

Studentska platforma za testiranje distribuiranih aplikacija

LAN (Local Area Network) namijenjena je povezivanju računala i drugih mrežnih uređaja na manjim udaljenostima, npr. u okviru jednog ureda, zgrade, postrojenja ili kuće. Komunikacija se odvija preko TCP/IP protokola. LAN omogućava dijeljenje podataka, uređaja kao i programa.

U računarskom praktikumu računala su povezana u LAN mrežu.

Za povezivanje i pokretanje procesa na više računala, bilo je potrebno sljedeće:

Odrediti računalo i njegov hostname koje će glumiti server, te port na kojem će osluškivati klijentska računala, koja se žele spojiti na Socket.

Sve što je za to trebalo promijeniti u Java kodu s predavanja jest u

Symbols.java na svakom računalu sa kojim radimo, pod nameServer stavimo hostname računala koje predstavlja server, te serverPort postaviti na neki cijeli broj veći od 1024 jer su svi manji zauzeti za sistemske zadatke. U klasi NameServer uz male promjene ispisa postojećih procesa koji su zapisatni u NameTable,

```
public static void main(String[] args) {

    NameServer ns = new NameServer();

    System.out.println("NameServer started");

    System.out.println("-----");

    try {

        ServerSocket listener = new ServerSocket(Symbols.ServerPort);

        while (true) {

            Socket aClient = listener.accept();

            ns.handleclient(aClient);

            System.out.println(" Name table elements:");

            for (int i = 0; i < 10; i++) {

                String name = ns.table.getName(i);

                String hostName = ns.table.getHostName(i);

                int port = ns.table.getPort(i);

                f(name != null)

                System.out.println("  name: " + name + ", hostName: " + ", port: " + port );

            }

            aClient.close();
```

```

    }

    } catch (IOException e) {

        System.err.println("Server aborted:" + e);

    }

}

}

```

možemo pokrenuti NameServer.java. Za lijepši ispis LinkerTestera , u klasi Name ispisujemo svaki puta što smo dodali u zabicu svih procesa:

```

public class Name {

    BufferedReader din;

    PrintStream pout;

    public void getSocket() throws IOException {

        Socket server = new Socket(Symbols.nameServer, Symbols.ServerPort);

        din = new BufferedReader(new InputStreamReader(server.getInputStream()));

        pout = new PrintStream(server.getOutputStream());

    }

    public int insertName(String name, String hname, int portnum)

        throws IOException {

        getSocket();

        System.out.println("Insert name: " + name + ", host name: " + hname + ", port number: " + portnum);

        pout.println("insert " + name + " " + hname + " " + portnum);

        pout.flush();

        return Integer.parseInt(din.readLine());

    }

    public PortAddr searchName(String name) throws IOException {

        getSocket();

        pout.println("search " + name);

```

```

pout.flush();

String result = din.readLine();

StringTokenizer st = new StringTokenizer(result);

int portnum = Integer.parseInt(st.nextToken());

String hname = st.nextToken();

return new PortAddr(hname, portnum);

}

```

Pregledniji ispis poruka u LinkerTester.java dobijemo:

```

public class LinkerTester {

public static void main(String[] args) {

    Linker comm = null;

    Msg m;

    try {

        String baseName = args[0];

        int myId = Integer.parseInt(args[1]);

        int numProc = Integer.parseInt(args[2]);

        System.out.println("LinkerTester started:");

        System.out.println(" base name: " + baseName + ", id: " + myId + ", number of
processes: " + numProc);

        System.out.println("-----");

        comm = new Linker(baseName, myId, numProc);

        for (int i = 0; i < numProc; i++)

            if (i != myId){

                comm.sendMsg(i, "my_tag", "Uspješno ste se povezali procese na vise racunala");

                //comm.sendMsg(i, "my_tag", "poruka_od_" + myId);

                System.out.println(" > Sent message: ");

                System.out.println(" >> To: " + i + ", from: " + myId);

                System.out.println(" >> Tag: " + "my_tag");

