SPAASM Assignment 2 - Documentation

Ondrej Krajčovič

Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Systémové Programovanie A Assemblery

xkrajcovico@stuba.sk

19. april 2025

IV.Semester of BC study

Assignment:

Number of Assignment**:**

Napíšte v jazyku C jednoduchý interaktívny program, "shell", ktorý bude opakovane čakať na zadanie príkazu a potom ho spracuje. Na základe princípov klient-server architektúry tak musí s pomocou argumentov umožňovať funkciu servera aj klienta. Program musí umožňovať spúšťať zadané príkazy a bude tiež interpretovať aspoň 4 z nasledujúcich špeciálnych znakov: # ; < > | \ . Príkazy musí byť možné zadať zo štandardného vstupu a tiež zo spojení reprezentovaných soketmi. Na príkazovom riadku musí byť možné špecifikovať prepínačom - p port číslo portu a/alebo prepínačom -u cesta názov lokálneho soketu na ktorých bude program čakať na prichádzajúce spojenia. Po spustení s prepínačom -h sa musia vypísať informácie o autorovi, účele a použití programu, zoznam príkazov. "Shell" musí poskytovať aspoň nasledujúce interné príkazy: help - výpis informácií ako pri -h, quit - ukončenie spojenia z ktorého príkaz prišiel, halt - ukončenie celého programu. Prednastavený prompt musí pozostávať z mena používateľa, názvu stroja, aktuálneho času a zvoleného ukončovacieho znaku, e.g. '16:34 user17@student#'. Na zistenie týchto informácií použite vhodné systémové volania s použitím knižničných funkcií. Na formátovanie výstupu, zistenie mena používateľa z UID a pod. môžte v programe využiť bežné knižničné funkcie. Spúšťanie príkazov a presmerovanie súborov musia byť implementované pomocou príslušných systémových volaní. Tie nemusia byť urobené priamo (cez asembler), avšak knižničná funkcia popen(), prípadne podobná, nesmie byť použitá. Pri spustení programu bez argumentov, alebo s argumentom "-s" sa program bude správať vyššie uvedeným spôsobom, teda ako server. S prepínačom "-c" sa bude správať ako klient, teda program nadviaže spojenie so serverom cez socket, do ktorého bude posielať svoj štandardný vstup a čítať dáta pre výstup. Chybové stavy ošetrite bežným spôsobom. Počas vytvárania programu (najmä kompilácie) sa nesmú zobrazovať žiadne varovania a to ani pri zadanom prepínači prekladača -Wall. Vo voliteľných častiach zadania sa očakáva, že tie úlohy budú mať vaše vlastné riešenia, nie jednoduché volania OS.

Bonuses:

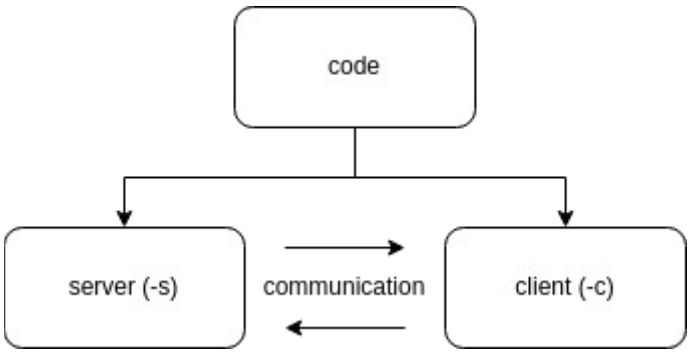
- I would like to try to fulfill these bonus requirements(3,4,5,7,23):
    - *3. (3 body) Interný príkaz stat vypíše zoznam všetkých aktuálnych spojení na ktorých prijíma príkazy, prípadne aj všetky sokety na ktorých prijíma nové spojenia.*
    - *4. (2 body) Interný príkaz abort n ukončí zadané spojenie.*
    - *5. (4 body) Interné príkazy listen a close (s príslušnými argumentami) pre otvorenie a zatvorenie soketu pre prijímanie spojení.*
    - *7. (2 body) S prepínačom "-i" bude možné zadať aj IP adresu na ktorej bude program očakávať spojenia (nielen port).*
    - *23. (1 bod) Dobré komentáre, resp. rozšírená dokumentácia, v anglickom jazyku.*

*note: I wanted to implement more (especially * as string substitute), but due to lack of time and necessary work on other projects, I was unable to

Solution:

Structure:

- Project is made of these elements:
  - spaasm.c - assembly code
  - spaasm - executable
  - files and directories for executing commands
  - documentation



Help:

Help (run by -h flag) displays all summary of all implemented flags and necessary information about tunning the code.

