

Лабораторная работа №3

Сборка и установка веб-сервера

Цель работы

Познакомиться с принципами установки в ОС Linux ПО из исходных кодов, а также с основами безопасной настройки Web-серверов.

Задания

1. Скомпилировать и установить из исходных кодов свежую версию веб-сервера **Apache**.
2. Скомпилировать и установить из исходных кодов свежую версию интерпретатора **php**.
3. Настроить с необходимым уровнем безопасности веб-сервер с применением интерпретатора **php**.
4. Проверить работоспособность созданной системы.

Теоретические сведения

Лицензирование и ПО с открытым исходным кодом

Лицензии на ПО — особый вид соглашения, ограничивающий права пользователя на получаемый им программный продукт. Распространение ПО с применением лицензий используется по той причине, что ПО, как и любой другой вид информации, поддается лёгкому копированию. Это, в свою очередь, не позволяет распространять ПО как обычный продукт, так как отсутствует механизм защиты от копирования, а это делает работу производителей ПО совершенно нерентабельным и бесполезным.

Для обычных продуктов существуют естественные механизмы защиты от копирования:

- технологическая сложность выполняемых при производстве операций
- закрытость части информации о производстве
- экономические причины: нерентабельность немассового производства
- и т.д.

При копировании информации (а значит и ПО) все эти механизмы не работают: копию информации можно сделать быстро, бесплатно, вне зависимости от того, как она создавалась и т.д.

Поэтому, для того, чтобы сделать функционирование производителей ПО обоснованным, применя-

ются юридические методы. В частности, пользователь покупает не саму копию ПО, а только право на его исп

Лицензии на ПО существуют не только для платных или проприетарных(с закрытым исходным кодом) программных продуктов, но и для ПО с открытым исходным кодом. Движение **Open Source Software** получило активное развитие в 1970-е годы в рамках развития проекта **GNU**. Название **GNU**

— это рекурсивное сокращение фразы "GNU's Not Unix!"("GNU — не Unix!"). Концепция свободной программы в соответствии с идеями проекта **GNU** определяет 4 основных правила работы с ПО с открытым исходным кодом:

1. Свобода выполнять программу, как вам угодно и в любых целях.
2. Свобода изучать, как работает программа, и адаптировать ее под свои нужды.

Для этого необходим доступ к исходному тексту.

3. Свобода распространять копии, чтобы помочь своему ближнему.

4. Свобода улучшать программу и делать ваши улучшения общедоступными к выгоде всего сообщества. Для этого необходим доступ к исходному тексту.

Лицензия, которая была создана в 1988 году для этого проекта называется **GNU GPL** (General Public License). По этой лицензии автор передаёт программное обеспечение в общественную собственность, а одна из основных идей заключённых в лицензии: **открытое закрыть нельзя!** Нельзя код, распространяемый под лицензией GPL "закрыть" и сделать проприетарным (хотя продавать его возможно).

Это правило называется **Copyleft**.

Кроме данного вида лицензии, ПО с открытым исходным также может распространяться со следующими лицензиями:

1. GPL (v1, v2, v3)
2. LGPL
3. AGPL
4. BSD — самая либеральная лицензия, не требующая даже **Copyleft**, только авторское право.
5. Apache — разработана с участием юристов IBM.
6. Artistic
7. CDDL (Common Development and Distribution License)
8. Ещё более 60 других: лицензии IBM, Apple, Eclipse Public License, Qt Public License, открытые лицензии Intel, Jabber, Zope и др.

Установка в ОС Linux ПО из исходных кодов

Традиционно установка OpenSource ПО из исходных кодов состоит из следующих этапов:

1. Получение архива исходников

- Одним из способов получения исходных кодов является скачивание файлов исходных кодов с официального сайта проекта. При этом способе можно получить самую свежую версию ПО. Это обладает как плюсами так и минусами. К недостаткам этого относят возможное наличие ошибок безопасности в свеженанписанном коде, что негативно может сказаться на безопасности системы в целом. Выгодой же является получение самой свежей версии ПО с реализацией наиболее новых и эффективных методов, алгоритмов, поддержкой новых функций и т.д.
- Другим методом является получение исходных кодов ПО из пакетов для вашего дистрибутива. Дистрибутивы Linux предоставляют возможность кроме установки бинарных (уже собранных) пакетов, скачивать/устанавливать их исходные коды. В Debian-based дистрибутивах для этого используется команда `apt-get source <имя_пакета>`. Такие исходники обычно содержат патчи безопасности и совместимости вашего дистрибутива, что повышает надёжность и безопасность собираемого ПО. Но версия исходных кодов соответствует версии бинарных пакетов устанавливаемых в дистрибутиве, что в зависимости от дистрибутива, может приводит к заметному устареванию некоторых пакетов.

