# Содержание

# Глава 7: Настройка системы: журналирование, системное время, пакетные задачи и пользователи

## 7.1 Ведение системного журнала

Системные программы записывают логи через **syslogd** (традиционный) или **journald** (современный, поставляется с systemd). Сообщение содержит: имя процесса, PID, метку времени, facility и severity.

| Компонент | Назначение |
|---|---|
| syslogd | Традиционная служба логирования |
| journald | Современная служба (systemd) |
| /var/log | Директория с логами |
| /var/log/journal | Двоичные логи journald |

### 7.1.1 Проверка настроек журнала

Определить тип логирования в системе:

- **journald**: команда `journalctl` (работает, если journald активна)
- **rsyslogd**: процесс в списке или `/etc/rsyslog.conf`
- **syslog-ng**: директория `/etc/syslog-ng`

Дополнительные логи: `/var/log/wtmp`, `/var/log/lastlog` (утилиты `last`, `lastlog`)

### 7.1.2 Поиск и мониторинг журналов

#### Фильтрация по времени

```
journalctl -S -4h               # последние 4 часа
journalctl -S 06:00:00          # с 6:00 сегодня
journalctl -S 2020-01-14        # с определённой даты
journalctl -S '2020-01-14 14:30:00'
journalctl -U <время>           # до указанного времени
```

#### Фильтрация по юнитам

```
journalctl -u cron.service      # логи конкретного сервиса
journalctl -F _SYSTEMD_UNIT     # все юниты в журнале
```

#### Поиск полей

```
journalctl -N                   # список всех полей
journalctl _PID=8792            # поиск по PID
```

**Примечание**: полный доступ требует прав суперпользователя или членства в группах `adm` или `systemd-journal`.

#### Фильтрация по тексту

```
journalctl -g 'kernel.*memory'  # поиск по regex
```

Часто важная информация находится рядом по времени - используйте метку времени совпадений и `journalctl -S`.

**Примечание**: параметр `-g` требует специальной библиотеки (есть не во всех дистрибутивах).

#### Фильтрация по загрузке системы

```
journalctl -b               # текущая загрузка
journalctl -b -1            # предыдущая загрузка
journalctl --list-boots     # список всех загрузок с ID и временем
journalctl -k               # только сообщения ядра
journalctl -r -b -1         # проверка корректного выключения
```

#### Фильтрация по важности

```
journalctl -p 3             # от уровня 0 (важное) до 3
journalctl -p 2..3          # уровни 2 и 3
```

Уровни: 0 (emergency) → 7 (debug). Фильтрация редко бывает полезной.

#### Простой мониторинг журнала

```
journalctl -f               # следить за новыми сообщениями (как tail -f)
```

Можно комбинировать с другими параметрами фильтрации (`-u`, `-S`, `-p` и т.д.).

### 7.1.3 Ротация файлов журнала

**Логротация** - удаление старых сообщений с помощью **logrotate**. Вместо удаления из файла старые логи разделяются на блоки:

- `auth.log` (новые)
- `auth.log.1`, `auth.log.2`, `auth.log.3` (старые)

Процесс ротации:

```
1. Удалить `auth.log.3`
2. `auth.log.2` → `auth.log.3`
3. `auth.log.1` → `auth.log.2`
4. `auth.log` → `auth.log.1`
```

Варианты в разных дистрибутивах:

- **Ubuntu**: сжатие (`.gz`) при переносе с позиции 1 на 2
- Другие: суффикс даты (`-20200529`)

**Конфликты ротации**: если logrotate ротирует файл, пока демон его пишет - сообщение запишется в переименованный файл благодаря тому, как Linux работает с открытыми файлами.

## 7.1.4 Обслуживание журналов journald

Логи в `/var/log/journal` **не требуют ротации** - journald самостоятельно удаляет старые сообщения.

Критерии удаления:

- Свободное место на ФС
- Процент от размера ФС
- Максимальный размер журнала
- Максимальный возраст сообщения

Настройка: страница `journald.conf(5)`.

