

Лабораторная работа №12

Отчет

Мосолов Александр Денисович

Содержание

1 Цель работы	4
2 Задание	5
3 Выполнение лабораторной работы	6
3.1 Настройка параметров времени	6
3.2 Управление синхронизацией времени	10
3.3 Автоматизация настройки (Provisioning)	14
4 Выводы	18
5 Ответы на контрольные вопросы	19

Список иллюстраций

3.1	Проверка статуса времени на сервере	6
3.2	Проверка статуса времени на клиенте	7
3.3	Поиск и смена часового пояса на Лондон	7
3.4	Возврат часового пояса Europe/Moscow	7
3.5	Вывод команды date на сервере	8
3.6	Вывод команды date на клиенте	8
3.7	Примеры форматирования date на сервере	9
3.8	Примеры форматирования date на клиенте	9
3.9	Арифметика времени с командой date	10
3.10	Проверка аппаратных часов на сервере	10
3.11	Проверка аппаратных часов на клиенте	10
3.12	Установка пакета chrony	11
3.13	Проверка источников (служба остановлена)	11
3.14	Настройка доступа в chrony.conf	12
3.15	Перезапуск службы и открытие порта firewall	12
3.16	Настройка клиента на локальный сервер	12
3.17	Успешная синхронизация на клиенте	13
3.18	Подробная статистика chronyc tracking	14
3.19	Подготовка каталогов на сервере	14
3.20	Скрипт настройки сервера ntp.sh	15
3.21	Подготовка каталогов на клиенте	15
3.22	Скрипт настройки клиента ntp.sh	16
3.23	Provisioning для сервера в Vagrantfile	16
3.24	Provisioning для клиента в Vagrantfile	17

1 Цель работы

Получение практических навыков по управлению системным временем, часовыми поясами, а также настройке протокола сетевого времени (NTP) с использованием службы chrony в среде Linux.

2 Задание

1. Изучить и применить команды настройки времени (`timedatectl`, `date`, `hwclock`).
2. Настроить серверную виртуальную машину в качестве NTP-сервера для локальной сети.
3. Настроить клиентскую машину на синхронизацию времени с локальным сервером.
4. Автоматизировать процесс настройки через скрипты Provisioning в Vagrant.

3 Выполнение лабораторной работы

3.1 Настройка параметров времени

В начале работы я проверяю текущее состояние подсистемы времени на сервере. Команда `timedatectl` выводит детальную информацию: * **Local time**: Текущее системное время (в данном случае совпадает с UTC). * **Universal time**: Время UTC. * **RTC time**: Время в аппаратных часах (Real Time Clock). * **Time zone**: Текущий часовой пояс (UTC). * **System clock synchronized**: yes указывает, что служба NTP активна и синхронизирует время (рис. 3.1).

```
[admosolov@server.admosolov.net ~]$ timedatectl
  Local time: Thu 2025-11-20 18:10:23 UTC
  Universal time: Thu 2025-11-20 18:10:23 UTC
    RTC time: Thu 2025-11-20 18:10:23
     Time zone: UTC (UTC, +0000)
  System clock synchronized: yes
    NTP enabled: active
    RTC in local TZ: no
[admosolov@server.admosolov.net ~]$
```

Рис. 3.1: Проверка статуса времени на сервере

Аналогичную проверку провожу на клиенте. Здесь уже установлен часовой пояс Europe/Moscow (+0300). Заметно, что локальное время отличается от UTC на +3 часа, что соответствует настройке зоны. Параметр RTC in local TZ: no говорит о том, что аппаратные часы хранят время в UTC, что является стандартом для Unix-систем (рис. 3.2).

```
[admosolov@client.admosolov.net ~]$ timedatectl
    Local time: Thu 2025-11-20 21:16:03 MSK
Universal time: Thu 2025-11-20 18:16:03 UTC
RTC time: Thu 2025-11-20 18:16:04
Time zone: Europe/Moscow (MSK, +0300)
System clock synchronized: yes
NTP service: active
RTC in local TZ: no
[admosolov@client.admosolov.net ~]$
```

