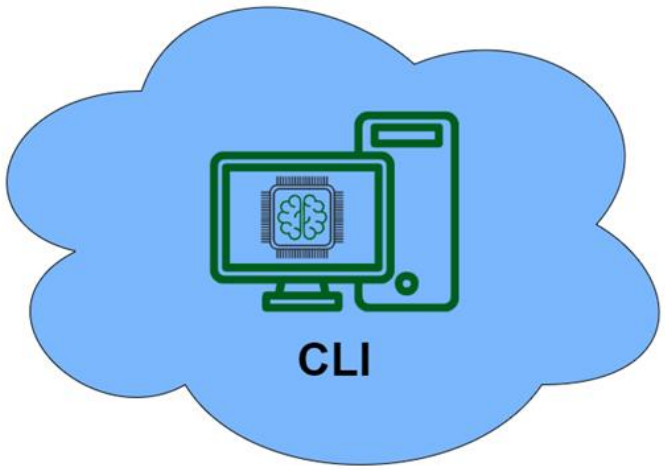
# Лабораторная работа №14.
# Конфигурация системы

**Цель работы:**

Изучение пользовательских и системных конфигурационных файлов в операционной системе ALT Linux, их структуры, назначения и различий между ними. Практическое применение знаний для настройки системы и создания пользовательских и общесистемных алиасов и скриптов.

**Оборудование, ПО:**

- CLI
- Справочная литература или доступ в сеть интернет.




| Машина | IP | Маска | Шлюз | DNS |
|---|---|---|---|---|
| CLI<br>(Альт<br>Рабочая станция 10.4) | 192.168.1.1 | 24 | - | 77.88.8.8 |

**Порядок работы:**

1. Переименовать ПК своей фамилией.
2. Вставить скриншоты результатов работы каждой команды.
3. Выполнить контрольные задания.

**Контрольные вопросы:**

1. Что такое конфигурационные файлы в операционной системе?
2. Какие конфигурационные файлы относятся к системным, а какие к пользовательским?
3. В чем разница между системными и пользовательскими алиасами?
4. Какой конфигурационный файл используется для настройки переменных окружения?
5. Как создать пользовательский алиас в ОС ALT Linux?
6. Какие файлы нужно редактировать для изменения общесистемных настроек?
7. Где находятся пользовательские конфигурационные файлы?

**Контрольные задания:**

1. Создайте алиас для команды ‘mkdir dir{1..10}’ так, чтобы он был доступен только пользователю student в системе. Докажите это, перезапустив терминал.
2. Создайте алиас для команды ‘touch file{1..10}’ так, чтобы он был доступен всем пользователям системе. Докажите это, перезапустив терминал.
3. Создайте скрипт, который будет инициализироваться при каждом запуске оболочки bash и выводить в терминал сообщение “Добрый день, коллеги”.
4. Измените “скелет” домашней директории пользователя, так, чтобы при создании новой учетной записи на рабочем столе по-умолчанию лежал ярлык веб-сайта www.gubkin.ru. Докажите это.
5. Создайте переменную окружения FIO, запишите в нее ваше ФИО. Выведите данную переменную в скрипте командой echo.
6. Объявите общесистемную переменную окружения GG, присвойте ей любое значение. Докажите что она “общесистемная” и доступна после перезапуска оболочки.
7. Измените приглашение командной строки в конфигурационных файлах так, чтобы оно было розовым и вместо имени текущего пользователя писалось ваше ФИО.

**Используемые источники:**

Официальная документация операционной системы Альт Рабочая Станция 10.4
https://docs.altlinux.org/ru-RU/alt-workstation/10.4/html/alt-workstation/index.html

Операционная система разделена на два подмножества: профиль и реализацию. Всё, что не потребует вмешательства пользователя, необходимо запрограммировать и использовать в готовом виде в качестве составных частей реализации. В Linux этому соответствуют программы и подпрограммы: ядро, модули, демоны, утилиты; используемые ими библиотеки и прочие разделяемые файлы и т. п. **Реализация** — это монолитная, неизменяемая часть системы, устроенная по типу «решений» основных задач, только задачи эти, как правило, не совпадают с задачами пользователей, а только помогают решать их, являясь как бы инструментами сборки «больших» решений.

