

Консольный PHP

Урок 1

Работаем с PHP-CLI, crontab, systemd, hup сигналами и
создаем своего демона





План курса







- 1 Консольный PHP
- 2 Backend API
- 3 Frontend API
- 4 Тестирование приложений

- 5 Продвинутое unit-тестирование
- 6 Кэширование в PHP
- 7 Очереди в PHP и RabbitMQ
- 8 Доставка приложений

WELCOME



Что будет на уроке сегодня

-  Викторина, которая построена на основании реальных вопросов, которые задают на собеседовании
-  Имитация работы выполнения заданий от тимлида
-  Получим опыт постановки ТЗ от тимлида
-  Научимся по шагам работать над задачей, которая ставится от тимлида
-  Приступим к реализации бота-напоминалки
-  Реализуем демона



Викторина

Построена на основании реальных вопросов,
которые задают на собеседовании



Преподаватель демонстрирует вопросы викторины, зачитывает их, а студенты пишут ответы в чате.



1. Что такое php-cli?

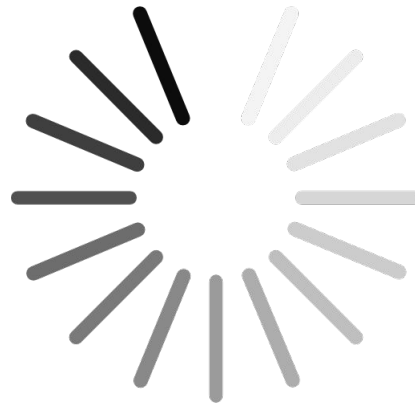
Вопрос без вариантов ответа



Подсказка: загляните в конспект



Совет: напишите ответ в чат или проговорите его



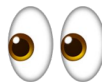


2. В каких случаях применяется php-cli?

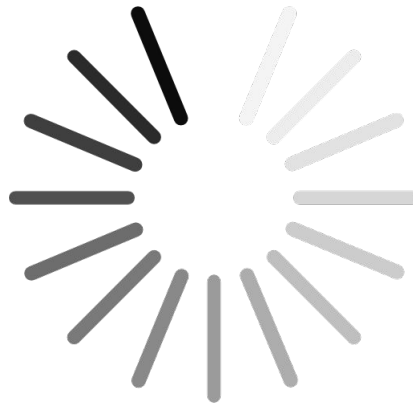
1. Для автоматизации задач: резервного копирования данных, генерации отчетов или обновления баз данных

2. Для любых операций, которые должны выполняться по расписанию или по запросу

3. Для автоматического подключения фреймворков



Подсказка: здесь 2 верных ответа





3. Удавалось ли сталкиваться с systemd? Какова его цель?

Вопрос без вариантов ответа



Подсказка: загляните в конспект



Совет: напишите ответ в чат или проговорите его

[● ◀] systemd



4. В чем преимущество systemd перед cron? Выберите верные утверждения:

1. systemd позволяет планировать запуск служб и выполнение задач

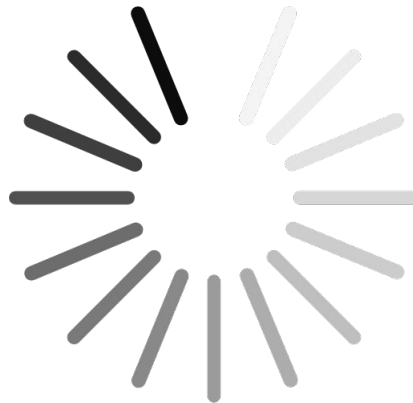
2. таймеры systemd определяются в файлах .timer и могут быть настроены для запуска по событиям (например, после загрузки системы) или с определенными интервалами

3. systemd подходит хорошо для периодических задач, которые не требуют сложной логики или зависимостей между задачами

4. таймеры systemd и связанные с ними службы настраиваются в файлах юнитов в каталоге /etc/systemd/system/



Подсказка: здесь 3 верных ответа





5. Посмотрите на фрагмент кода.

О чем строки: 1 – [Unit],
2 – Description и 3 – After? 📌

[Unit]: здесь мы описываем службу и указываем её зависимости.

Description: описание службы.

After: сообщаем, что служба должна запускаться после загрузки сети.

[Unit]: здесь настраиваем службу.

Description: тип нашей службы.

After: команда для запуска PHP скрипта.

[Unit]: здесь указываем настройки для установки службы

Description: цель, куда устанавливается служба.