2. Проверка верности скачанных файлов

Проверка скачанных файлов осуществляется с помощью контрольных сумм. Обычно на страницах проектов Open Source Software кроме ссылок на сами файлы расположены также контрольные суммы данных файлов, посчитанные одновременно с помощью различных алгоритмов.

Чтобы проверить контрольные суммы в Linux используются соответствующие утилиты:

`sha1sum`, `sha256sum` и т.д. Например,

```
user@host ~: md5sum.exe filename
```

```
317a8e45ad0597f90818d357c15d6b25 *filename
```

Кроме того, подлинность файлов можно проверить, есть для них распространяются также файлы с их подписью (файлы с расширением .asc). В таком случае для проверки используется утилита gpg:

```
user@host ~: gpg verify <file>.asc <file>
gpg: Signature made Sat May 4 19:34:08 2022 MSD using RSA key ID B115BDB6
gpg: Good signature from Alice (test key) <alice@wonderland.uk>
```

php

Downloads

Documentation

Get Involved

Help

Current Stable PHP 5.6.14 (Changelog)

- [php-5.6.14.tar.bz2 \(sig\)](#) [13,744Kb]
md5: 2e1332123a7e19d15ed2af2d1d6bd6fd
sha256: 36f295f11641c1839a5df00e693f685fd134c65e8a1d46e8ee0abae8662b2eb0
- [php-5.6.14.tar.gz \(sig\)](#) [17,892Kb]
md5: ae625e0cfcfdacea3e7a70a075e47155
sha256: 29baf77fca644f7f8e86028c40275b9e460342bdf9562d45f8f0498899cb738d
- [php-5.6.14.tar.xz \(sig\)](#) [11,304Kb]
md5: 96080ad8c5111446f58290cc6f18698c
sha256: c8edf6b05fd8a69ebd88d90c5c0975ee168502204622ad5cfd550bc222632d9
- [Windows downloads](#)

[GPG Keys for PHP 5.6](#)

Рис. 1: Пример страницы с контрольными суммами файлов

3. Распаковка архивов

По стандартам Linux исходные коды должны храниться в директории `/usr/src`. Поэтому распаковку исходных кодов рекомендуется проводить именно в данную директорию:

```
cd /usr/src
tar xvf <имя_файла_архива>.tar
```

4. Конфигурация исходных кодов

В директории с исходными кодами обычно находится скрипт `configure`, который с помощью опций командной строки позволяет настроить необходимые элементы. Эти элементы будут определять особенности функционала собираемого ПО, позволяет включить/отключить дополнительные модули, настроить особенности сборки и компиляции, а также директории, в которые будет установлено ПО. Узнать список поддерживаемых опций можно с помощью аргумента `-help`, например

```
configure help
Usage: configure [options] [host]
Options: [defaults in brackets after descriptions]
Configuration:
-cache-file=FILE cache test results in FILE
-help print this message
-no-create do not create output files
-quiet, -silent do not print "checking..." messages
-version print the version of autoconf that created configure
Directory and file names:
-prefix=PREFIX install architecture-independent files in PREFIX [/usr/local] -
exec-prefix=EPREFIX install architecture-dependent files in EPREFIX [same as
prefix] --libdir=DIR object code libraries in DIR [EPREFIX/lib]
```

`-includedir=DIR` C header files in DIR [PREFIX/include] `-oldincludedir=DIR` C header files for non-gcc in DIR [usr/include] Host type:
`-build=BUILD` configure for building on BUILD [BUILD=HOST] `-host=HOST` configure for HOST [guessed] `-target=TARGET` configure for TARGET [TARGET=HOST] Features and packages:
`-disable-FEATURE` do not include FEATURE (same as `-enable-FEATURE=no`) `-enable-FEATURE[=ARG]` include FEATURE [ARG=yes] `-with-PACKAGE[=ARG]` use PACKAGE [ARG=yes] `-without-PACKAGE` do not use PACKAGE (same as `-with-PACKAGE=no`)

`-x-includes=DIR` X include files are in DIR
`-x-libraries=DIR` X library files are in DIR
`-enable` and `-with` options recognized: `-enable-shared[=PKGS]` build shared libraries [default=yes]
`-enable-static[=PKGS]` build static libraries [default=yes]

К наиболее используемым опциям относятся `-disable/enable-FEATURE` позволяющие отключать/включать необходимые компоненты для сборки ПО, а также `-prefix`, которая определяет директорию, в которую будет устанавливаться ПО.

