

TCP Multithreaded Client/Server Design Document

Note: The code and (in-line) documentation for this project can also be found at the following GitHub link: <https://github.com/ariblack17/simple-server>.

Description

The Server: The server opens a serverSocket on a non-reserved port (12000), listens for a client, then waits for a wildcard query from that client once a connection has been established. Each client is handled on a separate thread, and each client is able to send multiple queries to the server for multiple responses.

The wildcard query supported in this server follows the format: $x?x$, where x is any combination of ASCII characters and $?$ is any single letter, or a wildcard. There may be multiple wildcard characters within each query. The query target is a text file locally hosted in relation to the server.

The Client: The client opens a clientSocket selected by the operating system, connects to the server, sends a wildcard query request, then receives and unpacks a response from the server. This process of sending and receiving messages repeats until the client disconnects.

Application Protocol: The request from the client contains a command, and the response from the server contains a status code, status message (indicating the number of words matching the query), and a body (containing an array of all matching words).

A client request is formatted as follows:

<i>Command</i>	→	<i>FIND <query></i>
<i>Bytes header</i>	→	<i>Bytes: <n></i>
<i>Connection-type header</i>	→	<i>Connection-type: multi-request</i>

Note that the Connection-type header does not have much use in the current implementation of this client/server architecture; mainly it was added to better model other popular application protocols, like HTTP. It does allow developers to see from the server's console which type of client has connected to the server (client programs intended for the basic single-threaded server or client programs intended for the multithreaded implementation).

The server response is formatted as follows:

<i>Status code (11 or 00)</i>	→	<i>Code 11</i>
<i>Status message (n: num matches)</i>	→	<i>Success: Found <n> matches for <query>.</i>
<i>Body (matches)</i>	→	<i>[<w1>, <w2>, ..., <wn>]</i>

If no matches are found, the status code returned is 00 and the status message is *No matches found.*

Trade Offs

Multithreaded, not Multi-Socketed: Although it would be overall better design to have one welcoming (server) socket and one individual connection socket for each incoming client connection to the server, since this was not required by the assignment – and would instead make a good project extension – it was not implemented here.

Program Structure: For ease of development, the design of the server and client programs was kept relatively simple. The decision to structure the program as an array of helper and handler functions invoked by some type of driver code (rather than a more object-oriented design) was meant to reduce complexity during development, and not necessarily to improve code optimization or readability.

Client Disconnect: For this assignment, client disconnect could be implemented in two major ways: disconnecting on the client's end without notifying the server (so the server must independently recognize that the client has terminated the connection), or disconnecting on the client's end after notifying the server. While the former might be more optimal – as it requires less messages to be exchanged between the client and server processes – the latter was chosen here because of its more straightforward (and its evidently functional) implementation.

Extensions

Multi-Socketed Design: As discussed in *Trade Offs* above, taking the time to establish a separate socket for each incoming client connection would allow us to more accurately claim that the server can serve multiple clients in parallel. Multithreading is good, but it is made better with multiple client sockets as well.

Improved Structure and Optimization: As discussed above, the server and client code (and the application protocol) could be re-examined to improve the structure and improve the overall programs' time and space complexities. Use of Python's *pandas* package early in the applications' development was a conscious first-step made to avoid lengthy computations, but there certainly still remains room for improvement.

Improving Client Disconnect: As discussed above, the only way for the client to disconnect from the server (without breaking the server) is to send a request for disconnect. Removing this requirement would reduce the volume of message exchange between clients, and it would improve the server's reliability (as the server would then be able to catch errors and continue running should client processes terminate or disconnect unexpectedly).

Generating Better "UI" for the Client: Currently, the client's interface with the server is exclusively served through the command line and console. Work could be done to create a GUI to improve usability on the client-side, or the console output could be improved so that server responses are more readable or so that users may see all of their query options as they write their requests (wildcard ? and ASCII characters, as well as any other additions).

