1. Opišite naloge jedra pri sistemu UNIX. Kaj so naloge jedra in kaj naloge lupine ob izvršitvi ukaza *rm moja_datoteka*?

Naloge jedra so:
- nadzor nad procesi, rojevanje in pobijanje procesov, upravljanje z resursi sistema
- oskrba procesov s podatki, zagotavljanje navideznega stroja (edinega procesa) za proces
- sprejema strojne prekinitve periferije, delo z datotekami, delo z virtualnim spominom, mrežni protokoli, večopravilnost (niti)

Ko uporabnik natipka ukaz (*rm datoteka*) v ukazni vrstici, lupina kreira (fork) nov proces (klon lupine), poišče kodo za ukaz rm, skopira kodo preko kode kloniranega procesa in kernelu sporoči, da je nov proces pripravljen za izvedbo. Osnovnemu procesu rečemo "oče", novemu pa "otrok".

2. V lupini *bash* napišite program *poslji*, ki za vsakega uporabnika prijavljenega na sistemu vpraša, ali naj mu pošlje obvestilo, shranjeno v datoteki *obvestilo*. Če je odgovor da (D), mu to obvestilo pošlje, sicer pa ne.

```
<*datoteka poslji*>
#!/bin/bash
who -q > seznam
for i in `sed '/\#/,$d' seznam`
do
  echo "Posljem obvestilo $i ?"
  read odgovor
  case $odgovor in
    d|D) write $i <obvestilo;;
    n|N) echo naslednji;;
    *) echo pravilno samo d ali n;;
  esac
done
rm seznam
<*konec poslji*>
```

3. V programskem jeziku C napišite program, ki je ekvivalenten znanemu programu *wc* operacijskega sistema UNIX. Predvidite tudi naslednja stikala: *-l* (štetje vrstic), *-w* (štetje besed), *-c* (štetje znakov).

```
<*zacetek wrdc.c*>
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#define true 1
#define false 0

main(int argc, char *argv[])
{
  FILE *fptr;
  int inword, numchar, numword, numline;
  char ch;
  char str;
  numchar=0;
  numword=0;
  numline=0;
  inword = false;
  fptr = fopen(argv[1], "r");
  if (fptr == NULL) {
    printf("Uporaba: wrdc ime_datoteke -opcije\n");
    printf("Opcije: -l <vrste, -c <znaki, -w <besed\n");
    exit(0);
    } //endif
  ch = fgetc(fptr);
  while (ch!=EOF) {
       if (isalpha(ch)) numchar++;
       if (inword && isspace(ch)) {
         numword++;
         inword=false;}
       else if ( !inword && isalnum(ch))
              inword=true;
       if (ch=='\n') numline++;
       ch=fgetc(fptr);
    } //endwhile

//izpis
if (argv[2]!=NULL) {
  if (strstr(argv[2],"-c")) printf("Stevilo crk:
%d\n",numchar);
  else if (strstr(argv[2],"-w")) printf("Stevilo besed:
%d\n",numword);
  else if (strstr(argv[2],"-l")) printf("Stevilo vrstic:
%d\n",numline);
  }
  else printf("crke: %d  besede: %d   vrste:
%d\n",numchar,numword,numline);
  printf("konec\n");
}
 <* konec wrdc.c *>
```

4. V programskem jeziku JAVA napišite program, ki je ekvivalenten znanemu programu *wc* operacijskega sistema UNIX. Predvidite tudi naslednja stikala: *-l* (štetje vrstic), *-w* (štetje besed), *-c* (štetje znakov).

