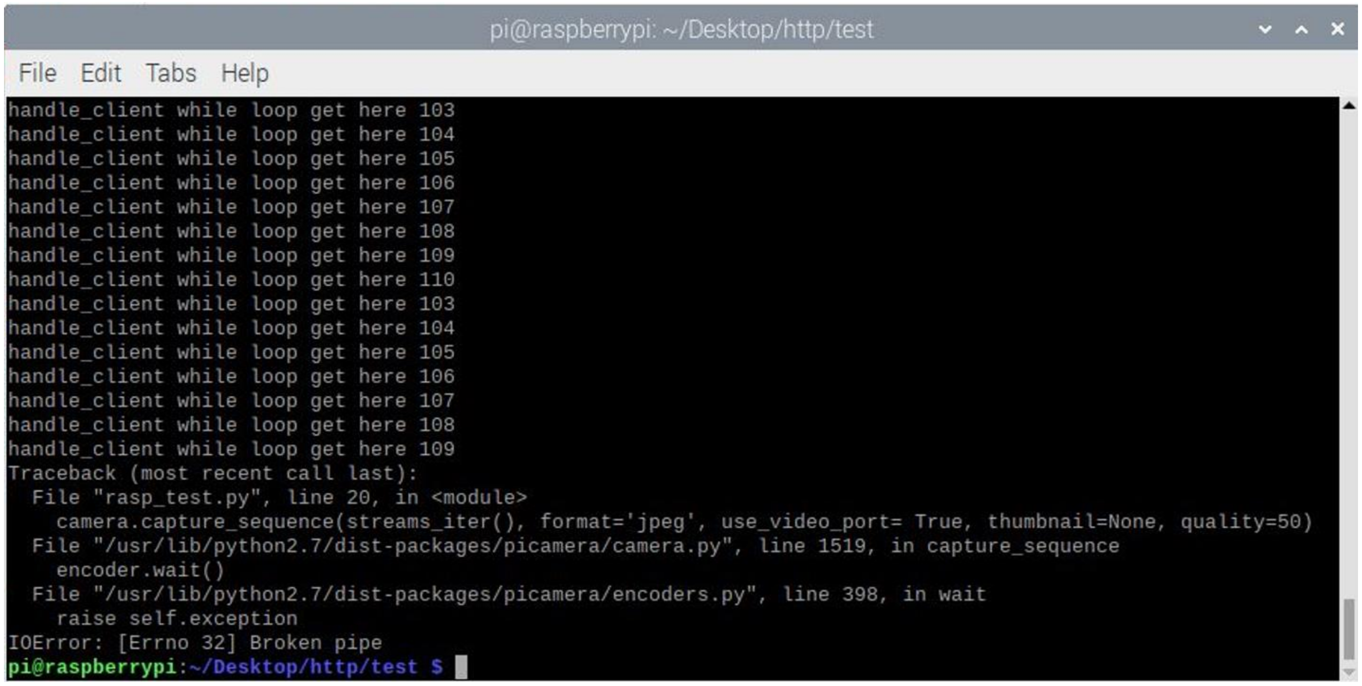
Problem and answer

Problem: Client can receive mjpeg streaming no problem, however, when client disconnected, my program can produce error and crash



```
handle_client while loop get here 103
handle_client while loop get here 104
handle_client while loop get here 105
handle_client while loop get here 106
handle_client while loop get here 107
handle_client while loop get here 108
handle_client while loop get here 109
handle_client while loop get here 110
handle_client while loop get here 103
handle_client while loop get here 104
handle_client while loop get here 105
handle_client while loop get here 106
handle_client while loop get here 107
handle_client while loop get here 108
handle_client while loop get here 109
Traceback (most recent call last):
  File "rasp_test.py", line 20, in <module>
    camera.capture_sequence(streams_iter(), format='jpeg', use_video_port= True, thumbnail=None, quality=50)
  File "/usr/lib/python2.7/dist-packages/picamera/camera.py", line 1519, in capture_sequence
    encoder.wait()
  File "/usr/lib/python2.7/dist-packages/picamera/encoders.py", line 398, in wait
    raise self.exception
IOError: [Errno 32] Broken pipe
pi@raspberrypi:~/Desktop/http/test $
```

Answer:
It seems the problem might occurs inside the write_multipart_header(), inside write_to_client(). It seems error inside linux function write(..)

What causes "[Errno 32] Broken pipe" in Python?
"Broken pipe" is essentially an IOError error (short for input/output error), which happened at the Linux system level. It usually occurs when reading and writing files, or in other words, doing file input/output or network input/output (via sockets).
The corresponding Linux system error is EPIPE, excerpted from GNU libc error codes:

Macro: int ***EPIPE***

"Broken pipe." There is no process reading from the other end of a pipe. Every library function that returns this error code also generates a `SIGPIPE` signal; this signal terminates the program if not handled or blocked. Thus, your program will never actually see `EPIPE` unless it has handled or blocked `SIGPIPE`.

we know that [Errno 32] Broken pipe is caused by the system sending SIGPIPE signal, which is an inter-process communication mechanism of Linux.
For example, SIGINT is another signal used internally by Linux system. In Linux, Ctrl+C will send a SIGINT signal to end the process, or we can use the kill command to achieve the same effect.

Python does not ignore SIGPIPE by default. Instead, it translates the signal into an exception and raises IOError: [Errno 32] Broken pipe every time it receives a SIGPIPE.
[Errno 32] Broken pipe when pipe outputs in Linux terminal
If you encounter [Errno 32] Broken pipe when trying to pipe output of a Python script to another program such as the below example, read on.

```
python <filename>.py | head
```

This pipeline syntax will create a process that sends data upstream, and a process that reads data downstream. When the downstream does not need to read upstream data, it will send a SIGPIPE signal to the upstream process.

When downstream no longer needs to read upstream data? For example, the head command in the example only needs to read enough lines to tell the upstream that I no longer need to read it, and it will send the SIGPIPE signal to the upstream process.

When the upstream process is a Python program, an error such as **IOError: [Errno 32] Broken pipe** will occur.

Avoid [Errno 32] Broken pipe by ignoring SIGPIPE

If you don't care too much about properly catching SIGPIPE and just need to get things running quickly, add the code snippet below to the top of your Python program.

```
from signal import signal, SIGPIPE, SIG_DFL
#Ignore SIG_PIPE and don't throw exceptions on it...
(http://docs.python.org/library/signal.html)
signal(SIGPIPE,SIG_DFL)
```

What the code does is redirecting SIGPIPE signals to the default SIG_DFL, which the system usually ignore.
But beware, the Python manual on signal library warn against this type of handling SIGPIPE

Do not set `SIGPIPE`'s disposition to `SIG_DFL` in order to avoid `BrokenPipeError`. Doing that would cause your program to exit unexpectedly also whenever any socket connection is interrupted while your program is still writing to it.

Properly catch IOError to avoid [Errno 32] Broken pipe

Since [Errno 32] Broken pipe is actually a IOError, you can place a try/catch block to catch it like the code snippet below :

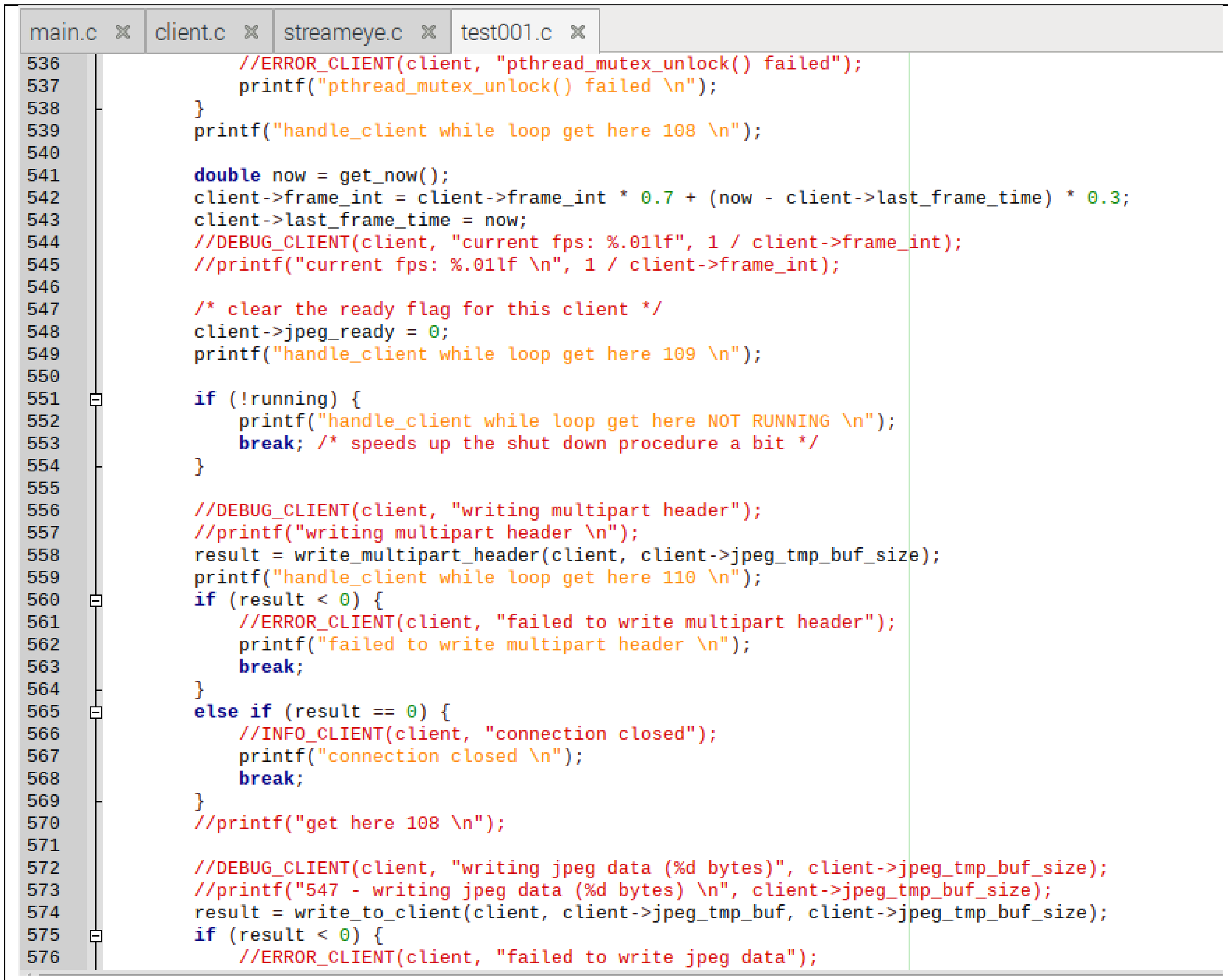
```
import sys, errno
try:
    ### IO operation ###
except IOError as e:
    if e.errno == errno.EPIPE:
        ### Handle the error ###
```

Possible solution for [Errno 32] Broken pipe in multi-process program.

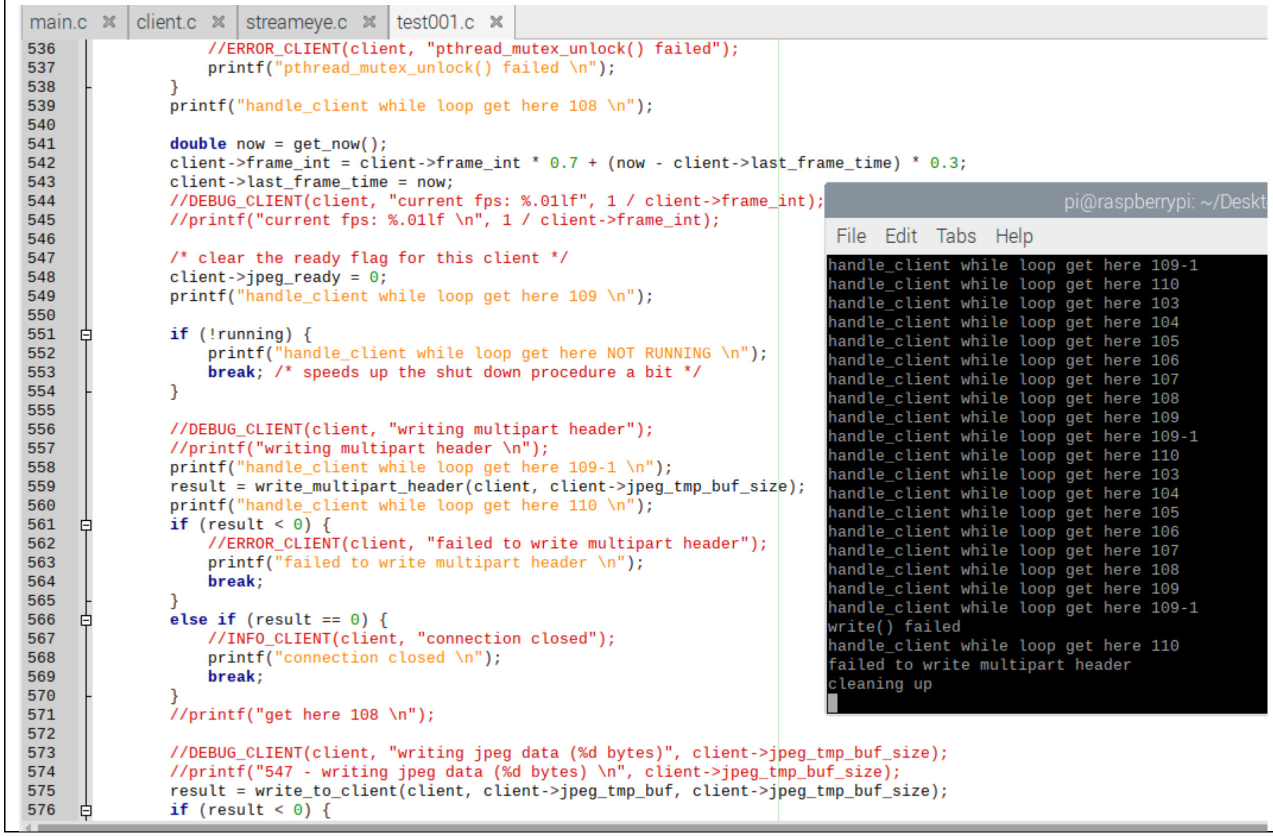
In programs that use worker processes to speed up processing and make use of multi-core CPUs, you can try reducing the number of the worker processes to see whether the error disappear or not.