Рис. 3.2: Проверка статуса времени на клиенте

Для изменения часового пояса сначала выполняю поиск доступных зон с помощью `list-timezones` и фильтра `grep`. Найдя зону `Europe/London`, применяю её командой `set-timezone`. Это действие изменяет смещение локального времени относительно UTC, но не меняет само значение UTC (рис. 3.3).

```
[admosolov@server.admosolov.net ~]$ timedatectl list-timezones | grep -i london
Europe/London
[admosolov@server.admosolov.net ~]$ sudo timedatectl set-timezone Europe/London
```

Рис. 3.3: Поиск и смена часового пояса на Лондон

Возвращаю корректный для работы часовой пояс `Europe/Moscow`. После выполнения команды вывод `timedatectl` подтверждает, что зона изменена на MSK (+0300), а локальное время пересчитано соответствующим образом (рис. 3.4).

```
[admosolov@server.admosolov.net ~]$ sudo timedatectl set-timezone Europe/Moscow
[admosolov@server.admosolov.net ~]$ timedatectl
    Local time: Thu 2025-11-20 18:13:23 MSK
Universal time: Thu 2025-11-20 18:13:23 UTC
    RTC time: Thu 2025-11-20 18:13:23
Time zone: Europe/Moscow (MSK, +0300)
System clock synchronized: yes
NTP service: active
RTC in local TZ: no
```

Рис. 3.4: Возврат часового пояса `Europe/Moscow`

Команда `date` без аргументов выводит текущее системное время с учетом установленной временной зоны. На сервере это выглядит следующим образом (рис. 3.5).

```
[admosolov@server.admosolov.net ~]$ date  
Thu Nov 20 09:16:24 PM MSK 2025  
[admosolov@server.admosolov.net ~]$
```

Рис. 3.5: Вывод команды date на сервере

На клиенте выполняю ту же команду. Сравнение показывает, что время на обеих машинах синхронизировано (с точностью до секунд, учитывая разницу во времени ввода команд) (рис. 3.6).

```
[admosolov@client.admosolov.net ~]$ date  
Thu Nov 20 09:16:26 PM MSK 2025  
[admosolov@client.admosolov.net ~]$
```

Рис. 3.6: Вывод команды date на клиенте

Команда date обладает широкими возможностями форматирования. Я протестировал следующие ключи форматирования на сервере: * +%Y-%m-%d: вывод только даты. * +%H:%M:%S.%3N: вывод времени с точностью до миллисекунд. * -R: вывод в формате RFC 2822 (используется в заголовках email). * -u: вывод времени UTC независимо от локальной настройки. * Переопределение переменной окружения TZ позволяет вывести время в другой зоне без изменения системных настроек (рис. 3.7).

```
[admosolov@server.admosolov.net ~]$ date +%Y-%m-%d  
2025-11-20  
[admosolov@server.admosolov.net ~]$ date +%H  
21  
[admosolov@server.admosolov.net ~]$ date +%H:%M:%S  
21:18:22  
[admosolov@server.admosolov.net ~]$ date +%H:%M:%S.%3N  
21:18:55.175  
[admosolov@server.admosolov.net ~]$ date -R  
Thu, 20 Nov 2025 21:19:11 +0300  
[admosolov@server.admosolov.net ~]$ date +%A  
Thursday  
[admosolov@server.admosolov.net ~]$ date -u  
Thu Nov 20 06:20:13 PM UTC 2025  
[admosolov@server.admosolov.net ~]$ TZ='Europe/London' date  
Thu Nov 20 06:20:42 PM GMT 2025  
[admosolov@server.admosolov.net ~]$
```

