Elektrotehnički fakultet
Univerziteta u Beogradu
Katedra za računarsku tehniku i informatiku

Praktikum iz Operativnih sistema

- rešenja za 2007. godinu -

Verzija: 3.6.2009 18:40

1. Otvorena knjiga

1. **Napisati na programskom jeziku Java programski kod servera koji može istovremeno da opslužuje više klijenata. Za opsluživanje pojedinačnog klijenta služi metod** `void serveClient(Socket s)`**. Za svakog klijenta koji se prijavi, napraviti posebnu nit za opsluživanje. Implementirati metod** `void serveClient(Socket s)`**, koji će svaku primljenu poruku bez izmena prosleđivati istom klijentu.**

```
//EchoClient.java
import java.net.*;
import java.io.*;

public class EchoClient implements Runnable
{
     private Socket socket;
     private Thread nit = new Thread(this);

     public EchoClient(Socket sock)
     {
           socket = sock;
           nit.start();
     }

     public void run()
     {
           try
           {
                 EchoServer.serveClient(socket);
           }
           catch(IOException e){}
     }
}
```

```
//EchoServer.java
import java.io.*;
import java.net.*;

public class EchoServer
{
          public static void main(String[] args) throws IOException
          {
          int port = 4000;

          ServerSocket serverSocket = null;

          try
          {
                serverSocket = new ServerSocket(port);
          }
          catch (IOException e)
          {
                System.err.println("Could not listen on port: " + port);
                System.exit(1);
          }

          while(true)
          {
                Socket clientSocket = null;
                try
                {
                          clientSocket = serverSocket.accept();
                }
                catch (IOException e)
                {
                          System.err.println("Accept failed.");
                          break;
                }

                System.out.println("New client accepted!");
                new EchoClient(clientSocket);

          }
```

```
            serverSocket.close();
      }

      public static void serveClient(Socket s)throws IOException
      {
                  PrintWriter out = new PrintWriter(s.getOutputStream(),
true);
            BufferedReader in = new BufferedReader(
                        new InputStreamReader(
                              s.getInputStream()));
            String inputLine, outputLine;
            while ((inputLine = in.readLine()) != null)
            {
                  System.out.println("Primljena poruka: " + inputLine);
                  outputLine = "done " + inputLine;
                  out.println(outputLine);
                  if (inputLine.equals("end"))
                              break;
            }
            out.close();
            in.close();
            s.close();
      }

}
```

2. Napisati na programskom jeziku Java programski kod servera koji opslužuje jednog po jednog klijenta, i posle zatvaranja svake konekcije počinje da sluša na sledećem portu (npr. počinje sa slušanjem na portu 4000, posle opsluživanja prvog klijenta, nastavlja da sluša na portu 4001 i tako redom zaključno sa portom 5000). Primljene poruke klijenata treba vratiti nazad klijentima i ispisati na serverskoj strani na standardnom izlazu.

```
//EchoServer.java
import java.net.*;
import java.io.*;

public class EchoServer
{
     private static final int MIN_PORT=4000;
     private static final int MAX_PORT=5000;

      public static void main(String[] args) throws IOException
     {
           int port = MIN_PORT;
           while(port <= MAX_PORT)
           {
                 ServerSocket serverSocket = null;

                 try
                 {
                       serverSocket = new ServerSocket(port);
                 }
                 catch (IOException e)
                 {
                       System.err.println("Could not listen on port: "
                             + port);
                       System.exit(1);
                 }
                 Socket clientSocket = null;
                 try
                 {
                       clientSocket = serverSocket.accept();
                 }
                 catch (IOException e)
                 {
                       System.err.println("Accept failed.");
```

```
                    System.exit(1);
                }
                PrintWriter out = new
                    PrintWriter(clientSocket.getOutputStream(), true);
                BufferedReader in = new BufferedReader(
                    new InputStreamReader(
                        clientSocket.getInputStream()));
                String inputLine, outputLine;
                int broj;
                while ((inputLine = in.readLine()) != null)
                {
                    System.out.println("Primljena poruka: "
                        + inputLine);
                    outputLine = inputLine;
                    out.println(outputLine);
                    if (inputLine.equals("end"))
                        break;
                }
                out.close();
                in.close();
                clientSocket.close();
                serverSocket.close();
                port++;
            }
        }
}
```