A large number of worker processes may conflict with each other when they try to take control of system resources or the permission to write into disk.

Code

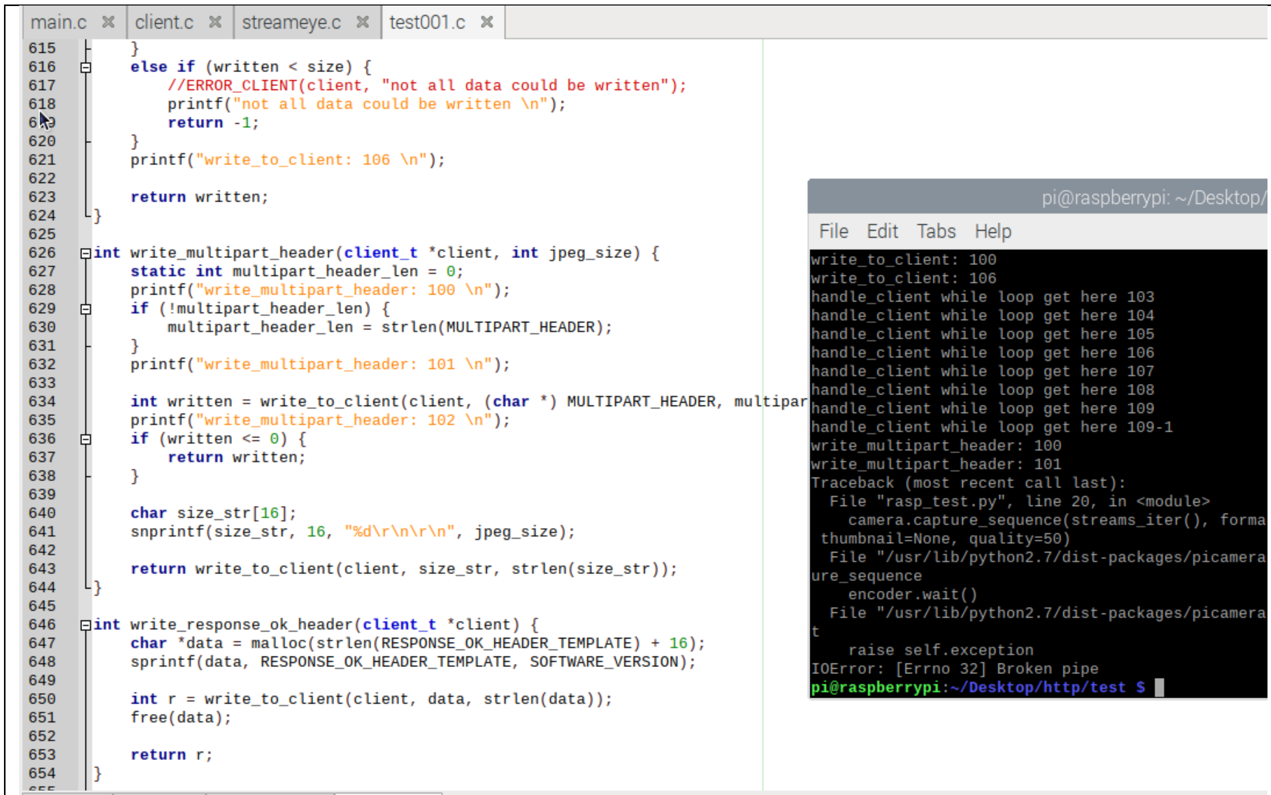


```c
            //ERROR_CLIENT(client, "pthread_mutex_unlock() failed");
            printf("pthread_mutex_unlock() failed \n");
        }
        printf("handle_client while loop get here 108 \n");

        double now = get_now();
        client->frame_int = client->frame_int * 0.7 + (now - client->last_frame_time) * 0.3;
        client->last_frame_time = now;
        //DEBUG_CLIENT(client, "current fps: %.01lf", 1 / client->frame_int);
        //printf("current fps: %.01lf \n", 1 / client->frame_int);

        /* clear the ready flag for this client */
        client->jpeg_ready = 0;
        printf("handle_client while loop get here 109 \n");

        if (!running) {
            printf("handle_client while loop get here NOT RUNNING \n");
            break; /* speeds up the shut down procedure a bit */
        }

        //DEBUG_CLIENT(client, "writing multipart header");
        //printf("writing multipart header \n");
        result = write_multipart_header(client, client->jpeg_tmp_buf_size);
        printf("handle_client while loop get here 110 \n");
        if (result < 0) {
            //ERROR_CLIENT(client, "failed to write multipart header");
            printf("failed to write multipart header \n");
            break;
        }
        else if (result == 0) {
            //INFO_CLIENT(client, "connection closed");
            printf("connection closed \n");
            break;
        }
        //printf("get here 108 \n");

        //DEBUG_CLIENT(client, "writing jpeg data (%d bytes)", client->jpeg_tmp_buf_size);
        //printf("547 - writing jpeg data (%d bytes) \n", client->jpeg_tmp_buf_size);
        result = write_to_client(client, client->jpeg_tmp_buf, client->jpeg_tmp_buf_size);
        if (result < 0) {
            //ERROR_CLIENT(client, "failed to write jpeg data");
```



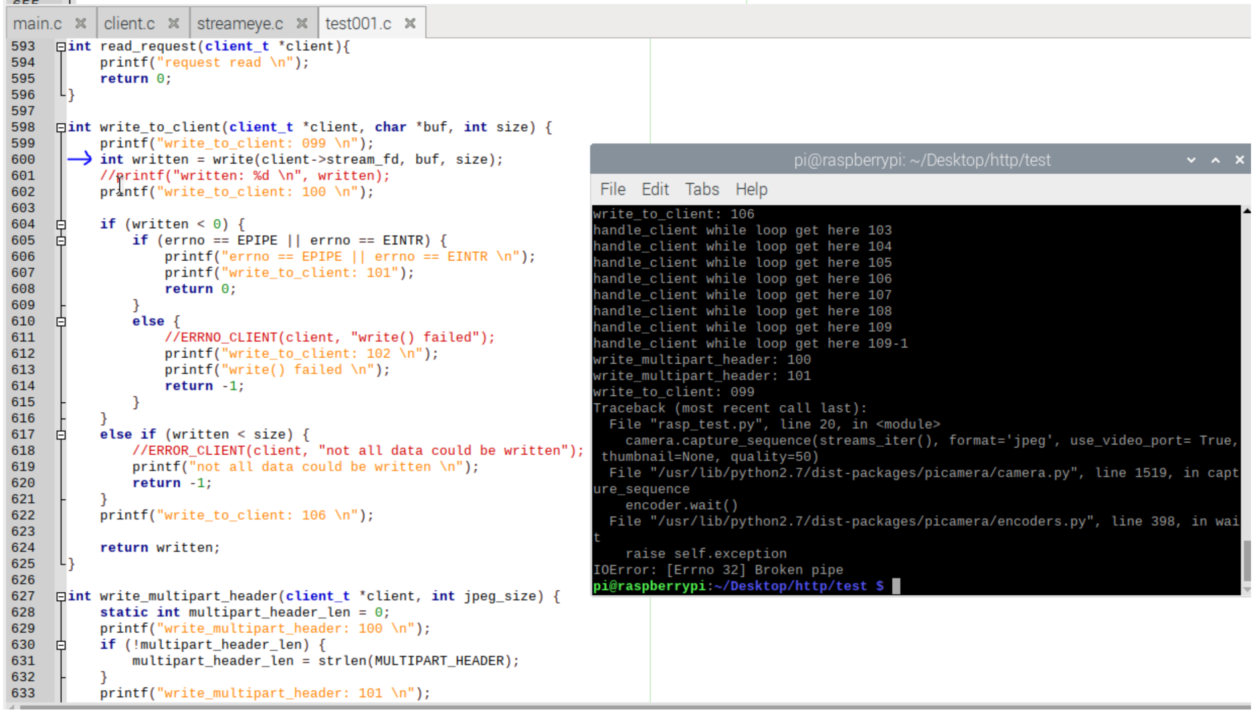
```c
            //ERROR_CLIENT(client, "pthread_mutex_unlock() failed");
            printf("pthread_mutex_unlock() failed \n");
        }
        printf("handle_client while loop get here 108 \n");

        double now = get_now();
        client->frame_int = client->frame_int * 0.7 + (now - client->last_frame_time) * 0.3;
        client->last_frame_time = now;
        //DEBUG_CLIENT(client, "current fps: %.01lf", 1 / client->frame_int);
        //printf("current fps: %.01lf \n", 1 / client->frame_int);

        /* clear the ready flag for this client */
        client->jpeg_ready = 0;
        printf("handle_client while loop get here 109 \n");

        if (!running) {
            printf("handle_client while loop get here NOT RUNNING \n");
            break; /* speeds up the shut down procedure a bit */
        }

        //DEBUG_CLIENT(client, "writing multipart header");
        //printf("writing multipart header \n");
        printf("handle_client while loop get here 109-1 \n");
        result = write_multipart_header(client, client->jpeg_tmp_buf_size);
        printf("handle_client while loop get here 110 \n");
        if (result < 0) {
            //ERROR_CLIENT(client, "failed to write multipart header");
            printf("failed to write multipart header \n");
            break;
        }
        else if (result == 0) {
            //INFO_CLIENT(client, "connection closed");
            printf("connection closed \n");
            break;
        }
        //printf("get here 108 \n");

        //DEBUG_CLIENT(client, "writing jpeg data (%d bytes)", client->jpeg_tmp_buf_size);
        //printf("547 - writing jpeg data (%d bytes) \n", client->jpeg_tmp_buf_size);
        result = write_to_client(client, client->jpeg_tmp_buf, client->jpeg_tmp_buf_size);
        if (result < 0) {
```

```
handle_client while loop get here 109-1
handle_client while loop get here 110
handle_client while loop get here 103
handle_client while loop get here 104
handle_client while loop get here 105
handle_client while loop get here 106
handle_client while loop get here 107
handle_client while loop get here 108
handle_client while loop get here 109
handle_client while loop get here 109-1
handle_client while loop get here 110
handle_client while loop get here 103
handle_client while loop get here 104
handle_client while loop get here 105
handle_client while loop get here 106
handle_client while loop get here 107
handle_client while loop get here 108
handle_client while loop get here 109
handle_client while loop get here 109-1
write() failed
handle_client while loop get here 110
failed to write multipart header
cleaning up
```



```c
    }
    else if (written < size) {
        //ERROR_CLIENT(client, "not all data could be written");
        printf("not all data could be written \n");
        return -1;
    }
    printf("write_to_client: 106 \n");

    return written;
}

int write_multipart_header(client_t *client, int jpeg_size) {
    static int multipart_header_len = 0;
    printf("write_multipart_header: 100 \n");
    if (!multipart_header_len) {
        multipart_header_len = strlen(MULTIPART_HEADER);
    }
    printf("write_multipart_header: 101 \n");

    int written = write_to_client(client, (char *) MULTIPART_HEADER, multipar
    printf("write_multipart_header: 102 \n");
    if (written <= 0) {
        return written;
    }

    char size_str[16];
    snprintf(size_str, 16, "%d\r\n\r\n", jpeg_size);

    return write_to_client(client, size_str, strlen(size_str));
}

int write_response_ok_header(client_t *client) {
    char *data = malloc(strlen(RESPONSE_OK_HEADER_TEMPLATE) + 16);
    sprintf(data, RESPONSE_OK_HEADER_TEMPLATE, SOFTWARE_VERSION);

    int r = write_to_client(client, data, strlen(data));
    free(data);

    return r;
}
```