wordcount v javi

```
import java.io.*;

class WordCount {

public static void main(String args[]) throws Exception {
  int numword = 0;
  int numchar = 0;
  int numline = 0;
  boolean inword = false;
  char ch;
  int input;

  if (args.length > 1)
   {

   FileInputStream fin = new FileInputStream(args[1]);

   while (( input = fin.read()) != -1)
     {
     ch = (char)input;

     if ( Character.isLetterOrDigit(ch)) { numchar++; }
     if ( inword && Character.isWhitespace(ch)) {
        numword++;
        inword = true;}
     else if ( !inword && Character.isLetterOrDigit(ch))
{ inword = true; }
     if (ch == '\n') { numline++; }
     }

   if (args.length == 2)  {
     if (args[2] == "-l") { System.out.println("Vrste: " +
numline);}
     if (args[2] == "-w") { System.out.println("Besede: " +
numword);}
     if (args[2] == "-c") { System.out.println("Znaki: " +
numchar);}
     }
   else {System.out.println("V:" + numline + " B:" + numword
+ " Z:" +
    numchar);}
   }
   else System.out.println("napaka pri klicu");
  }
}
```

17. januar 2002

1. Kako med seboj komunicirajo uporabnik, procesi, jedro in strojna oprema pri operacijskem sistemu UNIX (10%)

??

2. Kaj so naloge jedra pri operacijskem sistemu UNIX? (15%)

Naloge jedra so:
- nadzor nad procesi, rojevanje in pobijanje procesov, upravljanje z resursi sistema
- oskrba procesov s podatki, zagotavljanje navideznega stroja (edinega procesa) za proces
- sprejema strojne prekinitve periferije, delo z datotekami, delo z virtualnim spominom, mrežni protokoli, večopravilnost (niti)

3. Napišite ukazno datoteko *gostota* v lupini *bash*, ki beleži gostoto uporabe nekaterih procesov na sistemu. Ukazna datoteka sprejema en argument, ki je ime datoteke, ki hrani imena procesov, ki nas zanimajo. Evidenco vodite za ne več kot dva dneva nazaj. Gostoto preverjajte vsakih pet minut, evidenca pa naj hrani tudi informacijo o datumu, uri in seveda procesih.

```
#!/bin/bash

if [ $# -ne 1 ]
  then
    echo 'dovoljen samo en parameter: ime datoteke s
seznamom procesov'
  else                # tu se pa zacne resno delo
    echo >evidenca1
    echo >evidenca2
    echo >evidenca
    let dan=`date +%j` # date +%j vrne stevilko dneva v letu
    while true         #neskoncna zanka
      do
        let ndan=`date +%j`
        if [ $dan -ne $ndan ] #arhiv za dva dni nazaj
          then
            rm evidenca2
            cp evidenca1 evidenca2
            cp evidenca evidenca1
            echo > evidenca
          fi
        date >>evidenca
        for neki in `cat $1` # NAVEDNICE!!!!
          do
            let stevec=0
            ps -A|cut -c25-40 |grep $neki >seznam
            for drugo in `cat seznam`
            do
             let stevec=$stevec+1
            done # FOR
            echo $neki : $stevec krat >>evidenca
            rm seznam
          done # FOR
        echo >>evidenca
        sleep 300 #spanje 300 sekund
      done # WHILE
  fi
```

4. V programskem jeziku *C* napisite funkcijo, ki ji navedemo kot parameter polje števil (in dolžino polja), funkcija pa vrne **srednjo (povprečno)** vrednost teh števil. Napišite tudi program, ki prebere ta števila iz dane datotekein pokliče napisano funkcijo ter izpiše rezultat.

```
#include <stdio.h>
#define MAX 100
double srednja(int num[MAX], int i);

main(int argc, char *argv[]) {
  FILE *fptr;
  int num[MAX];
  int i;
  i=0;
  fptr=fopen(argv[1], "r");
  while (!feof(fptr)) {
    fscanf(fptr,"%d",&num[i]);
    i=i+1;
    }
  printf("neki dugaja %d :  %lf\n", i,  srednja(num, i));
  printf("konec\n");
  fclose(fptr);
}

double srednja(int num[MAX], int i) {
  double fsrednja;
  int j;
  int vsota;
  vsota=0;
  for (j=0; j<=i;j++)
    vsota = vsota + num[j];
  fsrednja= ((float) vsota)/ i;
  return(fsrednja);
}
```

5. V jeziku *Java* predvidite razred *KompleksnoStevilo* in uvedite metodo, ki predstavlja nastavitev vrednosti objektov iz tega razreda (nastavitev vrednosti števil), ter metodo, ki predstavlja izpis takega števila (pravzaprav vrednosti objekta). Nastavite krajšo aplikacijo ali aplet, ki predstavlja preizkus uvedenega razreda.