3. Napisati na programskom jeziku Java programski kod servera koji opslužuje jednog po jednog klijenta. Primljene poruke klijenata treba vratiti nazad klijentima i ispisati na serverskoj strani na standardnom izlazu. U slučaju da poruka sadrži samo broj, treba prestati sa slušanjem na tekućem portu i početi sa slušanjem na portu određenom brojem iz poruke.

```
//EchoServer.java
import java.net.*;
import java.io.*;

public class EchoServer
{
     private static final int MIN_PORT=1000;
     private static final int MAX_PORT=5000;

     public static void main(String[] args) throws IOException
     {
          int port = 4000;
          while(true)
          {
               ServerSocket serverSocket = null;

                    try
                    {
                    serverSocket = new ServerSocket(port);
               }
               catch (IOException e)
               {
                    System.err.println("Could not listen on port: "
                         + port);
                    System.exit(1);
               }
               Socket clientSocket = null;
               try
               {
                    clientSocket = serverSocket.accept();
               }
               catch (IOException e)
               {
                    System.err.println("Accept failed.");
                    System.exit(1);
```

```
                }

                PrintWriter out = new PrintWriter(
                      clientSocket.getOutputStream(), true);
                BufferedReader in = new BufferedReader(
                           new InputStreamReader(
                                 clientSocket.getInputStream()));
                String inputLine, outputLine;
                int broj;
                while ((inputLine = in.readLine()) != null)
                {
                      System.out.println("Primljena poruka: "
                                 + inputLine);
                      outputLine = inputLine;
                      out.println(outputLine);
                      if (inputLine.equals("end"))
                                 break;
                      try
                      {
                                 if ((broj = Integer.parseInt(inputLine))
                                       >= MIN_PORT && broj <= MAX_PORT)
                                 {
                                 port = broj;
                                       break;
                                 }
                      }
                      catch(NumberFormatException e){}
                }
                out.close();
                in.close();
                clientSocket.close();
                serverSocket.close();
          }
     }
}
```

4. **Napisati** `bash shell script` **pod imenom** `ludilo` **koji svim fajlovima u tekućem direktorijumu, takvim da im se ime završava nekim od zadatih sufiksa, dodaje na kraj imena** `-1`**. Sufiksi se zadaju kao parametri skripta.**

   Napisati shell komandu koja pokreće skript `ludilo` **za sve izvorne fjlove na programskim jezicima** `C` **ili** `C++`**.**

```
#!/bin/bash

SUFFIX="-1"

for i in $*
do
     for j in ./*"$i"
     do
          if [ -f "$j" ]
          then
               mv "$j" "$j$SUFFIX"
          fi
     done
done
```

```
./ludilo.sh .c .cpp
```

5. Napisati `bash shell script` pod imenom guards koji u tekućem direktorijumu pronalazi sve datoteke koje imaju sufuks `.h` i koje u jednom od prvih 5 redova imaju direktivu `#ifndef`.

Napisati drugi `bash shell script` koji koristeci napisanu komandu na svaku od nađenih datoteka dodaje sufiks `.safe`.

```
#!/bin/bash
#guards.sh

for dat in ./*.h
do
     if [ -f "$dat" ]; then
          head -n 5 "$dat" | grep -q "#ifndef"
          if [ "$?" -eq "0" ]; then
               echo "${dat#./}"
          fi
     fi
done
```

```
#!/bin/bash
#safe.sh

SUFFIX=".safe"

if [ "$#" -ne "1" ] ; then
     echo "error:usage:safe.sh path_to_guards.sh" >&2
     exit 1
fi

for dat in $($1)
do
     if [ -f  "$dat" ]; then
          mv "$dat" "$dat$SUFFIX"
     fi
done
```

