ТЕХНОАТОМ

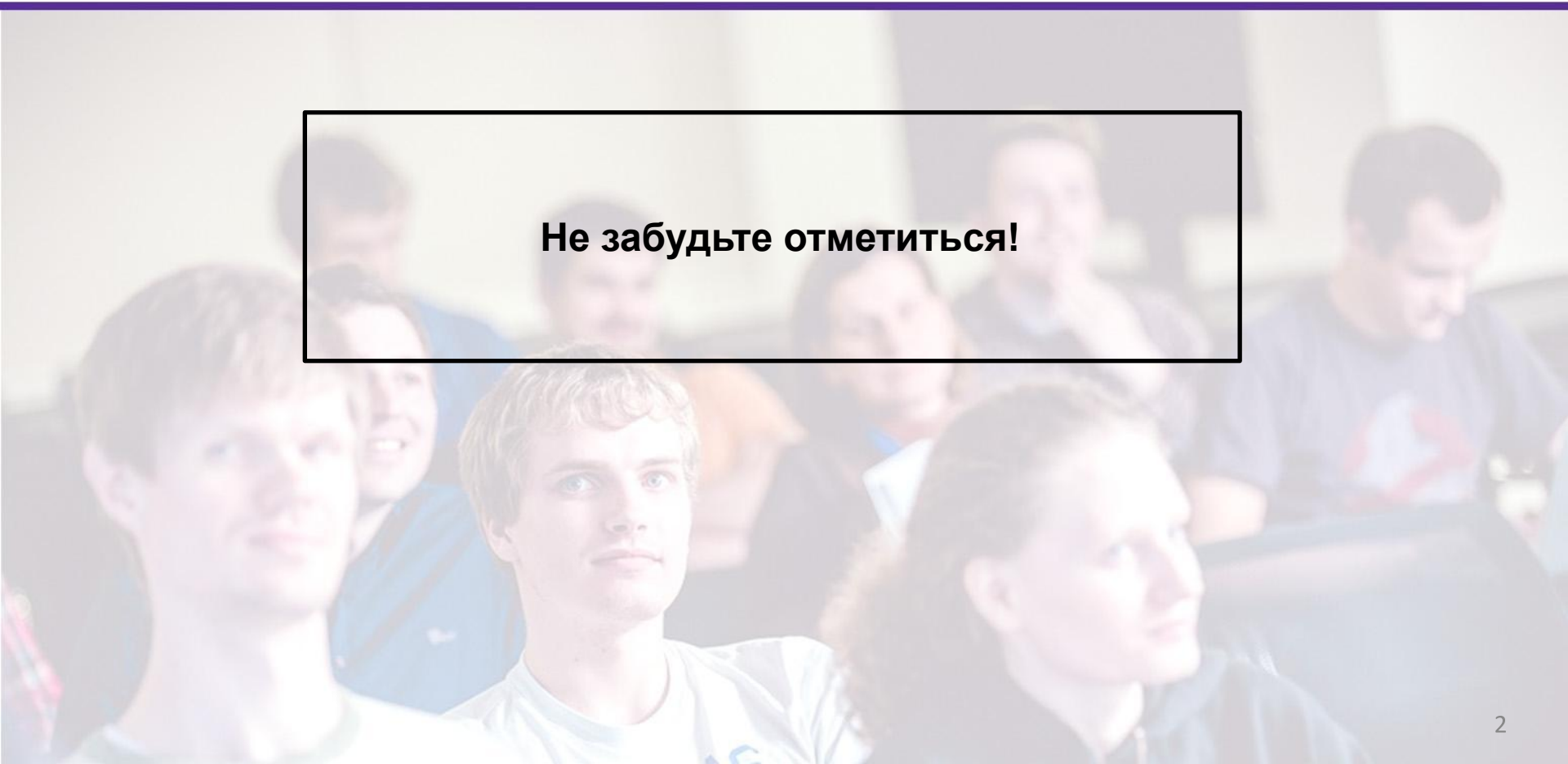
Потоки **GIL, Threads, multiprocessing, IPC,** механизмы синхронизации

Урок восьмой

Антон Кухтичев

Посещаемость

ТЕХНОАТОМ



Не забудьте отметиться!

