

Работа с планировщиком задач в Unix-подобных системах с использованием языка программирования Python

Болтовский Гавриил Александрович

Приамурский государственный университет им. Шолом-Алейхема

Студент

Аннотация

В рамках данной статьи рассматривается принцип работы планировщика задач cron, структура расписания добавляемых задач, а также его применение в работе API-сервиса. Также в статье рассмотрена работа модуля «python-crontab», позволяющий работать с планировщиком задач с помощью языка программирования Python. Описана реализация программных скриптов создания задач для cron, а также представлены скрипты задач, запускаемых по указанному расписанию.

Ключевые слова: Python, API, Crontab, задачи Cron

Working with the task scheduler in Unix-like systems using the Python programming language

Boltovsky Gavriil Alexandrovich

Sholom-Aleichem Priamursky State University

Student

Abstract

This article discusses the principle of operation of the cron task scheduler, the structure of the schedule for added tasks, and its application in the API service. The article also discusses the operation of the python-crontab module, which allows you to work with the task scheduler using the Python programming language. It describes the implementation of software scripts for creating cron tasks and presents scripts for tasks that are run according to a specified schedule.

Keywords: Python, API, crontab, cron tasks

1. Введение

1.1 Актуальность

В настоящее время планировщики задач остаются критически важным инструментом, несмотря на эволюцию архитектур и технологий. Их актуальность обусловлена не только рутинными задачами очистки логов, а фундаментальными потребностями цифровой инфраструктуры.

Планировщики позволяют минимизировать человеческий фактор в критических задачах, гарантируют предсказуемость и следование заданному алгоритму, автоматизируют ряд рутинных задач.

Умение работать с планировщиками задач не менее важный навык, как и написание программного кода, особенно во время продвижения микросервисной архитектуры, как предпочтительной во многих приложениях. При этом многие языки программирования предоставляют модули для работы с популярными планировщиками задач, что делает процесс их использования более простым для разработчиков.

1.2 Обзор исследований

С.В. Козлов и И.А. Карпов в своей статье описали назначение CRON-задач, как механизма запуска задач по расписанию [1]. В статье приведены примеры продуктовых задач, выполняющихся по расписанию, а также описана необходимость горизонтального масштабирования сервисов в микросервисной архитектуре.

И.В. Серебренников в своей работе описал принципы использования демона Cron в linux системах, так же рассмотрена интеграция Django и cron - сопутствующий модуль django-crontab [2]. Также в статье подвергнут анализу способ создания видеоряда с помощью утилиты opencv.

А.И. Нефеденков и Ж.Ж. Юлдашева рассмотрели планирование задач в Linux при помощи утилит cron и systemd timers, а также провели их сравнительный анализ [3]. Исследование проводилось с помощью виртуальных машин с установленным на них дистрибутивом АЛБТ Linux в среде VirtualBox.

1.3 Цель исследования

Цель исследования – рассмотреть принцип работы планировщика задач cron и написать для существующего API-сервиса скрипты для работы с задачами в crontab.

2 Материалы и методы

Для решения поставленных задач выбран язык программирования Python. Для внедрения предложенного решения выбрана система запуска программного кода в веб-ориентированных средах выбрана система ExecEngine.

3 Результаты и обсуждения

Планировщик задач – это механизм, позволяющий автоматически выполнять команды или скрипты в заданное время или с определённой периодичностью. Основными инструментами для этого является cron.

Cron представляет собой один из самых старых и надёжных инструментов для автоматизации в Unix/Linux, существующий с 1970-х годов. Демон (фоновый процесс) crond запускается при загрузке системы и постоянно работает в фоне. Каждую минуту он проверяет:

1. Системные crontab-файлы (/etc/crontab, /etc/cron.d/*).
2. Пользовательские crontab-файлы (хранятся в /var/spool/cron/crontabs/).

Если наступает время запуска задачи – cron выполняет команду в оболочке пользователя, указанного в задании. Можно отметить также и принцип работы с таблицей задач, просмотр которой доступен через `crontab`. У команд, находящихся в расписании, есть свой формат, подразумевающий несколько секций: расписание (когда выполняется задача), команда для исполнения и вывод результата (см. рис. 1).

Расписание включает в себя несколько значений: минуты, часы, день в месяце, месяц и день недели. Также предусмотрена возможность работы со специальными символами:

1. Символ «*» определяет любое значение на заданной позиции. Например, расписание «* * * * *» означает выполнение задачи каждую минуту.