## 7.1.5 Детали системного журналирования

### История и архитектура

**syslog** развился в 1980-х годах с sendmail (RFC 3164, затем RFC 5424). Классическая архитектура **«клиент - сервер»**.

**Механизм**: syslogd слушает Unix-сокет `/dev/log` и сетевые сокеты. Позволяет централизовать логи всей сети на одном сервере.

### Объекты и важность

| **Параметр** | **Назначение** |
|---|---|
| **Facility** (объект) | Категория службы (ядро, почта, принтер) |
| **Severity** (важность) | Уровни 0–7 (emerg до debug) |

| Параметр | Назначение |
|---|---|
| **Priority** (приоритет) | Комбинация facility + severity |

Уровни: **0** (emerg) / **1** (alert) / **2** (crit) / **3** (err) / **4** (warning) / **5** (notice) / **6** (info) / **7** (debug)

Ограничения: объекты жёстко определены (нет новых), но есть слоты `local0–local7` для пользовательских служб.

RFC 5424 поддерживает **структурированные данные** - пары ключ-значение для произвольных полей.

#### syslog vs journald

**syslog остаётся нужен**:

- Чёткое объединение логов на нескольких машинах
- rsyslogd модульна, может писать в разные форматы и БД

**journald**:

- Фокус на логирование одной машины в едином формате
- Может передавать логи в другие регистраторы
- systemd собирает выход серверов → journald

#### Тренды журналирования

Журналирование эволюционирует:

- Централизованное хранение → интернет-сервисы и БД
- Назначение: не только для чтения админом, но и для поиска, анализа, интеграции с приложениями
- Безопасность: добавлена шифрование, аутентификация (RFC 5424), но остаются проблемы с доверием и проверкой подлинности источников

## 7.2 Структура каталога /etc

Большинство конфигов системы находятся в **/etc**. Исторически каждая программа имела свой файл конфигурации, что привело к скученности каталога и сложностям в поддержке.

**Решение:** размещение файлов конфигурации в подкаталогах (/etc/systemd, /etc/grub.d и т.д.).

**Правило:** в /etc входят машинозависимые конфигурации (/etc/passwd, /etc/network), но не глобальные значения приложений и системные конфиги по умолчанию (хранятся в /usr/lib/systemd и др.).

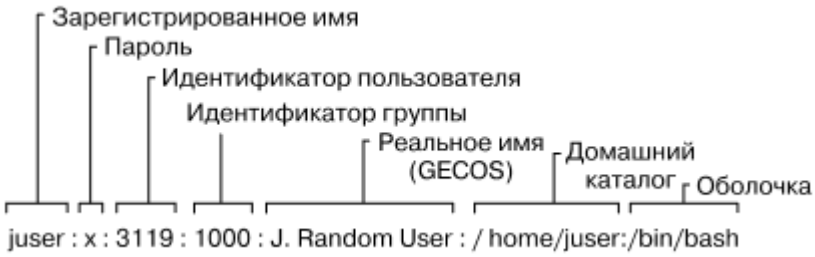
# 7.3 Файлы управления пользователями

В Unix пользователи - это числа (UID) в ядре, но на уровне ОС используются имена (usernames). Любая программа должна переводить имя в UID для работы с ядром.

## 7.3.1 Файл /etc/passwd

Текстовый файл сопоставляет имена пользователей с UID. **Формат:** 7 полей, разделённых двоеточиями.

| **Поле** | **Описание** |
|---|---|
| Имя пользователя | Логин (root, daemon и т.д.) |
| Пароль | x - в shadow-файле; * - вход запрещен; пусто - нет пароля (опасно!) |
| UID | Идентификатор пользователя в ядре; должен быть уникальным |
| GID | Идентификатор основной группы (из /etc/group) |
| GECOS | Полное имя пользователя (может содержать телефоны, комнаты) |
| Домашний каталог | Каталог пользователя |
| Оболочка | Программа при входе в сеанс (обычно /bin/bash или /bin/sh) |