6. **Napisati** `bash shell script` **pod imenom** `newuser` **koji proverava da li na sistemu postoji korisnik sa imenom koje je zadato kao prvi parametar komandne linije. Ako takav korisnik ne postoji, kreira korisnika sa zadatim imenom. U suprotnom, kreira korisnika sa imenom koje je zadato kao drugi parametar komandne linije. Na kraju, script treba da ispiše pod kojim je imenom stvorio korisnika. Ukoliko su oba imena zauzeta, script treba da ispiše odgovarajuću poruku.**

```
#!/bin/bash

if [ "$UID" -ne "0" ];then
     echo "error:samo superuser moze napraviti novog korisnika" >&2
     exit 1
fi

if [ "$#" -ne "2" ]; then
     echo "error:usage:newuser.sh username1 username2" >&2
     exit 2
fi



grep -q "$1" /etc/passwd
if [ "$?" -ne "0" ];then
     adduser "$1"
     echo "Napravio korisnika $1"
else

     grep -q "$2" /etc/passwd
     if [ "$?" -ne "0" ];then
           adduser "$2"
           echo "Napravio korisnika $2"
     else
           echo "error:oba username su zauzeta" >&2
           exit 3
     fi
fi
```

7. Koristeći POSIX threads napisati kod na jeziku C ili C++ za dve niti, proizvođača i potrošača. Niti razmenjuju podatke preko dva bafera koji naizmenično menjaju svoje uloge. Dok proizvođač upisuje podatke u prvi bafer, potrošač uzima podatke iz drugog bafera; kada proizvođač napuni prvi bafer i potrošač isprazni drugi, baferi menjaju uloge. Napisati kompletan programski kod proizvođača i potrošača uz sinhronizaciju preko uslovnih varijabli (pthread_cond_t). Pretpostaviti da je bafer statički niz. Napomena:u rešenju se koristi klasa Thread koja je data na vežbama.

```
#ifndef _consumer_h_
#define _consumer_h_

#include "Thread.h"
#include <cstdlib>

using namespace std;

class Consumer:public Thread
{
     int *a,*b,lena,lenb;
     pthread_cond_t* cond1,*cond2;
     pthread_mutex_t *mutex;
     int *pun;
     int *prazan;
     void switch_buffers();
     public:
     Consumer(int *aa,int lenBuff1,int *bb,int lenBuff2, pthread_cond_t
          *c1,pthread_cond_t *c2,pthread_mutex_t *m,int *pra,int *p);
     protected:
     virtual void run();
};
#endif //endof Consumer.h
```

```
//Consumer.cpp
#include <iostream>
#include "Consumer.h"
using namespace std;

Consumer::Consumer(int *aa,int lenBuff1,int *bb,int lenBuff2,
     pthread_cond_t *c1,pthread_cond_t *c2,pthread_mutex_t *m,
          int *pra,int *p)
{
     pun = p;
     prazan = pra;
     a = aa;
     lena = lenBuff1;
     b = bb;
     lenb = lenBuff2;
     cond1 = c1;
     cond2 = c2;
     mutex = m;
}

void Consumer::switch_buffers()
{
     int *c = a;
     a = b;
     b = c;
     int lc = lena;
     lena= lenb;
     lenb = lc;
}

void Consumer::run()
{
     while(22)
     {
          pthread_mutex_lock(mutex);
          if (*pun == 0)
          {
               pthread_cond_wait(cond1,mutex);
          }
```

```
            *pun=0;
            pthread_mutex_unlock(mutex);

                  for(int i = 0;i < lenb;i++)
            {
                  int d = b[i];
                  cout<<"Consumer: "<<d<<endl;
                  sleep((unsigned)(rand()/(RAND_MAX + .1)*5));
            }
            pthread_mutex_lock(mutex);
            if (*prazan == 0)
            {
                  *prazan = 1;
                  pthread_cond_signal(cond2);
            }
                  pthread_mutex_unlock(mutex);

            switch_buffers();

      }
}
```

```
#ifndef _producer_h_
#define _producer_h_

#include "Thread.h"
#include <pthread.h>
#include <cstdlib>

using namespace std;

class Producer:public Thread
{
     int *a, *b,lena,lenb;
     pthread_mutex_t* mutex;
     pthread_cond_t *cond1,*cond2;
     int *prazan;
     int *pun;
