# Задание 1

### Системные вызовы при старте httpd:

% time seconds usecs/call calls errors syscall

------ ----------- ----------- --------- --------- ----------------

0.00 0.000000 0 2 read

0.00 0.000000 0 1 write

0.00 0.000000 0 3 open

0.00 0.000000 0 3 close

0.00 0.000000 0 4 fstat

0.00 0.000000 0 11 mmap

0.00 0.000000 0 6 mprotect

0.00 0.000000 0 1 munmap

0.00 0.000000 0 1 brk

0.00 0.000000 0 2 rt\_sigaction

0.00 0.000000 0 1 rt\_sigprocmask

0.00 0.000000 0 1 1 access

0.00 0.000000 0 1 socket

0.00 0.000000 0 1 bind

0.00 0.000000 0 1 listen

0.00 0.000000 0 1 execve

0.00 0.000000 0 1 getrlimit

0.00 0.000000 0 1 arch\_prctl

0.00 0.000000 0 1 set\_tid\_address

0.00 0.000000 0 1 set\_robust\_list

------ ----------- ----------- --------- --------- ----------------

100.00 0.000000 44 1 total

**read** - чтение из файла

**write** - запись в файл

**open** - открытие файла

**close** – закрытие файла

**fstat** – статус файла

**mmap** – отмапить файл с устройства в память

**mprotect** – контроль доступа к памяти

**munmap** – размапить файл из памяти

**brk** - устанавливает конец сегмента данных в значение, указанное в аргументе

**rt\_sigaction** - получает и изменяет обработчик сигнала

**rt\_sigprocmask** - проверка и изменение блокировки сигналов

**access** - проверяет имеет ли процесс доступ к файлу

**socket** - создаёт конечную точку соединения и возвращает файловый дескриптор, указывающий на эту точку

**bind** - привязывает имя к сокету

**listen** - слушать соединения на сокете

**execve** - выполняет программу

**getrlimit** - получает ограничения использования ресурсов

**arch\_prctl** – устанавливает состояние процесса или потока, в зависимости от архитектуры

**set\_tid\_address** - устанавливает указатель идентификатора потока

**set\_robust\_list** - возвращает/назначает список надёжных фьютексов

**Системные вызовы при старте и одном запросе:**

% time seconds usecs/call calls errors syscall

------ ----------- ----------- --------- --------- ----------------

91.11 0.001650 4 438 recvfrom

3.48 0.000063 4 15 sendto

1.38 0.000025 5 5 close

0.88 0.000016 4 4 open

0.77 0.000014 7 2 munmap

0.66 0.000012 3 4 brk

0.61 0.000011 3 4 read

0.55 0.000010 5 2 accept

0.39 0.000007 1 12 mmap

0.17 0.000003 1 5 fstat

0.00 0.000000 0 1 write

0.00 0.000000 0 1 stat

0.00 0.000000 0 6 mprotect

0.00 0.000000 0 2 rt\_sigaction

0.00 0.000000 0 1 rt\_sigprocmask

0.00 0.000000 0 1 1 access

0.00 0.000000 0 1 socket

0.00 0.000000 0 1 bind

0.00 0.000000 0 1 listen

0.00 0.000000 0 1 execve

0.00 0.000000 0 1 getrlimit

0.00 0.000000 0 1 arch\_prctl

0.00 0.000000 0 1 set\_tid\_address

0.00 0.000000 0 1 set\_robust\_list

------ ----------- ----------- --------- --------- ----------------

100.00 0.001811 511 1 total

**recvfrom** - принимает сообщение из сокета

**sendto** - отправляет сообщения в сокет

**accept** - принять соединение на сокете

**stat** - считывает состояние файла

Системные вызовы от старта до введения данных в форме и перехода на страницу с результатом (запускался отдельный процесс)