Всё, чего может коснуться рука человека, из реализации переносится в профиль системы. **Профиль** задаёт поведение реализации на данных пользователя, и должен быть устроен так, чтобы пользователь мог его беспрепятственно изменять, если понадобится. С одной стороны, это может быть вариант «высокоуровневого программирования», когда пользователь описывает алгоритм решения и структуру используемых данных на некотором высокоуровневом языке (специализированном или общем, например, на shell). С другой стороны, задание свойств может превращаться в указание модификаторов поведения, когда пользователь перечисляет необходимые параметры работы программы, которые изменяют её заранее известную, но достаточно общую функциональность.

Таким образом система полностью описывается в виде набора необходимых компонентов реализации, запущенных с определенными профилями вместе с текущим состоянием каждого компонента. Поскольку компонент реализации не может изменяться, а его текущее состояние, наоборот, меняется постоянно и не управляется пользователем, можно считать, что систему задаёт её профиль. Это означает, что для того, чтобы продублировать работу системы на другом компьютере, достаточно установить там стандартную реализацию и перенести профиль (обычно занимающий несравненно меньше места) и пользовательское наполнение. Наполнение (файлы пользователей, содержимое веб-страниц и т. п.) может занимать много места, но оно входит в понятие «задача пользователя», поэтому забывать о нём нельзя.

**Профиль** – изменяемая часть системы, задающая её поведение во время работы.

Как проще всего создать профиль, если не всей системы, то хотя бы её компонента (программы)? Один из вариантов такой: снабдить программу функцией «сохранить настройки», тогда можно будет эту программу запустить, любым способом добиться её работоспособности, а после зафиксировать достигнутое состояние с помощью этой функции. При этом поначалу совершенно неважно, как выглядят эти настройки: программа заработала, значит, цель достигнута. Зачастую для того, чтобы собрать более или менее

отвечающий требованиям пользователя профиль, задействуется больше ресурсов, чем для работы самой программы.

Задание профиля с помощью командной строки — метод далеко не всегда удобный. Даже при работе с самой командной строкой используется **окружение** для сохранения настроек, чтобы не задавать их всякий раз и для всякой команды. А если говорить о сложных системных службах, свойства которых должны сохраняться не от сеанса к сеансу, а постоянно (в том числе при перезагрузке системы). Следовательно, профиль необходимо держать в **файле**.

**Конфигурационный файл –** текстовый файл, содержащий настройки какой-нибудь части системы (утилиты, демона и т. п.). Как правило, считывается ею при запуске. Типичный для Linux способ организации **профиля**.

Как уже было замечено, набор переменных окружения составляет особенный профиль, к которому чувствительны все запускаемые программы. Задаются переменные окружения обычно в командном сценарии, который тоже можно рассматривать как конфигурационный файл. Например, во многих дистрибутивах используется конфигурационный файл **/etc/sysconfig**

| |
|---|
| Какие файлы находятся в данной директории ? Что в них объявляется? Поясните содержимое файлов i18n и init |
| |

Если профиль слишком велик, держать его в одном конфигурационном файле не самый лучший вариант. Методов борьбы с неудобочитаемостью несколько. Например, файл можно разделить на несколько независимых друг от друга файлов так, что редактировать приходится только один из файлов, а программа во время самонастройки считывает все.

Другой способ опирается на то, что изменения, которые пользователь вносит в профиль, как правило существенно меньше объёма всего профиля. Поэтому может быть выгодно хранить все настройки по умолчанию в каком-нибудь файле, менять который вообще не надо, а файл пользовательских настроек использовать как бы «поверх», изменяя профиль в соответствии с ними после того, как выставлен профиль по умолчанию. Дополнительная

выгода такого способа — в том, что пользователь всегда может подглядеть в «большой» файл, чтобы узнать, как оформляется та или иная настройка.