After: сообщаем, что служба должна запускаться после загрузки сети.

```
1 [Unit]
2 Description=My PHP Script Service
3 After=network.target
4
5 [Service]
6 Type=simple
7 ExecStart=/usr/bin/php /path/to/your/php/script.php
8 WorkingDirectory=/path/to/your/php/script/directory
9 Restart=always
10
11 [Install]
12 WantedBy=multi-user.target
```



сложность вопроса – высокая



6. В чем разница между службой и таймером? Выберите верные утверждения:

Таймер планирует запуск задач или служб в определенное время или с определенными интервалами

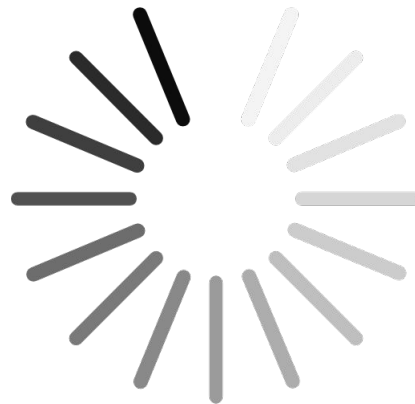
Служба может быть запущена по запросу, после загрузки системы или при других условиях

Таймер может быть запущен по запросу, после загрузки системы или при других условиях

Таймер создается в файлах с расширением .service



Подсказка: здесь 2 верных ответа





7. Что такое завершающие сигналы (HUP)? В каких случаях и для каких целей применяются. Приведите пример 1-2 сигналов.

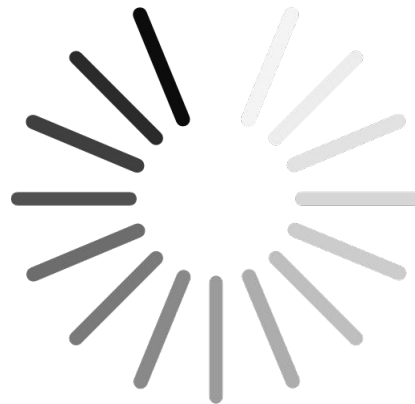
Вопрос без вариантов ответа



Подсказка: загляните в конспект



Совет: напишите ответ в чат или проговорите его



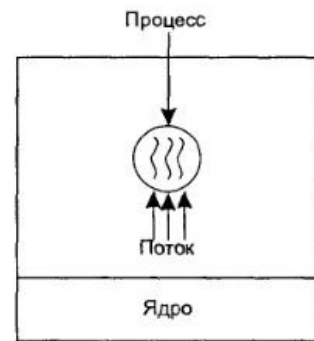
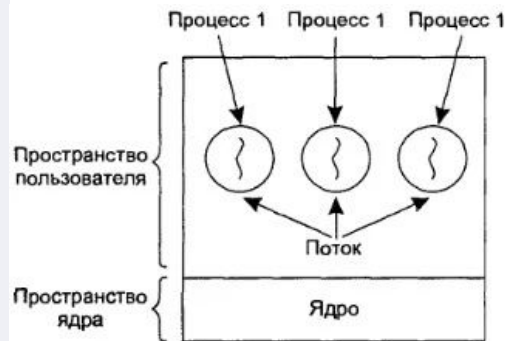


**8. В чем заключается существенное различие между потоком и процессом?
Выберите верное утверждение:**

1. Процесс имеет собственное адресное пространство, где хранит код, данные, файлы, необходимые для выполнения

2. Процессы не изолированы друг от друга

3. Потоки изолированы друг от друга





9. Перед вами определение легковесного процесса в Linux.

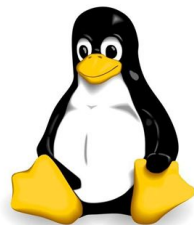
Какое слово пропущено?

1. Однопоточность

2. Многопоточность

3. Сверхпоточность

Легковесный процесс в Linux – это механизм в ядре операционной системы Linux, позволяющий реализовать _____ на уровне пользовательского пространства. То есть, это своего рода потоки, которые используют общие ресурсы внутри одного процесса (или "родительского" процесса), чтобы не забивать систему. LWP разделяют общее адресное пространство и другие ресурсы с другими LWP внутри одного процесса.



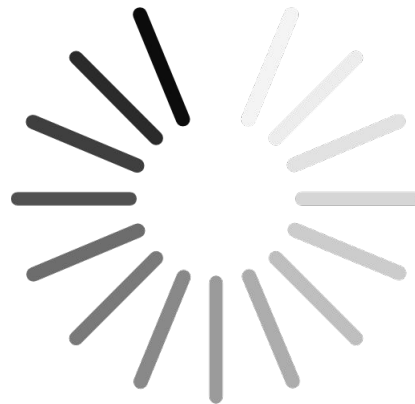


10. Вам нужно запустить скрипт каждые 20 секунд. Опишите какими способами можно это сделать и как бы вы это реализовали?

