Systemy operacyjne

Lista zadań nr 9

Na zajęcia 16 i 17 grudnia 2025

Należy przygotować się do zajęć czytając następujące materiały: [6, 1.5], [1, 11.4], [7, rodziały 2, 4, 5 i 8].

Zadanie 1. Na podstawie [6, 1.5] wyjaśnij zadania pełnione przez protokoły warstwy: **łącza**, **sieciowej** i **transportowej**. Zainstaluj i uruchom program wireshark[1]. Przechwyć kilka **pakietów** protokołów `UDP` i `TCP`, a następnie wytłumacz do czego służy **kapsułkowanie** (ang. *encapsulation*) i wyświetl (tj. kliknij dwukrotnie na pakiet) nagłówki **ramki**, **datagramu** i **segmentu**. Zidentyfikuj adres źródłowy i docelowy pakietu. Czemu protokoły warstwy łącza i sieciowej nie są używane do komunikacji między procesami użytkownika?

Komentarz: Program `tcpdump(8)` jest narzędziem pełniącym podobną funkcję co `wireshark`, ale działającym w trybie tekstowym.

Zadanie 2. Na podstawie [7, 2.3 i 2.4] omów różnice między protokołami warstwy transportowej: **datagramowym** `udp(7)` i **połączeniowym** `tcp(7)`. Czym różni się komunikacja półdupleksowa od dupleksowej? Jak `TCP` radzi sobie z zagubieniem **segmentu** lub faktem, że segmenty mogą przyjść do odbiorcy w innej kolejności niż zostały wysłane? Skąd protokół `TCP` wie kiedy połączenie zostało zerwane? Jaki problem rozwiązuje **sterowanie przepływem** (ang. *flow control*) implementowane przez `TCP`?

Wskazówka: Przy tłumaczeniu właściwości protokołów posłuż się analogią (np. wysyłanie listu, dzwonienie przez telefon, itp.)

Zadanie 3. Omów diagram [7, 4.1] komunikacji **klient-serwer** używającej protokołu `tcp(7)` przy pomocy interfejsu **gniazd strumieniowych**. W którym momencie następuje związanie gniazda z adresem lokalnym i zdalnym? Która ze stron komunikacji używa **portów ulotnych** (ang. *ephemeral*)? Co specyfikuje drugi argument wywołania systemowego `listen(2)`? Z jakim numerem portu jest związane gniazdo przekazywane do i zwracane z `accept(2)`? Skąd serwer wie, że klient zakończył połączenie?

Zadanie 4. Omów diagram [7, 8.1] komunikacji klient-serwer używającej protokołu `udp(7)` przy pomocy interfejsu **gniazd datagramowych**. Czemu, w przeciwieństwie do `TCP`, serwer może rozpocząć pracę zaraz po wykonaniu funkcji `bind(2)`? Z jakiej przyczyny interfejs `read(2)` i `write(2)` po stronie serwera może być niewystarczający? Przedstaw semantykę operacji `recvfrom(2)` i `sendto(2)`. Kiedy po stronie klienta następuje związanie gniazda `UDP` z adresem lokalnym? Na podstawie [7, 8.11] zreferuj efekt jaki przynosi wykonanie `connect(2)` na gnieździe klienta. Jakie ograniczenia poprzednio wymienionych funkcji zostały poprawione przez wywołania `recvmsg(2)` i `sendmsg(2)`?

Zadanie 5. Przyjrzyjmy się warunkom brzegowym, które występują w trakcie używania interfejsu gniazd `BSD`. Kiedy `read(2)` i `write(2)` na gniazdach strumieniowych zwracają *short counts*? Skąd wiemy, że odebrany datagram `UDP` nie został obcięty przez jądro w trakcie kopiowania do przestrzeni użytkownika? Z jakich przyczyn należy być przygotowanym na to, że operacje na gniazdach zwrócą «`EINTR`»? Co się stanie, jeśli klient spróbuje zapisać do gniazda powiązanego z połączeniem, które serwer zdążył już zamknąć? Dlaczego w kodzie funkcji «`open_listenfd`» użyto wywołania `setsockopt(2)` z opcją «`SO_REUSEADDR`» [3, 61.10]? Co by się stało gdyby programista o tym zapomniał?

[1] `https://www.wireshark.org`

Ściągnij ze strony przedmiotu archiwum «so25_lista_9.tar.gz», następnie rozpakuj i zapoznaj się z dostarczonymi plikami.

UWAGA! Można modyfikować tylko te fragmenty programów, które zostały oznaczone w komentarzu napisem «TODO».