Наконец, третий способ сделать конфигурационный файл удобочитаемым — это секционирование профиля, когда все настройки разбиваются на группы, каждой группе даётся собственное имя, и синтаксис конфигурационного файла проектируется так, чтобы границы групп хорошо различались при просмотре.

Как правило, конфигурационный файл считывается программой при запуске, отражая, таким образом, её состояние на момент старта. Изменения настроек работающей программы в конфигурационном файле, как правило, не отражаются. Тому есть несколько причин: не стоит превращать файл, изредка редактируемый пользователем, в файл, изменение которого происходит постоянно. Так, многие утилиты, особенно использующие графическую среду, могут записывать настройки в файл по окончании работы. Большинство конфигурационных файлов весьма удобно редактировать вручную, с помощью текстовых редакторов.

В **/etc** хранятся настройки системных служб, в том числе настройки по умолчанию пользовательских утилит, профили командных интерпретаторов, а также настройки, используемые в процессе загрузки системы. Там же располагаются и стартовые сценарии.

В **/etc** могут находиться не только файлы, но и подкаталоги и целые поддеревья каталогов. В ОС Альт используется подкаталог **/etc/sysconfig**. Этот каталог создаётся и заполняется файлами при установке системы. Некоторые стартовые сценарии, использующие полученные настройки, также лежат в этом каталоге или его подкаталогах. Если в системе есть каталог /etc/sysconfig, там должны оказаться настройки, относящиеся не к самим службам или утилитам, а к способу их запуска при загрузке, а также языковые и сетевые настройки, тип мыши и т. д.

В **/home/<user>/.*** расположены каталоги, принадлежащие пользователям системы. Отделение всех файлов, создаваемых пользователями, от прочих системных файлов даёт очевидное преимущество: серьёзное повреждение системы или необходимость обновления не затронет наиболее ценной информации — пользовательских файлов.

Из каталога **/etc/skel** файлы копируются в новый домашний каталог пользователя при создании новой учетной записи с помощью useradd или аналогичной команды. Он обычно включает шаблонные файлы конфигурации оболочки. Skel является производным от «skeleton» («скелет») и используется для запуска домашнего каталога при первом создании пользователя.

| |
|---|
| Добавьте файл в /etc/skel, в котором записано ваше ФИО. Создайте нового пользователя в системе. Покажите результат. |
| |

Оболочка bash считывает разные файлы конфигурации в зависимости от того, как запускается сеанс. Одно из различий между сеансами заключается в том, что оболочка запускается в рамках сеанса входа или без входа.

Оболочка входа — сеанс оболочки, начинающийся с аутентификации пользователя. Если вы выполняете вход в сеанс терминала или через SSH с аутентификацией, ваш сеанс оболочки будет настроен в виде оболочки со входом.

Если вы запускаете новый сеанс оболочки внутри аутентифицированного сеанса, то запускается сеанс **оболочки без входа**. При запуске вашей дочерней оболочки вам не потребовалось вводить данные для аутентификации.

Еще одним отличием, которое можно отметить, является интерактивный или неинтерактивный характер сеанса оболочки.

**Интерактивный** сеанс оболочки — это сеанс оболочки, прикрепленный к терминалу. **Неинтерактивный** сеанс оболочки не прикреплен к сеансу терминала.

Обычный сеанс, который начинается с SSH, как правило, представляет собой интерактивный сеанс входа. Скрипт, запускаемый из командной строки, обычно запускается в неинтерактивном сеансе без входа. То, относится ли сеанс оболочки к оболочке входа или без входа, определяет то, какие файлы будут считываться при инициализации сеанса оболочки.

