1. Co to jest sieciowa kolejność bajtów? Jakie mogą być skutki nie stosowania jej?

   Do informacji przesyłanych w sieci (np. w nagłówkach protokołów TCP/IP) przyjęto standardową kolejność bajtów: najpierw starszy bajt (*big endian*). Kolejność tę nazwano *sieciową kolejnością bajtów*.

2. Wyjaśnić do czego służy funkcja **gethostbyname ()** i **gethostbyaddr** () ..

   Funkcja `gethostbyname` pobiera ciąg znaków ASCII reprezentujący nazwę domenową komputera (parametr *`name`*) i zwraca wskaźnik do wypełnionej struktury `hostent` zawierającej między innymi 32-bitowy adres komputera.

   Funkcja `gethostbyaddr` pobiera adres IP (parametr *`addr`*) i zwraca wskaźnik do wypełnionej struktury `hostent` zawierającej między innymi nazwę domenową hosta.

3. Wyjaśnić w jaki sposób działa funkcja connect () w przypadku komunikacji bezpołączeniowej.

   Działanie funkcji connect() zależy od typu gniazda. W przypadku gniazda TCP jej zadaniem jest nawiązanie połączenia z serwerem. W przypadku gniazda UDP funkcja connect() przekazuje do jądra dane serwera. Mówimy wtedy o gnieździe UPD połączonym, co oznacza, że nie musimy określać adresu docelowego IP ani numeru portu.

4 Jakie sygnały i dlaczego powinny być obsługiwane przez serwer?

   Serwer powinien obsługiwać takie sygnały jak:
   SIGPIPE – sygnał jest generowany jeśli piszemy dane i proces czytający zakończy pracę lub gdy proces pisze dane do gniazda a drugi koniec połączenia sieciowego zakończy pracę – obsługuje przerwania w połączeniu
   SIGCHLD – sygnał jest generowany za każdym razem, gdy proces kończy się lub zatrzymuje. Domyślnie sygnał ten jest ignorowany, a więc proces macierzysty musi go przechwycić, jeśli chce być powiadomiony o każdej zmianie stanu potomka. Jeśli sygnał nie będzie przechwytywany to będą pozostawiały procesy zombie.

5. Wyjaśnić w jakiej sytuacji brak obsługi sygnału SIGPIPE może doprowadzić do przerwania pracy serwera. Czy wszystkiego rodzaju serwery są podatne na ten problem?

6. Wyjaśnić na czym polega różnica między funkcją **close** () i funkcja **shutdown**(). Podać przykłady zastosowania.

   - Funkcja close zamyka połączenie (w obu kierunkach) i usuwa gniazdo.
     - Funkcja `close` oznacza gniazdo o deskryptorze *`socket`* jako zamknięte, zmniejsza licznik odniesień do gniazda o 1 i natychmiast wraca do procesu. Proces nie może się już posługiwać tym gniazdem, ale warstwa TCP spróbuje wysłać dane z bufora wysyłkowego, po czym zainicjuje wymianę segmentów kończących połączenie TCP. Jednakże, jeśli po zmniejszeniu liczby odniesień do gniazda nadal jest ona > 0, nie jest inicjowana sekwencja zamykania połączenia TCP (wysłanie segmentu FIN).
   - Funkcja oznacza gniazdo *`socket`* jako zamknięte w kierunku określonym drugim parametrem. Inicjuje sekwencję zamykania połączenia TCP bez względu na liczbę odniesień do deskryptora gniazda.

7. Co to są dane poza pasmowe i w jaki sposób są przesyłane?

- Dane pozapasmowe (ang. *out-of-band date*) to dane, które są przesyłane z wyższym priorytetem. Każdy rodzaj warstwy transportowej obsługuje te dane w inny sposób.
- Do obsługi danych pozapasmowych w protokole TCP wykorzystywany jest tryb pilny (ang *urgent mode*). Można w ten sposób przesłać jeden znak jako dane pilne.

- Przesyłanie danych pozapasmowych:
  send(socket,"?",1,MSG_OOB);

8. Podaj co najmniej dwa przykłady konieczności użycia opcji gniazd.

`IPPROTO_TCP` - oprogramowanie TCP
`IPPROTO_IPV6` - oprogramowanie IPv6

9. Protokół TCP jest protokołem *strumieniowym.* Protokół UDP jest protokołem *datagramowym.* Jakie to ma znaczenie z punktu widzenia protokołu aplikacji?

10. Uzupełnić poniższy kod klienta tak, aby zapevr.ić sprawdzanie czy druga strona nie zamknęła połączenia. Sprawdzanie jest realizowane podczas czytania.

```
int sock, ret;
char buffer[MAX_BUFFER_SIZE+1];
sock=socket(AF_INET,SOCK_STREM,0);
/* Klient łączy się z serwerem ... */
ret=read(sock, buffer, MAX_BUFFER_SIZE);
/* uzupełnij ... */
```

```
if(ret<0)
      {
      Printf(„Druga strona zamknęła połączenie”);
      }
```

Trzeba sprawdzić czy funkcja read nie zwróciła wartości -1 co oznaczałoby że wystąpił bład

