Лекция 4. Семинар 4. CGI. Процессы

`strace bash -c date` - в этой команде `strace` показывает какие системы вызовы производит наш процесс. В данном случае это `bash`, а `bash` в свою очередь запускает `date`.
После этой команды выведется список системных вызовов, выделим основные по порядку:

- `execve("/usr/bin/bash", ["bash", "c", "date"]...` - первый системный вызов; `strace` начал запускать процесс `bash` разделившись, вместо себя другой исполняемый файл.
- `MAP...` - выделение памяти
- `openat...` - открытие различных файлов
- `execve("/usr/bin/date", ["date"]..` - вызов исполняемого файла `date`

Команды процессов (fork, wait, exec)

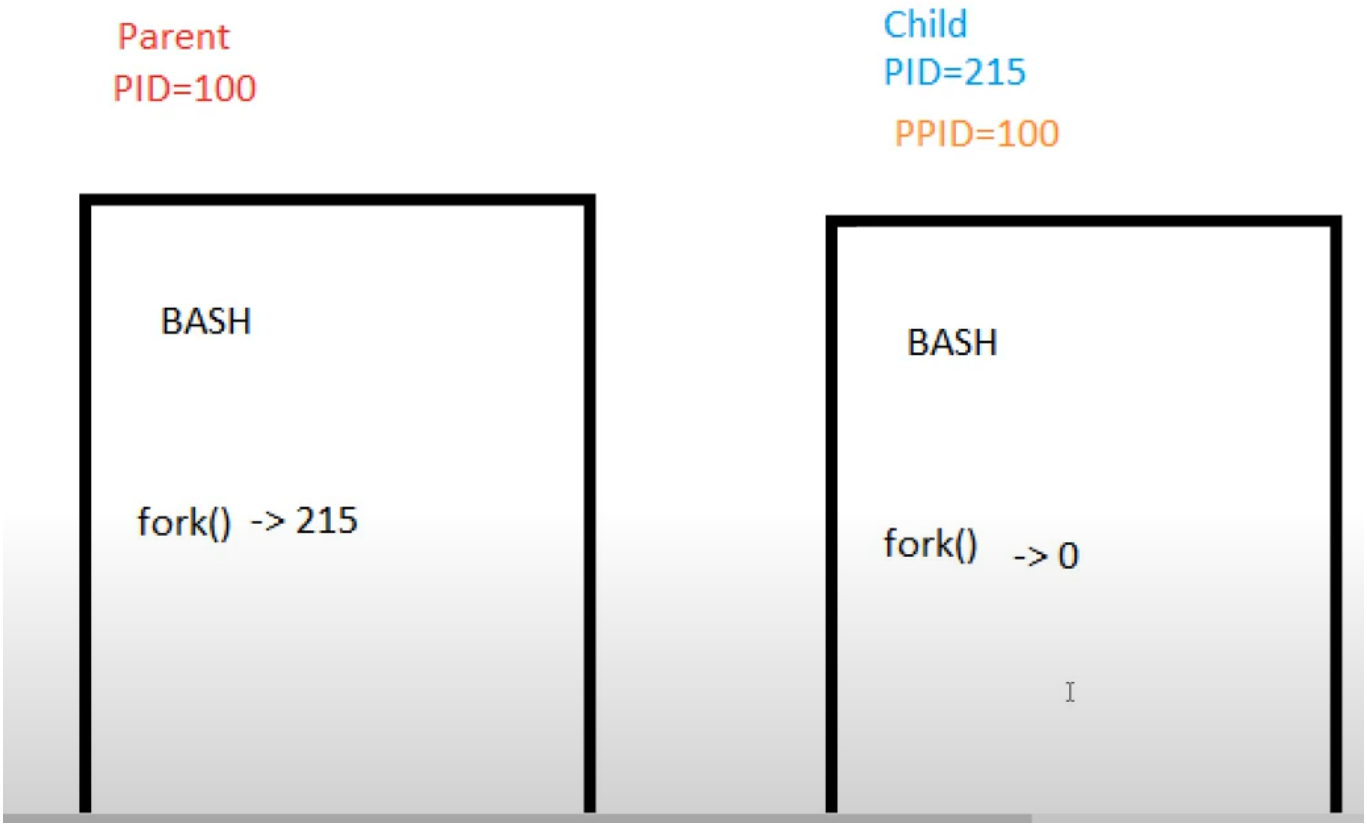
`fork()` - это система вызовов

synopsis

`pid_t fork(void)` - внутри ничего не передаётся, на выходе имеем id процесса.