Сеанс, запускаемый в виде сеанса входа, будет считывать данные конфигурации сначала из файла **/etc/profile.** Затем он будет искать первый файл конфигурации сеанса входа в домашнем каталоге пользователя, чтобы получить данные конфигурации для конкретного пользователя.

Сеанс считывает первый из файлов **~/.bash_profile**, который ему удается найти, и не считывает остальные файлы.

В отличие от этого сеанс, определенный в оболочке без входа, будет читать файл **/etc/bashrc** и затем использовать файл ~/.bashrc конкретного пользователя для создания окружения.

Неинтерактивные оболочки считывают значение переменной окружения BASH_ENV и указанный в нем файл для определения нового окружения.

Каждый запускаемый процесс система снабжает неким информационным пространством, которое этот процесс вправе изменять. Правила пользования этим пространством просты: в нём можно задавать именованные хранилища данных (переменные окружения), в которые записывать какую угодно информацию (присваивать значение переменной окружения), а впоследствии эту информацию считывать (подставлять значение переменной). Дочерний процесс — точная копия родительского, поэтому его окружение — также точная копия родительского. Если про дочерний процесс известно, что он использует значения некоторых переменных из числа передаваемых ему с окружением, родительский может заранее указать, каким из копируемых в окружении переменных нужно изменить значение. При этом, с одной стороны, никто (кроме системы, конечно) не сможет вмешаться в процесс передачи данных, а с другой стороны, одна и та же утилита может быть использована одним и тем же способом, но в изменённом окружении — и выдавать различные результаты.

**Области действия переменных:**

1. **Локальные переменные** доступны только в рамках текущей сессии или скрипта.

Синтаксис объявления:

$ example=’alt’

$ echo $example

| Результат работы |
| --- |
| |

2. **Переменные окружения** экспортируются и доступны для всех дочерних процессов текущей сессии.

Команда env выводит список всех переменных окружения. Кроме того, команда позволяет изменять этот список перед исполнением пользовательской команды. В некоторых командных оболочках она является встроенной, в некоторых – внешней командой.

$ env

| Результат работы |
| --- |
| |

Наиболее часто используемыми опциями являются опция –i, позволяющая игнорировать все системные переменные окружения, и опция –u, после которой должна идти имя переменной окружения, которую нужно игнорировать.

Для изменения значения произвольной переменной окружения перед запуском команды достаточно указать имя этой переменной и ее значение после символа равенства. Например, установим новое значение переменной окружения LANG на французский и выведем текущую дату и время с использованием этой локали:

$ echo $LANG

$ env LANG=fr_FR.UTF-8 date

| Результат работы |
| --- |
| |

Команда export используется в оболочках, таких как bash, для установки переменной окружения, чтобы она была доступна дочерним процессам текущей оболочки.

При запуске команды export без каких-либо параметров, будут выведены все экспортируемые переменные и функции:

$ export

| Результат работы |
| --- |
| |

Далее сделаем переменную example – переменной окружения.

$ export example

| Результат работы |
| --- |
| |

Основные опции:

1. Опция –p перечисляет имена всех переменных, используемых в текущей оболочке.

$ export –p

| Результат работы |
| --- |
| |

2. Опция –f экспортирует имена переменных в виде функций.

Пример (в терминале):

Создадим функцию:

$ function msg {

> echo “Hello, world”

> }

| Результат работы |
| --- |
| |

После экспорта вызовем функцию, используя ее имя в командной строке:

$ msg

| Результат работы |
| --- |
| |

Экспортируем функцию с помощью опции –f:

$ export –f msg

| Результат работы |
| --- |
| |

Запустим новую сессию дочерней оболочки и введем имя функции:

$ bash

$ msg

| Результат работы |
| --- |
| |

Функция работает даже после запуска дочерней оболочки, т.к. она была экспортирована ранее.

3. Опция –n удаляет указанные переменные и функции из списка экспортируемых переменных.