Рис. 3.7: Примеры форматирования date на сервере

Те же операции форматирования были проверены на клиентской машине для подтверждения идентичности функционала (рис. 3.8).

```
[admosolov@client.admosolov.net ~]$ date +%Y-%m-%d  
2025-11-20  
[admosolov@client.admosolov.net ~]$ date +%H:%M:%S  
21:18:30  
[admosolov@client.admosolov.net ~]$ date +%H:%M:%S.%3N  
21:19:27.807  
[admosolov@client.admosolov.net ~]$ date -R  
Thu, 20 Nov 2025 21:19:33 +0300  
[admosolov@client.admosolov.net ~]$ date +%A  
Thursday  
[admosolov@client.admosolov.net ~]$ date -u  
Thu Nov 20 06:20:21 PM UTC 2025  
[admosolov@client.admosolov.net ~]$
```

Рис. 3.8: Примеры форматирования date на клиенте

Команда date также позволяет выполнять арифметические операции с датами, что полезно при написании скриптов (например, для ротации логов). Я использовал флаг `--date` для вывода даты следующего дня, даты через неделю или неделю назад (рис. 3.9).

```
[admosolov@server.admosolov.net ~]$ date --date "1 day"
Fri Nov 21 09:21:15 PM MSK 2025
[admosolov@server.admosolov.net ~]$ date --date "2 days"
Sat Nov 22 09:21:27 PM MSK 2025
[admosolov@server.admosolov.net ~]$ date --date "1 week"
Thu Nov 27 09:21:40 PM MSK 2025
[admosolov@server.admosolov.net ~]$ date --date "1 week ago"
Thu Nov 13 09:21:46 PM MSK 2025
[admosolov@server.admosolov.net ~]$ date --date "tomorrow"
Fri Nov 21 09:21:59 PM MSK 2025
[admosolov@server.admosolov.net ~]$ date --date "tomorrow"
Fri Nov 21 09:22:06 PM MSK 2025
[admosolov@server.admosolov.net ~]$
```

Рис. 3.9: Арифметика времени с командой date

Утилита `hwclock` управляет аппаратными часами (RTC), которые работают от батарейки на материнской плате. Команда показывает время, хранящееся в чипе. Вывод подтверждает, что аппаратные часы идут по UTC, а ядро Linux при загрузке считывает это время и применяет смещение часового пояса (рис. 3.10).

```
[root@server.admosolov.net ~]# hwclock
2025-11-20 18:31:35.896897+00:00
```

Рис. 3.10: Проверка аппаратных часов на сервере

Проверка `hwclock` на клиенте показывает аналогичный результат. Разница между системным временем (`date`) и аппаратным (`hwclock`) обусловлена программной коррекцией дрейфа часов ядром ОС (рис. 3.11).

```
[root@client.admosolov.net ~]# hwclock
2025-11-20 18:32:50.349917+00:00
[root@client.admosolov.net ~]#
```

Рис. 3.11: Проверка аппаратных часов на клиенте

3.2 Управление синхронизацией времени

Для организации NTP-сервера использую пакет `chrony`. На сервере проверяю его наличие через пакетный менеджер `dnf`. Пакет уже установлен в системе (рис. 3.12).

```
[root@server.admosolov.net ~]# dnf -y install chrony
Extra Packages for Enterprise Linux 10 - x86_64
Extra Packages for Enterprise Linux 10 - x86_64
Rocky Linux 10 - BaseOS
Rocky Linux 10 - AppStream
Rocky Linux 10 - CRB
Rocky Linux 10 - Extras
Package chrony-4.6.1-1.el10.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
```

Рис. 3.12: Установка пакета chrony

До начала настройки я проверяю текущие источники времени командой `chronyc sources`. **Пояснение вывода:** * В выводе присутствуют внешние сервера (например, `dot.kkursor.ru`), но статус соединения показывает ошибку.