Test Cases

Structure: The following Client Queries section follows a simple structure – first (in bold/italics) is a line containing the client queries sent to the server for the given test case. Each query is separated by |. Below the query line is a brief description of the test cases' functionalities, and below this descriptor a screenshot showing the programs' outputs for the given queries.

The left half of the screenshot contains a terminal running the server process, and the right half contains two separate terminals running two client processes.

Note that screenshots showing proof of functionality and testing for the basic client/server programs can be found in the GitHub repository linked at the start of this document.

Client Queries:

????????? | ? | *quit*

Test server response for 10 wildcard characters, 1 wildcard character, and client disconnect.

[illegible]

(a) | -?

Test server response for a query including special characters (parentheses and dashes).

```
simple-server -- python3 tcp/multithread-server.py
~/git/simple-server python3 tcp/multithread-server.py ...
'morning(a)' 'mortal(a)' 'most(a)' 'motive(a)' 'motive(a)'
'mountain(a)' 'much(a)' 'must(a)' 'nazy(a)' 'near(a)' 'new(a)'
'nickel-and-dime(a)' 'not(a)' 'nonprescription(a)' 'nonsense(a)'
'occasional(a)' 'odd-job(a)' 'old(a)' 'omnibus(a)' 'on-line(a)'
'on-the-spot(a)' 'one(a)' 'one-man(a)' 'open(a)' 'other(a)' 'out(a)'
'outdoor(a)' 'outer(a)' 'outliving(a)' 'outside(a)' 'own(a)' 'pacifist(a)'
'pampering(a)' 'part(a)' 'particular(a)' 'party(a)' 'passing(a)'
'past(a)' 'peculiar(a)' 'pedigree(a)' 'pinpoint(a)' 'plant-eating(a)'
'preceding(a)' 'precipitating(a)' 'premier(a)' 'prescription(a)'
'present(a)' 'pressure-cooker(a)' 'presumed(a)' 'previous(a)' 'prime(a)'
'professed(a)' 'proper(a)' 'prospective(a)' 'protecting(a)'
'protesting(a)' 'prot(a)' 'pumped-up(a)' 'putative(a)' 'quack(a)'