<datoteka primerComplex.java>

```
import java.io.*;
class primerComplex {
  public static void main(String[] args) {
    Complex a1 = new Complex(8,7);
    Complex b1 = new Complex(22,4);
    Complex c1 = new Complex();
    System.out.println(" Prvo stevilo: " + a1.real + " i*" +
a1.imag);
    System.out.println(" Drugo stevilo: " + b1.real + " i*"
+ b1.imag);
    c1.vsota(a1,b1);
    System.out.println(" Njuna vsota: " + c1.real + " i*" +
c1.imag);
  }
}

class Complex {
  int real;
  int imag;
  Complex() {   // konstruktor
    real = 0;
    imag = 0;
   }
  Complex(int re, int im) {  // metoda za dodeljevanje
vrednosti
    real = re;
    imag = im;
   }

  public void vsota(Complex a, Complex b) {  // metoda za
vsoto
    real = a.real + b.real;
    imag = a.imag + b.imag;
   }
}     // definiran razred complex
```

1. Razložite vlogo sistemskih klicev na sistemu UNIX. Naštejte bistvene razlike pri delu z datotečnim sistemom z uporabo sistemskih klicev in pri delu z datotečnim sistemom z uporabo funkcij knjižnice.

V UNIXu vsi uporabniški programi in aplikacije uporabljajo vmesnik sistemskih klicev za dostopanje do sistemskih viro kot so diski, tiskalniki, spomin,... Vmesnik sistemskih klicev v UNIXu ponuja množico sistemskih klicev (C funkcij)
Naloga vmesnika sistemskih klicev je zagotoviti celovitos in nedotakljivost sistema (integriteto). Ves dostop do strojne opreme na nizkem nivoju poteka pod kontrolo operacijskega sistema, s tem pa je uporabnikovim programom onemogočeno onesposobljenje sistema.
Operacijski sistem ob sprejemu sistemskega klica preveri njegovo avtentičnost ali dovoljenje, nato pa ga izvede v imenu uporabniškega programa ter vrne rezultat. Če je zahteva napačna ali pa nedovoljena, sistem zahteve ne izvede in programu vrne kodo napake.
Sistemski klici so dosegljivi kot množica funkcij v Cju, saj je tudi UNIX napisan v tem jeziku
Sistemski klici so ***edini*** način za dostop do funkcij jedra kot so datotečni sistem, večopravilni mehanizem, komunikacija med procesi.

Razlike pri delu med sistemskimi klici in knjižnicami??????????????????????????????????????

2. Razložite pozicijske argumente in vidljivost spremenljiv v lupini *bash*. Kako je s pozicijskimi parametri ukazne datoteke in kako s pozicijskimi parametri funkcij? Kako je s spremenljivkami ukazne datoteke in kako s spremenljivkami funkcij?

Ko bash skript kliče funkcijo, se argumenti pri klicu funkcije spremenijo v pozicijske parametre. Uporabljamo jih na isti način, kot pozicijske argumente skripte. Argument št.0 (#0 - ime skripte) ostane nespremenjen.
Če se v funkciji izvrši komanda *return*, se funkcija konča, skripta se nadaljuje takoj za klicem funkcije, pozicijski argumenti pa se vmejo na prvotno vrednost. Če ukaz *return* vrne numerično vrednost, je dosegljiv kot izhodni status funkcije, drugače pa je izhodni status funkcije enak statusu zadnjega ukaza pred *return*.
Spremenkjivke v skriptu so globalne. Spremenljivke, ki so lokalne v funkciji, se deklarirjo s predpono *local*. Te spremenljivke so vidne samo v okviru funkcije, ki jih vsebuje.