При сборке ПО из исходных кодов рекомендуется использовать данную опцию, т.к. при установке с префиксом по умолчанию может произойти замена файлов аналогичного ПО установленного из пакетов вашего дистрибутива Linux. Для разделения ПО установленного из пакетов и из исходных кодов рекомендуется при установке ПО из исходных кодов использовать префикс

`/opt/...` или `/opt/local/...`

```
./configure prefix=/opt/local/apache 2.6.12
```

Кроме настройки установки эта команда также производит проверку наличия всего необходимого в системе для сборки и установки. Если необходимых компонент не хватает конфигурация завершится неудачей и будет выдано соответствующее сообщение.

5. Компиляция

Компиляция выполняется командой `make`. Это, обычно, наиболее трудоёмкая для компьютера процедура и её длительность существенно зависит от производительности процессора. В процессе будут выдаваться сообщения компилятора, в основном информационного характера. В случае ошибки компиляция будет прервана с соответствующим сообщением.

6. Инсталляция

Выполняется командой `make install`, которая производит создание необходимой структуры каталогов и копирование файлов в них.

В некоторых установщиках также содержится команда `make uninstall`, позволяющая произвести деинсталляцию уже установленного ПО. Однако эта функция реализована с некоторыми ограничениями: существует не во всех программах, для использования команды необходима директория со скомпилированными исходниками, деинсталлятор не проверяет удаляемые версии и, если файлы с нужными именами уже принадлежат другой программе, то всё равно удалит их.

Иногда деинсталляция может быть реализована и другими способами, но в основном это нестандартная функция для ПО из исходных кодов. Это ещё одна причина для использования префикса при установке, т.к. позволяет произвести деинсталляцию простым удалением директории-префикса.

Веб-сервер Apache

Apache — один из наиболее используемых в мире веб-серверов (в 2011 году он был установлен на 59% серверов в мире). Относится к ПО с открытым исходным кодом и распространяется по **одно-имённой** лицензии. Основными достоинствами **Apache** считаются надёжность и гибкость конфигурации. Он позволяет подключать внешние модули для предоставления данных, использовать СУБД для аутентификации пользователей, модифицировать сообщения об ошибках и т. д.

Apache состоит из ядра и модулей. Ядро включает в себя основные функциональные возможности, такие как обработка конфигурационных файлов, протокол HTTP и система загрузки модулей. Весь остальной функционал определяется модулями.

Apache имеет встроенный механизм виртуальных хостов. Он позволяет полноценно обслуживать на одном IP-адресе множество сайтов (доменных имён), отображая для каждого из них собственное содержимое.

Для каждого виртуального хоста можно указать собственные настройки ядра и модулей, ограничить доступ ко всему сайту или отдельным файлам. Некоторые модули, например, Apache-ITK позволяют запускать процесс httpd для каждого виртуального хоста с отдельными идентификаторами uid и guid.

Одним из важнейших элементов конфигурации **Apache** является модель мультипроцессинга, т.е. как веб-сервер будет обрабатывать несколько запросов одновременно (с помощью нескольких потоков или отдельных процессов и т.д.). Существует несколько модулей реализующих разные модели мультипроцессинга (на англ. MultiProcessing Modules - MPM):

- worker — мультипоточная + мультипроцессная модель
- pre-fork — несколько предварительно запущенных процессов. Без потоков, что необходимо, на-пример, для многих расширений **php**, не являющимся потокобезопасными.
- winnt, netware — MPM для соответствующих ОС.
- Apache-ITK, perchild — MPM с гибридной моделью, позволяющие обрабатывать запросы в процессах с правами различных пользователей для различных сайтов на данном сервере. Эта функция позволяет существенно увеличить безопасность в системах с несколькими виртуальными хостами.

Основная конфигурация сервера хранится в файле `/etc/apache2/httpd.conf` или `$prefix/etc/httpd`. Кроме того при установке из пакетов остальная конфигурация разнесена на несколько файлов соответствующих модулям **Apache** и виртуальным хостам.