     void switch_buffers();

     public:
     Producer(int* buff1,int lenBuff1,int* buff2,
          int lenBuff2,pthread_cond_t* c1,pthread_cond_t* c2,
               pthread_mutex_t* m,int *pra,int *p);

     protected:
     virtual void run();
};

#endif //endof Producer.h
```

```
//Producer.cpp
#include "Producer.h"

Producer::Producer(int* buff1,int lenBuff1,int* buff2,
     int lenBuff2,pthread_cond_t* c1,pthread_cond_t* c2,
          pthread_mutex_t* m,int *pra,int *p)
{
     prazan = pra;
     pun = p;
     a = buff1;
     lena =lenBuff1;
     b = buff2;
     lenb = lenBuff2;
     cond1 = c1;
     cond2 = c2;
     mutex = m;
}

void Producer::switch_buffers()
{
     int* c = a;
     a = b;
     b = c;
     int lenc = lena;
     lena= lenb;
     lenb = lenc;
}

void Producer::run()
{
     for (int i = 0;i < lenb;i++)
     {
          int d = (int)(rand()/(RAND_MAX + .1)*20);
          b[i] = d;
          cout<<"Produced: "<<d<<endl;
          sleep((unsigned)(rand()/(RAND_MAX + .1)*5));
     }

     pthread_mutex_lock(mutex);
```

```
    if(*pun == 0)
    {
          *pun = 1;
          pthread_cond_signal(cond1);
    }
    *prazan = 1;

    pthread_mutex_unlock(mutex);

    while(22)
    {
          pthread_mutex_lock(mutex);
                if(*prazan == 0)
                {
                      pthread_cond_wait(cond2,mutex);
                }
                *prazan = 0;
                pthread_mutex_unlock(mutex);

                for (int i = 0;i < lena;i++)
                {
                      int d = (int)(rand()/(RAND_MAX + .1)*20);
                      a[i] = d;
                      cout<<"Produced: "<<d<<endl;
                      sleep((unsigned)(rand()/(RAND_MAX + .1)*5));
                }

                switch_buffers();

                pthread_mutex_lock(mutex);
                if (*pun == 0)
                {
                      *pun = 1;
                      pthread_cond_signal(cond1);
                }
          pthread_mutex_unlock(mutex);
    }
}
```

```
//main.cpp
#include "Producer.h"
#include "Consumer.h"

#include "Thread.h"
#include <iostream>
using namespace std;

int main(void)
{
     int lena,lenb;
     cout<<"Duzina prvog bafera?    ";cin>>lena;
     cout<<"Duzina drugog bafera?    ";cin>>lenb;

     int *a = new int[lena],*b = new int[lenb];
     int s1 = 0,s2 = 0;

     pthread_cond_t c1,c2;
     pthread_cond_init(&c1,0);
     pthread_cond_init(&c2,0);
     pthread_mutex_t mutex;
     pthread_mutex_init(&mutex,0);

     Thread* p = new Producer(a,lena,b,lenb,&c1,&c2,&mutex,&s1,&s2);
     Thread* c = new Consumer(a,lena,b,lenb,&c1,&c2,&mutex,&s1,&s2);
      p->start();
     c->start();

     Thread::dispose(p);
     Thread::dispose(c);
     delete a;
     delete b;

     pthread_cond_destroy(&c1);
     pthread_cond_destroy(&c2);
     pthread_mutex_destroy(&mutex);

     pthread_exit(0);
     return 0;
}
```

8. Koristeći POSIX threads napisati kod na jeziku C ili C++ za dve niti, proizvođača i potrošača. Potrošač dobija podatke preko jednog kružnog bafera. Ukoliko potrošač sustigne proizvođača (isprazni bafer), treba da sačeka dok se ne pojave podaci koje treba da čita. Ukoliko proizvođač zaguši potrošača (prepuni bafer), treba da sačeka dok potrošač ne pročita neki podatak da bi mogao da nastavi. Kad bafer nije niti prazan niti pun, proizvođač i potrošač čekaju neko slučajno vreme pre sledećeg pristupa baferu. Napisati kompletan programski kod proizvođača i potrošača uz sinhronizaciju preko uslovnih varijabli (pthread_cond_t), kao i glavni program koji definiše potrebne podatke, te stvara i pokreće opisane dve niti. Pretpostaviti da je bafer statički niz.