% time seconds usecs/call calls errors syscall

------ ----------- ----------- --------- --------- ----------------

42.06 0.004297 3 1541 recvfrom

30.18 0.003084 7 455 sendto

13.38 0.001367 3 479 read

2.20 0.000225 5 42 11 open

2.07 0.000211 4 48 mmap

1.60 0.000163 4 40 close

1.45 0.000148 5 28 mprotect

1.23 0.000126 1 132 116 stat

0.97 0.000099 33 3 accept

0.69 0.000070 1 50 brk

0.63 0.000064 4 15 write

0.50 0.000051 26 2 dup2

0.42 0.000043 1 36 3 lseek

0.36 0.000037 2 18 fstat

0.35 0.000036 9 4 munmap

0.35 0.000036 0 73 1 rt\_sigaction

0.28 0.000029 29 1 wait4

0.23 0.000023 23 1 socket

0.22 0.000022 22 1 readlink

0.20 0.000020 10 2 2 access

0.18 0.000018 9 2 pipe

0.06 0.000006 6 1 bind

0.05 0.000005 0 16 15 ioctl

0.05 0.000005 5 1 listen

0.05 0.000005 3 2 execve

0.04 0.000004 2 2 getrlimit

0.04 0.000004 2 2 arch\_prctl

0.04 0.000004 2 2 set\_tid\_address

0.03 0.000003 2 2 rt\_sigprocmask

0.03 0.000003 1 3 getegid

0.03 0.000003 1 3 set\_robust\_list

0.02 0.000002 2 1 clone

0.02 0.000002 1 3 getuid

0.01 0.000001 0 3 getgid

0.01 0.000001 0 4 geteuid

0.00 0.000000 0 1 fcntl

------ ----------- ----------- --------- --------- ----------------

100.00 0.010217 3019 148 total

**dup2** - создаёт копию файлового дескриптора oldfd, используя для нового дескриптора самый маленький свободный номер файлового дескриптора

**lseek** - изменяет файловое смещение, используемое при чтении/записи

**wait4** - ожидает смену состояния в стиле BSD

**readlink** - считывает значение символьной ссылки

**pipe** - создаёт однонаправленный канал данных, который можно использовать для взаимодействия между процессами

**ioctl** - изменяет базовые параметры устройства, представленного в виде специального файла

**getegid** - возвращает эффективный идентификатор группы вызывающего процесса.

**clone** - создать процесс-потомок

**getuid** - возвращает фактический идентификатор (ID) пользователя вызывающего процесса.

**getgid** - возвращает действительный идентификатор группы вызывающего процесса.

**geteuid** - возвращает эффективный идентификатор пользователя вызывающего процесса.

**fcntl** - работа с файловым дескриптором

# Задание 2

### Функции tinyhttpd

int **main**(void) - точка входа, содержит бесконечный цикл, при получении запроса вызывается accept\_request

void **unimplemented**(int client) – генерирует http ответ с кодом ‘501 Method Not Implemented’

int **startup**(u\_short \*port) – начинает слушать порт

void **serve\_file**(int client, const char \*filename) – высылает файл (статику)

void **not\_found**(int client) - генерирует http ответ с кодом ‘404 NOT FOUND’

void **headers**(int client, const char \*filename) – возвращает HTTP заголовок с информацией о высылаемом файле

int **get\_line**(int sock, char \*buf, int size) – получает строку из сокета

void **execute\_cgi**(int client, const char \*path,

const char \*method, const char \*query\_string) – выполняет cgi скрипт в дочернем процессе

void **error\_die**(const char \*sc) – выводит ошибку и завершает выполнение

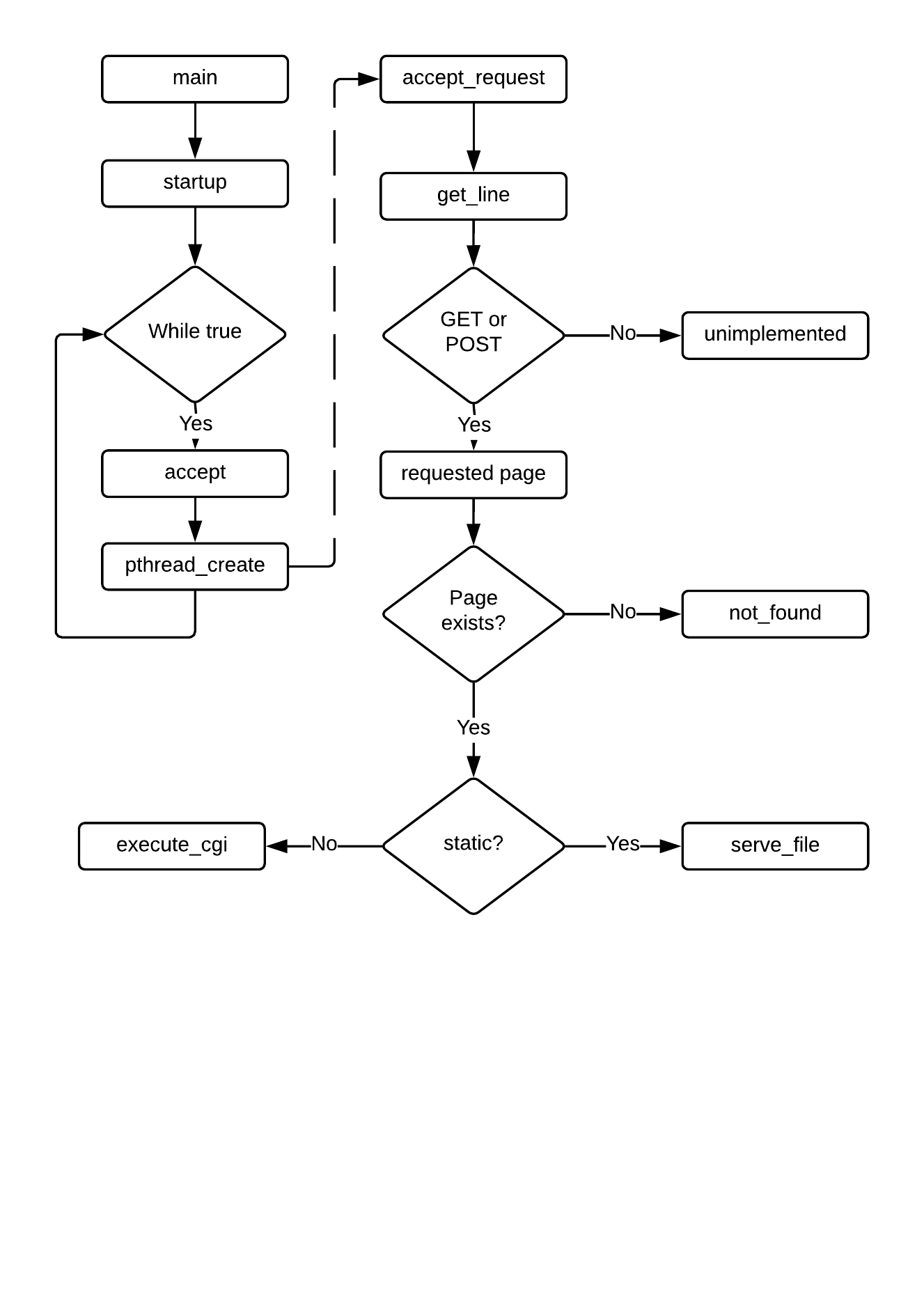
void **cannot\_execute**(int client) – генерирует ошибку выполнения CGI скрипта и отвечает http ответом ‘500 Internal Server Error’

void **cat**(int client, FILE \*resource) – отсылает файл в сокет

void **bad\_request**(int client) - генерирует http ответ с кодом ‘400 BAD REQUEST’

void **accept\_request**(void \*arg) – обрабатывает запрос на порту сервера

### Flowchart



# Задание 3

Скрипт:

#!/usr/bin/env bash

HTTPD\_PID="$(pgrep 'httpd')"

if ! (( $? == 0 ))

then

echo "$(date '+%d/%m/%Y %H:%M:%S') httpd was not running, starting..." >> /root/httpd\_start.log

./httpd &

else

echo "Tiny httpd is already running! PID: $HTTPD\_PID"

exit 1

fi

exit 0