2. Символ «,» позволяет задать список значений. Например, расписание «1,3,5 * * * *» означает выполнение задачи в 1-ю, 3-ю и 5-ую минуты.

3. Символ «/» позволяет задать период выполнения задачи относительно временной единицы. Например, расписание «*/10 * * * *» означает выполнение каждые 10 минут.

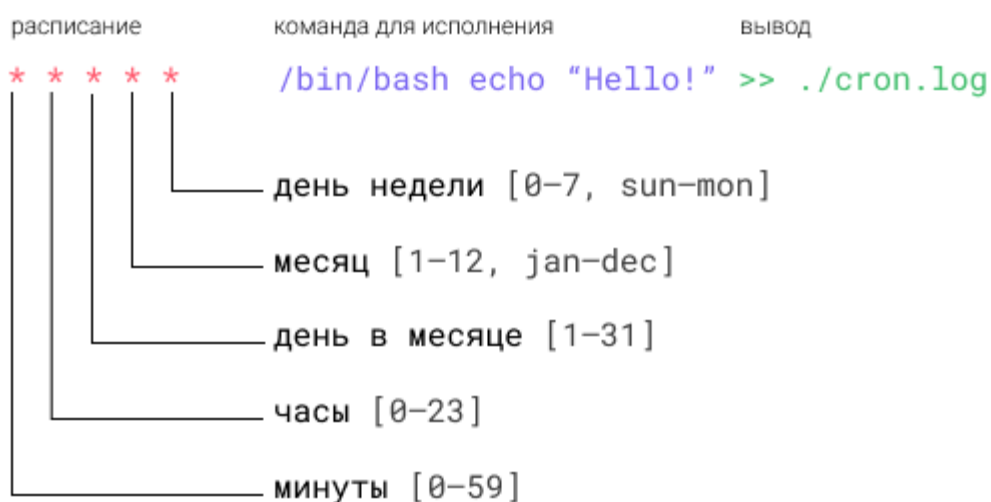


Рисунок 1. Формат `crontab`-файла

Многие языки программирования позволяют работать с cron с помощью соответствующих модулей. Исключением не является язык программирования Python. В Python предусмотрен модуль под названием «`crontab`», позволяющий реализовать полноценную работу с планировщиком задач. Модуль является сторонним, поэтому требуется установка через пакетный менеджер:

```
pip install python-crontab
```

В рамках данной статьи будем рассматривать реальный пример применения данного модуля в реализации задачи очистки таблиц базы данных.

С 2023-го года на базе Приамурского государственного университета им. Шолом-Алейхема работает информационная система по организации

обучения программированию в веб-пространстве. ИС использует дополнительный API-сервис ExecEngine, отвечающий за проверку программного кода на заранее заданных входных данных.

Одной из проблем работы API-сервиса является стремительное накопление отправленных клиентских запросов. За день в сервис может поступать до 2-3 тысяч запросов на запуск программного кода при интенсивной работе только одной учебной группы на сайте обучения программированию.

Таблица пользовательских отправок включает в себя более десятка и колонок, в некоторых могут быть сохранены длинные строки, например исходный код программы, запущенной на стороне сервера, закодированной в Base64.

Такое количество информации может создать серьёзную проблему для API-сервиса, если не очищать таблицы от устаревшей информации. Ручное удаление записей не может рассматриваться как решение проблемы в настоящих реалиях, когда существует много возможностей для автоматизации работы различных процессов. Поэтому стоит задействовать crontab.

API-сервис распространяется на бесплатной основе и исходный код доступен на Github, что создаёт дополнительно ещё одну задачу: как упростить задачу создания задач для cron, чтобы этот процесс не нужно было настраивать вручную? Для этого был разработан ряд Python-скриптов, размещённых в соответствующей директории. Основным является «cron_tasks_init.py», создающий задачи и сохраняющий новую информацию в crontab. Скрипт получает данные из конфигурации API (если это необходимо для задачи cron) и затем вызывает функции для создания задач в crontab.

На данный момент предусмотрены 2 задачи: очистка таблицы отправленных запросов со стороны клиентов и очистка таблицы токенов авторизации.