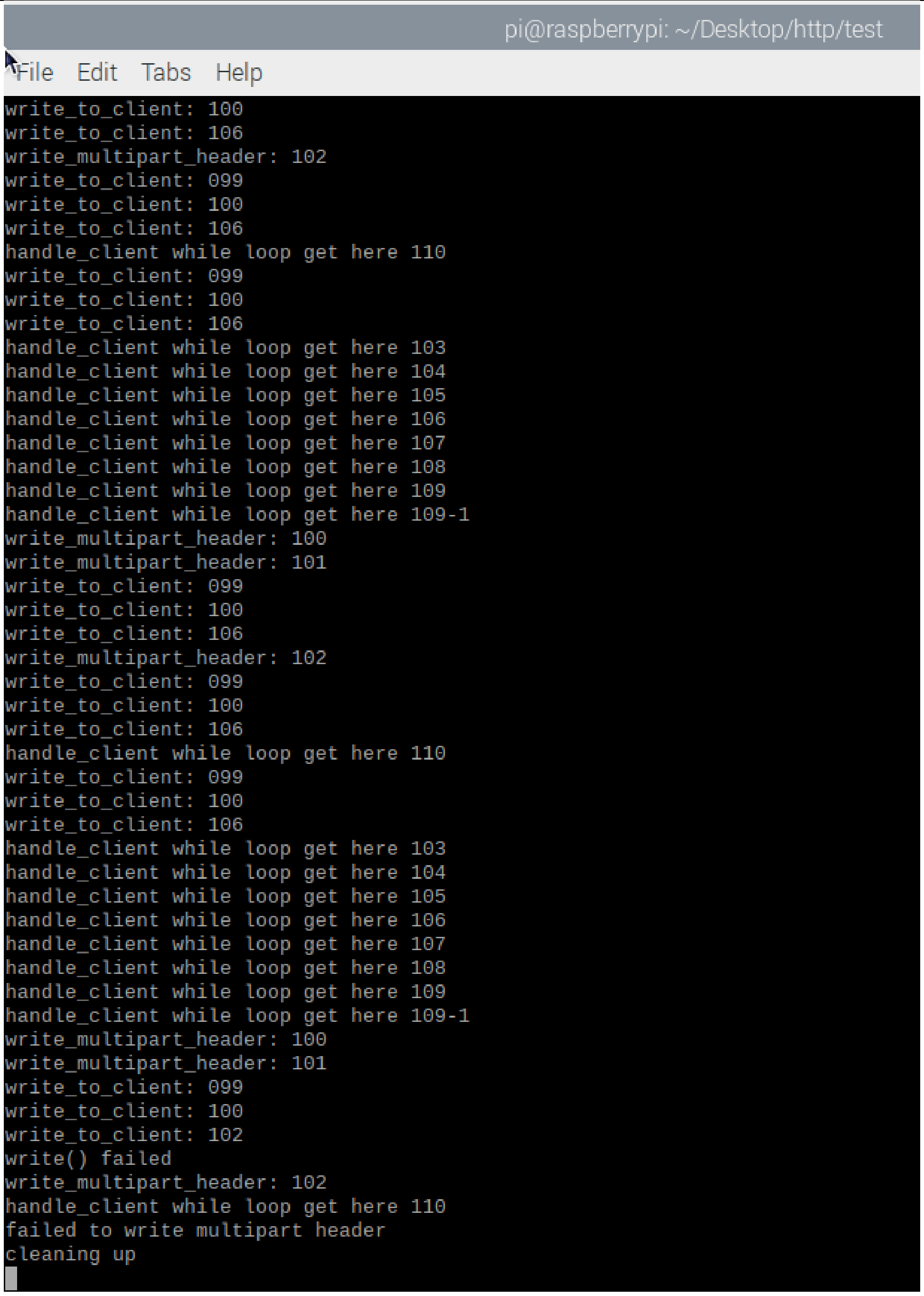
```
write_to_client: 100
write_to_client: 106
handle_client while loop get here 103
handle_client while loop get here 104
handle_client while loop get here 105
handle_client while loop get here 106
handle_client while loop get here 107
handle_client while loop get here 108
handle_client while loop get here 109
handle_client while loop get here 109-1
write_multipart_header: 100
write_multipart_header: 101
Traceback (most recent call last):
  File "rasp_test.py", line 20, in <module>
    camera.capture_sequence(streams_iter(), forma
 thumbnail=None, quality=50)
  File "/usr/lib/python2.7/dist-packages/picamera
ure_sequence
    encoder.wait()
  File "/usr/lib/python2.7/dist-packages/picamera
t
    raise self.exception
IOError: [Errno 32] Broken pipe
pi@raspberrypi:~/Desktop/http/test $
```



```c
int read_request(client_t *client){
    printf("request read \n");
    return 0;
}

int write_to_client(client_t *client, char *buf, int size) {
    printf("write_to_client: 099 \n");
    int written = write(client->stream_fd, buf, size);
    //printf("written: %d \n", written);
    printf("write_to_client: 100 \n");

    if (written < 0) {
        if (errno == EPIPE || errno == EINTR) {
            printf("errno == EPIPE || errno == EINTR \n");
            printf("write_to_client: 101");
            return 0;
        }
        else {
            //ERRNO_CLIENT(client, "write() failed");
            printf("write_to_client: 102 \n");
            printf("write() failed \n");
            return -1;
        }
    }
    else if (written < size) {
        //ERROR_CLIENT(client, "not all data could be written");
        printf("not all data could be written \n");
        return -1;
    }
    printf("write_to_client: 106 \n");

    return written;
}

int write_multipart_header(client_t *client, int jpeg_size) {
    static int multipart_header_len = 0;
    printf("write_multipart_header: 100 \n");
    if (!multipart_header_len) {
        multipart_header_len = strlen(MULTIPART_HEADER);
    }
    printf("write_multipart_header: 101 \n");
```

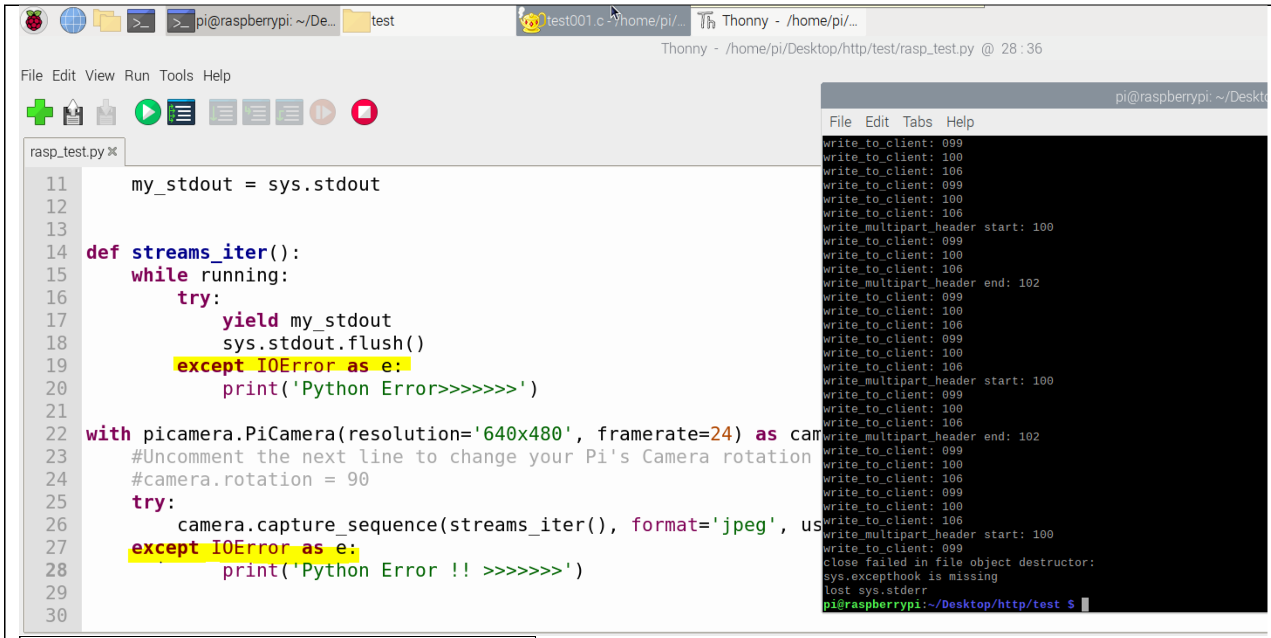
```
write_to_client: 106
handle_client while loop get here 103
handle_client while loop get here 104
handle_client while loop get here 105
handle_client while loop get here 106
handle_client while loop get here 107
handle_client while loop get here 108
handle_client while loop get here 109
handle_client while loop get here 109-1
write_multipart_header: 100
write_multipart_header: 101
write_to_client: 099
Traceback (most recent call last):
  File "rasp_test.py", line 20, in <module>
    camera.capture_sequence(streams_iter(), format='jpeg', use_video_port= True,
 thumbnail=None, quality=50)
  File "/usr/lib/python2.7/dist-packages/picamera/camera.py", line 1519, in capt
ure_sequence
    encoder.wait()
  File "/usr/lib/python2.7/dist-packages/picamera/encoders.py", line 398, in wai
t
    raise self.exception
IOError: [Errno 32] Broken pipe
pi@raspberrypi:~/Desktop/http/test $
```

Normal one should be like the following



```
write_to_client: 100
write_to_client: 106
write_multipart_header: 102
write_to_client: 099
write_to_client: 100
write_to_client: 106
handle_client while loop get here 110
write_to_client: 099
write_to_client: 100
write_to_client: 106
handle_client while loop get here 103
handle_client while loop get here 104
handle_client while loop get here 105
handle_client while loop get here 106
handle_client while loop get here 107
handle_client while loop get here 108
handle_client while loop get here 109
handle_client while loop get here 109-1
write_multipart_header: 100
write_multipart_header: 101
write_to_client: 099
write_to_client: 100
write_to_client: 106
write_multipart_header: 102
write_to_client: 099
write_to_client: 100
write_to_client: 106
handle_client while loop get here 110
write_to_client: 099
write_to_client: 100
write_to_client: 106
handle_client while loop get here 103
handle_client while loop get here 104
handle_client while loop get here 105
handle_client while loop get here 106
handle_client while loop get here 107
handle_client while loop get here 108
handle_client while loop get here 109
handle_client while loop get here 109-1
write_multipart_header: 100
write_multipart_header: 101
write_to_client: 099
write_to_client: 100
write_to_client: 102
write() failed
write_multipart_header: 102
handle_client while loop get here 110
failed to write multipart header
cleaning up
```

In python file I add try catch and get following

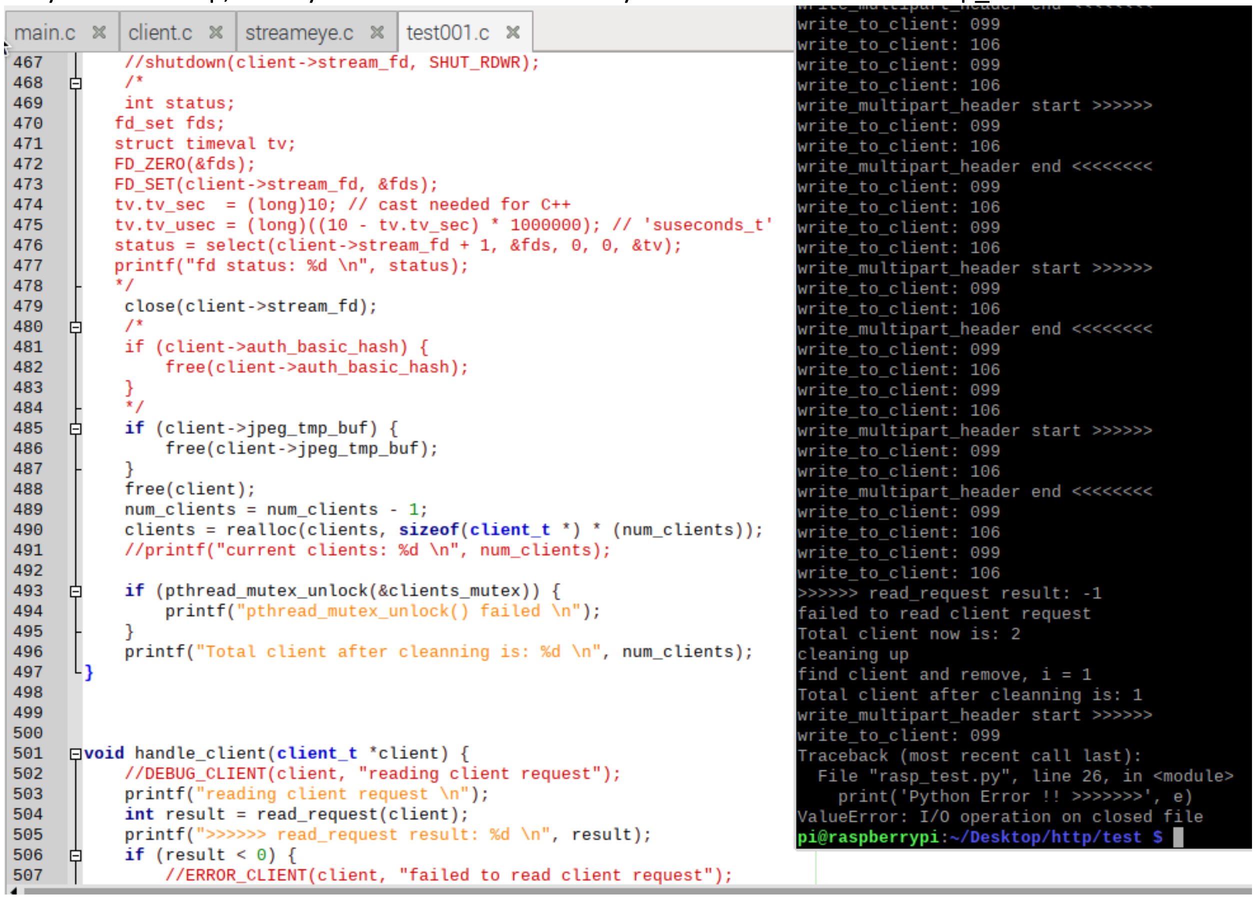


```python
    my_stdout = sys.stdout


def streams_iter():
    while running:
        try:
            yield my_stdout
            sys.stdout.flush()
        except IOError as e:
            print('Python Error>>>>>>>')

with picamera.PiCamera(resolution='640x480', framerate=24) as cam
    #Uncomment the next line to change your Pi's Camera rotation
    #camera.rotation = 90
    try:
        camera.capture_sequence(streams_iter(), format='jpeg', us
    except IOError as e:
            print('Python Error !! >>>>>>')
```

```
write_to_client: 099
write_to_client: 100
write_to_client: 106
write_to_client: 099
write_to_client: 100
write_to_client: 106
write_multipart_header start: 100
write_to_client: 099
write_to_client: 100
write_to_client: 106
write_multipart_header end: 102
write_to_client: 099
write_to_client: 100
write_to_client: 106
write_to_client: 099
write_to_client: 100
write_to_client: 106
write_multipart_header start: 100
write_to_client: 099
write_to_client: 100
write_to_client: 106
write_multipart_header end: 102
write_to_client: 099
write_to_client: 100
write_to_client: 106
write_to_client: 099
write_to_client: 100
write_to_client: 106
write_multipart_header start: 100
write_to_client: 099
close failed in file object destructor:
sys.excepthook is missing
lost sys.stderr
pi@raspberrypi:~/Desktop/http/test $
```

close failed in file object destructor:
sys.excepthook is missing
lost sys.stderr