**Пример:**

```
root:x:0:0:Superuser:/root:/bin/sh
juser:x:3119:1000:J. Random User:/home/juser:/bin/bash
nobody:*:65534:65534:nobody:/home:/bin/false
```

**Важно:**

1. Синтаксис строг, комментарии и пустые строки не допускаются
2. Пароли не хранятся в passwd, а хранятся в shadow-файле (доступен только root)
3. Запись в passwd + домашний каталог = учётная запись (account)

## 7.3.2 Особые пользователи

**Суперпользователь (root):** UID 0, GID 0 - единственный UID, имеющий значение для ядра.

**Псевдопользователи:** не могут войти в систему (оболочка /bin/false или /nologin), но системе разрешено запускать процессы от их имени. Примеры:

- **nobody** - непривилегированный, обычно не может ничего записывать (безопасность)

- **daemon** и др. - для системных сервисов

Это соглашения пользовательского пространства; ядро различает только UID 0.

## 7.3.3 Файл /etc/shadow

Теневой файл пароля содержит данные аутентификации: зашифрованные пароли и информацию об истечении срока. Введён для более гибкого и безопасного хранения паролей вместо /etc/passwd. Обычные пользователи не имеют прав на чтение.

Работает вместе с **PAM** (Pluggable Authentication Modules) и конфигом /etc/login.defs.

## 7.3.4. Управление пользователями и паролями

**Команды для обычных пользователей:**

- **passwd** - изменить свой пароль
- **chfn** - изменить полное имя (GECOS)
- **chsh** - изменить оболочку (должна быть в /etc/shells)

Все эти команды - suid-root утилиты (потому что только root может изменить /etc/passwd).

**Управление от суперпользователя:**

| **Операция** | **Команда** |
|---|---|
| Установить пароль | passwd <user> |
| Добавить пользователя | adduser |
| Удалить пользователя | userdel |
| Редактировать passwd безопасно | vipw |
| Редактировать shadow безопасно | vipw -s |

**Избегайте** прямого редактирования passwd текстовым редактором (ошибки синтаксиса, конфликты при параллельных изменениях).

**vipw** - создаёт резервные копии и блокирует файл во время редактирования для безопасности.

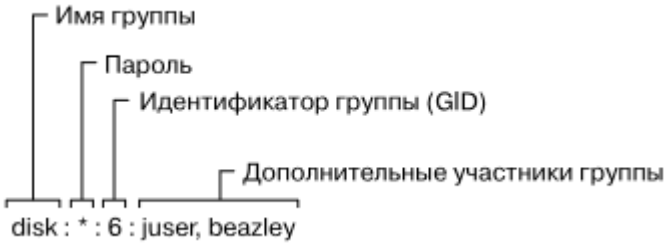
## 7.3.5 Работа с группами пользователей

**Группы** - способ организации совместного доступа к файлам между пользователями через биты разрешений. Менее актуальна в современных системах (общие рабочие станции стали редкостью).

Файл **/etc/group** определяет GID, соответствующие записям /etc/passwd.

| **Поле** | **Описание** |
|---|---|
| Название группы | Отображается в ls -l |
| Пароль группы | Почти никогда не используется; * или x (в /etc/gshadow) |

| Поле | Описание |
|---|---|
| GID | Идентификатор группы (должен быть уникальным) |
| Список пользователей | Необязателен; пользователи также в группе, если имеют соответствующий GID в passwd |




**Примечание:** дистрибутивы часто создают для каждого пользователя личную группу с его именем.

**Просмотр групп пользователя:**

```
groups
```

# 7.4 Команды getty и login

**getty** - подключается к терминалам и отображает приглашение входа. В Linux используется в основном на виртуальных терминалах (Ctrl+Alt+F1). Часто в виде **agetty**:

```
/sbin/agetty -o -p -- \u --noclear tty1 linux
```

**Процесс входа:**

```
1. getty заменяется программой **login**
2. login запрашивает пароль
3. При верном пароле login заменяется (exec) оболочкой пользователя
4. При ошибке: "Login incorrect"
```

Аутентификация выполняется через **PAM** (Pluggable Authentication Modules).

