

Systemy operacyjne

Sprawozdanie - laboratorium 10 „Programowanie sieciowe 1”

Andrzej Kołakowski
296586

1) Gniazda

1. Odpowiedzi na pytania

- Co identyfikuje adres IP a co port?

Adres IP to liczba służąca do identyfikacji urządzenia w sieci. Natomiast port służy do identyfikacji konkretnego procesu lub usługi sieciowej.

- Czym różni się deskryptor gniazda od deskryptora pliku?

Zbiorem funkcji które na nich operują. Część funkcji pokrywa się: `read(2)`, `write(2)`, `close(2)`, a inne nie – na przykład: `socket(2)` zamiast `open(2)`.

- Który argument funkcji [`socket`] służy do określenia typu gniazda?

Funkcja `socket` zadeklarowana jest jako: `socket(int domain, int type, int protocol)`, zatem jest to drugi argument.

- Jakie wartości przyjmuje ten argument dla gniazd połączeniowych a jakie dla bezpołączeniowych?

Dla gniazd połączeniowych przyjmuje wartość **SOCK_STREAM**, a dla bezpołączeniowych - **SOCK_DGRAM**.

- Jaki jest zakres liczbowy portów dostępnych do wykorzystania dla użytkownika niebędącego administratorem?

1024 - 65535

- Co oznaczają pojęcia: Big-Endian, Little-Endian, Network Byte Order?

Big endian to forma zapisu danych, w której najbardziej znaczący bajt umieszczony jest jako pierwszy. Alternatywna nazwa to **network byte order**.

Little endian to forma zapisu danych, w której najmniej znaczący bajt umieszczony jest jako pierwszy.

- Do czego służą funkcje: `htonl`, `htons`, `ntohl`, `ntohs`? Co oznaczają ich nazwy (od czego są to skróty)?

Funkcje te służą do konwersji wartości z kolejności bajtów hosta na tzw. network byte order i odwrotnie.

The `htonl()` function converts the unsigned integer `hostlong` from host byte order to network byte order.

The `htons()` function converts the unsigned short integer `hostshort` from host byte order to network byte order.

The `ntohl()` function converts the unsigned integer `netlong` from network byte order to host byte order.

The `ntohs()` function converts the unsigned short integer `netshort` from network byte order to host byte order.

- Liczbę w postaci szesnastkowej: `cafe` zapisano na dwóch bajtach w postaci: `feca`. Jaka to reprezentacja?

Little endian.

- Które ze struktur dotyczą protokołu IPv4 a które IPv6?

Struktury `sockaddr_in` i `in_addr` dotyczą IPv4, `sockaddr_in6` i `in6_addr` IPv6.

- W których strukturach i do którego pola zapisujemy adres w postaci binarnej dla adresu IP w wersji 4 i 6?

W polu `sin_addr` struktury `sockaddr_in` dla IPv4, oraz w polu `sin6_addr` struktury `sockaddr_in6` dla IPv6.

3) Prosty serwer

1. Przetestuj komunikację uruchamiając przykłady na dwóch różnych komputerach (np. uruchom serwer na swoim koncie na `student.agh.edu.pl` a klienta na komputerze laboratoryjnym lub odwrotnie). Czy pojawiają się problemy? Jeżeli tak, to jakie i dlaczego?

Testy przeprowadziłem na serwerach `borg.kis.agh.edu.pl` i `charon.kis.agh.edu.pl`.

```
kolaandr@borg: ~/Pulpit/SO
kolaandr@borg:~/Pulpit/SO$ ./server
server: got connection from 149.156.207.22
kolaandr@borg:~/Pulpit/SO$

kolaandr@borg: ~/Pulpit/SO
kolaandr@borg:~/Pulpit/SO$ telnet borg.kis.agh.edu.pl 3490
Trying 149.156.207.22...
Connected to borg.kis.agh.edu.pl.
Escape character is '^]'.
Hello, world!
Connection closed by foreign host.
kolaandr@borg:~/Pulpit/SO$
```