Алгоритм функции очистки таблицы отправленных запросов строится следующим образом: сначала сохраняется информация о пути к скрипту, интерпретатору Python и журналу выполнения задачи. Затем производится проверка на существование этой задачи и если она присутствует, то скрипт удаляет её. Затем скрипт создаёт новую задачу с предварительно сохранёнными данными и сообщает пользователю об этом. Задача очистки таблицы отправленных запросов выполняется с периодичностью в FREQUENCY_OF_CLEANING_SUBMISSIONS, заданное разработчиком до запуска API-сервиса:

```
def clear_submissions_init(cron):
    SCRIPT_PATH =
os.path.abspath("app/cron/clear_submissions.py")
    PYTHON_PATH = "/usr/bin/python3"
    LOG_PATH =
os.path.abspath("app/logs/submissions_cleanup.log")

    for job in cron.find_comment("auto-cleanup-submissions"):
```

```

        cron.remove(job)
    job = cron.new(command=f"cd /app && {PYTHON_PATH}
{SCRIPT_PATH} >> {LOG_PATH} 2>&1",
                    comment="auto-cleanup-submissions")
    if FREQUENCY_OF_CLEANING_SUBMISSIONS >= 1:
        job.day.every(FREQUENCY_OF_CLEANING_SUBMISSIONS)
    else:
        job.day.every(1)
    print(f"[INFO] Cron job set to run every
{FREQUENCY_OF_CLEANING_SUBMISSIONS} days: {job.slices}
{job.command}")

```

Сам скрипт очистки базы данных сохранён в той же директории, что и «cron_tasks_init.py». Алгоритм его работы также прост: создаётся подключение к базе данных (информация о подключении также определена в файле конфигурации) и производится полная очистка таблицы, после чего скрипт выводит информацию о количестве удалённых записей (см. рис. 2):

```

DATABASE_HOST: str = _parse_value(config['CONNECTION
SETTINGS']['DATABASE_HOST'])
DATABASE_NAME: str = _parse_value(config['CONNECTION
SETTINGS']['DATABASE_NAME'])
DATABASE_USER: str = _parse_value(config['CONNECTION
SETTINGS']['DATABASE_USER'])
DATABASE_PASSWORD: str = _parse_value(config['CONNECTION
SETTINGS']['DATABASE_PASSWORD'])

TABLE_NAME = "submissions"
TIME_COLUMN = "created_at"

DATABASE_URL =
f"postgresql+psycopg2://{DATABASE_USER}:{DATABASE_PASSWORD}@{DAT
ABASE_HOST}/{DATABASE_NAME}"
engine = create_engine(DATABASE_URL)

def cleanup_submissions():
    query = text(f"DELETE FROM {TABLE_NAME}")

    with engine.begin() as conn:
        result = conn.execute(query)
        print(f"[{datetime.utcnow()}] Removed Submissions:
{result.rowcount}")

if __name__ == "__main__":
    cleanup_submissions()

```

```

root@lab2b20ceda9:/app/logs# awk -F 'Submissions: ' '$2 != 0' submissions_cleanup.log
[2025-12-10 00:00:02.779873] Removed Submissions: 847
[2025-12-10 05:22:01.638225] Removed Submissions: 5
[2025-12-10 05:39:01.596722] Removed Submissions: 4
[2025-12-10 05:40:02.344305] Removed Submissions: 2
[2025-12-10 05:47:01.751455] Removed Submissions: 4
[2025-12-10 05:53:01.554875] Removed Submissions: 1
[2025-12-10 05:54:02.181164] Removed Submissions: 10
[2025-12-10 06:00:02.024156] Removed Submissions: 4
[2025-12-10 06:08:02.286178] Removed Submissions: 5
[2025-12-10 06:23:01.869077] Removed Submissions: 5
[2025-12-10 06:29:01.912094] Removed Submissions: 4
[2025-12-10 06:31:02.493792] Removed Submissions: 3
[2025-12-10 06:32:02.312679] Removed Submissions: 5
[2025-12-10 06:34:01.688727] Removed Submissions: 5
[2025-12-10 06:48:02.245591] Removed Submissions: 4
[2025-12-10 06:53:01.580501] Removed Submissions: 6
[2025-12-16 00:00:01.740336] Removed Submissions: 6
[2025-12-25 00:00:02.190271] Removed Submissions: 184
[2026-01-07 00:00:02.537626] Removed Submissions: 21
[2026-01-13 00:00:01.726272] Removed Submissions: 64
[2026-01-13 10:28:02.078514] Removed Submissions: 3

```

Рисунок 2. Журнал очистки таблицы отправленных запросов (логи с удалением нуля записей пропущены)

