицу для запоминания полного пути к исполняемому файлу, чтобы избежать многократного поиска одного и того же исполняемого файла в переменной окружения PATH. Однако новые инструменты требуется использовать сразу же после их установки. Отключение хэш-функции, заставляет оболочку искать переменную окружения PATH, всякий раз, когда программу необходимо запустить. Таким образом, оболочка найдет вновь скомпилированные инструменты в `$LFS/tools/bin`, как только они станут доступны, не запоминая предыдущую версию той же программы, предоставленную хост-дистрибутивом, в `/usr/bin` или `/bin`.

- **umask 022**

Установка значения пользовательской маски создания файлов (umask) 022 гарантирует, что вновь созданные файлы и каталоги доступны для записи только их владельцу, но будут доступны для чтения и выполнения остальным пользователям (при условии, что системный вызов open(2) использует режимы по умолчанию, новые файлы получат разрешения 644, а каталоги 755).

- **LFS=/mnt/lfs**

Переменная окружения **LFS** должна указывать на выбранную точку монтирования.

- **LC_ALL=POSIX**

Переменная LC_ALL управляет локализацией определенных программ, и формирует сообщения в соответствии с локализацией указанной страны. Установка в LC_ALL значения «POSIX» или «C» (они эквивалентны) гарантирует, что все будет работать должным образом в среде кросс-компиляции.

- **LFS_TGT=$(uname -m)-lfs-linux-gnu**

Переменная LFS_TGT устанавливает нестандартное, но совместимое описание компьютера для использования при создании кросс-компилятора и компоновщика, а также при кросс-компиляции временного набора инструментов. Дополнительная информация об этом представлена в Технические примечания по сборочным инструментам.

- **PATH=/usr/bin**

Многие современные дистрибутивы Linux объединили `/bin` и `/usr/bin`. В этом случае стандартной переменной **PATH** необходимо установить значение `/usr/bin/` для окружения из Глава 6. Когда это не так, следующая строка добавит `/bin` к пути.

- **if [! -L /bin]; then PATH=/bin:$PATH; fi**

Если /bin не является символической ссылкой, то его необходимо добавить в переменную PATH.

- **PATH=$LFS/tools/bin:$PATH**

Поместив $LFS/tools/bin перед стандартным PATH, кросс-компилятор, установленный в начале Главы 5, будет обнаружен оболочкой сразу после его установки. Это, в сочетании с отключением хеширования, ограничивает риск использования компилятора хоста вместо кросс-компилятора.

- **CONFIG_SITE=$LFS/usr/share/config.site**

В Главе 5 и Главе 6, если эта переменная не задана, сценарии `configure` могут попытаться загрузить элементы конфигурации, специфичные для некоторых дистрибутивов, из `/usr/share/config.site` в хост-системе. Переопределите её, чтобы предотвратить потенциальное влияние хоста.

- **export ...**

Приведенные выше команды установили некоторые переменные, чтобы сделать их видимыми в любых вложенных оболочках, мы экспортируем их.

> 📌 **Важно**:
>
> Некоторые коммерческие дистрибутивы добавляют недокументированный экземпляр `/etc/bash.bashrc` для инициализации **bash**. Этот файл потенциально может изменить среду пользователя `lfs` таким образом, что это может повлиять на сборку важных пакетов LFS. Чтобы убедиться, что пользовательская среда `lfs` чиста, проверьте наличие файла `/etc/bash.bashrc` и, если он есть, переименуйте его. От имени пользователя root, запустите:
>
> ```
> [ ! -e /etc/bash.bashrc ] || mv -v /etc/bash.bashrc /etc/bash.bashrc.NOUSE
> ```
>
> Когда пользователь `lfs` больше не нужен (в начале Главы 7) вы можете безопасно восстановить `/etc/bash.bashrc` (по желанию).
>
> Обратите внимание, что пакет LFS Bash, который мы создадим в Разделе 8.35, «Bash-5.2.21», не настроен на загрузку или выполнение `/etc/bash.bashrc`, поэтому этот файл бесполезен в готовой системе LFS.

Для многих современных систем с несколькими процессорами (или ядрами) время компиляции пакета можно сократить, выполнив «параллельную сборку», либо установив переменную среды, либо сообщив программе make, сколько ядер задействовать для сборки. Например, процессор Intel Core i9-13900K имеет 8 ядер P (производительность) и 16 ядер E (энергоэффективность), ядро P может одновременно запускать два потока, поэтому каждое ядро P моделируется ядром Linux как два логических ядра. В результате получается 32 логических ядра. Очевидный способ задействовать все эти логические ядра - разрешить make создавать до 32 заданий сборки. Это можно сделать, передав параметр `-j32` команде make:

```
make -j32
```

Или установите переменную окружения MAKEFLAGS, и ее содержимое будет автоматически использоваться make в качестве параметров командной строки:

```
export MAKEFLAGS=-j32
```

> 📌 **Важно**:
>
> Никогда не передавайте параметр **-j** без номера в **make** и не устанавливайте такой параметр в **MAKEFLAGS**. Иначе **make** будет создавать бесконечные задания сборки, что вызовет проблемы со стабильностью системы.

Чтобы использовать все логические ядра, доступные для сборки пакетов в Главе 5 и Главе 6, укажите параметр MAKEFLAGS в .bashrc сейчас:

```
cat >> ~/.bashrc << "EOF"
export MAKEFLAGS=-j$(nproc)
EOF
```

Замените **$(nproc)** количеством логических ядер, которые вы хотите использовать, если вы планируете использовать не все логические ядра.

Наконец, чтобы убедиться, что среда полностью подготовлена для сборки временных инструментов, перечитайте только что созданный профиль пользователя:

```
source ~/.bash_profile
```

← Создание пользователя LFS O SBU (Стандартная единица времени сборки) →