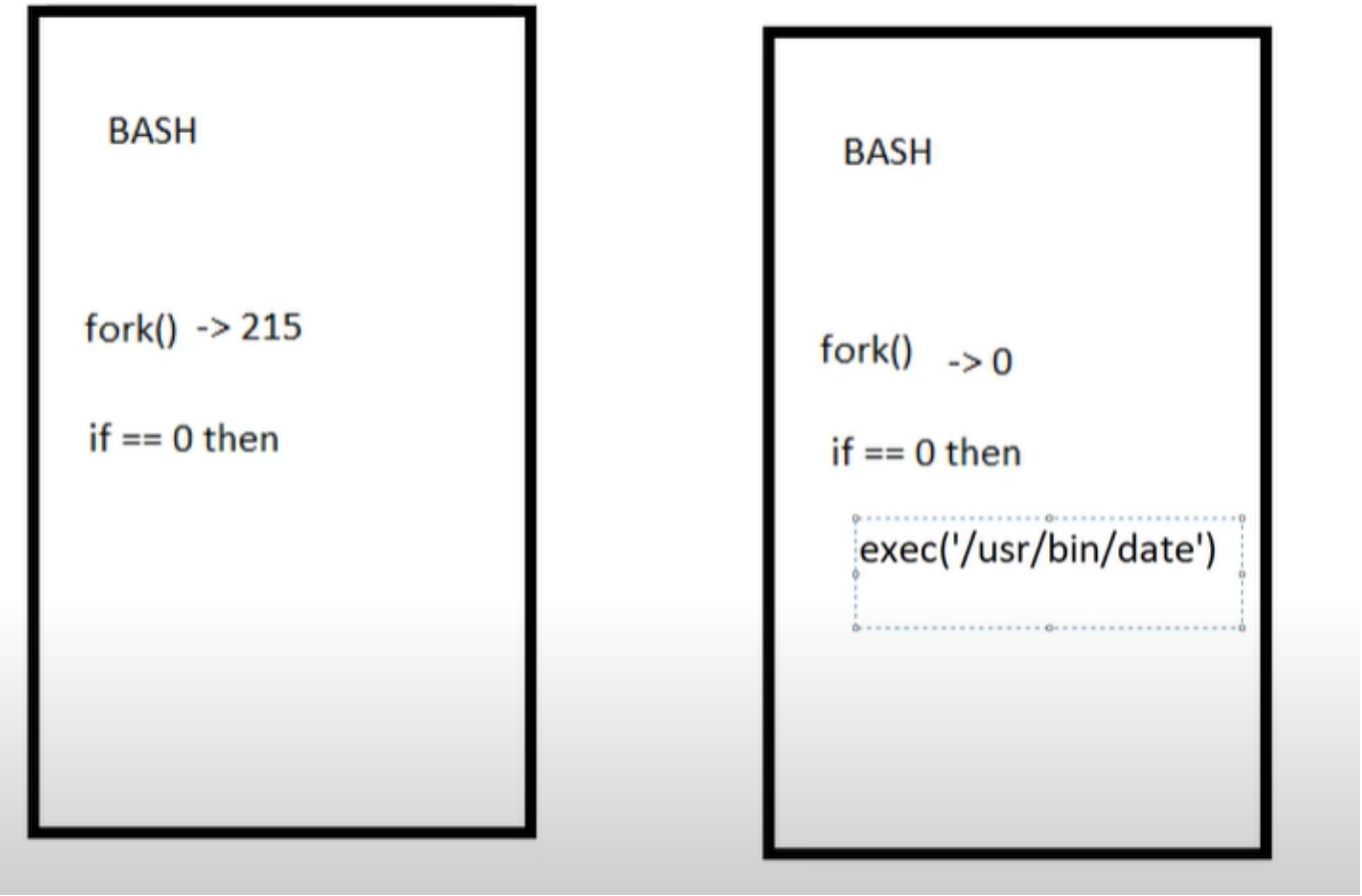
Пример создания процессов

На илюстрации ниже в левом процессе `bash` встретилось `fork()`, которое создало копию процесса. Левый процесс является родительским процессом и имеет **PID = 100**. Копия процесса (справа) имеет **PID = 215** и **PPID = 100 (Parent PID)** и является дочерним процессом. `fork()` - возвращает **PID** дочернего процесса, если дочернего процесса нет, то возвращает 0.



`wait()` - ожидает, пока процесс поменяет состояние. Например, родитель так может считать состояние ребёнка (если он не будет считывать, то состояние у ребёнка **"зомби"**)

`execve()` (одно из модификаций `exec`) - выполняет конкретную программ внутри тела процесса. Запускать *бинарный исполняемый файл* или *скрипт* (надо чтобы он начинался с **шебанга** `#!interpreter` - путь к интерпретатору)



`ps` - показывает *PID* процесов

Права на исполнение файл

`ls -l myfile` - просмотр прав на файл "myfile"
`-rwxr-xr-x` - флажок "x"(от слова "execute" - исполнять) на этих позициях означает, что у каждого есть доступ на исполнение файла
`sudo chmod a-x myfile` - убирает у всех доступ на исполнение файла
Расшифровка:

- `chmod` ("change mode") - меняет права
- `a-x` ("a" сокращение от "all") - убирает у всех права на исполнение файла