Komunikacja przez telnet w obrębie jednego serwera

```
kolaandr@borg: ~/Pulpit/SO
kolaandr@borg:~/Pulpit/SO$ ./server
server: got connection from 149.156.207.22
kolaandr@borg:~/Pulpit/SO$

kolaandr@borg: ~/Pulpit/SO
kolaandr@borg:~/Pulpit/SO$ ./client borg.kis.agh.edu.pl
Received: Hello, world!
kolaandr@borg:~/Pulpit/SO$
```

Komunikacja z wykorzystaniem programu client w obrębie jednego serwera

W obrębie jednego serwera możliwa jest komunikacja zarówno przez telnet, jak i wykorzystując program `client`.

```
kolaandr@charon: ~/Pulpit/so
kolaandr@charon:~/Pulpit/so$ ./server

kolaandr@borg: ~/Pulpit/SO
kolaandr@borg:~/Pulpit/SO$ ./client charon.kis.agh.edu.pl
^C
kolaandr@borg:~/Pulpit/SO$ telnet charon.kis.agh.edu.pl 3490
Trying 149.156.207.21...
^C
kolaandr@borg:~/Pulpit/SO$
```

Nie udało się uzyskać komunikacji między dwoma serwerami, zarówno wykorzystując telnet jak i program `client`. Myślę, że problemem może być wprowadzona przez administratora konfiguracja serwera uniemożliwiająca połączenie (blokada portów).

4) Rozbudowa serwera

1. Eternal vigilance

Zmodyfikowano serwer tak, aby po obsłużeniu klienta nie kończył działania, ale powracał do oczekiwania na kolejne połączenie.

root@localhost:~/Desktop/so/lab10	root@localhost:~/Desktop/so/lab10
<pre>File Edit View Search Terminal Help [root@localhost lab10]# ./server server: got connection from 127.0.0.1 [root@localhost lab10]#</pre>	<pre>File Edit View Search Terminal Help [root@localhost lab10]# ./client localhost Received: Hello, world! [root@localhost lab10]# ./client localhost connect: Connection refused [root@localhost lab10]#</pre>

Przed zmianami

root@localhost:~/Desktop/so/lab10	root@localhost:~/Desktop/so/lab10
<pre>File Edit View Search Terminal Help [root@localhost lab10]# ./server_4_1 server: got connection from 127.0.0.1 server: got connection from 127.0.0.1 server: got connection from 127.0.0.1 server: got connection from 127.0.0.1 [root@localhost lab10]#</pre>	<pre>File Edit View Search Terminal Help [root@localhost lab10]# ./client localhost Received: Hello, world! [root@localhost lab10]# ./client localhost Received: Hello, world! [root@localhost lab10]# ./client localhost Received: Hello, world! [root@localhost lab10]# ./client localhost Received: Hello, world! [root@localhost lab10]#</pre>

Po zmianach

2. Obsługa wielu klientów

Zmodyfikowano serwer tak, aby mógł obsługiwać jednocześnie więcej niż jednego klienta. Wykorzystano `fork()` do tworzenia procesów potomnych, tak aby każdy proces potomny obsługiwał jednego klienta.

```
root@localhost:~/Desktop/so/lab10 x root@localhost:~/Desktop/so/lab10 x
File Edit View Search Terminal Help File Edit View Search Terminal Help
[root@localhost lab10]# ./server_4_1 [root@localhost lab10]# ./client localhost
server: got connection from 127.0.0.1 Received: Hello, world!
[ ] [root@localhost lab10]# [ ]
```

```
root@localhost:~/Desktop/so/lab10 x
File Edit View Search Terminal Help
[root@localhost lab10]# ./client localhost
[ ]
```

Wcześniejsza wersja: mimo, że kolejny klient może się połączyć to musi czekać aż serwer zakończy obsługę poprzedniego (w celu symulacji obsługi dodano sleep(5))