* Сообщение `506 Cannot talk to daemon` внизу означает, что служба `chronyd` в данный момент остановлена или недоступна, поэтому клиент `chronyc` не может получить от нее данные (рис. 3.13).

```
[root@server.admosolov.net ~]# chronyc sources
MS Name/IP address      Stratum Poll Reach LastRx Last sample
=====
** dot.kkursor.ru        2    7   377    85   -212us[ -225us] +/-   23ms
^+ 39.178.111.109.sta.211.ru  1    8   377    18   -1618us[-1618us] +/-   29ms
^C^+ 51.250.110.169       3    7   377    86   -1106us[-1119us] +/-   34ms
506 Cannot talk to daemon
[root@server.admosolov.net ~]# qu...
```

Рис. 3.13: Проверка источников (служба остановлена)

Для того чтобы сервер начал отдавать время клиентам локальной сети, редактирую конфигурационный файл `/etc/chrony.conf`. Директива `allow 192.168.0.0/16` разрешает входящие NTP-запросы от всех узлов подсети `192.168.x.x`. Без этой строки сервер будет работать только как клиент внешних источников (рис. 3.14).

```
# Increase the minimum number of selectable sources
# the system clock.
#minsources 2

# Allow NTP client access from local network.
allow 192.168.0.0/16

# Serve time even if not synchronized to a time source
#local stratum 10
```

Рис. 3.14: Настройка доступа в chrony.conf

Чтобы применить настройки, перезапускаю службу chronyd. Также необходимо открыть порт 123 (UDP) в брандмауэре, так как протокол NTP использует именно его. Команда --permanent сохраняет правило после перезагрузки (рис. 3.15).

```
[root@server.admosolov.net ~]# systemctl restart chronyd
[root@server.admosolov.net ~]# firewall-cmd --add-service=ntp --permanent
success
[root@server.admosolov.net ~]# firewall-cmd --reload
success
[root@server.admosolov.net ~]#
```

Рис. 3.15: Перезапуск службы и открытие порта firewall

Перехожу к настройке клиента. В файле /etc/chrony.conf указываю адрес моего сервера: server server.admosolov.net iburst. **Пояснение:*** Опция iburst (Initial Burst) критически важна: она заставляет клиент отправить пачку из 8 пакетов при запуске службы, что позволяет синхронизировать время за несколько секунд, вместо стандартного ожидания в несколько минут (рис. 3.16).

```
# Specify directory for log files.
logdir /var/log/chrony

# Select which information is logged.
#log measurements statistics tracking

server server.admosolov.net iburst
^G Help          ^O Write Out      ^F Where Is
^X Exit          ^R Read File       ^\ Replace
```

Рис. 3.16: Настройка клиента на локальный сервер

После перезапуска службы на клиенте снова выполняю `chronyc sources`.

Пояснение вывода: * M (Mode): ^ означает сервер. * S (State): * означает, что данный источник (`server.admosolov.net`) выбран как текущий эталон времени. * Stratum: 3. Это означает, что мой сервер имеет страту 2 (берет время от внешних источников страты 1), а клиент, соответственно, получает страту 3. * Reach: 37 (в восьмеричной системе) показывает успешность последних попыток опроса. * **Вывод:** Синхронизация с локальным сервером работает корректно (рис. 3.17).

```
[root@client.admosolov.net ~]# systemctl restart chronyd
[root@client.admosolov.net ~]# chronyc sources
MS Name/IP address      Stratum Poll Reach LastRx Last sample
=====
^? admosolov.net          3   6    3    1 +2542us[+2542us] +/- 9646us
[root@client.admosolov.net ~]# chronyc sources
MS Name/IP address      Stratum Poll Reach LastRx Last sample
=====
^* server.admosolov.net    3   6   37   10 -196us[-458us] +/- 11ms
[root@client.admosolov.net ~]#
```

