**1.14.04.2015 прва**

Да се напише Shell скрипта која што најпрво ќе ги најде датотеките naredbi.txt“ и „vlez.txt“ во системот и ќе ги ископира во тековниот директориум. Потоа, ќе ја повика perl скриптата и ќе ги изврши една по една наредбите што се враќаат како одговор од PERL скриптата. Резултатот од извршување на секоја една од наредбите се печати во излезна датотека „izlez.txt“, во која што преку прилепување (append) го додава излезот на секоја една од наредбите.

БОНУС: Потоа, скриптата креира датотека „naredbi1.txt“ во која ги сместува последните 10 наредби од директориумот „/usr/bin“. Потоа, ја креира датотеката „vlez1.txt“ каде што ги има сите 10 наредби од „naredbi1.txt“ со наставка „--help“. На крај, ја повикува perl скриптата праќајќи ги „naredbi1.txt“ и „vlez1.txt“ како аргументи и ги извршува секоја една од наредбите што се враќаат како резултат. Резултатот од извршувањето се прикажува на екран.

```
#!/bin/bash
prva=$(find / type -f -name "naredbi.txt")
vtora=$(find / type-f -name "vlez.txt")

if[ -n $prva ]
then
      cp . $prva
fi

if[ -n $vtora ]
then
      cp . $vtora
fi

prethodna_output=$(prethodna.py)
echo "$prethodna_output" >> izlez.txt

//poslednite 10 naredbi od /usr/bin se smestuvaat vo naredbi1.txt so
nastavka --help
find /usr/bin -type f | tail -n 10 > naredbi1.txt
sed -e 's/$/ --help/' naredbi1.txt > vlez1.txt

prethodna_output_bonus=$(prethodna.py naredbi1.txt vlez1.txt)
echo "$prethodna_output_bonus"
```

**2.14.04.2015 втора**

Да се напише Shell скрипта која што најпрво ќе ги најде датотеките naredbi.txt и vlez.txt во системот и ќе ги избрише. Потоа, во тековниот директориум ќе креира празни датотеки naredbi.txt и vlez.txt и ќе ја повика perl скриптата и ќе ги изврши една по една од наредбите што се враќаат како одговор од PERL скриптата. Резултатот од извршување на секоја една од наредбите се печати во излезна датотека izlez.txt која што преку прилепување (append) го додава излезот на секоја една од наредбите.

БОНУС: Потоа, скриптата креира датотека „naredbi1.txt“ во која ги сместува последните 10 наредби од директориумот „/usr/bin“. Потоа, ја креира

датотеката „vlez1.txt“ каде што ги има сите 10 наредби од „naredbi1.txt“ со наставка „--help“. На крај, ја повикува perl скриптата праќајќи ги „naredbi1.txt“ и „vlez1.txt“ како аргументи и ги извршува секоја една од наредбите што се враќаат како резултат. Резултатот од извршувањето се прикажува на екран.

```
#!/bin/bash
prva=$(find / -name "naredbi.txt")
vtora=$(find / -name "vlez.txt")

if[ -n $prva ]
then
      rm "naredbi.txt"
fi

if[ -n $vtora ]
then
      rm "vlez.txt"
fi

touch naredbi.txt
touch vlez.txt

# Povikaj ja prethodnata i izvrsi ja sekoja naredba
prethodna_output=$(prethodna.py)

#pecatnje na rezultatite od izvrsuvanjeto na naredbite vo izlez.txt
echo "$prethodna_output" > "izlez.txt"

#bonus
find /usr/bin -type f | tail -n 10 > naredbi1.txt

#dodadi --help na sekoja naredba
sed -e 's/$/ --help/' naredbi1.txt > vlez1.txt
prethodna_output_bonus=$(prethodna.py naredbi1.txt vlez1.txt)
echo "$prethodna_output_bonus"
```

**3.30.03.2016 прва**

Да се напише Shell скрипта која што ќе ги најде сите SHELL скрипти во тековниот директориум и неговите под директориуми. Секоја SHELL скрипта започнува со линијата „#!/bin/bash“. Потоа, главната SHELL скрипта треба да направи статистика за сите SHELL скрипти кои што ги има пронајдено и во излезна датотека „stat_shell.txt“ да ги запише статистиките во формат: <ime_skripta> <pateka> <broj_promenlivi> <broj_granenja> <kompleksnost> Потоа, да ги изврши една по една пронајдените shell скрипти (без аргументи) и излезот да го запише во „izlez.txt“ датотека.

```
#!/bin/bash

shell_scripts=$(grep -rl '^#!/bin/bash' . | grep -E '\.sh$')
```

```
stat_file="stat_shell.txt"
if[ ! -e "$stat_file" ]
then
      touch "$stat_file"
fi

output_file="izlez.txt"
if[ ! -e "$output_file" ]
then
      touch "$output_file"
fi

for script in $shell_scripts
do
      script_name=$(basename "$script")
      script_path=$(realpath "$script")
      num_variables=$(grep -cE '^[[:space:]]*[a-zA-Z_][a-zA-Z0-9_]*='
"$script")
      num_branches=$(grep -cE '^[[:space:]]*(if|elif|else|for|while|case)'
"$script")

      complexity=$((num_variables + num_branches))

      echo "$script_name $script_path $num_variables $num_branches
$complexity" >> "$stat_file"

      bash "$script" >> "$output_file"
done
```