3. Ukaz *find [dir] file* išče datoteko *file* v delovnem direktoriju ali direktoriju *dir*, nato pa rekurzivno po datotečni strukturi od tega direktorija navzdol. Vzemimo, da tega ukaza ni na sistemu in ga nadomestimo z ukazno datoteko napisano v lupini *bash*. Napiši tako datoteko.

```
#!/bin/bash

list () {
  for i in `ls $1`
  do
    if (test -d $i) ; then
      ls $i > datoteka1
      for j in `cat datoteka1`
        do
        if ( $j -eq $dat ); then
          echo "Nahaja se na $i"
        fi
      done
    fi
  done
}

dat = $1
ls ./ > datoteka
for i in `cat datoteka`
  do
  if (test -d $i); then
    list $i
  fi
  if ( $i -eq $dat );then
    echo "datoteka je v direktoriju: $i"
    exit
  fi
done
```

4. Imamo dva programa, ki tečeta na različnih računalnikih, povezanih med seboj v mreži. Kako lahko program A pošlje pogramu B polje n celoštevilčnih podatkov? Povej le, kaj veš o najbolj primernem principu take komunikacije.

??

5. V programskem jeziku Java napiši kodo aplikacije, ki tabelira vrednosti x,y pozicije točke na krogu enote, če pri tem povečujemo kot alfa od 0 do 360 stopinj s prirastkom, ki ga podajamo kot parameter aplikacije.

```
import java.lang.Math;

class Krog {
  public static void main(String[] args) {
    double x;
    int korak;
    if (args.length == 1){
       korak = Integer.parseInt(args[0]);
       for (int i=0; i<=360; i=i + korak ) {
         x = Math.toRadians(i);
         System.out.println(i + ": x, y: " + Math.cos(x) +
", " + Math.sin(x));
       }
      }
    else
      System.out.println("Napaka pri klicu!!");
    }
}
```

1. Napišite program v lupini, ki pregleda datoteko /etc/passwd in izpiše vse lupine, ki jih uporabniki sistema uporabljajo, ter koliko uporabnikov uporablja posamezno lupino.

```
cut -f 7 -d : /etc/passwd | sort -r | uniq -c >bla
cut -c 1-5 bla > num
cut -c 6- bla | grep "/" > shell
echo brez lupine >>shell
paste shell num
rm num
rm shell
```

2. Napišite program v lupini, ki za vse datoteke v nekem direktoriju ter rekurzivno v vseh poddirektorijih izpiše datum njihov zadnje spremembe in zadnjega dostopa. Direktorij naj bo podan v ukazni vrstici.

```
ls -alRSa $1 > acesstime.log
ls -alRSc $1 > changetime.log
cat acesstime.log | sort +8 > accessSorted
cat changetime.log | sort +8 > changeSorted
cut -c43-55 accessSorted >a
cut -c43-70 changeSorted >b
paste a b > x
```

3. Napišite program v lupini, ki preišče vse datoteke v podanem direktoriju in njegovih poddirektorijih in izpiše najnovejšo datoteko (po datumu zadnje spremembe).

```
ls  -l -R | cut -c 43- |
sed -e s/Jan/01/ | sed -e s/Feb/02/ |sed -e s/Mar/03/ |sed
-e s/Apr/04/ |
sed -e s/May/05/ | sed -e s/Jun/06/ |sed -e s/Jul/07/ |sed
-e s/Aug/08/ |sed -e s/Sep/09/ |
sed -e s/Oct/10/ |sed -e s/Nov/11/ |sed -e s/Dec/12/ |
sort -r -k 1.7 |sort -r -k 1.4  | sort -r >>temp1
more -P -s temp1
echo -n Najnovejsa je:
more temp1 | head -1
echo --
rm temp1
```

