Gniazda BSD
implementacja w C#

Gniazda

Implementacja w C#:

```
Przestrzeń nazw:
    System.Net.Sockets
Klasa:
    public class Socket : IDisposable
```

Gniazda

Implementacja w C#:

Konstruktor:

```
public Socket(
    AddressFamily addressFamily,
    SocketType socketType,
    ProtocolType protocolType
)
```

Oryginał:

```
#include <sys/types.h>
#include <sys/socket.h>
int socket(int domain, int type, int protocol);
```

Gniazda

Implementacja w C#:

Parametry konstruktora:

```
public enum AddressFamily
```

przydatne wartości:

- InterNetwork - gniazdo dla IP4
- InterNetworkV6 - gniazdo dla IP6

Gniazda

Implementacja w C#:

Parametry konstruktora:

```
public enum SocketType
```

przydatne wartości:

- Stream – gniazdo strumieniowe
- Dgram – gniazdo datagramowe
- Raw – gniazdo surowe
- Seqpacket – gniazdo z numerowanymi pakietami

Gniazda

Implementacja w C#:

Parametry konstruktora:

```
public enum ProtocolType
```

przydatne wartości:

- Tcp
- Udp
- IPv4
- IPv6
- Raw

Gniazda

Przykład – tworzymy gniazdo dla klienta protokołu HTTP

```
using System.Net.Sockets;

Socket s = new Socket
   (AddressFamily.InterNetwork,
    SocketType.Stream,
    ProtocolType.Tcp);
```

Gniazda

Implementacja w C#:

```
Metoda:

public void Connect(
    string host,
    int port
)
```

Oryginał:

```
#include <sys/types.h>
#include <sys/socket.h>

int connect(int sockfd,struct sockaddr *serv_addr,socklen_t addrlen);
```

Gniazda

Wyjątki rzucane przez Connect():

ArgumentNullException – pusty adres hosta

ArgumentOutOfRangeException – niepoprawny port

SocketException – błąd wewnętrzny klasy Socket

ObjectDisposedException – gniazdo już zamknięto

NotSupportedException – niepoprawna rodzina

InvalidOperationException – gniazdo nasłuchuje

Gniazda

Przykład – łączymy się z serwerem HTTP

```
s.Connect(”www.wi.zut.edu.pl”, 80);
```

Gniazda

Implementacja w C#:

```
Metoda:

public int Send(
    byte[] buffer
)
```

Oryginał:

```
#include <sys/types.h>
#include <sys/socket.h>

ssize_t send(int s, const void *buf, size_t len, int flags);
```

Gniazda

Wyjątki rzucane przez Send():

```
ArgumentNullException – pusty bufor
SocketException – błąd wewnętrzny klasy Socket
ObjectDisposedException – gniazdo już zamknięto
```

Gniazda

Przykład – wysyłamy polecenie do serwera HTTP

```
s.Send(
   System.Text.Encoding.ASCII.GetBytes(
      ”GET / HTTP\r\n\r\n”
   )
);
```

Gniazda

Implementacja w C#:

Metoda:

```
public int Receive(
    byte[] buffer
)
```

Oryginał:

```
#include <sys/types.h>
#include <sys/socket.h>

ssize_t recv(int s, void *buf, size_t len, int flags);
```

Gniazda

Wynik funkcji Receive():

```
Wynik:

Liczba faktycznie odczytanych bajtów danych
lub (-1) w wypadku wystąpienia błędu
```

Gniazda

Wyjątki rzucane przez Receive():

`ArgumentNullException` – pusty bufor

`SocketException` – błąd wewnętrzny klasy Socket

`ObjectDisposedException` – gniazdo już zamknięto

`SecurityException` – niedostateczne uprawnienia

Gniazda

Przykład – odbieramy odpowiedź serwera HTTP

```
byte[] buff = new byte[1024];

while(s.Receive(buff) > 0)
   Console.Write(
      System.Text.Encoding.ASCII.GetString(
         buff));
```

Gniazda

Implementacja w C#:

```
własność:
public int Available { get; }
```

Liczba bajtów, jaką można bezpiecznie odczytać z gniazda

Gniazda

Implementacja w C#:

```
Metoda:
public void Close()
```

Oryginał:

```
#include <unistd.h>
int close(int fd);
```

Gniazda

Przykład – zamykamy połączenie z serwerem

```
s.close();
```

Gniazda

Kompletny klient HTTP:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net.Sockets;

namespace ConsoleApplication2
{
    class Program
    {
        static void Main(string[] args)
        {
            Socket s = new Socket(AddressFamily.InterNetwork,
                                  SocketType.Stream,ProtocolType.Tcp);
            s.Connect("www.wi.zut.edu.pl", 80);
            s.Send(System.Text.Encoding.ASCII.GetBytes(
                    "GET / HTTP\r\n\r\n"));
            byte[] buff = new byte[1024];
            while(s.Receive(buff) > 0)
                Console.Write(System.Text.Encoding.ASCII.GetString(buff));
        }
    }
}
```

Gniazda

A teraz napiszemy serwer protokołu HTTP – plan pracy:

- *Socket*
- *SetSockOpt*
- *Bind*
- *Listen*
- *Accept*

Gniazda

Implementacja w C#:

Metoda:

```
public void SetSocketOption(
    SocketOptionLevel optionLevel,
    SocketOptionName optionName,
    bool optionValue
)
```

Oryginał:

```
#include <sys/types.h>
#include <sys/socket.h>

int setsockopt(int s,int level,int optname,
               void *optval,socklen_t optlen);
```

Gniazda

Implementacja w C#:

Parametr metody:

```
public enum SocketOptionLevel;
```

Przydatne wartości:

- Socket
- IP
- IPv6
- Tcp
- Udp

Gniazda

Implementacja w C#:

Parametr metody:

```
public enum SocketOptionName;
```

Przydatne wartości:

- ReuseAddress → *konieczne dla serwera!*
- DontRoute
- OutOfBandInline

Gniazda

Przykład – włączamy opcję ReuseAddress na poziomie Socket

```
s.SetSocketOption(
      SocketOptionLevel.Socket,
      SocketOptionName.ReuseAddress,
      true);
```

Gniazda

Implementacja w C#:

```
Metoda:

public void Bind(
    EndPoint localEP
)
```

Oryginał:

```
#include <sys/types.h>
#include <sys/socket.h>

int bind(int sockfd, struct sockaddr *my_addr, socklen_t addrlen);
```

Gniazda

Implementacja w C#:

Parametr metody:

```
using System.Net;

public abstract class EndPoint →
public class IPEndPoint : EndPoint
```

konstruktor klasy IPEndPoint:

```
public IPEndPoint(IPAddress address, int port);
```

Gniazda

Implementacja w C#:

W przypadku pisania kodu serwera szczególnie przydatny jest składnik klasy IPAddress:

```
using System.Net;

public static readonly IPAddress Any;
```

Any reprezentuje wszystkie adresy IP znane systemowi serwera.

Gniazda

Wyjątki rzucane przez Bind():

`ArgumentNullException` – pusty adres EP

`SocketException` – błąd wewnętrzny klasy Socket

`ObjectDisposedException` – gniazdo już zamknięto

`SecurityException` – niedostateczne uprawnienia

Gniazda

Przykład – dowiązujemy się do portu 80

```
s.Bind(new IPEndPoint(IPAddress.Any, 80));
```

Gniazda

Implementacja w C#:

Metoda:

```
public void Listen(
    int backlog
)
```

Oryginał:

```
#include <sys/types.h>
#include <sys/socket.h>

int listen(int sockfd, int backlog);
```

Gniazda

Wyjątki rzucane przez Listen():

`SocketException` – błąd wewnętrzny klasy Socket

`ObjectDisposedException` – gniazdo już zamknięto

Gniazda

Przykład – zaczynamy nasłuchiwać

```
s.Listen(5);
```

Gniazda

Implementacja w C#:

Metoda:

```
public Socket Accept()
```

Oryginał:

```
#include <sys/types.h>
#include <sys/socket.h>

int accept(int s, struct sockaddr *addr, socklen_t *addrlen);
```

Gniazda

Wyjątki rzucane przez Accept():

`SocketException` – błąd wewnętrzny klasy Socket

`ObjectDisposedException` – gniazdo już zamknięto

`InvalidOperationException` – gniazdo nie nasłuchuje

Gniazda

Przykład – czekamy na klienta

```
for(Socket n = s.Accept(); ; n.Close()) {
:
:
}
```

Gniazda

Kompletny „serwer” HTTP:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Net.Sockets;

namespace ConsoleApplication3
{
    class Program
    {
        static void Main(string[] args)
        {
            Socket s = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
                                  ProtocolType.Tcp);
            s.SetSocketOption(SocketOptionLevel.Socket, SocketOptionName.ReuseAddress,
                              true);
            s.Bind(new IPEndPoint(IPAddress.Any, 80));
            s.Listen(5);
            Socket n;
            string doc = "<html><body><h1>HELLO WORLD!</h1></body></html>\n\n";
            for (; (n = s.Accept()) != null; n.Close())
            {
                byte[] query = new byte[100];
                while (n.Available > 0)
                    n.Receive(query);
                n.Send(System.Text.Encoding.ASCII.GetBytes(
                        "HTTP/1.1 200 OK\r\n" +
                        "Content-Length: " + Convert.ToString(doc.Length) + "\r\n" +
                        "Content-Type: text/html\r\n\r\n" +
                        doc
                ));
            }
        }
    }
}
```