**4.30.03.2016 втора**

Да се напише Shell скрипта која што ќе ги најде сите python скрипти во тековниот директориум и неговите под директориуми. Секоја PERL скрипта започнува со линијата „#!/usr/bin/perl“ или „#!/usr/bin/perl -w“. Потоа, SHELL скриптата треба да направи статистика за сите PERL скрипти кои што ги има пронајдено и во излезна датотека „stat_perl.txt“ да ги запише статистиките во формат: <ime_skripta> <pateka> <broj_skalarni> <broj_polinja> <broj_HASH> <broj_datoteki>
Потоа, да ги изврши една по една пронајдените perl скрипти (без аргументи) и излезот да го запише во „izlez.txt“ датотека.

**5.06.04.2017**

Да се напише SHELL скрипта која ги листа сите датотеки од тековниот директориум. Потоа, зема две по две датотеки од листата датотеки и ги праќа како влезни аргументи на повикот на PERL скриптата од задача 1. Резултатот од секој повик на PERL Скриптата се запишува во датотека креирана од спој на имињата на двете датотеки кои се праќаат на PERL Скриптата, притоа се запишува во продолжение (append).

```
#!/bin/bash

datoteki=`ls`#listanje site datoteki od tekoven direktorium

for datoteka1 in datoteki
do
      if [ -f "$datoteka1]
      then
            for datoteka2 in datoteki
            do
                  if [ -f "$datoteka2 ]
                  then
                        rezultat=$(python3 main.py $datoteka1 $datoteka2)
                        novaIme="${datoteka1}${datoteka}
                        $rezultat >> $novaIme
                  fi
            done
      fi
done
```

**6.10.04.2019**

Да се напише SHELL скрипта која ги листа сите датотеки од тековниот директориум. Потоа, имињата на оние датотеки кои што имаат помалку од 1000 зборови ги праќа како аргументи од командна линија на повик од PERL скриптата (сите имиња ги праќа во еден повик). За оние датотеки кои што имаат повеќе од 1000 зборови, креира посебен повик и нивните имиња ги праќа како еден аргумент од командна линија.

```
# SITE IMINJA GI PRAKJA NA EDEN POVIK
znaci treba kako konkatenacija da napravime

# IDEJATA E SO STRINGOT PRAZEN i na nego dodavame


#!/bin/bash

datoteki=$(ls)

site_datoteki_validni=""
site_datoteki_nevalidni=""

for datoteka in datoteki
do
      if [ -f "$datoteka" ]
      then
            echo "Ova e datoteka"
            sodrzina=$(cat $datoteka)
            brZborovi=$(echo $sodrzina | wc -w)
            if [ "$brZborovi" -lt "1000" ]
            then
```

```
                  site_datoteki_validni="$site_datoteki_validni $datoteka
" #ovde ima prazno mesto
            else
                  site_datoteki_nevalidni="$site_datoteki_nevalidni
$datoteka  "
            fi
      else
            echo "Ne e datoteka"
      fi
done

python3 main.py $site_datoteki_validni
python3 main.py $site_datoteki_nevalidni
```

