KIV/UPS - 2. Bonus: Server v jazyce C (s BSD sockety)

Autor: Milan Vlachovský (A21B0318P)

Zadání

- Bonusová úloha spočívá v implementaci serveru, který pracuje s následujícím protokolem:
    - Protokol je textový, používá transportní služby protokolu TCP, všechny zprávy jsou velkými písmeny a zakončené \n (zalomení řádku)
    - Jako první vyžaduje pozdrav HELLO
    - Na ten odešle náhodně vygenerované číslo s hlavičkou NUM:, např. NUM:42
    - Obratem očekává pouze číslo bez hlavičky, které je dvojnásobkem vygenerovaného čísla, např. 84
    - Pokud je odpověď správná, odpoví OK. Pokud ne, odpoví WRONG.
    - Na detekovanou chybu v protokolu (absence HELLO, odpověď není číslo, ...) odpovídá zprávou ERROR a pak hned ukončí spojení.
    - Server musí správně paralelizovat obsluhu klientů! Když to tedy jednomu klientovi trvá déle, nesmí blokovat ostatní!

Implementace

- Server je implementován v jazyce C s využitím BSD socketů.
- Struktura serveru zahrnuje hlavní smyčku, která poslouchá na specifikovaném portu a akceptuje nová připojení. Klienti jsou obsluhováni v paralelním režimu pomocí select() pro non-blocking IO operace.
- Detailní chování serveru:
    - Příprava Socketů:
        - Server očekává maximálně 30 klientů.
        - `prepare_sockets()`: Tato funkce inicializuje pole klientů (sockety) nastavením všech hodnot na -1, což indikuje, že žádný klient není připojen. Toto je základní krok pro správu více klientů.
    - Inicializace Master Socketu:
        - `init_master_socket()`: Vytvoří hlavní socket pomocí volání socket(). Tato funkce je základem pro přijímání nových připojení.
    - Nastavení Master Socketu:
        - `set_up_master_socket()`: Nastaví hlavní socket pro více připojení pomocí setsockopt() a připraví ho pro bindování a poslouchání. Umožňuje opakované použití portu a adresy.
    - Konfigurace Adresy:

        - `configure_address()`: Nastaví strukturu adresy sockaddr_in pro bindování socketu, včetně IP adresy a portu z konstant definovaných v const.h.

    - Přidání Aktivních Socketů:

        - `add_active_sockets()`: Přidává aktivní klienty do setu fd_set, který se používá v select() pro detekci aktivity. Funkce také vrátí nejvyšší file descriptor pro použití v select().

    - Ošetření Nového Připojení:

        - `handle_new_connection()`: Akceptuje nové připojení, přiřadí klientovi socket a inicializuje pro něj session. Při úspěchu přidává klienta do interní správy klientů a session.

    - Autentizace a Validace Session:

        - `handle_session_authentication()` a `handle_session_validation()`: Tyto funkce spravují proces autentizace (ověření příkazu HELLO) a následné validace (zpracování a ověření dvojnásobku zaslaného čísla). Pokud klient nesplní očekávání, spojení se ukončí s odpovědí WRONG.

    - Zpracování Vstupů/Výstupů Klienta:

        - `handle_client_io()`: Zpracovává všechny operace čtení a zápisu pro klienty. Spravuje stavový automat session, včetně přechodu mezi stavy CONNECTED a AUTHORIZED na základě platnosti dat.

    - Signal Handler:

        - `sigintHandler()`: Zachytává signály (např. SIGINT pro ukončení serveru) a nastavuje flag running na false, což vede k ukončení hlavní smyčky serveru.

    - Hlavní Smyčka Serveru:

        - `main()`: Obsahuje hlavní smyčku, která používá select() pro asynchronní zpracování více socketů. Spravuje časové limity, přijímání nových připojení a zpracování dat od existujících klientů. Zajišťuje, že server zůstává reaktivní i při vysokém počtu klientů.

Struktura

- `server.c`
    - Hlavní implementace serveru.
- `session.c, session.h`
    - Definice a implementace session managementu pro klienty.
    - Session představuje v jaké fázi komunikace se serverem se klient nachází.
- `session_set.c, session_set.h`
    - Správa množiny session, včetně přidání a odebrání session.
- `err_messages.c, err_messages.h`
    - Soubory obsahující definice funkcí pro tisk chybových hlášení spojených s network operacemi.
- `const.h`

    - Definice konstant použitých v serveru.
    - Zde se nastavuje IP adresa a port serveru.
- `Makefile`
    - Pro kompilaci serveru a jeho čistění.
- `client.py`
    - Python skript pro testování serveru pomocí simulace klientů.
- `docs/`
    - Adresář obsahující dokumentaci k serveru.
- `doc.html`
    - Zástupce k dokumentaci serveru.
- `soubory s prefixem log_`
    - Logy z testování serveru a klienta pomocí client.py.
- `server_ares`
    - Zkompilovaný server spustěný na <ares.fav.zcu.cz> pro validaci.

Návod na spuštění a použití

Sestavení serveru:

```
$ make
```

Tento příkaz kompiluje server a vytváří spustitelný soubor server.

Spuštění serveru:

```
$ ./server
```

Po spuštění server poslouchá na portu 8080 a je připraven přijímat klienty.