- `a+x` - даёт всем права на исполнение файла

Организация каналов между процессами

> **Канал** - самый простой и легкореализуемый способ связи процессов (можно передавать много информации и не нужно синхронизировать).

В каталоге `proc/{PID}/fd` - можно увидеть файлы-ссылки
Они могут ссылаться на `/dev/pts/0` - устройство псевдотерминала, где мы

работаем.
У каждого процесса есть 3 дискриптора при открытых файлов.
Назначение ссылок (цифрами указан номер дескриптора):
`stdin(0) -> proc -> stdout(1)`
`stdin(0) -> proc -> stderr(2)`
`stdin` - стандартный ввод
`stdout` - стандартный вывод для сообщений
`stderr` - стандартный выход для ошибок

Пример использования канала

`cat < /dev/zero | cat > /dev/null &` - `cat < /dev/zero` читает нули, `|` означает, что стандартный выход процесса `cat < /dev/zero` будет соединен со стандартным входом `cat > /dev/null` - специльное устройство, куда можно скидывать мусор. `&` - запускает всю конструкцию в фоновом режиме.
В каталоге `proc/{PID_1}/fd` и `proc/{PID_2}/fd` можно увидеть, что они соединены каналом `pipe` (у первого процесса это 1-ый дескриптор - вывод, у втрого это 0-ой дескриптор - ввод).

> Любые два дочерних процесса, у которых родитель общий можно соединить каналом.