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <unistd.h>

#define N 100

int head = 0,tail = 0,cnt = 0;

int buffer[N];

pthread_mutex_t mutex;
pthread_cond_t full,empty;

void* producer(void *arg)
{
     while(22)
     {
          int d;
          pthread_mutex_lock(&mutex);
          if (cnt == N)
          {
               pthread_cond_wait(&full,&mutex);
          }
          if(cnt == 0)
          {
               pthread_cond_signal(&empty);
          }
          d = (int)(rand()/(RAND_MAX + 1)*30);
```

```
            buffer[head] = d;
            printf("Produced: %d\n",d);
            head = (head + 1)%N;
            cnt++;
            pthread_mutex_unlock(&mutex);
            sleep((unsigned)(rand()/(RAND_MAX + .1)*5));
      }
      pthread_exit(NULL);
}

void* consumer(void *arg)
{
      while(22)
      {
            pthread_mutex_lock(&mutex);
            if(cnt == N)
            {
                  pthread_cond_signal(&full);
            }

            if(cnt == 0)
            {
                  pthread_cond_wait(&empty,&mutex);
            }

            printf("Consumed: %d\n",buffer[tail]);
            tail = (tail+1)%N;
            cnt--;
            pthread_mutex_unlock(&mutex);
            sleep((unsigned)(rand()/(RAND_MAX + .1)*5));
      }
      pthread_exit(NULL);
}
```

```
int main()
{
     pthread_t tprod,tcon;
     pthread_attr_t attr;

     srand(time(NULL));

     pthread_mutex_init(&mutex,NULL);
     pthread_cond_init(&full,NULL);
     pthread_cond_init(&empty,NULL);
     pthread_attr_init(&attr);

     pthread_attr_setdetachstate(&attr,PTHREAD_CREATE_JOINABLE);

     pthread_create(&tprod,&attr,producer,NULL);
     pthread_create(&tcon,&attr,consumer,NULL);



     pthread_join(tprod,NULL);
     pthread_join(tcon,NULL);



     pthread_attr_destroy(&attr);
     pthread_mutex_destroy(&mutex);
     pthread_cond_destroy(&full);
     pthread_cond_destroy(&empty);

     pthread_exit(NULL);