11. Uzupełnić poniższy kod serwera tak, aby zapewnić sprawdzanie czy druga strona nie zamknęła połączenia. Sprawdzanie jest realizowane podczas zapisu.

```
int servsock, ret, clisock, size;
char buffer[MAX_BUFFER_SIZE+1];
servsock=socket(AF_INET,SOCK_STREM,0);
/* ustawienie serwera */
clisock=accept(servsock, (struct sockaddr *)NULL, NULL);
/* odebranie zapytania i przygotowanie danych do wysłania */
ret=write(sock, buffer, size);
/* uzupełnij ... */
```

```
if(ret<0 || ret<size)
      {
      Printf(„Druga strona zamknęła połączenie”);
      }
```

Trzeba sprawdzić czy funkcja write nie zwróciła wartości -1 co oznaczałoby że wystąpił błąd i czy ilość wysłanych danych nie jest mniejsza od wielkości buffora którą wysyłamy

12. W serwerze używamy funkcji bind(), aby jawnie przekazać adres gniazdu. Mówimy, że gniazdo musi mieć adres, aby można było się z nim komunikować. Jednakże w programie klienta nie stosujemy funkcji bind(). W jaki sposób gniazdo klienta otrzymuje adres lokalny?

    Adres lokalnego gniazda klienta połączeniowego zostaje przydzielony przez jądro systemu przy próbie nawiązania połączenia z serwerem.

13. Co będzie jeśli w serwerze pominiemy wywołanie funkcji bind?

    serwerowi nie zostanie przypisany lokalny adres protokołowy tzn że serwer nie będzie posiadał informacji na którym porcie ma słychać czy nie będzie wiedział nić o wybranej rodzinie protokołów

14. Co to jest tryb rozgłoszeniowy?. Podać przykład (pseudo)kodu (serwera i klienta), który realizuje taki sposób komunikacji.

    Broadcast – rozsiewczy (rozgłoszeniowy) tryb transmisji danych polegający na wysyłaniu przez jeden port (kanał informacyjny) pakietów, które powinny być odebrane przez wszystkie pozostałe porty przyłączone do danej sieci (domeny broadcastowej).

15. Serwer współbieżny realizuje dwa jednoczesne żądania klientów. Czas realizacji każdego z żądań wynosi 2 sek. Ten sam serwer obsługuje pięć jednoczesnych żądań przez 5 sek.
    a) Co możemy powiedzieć o funkcji, która wykorzystuje serwer do realizacji żądań?
       Funkcja jest funkcją z któej korzystają wszystkie procesy. Funkcja powoduje zablokowanie pozaostalych procesow az do zakonczenia działania procesu któy ja wywołał
    b) Co można by powiedzieć o tej funkcji, gdyby czas realizacji każdego z żądań trwał 1 sek. obojętnie od liczby jednoczesnych żądań?
       Funkcja jest metoda nieblokującą pozostałe procesy
16. Napisać przykład prostej sesji komunikacji za pomocą protokołu HTTP.

    GET / HTTP/1.1
    GET / HTTP/1.1 200 OK

17. Napisać pseudokod serwera współbieżnego wieloprocesowego połączeniowego. Zaznaczyć miejsca, które mogą spowodować zablokowanie procesu głównego. Wyjaśnić przyczynę.

    gniazdo = socket()
    bind(gniazdo, adres, rozmiar)
    listen(gniazdo,5)
    gniazdo2 = accept(gniazdo) // tu jest zatrzymanie bo czekamy na polaczenie klienta
    if ((wartosc = fork()) ==0)
    zamknij gniazdo
    prztrwarzaj kilenta
    if(wartosc > 0)
    zamknij gniazdo2
    else
    error
    zamknij gniazdo2

18. Przeanalizować różnice zastosowania funkcji **close** () w przypadku serwera współbieżnego

wieloprocesowego i wielowątkowego.

w serwerze wieloprocesowym deskryptory otwartych gniazd sa kopiowane dlatego zamkniecie gniazda w dziecku nie powoduje zamkniecia gniazda w rodzicu natomiast w wielowatkowym jest inaczej, deskryptory sa wspoldzielone dlatego zamkniecie gniazda dziecka spowoduje zmakniecie gniazda rodzica

19. Co to jest serwer wielousługowy? Ile maksymalnie gniazd potrzebuje serwer połączeniowy współbieżny wielousługowy, jeśli liczba usług wynosi N?

N

20. Co to znaczy, że operacje gniazdowe są wykonywane w trybie blokującym? Na czym polega tryb nieblokujący? Podaj przykład wykorzystania trybu nieblokującego.

Każde gniazdo TCP ma bufor wysyłkowy. Do niego kopiowane są dane z bufora użytkowego aplikacji. Jeśli gniazdo jest gniazdem blokującym (ustawienie domyślne), powrót z funkcji write będzie oznaczał, że wszystkie dane z bufora aplikacji zostały umieszczone w tym buforze. Dane są usuwane z tego bufora dopiero po otrzymaniu potwierdzenia ACK. Operacje w trybie blokującym blokuj ąproces aż do zakonczenia wykonywania tej operacji np read czeka az otrzyma dane. Operacje w trybie nieblokującym nie blokują procesów. Read w trybie nieblokującym próbuje odczytać dane jeśli nie ma ich zwraca błąd do zmiennej erno a wykonywana jest dalsza cześć kodu(powrot z funkcji jest natychmiastowy) Program musi odpytywaćtaki deskryptor czy operacja jest juz gotowa(pulling)

WYKORZYSTANIE : robienie timeoutow