Окружение процесса

Путь к нему `/proc/{PID}/environ`.
Содержимое:

- `^@` -разделитель строк
- слева от `=` переменная окружения
- справа от `=` его значение

CGI

Передает:

1. Переменные
2. Сообщение
3. Метод

Настройка :

1. Заходим в каталог доступных модулей apache `cd /etc/apache2/mods-avialable/`
2. Включаем модуль `sudo a2enmod cgi`
3. Редактируем файл `sudo nano /etc/apache2/sites-enabled/000-default.conf`
4. После `DocumentRoot /var/www/html` добавляем `ScriptAlias "/cgi-bin/" "/var/www/cgi-bin/"` - каталог, где будут *cgi-скрипты*. Сохраняем файл
5. Запустим apache `sudo service apache2 restart`
6. Проверим статус apache `sudo service apache2 status`. Если видите `active (running)` - значит работает.
7. Переходим в корень `cd` и создаем каталог `sudo mkdir /var/www/cgi-bin`. Проверяем, что все могут заходить в каталог `ls -ld /var/www/cgi-bin`. Должно быть 3 флажка `r`
8. Заходим `cd /var/www/cgi-bin` и создаем `sudo nano test.py`
9. Добавляем *шебанг* `!/usr/bin/python3` (Важно! Путь к интерпретатору может быть другой, посмотрите его через команду `whereis`)
10. Пишем Содержимое CGI-скрипта
11. Делаем файл исполняемым `chmod a+x test.py`
12. Проверяем, что он запускается `./test.py`
13. Выполняем запрос, обратившись к скрипту `curl http://localhost/cgi-bin/test.py`. Можно добавить `-i` после `curl`, чтобы увидеть весь протокол. В конце надо добавить `--data "Text Message"`, если мы хотим передавать данные. Подробно Пример 2 Отправка текста. Самый сложный описан: Пример 5 Узнаем погоду https openweathermap org current
14. Если ошибка, можно посмотреть логи `cd /var/log/apach2` и `less error.log`. Смотрим в конец.

Содержимое CGI-скрипта

Пример 1. Какие переменные окружения есть

```
!usr/bin/python3
import os
print('Content-Type: text/plain\n\n')
print(os.environ)
```

Результат:

```
environ({'HTTP_HOST': 'localhost', 'HTTP_USER_AGENT': 'curl/
7.68.0', 'HTTP_ACCEPT': '*/*', 'PATH': '/usr/local/sbin:/usr
/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin', 'SERVER
_SIGNATURE': '<address>Apache/2.4.41 (Ubuntu) Server at loca
lhost Port 80</address>\n', 'SERVER_SOFTWARE': 'Apache/2.4.4
1 (Ubuntu)', 'SERVER_NAME': 'localhost', 'SERVER_ADDR': '127
.0.0.1', 'SERVER_PORT': '80', 'REMOTE_ADDR': '127.0.0.1', 'D
OCUMENT_ROOT': '/var/www/html', 'REQUEST_SCHEME': 'http', 'C
ONTEXT_PREFIX': '/cgi-bin/', 'CONTEXT_DOCUMENT_ROOT': '/var/
www/cgi-bin/', 'SERVER_ADMIN': 'webmaster@localhost', 'SCRIP
T_FILENAME': '/var/www/cgi-bin/test.py', 'REMOTE_PORT': '330
94', 'GATEWAY_INTERFACE': 'CGI/1.1', 'SERVER_PROTOCOL': 'HTT
P/1.1', 'REQUEST_METHOD': 'GET', 'QUERY_STRING': '', 'REQUES
T_URI': '/cgi-bin/test.py', 'SCRIPT_NAME': '/cgi-bin/test.py
```

Пример 2. Отправка текста

```
!usr/bin/python3
body = input()

print('Content-Type: text/plain\n\n')

print('Start...')
print(body)
print('...End\n')
```

- Запуск этого скрипта, чтобы проверить:
  `echo "Test message" | ./test.py`
- Используя http-запрос запускаем:
  `curl -i http://localhost/cgi-bin/test.py --data "Test message"`

Пример 2.1 Отправка текствого файла

- Создаем `nano out.txt` и напишем внутри

```
Send File
Send file
```

- Запускаем
  `curl -i http://localhost/cgi-bin/test.py --data "@out.txt"`

Пример 2.2 Отправка бинарного файла

Первый способ без контроля точного размера:

- Сгенерируем с помощью `cat < /dev/urandom > out.bin`
- Нажимаем Ctrl + C
- Смотрим размер с помощью `ls -lh out.bin`
  Второй способ с контролем размера:
- Сгенерируем с помощью `dd if=/dev/urandom of=out.bin count=10` (`dd` - конвертирует и копирует файл, `if` - input file, `of` - output file, `count` - количество записей (512 байт - 1 запись))
- Смотрим размер с помощью `ls -lh out.bin`: 5120B
- `nano out.bin`
- Запускаем
  `curl -i http://localhost/cgi-bin/test.py --data "@out.bin"`

Пример 3. Ввод для обработки больших данных

stdin
buffer

```
!usr/bin/python3
import sys
body = sys.stdin.buffer.read()

print('Content-Type: text/plain\n\n')

print('Start...')
print(body)
print('...End\n')
```

Пример 4. HTML-страница с формой

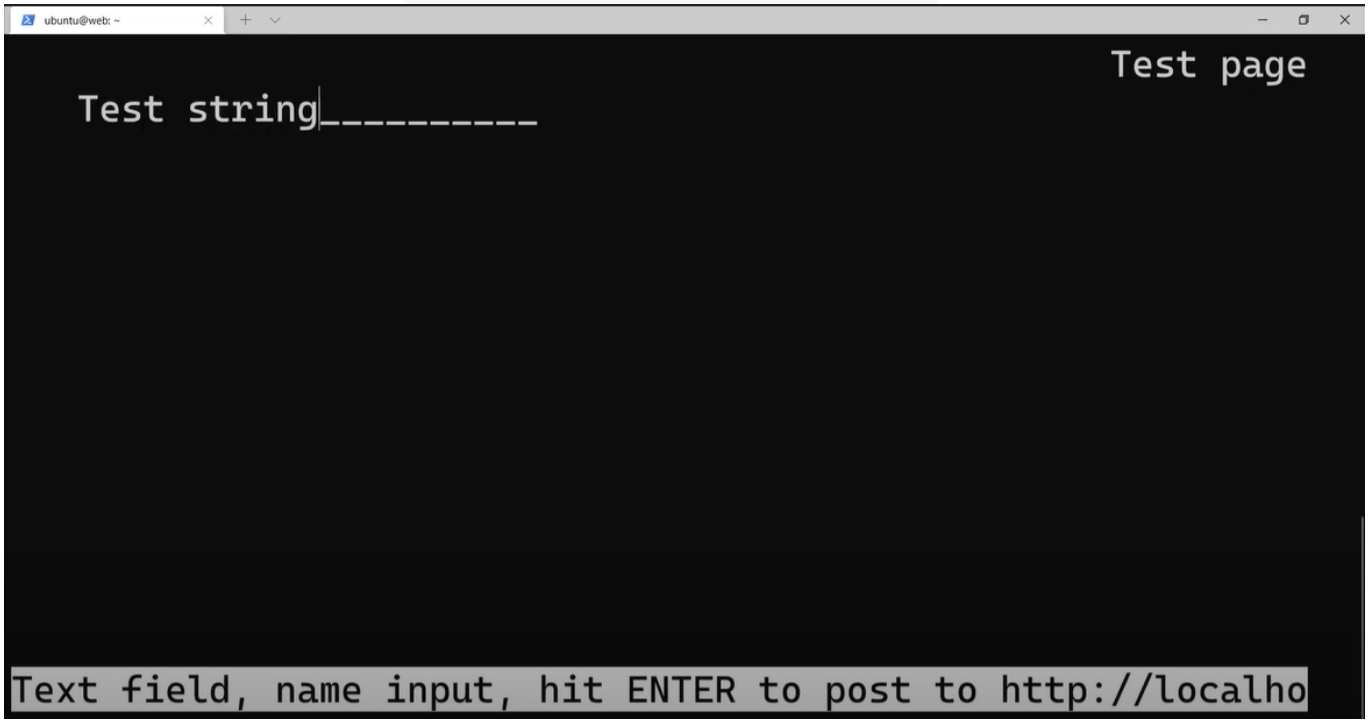
1. Создадим html-страницу `cd /var/www/html/` и `sudo nano test.html`