1. Использовать Cron

2. Использовать Systemd

3. Использовать RabbitMQ





**11. Вам нужно создать демона на PHP.
Какими способами вы будете это
делать?**

1. nohup и &

2. setsid

3. PHP-скрипт для демонизации

4. supervisor





12. Как реализовать уникальность демона?

1. Хэштегом

2. Использовать идентификатор процесса (PID)

3. Использовать файл блокировки


4. Никак





Практика





Представьте, что вы работаете разработчиком в компании мечты.

Тимлид озвучил, что устал рассылать своей команде напоминания о встречах и дедлайнах. Он мечтает освободиться от этой рутины и заняться действительно важными делами.

Поэтому тимлид поручил вам задание:
разработать корпоративный бот-напоминалку

Бот-напоминалка станет полезным инструментом для автоматизации напоминаний о важных событиях, задачах или встречах.

Для вас – это шанс и отличный способ показать свои способности и навыки!
Конечно, остальная команда пыталась вас остановить, но вы были быстрее.
Ну что, приступаем?





Дорогие студенты!

Все практические задания мы будем выполнять в IDE.
Вы можете использовать бесплатный **Visual Studio Code**:

[\[Гайд по VS code\]](#)

или

[PhpStorm](#), но у нее есть ограничение:

бесплатная пробная версия **дается на 30 дней.**

Пожалуйста, учтите этот момент



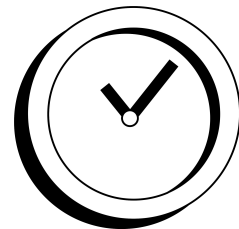


Задание 1. Разбор ТЗ с тимлидом

От тимлида к вам пришло задание разработать собственный бот-напоминалку и готовить для него скрипт на сервере.

Суть бота: вы можете написать в бот и попросить его отправлять вам заданное сообщение в заданный период времени.

Пример бота:



15 мин.

21.06.23 Bot: /start
21.06.23 Bot: Укажите название события
21.06.23 Пользователь: Напоминание о очередной еженедельной встрече по поводу цвета кнопок на сайте
21.06.23 Bot: Укажите ID пользователя
21.06.23 Пользователь: 78479879843
21.06.23 Bot: Укажите текст напоминания
21.06.23 Пользователь: Очень важная встреча по поводу кнопок на сайте пройдет в 14:00. Очень важно, чтобы ты был!
21.06.23 Bot: Укажите в какие дни Вам нужно отправлять сообщения
21.06.23 Пользователь: 0 0 * * 1
21.06.23 Bot: Я записал Ваше событие. Для нового события введите / start
01.06.24 Bot: Очень важная встреча по поводу кнопок на сайте пройдет в 14:00. Очень важно, чтобы ты был!

16:22 ✓✓

В качестве учебного примера мы возьмем бытовые запросы, так как наша цель на данный момент – научиться!

Впоследствии вы можете модернизировать и реализовать бота под иные нужды.



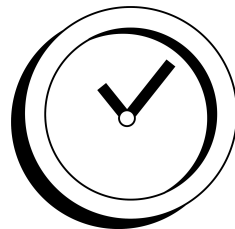
Задание 1. Разбор ТЗ с тимлидом

Так, а с чего начать? Вы наверняка задались этим вопросом!

Курировать начинает тимлид, поэтому задачей тимлида является **определить с командой:**

1. Порядок действий по разработке
2. Технологии, которые будем использовать
3. Структуру БД, которую будем использовать в проекте

Это 3 глобальные задачи, которые предстоит выполнить. Будет неплохо воспользоваться логикой деструктуризации, то есть разбить их на подзадачи (маленькие составляющие)



15 мин.



Также нужно установить инструменты:

- 1) Виртуальная машина Virtualbox с Linux Mint и всем необходимым инструментарием для прохождения курса

[\[ЗАГРУЗИТЬ\]](#)

Пароль: password

[\[ИНСТРУКЦИЯ\]](#)





Также нужно установить инструменты:

- 2) [Установленный php8.0+](#) (установлен на VM из 1го пункта)
- 3) `htop apt install htop` (установлен на VM из 1го пункта)
- 4) [supervisor apt install supervisor](#) (установлен на VM из 1го пункта)
- 5) [SQLite apt install sqlite3](#) (установлен на VM из 1го пункта)





Описание консольного микрофреймворка





Задание 2. Реализация сохранения правил

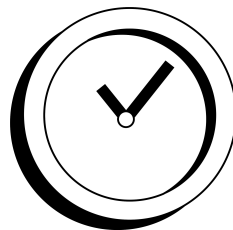
На вход подается строка:

```
php runner -c save_event --name 'Имя события' --receiver 'Айди получателя, пока  
любой' --text 'Текст напоминания' --cron '* * * * *'
```

Нужно реализовать логику обработки этого сообщения и сохранения его в базу данных.