Рис. 3.17: Успешная синхронизация на клиенте

Для более глубокого анализа использую команду `chronyc tracking`. **Пояснение вывода:** * **Reference ID:** Идентификатор сервера (`server.admosolov.net`), с которым идет синхронизация. * **Stratum:** 4 (в данный момент, возможно сервер переключился на источник страты 3). Чем меньше число, тем ближе к эталонным атомным часам. * **System time:** Отклонение системных часов от NTP-времени (0.0011 секунды медленнее). * **Last offset:** Смещение, вычисленное при последнем опросе (-0.0009 секунды). * **Root delay/dispersion:** Накопленные задержки и погрешности от корня (stratum 1). Эти данные подтверждают, что часы клиента жестко привязаны к серверу с высокой точностью (рис. 3.18).

```
[root@client.admosolov.net ~]# chronyc tracking
Reference ID      : C0A80164 (server.admosolov.net)
Stratum          : 4
Ref time (UTC)   : Thu Nov 20 18:47:08 2025
System time      : 0.001113357 seconds slow of NTP time
Last offset      : -0.000935379 seconds
RMS offset       : 0.000442909 seconds
Frequency        : 52.080 ppm slow
Residual freq    : -2.013 ppm
Skew              : 20.925 ppm
Root delay       : 0.014300395 seconds
Root dispersion  : 0.002505351 seconds
Update interval  : 65.0 seconds
Leap status       : Normal
[root@client.admosolov.net ~]#
```

Рис. 3.18: Подробная статистика chronyc tracking

3.3 Автоматизация настройки (Provisioning)

Чтобы настройки не пропали при пересоздании виртуальных машин и для соблюдения практики “Инфраструктура как код”, я создаю скрипты настройки. На сервере подготавливаю структуру каталогов для хранения эталонного конфига (рис. 3.19).

```
[root@server.admosolov.net ~]# cd /vagrant/provision/server
[root@server.admosolov.net server]# mkdir -p /vagrant/provision/server/ntp/etc
[root@server.admosolov.net server]# cp -R /etc/chrony.conf /vagrant/provision/server/ntp/etc/
[root@server.admosolov.net server]# cd /vagrant/provision/server
[root@server.admosolov.net server]# touch ntp.sh
[root@server.admosolov.net server]# chmod +x ntp.sh
[root@server.admosolov.net server]#
```

Рис. 3.19: Подготовка каталогов на сервере

Создаю скрипт `ntp.sh` для сервера. Он выполняет: установку `chrony`, копирование заранее подготовленного конфига в `/etc`, восстановление контекста SELinux (`restorecon`), настройку `firewall` и перезапуск службы (рис. 3.20).

```
root@server:/vagrant/provision/server
GNU nano 8.1
#!/bin/bash
echo "Provisioning script $0"

echo "Install needed packages"
dnf -y install chrony

echo "Copy configuration files"
cp -R /vagrant/provision/server/ntp/etc/* /etc

restorecon -vR /etc

echo "Configure firewall"
firewall-cmd --add-service=ntp
firewall-cmd --add-service=ntp --permanent

echo "Restart chronyd service"
systemctl restart chronyd
```

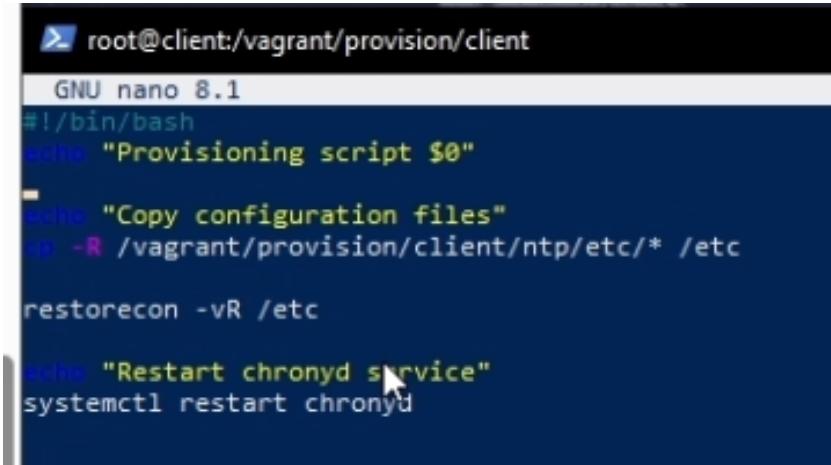
Рис. 3.20: Скрипт настройки сервера ntp.sh