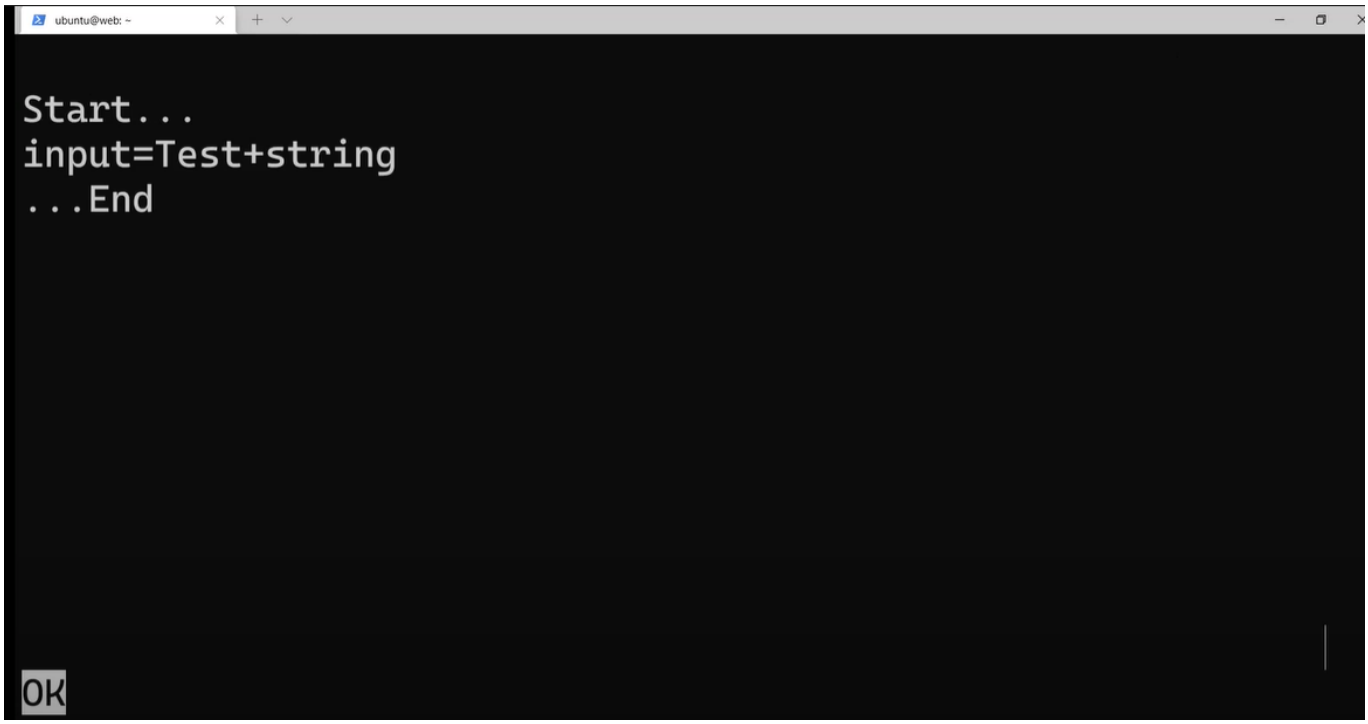
```
<!DOCTYPE html>

<html>
<head><title>Test page</title></head>
<body>
```

```
<form method="post" action="/cgi-bin/test.py">
<input type="text" name="input" value="Type here">
</form>
</body>
</html>
```