'quarter(a)' 'quasi(a)' 'quick-change(a)' 'quickset(a)' 'radiating(a)'
'raised(a)' 'ranking(a)' 'raw(a)' 'ready(a)' 'real(a)' 'rear(a)'
'rebel(a)' 'receding(a)' 'recovered(a)' 'redeeming(a)' 'regular(a)'
'repulsive(a)' 'reserve(a)' 'respective(a)' 'returning(a)' 'review(a)'
'right(a)' 'right-hand(a)' 'rightful(a)' 'road(a)' 'running(a)' 'rush(a)'
'salt(a)' 'screaming(a)' 'sea(a)' 'seaboard(a)' 'seashore(a)'
'secondary(a)' 'senile(a)' 'senior(a)' 'servile(a)' 'several(a)'
'simmering(a)' 'shrilling(a)' 'side(a)' 'single(a)' 'slave(a)'
'sleeping(a)' 'small(a)' 'sneaking(a)' 'sniffing(a)' 'sold-out(a)'
'some(a)' 'sophomore(a)' 'sourdo(a)' 'speaking(a)' 'specialist(a)'
'spring(a)' 'square(a)' 'stammering(a)' 'standing(a)' 'stay-at-home(a)'
'stern(a)' 'straightarrow(a)' 'stretch(a)' 'stretching(a)'
'subordinating(a)' 'succeeding(a)' 'such(a)' 'summer(a)' 'sunrise(a)'
'sunset(a)' 'supposed(a)' 'switch(a)' 'tail(a)' 'talking(a)' 'tall(a)'
'tarred-and-feathered(a)' 'then(a)' 'thieving(a)' 'third(a)' 'through(a)'
'top(a)' 'trade(a)' 'trans(a)' 'transsexual(a)' 'true(a)'
'true-to-life(a)' 'trumped-up(a)' 'twinkling(a)' 'under(a)'
'underclass(a)' 'up(a)' 'upset(a)' 'utility(a)' 'very(a)' 'virtual(a)'
'waking(a)' 'walk-in(a)' 'walk-to(a)' 'welcome(a)' 'well-know(a)'
'well-wishing(a)' 'wide-screen(a)' 'windup(a)' 'winter(a)' 'working(a)'
'young-begetting(a)' 'zero(a)']
found 414 matches for \{\a\}.

received request:
FIND quit
Bytes: 4
Connection-type: multi-request

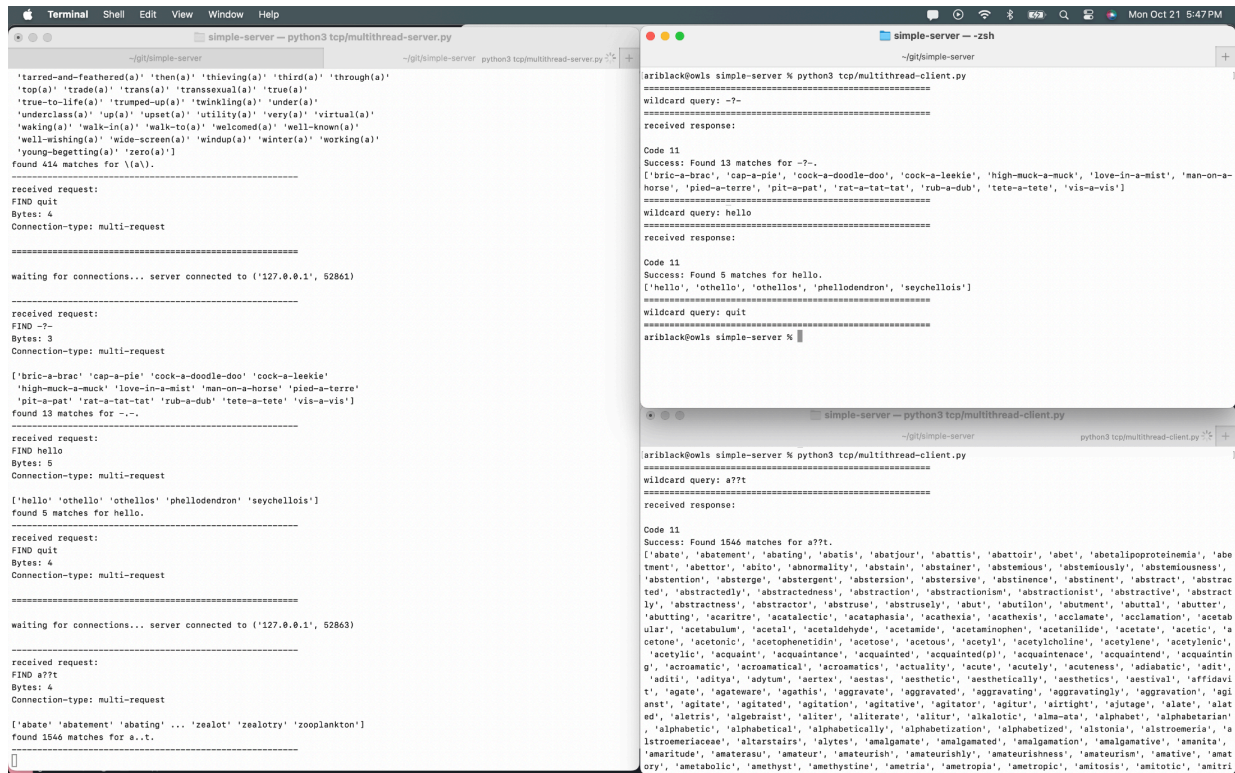
waiting for connections... server connected to ('127.0.0.1', 52863)

received request:
FIND -?
Bytes: 3
Connection-type: multi-request

['bric-a-brac' 'cap-a-pie' 'cock-a-doodle-doo' 'cock-a-leekie'
'high-muck-a-muck' 'love-in-a-mist' 'man-on-a-horse' 'pied-a-terre'
'pit-a-pat' 'rat-a-tat-tat' 'rub-a-dub' 'tete-a-tete' 'vis-a-vis']
found 13 matches for -?.
```