```

```

processe na vise racunala");
        System.out.println(">> Content: " + "Uspješno ste se povezali");
    }

    for (int i = 0; i < numProc; i++)

        if (i != myId)

            m = comm.receiveMsg(i);

    }

    catch (Exception e) {

        System.err.println(e);

    }

}
}
}

```

SSH (engl. Secure Shell) je mrežni protokol koji korisnicima omogućuje uspostavu sigurnog komunikacijskog kanala između dva računala putem nesigurne računalne mreže. SSH obično koristi port 22 tcp/udp, što je na Unix računalima definirano u datoteci `/etc/services`. SSH se obično koristi za prijavu s jednog računala na drugo i izvršavanje naredbi na drugom računalu.

Kako bismo pokrenuli na svakom od klijentskih računala istovremeno procese sa onog koji predstavlja server, potreban nam je ssh. OpenSSH Server omogućuje udaljeni pristup sustavu koristeći protokol SSH. Iz komandne ljuške on se može instalirati putem sljedeće naredbe (koja instalira pakete `openssh-server` i `openssh-client`): **\$ sudo apt-get install ssh**

Da bi se uspostavila veza na SSH-server, potrebno je koristiti neki SSH-klijent. Na Linuxu je to OpenSSH Client koji omogućuje rad u tekstualnom sučelju (komandnoj ljuški). Primjer spajanja na računalo 'host' za korisnika 'ivo' sa ssh:

\$ ssh [ivo@host](#)

Želimo li vidjeti IP adresu našeg lokalnog računala, odnosno sva mrežna sučelja preko kojih se može komunicirati sa trenutnim računalom na Linuxu koristimo naredbu: **\$ ifconfig**. Rezultat je sljedeći:

```
File Edit View Bookmarks Settings Help
silva@silva-Aspire-E1-522:~ > ifconfig
eth0      Link encap:Ethernet  HWaddr 30:65:ec:24:26:5e
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:28

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1847 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1847 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:256200 (256.2 KB)  TX bytes:256200 (256.2 KB)

wlan0     Link encap:Ethernet  HWaddr a4:db:30:b3:09:73
          inet addr:192.168.1.66  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::a6db:30ff:feb3:973/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:29991 errors:0 dropped:0 overruns:0 frame:0
          TX packets:21378 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:31675901 (31.6 MB)  TX bytes:3448476 (3.4 MB)
```

U računarskom praktikumu računala su povezana međusobno preko eth0 (Ethernet), gdje bi IP pojedinog računala pročitali pod inet 192.168..... Da su računala povezana preko Wi-Fi (WirelessLAN) čitali bi pod wlan0 inet 192.168....

SKRIPTE

Standardan primjer skriptog jezika za specifično okruženje: **Bash** za Unix operativne sustave. Naredba za otkrivanje gdje se bash interpreter nalazi :

\$ which bash

postavljanje skripte da bude izvršna datoteka

\$ chmod +x hello_world.sh

Izvršavanje bash skripte

./hello_world.sh ili **source hello_world.sh**

Za naš program imamo 3 jednostavne skripte.

Prva skripta:

```
#!/bin/bash

PATH=/opt/jdk1.8.0_65/bin:$PATH

export PATH

javac *.java

java NameServer
```

Prva linija svake bash skripte započinje znakom `#!/` "she-bang" ili "hash-bang". Nakon toga slijedi `/bin/bash` put kojim pokazujemo Unix sistemu kojim interpreterom (programom) želimo pokrenuti datoteku. Zapravo će se pri svakom pokretanju skripte, pozivati `/bin/bash naziv_skripte.sh`.