Аналогичные действия провожу на клиенте. Создаю каталог для хранения клиентской версии chrony.conf (рис. 3.21).

```
[root@client.admosolov.net ~]# cd /vagrant/provision/client
[root@client.admosolov.net client]# mkdir -p /vagrant/provision/client/ntp/etc
[root@client.admosolov.net client]# cp -R /etc/chrony.conf /vagrant/provision/client/ntp/etc/
[root@client.admosolov.net client]# cd /vagrant/provision/client
[root@client.admosolov.net client]# touch ntp.sh
[root@client.admosolov.net client]# chmod +x ntp.sh
[root@client.admosolov.net client]#
```

Рис. 3.21: Подготовка каталогов на клиенте

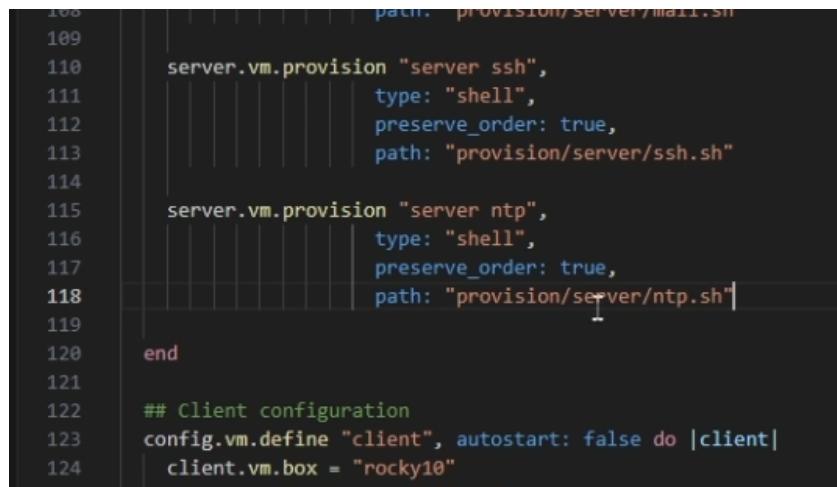
Скрипт клиента ntp.sh проще: он копирует конфигурационный файл (где прописан локальный сервер) и перезапускает службу (рис. 3.22).



```
root@client:/vagrant/provision/client
GNU nano 8.1
#!/bin/bash
echo "Provisioning script $0"
-
echo "Copy configuration files"
cp -R /vagrant/provision/client/ntp/etc/* /etc
restorecon -vR /etc
echo "Restart chronyd service"
systemctl restart chronyd
```

Рис. 3.22: Скрипт настройки клиента ntp.sh

В завершение, подключаю эти скрипты в Vagrantfile. Для сервера добавляю блок `vm.provision` типа `shell`, указывая путь к скрипту. Это гарантирует, что при команде `vagrant up` сервер будет настроен автоматически (рис. 3.23).



```
108      path: "provision/server/main.sh"
109
110    server.vm.provision "server ssh",
111      type: "shell",
112      preserve_order: true,
113      path: "provision/server/ssh.sh"
114
115    server.vm.provision "server ntp",
116      type: "shell",
117      preserve_order: true,
118      path: "provision/server/ntp.sh"
119
120  end
121
122  ## Client configuration
123 config.vm.define "client", autostart: false do |client|
124   client.vm.box = "rocky10"
```

