Привет! Это Хабр Карьера и Хекслет

Вот и первое тестовое, оно самое лёгкое и рассчитано на уровень стажёра бэкенд-разработчика. Для выполнения тестового вам могут пригодиться сразу три курса:

- Python: Списки
- Python: Функции
- Python: Деревья

Попробуйте сначала справиться самостоятельно, а если где-то застрянете — обращайтесь к курсам. Если понимаете, что это задание пока слишком сложное, выделите время на изучение курсов до 15 декабря — пока доступ к курсам Хекслета открыт бесплатно.

📌 Советуем завести отдельный док, где вы будете собирать результаты каждого задания.

Ну что, поехали!

Описание задания

Реализуйте симуляцию распределённой системы, в которой несколько процессов выполняют задания. Это не реальная распределённая система серверов, а её упрощённая модель, демонстрирующая работу с алгоритмами распределения и управления очередями.

Программа должна:

1. Управлять очередью заданий, добавляя и извлекая задания для выполнения.
2. Распределять задания между процессами (симулирующими сервера) оптимальным образом, исходя из заданных условий, например:

- Процессы имеют разную производительность (время выполнения одного задания).
- Задания различаются по сложности и длительности выполнения.

3. Использовать подходящую модель взаимодействия между процессами:

- IPC (межпроцессовое взаимодействие).
- любой другой вариант взаимодействия процессов, но без использования готовых брокеров сообщений (как kafka или rabbitmq).

4. Процессы/сервера должны создаваться один раз, а не каждый раз под каждое задание.

Функциональность

1. **Инициализация системы**:

- Пользователь задает количество серверов (например, 3).
- Каждый сервер может одновременно выполнять только одно задание.

2. **Распределение заданий**:

- Алгоритм выбирает сервер с минимальной загрузкой (общее время выполнения заданий на сервере).

- Если все серверы заняты, задание остается в очереди и ждет освобождения сервера.

3. **Обработка заданий**:

- Сервер выполняет задание, после чего освобождается для следующего задания.

4. **Просмотр состояния**:

- Программа отображает состояние серверов: выполняемое задание и оставшееся время выполнения.
- Программа отображает задания в очереди.

Пример работы программы

1. Инициализация:

```
Добро пожаловать в симулятор распределенной системы.
Введите количество серверов: 3
Состояние серверов:
Сервер 1: пусто
Сервер 2: пусто
Сервер 3: пусто
```

2. Добавление заданий:

```
Команда: добавить 5
Задание с 5 секундами выполнения направлено на Сервер 1.
Команда: добавить 3
Задание с 3 секундами выполнения направлено на Сервер 2.
Команда: добавить 7
Задание с 7 секундами выполнения направлено на Сервер 3.
Команда: добавить 2
Задание с 2 секундами выполнения добавлено в очередь.
```

3. Обработка заданий:

```
Состояние серверов:
Сервер 1: выполняет задание (осталось 4 сек.)
Сервер 2: выполняет задание (осталось 2 сек.)
Сервер 3: выполняет задание (осталось 6 сек.)
Очередь заданий: [2]

Обработка:
- Сервер 2 освобождается, задание из очереди направлено на Сервер 2.
- Сервер 1 завершает выполнение.
```

4. Просмотр состояния:

```
Состояние серверов:
Сервер 1: пусто
Сервер 2: выполняет задание (осталось 1 сек.)
Сервер 3: выполняет задание (осталось 3 сек.)
Очередь заданий: нет.
```

Алгоритм распределения

1. **Очереди серверов**:

- Каждому серверу соответствует очередь текущих заданий.

2. **Алгоритм выбора сервера**:

- Рассчитать суммарное время выполнения всех заданий на каждом сервере.
- Направить задание на сервер с минимальным временем.

3. **Обработка заданий**:

- Уменьшать оставшееся время выполнения текущего задания на каждом сервере.
- Удалять задание из очереди сервера, когда оно выполнено.

Критерии оценки

1. Работоспособность — программа корректно распределяет и обрабатывает задания.
2. Алгоритм — умение работать с очередями и оптимизировать распределение.
3. Качество кода — чистота, читаемость, комментарии.

Требования к результату

1. Загрузите код на GitHub или другой репозиторий.
2. Добавьте README-файл с:

- Инструкцией по запуску.
- Примером работы программы.

Расширения (по желанию)

Добавьте приоритеты для заданий, чтобы более важные задания обрабатывались раньше.