**Примечание:** устаревшие параметры вроде скорости бодов (38400) игнорируются виртуальными терминалами.

**В современных системах getty и login редко используются:** большинство входят через графический интерфейс (всякие DM) или SSH.

# 7.5 Установка времени

Системные часы ядра должны быть максимально точными. Ядро опирается на аппаратные часы RTC (Real-Time Clock) при загрузке, но часы ядра дрейфуют со временем.

**Основной принцип:** используйте **UTC** для системных часов, чтобы избежать проблем с часовыми поясами и летним временем.

Время ядра - это секунды с 1970-01-01 00:00 UTC

RTC - Аппаратные часы с питанием от батареи; менее точны

Time drift - это сдвиг между временем ядра и истинным; исправляется демоном NTP, не hwclock[1)]

**Установка RTC на UTC:**

```
hwclock --systohc --utc
```

## 7.5.1 Представление времени ядра и часовые пояса

Системные часы = секунды с 1970-01-01 00:00 UTC. Пользовательские программы преобразуют в локальное время.

**Управление часовыми поясами:**

- **/etc/localtime** - текущий часовой пояс (двоичный файл)
- **/usr/share/zoneinfo** - база часовых поясов

**Установка часового пояса:**

```
# Копирование или симлинк
ln -sf /usr/share/zoneinfo/Europe/Moscow /etc/localtime

# Или интерактивная утилита
tzselect
```

**Использование альтернативного пояса:** (через переменные окружения)

```
# На сеанс
export TZ=US/Central
date

# На одну команду
TZ=US/Central date
```

## 7.5.2 Сетевое время

При постоянном интернете используйте демон NTP для синхронизации времени с удалённым сервером.

| Демон | Описание | Применение |
|---|---|---|
| **timesyncd** | Встроен в systemd; по умолчанию включен | Системы с интернетом (рекомендуется) |
| **ntpd** | Старый демон; нужно отключить timesyncd | Если требуется расширенный контроль |

| Демон | Описание | Применение |
|---|---|---|
| **chronyd** | Поддерживает время при периодическом интернете | Мобильные системы, ноутбуки |

**Конфигурация timesyncd:** /etc/systemd/timesyncd.conf (изменение серверов времени).

**Информация о серверах:** www.ntppool.org

**Синхронизация RTC с сетевым временем:**

```
# После синхронизации с NTP
hwclock --systohc --utc
```

Многие дистрибутивы это делают автоматически.

# 7.6 Планирование повторяющихся задач с помощью команды cron и юнитов таймера

Два способа запуска повторяющихся задач: **cron** и **таймеры systemd**. Используется для автоматизации (например, ротация логов).

**Формат crontab:**

Строка: **минута час день_месяца месяц день_недели команда**

| Поле | Диапазон | Пример |
|---|---|---|
| минута | 0-59 | 15 |
| час | 0-23 | 09 |
| день месяца | 1-31 | * |
| месяц | 1-12 | * |
| день недели | 0-7 (0,7=воскресенье) | * |

**Символ * = все значения**

```
# Ежедневно в 9:15
15 09 * * * /home/juser/bin/spmake

# Только 14-го числа в 9:15
15 09 14 * * /home/juser/bin/spmake

# 5-го и 14-го числа в 9:15
15 09 5,14 * * /home/juser/bin/spmake
```

## 7.6.1 Установка файлов Crontab

Каждый пользователь имеет свой crontab в **/var/spool/cron/crontabs/**

Команды:

| | |
|---|---|
| crontab file | Установить файл как текущий crontab |
| crontab -l | Список заданий |
| crontab -e | Редактировать текущий crontab |
| crontab -r | Удалить crontab |

## 7.6.2 Системные файлы Crontab

**/etc/crontab** - системный файл (редактировать напрямую, не через crontab)

Отличие: дополнительное поле **пользователя** перед командой

```
# Формат: минута час день месяц день_недели пользователь команда
42 6 * * * root /usr/local/bin/cleansystem > /dev/null 2>&1
```

Дополнительные файлы в **/etc/cron.d/** (любое имя, формат как /etc/crontab)

## 7.6.3 Юниты таймера

Альтернатива cron - **таймер systemd**. Требует два юнита: **таймер** (активация) + **служебный юнит** (задача).