Интерпретатор PHP

PHP (**PHP**: **H**ypertext **P**reprocessor) — скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений. Поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов.

Язык и его интерпретатор разрабатываются в рамках проекта с открытым кодом и распространяется под собственной лицензией **PHP License**.

PHP настолько популярен благодаря своей простоте, скорости выполнения, богатой функциональности, кроссплатформенности и распространению исходных кодов на основе

лицензии PHP. К крупнейшим сайтам, использующим PHP, относятся Facebook, Wikipedia и др.

PHP-скрипты обычно обрабатываются интерпретатором в порядке, обеспечивающем кроссплатформенность разработанного приложения:

1. лексический анализ исходного кода и генерация лексем
2. синтаксический анализ полученных лексем
3. генерация байт-кода
4. выполнение байт-кода интерпретатором (без создания исполняемого файла)

Для увеличения быстродействия приложений возможно использование специального программного обеспечения, так называемых акселераторов. Принцип их работы заключается в кэшировании однажды сгенерированного байт-кода в памяти и/или на диске, таким образом, из процесса работы приложения исключаются этапы 1—3, что в общем случае ведёт к значительному ускорению работы.

Также, как и веб-сервер **Apache** интерпретатор состоит из ядра и подключаемых модулей, «расширений», представляющих собой динамические библиотеки. Расширения позволяют дополнить базовые возможности языка, предоставляя возможности для работы с базами данных, сокетами, динамической графикой, криптографическими библиотеками, документами формата PDF и тому подобным. Существует огромное количество расширений, однако в стандартную поставку входит лишь несколько десятков наиболее стабильных и используемых. Множество расширений доступно в репозитории PECL.

Сам язык, как и его интерпретатор, непосредственно не связаны ни с каким веб-сервером, и могут быть использованы совершенно независимо. Тем не менее наиболее частым применением является генерация веб-страниц. С этой точки зрения интерпретатор языка может работать в следующих режимах:

- В качестве модуля к веб-серверу

Например, для **Apache** модуль `mod_php`. В этом случае интерпретатор **PHP** выполняется в окружении процесса веб-сервера. Веб-сервер управляет количеством запущенных процессов **PHP** и сообщает им какие скрипты требуется исполнить.

- **CGI SAPI**.

Использование технологии CGI подразумевает запуск нового процесса для обработки каждого запроса. Для исполнения PHP скрипта веб-сервер запускает `./php-cgi /path/to/script.php`. При этом интерпретатор PHP исполняет только один скрипт, после чего заканчивает свою работу. Но затраты на запуск нового процесса интерпретатора и его инициализацию очень часто сопоставимы или даже превышают затраты на исполнение самого PHP скрипта. Для решения этой проблемы в CGI SAPI был введен режим FastCGI. В этом режиме PHP интерпретатор запускается как независимый сервер, обрабатывающий входящие запросы на исполнение PHP скриптов по протоколу FastCGI, что позволяет ему работать с любым веб-сервером поддерживающим этот протокол.

- В качестве скрипта командной строки (**CLI SAPI**)

В этом случае интерпретатор является исполняемым файлом, который может быть вызван пользователем из командной строки; скрипт выполняется в окружении вызвавшего пользователя. В этом случае возможно использование PHP для создания клиентских GUI-приложений.

Поддержка каждого из этих режимов может быть включена в момент компиляции.

Интерпретатор **PHP** имеет специальный конфигурационный файл — `php.ini`,

содержащий множество настроек, изменение которых влияет на поведение интерпретатора. Имеется возможность отключить использование ряда функций, изменить ограничения на используемую скриптом оперативную память, время выполнения,

объём загружаемых файлов, настроить журналирование ошибок, работу с сессиями и почтовыми сервисами, подключить дополнительные расширения, а также многое другое. Возможно дробление большого конфигурационного файла на части. Например, широко распространена практика вынесения настроек расширений в отдельные файлы.