     return 0;
}
```

9. Koristeći POSIX threads napisati kod na jeziku C ili C++ za dve niti, proizvođača i potrošača. Potrošač dobija podatke podatke preko jednog kružnog bafera. Ukoliko se desi da potrošač sustigne proizvođača, i obrnuto, ukoliko proizvođač sustigne potrošača (prepuni bafer) treba prijaviti grešku i prekinuti obe niti. Proizvođač i potrošač čekaju neko slučajno vreme pre sledećeg pristupa baferu. Napisati kompletan programski kod proizvođača i potrošača uz sinhronizaciju preko uslovnih varijabli (pthread_cond_t), kao i glavni program koji definiše potrebne podatke, te stvara i pokreće opisane dve niti. Pretpostaviti da je bafer statički niz.

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <unistd.h>

#define N 100

int head = 0,tail = 0,cnt = 0;

int buffer[N];

pthread_mutex_t mutex;
pthread_cond_t full,empty;

void* producer(void *arg)
{
     while(22)
     {
          int d;
          pthread_mutex_lock(&mutex);
          if (cnt == N)
          {
               cnt = -1;
               pthread_cond_wait(&full,&mutex);
               printf("Bafer prepunjen:Obustavljen rad!\n");
               pthread_mutex_unlock(&mutex);
               pthread_exit(NULL);
          }
          if(cnt == -1)
          {
               pthread_cond_signal(&empty);
```

```
                pthread_mutex_unlock(&mutex);
                pthread_exit(NULL);
          }
          d = (int)(rand()/(RAND_MAX + .1)*30);
          buffer[head] = d;
          printf("Produced: %d\n",d);
          head = (head + 1)%N;
          cnt++;
          pthread_mutex_unlock(&mutex);
          sleep((unsigned)(rand()/(RAND_MAX + .1)*3));
     }
     pthread_exit(NULL);
}

void* consumer(void *arg)
{
     while(22)
     {
                pthread_mutex_lock(&mutex);
          if(cnt == -1)
          {
                cnt = N;
                while(cnt > 0)
                {
                     printf("Consumed: %d\n",buffer[tail]);
                     tail = (tail+1)%N;
                     cnt--;
                }
                pthread_cond_signal(&full);
                pthread_mutex_unlock(&mutex);
                pthread_exit(NULL);
          }

          if(cnt == 0)
          {
                cnt = -1;
                pthread_cond_wait(&empty,&mutex);
                pthread_mutex_unlock(&mutex);
                printf("Bafer je prazan:Obustavljen rad\n");
                pthread_exit(NULL);
```

```
            }
            printf("Consumed: %d\n",buffer[tail]);
            tail = (tail+1)%N;
            cnt--;
            pthread_mutex_unlock(&mutex);
            sleep((unsigned)(rand()/(RAND_MAX + .1)*5));
        }
        pthread_exit(NULL);
}

int main()
{
        pthread_t tprod,tcon;
        pthread_attr_t attr;

        srand(time(NULL));

        pthread_mutex_init(&mutex,NULL);
        pthread_cond_init(&full,NULL);
        pthread_cond_init(&empty,NULL);
        pthread_attr_init(&attr);

        pthread_attr_setdetachstate(&attr,PTHREAD_CREATE_JOINABLE);

        pthread_create(&tprod,&attr,producer,NULL);
        pthread_create(&tcon,&attr,consumer,NULL);



        pthread_join(tprod,NULL);
        pthread_join(tcon,NULL);



        pthread_attr_destroy(&attr);
        pthread_mutex_destroy(&mutex);
        pthread_cond_destroy(&full);
        pthread_cond_destroy(&empty);

        pthread_exit(NULL);

        return 0;
```

```
}
```

2. Zatvorena knjiga

1. Šta je rezultat rada sledećih komandi:

```
alias lm="ls -l"
```

Postavlja smenu simbola, tako da kad god korisnik otkuca **lm**, to će se pre izvršavanja smeniti sa **ls -l**.

```
pwd
```

Ispisuje ime tekućeg direktorijuma.

```
ls ~/. -l
```

Lista sadržaj home direktorijuma za korisnika koji je zadao komandu, zajedno sa dodatnim informacijama o fajlovima: prava pristupa, glasnika, grupu, tip fajla, ...

```
mkfifo x1
```

Pravi **named pipe** pod imenom **x1** u tekućem direktorijumu.

```
rm -rf *
```

Bezuslovno briše kompletan sadržaj tekućeg direktorijuma.

```
kill -9 42456
```

Bezuslovno ubija proces čiji je **PID** jednak **42456**.

```
chmod o+x *.sh
```

Za sve fajlove u tekućem direktorijumu, čije se ime završava sa **.sh**, postavlja pravo pristupa **x** (**execute**), za ostale korisnike (**other**).

```
ps axf | grep lamboot
```

Među svim trenutno aktivnim procesima pronalazi proces sa imenom **lamboot** i tu linuju ispisuje na standarnom izlazu.

```
jobs
```

Prikazuje listu poslova pokrenutih od strane korisnika.

```
man g++
```

Prikazuje man stranicu za program **g++**.

```
tail -f /var/log/messages
```

Ipisuje poslednjih nekoliko linija fajla **/var/log/messages**, opcija **-f** omogućava dodavanje podataka kako se fajl povecava.

```
ls -la
```

Lista sadržaj tekućeg direktorijuma uključujući i skrivene fajlove (one čija imena počinju tačkom), zajedno sa dodatnim informacijama o fajlovima: prava pristupa, glasnika, grupu, tip fajla, ...

```
mkdir pera
```

Kreira direktorijum sa imenom **pera** u tekućem direktorijumu.

```
sudo su
```

Pokrece komandu **su** kao superuser. Komanda **su** menja tekucek korisnika u superuser.

```
killall joe
```