a??t | hello

Test server response for general case queries (1) including multiple wildcard and non-wildcard characters, and (2) including no wildcards.



```
Terminal Shell Edit View Window Help
simple-server -- python3 tcp/multithread-server.py
~/git/simple-server python3 tcp/multithread-server.py %
'tarred-and-feathered(a)' 'then(a)' 'thieving(a)' 'third(a)' 'through(a)'
'top(a)' 'trade(a)' 'trans(a)' 'transsexual(a)' 'true(a)'
'true-to-life(a)' 'trumped-up(a)' 'twinkling(a)' 'under(a)'
'underclass(a)' 'up(a)' 'upset(a)' 'utility(a)' 'very(a)' 'virtual(a)'
'waking(a)' 'walk-in(a)' 'walk-to(a)' 'welcomed(a)' 'well-known(a)'
'well-wishing(a)' 'wide-screen(a)' 'windup(a)' 'winter(a)' 'working(a)'
'young-begging(a)' 'zero(a)'}
found 414 matches for \{a\}.
-----
received request:
FIND quit
Bytes: 4
Connection-type: multi-request
-----
waiting for connections... server connected to ('127.0.0.1', 52861)
-----
received request:
FIND -?
Bytes: 3
Connection-type: multi-request
-----
['bric-a-brac' 'cap-a-pie' 'cock-a-doodle-doo' 'cock-a-leekie'
'high-muck-a-muck' 'love-in-a-mist' 'man-on-a-horse' 'pied-a-terre'
'pit-a-pat' 'rat-a-tat-tat' 'rub-a-dub' 'tete-a-tete' 'vis-a-vis']
found 13 matches for -?-.
-----
received request:
FIND hello
Bytes: 5
Connection-type: multi-request
-----
['hello' 'othello' 'othellos' 'phellodendron' 'seychellois']
found 5 matches for hello.
-----
received request:
FIND quit
Bytes: 4
Connection-type: multi-request
-----
waiting for connections... server connected to ('127.0.0.1', 52863)
-----
received request:
FIND a??t
Bytes: 4
Connection-type: multi-request
-----
['abate' 'abatement' 'abating' ... 'zealot' 'zealotry' 'zooplankton']
found 1546 matches for a..t.
-----

simple-server -- python3 tcp/multithread-client.py
~/git/simple-server python3 tcp/multithread-client.py %
ariblack@owls simple-server % python3 tcp/multithread-client.py
-----
wildcard query: -?-
received response:
-----
Code 11
Success: Found 13 matches for -?-.
['bric-a-brac', 'cap-a-pie', 'cock-a-doodle-doo', 'cock-a-leekie', 'high-muck-a-muck', 'love-in-a-mist', 'man-on-a-horse', 'pied-a-terre', 'pit-a-pat', 'rat-a-tat-tat', 'rub-a-dub', 'tete-a-tete', 'vis-a-vis']
-----
wildcard query: hello
received response:
-----
Code 11
Success: Found 5 matches for hello.
['hello', 'othello', 'othellos', 'phellodendron', 'seychellois']
-----
wildcard query: quit
received response:
-----
ariblack@owls simple-server %

simple-server -- python3 tcp/multithread-client.py
~/git/simple-server python3 tcp/multithread-client.py %
ariblack@owls simple-server % python3 tcp/multithread-client.py
-----
wildcard query: a??t
received response:
-----
Code 11
Success: Found 1546 matches for a??t.