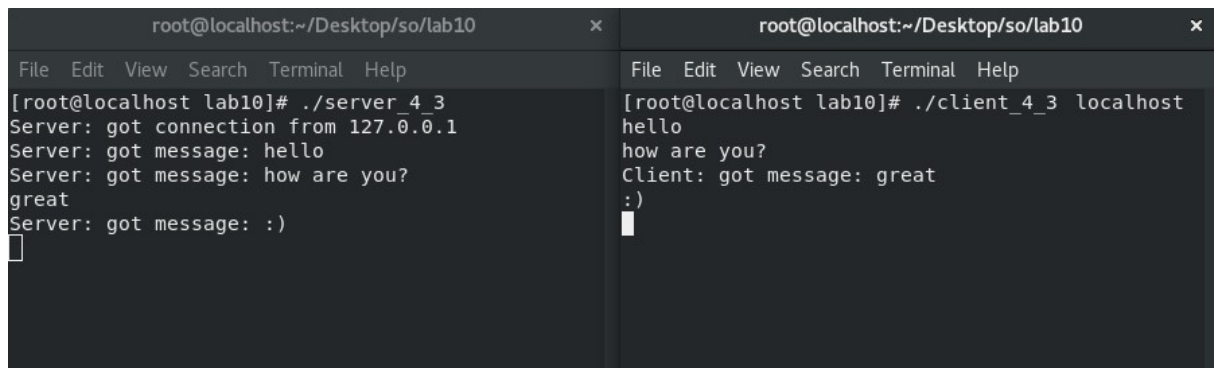
```
root@localhost:~/Desktop/so/lab10 x root@localhost:~/Desktop/so/lab10 x
File Edit View Search Terminal Help File Edit View Search Terminal Help
[root@localhost lab10]# ./server_4_2 [root@localhost lab10]# ./client localhost
server: got connection from 127.0.0.1 Received: Hello, world!
server: got connection from 127.0.0.1 [root@localhost lab10]# [ ]
[ ]
```

```
root@localhost:~/Desktop/so/lab10 x
File Edit View Search Terminal Help
[root@localhost lab10]# ./client localhost
Received: Hello, world!
[root@localhost lab10]#
```

Nowa wersja: obsługa wielu klientów jednocześnie

3. Komunikator

Zmodyfikowano serwer tak, aby umożliwił prowadzenie dialogu (między klientem a serwerem) jak popularne komunikatory internetowe. Po akceptacji połączenia program tworzy dwa procesy potomne, jeden do czytania portu, drugi do pisania.

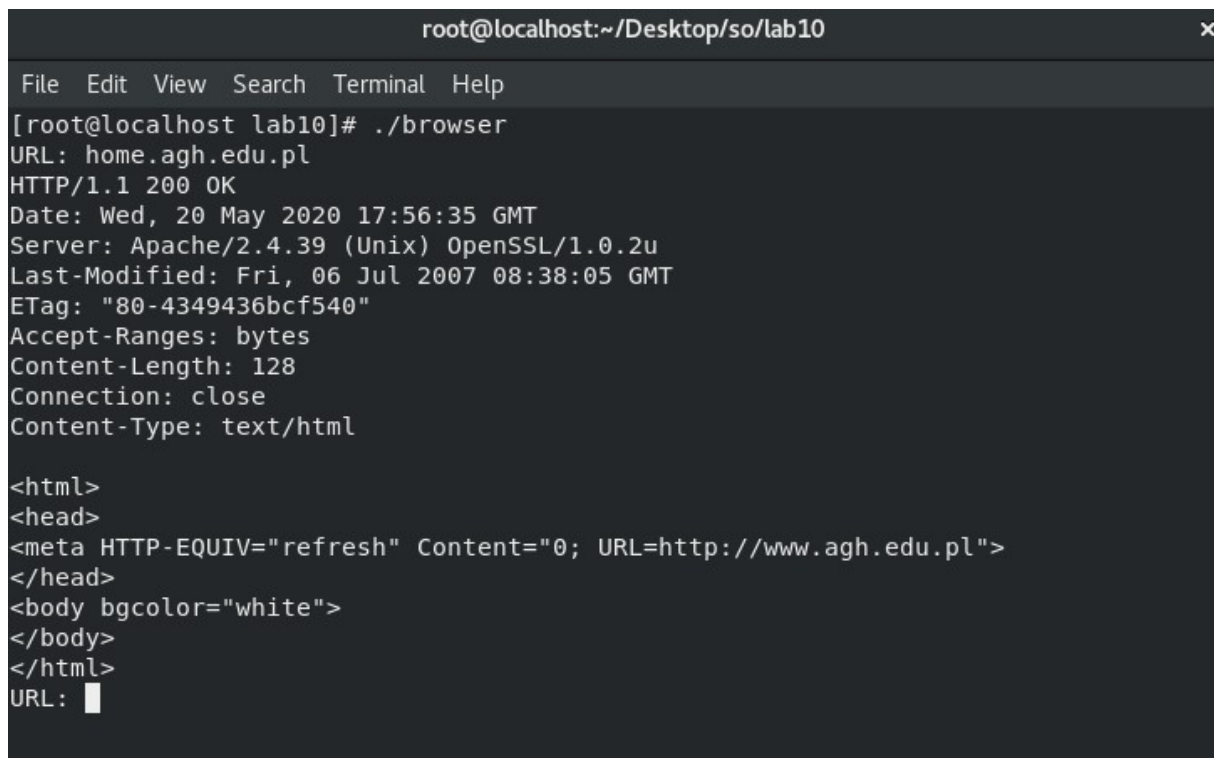


```
root@localhost:~/Desktop/so/lab10 x root@localhost:~/Desktop/so/lab10 x
File Edit View Search Terminal Help File Edit View Search Terminal Help
[root@localhost lab10]# ./server_4_3 [root@localhost lab10]# ./client_4_3 localhost
Server: got connection from 127.0.0.1 hello
Server: got message: hello how are you?
Server: got message: how are you? Client: got message: great
great :)
Server: got message: :)
█
```