Файлы в **/etc/systemd/system/**

### Юнит таймера (loggertest.timer)

```
[Unit]
Description=Example timer unit

[Timer]
OnCalendar=*-*-* *:00,20,40
Unit=loggertest.service

[Install]
WantedBy=timers.target
```

**OnCalendar** - время в формате год-месяц-день час:минута:секунда

Запускается каждые 20 минут. Синтаксис:

- * = все значения
- , = несколько значений (пример: 00,20,40)
- */N = каждые N единиц (пример: *:00/20 = каждые 20 минут)

Полный синтаксис: страница **systemd.time(7)**

**Связанный служебный юнит (loggertest.service)**

```
[Unit]
Description=Example Test Service

[Service]
Type=oneshot
ExecStart=/usr/bin/logger -p local3.debug I\'m a logger
```

**Type=oneshot** - служба запускается, выполняет команду и завершается

Преимущества oneshot:

- Несколько **ExecStart** в одном файле
- Строгий порядок зависимостей (Wants, Before)
- Точные записи о времени в журнале

Быстрые задачи могут завершиться до записи в журнал (эффект гонок)

### 7.6.4. Утилита cron против юнитов таймера

| **Критерий** | **cron** | **Таймеры systemd** |
|---|---|---|
| Сложность | **Простая** | Сложнее |
| Совместимость | **Высокая** (множество служб) | Зависит от системы |
| Пользовательские задачи | **Легко** | Требует юнит-файлов |
| Отслеживание процессов (cgroups) | Слабо | **Эффективно** |
| Логирование | Базовое | **Детальное** (journal) |
| Параметры времени | Ограничены | **Расширенные** |
| Зависимости systemd | Нет | **Да** |

**cron вряд ли исчезнет** благодаря простоте. **Таймеры - будущее** системных задач.

Команда **systemd-run** создаёт таймеры без юнит-файлов, но многие предпочитают cron

## 7.7 Планирование разовых задач с помощью службы at

Выполнить задачу **один раз в будущем** без cron:

```
$ at 22:30
at> myjob
# Ctrl+D для завершения
```

| **Команда** | **Действие** |
|---|---|
| atq | Список запланированных задач |
| atrm | Удалить задачу |
| at 22:30 30.09.15 | Запланировать на дату и время |

### 7.7.1 Аналоги таймера

Замена **at** на **systemd-run**:

```
# systemd-run --on-calendar='2022-08-14 18:00' /bin/echo this is a test
Running timer as unit: run-rbd000cc6ee6f45b69cb87ca0839c12de.timer
Will run service as unit: run-rbd000cc6ee6f45b69cb87ca0839c12de.service

# Смещение по времени (выполнить через 30 минут)
# systemd-run --on-active=30m /bin/command

# Просмотр таймеров
systemctl list-timers
```

**–on-calendar** требует **будущую дату и время**, иначе таймер запустится ежедневно

## 7.8. Юниты таймера обычных пользователей

Создать таймер от имени обычного пользователя:

```
$ systemd-run --user --on-calendar='2022-08-14 18:00' /bin/echo test
```

**Проблема:** таймер не запустится, если пользователь вышел из системы

**Решение - сохранить менеджер пользователя:**

```
# Для текущего пользователя
$ loginctl enable-linger

# Для другого пользователя (от root)
# loginctl enable-linger user
```

## 7.9 Доступ пользователя

Управление входом, переключением пользователей и защитой. **Продвинутый уровень.**

### 7.9.1 ID пользователей и переключение пользователей

**Переключение = смена идентификатора пользователя (UID)** через:

```
1. Исполняемый файл **setuid** (пример: sudo, su)
2. Системные вызовы **setuid()**
```

