Kraków, 1 II 2020

AF_LOCAL stress test

(zadania zaliczeniowe, #3)

Należy stworzyć dwa programy, działające w modelu klient-serwer w nietypowej, odwróconej konfiguracji. Serwer (`massivereader`) oczekuje na połączenia TCP, za pomocą których klient przekazuje adresy gniazd w domenie `AF_LOCAL`. Ponadto, serwer pełni rolę klienta, łącząc się na podane adresy i odczytując z nich komunikaty.
Natomiast `multiwriter` jest wersją klienta, ukierunkowaną na przetestowanie wydolności komunikacji w domenie `AF_UNIX`. Program tworzy wiele gniazd, z którymi łączy się serwer, i w zadanym tempie wysyła przez nie komunikaty.

Uwaga. Należy tak skonstruować programy, aby zminimalizować opóźnienia obsługi (maksymalne i sumaryczne) i unikać gubienia komunikatów.

1. Serwer

Program pełni równolegle dwie role: serwera używającego protokołu TCP, oraz klienta odbierającego komunikaty przekazywane w domenie `AF_LOCAL`. Jako serwer, otrzymuje od klientów adresy z domeny lokalnej, z którymi ma się połączyć (staje się klientem). W roli klienta, odbiera strumień komunikatów, które zapisuje do utworzonego przez siebie pliku.

---

program: `massivereader`
parametry:

`<int>`	numer portu, na którym nasłuchuje na połączenia TCP
`O <łańcuch>`	prefiks ścieżki do pliku

specyfikacja:

§1 Rola serwera
Program oczekuje pod adresem `TCP/127.0.0.1`, na porcie wskazanym przez parametr pozycyjny. Po nawiązaniu połączenia pobiera strumień danych, interpretowanych jako ciąg struktur typu `sockaddr_un`.
Dla każdej z odebranych struktur serwer próbuje połączyć się w trybie strumieniowym z zawartym w niej adresem, a klientowi odsyła informację o wyniku. W przypadku sukcesu odsyła tę samą strukturę, a w przypadku niepowodzenia odsyła zmodyfikowaną, w której pole `sun_family` ma ustawioną wartość `-1`.
Deskryptor nawiązanego połączenia, wraz z adresem jest zapamiętywany i wykorzystywany przy realizacji roli klienta.

§2 Rola klienta
Z gniazda domeny `AF_LOCAL` program odbiera komunikaty w następującym formacie:

- łańcuch znaków zakończony `'\0'`,
- łańcuch bajtów długości 108 znaków,
- łańcuch bajtów długości `sizeof(struct timespec)`.

Pierwsze pole jest właściwym komunikatem. Drugie służy do weryfikacji nadawcy, a trzecie zawiera znacznik czasu wysłania i służy do szacowania opóźnień w obsłudze.
Weryfikacja komunikatu polega na porównaniu zawartości drugiego pola z zapamiętanym adresem, powiązanym z używanym gniazdem. Jeżeli nie ma zgodności, to pakiet zostaje odrzucony.

Kroki podejmowane po odczytaniu komunikatu:

1. pobranie znacznika czasu globalnego (*wall clock*) i wyznaczenie opóźnienia obsługi (względem czasu wysłania),
2. weryfikacja nadawcy,
3. utworzenie reprezentacji tekstowych znacznika czasu i opóźnienia, zgodnie z 3,
4. zapisanie do właściwego pliku wyników w formacie:
   - reprezentacja tekstowa znacznika,
   - właściwy komunikat,
   - reprezentacja tekstowa czasu opóźnienia.

   separatorem między polami wyniku ma być sekwencja znaków `' : '`.

§3 Pliki do logowania transmisji
W czasie działania `massivereader` może utworzyć i zapisać wiele plików. Pierwszy taki plik powstaje na początku działania programu, a kolejne w reakcji na sygnał `SIGUSR1`.
Nazwy plików są generowane na podstawie prefiksu zadanego parametrem, do którego doklejane są kolejne liczby z zakresu od 0 do 999. Dodawany tekst ma być długości trzech znaków (cyfr) i być uzupełniany zerami (`"%03d"`) Jeżeli utworzona ścieżka wskazuje na miejsce, do którego nie ma praw odczytu, to generowana jest następna. Jeżeli wskazuje na istniejący plik, to zostanie on nadpisany.
Po utworzeniu nowego pliku, stary ma zostać zamknięty.