5. Napišite program v lupini, ki v trenutnem direktoriju preimenuje datoteke s podano končnico v drugo končnico. Končnici sta podani v ukazni vrstici

```
#!/bin/bash
if test $# != 2
then
  echo "Uporaba : naloga7 koncnica1 koncnica2"
fi
for datoteka in `ls *$1 2>/dev/null`
do
  mv $datoteka `basename $datoteka $1`$2
done
```

6. Napišite program v lupini, ki vsake pet minut preveri, če se na sistemu pojavi nek proces in ga poskusa ubiti. Ime procesa podamo v ukazni vrstici

```
# uporaba : ubij ime_procesa
while [1]
do
sleep 300
if [`ps -ef | grep $1 | cut -c1-4`]
   killall $1
   fi
done
```

7. Napišite program v lupini, ki beleži dnevnik dosegljivosti računalnikov. Kot argument mu podamo datoteko, ki vsebuje spisek računalnikov, program pa naj vsakih pet minut preveri, če so dosegljivi. V neko datoteko (dnevnik) naj zapisuje, če kateri od računalnikov ni dosegljiv, njegovo ime in čas, ko ni bil dosegljiv.

```
while true
do
for i in `cat spisek.txt`
  do
    ping -c 2 $i &>output|| echo "$i ni dosegljiv ob `date +
%T`">>logfile
 done
 sleep 3
done
```

7.c Napišite preprost program DIR v C-ju, ki izpiše seznam datotek v trenutnem direktoriju

```
#include<stdio.h>
#include<sys/types.h>
#include<dirent.h>
#include<sys/stat.h>
struct stat sbuf;
struct dirent *directory;
int main (int argc, char *argv[]){
  DIR *dp;
  dp=opendir(".");
  while((directory=readdir(dp))!=NULL) {
    fprintf(stdout,"%s\n", directory -> d_name);
  }
}
```

8.c V C-ju napišite preprost program EXEC, ki požene ukaz, ki ga navedemo kot parameter ukazne vrstice.

```
#include <unistd.h> // potrebno za exec pod UNIXom
#include <stdio.h>
main(int argc, char *argv[]){
  char str[20];
  char x;
  strcpy(argv[1],str);
  strcat(str,"&");
  printf("\n%s",str);
  execl("/usr/local/bin/bash","sh", "-c", str);}
```

9.c V C-ju napišite preprost program DIREXE, ki izpiše vse datoteke v trenutnem direktoriju, katere lahko uporabnik zažene.

```
#include<stdio.h>
#include<sys/types.h>
#include<dirent.h>
#include<sys/stat.h>
#include<sys/dir.h>
   struct stat sbuf;
   struct dirent *directory;
   int main (int argc, char *argv[]){
    DIR *dp;
    dp=opendir(".");
    while((directory=readdir(dp))!=NULL) {
     stat(directory->d_name, &sbuf);
     if((sbuf.st_mode &S_IFMT) == S_IFREG)
fprintf(stdout,"%s\n",directory>d_name);
   }
 }
```

V programskem jeziku C napišite program, ki prešteje posamezne črke v tekstovni datoteki,katere ime podamo v ukazni vrstici. Program naj za vsako črko abecede izpiše na zaslon,kolikokrat se ta črka pojavi v dani datoteki. Pri tem ne razlikujemo velikih in malih črk.

```
#include <stdio.h>
#define MAXCRK 32
void main(int argc, char *argv[]) {
  FILE *fd;
  char c; int i;
  int crke[MAXCRK];
  if (argc > 1) {
    for (i=0; i<MAXCRK; i++)
      crke[i] = 0;
    fd = fopen(argv[1], "rt");
    while ( !feof(fd) ) {
      c = fgetc(fd);
      if ( (c>='A') && (c<='Z') )
        crke[c-'A']++;
      else if ( (c>='a') && (c<='z') )
      crke[c-'a']++;
    } //while
    for(c='a'; c<='z'; c++)
    printf("%c se pojavi %d-krat\n", c, crke[c-'a']);
  }
else
  printf("NAPAKA: Manjka ime datoteke.\n");
}
```