**7.19.11.2019**

Да се напише SHELL скрипта која како аргумент од командна линија добива име на влезна датотека. Доколку не се прати аргумент на командна линија, тогаш се пребарува датотека со име „naredbi.txt“ во тековниот директориум и неговите поддиректориуми и доколку се најде, тогаш се копира таа датотека во тековниот директориум (доколку истата не е во тековниот директориум). Потоа, скриптата пребарува во тековниот директориум и неговите поддиректориуми датотека со име „actions.txt“. Доколку најде, ја повикува PERL скриптата од првата задача, каде како прв аргумент се праќа датотеката „naredbi.txt“, а како втор аргумент се праќа датотеката „actions.txt“, притоа втората датотека се праќа со апсолутната патека каде ќе биде пронајдена. Секоја една наредба што ќе се врати од извршувањето на PERL скриптата се извршува од страна на SHELL скриптата. Доколку наредбата врати некаков текст од извршувањето, тогаш се печати името на наредбата и грешка во продолжение. Доколку не врати текст, тогаш се печати името на наредбата и текстот „OK“ во продолжение.

```
#!/bin/bash

if [ "$#" -eq "0" ]
then
      prva_datoteka="naredbi.txt."
else
      prva_datoteka=$1
fi

prva_pateka=$(find . -name $prva_datoteka)
if [ -n "$prva_pateka" ]
then
      echo "Datotekata ja ima vo tekovniot direktorium ili vo negovite
poddirektoriumi"
      vo_tekoven=$(ls | grep "$prva_datoteka")
      if [ -n "$vo_tekoven" ]
      then
            echo "Datotekata ja ima vo tekovniot direktorium"
      else
            echo "Datotekata ja kopirame vo tekovniot direktorium"
```

```
            cp $prva_datoteka .
      fi



      vtora_datoteka="actions.txt"
      vtora_pateka=$(find . -name $vtora_datoteka)
      if [ -n "$vtora_pateka" ]
      then
            echo "Datotekata ja ima vo tekovniot direktorium ili vo
negovite poddirektoriumi"
            vtora_apspateka=$(find / -name $vtora_datoteka)
            rezultat=$(python3 main.py $prva_datoteka $vtora_apspateka)

            # se vrakjaat naredbi zatoa mora vo for ciklus NE E SAMO EDNA
NAREDBA
            for naredba in rezultat
            do
                  sodrzina_naredba=$(cat $naredba)
                  if [ -n "$sodrzina_naredba" ]
                  then
                        echo "$naredba ERROR"
                  else
                        echo "$naredba OK"
                  fi
            done
      else
            echo "Datotekata $vtora_datoteka ne postoi"
      fi

else
      echo "Datoteka so ime $prva_datoteka ne postoi"
fi
```

**8.31.01.2019**

Да се напише Shell скрипта која што ги наоѓа сите именици во тековниот директориум т.е. ги бара сите датотеки кои што завршуваат на екстензија .vcf. Доколку има повеќе именици, тогаш на крајот во првиот именик од листата именици се додава содржината на другите именици и се повикува PERL скриптата со името на првиот именик.

```
# datoteki so ekstenzija .vcf
# kike.vcf
  david.vcf


# NE ZABORAVAJ DA BROIS LINII - NE TRCAJ!!! wc -l !!!
# PRVA DATOTEKA? -> cat $datoteki | head -n 1
```

```
#!/bin/bash

datoteki=$(ls | grep -o ".*\.vcf")
kolku=$(cat $datoteki | wc -l)

if [ "$kolku" -gt "1" ]
then
      echo "Ima povekje od edna datoteka"
      prva=$(cat $datoteki | head -n 1)

      # pocnuvam od vtorata datoteka
      tmp=2
      while [ "$tmp" -le "$kolku" ]
      do
            # ja zemam datotekata!
            ime_datoteka=$(cat $datoteki | head -n $tmp)
            sodrzia_datoteka=$(cat $ime_datoteka)
            $sodrzina_datoteka >> $prva

            tmp=$(expr $tmp + 1)
      done

      python3 main.py $prva
fi
```

**9.Задачи прва**

Да се напише Shell скрипта која што најпрво ќе ги најде датотеките „naredbi.txt“ и „vlez.txt“ во системот и ќе ги ископира во тековниот директориум. Потоа, ќе ја повика perl скриптата и ќе ги изврши една по една наредбите што се враќаат како одговор од PERL скриптата. Резултатот од извршување на секоја една од наредбите се печати во излезна датотека „izlez.txt“, во која што преку прилепување (append) го додава излезот на секоја една од наредбите.

```
#!/bin/bash

naredbi_pateka=$(find / -name naredbi.txt)
vlez_pateka=$(find / -name vlez.txt)

if [ -n "$naredbi_pateka" ]
then
      echo "Ne postoi vo sistemot!"
fi

if [ -n "$vlez_pateka" ]
then
      echo "e postoi vo sistemot!"
fi
```

```
jaImaTekoven1=$(ls | grep "naredbi.txt")
if [ -z "$jaImaTekoven1" ]
then
      cp naredbi.txt .
else
      echo "naredbi.txt ja ima vo tekovniot direktorium"
fi

jaImaTekoven2=$(ls | grep "vlez.txt")
if [ -z "$jaImaTekoven2" ]
then
      cp naredbi.txt .
else
      echo "vlez.txt ja ima vo tekovniot direktorium"
fi



rezultat=$(python3 main.py naredbi.txt vlez.txt)
for naredba in $rezultat
do
      cat $naredba >> izlez.txt
done
```