В `action` *относительный путь*, можно поменять на *абсолютный*: `http://localhost/cgi-bin/test.py`

2. Сохраняем и смотрим `links http://localhost/test.html`





> Можно заметить, что вместо пробела стоит "+". Так как был применен такой метод кодирования.
> 3. Поменяем python-скрипт, добавив библиотеку cgi

```
!usr/bin/python3
import sys
import cgi
form = cgi.FieldStorage()

print('Content-Type: text/plain\n\n')

print('Start...')
print(form["name"].value)
print('...End\n')
```

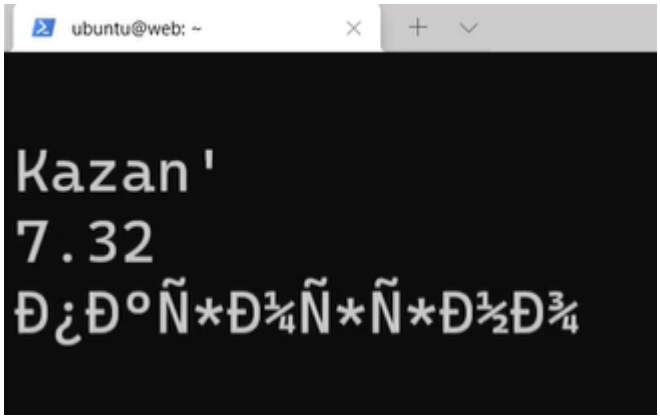
Пример 5. Узнаем погоду

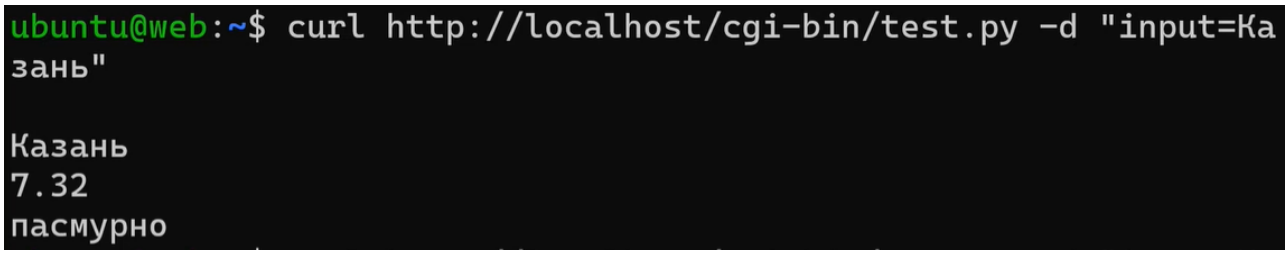
1. Оставляем html-страницу из Пример 4 HTML-страница с формой
2. Правим *cgi-скрипт*:

```
!usr/bin/python3
import sys
import cgi
import requests

print('Content-Type: text/plain')
print()

W_URL='https://api.openweathermap.org/data/2.5/weather'


form = cgi.FieldStorage()

if "input" in form:
    city = form["input"].value
    w_params = {'q': city,
    'appid': '{API KEY}',
    'units': 'metric',
    'lang': 'ru'}
```

```
    w_resp = requests.get(W_URL, params = w_params).json()
    desc = w_resp['weather'][0]['description']
    temp = w_resp['main']['temp']

print(city)
print(temp)
print(desc)
```

- Запускаем скрипт `links http://localhost/test.html`
  Результат:
  

  Проблемы с кодировкой :(
- Можно посмотреть другой командой `curl http://localhost/cgi-bin/test.py -d "input=Казань" -X "POST"` - имитация команды `links`
  Результат:
  

- Чтобы потестировать через `./test.py`: можно закомментировать место, где мы получаем данные из формы или подставить туда что-то.

Интересные ссылки

Стандарт иерархии файловой системы:
FHS