Sljedeće dvije linije omogućuju kompiliranje i pokretanje java datoteka bez navođenja puta do direktorija gdje je java na računalu instalirana. Riječ 'export' omogućuje da pri svakom pokretanju ove skripte, u varijablu `PATH` dodan je put do jave, nastavljajući se na stari `PATH`. Kada to ne bi dodali u `PATH`, kompilirali i pokretali bi java programe sa: `/opt/jdk1.8.0_65/bin javac ...` odnosno `/opt/jdk1.8.0_65/bin java ...` (`/opt/jdk1.8.0_65/bin` je put do jave u računarskom praktikumu).

`javac *.java` kopilira sve java datoteke u trenutnom direktoriju, gdje se skripta nalazi.

Posljednja linija skripte pokreće `NameServer` program (datoteku).

Rezultat pokretanja skripte: Server čeka na procese sa klijentskih računala da zatraže komunikaciju.

Druga skripta:

```
#!/bin/bash

PATH=/opt/jdk1.8.0_65/bin:$PATH

export PATH

mate-terminal -e 'bash bashC.sh 0 2 user@host1'

mate-terminal -e 'bash bashC.sh 1 2 user@host2'
```

Druga skripta za razliku od prve ima svrhu otvaranja novih terminal-a (komandnih ljuski) u kojima izvršavamo treću skriptu `bashC.sh` proslijeđujući joj dva parametra.

Naredba *mate-terminal* otvara novi terminal (komandnu ljusku). Oznaka '-e' znači execute onoga što slijedi u stringu koji slijedi iza oznake. U stringu je naredba za pokretanje skripte bashC.sh sa parametrima : broj procesa trenutnog klijentskog računala i ukupan broj procesa koje želimo pokrenuti. Treći parametar koji navodimo je naziv klijentskog računala, tj. njegov hostname. Otvorimo onoliko terminala (pokrenemo skripti) koliko želimo povezanih procesa.

Treća skripta:

```
#!/bin/bash

PATH=/opt/jdk1.8.0_65/bin:$PATH

export PATH

i=$1

N=$2

h=$3

ssh $h << EOF

    cd /.home/student1/user/public_html/dprojekt

    java LinkerTester base $i $N

EOF

$SHELL
```

U trećoj skripti 4., 5. i 6. linija označavaju redni broj argumenata *i*, *N* i *h*. Argument *i* je broj procesa koji se pokreće na klijentskom računalu čiji je hostname treći argument *h*, drugi argument *N* je ukupan broj procesa koje pokrećemo i povezujemo. Prvo se naredbom *ssh \$h* povezujemo na računalo čiji je hostname *h*. Povezali smo se (ssh login) koristeći šifru (password) usera na čiji račun ulazimo. Mogli smo ukloniti unošenje passworda koristeći ključ (private-public key) koji smo generirali. Riječ *EOF* (tag) označava početak i kraj višeretčanog stringa. Može biti bilo koja druga riječ, samo da je ista na početku i kraju stringa. Uobičajeno se koriste velika tiskana slova. Na taj način izvršit će se naredbe ulaska u direktorij (*cd /..*) gdje se nalazi datoteka *LinkerTester.java* i pokretanje *LinkerTester.java* datoteke (*java ...*). Varijabla *\$SHELL* na kraju skripte omogućuje da nakon izvršavanja skripte terminal se ne zatvori. *\$SHELL* sadrži put (path) do interpretera koji ovdje koristimo.

Pokretanjem sve tri skripte dobivamo sljedeći rezultat:

1. Pokrenuli smo NameServer.java

```
Terminal
File Edit View Search Terminal Help
gaspar13@pr5-05 ~/.home/student1/gaspar13/public_html/dprojekt $ Terminal
The program 'Terminal' is currently not installed. To run 'Terminal' please ask
your administrator to install the package 'terminal.app'
gaspar13@pr5-05 ~/.home/student1/gaspar13/public_html/dprojekt $ MATE Terminal
MATE: command not found
gaspar13@pr5-05 ~/.home/student1/gaspar13/public_html/dprojekt $ mate-terminal
gaspar13@pr5-05 ~/.home/student1/gaspar13/public_html/dprojekt $ source bash02.sh
LinkerTester started:
  base name: base, id: 1, number of processes: 2
-----
Neighbors: [0]
java.net.ConnectException: Connection refused
gaspar13@pr5-05 ~/.home/student1/gaspar13/public_html/dprojekt $ source bashS.sh
NameServer started
-----
Name table elements:
  name: base1, hostName: , port: 2012
Name table elements:
  name: base1, hostName: , port: 2012
  name: base0, hostName: , port: 2011
Name table elements:
  name: base1, hostName: , port: 2012
  name: base0, hostName: , port: 2011
```

NameServer čeka na klijente da preko socketa zatraže povezivanje islanje poruka.

NameServer će svaki proces sa njegovim podacima (ime, hostname, port) spremiti u tablicu svih procesa, Name table. Koja je pokretanjem servera ispisana.

2. Pokrenuli smo proces na klijentu pod rednim brojem 0 i 1:

```
Terminal
File Edit View Search Terminal Help
Pseudo-terminal will not be allocated because stdin is not a terminal.
gaspar13@pr5-04's password:
Welcome to Linux Mint 17 Qiana (GNU/Linux 3.13.0-24-generic x86_64)

Welcome to Linux Mint
* Documentation: http://www.linuxmint.com
LinkerTester started:
  base name: base, id: 1, number of processes: 2
-----
Neighbors: [0]
Insert name: base1, host name: pr5-04, port number: 2012
> Sent message:
>> To: 0, from: 1
>> Tag: my_tag
>> Content: Uspješno ste se povezali procese na vise racunala
< Received message:
<< From: 0, to: 1
<< Tag: my_tag
<< Content: Uspješno ste se povezali procese na vise racunala
$ ~/.home/student1/gaspar13/public_html/dprojekt $
```


Poruke su razmijenjene (Sent i Recived message):

```
File Edit View Search Terminal Help
Terminal
File Edit View Search Terminal Help
Pseudo-terminal will not be allocated because stdin is not a terminal.
gaspar13@pr5-06's password:
Welcome to Linux Mint 17 Qiana (GNU/Linux 3.13.0-24-generic x86_64)

Welcome to Linux Mint
* Documentation: http://www.linuxmint.com
LinkerTester started:
base name: base, id: 0, number of processes: 2
-----
Neighbors: [1]
Insert name: base0, host name: pr5-06, port number: 2011
> Sent message:
>> To: 1, from: 0
>> Tag: my_tag
>> Content: Uspješno ste se povezali procese na više računala
< Received message:
<< From: 1, to: 0
<< Tag: my_tag
<< Content: Uspješno ste se povezali procese na više računala
gaspar13@pr5-06:~$ ./home/student1/gaspar13/public_html/dprojekt $
```

Kao što je ovdje izvedeno za 2 procesa na 2 različita računala, za 10 različitih procesa na 10 računala, u drugoj skripti promijenimo :

```
#!/bin/bash
```

```
PATH=/opt/jdk1.8.0_65/bin:$PATH
```

```
export PATH
```

```
mate-terminal -e 'bash bashC.sh 0 10 user@host1'
```

```
mate-terminal -e 'bash bashC.sh 1 10 user@host2'
```

```
. . . . .
. . . . .
. . . . .
```

```
mate-terminal -e 'bash bashC.sh 9 10 user@host10'
```

Dobivena platforma omogućuje drugim programima vezanim za ovaj kolegij, izvođenje u realističnim uvjetima.

Primjer: Chat (7.poglavlje + prošlogodišnja GUI aplikacija chat-a)

Potrebno je koristiti java klase 7.poglavlja. Pokretanje je ekvivalentno prethodnom primjeru, izuzev naredbe za pokretanje LinkerTester-a, pokrece se :

java Chat \$i \$N causal

(kauzalni uređaj)

Leonora Gašpar
Silva Haberl

