```
"""Run with 'python server.py <port number>'"""

from records import FileRequest, FileResponse, ENCODING_TYPE
import socket
from common import *
import time
import sys


def get_server_port_number():
    """Parses passed command line arguments to get the port number."""
    # Get cammand line arg and check that is exists
    try:
        port_num_str = sys.argv[1].strip()
    except IndexError:
        error(MISSING_ARG_ERR)

    return convert_portno_str(port_num_str)


def build_file_response(file_name):
    if file_exists_locally(file_name):
        status_code = 1
        file_bytearray = bytearray(open(file_name, "rb").read())

    else:  # File doesn't exist locally
        status_code = 0
        file_bytearray = bytearray(0)

    print("Outgoing file len:", len(file_bytearray))
    return FileResponse(file_bytearray, status_code)


def server():
    port_num = get_server_port_number()

    # Create and Bind
    try:
        sockfd = socket.socket()
        sockfd.settimeout(TIMEOUT)
        sockfd.bind((LOCAL_HOST, port_num))
    except socket.gaierror:
        sockfd.close()
        error(COULDNT_BIND_ERR)

    # Listen
    try:
        sockfd.listen()
    except socket.gaierror:  # What error should this be?
        sockfd.close()
        error(SOCKET_LISTEN_ERR)


    client_socket = None
    try:
        # Accept incomming requests
        while True:

            # Accept incomming connection request
            client_socket, client_addr = sockfd.accept()
            client_socket.settimeout(TIMEOUT)

            # Recieve header from connection
            client_request_header = client_socket.recv(FileRequest.header_byte_len()
)
            if len(client_request_header) != FileRequest.header_byte_len():
                error(INVALID_FILE_REQUEST_ERR)

            # Check header validity
            if not FileRequest.is_valid_header(client_request_header):
                error(INVALID_FILE_REQUEST_ERR)
            # Extract filenameLen from header
            n = FileRequest.get_filenameLen_from_header(client_request_header)
```

```
            # Read just the filename from socket
            filename_bytes = client_socket.recv(n)
            filename = filename_bytes.decode(ENCODING_TYPE)

            # Check length of recieved bytes
            if len(filename_bytes) != n:
                error(INVALID_FILE_REQUEST_ERR)

            # Make a FileResponse object and send it to the client
            file_response = build_file_response(filename)
            d = file_response.get_bytearray()
            client_socket.send(d)


    except socket.timeout:
        error(TIMOUT_ERR)
    finally:
        if client_socket is not None:
            client_socket.close()
        sockfd.close()


server()
```