Еще 2 примера использования переменной export:

1. Установим vim в качестве тестового редактора по умолчанию с помощью переменной окружение EDITOR:

$ export EDITOR=/usr/bin/vim

$ export | grep EDITOR

2. Изменим цвет строки приглашения Bash по умолчанию. Для этого изменим и экспортируем значение переменной окружения PS1. Значения PS1 по умолчанию задаются в файле /etc/bashrc.

$ export PS1='\[\e[1;32m\][\u@\h \W]\$\[\e[0m\]'

| Результат работы. Что изменилось? |
| --- |
| |

Команда **set** используется для изменения значений параметров оболочки и отображения переменных в скриптах Bash.

$ set

| Результат работы |
| --- |
| |

Команда set имеет три выходных значения:

0 – успешное завершение

1 – сбой, вызванный недопустимым аргументом

2 – сбой, приводящий к сообщению об использовании, обычно из-за отсутствия аргумента.

Основные опции:

1. Опция –e осуществляет немедленный выход при завершении команды с ненулевым статусом.

2. Опция –C запрещает перезапись существующих обычных файлов.

Создадим файла с именем myfile. При попытке перезаписать его содержимое появляется ошибка:

$ echo “An existing file” > myfile

$ set –C

$ echo “Editing an existing file” > myfile

| Результат работы |
| --- |
| |

3. Опция –u рассматривает неустановленные переменные как ошибку при замене.

4. Опция –b сообщает о прекращении работы.

При отладке сценарием оболочки необходимо периодическое включение/отключение отладочной информации.

Для включение отладочной информации команда используется с параметром -x:

$ set –x

$ echo “Проверка”

| Результат работы |
| --- |
| |

Для отключения отладочной информации утилита используется с параметром +x:

$ set +x

$ echo “Проверка”

| Результат работы |
| --- |

**В чем отличие переменных окружения и переменных оболочки?**

Переменные окружения определяются для текущей оболочки и наследуются любыми дочерними оболочками или процессами. Переменные окружения необходимы для передачи информации процессам, которые запущены в оболочке.

Переменные оболочки содержаться исключительно в оболочке, внутри которой они были заданы или определены. Часто они используются для отслеживания кратковременных данных, например, текущего рабочего каталога.

Каждый сеанс оболочки отслеживает собственные переменные оболочки и окружения. Мы можем получить доступ к этим переменным разными способами.

Мы можем просмотреть список всех наших переменных окружения с помощью команд env или printenv. Разница между двумя командами становится заметна только при использовании более специфического функционала.

**РЕЗЮМЕ**

Настройка оболочки — это в первую очередь настройка окружения. В начале сеанса работы (при запуске стартового командного интерпретатора) выполняется общесистемный профиль, стартовый сценарий из файла **/etc/profile и /etc/profile.d**. Следом выполняется персональный профиль пользователя, сценарий, находящийся в домашнем каталоге, и называющийся **.profile**. Этот сценарий пользователь может видоизменять, как ему заблагорассудится.

Что касается bash, то структура его стартовых файлов сложнее. Прежде всего, ~/.profile выполняется только если в домашнем каталоге нет файла .bash_profile или .bash_login, иначе стартовый сценарий берётся оттуда. В эти файлы можно помещать команды, несовместимые с другими версиями shell, например, управление сокращениями или привязку функций к клавишам.

Кроме того, каждый интерактивный (взаимодействующий с пользователем), но не стартовый bash выполняет системный и персональный конфигурационные сценарии **/etc/bashrc, /etc/bashrc.d и ~/.bashrc**. Чтобы стартовый bash также выполнял ~/.bashrc, соответствующую команду необходимо вписать в ~/.bash_profile. Далее, каждый неинтерактивный (запущенный для выполнения сценария) bash сверяется с переменной окружения BASH_ENV и, если в этой переменной записано имя существующего файла, выполняет команды оттуда. Наконец, при завершении стартового bash выполняются команды из файла ~/.bash_logout.