Gasi proces sa imenom **joe**.

```
chmod o-x /home/user1
```

Za direktorijum **/home/user1** ukida pravo **x** (**execute**), za ostale korisnike (**other**).

```
top
```

Ispisuje trenutno aktivne poslove, podatke o tome koliko resursa zauzimaju,itd. Prikaz se osvežava periodično sa nekom periodom T.

```
echo `pwd`
```

Ispisuje rezultat naredbe **pwd**, odnosno tekući direktorijum.

```
echo 'pwd'
```

Ispisuje **pwd**.

```
chmod 664 child1 child2
```

Za fajlove pod imenom **child1** i **child2** u tekućem direktorijumu postavlja prava pristupa takva da vlasnik i grupni vlasnik mogu da čitaju i da pišu, dok ostali korisnici mogu samo da čitaju te fajlove.

```
chmod o+r /home/user1
```

Za direktorijum /home/user1 daje se pravo **r** (**read**), za ostale korisnike (**other**).

2. Detaljno objasniti priloženi ispis dobijen pozivom komande ls -l.

```
drwxr-xr-x 10 pos pos 4096 2007-06-22 11:13 code
lrwxrwxrwx 1 pos pos 24 2007-04-29 17:20 deb_paketi ->
/var/cache/apt/archives/
drwxr-xr-x 3 pos pos 4096 2007-06-23 00:08 Desktop
-rw------- 1 pos pos 266782 2007-05-25 10:18 weblab_20070525_1018.sql
```

Pozivom komande **ls -l** dobijamo spisak fajlova u tekucem direktorijumu. Prvi znak prve kolone oznacava da li se radi o običnom fajlu (**-**), direktorijumu (**d**) ili linku(**l**). Sledećih devet znakova prikazuju prava pristupa za svaku od tri tipa korisnika, **r** - dozvoljeno čitanje, **w** - dozvoljen upis, **x** - dozvoljeno izvršavanje **-** - pravo pristupa

nije dozvoljeno. Prava pristupa za vlasnika su prva tri znaka, druga tri su za vlasničku grupu i poslednja tri su za sve ostale korisnike. Broj u drugoj koloni označava broj fajlova u okviru direktorijuma ili **1** ako se radi o običnom fajlu. Treća kolona oznacava vlasnika fajla, a četvrta vlasničku grupu. Peta kolona oznacava veličinu fajla u bajtovima. Šesta i sedma kolona predstavljaju datum i vreme poslednje modifikacije. Poslednja kolona predstavlja ime fajla. U slučaju da se radi o **symbolic link** fajlu iza imena nalazi se strelica sa putanjom na koju link ukazuje.

3. Detaljno objasiniti prve tri kolone u priloženom ispisu. Ispis je dobijen pozivom komande cat /etc/fstab.

```
/dev/sda6 /                ext3 defaults,errors=remount-ro 0 1
/dev/sda7 /home            ext3 defaults 0 2
/dev/sda1 /media/win_c     ntfs defaults,nls=utf8,umask=007,gid=46 0 1
/dev/sda8 /media/win_d     vfat defaults,utf8,umask=007,gid=46 0 1
/dev/sda5 none             swap sw 0 0
/dev/hda  /media/cdrom0    udf,iso9660 user,noauto 0 0
```