12C Kopiranje ene datoteke, znak po znak, v drugo:

```
#include < stdio.h
  /* Kopiranje datoteke */
  main() {
    FILE *fp1,  *fp2;
    int ch;
    if ((fp1= fopen("inpFile", "r")) ==NULL) {
      fprintf(stderr,"Datoteke ne morem odpreti");
      exit (1); }
    fp2 = fopen ("outFile", "w");
    while ( (ch = getc(fp1)) != EOF)
      putc(ch, fp2);
    fclose( fp1); fclose (fp2);
    exit (0);
}
```

Normiraj polje

```
import java.io.*;

public class NormirajPolje {

  static void normirajPolje(double p[], int n) {
  /* funkcija normira polje p z n realnimi stevili */
    double max;
    int i;
    /* najprej poiscemo maksimalni element */
    max = p[0];
    for (i=0; i<n; i++)
      if (p[i]> max) max = p[i];
    for (i=0; i<n; i++) p[i] = p[i]/max;
  }

  public static void main(String args[]) {

  double polje[] = {10.0, 20.0, 22.0, 15.0, 30.0 };
  int num;
  int i;
  num = 5;  /* stevilo elementov v polju */
  System.out.print("Pred normiranjem:");
  for(i=0;i<num;i++) System.out.print(polje[i] + " ");
  normirajPolje(polje, num);
  System.out.print("\nPo normiranju:   ");
  for(i=0;i<num;i++) System.out.print(polje[i] + " ");
  }
}
```

Inverzija vrstic

```
import java.io.*;
public class Inverzija {
  public static void main(String args[]) throws Exception {
    int c, i = -1;
    FileInputStream fin;
    char line[] = new char[80];
  try {
    fin=new FileInputStream("Inverzija.java");
      while ((c=fin.read())!=-1) { // beremo do konca
datoteke
      if(c != '\n') line[++i] = (char) c;
      else {
      while(i>=0) System.out.print(line[i--]);
      System.out.println();
      }
   }
   }
    catch (FileNotFoundException e) {
    System.out.println("Ne morem odpreti datoteke " +
e.getMessage());
    }
  }
}
```

Kalkulator

```
import java.io.*;
public class Calc {
  public static void main(String[] args) throws Exception {
  double a,b,c =0 ;
  char znak;
  if(args.length<3) System.out.println ("premalo podatkov");
  else {
    a = new Double(args[0]).doubleValue(); /* konverzija
obeh podatkov */
    b = new Double(args[2]).doubleValue();
    znak = args[1].charAt(0); /* tu je znak operacije */
    switch (znak) {
      case '+': c = a + b; break;
      case '-': c = a - b; break;
      case '*': c = a * b;
    }
    System.out.println( a + " " + znak + " " + b + " = " +
c);
   } // else
 }
}

import java.io.*;
  class Test {
    public static void main (String argv[]) {
      DataOutputStream dout;
      try {
         // pišemo v datoteko preko vmesnega pomnilnika
      dout=new DataOutputStream(new BufferedOutputStream(new
FileOutputStream("pisanje.dat")));
      for (int i=0;i<=100;i++)
        dout.writeFloat((float)i);      // pišemo realna
(binarni zapis) stevila
        dout.close();
       }
      catch (Exception e) {
     System.out.println("Napaka!!!");
      }
   }
 }
```

Stevilo enic

```
import java.io.*;

public class SteviloEnic {
  static int steviloEnic(int i) {
  /* funkcija vrne stevilo setiranih bitov */
  int j, num;
  num = 0;
  for(j=0;j<32; j++){
    if(( i % 2) ==1 ) num++;
    i = i /2;
  }
  return num;
}

public static void main(String[] args) throws Exception {
/* preizkus funkcije steviloEnic */
  int i, k;
  for (i=0; i<16; i++) {
    k = steviloEnic (i);
  System.out.println ("Stevilo setiranih bitov v " + i + "
je " + k + " \n");
  }
 }
}
```