Порядок выполнения работы

1. Установите из исходных кодов свежую версию веб-сервера **Apache**

- (a) Скачайте с официального сайта (<http://httpd.apache.org/>) проекта Apache свежую копию исходных кодов веб-сервера.
- (b) Проверьте верность скачанных файлов.
Распакуйте архив с кодами в папку, предназначенную для хранения исходных кодов ПО в Linux.(см. выше)
- (c) Перейдите в распакованную папку и выполните конфигурацию исходных кодов выбрав необходимые опции.
Увидеть список опций можно с помощью команды

```
./configure help
```

Не забудьте установить путь установки (см. выше)

- (e) Скомпилируйте сервер командой

```
make
```

- (f) Если компиляция завершилась удачно и никаких сообщений об ошибках выдано не было установите **apache** командой

```
make install
```

2. Установите из исходных кодов свежую версию интерпретатора **php**

- (a) Скачайте с официального сайта проекта **PHP** (<http://php.net/>) свежую копию исходных кодов интерпретатора.
- (b) Распакуйте архив с кодами в папку, предназначенную для хранения исходных кодов ПО в Linux.(см. выше)
- (c) Перейдите в распакованную папку и выполните конфигурацию исходных кодов выбрав необходимые опции, **аналогично конфигурации apache**.
Необходимой опцией является `-with-apxs2`, также включите поддержку графики (`-with-gd`), **mysql**, **postgres** и не забудьте установить путь установки (см. выше)
- (d) Скомпилируйте и установите **PHP** командой

```
make && make install
```

3. Добавьте в систему нового пользователя и группу **www**. Созданная учётная запись должна быть заблокирована (см. `usermod -lock`).

4. Создайте директорию `/home/www`, которая будет корнем вашего веб-сервера.

В этой папке разместите файл `index.html` с тестовым содержимым для проверки работоспособности вашего сервера.

5. Настройте **apache** в соответствии со следующими требованиями:

- (a) Запускаться под учётной записью **www**
- (b) Использовать установленный вами модуль **php** для файлов с расширением `.php`
- (c) В качестве корневой директории для веб-сервера использовать `/home/www`
- (d) В качестве индексных файлов использовать `index.php`, `index.html`

- (е) Не использовать автоиндексацию, т.е. если в директории нет индексных файлов, то показывать ошибку.
6. Выполните проверку работоспособности вашего веб-сервера:
- (а) Откройте веб-браузером тестовую страницу, созданную в предыдущих пунктах на вашем сервере
- (б) Создайте в корне веб-сервера файл `test.php` со следующим содержимым:

```
<? p h p
// Показать всю информацию, по умолчанию
INFO_ALL p h p i n f o ( ) ;
?>
```

- (с) Откройте веб-браузером созданную вами страницу `test.php`. Если настройка **php** была произведена верно, то должна отобразиться страница, пример которой показан на рисунке:
7. Не забудьте удалить файл `test.php` или перенести его за пределы корня веб-сервера, т.к. неконтролируемый доступ к данному файлу приводит к раскрытию большого количества конфигурационной информации, что является существенной угрозой безопасности веб-сервера.

Контрольные вопросы

1. Как настроить запуск веб-сервера с учётными данными конкретного пользователя?
2. Какие этапы установки ПО из исходных кодов?
3. Какая директория обычно используется для установки стороннего ПО в дистрибутивах **Linux**?
4. Для чего используется опция конфигурации исходных кодов **PHP** `-with-apxs2` ?


PHP Version 5.4.6-1ubuntu1.1	
	
System	Linux ubuntu 3.5.0-22-generic #34-Ubuntu SMP Tue Jan 8 21:41:11 UTC 2013 i686
Build Date	Nov 15 2012 01:02:52
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d
Additional .ini files parsed	/etc/php5/apache2/conf.d/10-pdo.ini, /etc/php5/apache2/conf.d/20-mysql.ini, /etc/php5/apache2/conf.d/20-mysqli.ini, /etc/php5/apache2/conf.d/20-pdo_mysqli.ini
PHP API	20100412
PHP Extension	20100525
Zend Extension	220100525
Zend Extension Build	API220100525.NTS
PHP Extension Build	API20100525.NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled

Рис. 2: Пример страницы с функцией `phpinfo`

Основная литература

1. Страницы man
 - (a) `man httpd.conf`
 - (b) `man php.ini`
 - (c) `man apache`

Дополнительная литература

1. Заяц, А.М. Администрирование информационных систем [Электронный ресурс]: учебное пособие / А.М. Заяц. — Электрон. дан. — Санкт-Петербург: СПбГЛТУ, 2011. — 140 с. — Режим доступа: <https://e.lanbook.com/book/45448>. — Загл. с экрана.

Информационно-справочные и поисковые системы

1. <http://wiki.qemu.org/Manual>
2. <https://www.debian.org/doc/>
3. <http://php.net/manual/ru/install.php>
4. <http://httpd.apache.org/docs/>