Программа на сегодня

ТЕХНОАТОМ

- Потоки GIL
- Threads
- multiprocessing
- IPC
- Механизмы синхронизации

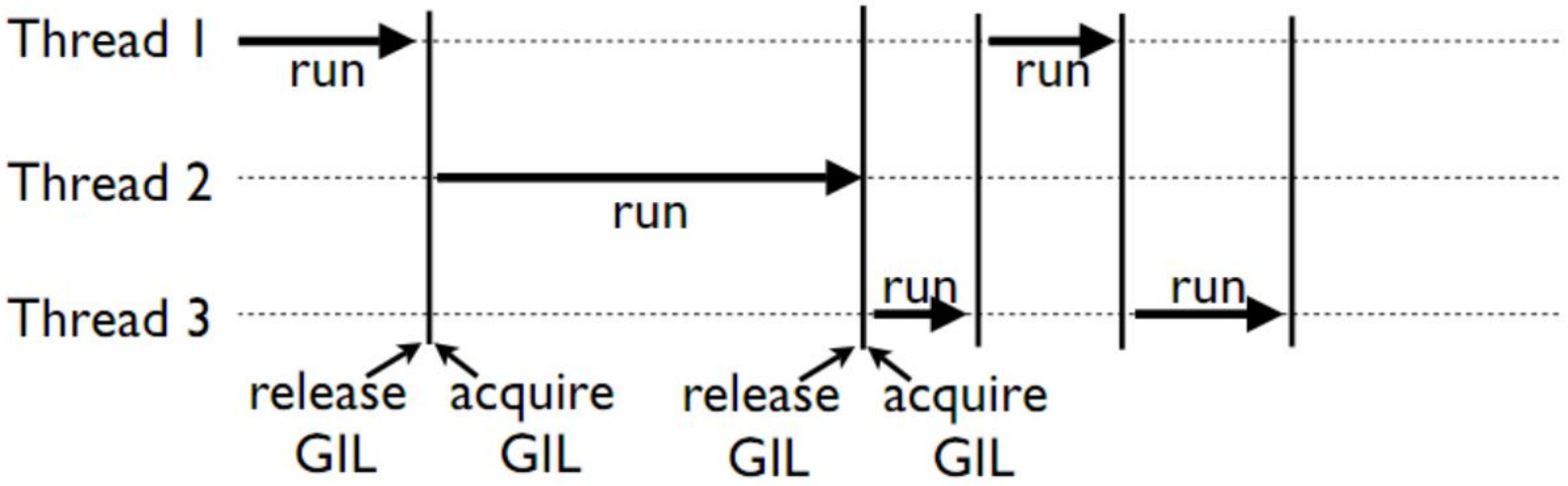
TEXHOATOM

GIL

Global Interpreter Lock (GIL) — это способ синхронизации потоков, который используется в некоторых интерпретируемых языках программирования.

Global Interpreter Lock

TEXHOATOM



TEXHOATOM

Threads

Поток (thread) — это, сущность операционной системы, процесс выполнения на процессоре набора инструкций, точнее говоря программного кода.

```
from threading import Thread

thread1 = Thread(target=function_name, args=(arg1, arg2,))

class MyThread(threading.Thread):
    def __init__(self, num):
        super().__init__(self, name="threddy" + num)
        self.num = num
    def run(self):
        print ("Thread ", self.num),

thread2 = MyThread("2")
```

TEXHOATOM

multiprocessing

Процесс (process) — не что иное, как некая абстракция, которая инкапсулирует в себе все ресурсы процесса (открытые файлы, файлы отображенные в память) и их дескрипторы, потоки и т.д.

Состоит из:

1. образ машинного кода;
2. область памяти, в которую включается исполняемый код, данные процесса (входные и выходные данные), стек вызовов и куча (для хранения динамически создаваемых данных);
3. дескрипторы операционной системы (например, файловые дескрипторы);
4. состояние процесса.

multiprocessing

TEXHOATOM

```
from multiprocessing import Process
import os

def info(title):
    print(title)
    print('module name:', __name__)
    print('parent process:', os.getppid())
    print('process id:', os.getpid())

def f(name):
    info('function f')
    print('hello', name)

if __name__ == '__main__':
    info('main line')
    p = Process(target=f, args=('bob',))
    p.start()
    p.join()
```

TEXHOATOM

IPC

Из механизмов, предоставляемых ОС и используемых для IPC, можно выделить:

- механизмы обмена сообщениями;
- механизмы синхронизации;
- механизмы разделения памяти;
- механизмы удалённых вызовов (RPC).

