"BASH" LUPINA

VAJE

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 > acessSorted
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/Okt/10/ |
sed –e s/Nov/11/ | sed –e s/Dec/12/ | 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.

```
if test $# !=2
then
echo "Uporaba: PROG 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.

```
while true
do
ps > bla
grep $1 bla| cut -c 1-7 > bla2
for i in `cat bla2`
  do
    kill $i
  done
rm bla2
rm bla
sleep 300
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 "racunalnik $i ni dosegljiv ob `date +%T`">>logfile
  done
  sleep 300
done
```

"BASH" LUPINA

NALOGE IZ PREJŠNJIH IZPITOV

1. Napišite program *izpisi [dir]* v lupini *bash*, ki za vsako datoteko v direktoriju *[dir]* ter rekurzivno v vseh poddirektorijih interaktivno vpraša ali naj to datoteko prekopira v domači direktorij uporabnika (~). Če je odgovor da, (Y) potem to datoteko tja prekopira, sicer pa ne.

```
[$# -eq0]&& set--
for i in $1/*
do
        if[-d $i];then
                izpis $i
        fi
        if [-f $i];then
                echo -e "Kopiram? [dn]\c"
                vprasaj && cp $i ~/
        fi
done
```

PROGRAMSKI JEZIK C

NALOGE IZ PREJŠNJIH IZPITOV

1. V jeziku C napišite program grep, ki omogoča iskanje niza znakov textovni datoteki. Program naj na standardni izhod izpiše vse tiste vrstice datoteke, ki vsebujejo iskani niz. Niz znakov in ime datoteke naj program sprejme kot argument ukazne vrstice. Za delo z datoteko smete uporabljati le sistemske klice. Sistemski klici so razloženi v prilogi.

```
#include <stdio.h>
#include <string.h>

int main(int argc, char **argv)
{
  FILE *fp;
  char line[255];

  if ( (fp=fopen(argv[2], "r")) == NULL) { return(1); }

  while ( fgets(line, sizeof(line), fp) != NULL)
     if (strstr(line, argv[1])) printf("%s", line);

  fclose(fp);
  return 0;
}
```

2. V jeziku c napišite program append, ki omogoča dodajanje vsebine dane datoteke na konec druge že obstoječe datoteke. Ime izvorne in ime ciljne datoteke naj program sprejme kot argument ukazne vrstice.

```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>

int main(int argc, char **argv)
{
  int in, out;
  char buf[1024];
  int n;

  if ( (in = open(argv[1], O_RDONLY)) <= 0) return 1;
  if ( (out = open(argv[2], O_WRONLY)) <= 0) return 2;

  lseek(out, 0, SEEK_END);

  while ( (n=read(in, &buf, sizeof(buf))) > 0 )
    write(out, buf, n);

  close(in);
  close(out);

}
```

3. V jeziku C napišite program dostopne, ki v trenutnem delovnem direktoriju izpiše vse tiste datoteke, ki jih lahko prebere uporabnik sistema.

```
#include <stdio.h>
#include <dirent.h>
#include <sys/stat.h>

main()
{
  DIR *dir;
  struct dirent *d;
  struct stat st;

  dir = opendir(".");
  while ( (d=readdir(dir)) != NULL )
  {
    stat(d->d_name, &st);
    if (S_ISREG(st.st_mode) && ( (st.st_mode & S_IRUSR) == S_IRUSR) )
        printf("%s\n", d->d_name);
  }
}
```

4. Napišite program dirdir, ki izpiše imena direktorijev v trenutnem delovnem direktoriju. Imena drugih datotek naj program ne izpisuje. Napotki:
Klic funkcije stat(dir,&sbuf), kjer je dir ime datoteke, &sbuf pa naslov strukture tipa stat, napolni to strukturo s podatki o datoteki. Pomembna komponenta te strukture je st_mode. Če vsebino te komponente maskiramo (naredimo logični IN) z masko S_IFMT, dobimo vrednost, ki pove za aktero datoteko gre. Nekaj možnosti je: S_IFREG (datoteka je običajna datoteka), S_IDFIR (datoteka je direktorij), S_IFLNK (simbolični link) ali S_FIFO (sklad FIFO).
Prototip funkcije stat je definiran v <sys/types.h>.

```
#include <sys/dir.h>
#include <stdio.h>
#include <sys/types.h>

main()
{
  char *dir;
  struct stat sbuf;
  DIR *dp;
  Struct direct *directory
  Dp=opendir(“.”);
  While ((directory=readdir(dp)) !=NULL)
  {
    dir=directory -> d_name;
    stat(dir, &sbuf);
    if ((sbuf.st_mode&S_IFMT)==S_IFDIR)
      printf(“%s\n, dir);
  }
  return 0;
}
```

JAVA

NALOGE IZ PREJŠNJIH IZPITOV

1. V jeziku java napišite program, ki na zaslon izpiše naslednji tekst:

FRI
Univerza v LJ

```
import java.io.*;

class druga {
        public static void main(String[] args)
        {
                System.out.println("FRI");
                System.out.println("Univerza v LJ");
        }
}
```

Kaj moraš narediti, da bo ta program preveden in izveden?

```
 $ javac druga.java
 $ java druga
```

2. Razvijte v jeziku java razred ura in iz njega izpeljite razred budilka. Realizirajte javanski program ali applet, ki improvizira tako budilko.(Ta izpiše na zaslon "Cin-Cin", ko poteče določen čas)

```
import java.io.*;
import java.awt.*;

class Ura
{
  int cas;
  public void NastaviUro(int nova) { cas = nova; }
  public int KolikoJeUra() { return cas; }
  public void TikTak() { cas = cas + 1; }
}

class Budilka extends Ura
{
  int kdaj_zvoni;

  void NastaviBudilko(int cas) { kdaj_zvoni=cas; NastaviUro(0); }
  void Pozeni()
  {
    int cas;
    Thread me = Thread.currentThread();

    while (true)
    {
      try { me.sleep(1000); }
      catch (InterruptedException e) {}
      TikTak();
      cas = KolikoJeUra();
      if (cas >= kdaj_zvoni) { System.out.println("Cin-Cin"); }
    }

  }
}

class budilka {
  public static void main(String[] args)
  {
    Budilka b = new Budilka();
    b.NastaviBudilko(10);
    b.Pozeni();
  }
}
```

3. Razvijte v jeziku Java razred LIK z lastnostima obseg in povrsina. Iz tega razreda izpeljite razred PRAVOKOTNIK, ki ima dodatni lastnosti a in b (stranici pravokotnika). Izpeljani razred ima še metodi getPovrsina() in getObseg(). Sestavite javansko aplikacijo ali applet, ki uporabi te razrede in v njem imejte dva primerka (instance) pravokotnikov. Za oba primerka pravokotnikov izpišite njun obseg.

```
import java.io.*;

class LIK {
  int obseg;
  int povrsina;
}

class PRAVOKOTNIK extends LIK {
  int a,b;
  int GetPovrsina() { return a*b; }
  int GetObseg() { return 2*(a+b); }
}

class sesta {
  public static void main(String[] args)
  {
    PRAVOKOTNIK p1 = new PRAVOKOTNIK();
    PRAVOKOTNIK p2 = new PRAVOKOTNIK();

    p1.a=10; p1.b=10;
    p2.a=15; p2.b=5;
    System.out.println(p1.GetObseg());
    System.out.println(p2.GetObseg());

  }
}
```