Why after cleanup, still try to send multiheader. Maybe need to look into cleanup_client

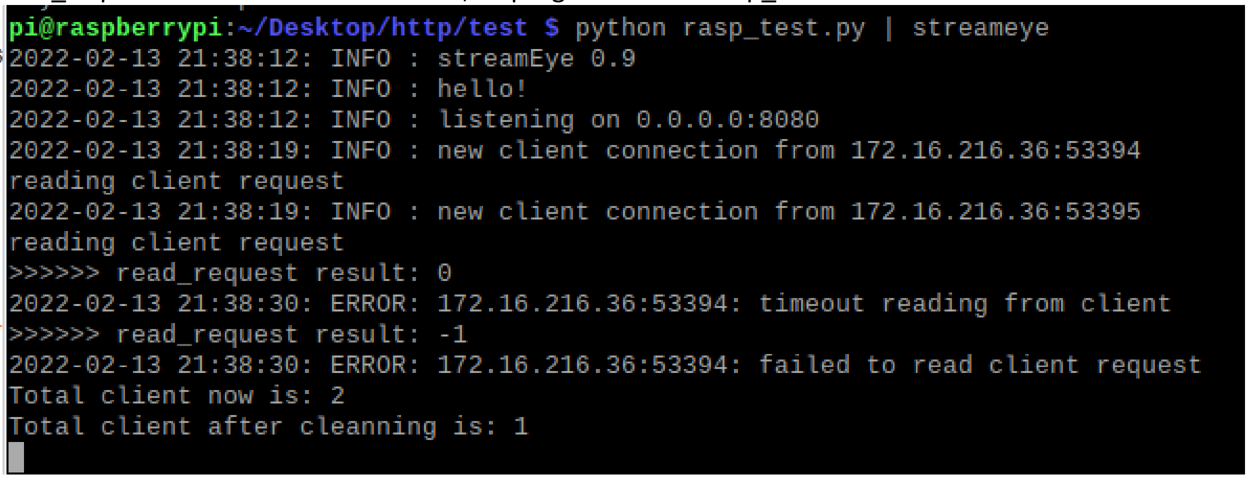


```c
    //shutdown(client->stream_fd, SHUT_RDWR);
    /*
    int status;
   fd_set fds;
   struct timeval tv;
   FD_ZERO(&fds);
   FD_SET(client->stream_fd, &fds);
   tv.tv_sec  = (long)10; // cast needed for C++
   tv.tv_usec = (long)((10 - tv.tv_sec) * 1000000); // 'suseconds_t'
   status = select(client->stream_fd + 1, &fds, 0, 0, &tv);
   printf("fd status: %d \n", status);
   */
    close(client->stream_fd);
    /*
    if (client->auth_basic_hash) {
        free(client->auth_basic_hash);
    }
    */
    if (client->jpeg_tmp_buf) {
        free(client->jpeg_tmp_buf);
    }
    free(client);
    num_clients = num_clients - 1;
    clients = realloc(clients, sizeof(client_t *) * (num_clients));
    //printf("current clients: %d \n", num_clients);

    if (pthread_mutex_unlock(&clients_mutex)) {
        printf("pthread_mutex_unlock() failed \n");
    }
    printf("Total client after cleanning is: %d \n", num_clients);
}



void handle_client(client_t *client) {
    //DEBUG_CLIENT(client, "reading client request");
    printf("reading client request \n");
    int result = read_request(client);
    printf(">>>>>> read_request result: %d \n", result);
    if (result < 0) {
        //ERROR_CLIENT(client, "failed to read client request");
```

```
write_to_client: 099
write_to_client: 106
write_to_client: 099
write_to_client: 106
write_multipart_header start >>>>>>
write_to_client: 099
write_to_client: 106
write_multipart_header end <<<<<<<<
write_to_client: 099
write_to_client: 106
write_to_client: 099
write_to_client: 106
write_multipart_header start >>>>>>
write_to_client: 099
write_to_client: 106
write_multipart_header end <<<<<<<<
write_to_client: 099
write_to_client: 106
write_to_client: 099
write_to_client: 106
write_multipart_header start >>>>>>
write_to_client: 099
write_to_client: 106
write_multipart_header end <<<<<<<<
write_to_client: 099
write_to_client: 106
write_to_client: 099
write_to_client: 106
>>>>>> read_request result: -1
failed to read client request
Total client now is: 2
cleaning up
find client and remove, i = 1
Total client after cleanning is: 1
write_multipart_header start >>>>>>
write_to_client: 099
Traceback (most recent call last):
  File "rasp_test.py", line 26, in <module>
    print('Python Error !! >>>>>>', e)
ValueError: I/O operation on closed file
pi@raspberrypi:~/Desktop/http/test $
```

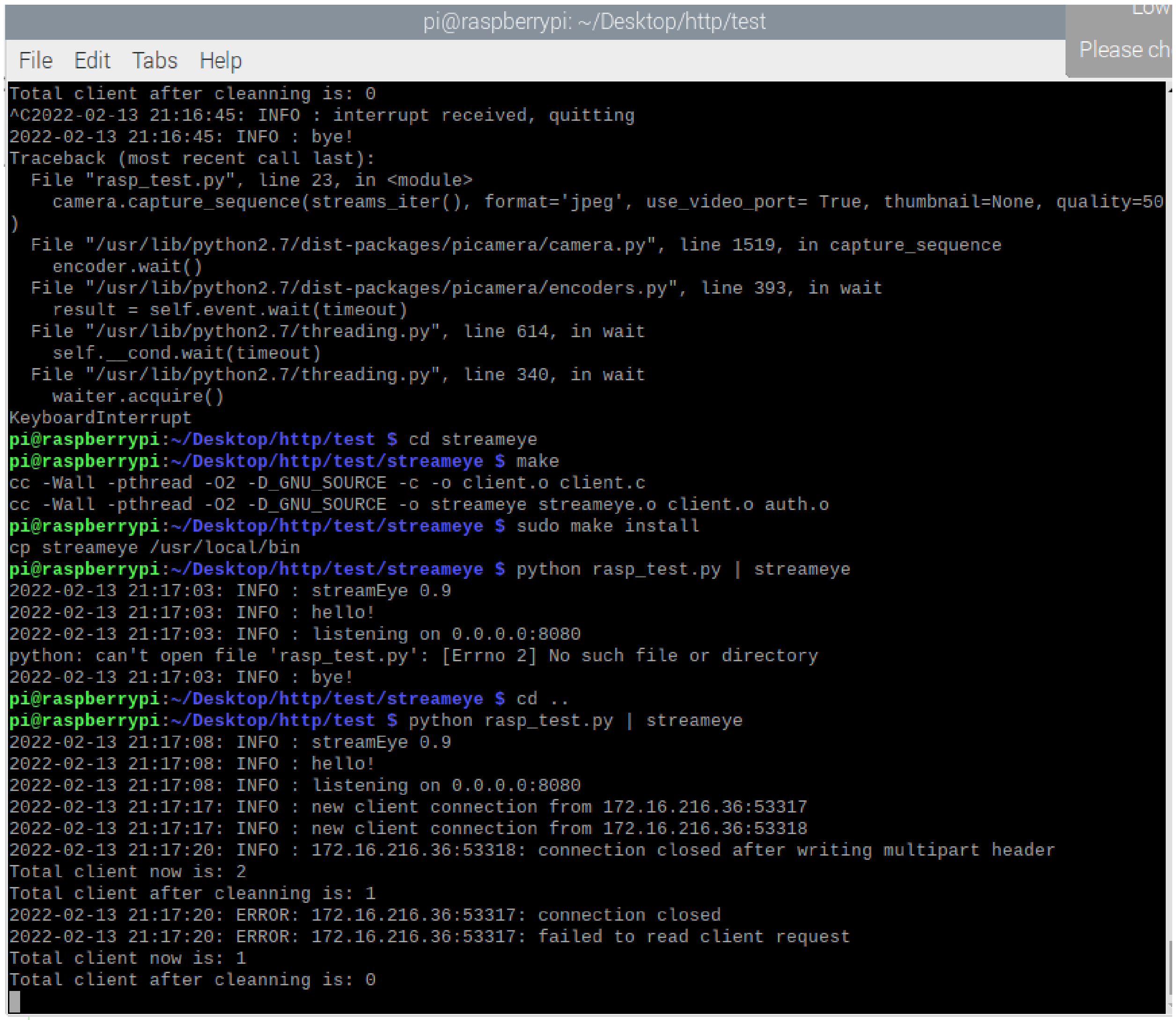
The following is how the original streameye should do after cleanup_client

At very first client browser connect, it will directly connect with two port (so client number is now 2). After a while, read_request cannot read one of client, so program will cleanup_client. so client number is now 1



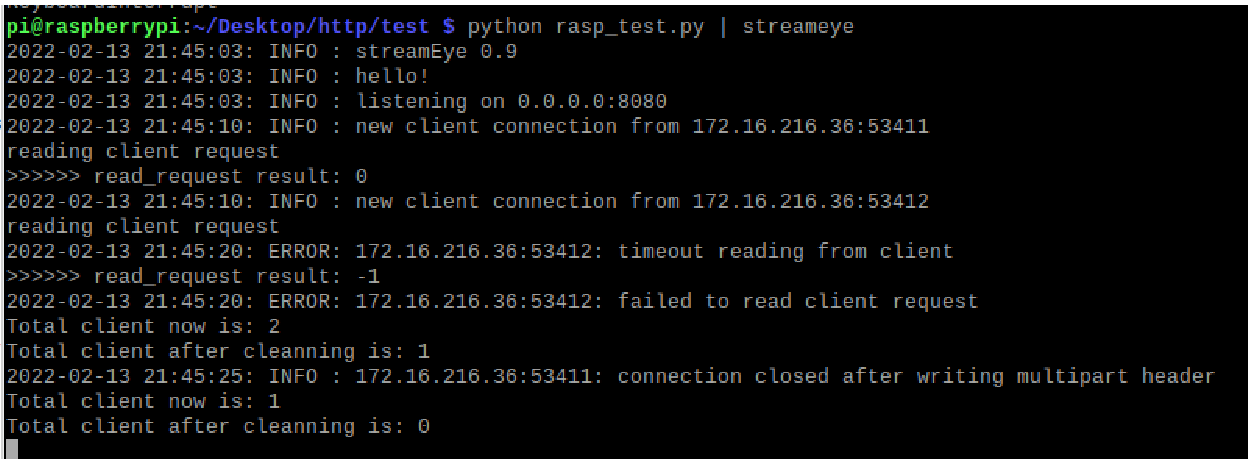
```
pi@raspberrypi:~/Desktop/http/test $ python rasp_test.py | streameye
2022-02-13 21:38:12: INFO : streamEye 0.9
2022-02-13 21:38:12: INFO : hello!
2022-02-13 21:38:12: INFO : listening on 0.0.0.0:8080
2022-02-13 21:38:19: INFO : new client connection from 172.16.216.36:53394
reading client request
2022-02-13 21:38:19: INFO : new client connection from 172.16.216.36:53395
reading client request
>>>>>> read_request result: 0
2022-02-13 21:38:30: ERROR: 172.16.216.36:53394: timeout reading from client
>>>>>> read_request result: -1
2022-02-13 21:38:30: ERROR: 172.16.216.36:53394: failed to read client request
Total client now is: 2
Total client after cleanning is: 1
```

The following is I close browser immediately I get connected, didn't wait for first client being removed by program after timeout.