Рис. 3.23: Provisioning для сервера в Vagrantfile

Аналогичную запись добавляю для клиента. Теперь вся лабораторная работа полностью автоматизирована и воспроизводима (рис. 3.24).

```
160      preserve_order: true,
161      path: "provision/client/mail.sh"
162
163      client.vm.provision "client ntp",
164          type: "shell",
165          preserve_order: true,
166          path: "provision/client/ntp.sh"
167    end
168  end
169
```

Рис. 3.24: Provisioning для клиента в Vagrantfile

4 Выводы

В ходе лабораторной работы я научился: 1. Диагностировать и изменять настройки системного времени и часовых поясов в Linux (`timedatectl`). 2. Использовать команду `date` для отображения, форматирования и вычисления дат. 3. Понимать разницу между системным временем и аппаратными часами (`hwclock`). 4. Настраивать службу `chrony`: * Конфигурировать сервер для обслуживания локальной сети (директива `allow`). * Настраивать клиент для синхронизации с конкретным сервером и использования режима `iburst`. * Интерпретировать вывод команд диагностики `chronyc sources` и `chronyc tracking`. 5. Настраивать правила межсетевого экрана (`firewall-cmd`) для пропуска NTP-трафика. 6. Писать Bash-скрипты для автоматической настройки сервисов и интегрировать их в Vagrant.

5 Ответы на контрольные вопросы

- Почему важна точная синхронизация времени для служб баз данных?** Для обеспечения целостности транзакций (ACID), корректной работы механизмов репликации (определение последней версии данных), правильного восстановления из журналов (logs) и временных меток (timestamps) при записи данных. Рассинхронизация может привести к потере данных или нарушению логики работы распределенных БД.
- Почему служба проверки подлинности Kerberos сильно зависит от правильной синхронизации времени?** Протокол Kerberos использует метки времени в выдаваемых билетах (tickets) для предотвращения “атак повторного воспроизведения” (replay attacks). Если время на клиенте и KDC (Key Distribution Center) различается больше, чем на допустимое значение (обычно 5 минут), билет считается недействительным, и аутентификация отклоняется.
- Какая служба используется по умолчанию для синхронизации времени на RHEL 7?** Служба chrony (демон Chrony). Она заменила устаревший ntpd в качестве решения по умолчанию.
- Какова страта по умолчанию для локальных часов?** Если сервер не синхронизирован с внешним источником, его локальным часам обычно присваивается высокая страта (например, 10), чтобы клиенты предпочитали другие, более точные источники, если они доступны. В конфиге это задается директивой local stratum 10.

- 5. Какой порт брандмауэра должен быть открыт, если вы настраиваете свой сервер как одноранговый узел NTP? Порт 123 протокола UDP.** NTP использует этот порт как для отправки, так и для получения запросов времени.
- 6. Какую строку вам нужно включить в конфигурационный файл chrony, если вы хотите быть сервером времени, даже если внешние серверы NTP недоступны?** `local stratum 10`. Эта команда предписывает Chrony продолжать обслуживать клиентов, используя локальные системные часы как эталон, даже если связь с вышестоящими NTP-серверами потеряна (“orphaned mode”).
- 7. Какую страту имеет хост, если нет текущей синхронизации времени NTP?** Формально, если синхронизация отсутствует, страта не определена или считается равной 16 (бесконечность/недостижимость в терминологии NTP), что означает, что время данного хоста недостоверно.
- 8. Какую команду вы бы использовали на сервере с chrony, чтобы узнать, с какими серверами он синхронизируется?** `chronyc sources` (для краткого списка) или `chronyc sources -v` (для списка с пояснениями заголовков столбцов).
- 9. Как вы можете получить подробную статистику текущих настроек времени для процесса chrony вашего сервера?** `chronyc tracking` – эта команда выводит системную информацию о процессе синхронизации: идентификатор эталона, страту, текущее смещение, задержку, частоту коррекции и статус “Leap status”.