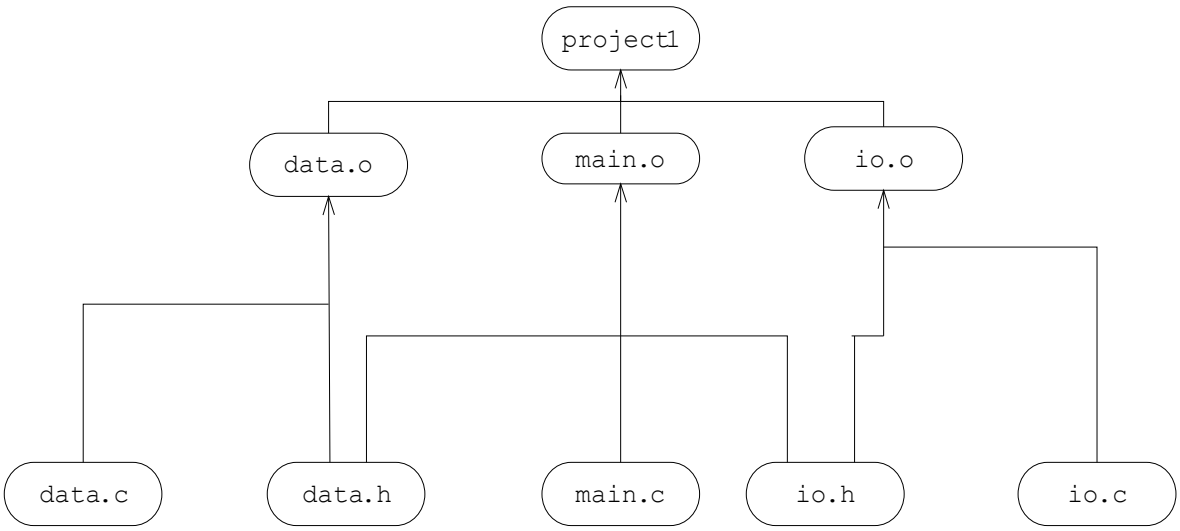
Prva kolona označava fajl u **/dev** direktorijumu koji služi kao neposredna veza sa hardverom. Imena ovih fajlova su simbolička, tako da se lako može odrediti o kom disku (particiji), odnosu uređaju za skladistenje podataka. Fajlovi koji u imenu imaju **sda** su obicno particije na disku (krajnje slovo **a** označava da se radi o prvom hard disku, oznaka za drugi hard disk je **b** itd.), pri čemu primarne particije imaju broj na kraju od **1** do **4**, a logičke particije imaju brojeve veće od **4**. **/dev/hda** je optički uređaj. Druga kolona predstavlja **mounting point**, odnosno direktorijum na kojoj je montiran odgovarajući fajl sistem. **Swap** particije nemaju **mounting point**. Treća kolona označva tip fajl sistema na uređaju. Fajl sistemi **ext3**, **ntfs**, **vfat** se koriste za smeštanje podataka na particijama. **Swap** ožnačava da se taj deo koristi za zamenu stranca virtualne memorije i na njemu nema fajl sistema već operativni sistem pristupa direktno svakom bloku u okviru tog prostora na disku. Fajl sistem **udf** je standardni fajl sistem za optičke medijume.

4. Detaljno objasniti priloženi Makefile i nacrtati graf zavisnosti fajlova.

```
project1: data.o main.o io.o
          gcc data.o main.o io.o -o project1
data.o: data.c data.h
          gcc -c data.c
main.o: data.h io.h main.c
          gcc -c main.c
io.o: io.h io.c
          gcc -c io.c
clean:
          rm -rf *~ *.o *.bak project1
```

Primarna meta ovog fajla je **project1**. On zavisi od fajlova **data.o** **main.o** i **io.o**. Ako fajl **project1** ne postoji ili je on stariji od fajlova od kojih zavisi izvršiće se komanda **gcc data.o main.o io.o -o project1**. Na sličan način se određuje pravila za pravljenje ostalih meta. Samo je meta **clean** različita. Ona je lažna meta i ne zavisi ni od jednog fajla. Ona služi za brisanje svega sto je makefile napravio i ona

se može pokrenuti samo ako se prilikom poziva make navede kao parametar (**`make clean`**).



Grafik zavisnosti fajlova

5. a) Šta je i čemu služi pipe?

b) Prikazati kako se stvara i koristi named pipe (prikazati i čitanje i pisanje).

c) Napisati shell komandu koja će u pozadini da pokrene program AB koji će da pročita sve podatke iz cevovoda cev0, da ih obradi i rezultate ispiše u cevovod cev1.

a) **`Pipe`** je Unix-ov primitivni način komunikacije između procesa. Može biti imenovan kda se ponaša kao fajl ili neimenovan, kada služi za prosleđivanje izlaza neke naredbe ulazu druge.

b)

```
mkfifo cevovod
ls > cevovod
cat < cevovod
```

c)

```
./AB < cev0 > cev1 &
```