**Правила ядра (основные)**

1. Процесс может запустить `setuid` файл, если **имеет права доступа** на него
2. **Суперпользователь `(UID 0)`** может вызвать `setuid()` и **стать любым пользователем**
3. **Обычный пользователь** практически **не может использовать `setuid()`**

**Следствие:** sudo имеет права setuid от root → может вызвать setuid() и переключиться на другого пользователя

Переключение пользователей независимо от паролей и имён - чисто концепция пользовательского пространства (как /etc/passwd)

## 7.9.2 Владельцы процессов, действующий UID, реальный UID и сохраненный UID

Каждый процесс имеет **несколько UID идентификаторов**:

| UID | Назначение | Пример |
|---|---|---|
| **euid** (effective) | Определяет **права доступа** к файлам (исполнитель) | Setuid программа меняет на владельца файла |
| **ruid** (real) | **Кто инициировал** процесс (владелец) | Обычный пользователь, запустивший sudo |
| **saved UID** | Процесс может переключиться между euid и ruid | Позволяет вернуться к исходному UID |
| **fsuid** | Редко: доступ к файловой системе | Практически не используется |

**Аналогия:** euid = исполнитель, ruid = владелец

**ruid определяет, кто может завершить/контролировать процесс**

Пример: пользователь A запускает setuid программу от B → A остаётся владельцем (ruid) и может убить процесс

**Просмотр UID**

```
$ ps -eo pid,euser,ruser,comm
```

Для большинства процессов euid = ruid (поэтому они совпадают в выводе по умолчанию)

**Типичное поведение setuid программ**

**sudo, su и похожие программы явно меняют оба UID (euid и ruid)** чтобы избежать побочных эффектов и проблем с доступом

Если нужно запретить sudo менять ruid - добавить в **/etc/sudoers**:

```
Defaults stay_setuid
```

**Осторожно**: может повлиять на другие setuid программы

#### Последствия для безопасности

Setuid программы - **основной путь взлома систем**

**Критические риски:**

- Setuid копия bash → **любой локальный пользователь = полный контроль**
- Баги в setuid программах = вторжение
- Большое количество setuid файлов = большой риск

**Защита:**

- **Минимизировать** количество setuid программ
- **Проверять качество** setuid кода
- **Идентифицировать пользователей** надёжными паролями

## 7.9.3 Идентификация пользователя, аутентификация и авторизация

Система безопасности включает три области:

- **Идентификация** - определение, кем является пользователь
- **Аутентификация** - доказательство личности
- **Авторизация** - определение разрешенных действий

Ядро Linux оперирует только числовыми UID. Вся аутентификация (имена, пароли) реализована в userspace.

**Упрощенный процесс получения имени пользователя:**

1. Вызов `getuid()` для получения `euid`
2. Открыть `/etc/passwd`
3. Прочитать строку, разобрать по полям (разделитель ':')
4. Сравнить `UID` из файла с полученным
5. При совпадении - найдено имя пользователя

## 7.9.4 Применение библиотек для получения информации о пользователе

**Стандартные библиотеки** упрощают работу. Вместо ручной реализации используется getpwuid() после geteuid().

**Преимущества:** Изменение источника данных (с /etc/passwd на LDAP) не требует изменения программ.

**Недостатки традиционного подхода с паролями в /etc/passwd:**

- Отсутствие единого стандарта шифрования
- Требует доступа к зашифрованному паролю
- Запрос пароля при каждой аутентификации
- Не поддерживает альтернативные методы (токены, смарт-карты, биометрия)

# 7.10 Подключаемые модули аутентификации (PAM)

**PAM** (Pluggable Authentication Module) - стандарт от Sun Microsystems (1995) для гибкой аутентификации. Система разделяемых библиотек позволяет добавлять методы аутентификации (двухфакторная, ключи) без изменения приложений.

Модули - динамически загружаемые объекты (.so). Пример: pam_unix.so проверяет пароль.