['abate', 'abatement', 'abating', 'abatis', 'abstjour', 'abattis', 'abattoir', 'abet', 'abetalipoproteinemia', 'abetment', 'abetter', 'abito', 'abnormality', 'abstain', 'abstainer', 'abstemious', 'abstemiously', 'abstemiousness', 'abstention', 'absterge', 'abstergent', 'abstersion', 'abstersive', 'abstinence', 'abstinent', 'abstract', 'abstracted', 'abstractedly', 'abstractedness', 'abstraction', 'abstractionism', 'abstractionist', 'abstractive', 'abstractly', 'abstractness', 'abstractor', 'abstruse', 'abstrusely', 'abut', 'abutilon', 'abutment', 'abuttal', 'abutter', 'abutting', 'acacitry', 'acatalectic', 'acataphasia', 'acathasis', 'acathexis', 'acclamate', 'acclamation', 'acetal', 'acetabulum', 'acetal', 'acetaldehyde', 'acetamide', 'acetaminophen', 'acetanilide', 'acetate', 'acetic', 'acetone', 'acetonic', 'acetophenetidin', 'acetose', 'acetous', 'acetyl', 'acetylcholine', 'acetylene', 'acetylenic', 'acetylic', 'acquaint', 'acquaintance', 'acquainted', 'acquainted(p)', 'acquaintance', 'acquaintend', 'acquaintin', 'acromastic', 'acromastical', 'acromastics', 'actuality', 'acute', 'acutely', 'acuteness', 'adiabatic', 'adit', 'aditi', 'aditya', 'adytum', 'aerter', 'aerter', 'aesthetic', 'aesthetically', 'aesthetics', 'aestival', 'affluvi', 'agate', 'agateware', 'agathis', 'aggravate', 'aggravated', 'aggravating', 'aggravatingly', 'aggravation', 'aggravant', 'agitate', 'agitated', 'agitation', 'agitative', 'agitator', 'agitur', 'airtight', 'ajutage', 'alate', 'alated', 'aletris', 'algebraist', 'aliter', 'aliterate', 'alitur', 'alkalotic', 'alma-ata', 'alphabet', 'alphabetarian', 'alphanumeric', 'alphanumeric', 'alphabetically', 'alphabetization', 'alphabetized', 'alstonia', 'alstromeria', 'alstromeriaceae', 'altarstairs', 'alytes', 'amalgamate', 'amalgamated', 'amalgamation', 'amalgamative', 'amanita', 'amaritude', 'amaterasu', 'amateur', 'amateurish', 'amateurishly', 'amateurishness', 'amateurism', 'amative', 'amat', 'amatory', 'amatabolic', 'amethyst', 'amethystine', 'ametria', 'ametropia', 'ametropic', 'amitosis', 'amitotic', 'amitri']
```


Test server response for a query that has no matches.

```

simple-server -- python3 tcp/multithread-server.py
~/git/simple-server
ariblack@owls simple-server % python3 tcp/multithread-server.py
=====
server started successfully!
=====
waiting for connections... server connected to ('127.0.0.1', 53406)

-----
received request:
FIND 00
Bytes: 2
Connection-type: multi-request

[]
found 0 matches for 00.
-----
[]

simple-server -- python3 tcp/multithread-client.py
~/git/simple-server
ariblack@owls simple-server % python3 tcp/multithread-client.py
=====
wildcard query: 00
=====
received response:
=====
Code 0
No matches found
[]
=====
wildcard query: 

```

Known Issues: The code does not work as intended when the client or server processes terminate unexpectedly prior to disconnect. Client premature disconnect causes an error in the server process, and server disconnect does not notify the client of disconnect (and errors are not currently being handled or caught in any way by the server or client). Additionally, sending characters that are not in the ASCII library to the server may cause an error (?, -, and () are okay).

Resources

No specific sources were referenced for the development of the TCP client/server programs.