2. Klient

Program `multiwriter` realizuje zadania klienta (`AF_INET`) i serwera (`AF_LOCAL`) wobec `massivereader`. Jako serwer obsługuje wiele połączeń, przez które, w określonym tempie, wysyła komunikaty. Dzięki tym właściwościom może być używany do testowania jakość implementacji serwera i ogólnie, wydajność komunikacji w domenie `AF_LOCAL`.

---

program: `multiwriter`
parametry:

`-S <int>`	ilość połączeń `AF_LOCAL`
`-p <int>`	port serwera rejestrującego adresy `AF_LOCAL`
`-d <float>`	minimalne odstępy pomiędzy komunikatami (w μsekundach)
`-T <float>`	całkowity czas pracy (w centysekundach)

specyfikacja:

§1 Gniazda w domenie `AF_LOCAL`

Program tworzy jedno gniazdo w trybie strumieniowym, w domenie `AF_LOCAL`. Jego adres ma być losowym łańcuchem bajtów, spełniającym warunki *Linux Abstract Domain Namespace*. Pozostałe gniazda w tej domenie (w liczbie określonej parametrem `-S`) mają powstawać w wyniku nawiązywania połączeń.
Po utworzeniu zadanej liczby gniazd, podstawowe gniazdo ma zostać usunięte, a adres zwolniony. (Dotyczy to także sytuacji, gdy druga strona poinformuje o niemożności wykonania połączenia.)

§2 Rejestracja adresu `AF_LOCAL`

Program łączy się z serwerem znajdującym się pod adresem `127.0.0.1`, na porcie podanym w parametrze `-p`. Uzyskanym kanałem wysyła strukturę użytą do rejestracji gniazda w domenie `AF_LOCAL`. Wysyła ją tyle razy, ile było zażądanych połączeń (`-S`). Jednocześnie odbiera odpowiedzi informujące, czy drugiej stronie udało się nawiązać połączenia.
Gdy program otrzyma już wszystkie odpowiedzi rozłącza się, zwalnia użyte gniazdo (`AF_INET`) i przechodzi do etapu wysyłania komunikatów.
Uwaga. Ponieważ druga strona może w dowolnym momencie zamknąć jakieś połączenie, `multiwriter` *musi monitorować ich stan i aktualizować pulę nadających się do użycia.*

§3 Wysyłanie komunikatów (`AF_LOCAL`)

Komunikaty są wysyłane cyklicznie, w odstępach czasu niekrótszych niż określony parametrem `-d`. Gdy nastąpi właściwy moment, wykonywane są następujące kroki.

1. Pobierany jest znacznik czasu globalnego (*wall clock*) i tworzona jest jego reprezentacja tekstowa, zgodnie z opisem w sekcji 3.
2. Losowane jest jedno z działających gniazd.
3. Za jego pośrednictwem wysyłany jest komunikat składający się z:
   · tekstowej reprezentacji znacznika czasu, zakończonej `'\0'`,
   · adresu gniazda, który był wysłany przy rejestracji,
   · liczbowej wartości znacznika (struktura `timespec`).
4. Ponownie pobierany jest znacznik i na jego podstawie określane są:
   — czas operacji przygotowania i wysłania komunikatu,
   — ile zostało lub brakło czasu do kolejnego cyklu.
   Czas obsługi jest sumowany dla wszystkich komunikatów. Natomiast spośród wyników uzyskanych dla drugiego parametru, dla wszystkich cykli, zapamiętywane są wartości ekstremalne.

§4 Czas działania

Przed przystąpieniem do etapu wysyłania komunikatów, `multiwriter` rozpoczyna odliczanie czasu przeznaczonego na pracę. Pomiar dotyczy rzeczywistego czasu posiadania dostępu do procesora. Po przekroczeniu okresu zadanego parametrem `-T`, program przerywa działanie. Na zakończenie wypisuje na standardowym wyjście komunikat zawierający informacje o sumarycznym czasie poświęconym na wysyłanie komunikatów oraz o minimalnym i maksymalny odstępie do nadchodzącego cyklu.

3. Reprezentacja tekstowa czasu

Okres czasu ma być reprezentowany w formacie: `mmm*:ss,cc.dd.ee.fff`, gdzie poszczególne pola oznaczają:

- `mmm*` - liczbę minut, zapisaną na przynajmniej dwóch pozycjach
- `ss` - liczba sekund (od 00 do 59),
- `ccddeefff` - ilość nanosekund.

Procedura generująca reprezentację ma być bezpieczna dla wątków i obsługi sygnałów (np., nie można użyć nic z rodziny `printf`).