Zadanie 6. Zmodyfikuj program «hostinfo.c» w taki sposób, aby wyświetlał adresy IPv4 oraz IPv6 dla danej nazwy serwera. Dodatkowo należy przekształcić nazwę usługi przekazanej jako opcjonalny trzeci parametr programu na numer portu. Poniżej przykład:

```
# hostinfo www.google.com https
216.58.215.68:443
[2a00:1450:401b:803::2004]:443
```

Co należałoby zrobić, żeby program rozpoznawał usługę o nazwie «tftp»?

Zadanie 7. Zapoznaj się z kodem źródłowym serwera «echoserver.c» i klienta «echoclient.c» usługi podobnej do «echo». Twoim zadaniem jest taka modyfikacja serwera, by po odebraniu sygnału «SIGINT» wydrukował liczbę bajtów odebranych od wszystkich klientów, po czym zakończył swe działanie.

Używając programu «watch» uruchom polecenie «netstat -ptn», aby obserwować stan połączeń sieciowych. Wystartuj po jednym procesie serwera i klienta używając wybranego portu (np. 7777). Wskaż na wydruku końce połączenia należące do serwera i klienta. Następnie wystartuj drugą instancję klienta. Czemu nie zachowuje się ona tak samo jak pierwsza? Co zmieniło się na wydruku z «netstat»?

Uruchom program wireshark. Na interfejsie sieciowym loopback każ mu nasłuchiwać pakietów TCP przychodzących na port, który wybrano do komunikacji między klientem i serwerem. Wskaż pakiety przesyłane w trakcie otwierania i zamykania połączenia oraz w trakcie przesyłania danych między klientem i serwerem.

Zadanie 8. Serwer z poprzedniego zadania nie radził sobie ze współbieżną obsługą wielu połączeń. Serwer z pliku «echoclient-fork.c» naprawia to poważne ograniczenie z użyciem wywołania «fork». Zadaniem głównego procesu jest odbieranie połączeń i delegowanie ich obsługi do podprocesów.

Proces serwera musi zliczać liczbę bajtów odebranych od klientów. W tym celu przydziela dzieloną pamięć anonimową, w której przechowuje tablicę «client». Przy starcie podprocesu umieszcza w tablicy odpowiedni wpis za pomocą procedury «addclient». Żeby uniknąć wyścigów każdy podproces zwiększa własny licznik «nread». Po zakończeniu podprocesu należy wywołać procedurę «delclient», która doda zawartość prywatnego licznika klienta, do globalnego licznika serwera.

W dowolnym momencie działanie serwera może zostać przerwane przy pomocy sygnału «SIGINT». Należy wtedy poczekać na zakończenie podprocesów i wydrukować zawartość globalnego licznika serwera. Poniżej przykładowy wydruk z sesji serwera:

```
# ./echoserver-fork 8000
[9047] Connected to localhost:36846
[9105] Connected to localhost:36850
[9047] Disconnected!
^C
Server received quit request!
[9105] Disconnected!
Server received 22 bytes
#
```

Zadanie 9 (bonus). Zapoznaj się z procedurami «echo» w pliku «echoclient.c», «echoserver.c» i «echoserver-fork.c». Usuń komentarz otaczający wywołanie funkcji exit(3) odpowiednio w kliencie i serwerze. Zaobserwuj występujące usterki, a następnie wytłumacz ich źródło. Jak naprawić te programy, by nie ulegały awarii?

Literatura

[1] „*Computer Systems: A Programmer's Perspective*"
Randal E. Bryant, David R. O'Hallaron
Pearson Education Limited; 3rd edition; 2016

[2] „*Advanced Programming in the UNIX Environment*"
W. Richard Stevens, Stephen A. Rago
Addison-Wesley Professional; 3rd edition; 2013

[3] „*The Linux Programming Interface: A Linux and UNIX System Programming Handbook*"
Michael Kerrisk
No Starch Press; 1st edition; 2010

[4] „*Systemy operacyjne*"
Andrew S. Tanenbaum, Herbert Bos
Helion; wydanie czwarte; 2015

[5] „*Operating Systems: Three Easy Pieces*"
Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau
https://pages.cs.wisc.edu/~remzi/OSTEP/

[6] „*Computer Networking: A Top-Down Approach*"
James F. Kurose, Keith Ross
Pearson; 8th Edition; 2021

[7] „*Unix Network Programming, Volume 1: The Sockets Networking API*"
W Richard Stevens, Bill Fenner, Andrew M. Rudoff
Addison-Wesley Professional; 3rd edition; 2003

[8] „*Dynamic Storage Allocation: A Survey and Critical Review*"
Paul R. Wilson, Mark S. Johnstone, Michael Neely, David Boles
https://www.cs.hmc.edu/~oneill/gc-library/Wilson-Alloc-Survey-1995.pdf