Работа функции создания задачи удаления токенов авторизации совпадает с ранее описанной функцией удаления пользовательских запросов:

```

def clear_api_tokens_init(cron):
    SCRIPT_PATH =
os.path.abspath("app/cron/clear_api_tokens.py")
    PYTHON_PATH = "/usr/bin/python3"
    LOG_PATH = os.path.abspath("app/logs/api_cleanup.log")

    for job in cron.find_comment("auto-cleanup-api-tokens"):
        cron.remove(job)
        job = cron.new(command=f"cd /app && {PYTHON_PATH}
{SCRIPT_PATH} >> {LOG_PATH} 2>&1",
                        comment="auto-cleanup-api-tokens")
        if FREQUENCY_OF_CLEANING_API_TOKENS >= 1:
            job.minute.every(FREQUENCY_OF_CLEANING_API_TOKENS)
        else:
            job.minute.every(1)
        print(f"[INFO] Cron job set to run every
{FREQUENCY_OF_CLEANING_API_TOKENS} minutes: {job.slices}
{job.command}")

```

Также сам скрипт очистки таблицы токенов авторизации совпадает со скриптом очистки таблицы пользовательских запросов, за тем исключением, что удаляются только уже неактивные токены (время действия которых истекло):

```

def cleanup_api_tokens():
    cutoff_date = datetime.utcnow() -
timedelta(minutes=MINUTES_EXPIRE)

```

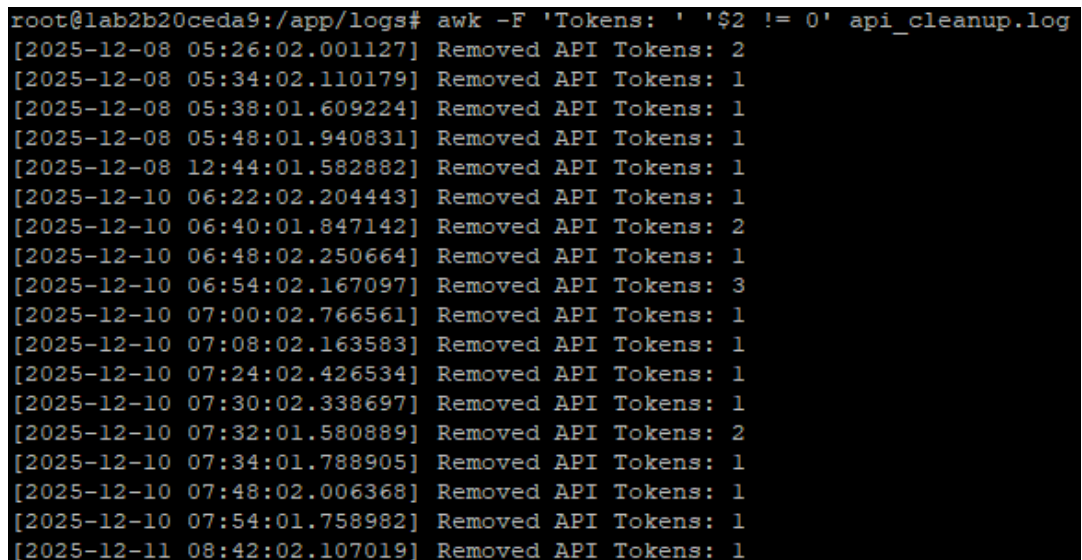
```

query = text(f"DELETE FROM {TABLE_NAME} WHERE {TIME_COLUMN}
< :cutoff")

with engine.begin() as conn:
    result = conn.execute(query, {"cutoff": cutoff_date})
    print(f"[{datetime.utcnow()}] Removed API Tokens:
{result.rowcount}")

if __name__ == "__main__":
    cleanup_api_tokens()