Виды межпроцессного взаимодействия

ТЕХНОАТОМ

- Файл
- Сигнал
- Сокет
- Каналы (именованные и неименованные)
- Семафор
- Разделяемая память
- Обмен сообщениями
- Проецируемый в память файл (mmap)
- Очередь сообщений (Message queue)
- Почтовый ящик

Сигнал

ТЕХНОАТОМ

```
import signal
import os
import time

def receive_signal(signalNumber, frame):
    print('Received:', signalNumber)
    raise SystemExit('Exiting')
    return

if __name__ == '__main__':
    signal.signal(signal.SIGUSR1, receive_signal)

    signal.signal(signal.SIGINT, signal.SIG_IGN)

    print('My PID is:', os.getpid())

    signal.pause()
```

Сокет

ТЕХНОАТОМ

```
import socket

# Клиент
client = socket.socket(socket.AF_UNIX, socket.SOCK_DGRAM)
client.connect("/tmp/python_unix_sockets_example")
client.send(x.encode('utf-8'))

# Сервер
server = socket.socket(socket.AF_UNIX, socket.SOCK_DGRAM)
server.bind("/tmp/python_unix_sockets_example")
datagram = server.recv(1024)
```

Каналы

ТЕХНОАТОМ

```
# sender.py
import os

path = "/tmp/my_program.fifo"
os.mkfifo(path)

fifo = open(path, "w")
fifo.write("Hello World!\n")
fifo.close()
```

```
# receiver.py
import os
import sys

path = "/tmp/my_program.fifo"
fifo = open(path, "r")
for line in fifo:
    print(f"Получено: {line}")
fifo.close()
```

Проецируемый в память файл (mmap)

ТЕХНОАТОМ

```
import mmap

with open("hello.txt", "w") as f:
    f.write("Hello Python!\n")

with open("hello.txt", "r+") as f:
    map = mmap.mmap(f.fileno(), 0)
    print(map.readline())  # prints "Hello Python!"
    print(map[:5])  # prints "Hello" have same size
    map[6:] = " world!\n"
    map.seek(0)
    print(map.readline())  # prints "Hello  world!"
    map.close()
```

Встроенные в multiporcessing способы

ТЕХНОАТОМ

- `Array`

```
result = multiprocessing.Array('i', 4)
square_sum = multiprocessing.Value('i')
```

- `Manager`

```
with multiprocessing.Manager() as manager:
    # creating a list in server process memory
    records = manager.list([('Jesse', 10), ('Walter', 9),
('Tuco',9)])
```

- `Queue`

```
q = multiprocessing.Queue()
```

- `Pipe`

```
parent_conn, child_conn = multiprocessing.Pipe()
```

Домашнее задание

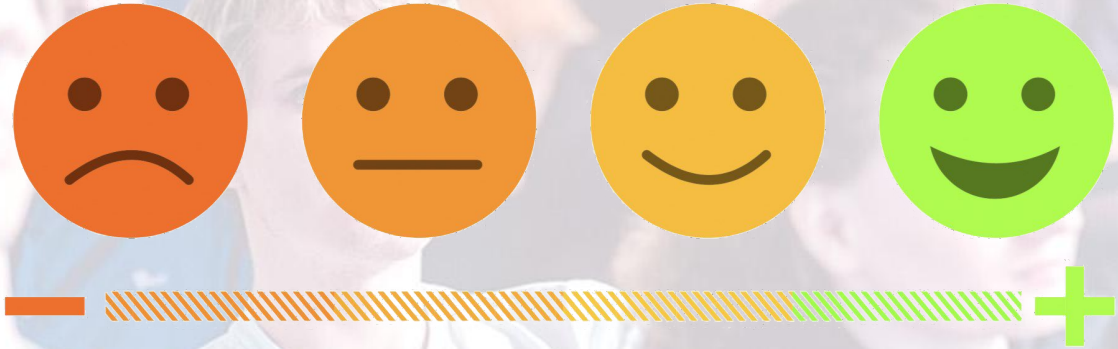
ТЕХНОАТОМ

1. Написать демона master-worker (количество воркеров задаётся в конфиге), принимающий запросы, содержащий URL, и дающий каждому воркеру URL (например по алгоритму round robin), обкачивает его и возвращает содержимое этого url без тегов. Предусмотреть возможность остановить демона при помощи сигнала SIGUSR1, при это печатает в конце количество скаченных URL за всё время работы.

Оценивание занятие

ТЕХНОАТОМ

Не забудьте оценить занятие!



ТЕХНОАТОМ

Спасибо за внимание!

Антон Кухтичев

a.kukhtichev@corp.mail.ru