I Compiled the project using standard apt linux gcc, using -o compilation flag and Wall warning display flag. For me it hadn't retuned any errors or warnings

Showcase of Help funciton

```
ondrej@Ondrej:~/Downloads/fiit-temp/spaasm/fin/final$ ./spaasm -h
Remote Command Execution Tool
Usage: ./spaasm [-s (server) | -c (client)] [-p port] [-i ip]

Options:
  -s        Run in server mode
  -c        Run in client mode
  -p PORT   Specify port number (default: 55555)
  -i IP     Specify IP address (default: 127.0.0.1)
  -h        Show this help message

Examples:
  Start server: ./spaasm -s
  Connect client: ./spaasm -c -i 192.168.1.100 -p 55555
ondrej@Ondrej:~/Downloads/fiit-temp/spaasm/fin/final$  gcc -Wall -o spaasm fin.c
ondrej@Ondrej:~/Downloads/fiit-temp/spaasm/fin/final$
```

Server:

When running in server mode(using -s flag), the program acts as a command execution host. It listens for incoming client connections on specified ports (default: 55555). Each connected client gets a dedicated child process, allowing multiple users to interact with the shell simultaneously.

Note: default IP address is loopback: 127.0.0.1

he server supports internal commands for administration:

- stat – Lists all active client connections (IP, port, connection time) and open listening ports.
- listen [port] – Opens a new socket to accept connections on the given port.
- close [port] – Stops listening on a port but keeps existing connections alive.
- abort [fd] – Forcefully terminates a client connection by file descriptor.
- halt – Shuts down the entire server, closing all connections.

Commands from clients are executed similarly to a local shell, with support for pipes, redirections, comments and serialization (| < > # ;). The server sends output back to the client over the socket

Showcase of usage from server’s perscpective:

```
ondrej@Ondrej:~/Downloads/fiit-temp/spaasm/fin/final$ ./spaasm -s
Server ready. Use 'listen PORT' to start listening on a port
Available commands: stat, listen PORT, close PORT, abort FD, halt
listen 55555
Listening on port 55555
Closed port 55555
Client connected from 127.0.0.1:43814
open input: No such file or directory
stat
Active connections:
FDIP AddressPortConnected Since
4127.0.0.143814Sat Apr 19 22:38:17 2025
abort 4
Connection 4 aborted
close 55555
Closed port 55555
halt
Server shutdown initiated
ondrej@Ondrej:~/Downloads/fiit-temp/spaasm/fin/final$
```

Client:

In client(run by -c) mode, the program connects to a server (specified via -i [IP] -p [port]) and acts as a remote terminal. The user types commands locally, which are sent to the server for execution. The server's response is then displayed on the client.

- The client mirrors the server's prompt (user@hostname#) to simulate a local shell.
- Supports all shell features (pipes, redirections, comments etc.) since commands are forwarded to the server.
- quit closes the connection but leaves the server running.

The client-server communication uses length-prefixed messages (4-byte header + payload) to ensure correct parsing.

Showcase of usage from client’s perspective:

```
ondrej@Ondrej:~/Downloads/fiit-temp/spaasm/fin/final$ ./spaasm -c
Connected to 127.0.0.1:55555
22:38:17 ondrej@Ondrej# ls
spaasm
spaasm.c
test
22:38:19 ondrej@Ondrej# cd test
/home/ondrej/Downloads/fiit-temp/spaasm/fin/final/test
22:38:22 ondrej@Ondrej# echo "raz dva tri" | wc -w
3
22:38:41 ondrej@Ondrej# echo "one" ; echo "two"
"one"
"two"
22:39:00 ondrej@Ondrej# echo hej #ho - comment
hej
22:39:19 ondrej@Ondrej# ls
ahoj.txt
hello.txt
22:39:30 ondrej@Ondrej# more ahoj.txt
hi
22:39:43 ondrej@Ondrej# echo "hola" >ahoj.txt
22:39:57 ondrej@Ondrej# ls
ahoj.txt
hello.txt
22:39:59 ondrej@Ondrej# more ahoj.txt
"hola"
22:40:02 ondrej@Ondrej# help
Error: command exited with status 1
execvp: No such file or directory
22:40:24 ondrej@Ondrej# quit
ondrej@Ondrej:~/Downloads/fiit-temp/spaasm/fin/final$
```

Table of Contents

Sources:
Assignment
Personal Experience
Stack Owerflow – Implementation solutions
Wikipedia – socet & UNIX theory
Deepseek - debug