Warto zwrócić uwagę, że wiadomości nie muszą być wymieniane naprzemiennie. Dzięki wykorzystaniu dwóch procesów jedna strona może wysłać zero, jedną lub kilka wiadomości przed odbiorem kolejnych.

5) Protokół HTTP

1. Przeglądarka – napisanie programu według schematu



```
root@localhost:~/Desktop/so/lab10 x
File Edit View Search Terminal Help
[root@localhost lab10]# ./browser
URL: home.agh.edu.pl
HTTP/1.1 200 OK
Date: Wed, 20 May 2020 17:56:35 GMT
Server: Apache/2.4.39 (Unix) OpenSSL/1.0.2u
Last-Modified: Fri, 06 Jul 2007 08:38:05 GMT
ETag: "80-4349436bcf540"
Accept-Ranges: bytes
Content-Length: 128
Connection: close
Content-Type: text/html

<html>
<head>
<meta HTTP-EQUIV="refresh" Content="0; URL=http://www.agh.edu.pl">
</head>
<body bgcolor="white">
</body>
</html>
URL: █
```

6) Pytania kontrolne

- Czym są i do czego służą gniazda?

Gniazda to wewnętrzne punkty końcowe służące wysyłaniu lub odbieraniu danych.

- Czym się różni komunikacja połączeniowa od bezpołączeniowej?

W trybie połączeniowym procesy muszą ustanowić logiczne połączenie, aby rozpocząć komunikację. W trybie bezpołączeniowym nie nawiązują połączenia, tylko wysyłają do siebie niezależne komunikaty zwane datagramami.

- Co to jest Network Byte Order, Host Byte Order - czym się różnią, na czym polegają?

Host Byte Order to kolejność zapisu bajtów na danej maszynie (np. **Little Endian** lub **Big Endian** – różnice między nimi omówiłem w sekcji 1).

Natomiast **Network Byte Order** zawsze oznacza **Big Endian**.

- Co to są adresy IPv4 i IPv6, jaką mają postać?

Adres IP to liczba nadawana interfejsowi sieciowemu służąca identyfikacji elementów sieci w warstwie trzeciej modelu OSI.

Adres IPv4 jest 32-bitową liczbą całkowitą, natomiast IPv6 128-bitową liczbą całkowitą.

- Do czego służą porty?

Porty służą do identyfikacji konkretnego procesu lub usługi sieciowej.

- Jakie są typowe numery portów dla http, ftp, ssh itp.?

http - 80, ftp - 21, ssh - 22, telnet - 23.

- Jak wygląda komunikacja z użyciem protokołu HTTP?

Komunikacja HTTP realizowana jest poprzez wysłanie żądania (request) do serwera, który następnie generuje odpowiedź (response).