## 7.10.1 Конфигурация PAM

Файлы конфигурации: **/etc/pam.d/** (старые системы: /etc/pam.conf)

**Структура строки конфигурации (3 поля):**

| Поле | Описание | Пример |
|---|---|---|
| **Тип функции** | Операция, запрашиваемая приложением | auth, account, session, password |
| **Аргумент управления** | Поведение при успехе/неудаче | sufficient, requisite, required |
| **Модуль** | Модуль аутентификации | pam_shells.so |

**Типы функции:**

- **auth** - аутентификация (кто ты?)
- **account** - проверка статуса учетной записи
- **session** - действия только для текущего сеанса (motd)
- **password** - изменение пароля

**Управляющие аргументы (стек правил):**

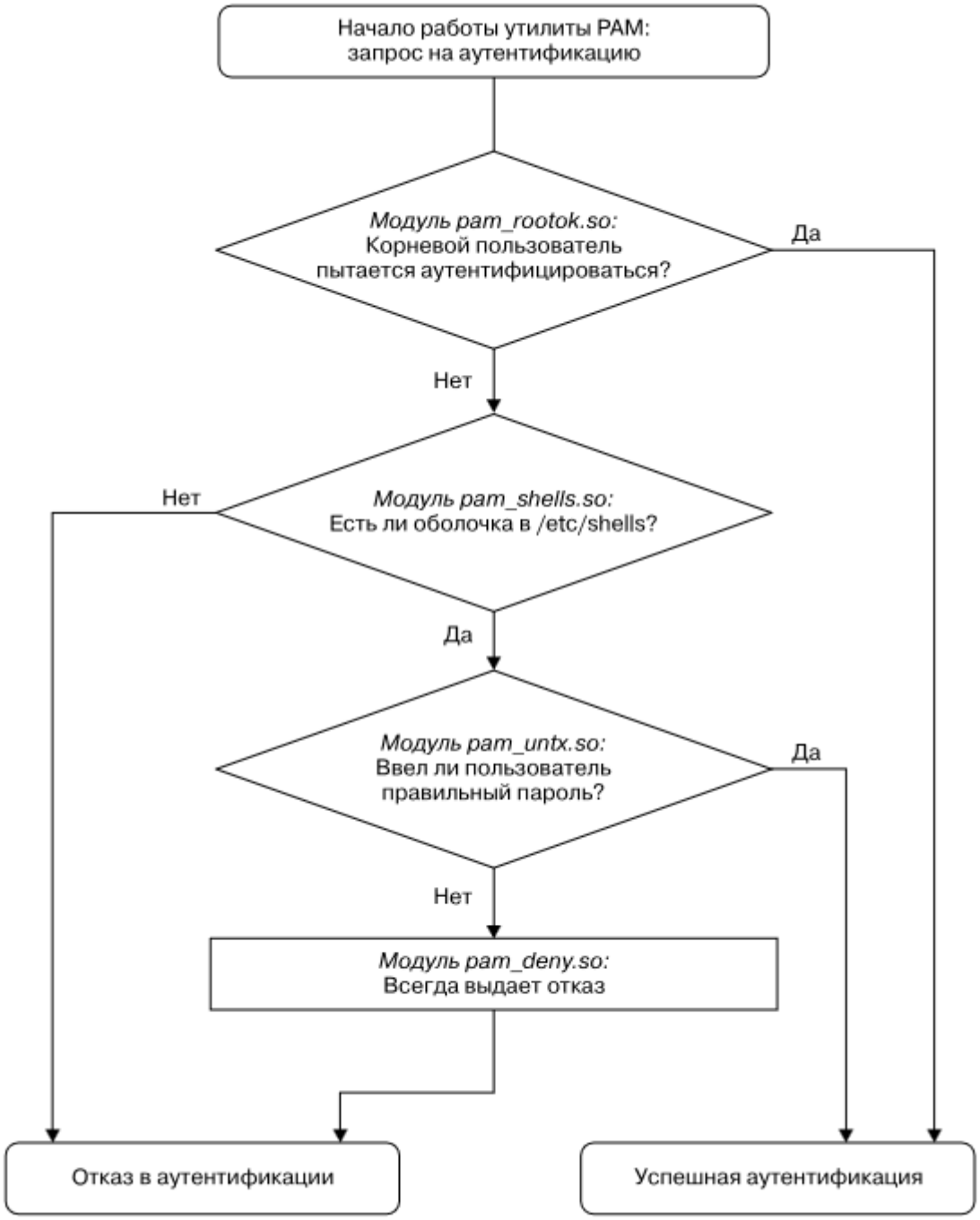
- **sufficient** - успех → немедленный успех, пропустить остальные; неудача → далее
- **requisite** - успех → далее; неудача → немедленный отказ
- **required** - успех → далее; неудача → далее, но всегда отказ в конце