Для получение атрибутов командной строки рекомендуется использовать **getopt**.

Вы можете воспользоваться заготовкой, но я вам рекомендую проделать задание самостоятельно.



15 мин.

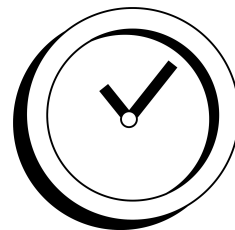


Задание 3. Реализовать скрипт, который проверяет, есть ли задания на отправку в эту минуту, и делает нужные действия с использованием crontab

Нужно реализовать скрипт-обработчик заданий на отправку в мессенджер.

Для этого нужно получить все задания из базы данных и сравнить значения времени запуска с текущими минутой, часом, днем, месяцем и днем недели.

Если значения совпадают, то отправляется сообщение через **EventSender**. Пока мы не приняли решение, с каким мессенджером мы работаем, поэтому EventSender выглядит так:



15 мин.

```
<?php
namespace App\EventSender;

class EventSender
{
    public function sendMessage(string $receiver, string $message)
    {
        echo date('d.m.y H:i') . " Я отправил сообщение $message получателю с id $receiver\n";
    }
}
```



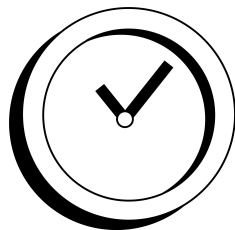
Задание 4. Реализация демона, который отправляет напоминания с использованием `supervisorctl`

Сделать то же самое, что и в задании № 3, но при этом важно, чтобы скрипт запускался как демон, то есть работал бесконечно.

Для этого достаточно сделать бесконечный цикл **while**, из которого потом мы будем вызывать, созданный нами **HandleEventsCommand**.

При этом продумайте механизм обработки HUP сигналов!

Если принят сигнал SIGTERM, SIGINT или SIGHUP, то скрипт должен сохранять текущие обрабатываемые минуту, час, день, месяц и день недели, которые потом подхватит и обработает при новом запуске.



20 мин.



Домашнее задание

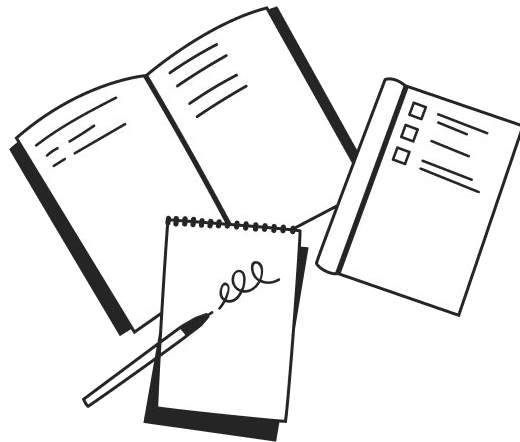




Домашнее задание

Задание

1. Все задания с практики должны быть выполнены и загружены в репозиторий. В качестве выполненного домашнего задания присылайте ссылку на pull request. Рекомендуется [github](#), но можно использовать любой. Помните, что это ваш будущий проект, который вы будете защищать перед будущим работодателем, поэтому относитесь к нему соответствующе.
2. Пришлите содержимое вашего crontab сообщением (вывод команды crontab -l)
3. Опишите результат взаимодействия с systemctl. Как вы считаете в каких задачах использование демонов лучше, чем использование crontab?
4. ✱ Запустить демон с помощью systemd, вместо systemctl. Допустимо написать только worker.service файл.



На проверку отправляйте:

- ссылку на pull request в вашем репозитории с домашним заданием

✱ – дополнительное задание 💪



Освежаем знания по [Git](#)










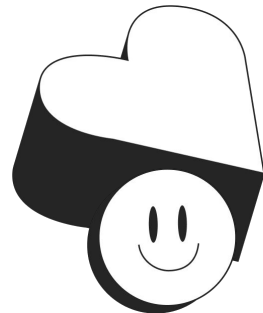
Подведем итоги





Подведение итогов

-  сделали первый и уверенный шаг в подготовке к собеседованию, потренировались в ответах на типичных вопросах
-  научились работать в команде, понимать требования и выполнять задачи, поставленные тимлидом
-  научились разбивать задачу на более мелкие подзадачи, чтобы легче было управлять проектом и следить за прогрессом
-  сделали первый и уверенный шаг в создании бота, который будет напоминать о важных событиях или задачах
-  научились создавать демона, который будет выполнять определенные задачи в фоновом режиме





Спасибо за внимание