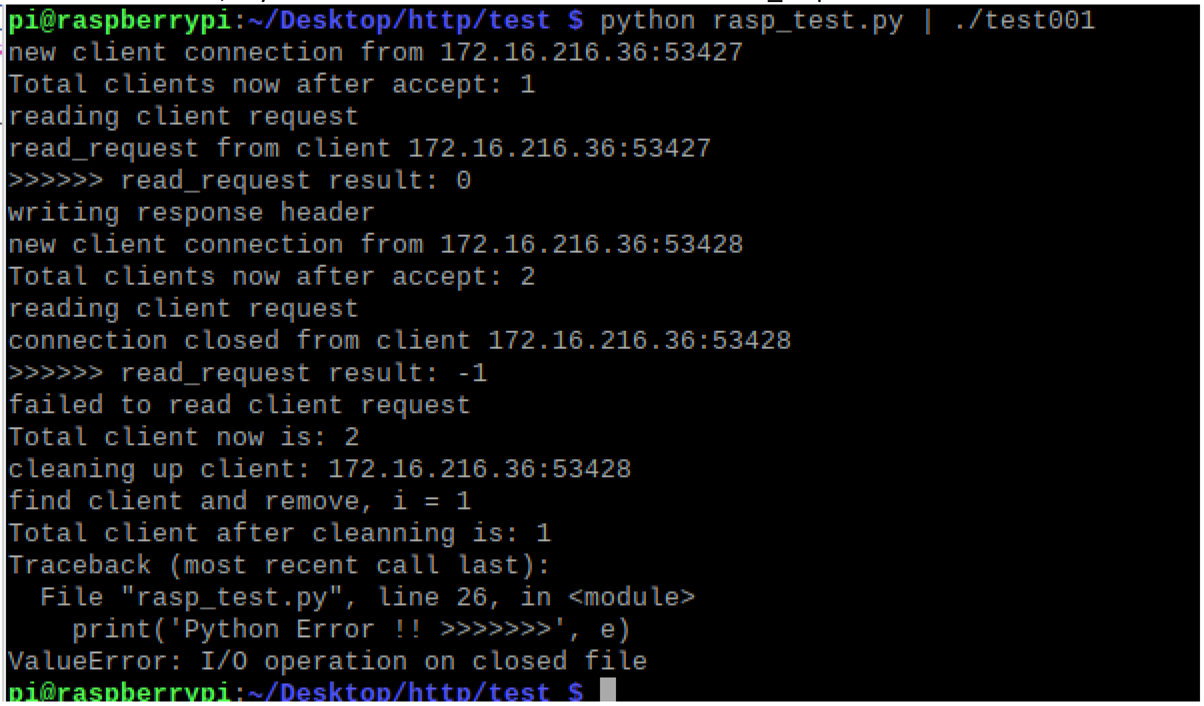
```
Total client after cleanning is: 0
^C2022-02-13 21:16:45: INFO : interrupt received, quitting
2022-02-13 21:16:45: INFO : bye!
Traceback (most recent call last):
  File "rasp_test.py", line 23, in <module>
    camera.capture_sequence(streams_iter(), format='jpeg', use_video_port= True, thumbnail=None, quality=50
)
  File "/usr/lib/python2.7/dist-packages/picamera/camera.py", line 1519, in capture_sequence
    encoder.wait()
  File "/usr/lib/python2.7/dist-packages/picamera/encoders.py", line 393, in wait
    result = self.event.wait(timeout)
  File "/usr/lib/python2.7/threading.py", line 614, in wait
    self.__cond.wait(timeout)
  File "/usr/lib/python2.7/threading.py", line 340, in wait
    waiter.acquire()
KeyboardInterrupt
pi@raspberrypi:~/Desktop/http/test $ cd streameye
pi@raspberrypi:~/Desktop/http/test/streameye $ make
cc -Wall -pthread -O2 -D_GNU_SOURCE -c -o client.o client.c
cc -Wall -pthread -O2 -D_GNU_SOURCE -o streameye streameye.o client.o auth.o
pi@raspberrypi:~/Desktop/http/test/streameye $ sudo make install
cp streameye /usr/local/bin
pi@raspberrypi:~/Desktop/http/test/streameye $ python rasp_test.py | streameye
2022-02-13 21:17:03: INFO : streamEye 0.9
2022-02-13 21:17:03: INFO : hello!
2022-02-13 21:17:03: INFO : listening on 0.0.0.0:8080
python: can't open file 'rasp_test.py': [Errno 2] No such file or directory
2022-02-13 21:17:03: INFO : bye!
pi@raspberrypi:~/Desktop/http/test/streameye $ cd ..
pi@raspberrypi:~/Desktop/http/test $ python rasp_test.py | streameye
2022-02-13 21:17:08: INFO : streamEye 0.9
2022-02-13 21:17:08: INFO : hello!
2022-02-13 21:17:08: INFO : listening on 0.0.0.0:8080
2022-02-13 21:17:17: INFO : new client connection from 172.16.216.36:53317
2022-02-13 21:17:17: INFO : new client connection from 172.16.216.36:53318
2022-02-13 21:17:20: INFO : 172.16.216.36:53318: connection closed after writing multipart header
Total client now is: 2
Total client after cleanning is: 1
2022-02-13 21:17:20: ERROR: 172.16.216.36:53317: connection closed
2022-02-13 21:17:20: ERROR: 172.16.216.36:53317: failed to read client request
Total client now is: 1
Total client after cleanning is: 0
```

The following is I close browser after waiting for first client being removed by program after timeout.



```
pi@raspberrypi:~/Desktop/http/test $ python rasp_test.py | streameye
2022-02-13 21:45:03: INFO : streamEye 0.9
2022-02-13 21:45:03: INFO : hello!
2022-02-13 21:45:03: INFO : listening on 0.0.0.0:8080
2022-02-13 21:45:10: INFO : new client connection from 172.16.216.36:53411
reading client request
>>>>>> read_request result: 0
2022-02-13 21:45:10: INFO : new client connection from 172.16.216.36:53412
reading client request
2022-02-13 21:45:20: ERROR: 172.16.216.36:53412: timeout reading from client
>>>>>> read_request result: -1
2022-02-13 21:45:20: ERROR: 172.16.216.36:53412: failed to read client request
Total client now is: 2
Total client after cleanning is: 1
2022-02-13 21:45:25: INFO : 172.16.216.36:53411: connection closed after writing multipart header
Total client now is: 1
Total client after cleanning is: 0
```

On the other hand, my code seems not to auto timeout the read_request



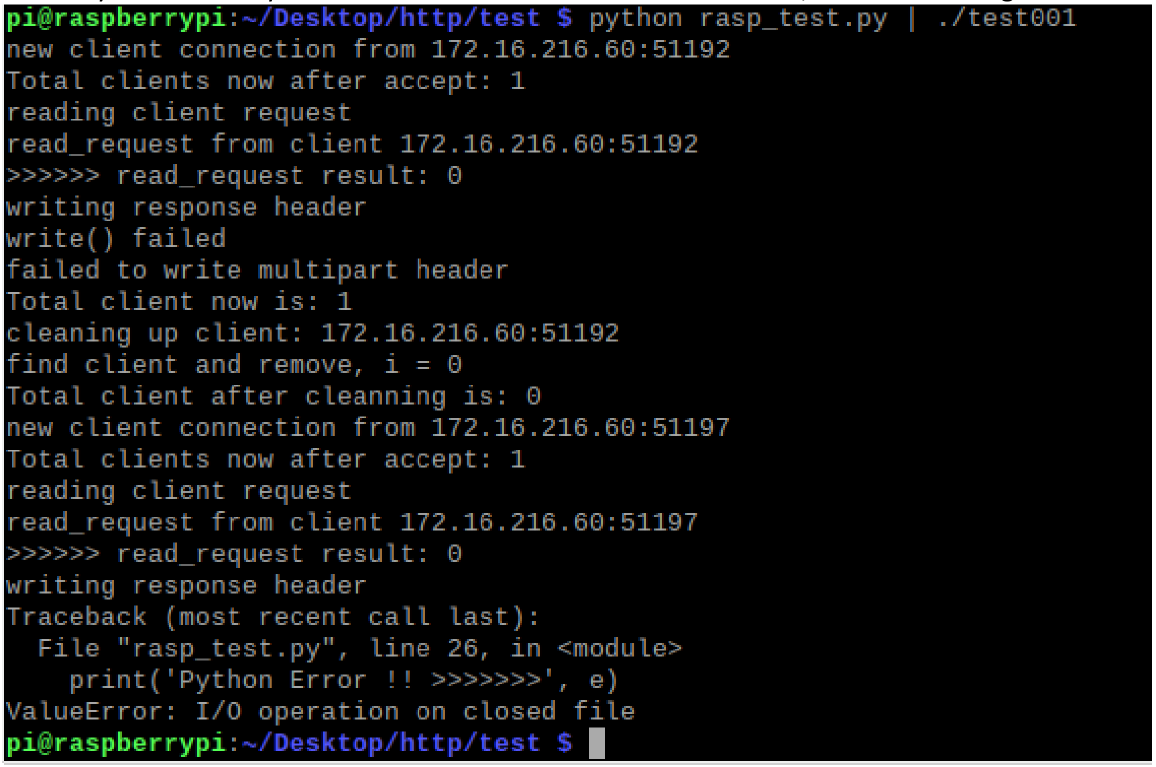
```
pi@raspberrypi:~/Desktop/http/test $ python rasp_test.py | ./test001
new client connection from 172.16.216.36:53427
Total clients now after accept: 1
reading client request
read_request from client 172.16.216.36:53427
>>>>>> read_request result: 0
writing response header
new client connection from 172.16.216.36:53428
Total clients now after accept: 2
reading client request
connection closed from client 172.16.216.36:53428
>>>>>> read_request result: -1
failed to read client request
Total client now is: 2
cleaning up client: 172.16.216.36:53428
find client and remove, i = 1
Total client after cleanning is: 1
Traceback (most recent call last):
  File "rasp_test.py", line 26, in <module>
    print('Python Error !! >>>>>>>', e)
ValueError: I/O operation on closed file
pi@raspberrypi:~/Desktop/http/test $
```

Actually, I found out that I forgot to add the following code inside function wait_for_client so that we can set timeout for socket. The following code does shows up in streameye.c

```
/* set socket timeout */

    struct timeval tv;

    tv.tv_sec = 1;
    tv.tv_usec = 0;

    setsockopt(stream_fd, SOL_SOCKET, SO_RCVTIMEO, (char *) &tv, sizeof(struct timeval));
    setsockopt(stream_fd, SOL_SOCKET, SO_SNDTIMEO, (char *) &tv, sizeof(struct timeval));
```

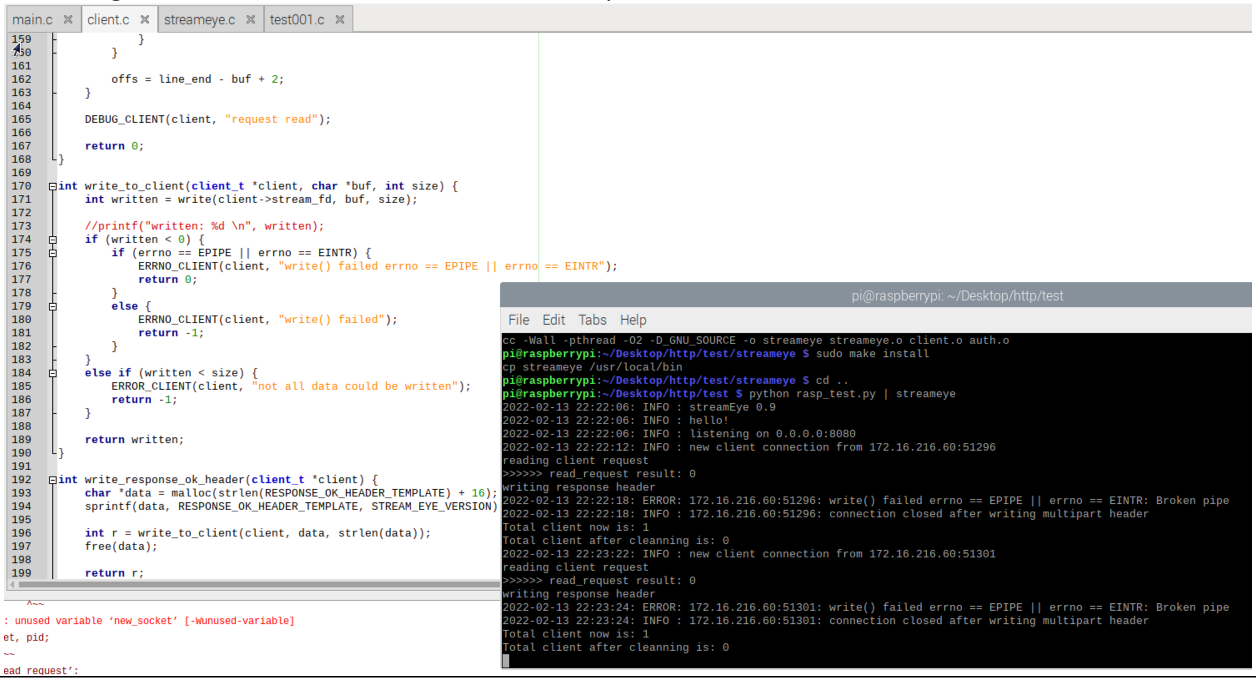
So after we add the above code, now my program will auto timeout the read_request

I don't know why sometime my code would work when client disconnected, but sometimes go into error.



```
pi@raspberrypi:~/Desktop/http/test $ python rasp_test.py | ./test001
new client connection from 172.16.216.60:51192
Total clients now after accept: 1
reading client request
read_request from client 172.16.216.60:51192
>>>>>> read_request result: 0
writing response header
write() failed
failed to write multipart header
Total client now is: 1
cleaning up client: 172.16.216.60:51192
find client and remove, i = 0
Total client after cleanning is: 0
new client connection from 172.16.216.60:51197
Total clients now after accept: 1
reading client request
read_request from client 172.16.216.60:51197
>>>>>> read_request result: 0
writing response header
Traceback (most recent call last):
  File "rasp_test.py", line 26, in <module>
    print('Python Error !! >>>>>>>', e)
ValueError: I/O operation on closed file
pi@raspberrypi:~/Desktop/http/test $ 
```