**Пример стека (функции аутентификации chsh):**

```
auth sufficient pam_rootok.so
auth requisite pam_shells.so
auth sufficient pam_unix.so
auth required pam_deny.so
```

При такой конфигурации, когда команда chsh запрашивает у PAM функцию аутентификации, PAM выполняет следующее

1. Модуль pam_rootok.so проверяет, не пытается ли суперпользователь пройти аутентификацию. Если это так, процесс немедленно выполнится успешно и не будет пытаться выполнить дальнейшую аутентификацию. Это сработает, потому что аргумент управления имеет значение sufficient, что означает: выполнения этого действия достаточно для того, чтобы модуль PAM немедленно сообщил об успехе chsh. В противном случае он переходит к шагу 2.
2. Модуль pam_shells.so проверяет, указана ли оболочка пользователя в файле /etc/shells. Если ее там нет, модуль возвращает ошибку и аргумент управления requisite указывает, что модуль PAM должен немедленно сообщить об этом сбое chsh и не пытаться выполнить аутентификацию. В противном случае модуль успешно завершит выполнение и в соответствии с требованиями аргумента управления requisite перейдет к шагу 3.
3. Модуль pam_unix.so запрашивает у пользователя пароль и проверяет его. Аргументу управления присвоено значение sufficient, поэтому успеха данного модуля (правильного пароля) достаточно, чтобы PAM сообщил об этом оболочке chsh. Если пароль неверен, PAM переходит к шагу 4.
4. Модуль pam_deny.so никогда не выполняется, и поскольку аргумент управления установлен в значение required, PAM сообщает об ошибке в оболочку chsh. Это значение используется по умолчанию для случаев, когда больше нет вариантов действий. (Обратите внимание, что аргумент управления required не приводит к немедленному сбою функции PAM: он будет запускать все строки, оставшиеся в его стеке, но при этом модуль всегда будет сообщать приложению о сбое.)




**Аргументы модулей:** После имени модуля (пример: `pam_unix.so nullok` - пароль может отсутствовать)

## 7.10.2 Советы по синтаксису конфигурации PAM

- Список модулей: `man -k pam_`
- Локализация: `locate pam_unix.so`
- Комментарии в /etc/pam.d указывают на источник генерации
- **/etc/pam.d/other** - конфигурация по умолчанию (обычно запрещает всё)
- Включение файлов: `@include` или управляющие аргументы
- Дополнительные настройки в **/etc/security**

## 7.10.3 Модули PAM и пароли

**/etc/login.defs** - конфигурация shadow-паролей (алгоритм шифрования). Используется редко при наличии PAM.

**Установка пароля (функция password):**

```
password sufficient pam_unix.so obscure sha512
```

Аргументы: obscure (проверка надежности), sha512 (алгоритм).

**Проверка пароля (функция auth):** pam_unix.so автоматически определяет алгоритм через libcrypt (перебор), нет аргументов шифрования.

**Поиск конфига:** `grep password.*unix /etc/pam.d/*`

1)

Не пытайтесь исправить сдвиг во времени с помощью команды hwclock, потому что системные события, основанные на времени, могут быть потеряны или искажены. Вы можете запустить утилиту, например adjtimex, чтобы плавно обновлять часы на основе RTC, но лучше корректировать системное время с помощью демона сетевого времени