Testování serveru:

- Může se použít přiložený client.py skript pro simulaci klientů a testování serveru.
    - Tento skript vyžaduje Python a předpokládá, že server běží na localhostu na portu 8080.

      ```
      $ python client.py
      ```

Skript simuluje více klientů, kteří se pokusí autentizovat a validovat s serverem. Někteří klienti schválně provádějí chyby v protokolu, aby bylo možné otestovat chování serveru vůči chybám.

Server lze ukončit stisknutím `Ctrl+C`.

Výstup

- Po spuštění serveru a klienta je možné sledovat výstup na konzoli.

- Zde je zagregorován výstup z testování serveru pomocí client.py:

```
==28193== Memcheck, a memory error detector
==28193== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et
al.
==28193== Using Valgrind-3.18.1 and LibVEX; rerun with -h for
copyright info
==28193== Command: ./server_debug
==28193==
Configured IP: 127.0.0.1, Port: 8080
Binded master socket to port 8080
Listening on port 8080
Waiting for connections ...
Timeout occurred.
Activity detected.
New connection:
socket fd is 4, ip(127.0.0.1), port(55414)
Adding to list of sockets as 0
> Client 0: Connected to server at 127.0.0.1:8080 from port 55414
---
===
Activity detected.
New connection:
socket fd is 5, ip(127.0.0.1), port(55418)
Adding to list of sockets as 1
> Client 1: Connected to server at 127.0.0.1:8080 from port 55418
---
Host interaction: ip(127.0.0.1), port(55414)
Authentication successful.
> Client 0: Sent to server: HELLO
---
===
Activity detected.
New connection:
socket fd is 6, ip(127.0.0.1), port(55426)
Adding to list of sockets as 2
> Client 2: Connected to server at 127.0.0.1:8080 from port 55426
---
Host interaction: ip(127.0.0.1), port(55414)
Validation successful.
Closing sockets
> Client 0: Received from server: NUM:4757
> Client 0: Extracted number: 4757
> Client 0: Sent to server: 9514
> Client 0: Received from server: OK
> Client 0: Connection closed by the server
---
Host interaction: ip(127.0.0.1), port(55418)
Authentication failed.
> Closing sockets
> Client 1: Sent to server: BYE
> Client 1: Received from server: WRONG
> Client 1: Connection closed by the server
```

```
---
===
Activity detected.
New connection:
socket fd is 4, ip(127.0.0.1), port(55438)
Adding to list of sockets as 0
> Client 3: Connected to server at 127.0.0.1:8080 from port 55438
---
Host interaction: ip(127.0.0.1), port(55426)
Authentication successful.
> Client 2: Sent to server: HELLO
---
===
Activity detected.
Host interaction: ip(127.0.0.1), port(55438)
Authentication successful.
> Client 3: Sent to server: HELLO
---
===
Activity detected.
Host interaction: ip(127.0.0.1), port(55438)
Validation failed.
Closing sockets
> Client 3: Received from server: NUM:5817
> Client 3: Extracted number: 5817
> Client 3: Sent to server: some_error
> Client 3: Received from server: WRONG
> Client 3: Connection closed by the server
---
Host interaction: ip(127.0.0.1), port(55426)
Validation failed.
Closing sockets
> Client 2: Received from server: NUM:843
> Client 2: Extracted number: 843
> Client 2: Sent to server: 1687
> Client 2: Received from server: WRONG
> Client 2: Connection closed by the server
---
===
Activity detected.
New connection:
socket fd is 4, ip(127.0.0.1), port(55450)
Adding to list of sockets as 0
> Client 5: Connected to server at 127.0.0.1:8080 from port 55450
---
===
Activity detected.
New connection:
socket fd is 5, ip(127.0.0.1), port(55446)
Adding to list of sockets as 1
> Client 4: Connected to server at 127.0.0.1:8080 from port 55446
---
Host interaction: ip(127.0.0.1), port(55450)
Authentication successful.
```

```
> Client 5: Sent to server: HELLO
---
===
Activity detected.
Host interaction: ip(127.0.0.1), port(55446)
Host disconnected
Closing sockets
> Client 4: Client disconnected before authentication
---
===
Activity detected.
Host interaction: ip(127.0.0.1), port(55450)
Validation successful.
Closing sockets
> Client 5: Received from server: NUM:9733
> Client 5: Extracted number: 9733
> Client 5: Sent to server: 19466
> Client 5: Client disconnected before validation
---
===
Caught signal 2

Error occurred while running select():
    A signal was caught.
==28193==
==28193== HEAP SUMMARY:
==28193==     in use at exit: 0 bytes in 0 blocks
==28193==   total heap usage: 8 allocs, 8 frees, 4,528 bytes allocated
==28193==
==28193== All heap blocks were freed -- no leaks are possible
==28193==
==28193== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from
0)
```

- Zprávy klienta mají prefix "> ".

Testování

- Server byl testován na lokálním počítači (Linux Mint) s využitím Python skriptu client.py a nástroje Valgrind pro detekci memory leaks s tím, že k žádným memory leaks nedošlo.
- Server běžel na adrese localhost:8080.
- Dále prošel testováním z https://kiv-ubl.kiv.zcu.cz/ups/
  - Login: ***mvlach***