I found that streameye.c will also suffer from broken pipe like me (after adding printf the error in streameye.c). But it seems it got error handle while I don't. That is the problem I think



main.c | client.c | streameye.c | test001.c

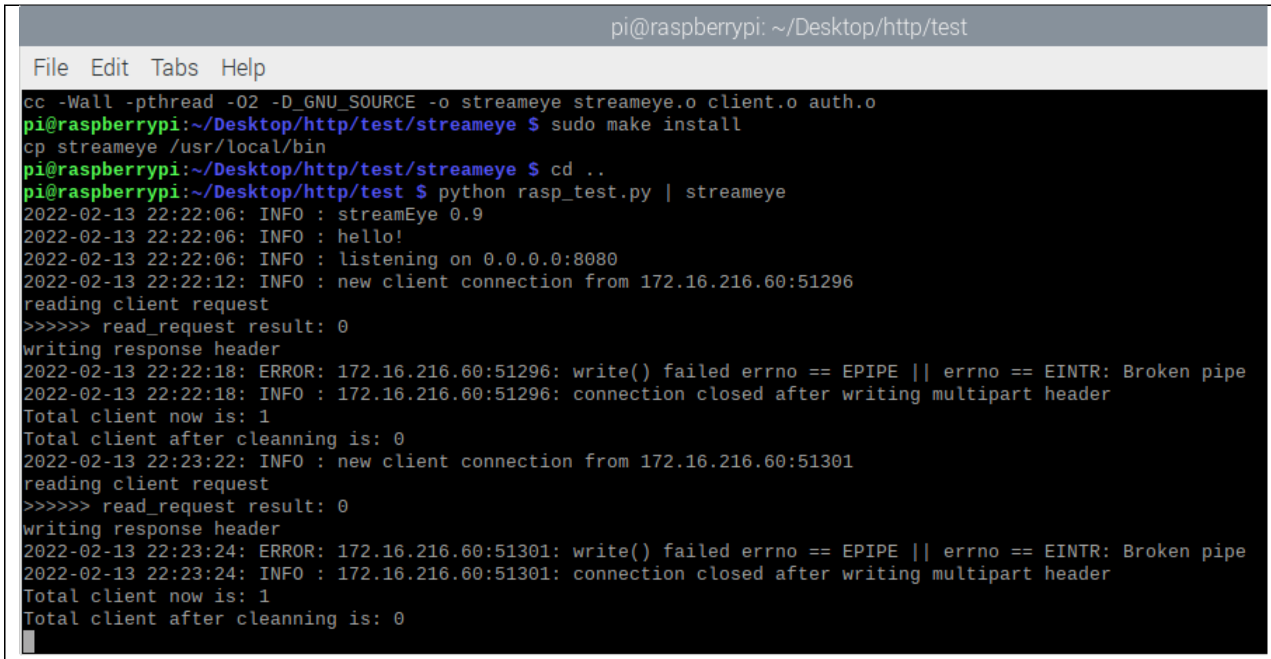
```c
            }
        }

        offs = line_end - buf + 2;
    }

    DEBUG_CLIENT(client, "request read");

    return 0;
}

int write_to_client(client_t *client, char *buf, int size) {
    int written = write(client->stream_fd, buf, size);

    //printf("written: %d \n", written);
    if (written < 0) {
        if (errno == EPIPE || errno == EINTR) {
            ERRNO_CLIENT(client, "write() failed errno == EPIPE || errno == EINTR");
            return 0;
        }
        else {
            ERRNO_CLIENT(client, "write() failed");
            return -1;
        }
    }
    else if (written < size) {
        ERROR_CLIENT(client, "not all data could be written");
        return -1;
    }

    return written;
}

int write_response_ok_header(client_t *client) {
    char *data = malloc(strlen(RESPONSE_OK_HEADER_TEMPLATE) + 16);
    sprintf(data, RESPONSE_OK_HEADER_TEMPLATE, STREAM_EYE_VERSION)

    int r = write_to_client(client, data, strlen(data));
    free(data);

    return r;
```

```
    ^~~
: unused variable ‘new_socket’ [-Wunused-variable]
et, pid;
~~
ead_request’:
```

pi@raspberrypi: ~/Desktop/http/test

File Edit Tabs Help

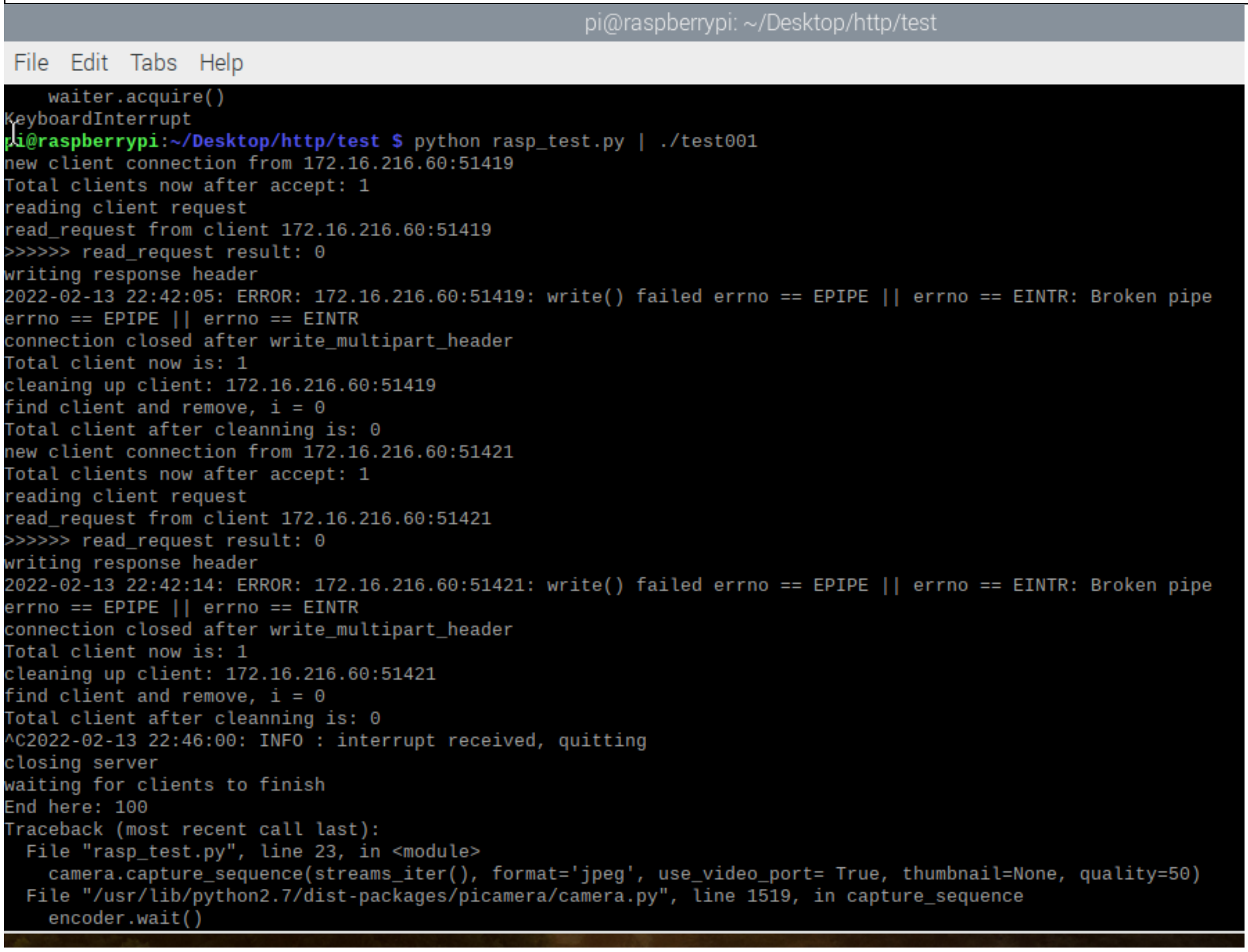
```
cc -Wall -pthread -O2 -D_GNU_SOURCE -o streameye streameye.o client.o auth.o
pi@raspberrypi:~/Desktop/http/test/streameye $ sudo make install
cp streameye /usr/local/bin
pi@raspberrypi:~/Desktop/http/test/streameye $ cd ..
pi@raspberrypi:~/Desktop/http/test $ python rasp_test.py | streameye
2022-02-13 22:22:06: INFO : streamEye 0.9
2022-02-13 22:22:06: INFO : hello!
2022-02-13 22:22:06: INFO : listening on 0.0.0.0:8080
2022-02-13 22:22:12: INFO : new client connection from 172.16.216.60:51296
reading client request
>>>>>> read_request result: 0
writing response header
2022-02-13 22:22:18: ERROR: 172.16.216.60:51296: write() failed errno == EPIPE || errno == EINTR: Broken pipe
2022-02-13 22:22:18: INFO : 172.16.216.60:51296: connection closed after writing multipart header
Total client now is: 1
Total client after cleanning is: 0
2022-02-13 22:23:22: INFO : new client connection from 172.16.216.60:51301
reading client request
>>>>>> read_request result: 0
writing response header
2022-02-13 22:23:24: ERROR: 172.16.216.60:51301: write() failed errno == EPIPE || errno == EINTR: Broken pipe
2022-02-13 22:23:24: INFO : 172.16.216.60:51301: connection closed after writing multipart header
Total client now is: 1
Total client after cleanning is: 0
```



```
cc -Wall -pthread -O2 -D_GNU_SOURCE -o streameye streameye.o client.o auth.o
pi@raspberrypi:~/Desktop/http/test/streameye $ sudo make install
cp streameye /usr/local/bin
pi@raspberrypi:~/Desktop/http/test/streameye $ cd ..
pi@raspberrypi:~/Desktop/http/test $ python rasp_test.py | streameye
2022-02-13 22:22:06: INFO : streamEye 0.9
2022-02-13 22:22:06: INFO : hello!
2022-02-13 22:22:06: INFO : listening on 0.0.0.0:8080
2022-02-13 22:22:12: INFO : new client connection from 172.16.216.60:51296
reading client request
>>>>>> read_request result: 0
writing response header
2022-02-13 22:22:18: ERROR: 172.16.216.60:51296: write() failed errno == EPIPE || errno == EINTR: Broken pipe
2022-02-13 22:22:18: INFO : 172.16.216.60:51296: connection closed after writing multipart header
Total client now is: 1
Total client after cleanning is: 0
2022-02-13 22:23:22: INFO : new client connection from 172.16.216.60:51301
reading client request
>>>>>> read_request result: 0
writing response header
2022-02-13 22:23:24: ERROR: 172.16.216.60:51301: write() failed errno == EPIPE || errno == EINTR: Broken pipe
2022-02-13 22:23:24: INFO : 172.16.216.60:51301: connection closed after writing multipart header
Total client now is: 1
Total client after cleanning is: 0
```

Finally, adding the signal part, problem fixed

```
#include <signal.h> //singal

/*
#define INFO(fmt, ...)          fprintf(stderr, "%s: INFO : " fmt "\n", str_timestamp(), ##__VA_ARGS__)
#define ERROR(fmt, ...)         fprintf(stderr, "%s: ERROR: " fmt "\n", str_timestamp(), ##__VA_ARGS__)
#define ERRNO(msg)              ERROR("%s: %s", msg, strerror(errno))
#define ERROR_CLIENT(client, fmt, ...)  ERROR("%s:%d: " fmt, client->addr, client->port, ##__VA_ARGS__)
#define ERRNO_CLIENT(client, msg)       ERROR_CLIENT(client, "%s: %s", msg, strerror(errno))
*/

int main(int argc, char *argv[]){
******
/* signals */
   DEBUG("installing signal handlers");
   struct sigaction act;
   act.sa_handler = bye_handler;
   act.sa_flags = 0;
   sigemptyset(&act.sa_mask);

   if (sigaction(SIGINT, &act, NULL) < 0) {
      //ERRNO("sigaction() failed");
      return -1;
   }
   if (sigaction(SIGTERM, &act, NULL) < 0) {
      //ERRNO("sigaction() failed");
      return -1;
   }
   if (signal(SIGPIPE, SIG_IGN) == SIG_ERR) {
      //ERRNO("signal() failed");
      return -1;
   }

.......
}

void bye_handler(int signal) {
   if (!running) {
      //INFO("interrupt already received, ignoring signal");
      return;
   }

   //INFO("interrupt received, quitting");
   running = 0;
}

char *str_timestamp() {
   static __thread char s[20];

   time_t t = time(NULL);
   struct tm *tmp = localtime(&t);

   strftime(s, sizeof(s), "%Y-%m-%d %H:%M:%S", tmp);

   return s;
}
```



```
    waiter.acquire()
KeyboardInterrupt
pi@raspberrypi:~/Desktop/http/test $ python rasp_test.py | ./test001
new client connection from 172.16.216.60:51419
Total clients now after accept: 1
reading client request
read_request from client 172.16.216.60:51419
>>>>>> read_request result: 0
writing response header
2022-02-13 22:42:05: ERROR: 172.16.216.60:51419: write() failed errno == EPIPE || errno == EINTR: Broken pipe
errno == EPIPE || errno == EINTR
connection closed after write_multipart_header
Total client now is: 1
cleaning up client: 172.16.216.60:51419
find client and remove, i = 0
Total client after cleanning is: 0
new client connection from 172.16.216.60:51421
Total clients now after accept: 1
reading client request
read_request from client 172.16.216.60:51421
>>>>>> read_request result: 0
writing response header
2022-02-13 22:42:14: ERROR: 172.16.216.60:51421: write() failed errno == EPIPE || errno == EINTR: Broken pipe
errno == EPIPE || errno == EINTR
connection closed after write_multipart_header
Total client now is: 1
cleaning up client: 172.16.216.60:51421
find client and remove, i = 0
Total client after cleanning is: 0
^C2022-02-13 22:46:00: INFO : interrupt received, quitting
closing server
waiting for clients to finish
End here: 100
Traceback (most recent call last):
  File "rasp_test.py", line 23, in <module>
    camera.capture_sequence(streams_iter(), format='jpeg', use_video_port= True, thumbnail=None, quality=50)
  File "/usr/lib/python2.7/dist-packages/picamera/camera.py", line 1519, in capture_sequence
    encoder.wait()
```

I later found that I don’t need to add so much code, I only need the following.

```
#include <signal.h> //singal
#include <errno.h>
```

```
if (signal(SIGPIPE, SIG_IGN) == SIG_ERR) {
	//ERRNO("signal() failed");
	return -1;
 }
//above code means ignore SIGPIPE
```

SIGPIPE is for situations like this:

Code:

```
$ grep "pattern" < reallyhugefile | head
```

grep might print millions of lines, but head only reads 10 then quits. Once head closes the read-end and quits, grep gets SIGPIPE, which kills it, forcing it to quit early instead of processing the entire file uselessly.

