Systemy operacyjne

Lista zadań nr 9

Na zajęcia 10 grudnia 2020

Należy przygotować się do zajęć czytając następujące rozdziały książek lub publikacji:

- Computer Systems: A Programmer's Perspective (wydanie trzecie): 11.4
- Unix Network Programming: The Sockets Networking API (wydanie trzecie): 2, 4, 5 i 8

Zadanie 1. Na podstawie §2.3 i §2.4 omów różnice między: protokołem `ip(7)`, **datagramowym** protokołem `udp(7)` i **połączeniowym** protokołem `tcp(7)`. Czym różni się komunikacja półdupleksowa od dupleksowej? Jak TCP radzi sobie z zagubieniem **segmentu** lub zamianą kolejności wysłanych segmentów po stronie odbierającego? 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 2. Omów diagram 4.1 komunikacji **klient-serwer** używającej protokołu `tcp(7)` przy pomocy interfejsu **gniazd strumieniowych**. W którym momencie następuje związanie gniazda z adresem będącym parą $(\text{numer IP}, \text{port})$? Która ze stron komunikacji używa **portów efemerycznych** (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 3. Omów diagram 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 jest niewystarczający? Przedstaw semantykę operacji `recvfrom(2)` i `sendto(2)`. Na podstawie §8.11 zreferuj efekt jaki przynosi wykonanie `connect(2)` na gnieździe klienta.

Zadanie 4. 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 nie został obcięty? 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»? Co by się stało gdyby programista o tym zapomniał?

Wskazówka: Zapoznaj się z §5.8 – §5.15 i §7.4.

Ściągnij ze strony przedmiotu archiwum «so20_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 5. 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 6. 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. 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»?

Zadanie 7 (2). 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
#
```