**10.Задачи втора**

Да се напише SHELL скрипта која како аргумент добива листа од имиња од датотеки. Скриптата, за секоја датотека (име од датотека) да провери дали ја има како датотека во тековниот директориум, или под-директориуми. Доколку ја има, проверува дали датотеката (во нејзината содржина) има барем еден валиден датум (DD-MM-YYYY), и доколку нема, ја игнорира. Доколку има валиден датум, тогаш ја повикува PERL скриптата и како прв аргумент го праќа името на датотеката, и т.н. со сите датотеки што се пратени како аргументи од командна линија. За секоја датотека што ќе биде пратена на PERL скриптата, се печати на екран нејзиното име и колку различни датуми се пронајдени.

```
# SO grep -o kje mi gi vrati site zborovi koi go ispolnile uslovot a ne
cela redica
# zatoa mozam da iskoristam uniq -u za toa kolku bile unikatni i wc -w za
da gi izbrojam


#!/bin/bash

for datoteka in $@
do
      ja_ima=$(find . -name $datoteka)
      if [ -n "$ja_ima" ]
      then
            echo "Datotekata ja ima vo tekovniot direktorium"
            sodrzina=$(cat $datoteka)
```

```
            validno=$(echo $x | grep -o "[1-31]\-[1-12]\-[0-9][0-9][0-9][0-
9]")
            if [ -n "$validno" ]
            then

                  python3 main.py $datoteka

                  echo "Vo datotekata postoi validen datum"
                  kolku_unikatni=$(echo $validno | uniq -u | wc -w)
                  echo "Ime na datoteka: $datoteka"
                  echo "Ima $kolku_unikatni razlicni datumi vo datotekata"
            else
                  echo "Vo datotekata nema validen datum!"
            fi
      fi
done



# OVA GLEDAJ ZA POMOS

cat -> 12.15.2002 kristina 13.14.2010 11.11.2011
grep -0 -> 12.15.2002 13.14.2010 11.11.2011
uniq -u | wc -w -> br na razlicni datumi


ako go koristime obicno grep togas pretpostavuvame deka kolku sto imame
razlicni linii tolku imame razlicni datumi
Denes e 15.11.2023 i e sreda.
grep -> 15.11.2023 i e sreda.
awk '{print $1}' -> 15.11.2023
uniq -u | wc -l
```

**11.Задачи трета**

Да се напише SHELL скрипта која како аргумент добива бројка X. Скриптата треба да провери дали во тековниот директориум ја има датотеката „files.txt“. Доколку ја има, тогаш продолжува да се извршува скриптата. Во датотеката, во секоја линија одделно, има имиња на датотеки. Скриптата треба да провери дали има најмалку онолку имиња на датотеки колку што е бројката X што се праќа како аргумент на скриптата, и доколку има најмалку онолку имиња на датотеки колку што е X, продолжува да се извршува скриптата, со тоа што исто така проверува дали првите X датотеки постојат во тековниот директориум. Потоа, скриптата ја повикува PERL скриптата од првата задача, со тоа што како прв аргумент на скриптата испраќа вредност X/2, а во продолжение ги испраќа првите X датотеки од „files.txt“. БОНУС: откако ќе заврши PERL скриптата, на екран да се отпечати содржината на излезните датотеки т.е. на вторите X/2 датотеки што се испраќаат на PERL скриптата

```
# BROENJE LINII NA DATOTEKA cat datoteka | wc -l
# zemanje na odredeni linii so head -n broj
```

```
#!/bin/bash

x=$1

ja_ima=$(find . -name files.txt)

if [ -n "$ja_ima" ]
then
      echo "Ja ima datotekata files.txt"

      # BROJ NA LINII, vika sekoja datoteka E VO POSEBEN RED
      br_linii=$(cat files.txt | wc -l)
      if [ "$br_linii" -ge "$x" ]
      then
            # tmp pocnuva od 1 NE od 0
            tmp=1
            while [ $tmp -neq $x ]
            do
                  datoteka=$(cat files.txt | head -n $tmp)
                  datoteka_postoi=$(ls | grep $datoteka)
                  if [ -n "$datoteka_postoi" ]
                  then
                        echo "Datotekata $datoteka postoi vo tekovniot
direktorium"
                  else
                        echo "Datotekata $datoteka NE postoi vo tekovniot
direktorium"
                  fi

                  tmp=$(expr $tmp + 1)
            done
      fi

      # POVIKUVANJE NA PYTHON
      x_datoteki=$(cat files.txt | head -n $x)
      x_pola=$(expr $x / 2)
      python3 main.py $x_pola $x_datoteki
```