If you don't want your program to be killed, handle or block SIGPIPE yourself. You will start getting write-errors with errno set to EPIPE instead.

```
seq | head -n 1
```

The command from above creates two processes, which are connected by a <man:pipe(2)>. `seq` writes its infinite sequence of numbers to `STDOUT`, while `head` reads the other end of the pipe as `STDIN`. It reads the first line and then exits. But what stops `seq` from running until the collapse of the universe?
The Linux kernel only allocates a finite sized buffer for that pipe. The size of that buffer changed over time from *4 KiB* to *64 KiB* to *configurable*, but still defaults to 1 MiB. See <man:pipe(7)> for more details about the getting the size.
After `seq` filled up that buffer its next call to <man:write(2)> will block until the reader has read some data and thus has freed some space in the buffer. But as soon as `head` terminated, there will never be any other reader who can do that. The Linux kernel thus sends `SIGPIPE` to `seq` to signal it, that no reader is left. The default action for that signal is *terminate* the process.

If the calling process is ignoring SIGPIPE, then <man:write(2)> fails with the error EPIPE.

Reference:
Broken pipe: https://linuxpip.org/broken-pipe-python-error/
close failed in file object destructor: https://stackoverflow.com/questions/42722411/errors-at-python-program-exit-close-failed-in-file-object-destructor-sys-ex
signal: https://www.tutorialspoint.com/c_standard_library/c_function_signal.htm
SIGPIPE and EPIPE : https://www.unix.com/programming/171395-sigpipe-epipe.html
SIGPIPE, EPIPE: https://pmhahn.github.io/SIGPIPE/
Why does SIGPIPE exist: https://stackoverflow.com/questions/8369506/why-does-sigpipe-exist/9337925

Problem: Cannot find any user defined variable called errno
Answer: 1. The <errno.h> header file defines the integer variable errno, which is set by system calls and some library functions in

the event of an error to indicate what went wrong.

2. The following is the list of output from errno -l

```
  1  EPERM            Operation not permitted
  2  ENOENT            No such file or directory
  3  ESRCH            No such process
  4  EINTR            Interrupted system call
  5  EIO            Input/output error
  6  ENXIO            No such device or address
  7  E2BIG            Argument list too long
  8  ENOEXEC            Exec format error
  9  EBADF            Bad file descriptor
 10  ECHILD            No child processes
 11  EAGAIN            Resource temporarily unavailable
 11  EWOULDBLOCK        Resource temporarily unavailable
 12  ENOMEM            Cannot allocate memory
 13  EACCES            Permission denied
 14  EFAULT            Bad address
 15  ENOTBLK            Block device required
 16  EBUSY            Device or resource busy
 17  EEXIST            File exists
 18  EXDEV            Invalid cross-device link
 19  ENODEV            No such device
 20  ENOTDIR            Not a directory
 21  EISDIR            Is a directory
 22  EINVAL            Invalid argument
 23  ENFILE            Too many open files in system
 24  EMFILE            Too many open files
 25  ENOTTY            Inappropriate ioctl for device
 26  ETXTBSY            Text file busy
 27  EFBIG            File too large
 28  ENOSPC            No space left on device
 29  ESPIPE            Illegal seek
 30  EROFS            Read-only file system
 31  EMLINK            Too many links
 32  EPIPE            Broken pipe
 33  EDOM            Numerical argument out of domain
 34  ERANGE            Numerical result out of range
 35  EDEADLK            Resource deadlock avoided
 35  EDEADLOCK          Resource deadlock avoided
 36  ENAMETOOLONG       File name too long
 37  ENOLCK            No locks available
 38  ENOSYS            Function not implemented
 39  ENOTEMPTY          Directory not empty
 40  ELOOP            Too many levels of symbolic links
 42  ENOMSG            No message of desired type
 43  EIDRM            Identifier removed
 44  ECHRNG            Channel number out of range
 45  EL2NSYNC            Level 2 not synchronized
 46  EL3HLT            Level 3 halted
 47  EL3RST            Level 3 reset
 48  ELNRNG            Link number out of range
 49  EUNATCH            Protocol driver not attached
 50  ENOCSI            No CSI structure available
 51  EL2HLT            Level 2 halted
 52  EBADE            Invalid exchange
 53  EBADR            Invalid request descriptor
 54  EXFULL            Exchange full
 55  ENOANO            No anode
 56  EBADRQC            Invalid request code
 57  EBADSLT            Invalid slot
 59  EBFONT            Bad font file format
 60  ENOSTR            Device not a stream
 61  ENODATA            No data available
 62  ETIME            Timer expired
 63  ENOSR            Out of streams resources
 64  ENONET            Machine is not on the network
 65  ENOPKG            Package not installed
 66  EREMOTE            Object is remote
 67  ENOLINK            Link has been severed
 68  EADV            Advertise error
 69  ESRMNT            Srmount error
 70  ECOMM            Communication error on send
 71  EPROTO            Protocol error
 72  EMULTIHOP          Multihop attempted
 73  EDOTDOT            RFS specific error
 74  EBADMSG            Bad message
 75  EOVERFLOW          Value too large for defined data type
 76  ENOTUNIQ           Name not unique on network
 77  EBADFD            File descriptor in bad state
 78  EREMCHG            Remote address changed
 79  ELIBACC            Can not access a needed shared library
 80  ELIBBAD            Accessing a corrupted shared library
 81  ELIBSCN            .lib section in a.out corrupted
 82  ELIBMAX            Attempting to link in too many shared libraries
 83  ELIBEXEC           Cannot exec a shared library directly
 84  EILSEQ            Invalid or incomplete multibyte or wide character
```

85	ERESTART	Interrupted system call should be restarted
86	ESTRPIPE	Streams pipe error
87	EUSERS	Too many users
88	ENOTSOCK	Socket operation on non-socket
89	EDESTADDRREQ	Destination address required
90	EMSGSIZE	Message too long
91	EPROTOTYPE	Protocol wrong type for socket
92	ENOPROTOOPT	Protocol not available
93	EPROTONOSUPPORT	Protocol not supported
94	ESOCKTNOSUPPORT	Socket type not supported
95	ENOTSUP	Operation not supported
95	EOPNOTSUPP	Operation not supported
96	EPFNOSUPPORT	Protocol family not supported
97	EAFNOSUPPORT	Address family not supported by protocol
98	EADDRINUSE	Address already in use
99	EADDRNOTAVAIL	Cannot assign requested address
100	ENETDOWN	Network is down
101	ENETUNREACH	Network is unreachable
102	ENETRESET	Network dropped connection on reset
103	ECONNABORTED	Software caused connection abort
104	ECONNRESET	Connection reset by peer
105	ENOBUFS	No buffer space available
106	EISCONN	Transport endpoint is already connected
107	ENOTCONN	Transport endpoint is not connected
108	ESHUTDOWN	Cannot send after transport endpoint shutdown
109	ETOOMANYREFS	Too many references: cannot splice
110	ETIMEDOUT	Connection timed out
111	ECONNREFUSED	Connection refused
112	EHOSTDOWN	Host is down
113	EHOSTUNREACH	No route to host
114	EALREADY	Operation already in progress
115	EINPROGRESS	Operation now in progress
116	ESTALE	Stale file handle
117	EUCLEAN	Structure needs cleaning
118	ENOTNAM	Not a XENIX named type file
119	ENAVAIL	No XENIX semaphores available
120	EISNAM	Is a named type file
121	EREMOTEIO	Remote I/O error
122	EDQUOT	Disk quota exceeded
123	ENOMEDIUM	No medium found
124	EMEDIUMTYPE	Wrong medium type
125	ECANCELED	Operation canceled
126	ENOKEY	Required key not available
127	EKEYEXPIRED	Key has expired
128	EKEYREVOKED	Key has been revoked
129	EKEYREJECTED	Key was rejected by service
130	EOWNERDEAD	Owner died
131	ENOTRECOVERABLE	State not recoverable
132	ERFKILL	Operation not possible due to RF-kill
133	EHWPOISON	Memory page has hardware error

Reference:

1. https://stackoverflow.com/questions/503878/how-to-know-what-the-errno-means
2. https://man7.org/linux/man-pages/man3/errno.3.html

Problem: Error - undefined reference to 'pthread_create' with C program in GCC Linux

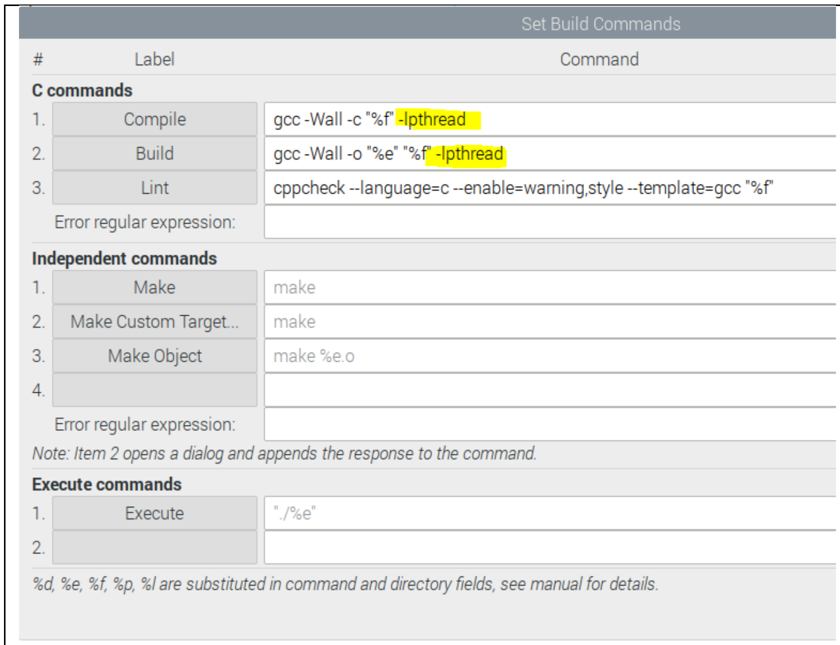
Answer

1. Include Header file

```
#include <stdio.h>
#include <pthread.h>
```

2. Compile command

```
gcc  main.c -o main -lpthread
```



Set Build Commands

#	Label	Command
C commands		
1.	Compile	gcc -Wall -c "%f" -lpthread
2.	Build	gcc -Wall -o "%e" "%f" -lpthread
3.	Lint	cppcheck --language=c --enable=warning,style --template=gcc "%f"
	Error regular expression:	
Independent commands		
1.	Make	make
2.	Make Custom Target...	make
3.	Make Object	make %e.o
4.		
	Error regular expression:	

Note: Item 2 opens a dialog and appends the response to the command.

#	Label	Command
Execute commands		
1.	Execute	"./%e"
2.		

%d, %e, %f, %p, %l are substituted in command and directory fields, see manual for details.

Reference:
https://www.includehelp.com/c-programming-questions/error-undefined-reference-to-pthread-create-in-linux.aspx

Problem:
1. My http server program stuck at accept function.
2. After accept client, reply with some jpeg data to client, my code stuck at pthread_cond_wait(…). It doesn't go back to main thread to process the camera input data.

Answer:
accept is a blocking call unless you specify the socket to be nonblocking. You can achieve this with the following:

```
fcntl(sock_desc, F_SETFL, fcntl(sock_desc, F_GETFL, 0) | O_NONBLOCK);
```

You can do error checking with the return value from fcntl.

Actually I forgot to copy this part of code from streameye.c

Reference:
https://stackoverflow.com/questions/30733924/server-program-gets-stuck-at-accept-function/30734811

Problem: Difference between pthread and fork on gnu/Linux

Answer:
In C there are some differences however:

fork()

- Purpose is to create a new process, which becomes the child process of the caller
- Both processes will execute the next instruction following the fork() system call
- Two identical copies of the computer's address space,code, and stack are created one for parent and child.

Thinking of the fork as it was a person; Forking causes a clone of your program (process), that is running the code it copied.

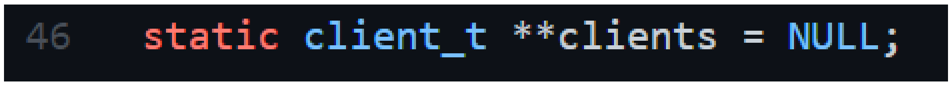
pthread_create()

- Purpose is to create a new thread in the program which is given the same process of the caller
- Threads within the same process can communicate using shared memory. (Be careful!)
- The second thread will share data,open files, signal handlers and signal dispositions, current working directory, user and group ID's. The new thread will get its own stack, thread ID, and registers though.

Continuing the analogy; your program (process) grows a second arm when it creates a new thread, connected to the same brain.