```



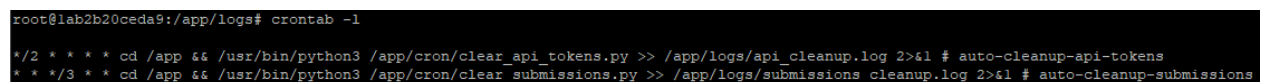
```

root@lab2b20ceda9:/app/logs# awk -F 'Tokens: ' '$2 != 0' api_cleanup.log
[2025-12-08 05:26:02.001127] Removed API Tokens: 2
[2025-12-08 05:34:02.110179] Removed API Tokens: 1
[2025-12-08 05:38:01.609224] Removed API Tokens: 1
[2025-12-08 05:48:01.940831] Removed API Tokens: 1
[2025-12-08 12:44:01.582882] Removed API Tokens: 1
[2025-12-10 06:22:02.204443] Removed API Tokens: 1
[2025-12-10 06:40:01.847142] Removed API Tokens: 2
[2025-12-10 06:48:02.250664] Removed API Tokens: 1
[2025-12-10 06:54:02.167097] Removed API Tokens: 3
[2025-12-10 07:00:02.766561] Removed API Tokens: 1
[2025-12-10 07:08:02.163583] Removed API Tokens: 1
[2025-12-10 07:24:02.426534] Removed API Tokens: 1
[2025-12-10 07:30:02.338697] Removed API Tokens: 1
[2025-12-10 07:32:01.580889] Removed API Tokens: 2
[2025-12-10 07:34:01.788905] Removed API Tokens: 1
[2025-12-10 07:48:02.006368] Removed API Tokens: 1
[2025-12-10 07:54:01.758982] Removed API Tokens: 1
[2025-12-11 08:42:02.107019] Removed API Tokens: 1

```

Рисунок 3. Журнал очистки таблицы токенов авторизации (логи с удалением нуля записей пропущены)

Задачи можно просмотреть, вызвав соответствующую команду в терминале ОС: «crontab -l». Данная команда покажет пользователю все задачи с заданным для них расписанием (см. рис. 4).



```

root@lab2b20ceda9:/app/logs# crontab -l
*/2 * * * * cd /app && /usr/bin/python3 /app/cron/clear_api_tokens.py >> /app/logs/api_cleanup.log 2>&1 # auto-cleanup-api-tokens
*/3 * * * * cd /app && /usr/bin/python3 /app/cron/clear_submissions.py >> /app/logs/submissions_cleanup.log 2>&1 # auto-cleanup-submissions

```

Рисунок 4. Таблица Сгон после создания задач на очистку БД

В качестве демонстрации работы скриптов очистки таблиц были заданы частые запуски в crontab, что делать не рекомендуется (задача очистки таблицы пользовательских отправок выполняется каждую минуту в дни, номер которых равен трём). Можно предложить следующую доработку:

```

if FREQUENCY_OF_CLEANING_SUBMISSIONS >= 1:
    job.setall(f"0 0 */{FREQUENCY_OF_CLEANING_SUBMISSIONS} * *")
else:
    job.setall("0 0 * * *")

```

То есть вместо использования соответствующего метода рекомендуется задавать в таком случае расписание в виде строки. В таком случае задача будет выполняться в 00:00 каждые 3 дня ровно один раз.

Также API-сервис уведомляет разработчика о создании новых задач через «cron_tasks_init.py» (см. рис. 5). Разработчик может узнать периодичность выполнения задачи и её расписание в crontab.

```
api-1 | [INFO] Cron job set to run every 2 minutes: */2 * * * * cd /app && /usr/bin/python3 /app/cron/clear_api_tokens.py >> /app/logs/api_cleanup.log 2>&1
api-1 | [INFO] Cron job set to run every 3 days: 0 0 */3 * * cd /app && /usr/bin/python3 /app/cron/clear_submissions.py >> /app/logs/submissions_cleanup.log 2>&1
```

Рисунок 5. Информация о создании задач в Cron в API-сервисе при запуске

4 Выводы

Таким образом, в рамках данной статьи был рассмотрен cron, модуль для работы с crontab в Python, а также описан пример реализации задач в виде скриптов на Python для периодической очистки таблиц базы данных API-сервиса.

Библиографический список

1. Козлов С.В., Карпов И.А. Масштабирование CRON-задач в распределенных системах // International Journal of Open Information Technologies. 2025. Т.13. № 10. С. 50-57.
2. Серебренников И.В. Использование django-crontab и библиотеки CV2 для организации отдельных кадров в видео // Лучшие научные исследования 2022: Сборник статей VII Международного научно-исследовательского конкурса, Пенза, 05 сентября 2022 года. Пенза: Наука и Просвещение (ИП Гуляев Г.Ю.). 2022. С. 40-42.
3. Нефеденков А.И., Юлдашева Ж.Ж. Использование systemd timers в Linux в качестве замены cron // Научный Альманах ассоциации France-Kazakhstan. 2025. № 2. С. 269-277.