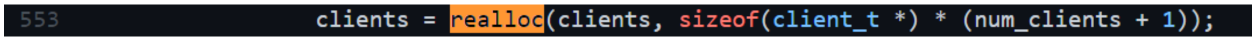
Reference:
https://stackoverflow.com/questions/5514464/difference-between-pthread-and-fork-on-gnu-linux

Problem: in streameye.c, it declare clients variable with NULL value. Later on, it can use clients[i] to access different client data



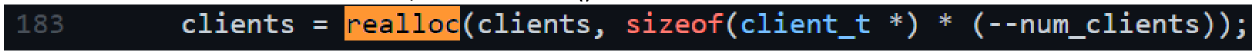
```
static client_t **clients = NULL;
```

When there is new client coming in, streameye.c only use realloc() function



```
                clients = realloc(clients, sizeof(client_t *) * (num_clients + 1));
```

When there is client disconnected, it use realloc() function



```
        clients = realloc(clients, sizeof(client_t *) * (--num_clients));
```

Answer
From Open Groups' specifications (https://pubs.opengroup.org/onlinepubs/009695399/functions/realloc.html):

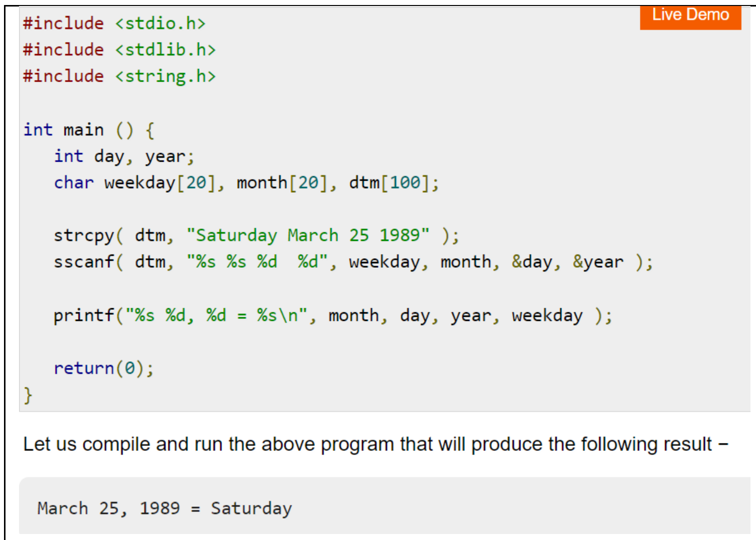
If ptr is a null pointer, realloc() shall be equivalent to malloc() for the specified size.

If ptr does not match a pointer returned earlier by calloc(), malloc(), or realloc() or if the space has previously been deallocated by a call to free() or realloc(), the behavior is undefined.

Reference:
https://stackoverflow.com/questions/4459275/is-a-malloc-needed-before-a-realloc
Dynamic allocate array: https://www.geeksforgeeks.org/dynamic-memory-allocation-in-c-using-malloc-calloc-free-and-realloc/

Problem: What is sscanf function

Answer:
int sscanf(const char *str, const char *format, ...) reads formatted input from a string.



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main () {
   int day, year;
   char weekday[20], month[20], dtm[100];

   strcpy( dtm, "Saturday March 25 1989" );
   sscanf( dtm, "%s %s %d  %d", weekday, month, &day, &year );

   printf("%s %d, %d = %s\n", month, day, year, weekday );

   return(0);
}
```

Let us compile and run the above program that will produce the following result −

```
March 25, 1989 = Saturday
```

Reference:
https://www.tutorialspoint.com/c_standard_library/c_function_sscanf.htm

Problem: How do I share variables between different .c files

Answer:
In fileA.c:

```
int myGlobal = 0;
```

In fileA.h

```
extern int myGlobal;
```

In fileB.c:

```
#include "fileA.h"
myGlobal = 1;
```

So this is how it works:

- the variable lives in fileA.c
- fileA.h tells the world that it exists, and what its type is (int)
- fileB.c includes fileA.h so that the compiler knows about myGlobal before fileB.c tries to use it.

Reference:
https://stackoverflow.com/questions/1045501/how-do-i-share-variables-between-different-c-files

Problem: How does makefile work

Answer:

Reference:

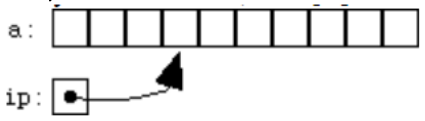
https://opensource.com/article/18/8/what-how-makefile

Problem: What does it mean pointer plus/minus integer

Answer:
This question is about Pointer Arithmetic.
Pointers do not have to point to single variables. They can also point at the cells of an array. For example, we can write

```
int *ip;
int a[10];
ip = &a[3];
```

and we would end up with ip pointing at the fourth cell of the array a (remember, arrays are 0-based, so a[0] is the first cell). We could illustrate the situation like this:



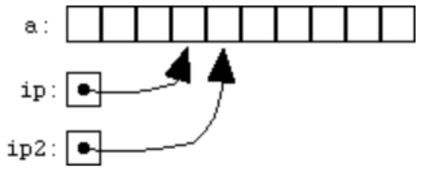
We'd use this ip just like the one in the previous section: *ip gives us what ip points to, which in this case will be the value in a[3].
Once we have a pointer pointing into an array, we can start doing pointer arithmetic. Given that ip is a pointer to a[3], we can add 1 to ip:

```
ip + 1
```

What does it mean to add one to a pointer? In C, it gives a pointer to the cell one farther on, which in this case is a[4]. To make this clear, let's assign this new pointer to another pointer variable:

```
ip2 = ip + 1;
```

Now the picture looks like this:



If we now do

```
*ip2 = 4;
```

we've set a[4] to 4. But it's not necessary to assign a new pointer value to a pointer variable in order to use it; we could also compute a new pointer value and use it immediately:

```
*(ip + 1) = 5;
```

In this last example, we've changed a[4] again, setting it to 5. The parentheses are needed because the unary ``contents of'' operator * has higher precedence (i.e., binds more tightly than) the addition operator. If we wrote *ip + 1, without the parentheses, we'd be fetching the value pointed to by ip, and adding 1 to that value. The expression *(ip + 1), on the other hand, accesses the value one past the one pointed to by ip.

Given that we can add 1 to a pointer, it's not surprising that we can add and subtract other numbers as well.

Of course, pointers are not limited to ints. It's quite common to use pointers to other types, especially char.
One question that comes up is whether the expression *p++ increments p or what it points to. The answer is that it increments p. To increment what p points to, you can use (*p)++.

When you're doing pointer arithmetic, you have to remember how big the array the pointer points into is, so that you don't ever point outside it.

Let's see other code

```
#include<stdio.h>
#include<string.h>
#include<conio.h>

main()
{
char s[30], t[20];
char *found;

/* Entering the main string */
puts("Enter the first string: ");
gets(s);

/* Entering the string whose position or index to be displayed */
puts("Enter the string to be searched: ");
gets(t);

/*Searching string t in string s */
found=strstr(s,t);
if(found)
   printf("Second String is found in the First String at %d position.\n", found - s);
else
   printf("-1");
getch();
}
```

Assuming you're wondering about the expression found-s, then what's happening is that you subtract two pointers.

Arrays naturally decay to pointers to their first element. That means plain s is equal to &s[0], which is what's happening here: found-s is equal to found - (&s[0]).

And the subtraction works because found is pointing to an element inside the array s, so the pointers are related (which is a requirement for pointer subtraction). The result is the difference (in elements) between the two pointers.

Reference:
https://www.eskimo.com/~scs/cclass/notes/sx10b.html
https://stackoverflow.com/questions/60095585/how-can-a-character-array-be-subtracted-from-a-pointer

Problem: How to parse MJPEG file

Answer:
Since each JPEG starts with **0xFF 0xD8** as Start of Image marker and ends with **0xFF 0xD9**.

When processing multipart/x-mixed-replace, what you are *supposed* to do is:

1. read and discard the HTTP response body until you reach the first MIME boundary specified by the Content-Type response header.
2. then read a MIME entity's headers and data until you reach the next matching MIME boundary.
3. then process the entity's data as needed, according to its headers (for instance, displaying a image/jpeg entity onscreen).
4. if the connection has not been closed, and the last boundary read is not the termination boundary, go back to 2, otherwise stop processing the HTTP response.

Reference:
https://stackoverflow.com/questions/47729941/mjpeg-over-http-specification

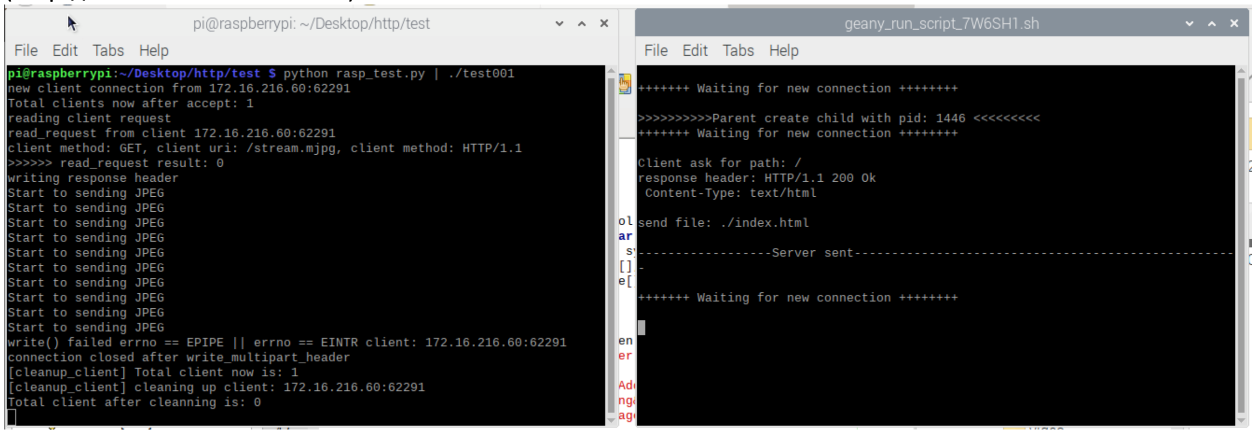
Problem: Want to use command line to execute python file.
1. Bash: syntax error near unexpected token '(' – Python
2. –bash: ./manage.py: Permision denied

Answer:
1. add #!/usr/bin/env python at the top of your script,
or call your script using **python myscript.py**

2. You need to make manage.py executable to excecute it.
Do chmod +x manage.py to make it excecutable. Alternately you can do python manage.py <cmd> instead.

Reference:
https://stackoverflow.com/questions/10676050/bash-syntax-error-near-unexpected-token-python/10676069

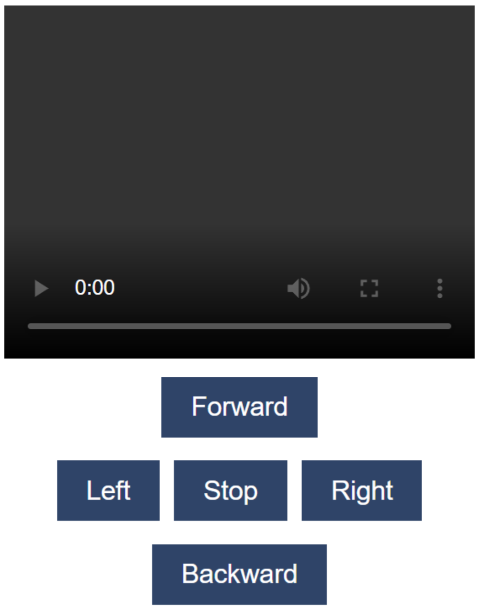
Problem: There are two program running. One is serving the web page (172.16.216.206:8081), one is serving the MJPEG (http://172.16.216.206:8085).


pi@raspberrypi: ~/Desktop/http/test
File Edit Tabs Help
geany_run_script_7W6SH1.sh
File Edit Tabs Help

```
pi@raspberrypi:~/Desktop/http/test $ python rasp_test.py | ./test001
new client connection from 172.16.216.60:62291
Total clients now after accept: 1
reading client request
read_request from client 172.16.216.60:62291
client method: GET, client uri: /stream.mjpg, client method: HTTP/1.1
>>>>>> read_request result: 0
writing response header
Start to sending JPEG
Start to sending JPEG
Start to sending JPEG
Start to sending JPEG
Start to sending JPEG
Start to sending JPEG
Start to sending JPEG
Start to sending JPEG
Start to sending JPEG
Start to sending JPEG
write() failed errno == EPIPE || errno == EINTR client: 172.16.216.60:62291
connection closed after write_multipart_header
[cleanup_client] Total client now is: 1
[cleanup_client] cleaning up client: 172.16.216.60:62291
Total client after cleanning is: 0
```

```
+++++++ Waiting for new connection ++++++++

>>>>>>>>>>Parent create child with pid: 1446 <<<<<<<<<
+++++++ Waiting for new connection ++++++++

Client ask for path: /
response header: HTTP/1.1 200 Ok
 Content-Type: text/html

send file: ./index.html

-----------------Server sent--------------------------------------------------------
-

+++++++ Waiting for new connection ++++++++
```

After connect, the video player will keep spinning for couple of seconds and then stop.



The browser

Request URL: http://172.16.216.206:8085/
Request Method: GET
Status Code: 200 OK

```
Remote Address: 172.16.216.206:8085
Referrer Policy: strict-origin-when-cross-origin
Cache-Control: no-cache, private
Connection: close
Content-Type: multipart/x-mixed-replace; boundary=--FrameBoundary
Expires: 0
Max-Age: 0
Pragma: no-cache
Server: RaspberryPi/1.0.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/sig
ned-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,zh-TW;q=0.8,zh;q=0.7
Connection: keep-alive
Host: 172.16.216.206:8085
sec-gpc: 1
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/98.0.4758.102 Safari/537.36
```

Answer:
So finally, this problem is nothing to do with CORS. The problem is at the index.html. At first, I use the following to request for my MJPEG file

```
<video src=http://172.16.216.206:8085/stream.mjpg></video>
```

Then I change it to the following

```
<img src="http://172.16.216.206:8085/stream.mjpg" width="320" height="240">
```

Problem fix.

Reference:
https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

> The Cross-Origin Resource Sharing standard works by adding new HTTP headers that let servers describe which origins are permitted to read that information from a web browser. Additionally, for HTTP request methods that can cause side-effects on server data (in particular, HTTP methods other than `GET`, or `POST` with certain MIME types), the specification mandates that browsers "preflight" the request, soliciting supported methods from the server with the HTTP `OPTIONS` request method, and then, upon "approval" from the server, sending the actual request. Servers can also inform clients whether "credentials" (such as Cookies and HTTP Authentication) should be sent with requests.
>
> The only way to determine what specifically went wrong is to look at the browser's console for details.

Problem:

Answer:

Reference: