

FTP Mirror

Zadani

Program dostane na stdin URL ve tvaru
ftp://[username:password@]host_address/directory[/file_name].

Má za úkol okopírovat tento adresář (soubor) na lokální disk. Soubor nesmí být kopírován pokud už stejný existuje (soubor není stejný, když má jinou délku, nebo když má soubor jiné datum). Při porovnávání datumů je potřeba, vzhledem k problémům s různými formáty času i časovými zónami neporovnávat "remote" čas s lokálním časem souboru, ale je potřeba mít uschované "remote" čas souborů z minulého bihu a porovnávat vůči nim. Soubory, které zmizí na vzdálené straně je potřeba smazat i lokálně. Není-li uvedeno uživatelské jméno a heslo v URL, je potřeba použít anonymní přístup.

Reseni

```
ftp -n > ftpout << END
open u-sl3.ms.mff.cuni.cz
user jkof8661 ahojlidi
cd ftp
dir -R
close
quit
END

touch seznam
touch smazat
touch stahnout
cat ftpout | sed '1d;2d;3d' > templ
cat templ | awk '
BEGIN      { pr=0; smazat=0 }

/'^$'/ && pr==0 { pr=1; print; next }
pr == 1      { pr=0; print " " $0; next }
/total / { smazat=2; next }
smazat > 0 { smazat=smazat-1; next }
smazat==0 { print $1 " " $6 $7 $8 " " $9; next }
' > ftpout

cat ftpout | sed '/^d/d' > templ
cat templ | cut -d" " -f2- > ftpout
cat ftpout | sed 's/^ //;s:/$/\\/' > templ
cp templ zaloha
echo "added" >> templ
cat seznam >> templ
cat templ | awk '
BEGIN      { prazdny=0; path=""; added=0 }

/'^$'/      { prazdny=1; next }
prazdny==1 && added==0 { prazdny=0; path=$0; print path >> "adresare"; next }
}
prazdny==1   { prazdny=0; path=$0; next }
/added/      { added=1; path=""; next }
added==0     { soubory[path $2]=$1; next }
added==1     { if (soubory[path $2]==$1) {soubory[path $2]="nestahovat" }
```

```
else if (soubory[path $2]=="") { print path $2 >> "smazat"
}
}
END      { for (x in soubory) {if (soubory[x]!="nestahovat" &&
soubory[x]!="") { print "get " x >> "stahnout" }
}
},

if [ ! "`wc -l < smazat`" -eq 0 ]; then rm `cat smazat`; fi
echo "open u-sl3.ms.mff.cuni.cz" > ftpin
echo "user jkof8661 ahojlidi" >> ftpin
echo "cd ftp" >> ftpin
cat adresare | while read x; do if [ ! -d $x ]; then mkdir $x; fi; done
cat stahnout >> ftpin
echo "close" >> ftpin
echo "quit" >> ftpin
cat ftpin | ftp -n > ftpout

rm smazat stahnout templ ftpout seznam ftpin adresare
mv zaloha seznam
```

Aligator

Zadani

Mas soubor db - jako databaze. Ten ma "vhodny format" obsahujici polozky:

```
login
cele jmeno bez diakritiky ve tvaru jmeno mezera prijmeni
[ adresa ]
[ klic ]
```

Teda maximalne 4 polozky pricemz posledni dve jsou nepovinne. Jeste se k nim vratim.

Dale existuje soubor /etc/aliases a obsahuje radky bud:

```
jmeno.prijmeni:adresa,adresa,adresa,....
nebo
alias:adresa
```

Tebe zajimaji jenom radku jmeno.rijmeni. Druhe jsou tam jenom pro paradu, aby to mel Forst cim zpestřit.

Tedy soubor db byl textovy soubor v libovolnem vhodnem formatu, jak uz jsem psal. Forst primo rekl ze nejidealnejsi by byl format:
login:full_name:[adresa]:[klic]

-Ty dve polozky jsou nepovinne. Kazdy student ma takovyto zaznam v databazi. A soubor aliases obsahuje veskere studenty kteri zadali o udeleni emailove adresy tvaru jmeno.prijmeni@...

Zamerne pisu ze soubor /etc/aliases muze mit u jednoho jmena vice adres. To z toho duvodu ze muze byt vic nez jeden clovek se stejnym jmenem. Treba Josefu Novaku bude pravdepodobne vic nez jeden. Pak bude na tomto radku taky vic nez jedna adresa.

No to bylo jen tak na uvod.

A ted konecne ten script.

Mas napsat script který na vstupu dostane mail.

Mail je ve tvaru: hlavicka oddelena od tela jednou prazdnou radkou. Tak jak to znamo.

Dal muzes predpokladat ze se tam vyskutejuje jednoznacne radek s nazvem:

From:adresa

Muzes predpokladat ze za From: je primo adresa. Zadne znaky navic nebo tak neco... značne zjednoduseni.

Dal se tam muze vyskytovat radek se tvarem:

Subject: text

kde text muze mit tvar: ALIGATOR login adresa [klic]

Ale taky nemusí.

Pokud nenajdes takovyto tvar subjectu, pak musis najít úplně stejný tvar na prvním radku v části těla mailu...

To vlastně definuje příkaz pro databázi.

Tvým úkolem je aby script zpracoval přicházející mail následovně:

1) Dostane-li mail se subjectem jaký jsem popsal bez klice tak se podívá do souboru db. Pokud tam není uvedena adresa u příslušného loginu (ten máš v subjectu), tak tam tuto adresu (která je za slovy ALIGATOR login) vloží a zároveň upraví /etc/aliases tak aby příslušný radek s fullname obsahoval danou adresu. Pokud takový radek neexistuje tak ho vygeneruje. Posle mail na tuto adresu ze zaregistrování proběhlo úspěšně.

2) Dostane-li mail se subjectem jaký jsem popsal bez klice tak se podívá do

souboru db. Pokud tam už je uvedena adresa znamená to že uživatel ji chce změnit. Ty tedy musíš vygenerovat "náhodný retezec" který zapíšeš do db na místo klice a taky tento náhodný retezec pošleš uživateli na starou adresu (tu máš stále v db) - náhodný retezec ???

3) Dostane-li mail se subjectem jaký jsem popsal s klicem tak se podívá do db jestli tento klic s ním odpovídá. Pokud ano je vše v pořádku a ty můžeš změnit adresu příslušného uživatele v /etc/aliases. Nezapomeň že na jednom řádku může být více než jedna adresa uživateli a ty musíš vybrat tu správnou a změnit ji. Zároveň musíš smazat klic v souboru db a změnit adresu starou za novou.

Taky jsi musel upozornit uživatele že vše je OK.

4) Pokud nastane jakýkoliv jiný případ, třeba že klic který ti pošle uživatel na zpět nebo třeba že ti pošle mail bez klice a ty máš nějaký klic zapsaný v souboru db (pak tento klic musíš také smazat) a další jiné chyby způsobí že pošleš mail uživateli o neúspěšnosti akce.

Toto všechno jsem popisoval pouze pro případ že se takto definovaný retezec (ALIGATOR login adresa [klic]) objevuje v subjectu. Pokud ale subject nenajdeš nebo najdeš jiný retezec tak se podíváš na první radek těla mailu. A tam se musí vyskytovat úplně stejný tvar. Pokud ne je mail neplatný a zahodíš ho.

Jsi také dále povinen kontrolovat ty chyby.

1. uživatel dostane mail s unikátním retezcem který jsi ty náhodně vygeneroval a uživatel zadá pouze Reply. Co se ale stane. V subjectu se objeví najednou zname "Re". A to musíš preventivně vždy nějakým způsobem odstranit. Dal když odpovíš tak se na každém řádku těla mailu objevuje retezec ">" a s tím udeleš to samé.

2) Musíš zajistit aby do databáze neměli přístup najednou dvě instance scriptu. Dovedeš si představit kdyby najednou do db nebo /etc/aliases najednou zapisovali dva scripty. Teda to chce nějaký zámek. Nejlepe pomocí nějakého souboru.

3) Na začátku scriptu si musíš definovat proměnou REJECT tvaru:

```
REJECT="root administrator Aligator..."
```

A to ti říká že pokud dostaneš mail od člověka jehož adresa před @ je obsazena v proměnné REJECT tak se nic nevykoná a script skončí.

Bylo by totiž asi blbě kdyby posílal skript mailů Aligatorovi a ten je porad dal posílat sám sobě. Jak říkal Forst: Elegantní nekonečný cyklus.

Reseni

```
#!/bin/sh
```

```
REJECT="root administrator Aligator"
login=""
adresa=""
from=""
```

```
# nacistani hlavicky podle ALIGATORa
```

```
loadheader() {
```

```
read aligator login adresa klic << END_READ
$1
END_READ
```

```
if [ "$login" = '' -o "$adresa" = '' -o "$aligator" != "ALIGATOR" ] ; then
    login=""
    adresa=""
```

```

        klic=""
fi
}

generuj_klic() {
    klic="$${!}"
}

#zamykac souboru - tohle nevim, jak se dela :- (
lock_files() {
    echo 'Locking files'
}

#odemykac souboru
unlock_files() {
    echo 'Unlockong files'
}

# cteni hlavicky mailu

while read line ; do
    case "$line" in
        From:* )
            from=`echo "$line" | sed "s/^From:/"`
            ;;
        Subject: )
            subj=`echo "$line" | sed "s/^Subject:/"`
            subj=`echo "$subj" | sed "s/^Re:/"`
            loadheader "$subj"
            ;;
        "" )
            break
            ;;
    esac
done

# test na REJECT
name=`echo "$from" + cut -d\@ -f1`
if echo "$REJECT" | grep "$name" > /dev/null ; then
    exit 0
fi

# nepodarilo se ze subjectu, zkusime z tela
if [ "$login" = "" ] ; then
    read line
    line=`echo "$line" | sed "s/^>/"`
    loadheader "$line"
fi

# zrejme neplatny mail
if [ "$login" = "" ] ; then
    exit 0
fi

lock_files

if [ "$klic" = "" ] ; then
else
fi

```

```

# nacteni dat o nasem uzivateli
line=`grep "^$login" db `
dbname=''
dbadresa=''
dbklic=''
exist=0

# tohle by melo VZDY probehnout, protoze uzivatele v databazi byt uz
musi!!!
if $? ; then
    IFS='.'
    read nic dbname dbadresa dbklic << END_READ

$line
END_READ
    exist=1
else
    echo "UNKNOWN USER"
    exit 0
fi

dotname=`echo "$dbname" | tr "[]" "[:]"`

# test na spravnost
if [ "$klic" = "" ] ; then
    ##### 1. pripad
    if [ "$dbadresa" = "" ] ; then
        ed db << ED_END
g/^$login/ s/^.*$/{login}:${dbname}:${adresa}:/
wq
ED_END
        if grep "$dotname" aliases > /dev/null ; then
            ed aliases << ED_END
g/^$dotname:/ s/^\\(.*)$/{adresa},/
wq
ED_END
        else
            echo "${dotname}:${adresa}," >> aliases
        fi
        echo 'operation sucessfull - 1. pripad'
        # | mail -s "Aligator" "$adresa"
    ##### 2. pripad
    else
        generuj_klic
        ed db << ED_END
g/^${login}/ s/^.*$/{login}:${dbname}:${dbadresa}:${klic}/
ED_END
        echo "ALIGATOR $login $adresa $klic"
    else
    ##### 3. pripad
        if [ "$klic" = "$dbklic" ] ; then
            ed aliases << ED_END
g/^$dotname:/ s/$dbadresa/$adresa/
wq
ED_END
            echo "3. cast - VSE OK"
        else
            echo "NESCHODA KLICU!!!"
        fi
    fi

unlock_fines

```

Find

Zadani

Co by to melo umet - parametry :

find c1 c2 c3 ... cn

c1 - cn jsou cesty kde zacina hledat, prohledava rekurzivne. + jeste v tom mohou byt tyhle optiony :

-name blabla , kde blabla je neco wildcardovyho...

-newer file - soubory mladsi nez file

-type [dfl] - typ (jen jeden naraz :o) (adresar, soubor, symbolic link)

-user xxx - kde xx je bud jeho jmeno, nebo UID

+ jeste akce, co to ma provest s nalezenym souborem...

-print - jen ho to vypise

-ls - da to na nej ls -l

-exec cosi cosi ... cosi ; spusti cosi cosi .. cosi - proste vsechno mezi -exec a strednikem.

defaultne to provadi print, vzdy maximalne jednu akci.

1. jeste byl jeden docela dulezity parametr -depth, který mel hodnotu bud n, -n nebo +n (kde n je cele cislo) a script pak prohledaval soubory jenom v dane hloubce (adresarove), do dane hloubky nebo od dane hloubky.

2. v -exec cosi cosi cosi se jeste (i kdyz to je zrejme) obcas mohlo vyskytnout {} a za to se pak dosadilo jmeno souboru ..

Reseni

```
#!/bin/sh
```

```
# nastaveni defaultu
name=""
newer="0000-00-00 00:00:00"
type=""
user=".*"
action="-print"
execstr=""
dfrom="0"
dto="256"
depth="0"
```

```
## prohledava podadresar - s celou cestou, který dostane jako 1. a jediny
parametr
searchdir() {
    depth=`expr "$depth" + 1`
    tmp="/tmp/find.$$.$depth"
    rm -f "$tmp"
    touch "$tmp"

    cd "$1"
    ls -l -d --full-time $name | {
        while read prava l lsuser group size dat1 dat2 dat3 jmeno ; do
            #echo "$prava $type.* $lsuser $user"
            expr match "$prava" "$type.*" > /dev/null
            rexpr="$?"
            filetime="$dat1 $dat2"
            novejsi=`echo "$filetime
$newer" | sort -r | head -1`
            expr match "$lsuser" "$user" > /dev/null
            ruser="$?"
            #
            echo "$rexpr" -eq "0" -a "$novejsi" -ne "$newer" -a "$ruser"
            -eq "0" -a "$dfrom" -le "$depth" -a "$dto" -ge "$depth"
            if [ "$rexpr" -eq "0" -a "$novejsi" != "$newer" -a "$ruser" -
            eq "0" -a "$dfrom" -le "$depth" -a "$dto" -ge "$depth" ] ; then
                case "$action" in
                    "-print" )
                        echo "$1/$jmeno"
                        ;;
                    "-ls" )
                        ls -l -d "$1/$jmeno"
                        ;;
                    "-exec" )
                        estr=`echo "$execstr" | sed "s/{}/$jmeno/g"`
                        eval "$estr"
                        ;;
                esac
            fi
            # priprava na rekurzi do podadresaru
            if [ -d "$jmeno" ] ; then
                echo "$1/$jmeno" >> "$tmp"
            fi
        done
    }
    while read subdir ; do
        searchdir "$subdir"
    done < "$tmp"
    rm -f "$tmp"
    depth=`expr "$depth" - 1`
}
```

```

# zpracovani parametru
thisdir=`pwd`
while [ "$#" -ne 0 ] ; do
    case "$1" in
        '-name' )
            name="$2"
            shift 2
            ;;
        '-newer' )
            newer=`ls -l -d --full-time "$2" | {
                read prava l lsuser group size dat1 dat2 dat3 jmeno
                echo "$dat1 $dat2"
            }`
            shift 2
            ;;
        '-type' )
            type="$2"
            if [ "$type" = 'f' ] ; then
                $type='- '
            fi
            ;;
        '-user' )
            user="$2"
            tuser=`grep "^[^:]*:[^:]*:$2:" /etc/passwd | sed
's/\([^:]*\):.*$/\1/'`
            if [ "$tuser" != "" ] ; then
                user="$tuser"
            fi
            shift 2
            ;;
        '-print' | '-ls' )
            action="$1"
            shift 1
            ;;
        '-exec' )
            action='-exec'
            shift 1
            while [ "$1" != ';' ] ; do
                execstr="${execstr}$1 "
                shift 1
            done
            shift 1
            ;;
        '-depth' )
            if expr "$2" : '-[0-9]*' > /dev/null ; then
                dto=`echo "$2" | sed 's/^-//`
            else
                dfrom=`echo "$2" | sed 's/^+//`
            fi
            shift 2
            ;;
        * )
            cd "$thisdir"
            cd "$1"
            new=`pwd`
            echo "$new" >> "/tmp/find.$$"
            cd "$thisdir"
            shift 1
            ;;
    esac
done

```

```

if [ ! -f "/tmp/find.$$" ] ; then
    echo `pwd` >> "/tmp/find.$$"
fi

# nacteme lokaci
while read location ; do
    cd "$thisdir"
    depth=0
    searchdir "$location"
done < "/tmp/find.$$"

#uklid po sobe
rm -f "/tmp/find.$$*"

```

Forstuv orientacni beh

Zadani

getob a showob

getob ma za ukol nasledujici:

z webu stahnout dva soubory ob_d.htm a ob_h.htm (divky a hosi)

ktere jsou na serveru www.ob.cz v adresari behy

a kazdy z techto souboru obsahuje asi neco takovedleho:

```
...
...
<PRE>
<A HREF="nnn.htm">nnn</A>
<A HREF="nnn.htm">nnn</A>
<A HREF="nnn.htm">nnn</A>
...

</PRE>
...
```

kde nnn je vzdy trojmistne cislo skupiny.

(jsou tu tedy vypsany vsechny existujici skupiny o.b.)

Dale jsou na webu soubory "skupina_nnn.htm" pro kazdy nnn (viz vyse)
ktera obsahuje neco jako:

```
...
<PRE>
<B>Jmeno Prijmeni</B> neco neco <B>id(viz nize)</B>
datum1 body1 datum2 body2 datum3 body3 datum4 body4
datum5 body5 datum6 body6 datum7 body7 datum8 body8
...
<B>Jmeno Prijmeni</B> neco neco <B>id(viz nize)</B>
datum1 body1 datum2 body2 datum3 body3 datum4 body4
datum5 body5 datum6 body6 datum7 body7 datum8 body8
...
...

</PRE>
```

tedy jsou zde vyjmenovani vsichni clenove dane skupiny.

id je ve tvaru AArrcckk (nebo neco podobneho),

AA jsou nejaka pismena, rr - rok narozeni, cc poradove cislo

(u zen +50) a kk jeste nejaka pismena.

Pod jmenem se vzdy vyskytuji 4 zaznamy datum - body na jedne radce,

radek je jen nekolik a posledni radka nemusí byt cela.

Ukolem prvnioho souboru je tedy uložit si pro kazdeho clena kazde skupiny
jmeno,rok narozeni a body za kazdy zavod (do nejakeho souboru)

showob

si precte soubor, který jsme si vytvorili a syntaxi ma nasledujici

showob <D|H> <n> [from to]

tzn:

D|H - vypise bud divky, nebo hochy.

n - secte body z prvni n bezenych zavodu (podle nich urci poradí)

from to - roky (vypise poze lidi, kteří se narodili v danem období)

ma za ukol tedy vypsát poradí lidí podle specifikace.

pozn: ve družtvech hochu se mohou vyskytovat ženy a naopak.

Puvodni zadani bylo trochu jiné, ale jelikož si ho přesne nepamatuju

tak je to alespon priblizne.

Webovske stranky se maji stahovat nasledovne (ve skutečnosti to tak nejde):

do standartního vstupu prikazu telnet <wwwserver> poslat

"get adresar/soubor" a dve odradkovani. Obsah bude na stand. vystupu.

Getob

```
#!/bin/sh
```

```
#stazeni souboru
```

```
nnnfile="nnn.$$"
```

```
#echo 'get ob_d.htm
#
#get ob_h.htm
#
#' | telnet www.ob.cz
```

```
cat 03.test | sed -n '/<PRE>/,<\/PRE>/p' | cut -s -d\" -f2 | {
while read line ; do
```

```
#echo "get skupina_$line
#
#" | telnet www.ob.cz
```

```
cat "03.$line" | sed -n "/<PRE>/,<\/PRE>/p" | sed "s/<B>/:/g;s/<\/B>/:/g"
| awk -- '
BEGIN { line = ""; first = 1; points = 0; }
/^(\\)?PRE>/ { next; }
#{ print }
/^:/ {
```

```
if (line != "") print line ;
line = "";
for ( i = 1; i <= NF ; i++ ) line = line " " $i;
split(line, header, ":");
rr = substr(header[4], 3, 2);
cc = substr(header[4], 5, 2);
```

```
if ( cc > 50 ) pohl = "D";
else pohl = "H";
line = sprintf("%s:%s:%s:", header[2], rr, pohl);
first = 1;
points = 0;
next;
```

```
}
```

```
{
for ( i = 2; i < NF; i += 2 ) {
points += $i;
if (first == 1) { line = line points; first = 0; }
else line = line "," points;
}
```

```

}

END { if ( line != "" ) print line ; }

,

done

} > 03_output

```

Showob

```

#!/bin/bash

# zpracovani parametru

if [ "$#" -lt "1" ] ; then
    echo uhsdfuioawrhuioarh
    exit 0
fi

from=00
to=99
n=1
nevypis="X"

if [ "$1" = "D" ] ; then
    nevypis="H"
    shift 1
fi

if [ "$1" = "H" ] ; then
    nevypis="D"
    shift 1
fi

if [ "$#" -lt "1" ] ; then
    echo ERRORRR
    exit 1
fi

n=$1

if [ "$#" -eq "3" ] ; then
    from="$2"
    to="$3"
fi

awk "
BEGIN { FS=":" }
{ if ( (\$3 != \"$nevypis\") && (\$2 >= $from) && (\$2 <= $to)) {
    if ( split(\$4, body, "\",\") >= $n )
        printf(\"%s:%d\\n\", \$1, body[$n])
    } }
" 03_output | sort -t: -n -k2 -r | cut -d: -f1

```

FTP Mail

Zadani

Jde vlastne o FTP server pracujici prostrednictvim mailu
Vy napisete prikazy mailem a on vam posle jejich vysledek.
server ma v urcite promenne zapsane uzivatele, kteri ho nesmeji pouzivat
tvaru:
'jmeno jmeno jmeno ... jmeno' kde jmena jsou zacatky mailu (to pred @) a
plati to
pro vsechny mozne mailly s timto zacatkem

dale je v prostredi promenna FTPROOT, která obsahuje adresar, který je
v ramci ftpmailu bran jako zakladni (/), pricemz zadnym prikazem se nesmite
dostat pod nej. tedy treba kdyz FTPROOT='/home/ftpmail', pak se nikdo
nemuze dostat do /etc/passwd :-)
prikazy by to melo zvladat nasledujici:

```
cd (cesta)
... prejde do adresare cesta (bere se to v ramci FTPROOT)
get (jmeno soub. prip s cestou)
... posle po mailu soubor a to tak, ze ho zabali pomoci
    uuencode (viz 1. termin) a vystup prida do tela zpat. mailu
mget (jmeno soub. prip s cestou)
... to same jako get, ale soubor muse byt zadan i jako wildcard
    tzn. napr /adr/*

rget (jmeno adresare)
... posle adresar, který si muzete zabalit tarem(ten to umi viz
1.termin) a posle vam zase uuencodovanej soubor.
ls [cesta]
... posle vypis adresare
put (jmeno souboru)
... ulozi na server soubor, který je vypsán za tímto prikazem
a to uuencodovanej. na prvním radce je mimo jiné:
    BEGIN (mozno jiné jmeno) (prava)
    tedy puvodne to mohl byt jiné soubor, ale my ho chceme uložit
    pod novým názvem.(prava nejsou dulezita, to si uuencode udela
    sam)
```

příklad:
kdyz napiseme mail takto:

```
cd ahoj
get nazdar
put cau
BEGIN hello rw-rw-rwx
HJGsakjhlkdsaf89734lksdficxhv89ew;fsdds
dfj;l0-fvbz;lckvbg9h54kcf;bi-df0ihg;kfg
fdjglsdkfjglkvjcbj;lfhj
END
```

tak vam ftpserver odpovi treba

```
CD command successful
BEGIN nazdar rwxrwx---
dsfgalkdsjlkfasdjdssdfasdf87945tjfd6t4
lkjh43tvsdflKHLkhLKZFHsdkfj897casih897Y
dsfklk
END
```

PUT command successful

Pricemz soubor, který vam vrati je FTPROOT/ahoj/nazdar
a soubor,k tery ste poslali se ulozi jako cau.

Avsak server musi zvladat i chyby, takže když mu date blbej prikaz, tak vam
na nej neodpovi.
Musi take zvladat vice uzivatelu najednou (pouzije se \$\$ pro odkladaci
soubory)

Reseni

```
#!/bin/sh
```

```
echo ftpmailer
```

```
#zakazani uzivatele
zakaz=" test hugo pepa"
FTPROOT="/home/berny/UnixPis/06"
```

```
#zpracovani hlavicky
```

```
from=" "
infrom=0
while read line ; do
  case "$line" in
    "From:*" )
      from=`echo "$line" | sed "s/From: *//"`
      infrom=1
      ;;
    " *" )
      if [ infrom = 1 ] ; then
        from="$from"`echo "$line" | sed "s/ */" `
      fi
      ;;
    * )
      infrom=0
      ;;
  esac
done
```

```
#uroznuti toho pred zavinacem
from=`echo "$from" | cut -d\@ -f1`
```

```
#test na spravnost uzivatele
if [ echo "$zakaz" | grep "$from" ] ; then
  exit 1
fi
```

```
cd "$FTPROOT"
```

```
#zpracovavani pozadavku od uzivatele
```

```
{
while read line ; do
  case "$line" in
    "cd *" )
      CDPATH=`echo "$line" | sed "s/cd //"`
      if [ cd "$CDPATH" ] ; then
        if [ ! pwd | grep "$FTPROOT" ] ; then
          echo Chyba pri zmene adresare. Presun do $FTPROOT
          cd "$FTPROOT"
        else
          echo CD command sucessfull
        fi
      else
        echo CD command unsucessfull
      fi
    ;;
  esac
done
```

```
"get *" )
  FILENAME=`echo "$line" | sed "s/get //"`
  tar -c "$FILENAME" > tar.$$
  uuencode "tar.$$"
  rm -f tar.$$
;;
"put *" )
  FILENAME=`echo "$line" | sed "s/put //"`
  {
    while read file ; do
      echo file
      if [ "$file" = "END" ] ; then
        break;
      fi
    done
  } | uudecode
  echo PUT command sucessfull
;;
esac
done
}
```

Kalkulacka

Zadani

Meli jsme delat kalkulacku, která vyhodnocuje výraz v postfixu - tak jednoduchý to zase nebylo.

Ze vstupu si čte stringy a podle toho, co ti přišlo si to hodil na zásobník popřípadě vyhodnotil, přiřadil do nové proměnné (ta je taky v zásobníku) to co máš v zásobníku.

Co si používal:

```
celá čísla (nebo přirozená)
+ - * /
proměnné - např: aa, abx
nebo indexované proměnné
3 aa ] -> tzn. aa[3]
navíc: ten index mohl být taky proměnná a taky
treba indexována
```

A teď stringy, co nebyly proměnné ani nic jiného, jen vypisovaly nebo prováděly operace se zásobníkem.

```
= do proměnné před rovnásem zápis co je na zásobníku a to samé
z něho vymaz
```

```
. - to taky něco dělalo
```

) to byla jen záležka pro další věc, max - když tohle se obešlo na vstupu, měl si vypsat největší operand na zásobníku

zadal si regexp nebo wildcard a vypsal si hodnoty všech proměnných, co odpovídají tomu re nebo wi.

pak jsi když si něco zadal ze vstupu, tak ti to vypsal zásobník

Takže třeba

```
3 ab ] aa 3 + * i 2 + aa ] =
udělalo tohle aa[i+2] := (3 + aa) * ab[3]
a v zásobníku nezůstalo nic
```

Reseni

```
#!/bin/sh
```

```
zas="zásobník"
rm -f "$zas"
touch "$zas"
v="vars"
```

```
a=""
b=""
```

```
# vyndá něco ze zásobníku
popzas() {
    tail -1 "$zas"
    ed "$zas" << ED_END
    \sd
    wq
    ED_END
}
```

```
# vyhodnotí to, co se mu dá - jako parametr předpokládá co se vyhodnocuje
value() {
    case "$1" in
        [0-9]* )
            echo "$1"
            ;;
        * )
            if [ `echo "$1" | grep '[.*\]'` ]; then
                inside=`echo "$1" | sed 's/^[^\\]*\\([().*\\][^\\]*$/\\1/'`
                inval=`value "$inside"`
                varname=`echo "$1" | sed
"s/\\[[\\$inside\\]\\/[\\$inval\\]/"`
                else
                    varname="$1"
            fi
            cat "${v}/${varname}"
            ;;
    esac
}
```

```
# vyndá ze zásobníku a ohodnotí
popvalue() {
    value `popzas`
}
```

```
# začátek skriptu
```

```
rm -rf "$v" > /dev/null
mkdir "$v"
```

```
tr '[ ]' '[]' | {
    while read command ; do
        if [ "$command" = "" ] ; then
            continue
        fi
        case "$command" in
            '+' | '*' | '-' | '/' )
                a=`popvalue`
                b=`popvalue`
                expr "$a" "$command" "$b" >> "$zas"
                ;;
            '=' )
                b=`popzas`
                a=`popvalue`
                echo "$a" > "${v}/${b}"
                ;;
            '[' )
```

```

a=`popzas`
b=`popvalue`
echo "$a[$b]" >> "$zas"
;;
# tohle se uziva misto  zpetne zavorky - je to kuli casu a vimu
'#' )
echo writing zasobnik
max=`head -1 "$zas" `
maxv=`value "$max"`
{
    while read val ; do
        valv=`value "$val"`
        if [ "$valv" -gt "$maxv" ] ; then
            maxv=valv
            max=val
        fi
    done
} < "$zas"
echo "$max = $maxv"
;;
# takhle se zadava wild card na vypis vseh promennych
/* )
cd "$v"
$wild=`echo "$command" | sed "s#/##" `
ls -l $wild | {
    while read file; do
        echo -n "$file = "
        cat "$file"
    done
}
cd ..
;;
* )
echo "$command" >> "$zas"
;;
esac
done
}

#rm -f "$zas"

```

Maker

Zadani

No meli jsme udelat maker. Tedy file kterej by vypadal asi takhle:

```
all : program1 program2
```

```
program1.o: prog1.c utils.h blabla.h
cc -o prog1.c
```

```
program1.o: prog2.c utils.h utils2.h
cc -o prog2.c
```

```
program1: program1.o utils1.h      [****]
cc -c program1.o
```

(nejsem si uplne jistej ze to ma vypadat presne takhle - vsechny ty optiony a tak, ale v principu by to melo fachtat)
(kdyz tak se mrkni k Forstovi do slajdu na make)

Zadani znelo:

mas adresar a v nem *.c a *.h soubory
1) pro kazdy *.c - nalezt radky #include="nejakejsoubor.h"
pak prolezt vsechny ty 'nejakejsoubor.h' a hledat to samy (pozor na zacykleni!)
a s timhle pak uz tvoris ty jednotlivy radky v tom 'maker'u
2) prohlednout vsechny *.c znovu a tentokrat hledas, jestli to nejsou hlavni file-y
--> main(
kdyz nejake takovej najdes tak:

```
pomoci make vytvorit z techle *.c object fiiles --> *.o
pomoci prikazu ng{ted si nejsem jistej} -f *.o      -->
na vystupu je neco takovyhodle
12AC4555 U {nevyresena reference}
1456ECB5 T {promenna}
678CBD12 D {funkce}
```

a ted si nechas vypsat tim samym zpusobem vsechny ostatni *.o
a hledas na radcichs T nebo D tu tvoji nevyresenou referenci a pridavas ty file-y na ten samej radek [viz (*****)]

Reseni

```
#!/bin/sh
```

```
# preparsovani souboru
ls -l *.cpp *.h | {
```

```
while read file ; do
    touch "$file".x
    cat "$file" | grep -E "^#include *\"[^\"].*\"" | sed -e "s/^#include
\\\"\\\"([^\"]*)\\\"\\\".*/\\1/" > "$file".x
done
}
```

```
# vytvoreni zavislosti
```

```
ls -l *.cpp.x | {
```

```
while read file ; do
#   echo $file
    lines=`wc -l "$file" `
    add=1
    while [ $add = 1 ] ; do
        cat "$file" | {
            while read line ; do
                #   echo including : $line > /dev/tty
                if [ "$line" = "" ] ; then
                    continue
                fi
                cat "$line".x
            done
        } > "$file".b
        cat "$file".b >> "$file"
        sort -u "$file" > "$file".b
        mv -f "$file".b "$file"
        lines2=`wc -l "$file" `
        if [ "$lines" = "$lines2" ] ; then
            add=0;
        fi
        lines="$lines2"
    done
done
}
```

```

# vytvoreni make filu

# tohle jsem netestovl :-) - vytvorime ocka
#find *.cpp -exec cc -o {} \;

#vytvoreni zacatku makefilu

ls -l *.cpp | {

while read filename ; do
    obfile=`echo "$filename" | sed "s/\.cpp\$/\.o/"`
    echo -n "${obfile}: $filename "
    cat "$filename".x | {
        while read line ; do
            echo -n "$line "
        done
    }
    echo
    echo "      cc -o $filename"
done
}
# > Makefile

# nekaj ten uklid
rm -f *.cpp.x *.h.x

exit 0

#vytvorime pomocne soubory, kam sizapamatujeme, co ktery sobor ma v sobe
# uz rozparsevane

ls -l *.o | {
while read filename ; do
    XXX -f "$filename" | grep "^[^ ]* U .*$" | sed "s/^[^ ]* U //" >
"$filename".U
    XXX -f "$filename" | grep "^[^ ]* [DT] " | sed "s/^[^ ]* [DT] //" >
"$filename".TD
done

}

# A protoze jsem nedelal *.o soubory, tak jsem tohle vsechno dal netestoval
taky :-)
have=TD.$$
need=U.$$
touch "$have"
touch "$need"

```

```

ls -l *.o.TD | {

ID=1
all="all :"
while read filenameTD ; do
    if [ cat "$filenameTD" | grep "main" ] ; then
        all="$all program${ID}"
        file=`echo "$filenameTD" | sed "s/\.TD$//" `
        cp "$file".TD "$have"
        cp "$file".U "$need"

        add=1
        while [ "$add" = 1 ] ; do
            while read line ; do

                done < "$need"
            done

            ID=`eval "$ID" + 1`
        fi
    done
    echo "$all"
    echo
} >> Makefile

#smazeme pomocne subory
rm -f *.o.U *.o.TD *.o "$have" "$need"

```

Server Whols (DB zamestnancu fakulty)

Zadani

Jedna se o hierarchickou strukturu - tedy jde vlastne o strom, kde uzly reprezentuji napr. katedru, sekretariat atd. Listy jsou pak jednotlivi zamestnanci. Kazdy uzal muze mit pod sebou libovolne mnozstvi jinych uzlu a listu. List sazmozrejme nikoliv. Oproti stromu je zde jeste zmena: na list muze ukazovat vice uzlu (pakliže pracovník pracuje na více katedrach.)

z uzlu musime byt schopni vyrazit tyto informace (priblizne):
jmeno,email,phone,vedouci ktedry, seznam poduzlu, seznam pracovniku.

z listu pro zmenu:
jmeno,email,phone,seznam uzlu kam dany pracovník patri (pokud dany clovicek pracuje na vicero katedrach).

Navrhnete databazi (mozno realizovat i pomoci adresarove struktury).
Pozor - mohou se vykytovat i pracovníci i katedry, kteří/a maji stejná jména.

Naprogramujte pro tuto databazi :

a) Server, který odpovida na dotazy whois

- 1) dotazy se posilaji pres TCP port 43, obsahem zadosti je pouze regularni vyraz (nebo wildcard - co se vam hodi vic)
- 2) mail na whois@xx.xx.xx.xx, který obsahuje zadost v subjectu nebo ma subject prazdny, a zadost v tele.

Pro Osobu:
Odpovedi je jmeno, email, phone, seznam kateder, jez oblazuje svoji pritomnosti,

Pro Uzel:
jmeno, email, phone, vedouci, seznam pracovniku teto katedry.

b) Program udrzby databaze pro administratora :

- Pridej osobu
- Pridej uvazek osobe (tedy vlastne pridej osobu na dalsi katedru...)
- Smaz uvazek - maze uvazek, v pripade posledni uvazku smaz i

pracovnika
Zmen vedouciho katedry.
Zmen udaje o osobe - telefon, atd...

Tot cele.

Server

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

```
#server - jako retezec pro hledani prijima REGEXP
```

```
if [ $# -ne "1" ] ; then
    echo spousti se s 1 parametrem
    exit 0
fi
```

```
grep "[0-9]*:$1:" katedry |
{
    IFS=:
    while read ID name email tel vedouci ; do
        echo -n $name," $email"," $tel"," " "
        grep "^$vedouci:" zamest | { read ID name rest ; echo -n $name," " ;
    }
        grep ":[0-9,]*$ID[0-9,]*$" zamest | { while read ID name rest ; do
echo -n $name," " ; done }
        echo
    done
}
```

```
grep "[0-9]*:$1:" zamest |
```

```
{
    IFS=:
    while read ID name email tel katedry ; do
        echo -n $name $email $tel " "
        ( tr '[,] ' '[ ]' | {
            while read katID ; do
                grep "^$katID:" katedry | { read ID name rest ; echo -n $name,"
; }
            done
        }
    ) << END
$katdry
END
    echo

    done
}
```

Admin

```
#!/bin/sh

# funkce na hledani maximalniho ID + 1
getNextID() {
    cut -d: -f1 "$1" | sort -r -n | { read max;  expr $max + 1 ;  }
}

case $1 in
-[aA] )
# dalsi mapametry jsou jmeno email telefon IDkatedra IDkatedra....
    ZID=0
    getNextID zamest | {
        read ZID;
        zamrec="$ZID:$2:$3:$4:"
        shift 4
        for KID in $@ ; do
            zamrec="$zamrec$KID,"
        done
        echo $zamrec
    } >> zamest
;;

-[uU] )
#pridani uvazku osobe dalsi parametry jsou ID osoby, ID katedry
    if ! grep "^${2}.*:[^:]*${3}[^:]*" zamest > /dev/null ; then
    {

ed zamest << ED
/^$2:/
s/\$/${3}/
w
q
ED

    } > /dev/null

    fi
;;

-[Ss] )
# smazani uvazku - IDosoby IDkatedry

{
ed zamest << ED
/^$2:/
s/${3},//
g/:\$/ d
w
q
ED
    } > /dev/null

;;
```

```
-[vV] )
# zmena vedouciho - IDkatedry nove IDvedouciho
{
ed katedry << ED
/^$2:/
s/[^:]*\$/${3}/
w
q

ED

} > /dev/null
;;

-[oO] )
# zmena udaju o osobe - IDosoby email telefon
grep "^$2" zamest | {
    IFS=':'
    read id jmeno email tel zblytek

ed zamest << ED
/^$2:/
s/:$email:/:$3:/
s/:$tel:/:$4:/
w
q
ED

} > /dev/null

;;

esac
```

Sort

Zadani

napiste sort, s prepinaci -d + - -r -n a jeste nejake dalsi .

Smeli jsme pouzit v podstate cokoliv (krome samotneho sortu).
Nejvhodnejši reseni bylo vytvorit si tridici zvlást klíce a podle nich pak mergesortem slevat puvodni radky.

Reseni

```
#!/bin/sh
```

```
# rika, kterym se ma zacit porovnavat, a kterym uz se nema
a=1
b=2
d=':'
```

```
r=0
n=0
file="$1"
```

```
bsort=0
```

```
# nacteni parametru pro sorteni
```

```
for par in $@ ; do
case "$par" in
    -r )
        r=1
        ;;
    -n )
        n=1
        ;;
    -b )
        bsort=1
        ;;
    -d? )
        d=`echo "$par" | sed "s/^.//"`
        ;;
    +[0-9]* )
        a=`echo "$par" | sed "s/^.//"`
        a=`expr $a + 1`
        ;;
    -[0-9]* )
        b=`echo "$par" | sed "s/^.//"`
        ;;
    * )
        file="$par"
        ;;
esac
```

```
done
```

```
# vraci 1, pokud je 1 > 2 ; = vraci 0; 2 > 1 vraci -1
# porovnaava uz primo dve pole
compare() {
```

```
#     echo "testing $1 ### $2" > /dev/tty
    res=0
    if [ "$n" = "0" ] ; then
res=`awk "BEGIN{
    a = \"$1\"
    b = \"$2\"
    if (a > b) print(\"1\");
    else if (b > a) print(\"-1\");
    else print(\"0\");
}\"`
```

```
    else
        if [ "$1" -gt "$2" ] ; then
            res='1'
        elif [ "$1" -lt "$2" ] ; then
            res='-1'
        else
            res='0'
        fi
    fi
```

```
    if [ "$r" = '1' -a "$res" != 0 ] ; then
        if [ "$res" = 1 ] ; then
            res='-1'
        else
            res='1'
        fi
    fi
    echo -n "$res"
}
```

```
# porovnaava dve ruzne radky
# navratova hodnota stejna, jako u compare
comparelines() {
```

```
    field="$a"
    result='0'
    while [ "$result" = '0' -a "$field" -lt "$b" ] ; do
        fa=`echo "$1" | cut -d"$d" -f"$field"`
        fb=`echo "$2" | cut -d"$d" -f"$field"`
        result=`compare "$fa" "$fb"`
        echo "$result" > /dev/tty
        field=`expr $field + 1`
    done
    echo -n "$result"
}
```



```

buble() {
    i='0'
    while [ $i -lt $lines ] ; do
        i=`expr $i + 1`
        echo "Bouble sorting : $i/$lines"
        j='1'
        while [ $j -lt $lines ] ; do
            k=`expr $j + 1`
            linea=`cat sort.tmp.$j`
            lineb=`cat sort.tmp.$k`
            cres=`comparelines "$linea" "$lineb"`
            if [ "$cres" = '1' ] ; then
                echo "$linea" > sort.tmp.$k
                echo "$lineb" > sort.tmp.$j
            fi
            j="$k"
        done
    done

# dostava dva parametry $1 - od kud, $2 kam
qsort() {
    if [ $1 -lt $2 ] ; then
        echo "Quick sorting : $1-$2"

        i="$1"
        j="$2"
        kraj=`cat sort.tmp.$1`
        while [ "$i" -lt "$j" ] ; do
            echo "$i $j"
            linei=`cat sort.tmp.$i`
            cres=`comparelines "$linei" "$kraj"`
            while [ "$cres" != '1' -a "$i" -lt "$j" ] ; do
                i=`expr $i + 1`
                linei=`cat sort.tmp.$i`
                cres=`comparelines "$linei" "$kraj"`
                echo "i=$i"
            done
            linej=`cat sort.tmp.$i`
            cres=`comparelines "$linej" "$kraj"`
            while [ "$cres" != '-1' -a "$i" -lt "$j" ] ; do
                j=`expr $j - 1`
                linej=`cat sort.tmp.$j`
                cres=`comparelines "$linej" "$kraj"`
                echo "j=$j"
            done
            echo "$linej" > sort.tmp.$i
            echo "$linei" > sort.tmp.$j
        done

        if [ $i -eq $2 ] ; then
            i=`expr $i - 1`
            echo "$kraj" > sort.tmp.$2
            echo "$linei" > sort.tmp.$1
        fi

        ( qsort $1 $i )
        i=`expr $i + 1`
        ( qsort $i $2 )
    fi
}

```

```

sekacsouboru() {
    # rozsekani vstupu na radky :-)
    lines='0'
    while read line ; do
        lines=`expr "$lines" + 1`
        echo "$line" > "sort.tmp.$lines"
    done

##### zacatek skriptu
sekacsouboru < "$file"

if [ "$lines" -gt "0" ] ; then
    #pripadne sesorteni

#         sekacsouboru < "$file"

        if [ $bsort = 1 ] ; then
            buble
        else
            ( qsort 1 $lines )
        fi

#a vypsani
        tmp='0'
        while [ "$tmp" -lt "$lines" ] ; do
            tmp=`expr "$tmp" + 1`
            cat "sort.tmp.$tmp"
        done
    fi

rm -f sort.tmp.*

```

UID list

Zadani

Máme server MFF, který se jmenuje master. Dále máme servery jednotlivých kateder (napo. KSVI), které mají svoje subdomény. Každá katedra si eviduje zaměstnance ve své databázi (napo. soubor "ksvi.uid") na svém serveru a má poidílen nijaký interval eísel UID, které mu e vyu ívat. Na master serveru je potom databáze "uidlist", která vznikne spojením v ech katederních databází. Na v ech katederních servrech je ulo ena kopie "uidlist". Napi te sadu programu, která tento systém realizuje a program "getuid".

Soubory "*.uid" a "uidlist" mají tuto strukturu:

```
UID    login    fullname
```

(pozor: fullname mu e obsahovat i mezery, proto je vhodné pou ít na oddílování jiný znak)

Na v ech poeítaeích je ulo en soubor uid.cfg, který má tuto strukturu:

```
Jméno_katedry    first_UID    number_UID    adresa_katederního_serveru
```

Je tedy toeba udílat tyto úkoly:

Vyrobí soubory "*.uid". To se udílá tak, e si každá katederní server z promínné \$HOSTNAME zjistí svoji adresu, poté se podívá do souboru uid.cfg a od tamtud si zjistí rozsah UID svých zaměstnancu. Potom prohledá /etc/passwd a vytáhne z niho ty lidi, které k nimu patoí. Z toho potom vytvoí soubor "jméno_katedry.uid" Soubor "*.uid" nijak poslat na server master. Je jedno, jestli pomocí FTP, emailu nebo nieeho jiného. Udr ovat na serveru "uidlist". Je jedno, jestli se to bude díť okam ítí po obdr ení každého nového "*.uid" nebo toeba jednou za týden hromadní. Zminíný "uidlist" dostat zase zpátky na v echny katederní servery. Vytvoí program "getuid", který bude dostávat dva parametry: jméno katedry a fullname nijakého u ívatele (poíkklad volání programu: getuid KSVI Pavel Topfer). Tento program bude spou ít na katederních servrech. Program prohledá "uidlist" a kdy tam u ívatele najde, tak vypí e jeho UID a login. Kdy ho tam nenajde a dotazovaná katedra je jiná, ne jeho katedra, tak zaove. Kdy ho tam nenajde a dotazovaná katedra je jeho (to opít zjistí z "uid.cfg"), tak si najde poslední vyu ívané UID na jeho katedoe a vypí e toto eíslo zvý ené o jednieku. (To kvůli tomu, aby potom nikdo s touto informací mohl zalo íť nový úeet tomuto elovíku). To v e. Poípadné nejasností adresujte na david@steiner.cz

MakeUid

```
#!/bin/sh
HOSTNAME='www.ksvi.cz'
adresa=$HOSTNAME
#----- zitim rozsah UID -----
while read m o d adr ; do
    if [ "$adr" = "$adresa" ]; then
        menokat=$m;
        od=$o
        do=$d
        break
    fi
done < uid.cfg
#-----prehladam etc -----
IFS=':'
while read log p uid gid name home; do
    if [ "$uid" -le "$do" -a "$od" -le "$uid" ]; then #ten je nas
        echo "$uid:$log:$name">>$menokat.uid
    fi
done < etc
#-----poslat to masterovi-----
# cat $menokat.uid | mail -s NEW UID $master
#rm $menokat.uid
```

GetUid

```
#!/bin/bash
#----- check par -----
if [ "$#" = 0 -o "$#" -ge 4 ] ; then
    echo error par
    exit 1
fi
kat=$1
if [ "$#" = 2 ]; then
    meno=$2
else
    meno="$2 $3"
fi
#-----prejdem uidlist-----
IFS=":"
nasiel='false'

while read uid log name; do
    if [ "$meno" = "$name" ]; then
        echo "nasiel som- uid: $uid logname: $log"
        nasiel='true'
        break
    fi
done <uidlist

IFS=" "
if [ "$nasiel" = "false" ]; then #ak nenasiel
    adr="www.ksvi.cz"
    while read m o d ad; do # zistim meno katedry
        if [ "$adr" = "$ad" ]; then
            menokat=$m
            break
        fi
    done <uid.cfg
    if [ "$menokat" != "$kat" ]; then #zla katedra
        echo zla katedra
        exit 1
    else #dobra katedra
        IFS=':'
        while read uid rest; do
            echo $uid >>uid.$$
            done<$kat.uid
            max=`sort -n uid.$$ | tail -1`
            echo `expr $max + 1`
        fi
    fi

#-----uklid-----
test -f uid.$$ && rm uid.$$
```

General

Zadani

i na nas zase pekne smlsnul Novy priklad s nazvem GENERAL (generuj aligatora)
Ukolem je udelat skript GENERAL [jmeno_fakulty] ... nepovinný parametr.
Prazdny parametr bere implicitne vsechny fakulty
Mate soubory FAKULTY (seznam vseh fakult UK, kde na kazdem radku je jmeno_fakulty, "soubor.txt" s udaji o studentech, "format.cd" ... jakou cestinu
dana fakulta pouziva ... 1 radkovy soubor obsahujici ceske znaky, tento soubor
porovnete se souborem "ascii.cd").
Dlasi je ALIGATOR.DB, coz je databaze vseh studentu, ve formatu: rodne cislo,
Prijmeni a Jmeno (cesky), jmeno.prijmeni (presne takto psany), fakulta, rocnik, email (tento udaj si vyplnuje sam student, zalozenim konta na aligatoru).
Udaje v souboru.txt maji format: rodne cislo,jmeno, prijmeni (!!zvlast!!),rocnik

Vasim ukolem je spustenim skriptu vzit data ze souboru.txt dane fakulty, porovnat, zda je student zapsan na aligatorovi a prislusne upravit jeho udaje:
napr. prepsat rocnik, (divky zmena prijmeni apod ...). Jestlize student v teto
databazi neni, pak je novy a je treba ho do ni pridat, Naopak nenajdu li studenta v souboru txt a na aligatorovi uveden je, pak student ukoncil studium
na dane fakulte a je treba ma ucet zrusit s dobou prodleni 1 mesic (tj, nenajdu ho =
pridam do polozky udaj s aktualnim datumem a jestlize tam uz tato polozka je a je
starsi 1 mesice, pak studenta smazu z databaze). Musim si lae dat pozor, protoze 1 student
muze navstevovat vice fakult ... potom ma vice polozek na aligatorovi a ja musim
zrusit jen ten, ktery odpovida fakulte, kterou opustil ...
Vysledek ma byt utridena databaze ALIGATOR.DB, s aktualnimi udaji ...
Hodne stesti, vy co to budete zkouset :-))

Reseni

```
#!/bin/sh

#----- check par -----
--
if [ "$#" -ge 3 ]; then
    echo error par
    exit 1
fi
#----- zisti ci fakulta existuje-----
--
check_fak()
{
    ok='false'
    while read fak s b; do
        if [ "$fak" = "$1" ]; then
            ok='true'
            break
        fi
    done<fakulty
}
#-----zisti ci je student na aligatorovi-----
--
check_stud()
{
    rodc=$1
    IFS=:
    jetam='false'
    while read rc m mpr fak roc mail; do
        if [ "$rodc" = "$rc" ]; then
            jetam='true'
            break
        fi
    done<aligator.db
    IFS=" "
}
```

```

#-----pridaj studaka-----
--
pridaj()
{
p_rc=$1
p_meno=$2
p_pr=$3
p_roc=$4
echo "$p_rc:$p_pr $p_meno:$p_meno.$p_pr:$fakulta:$p_roc">>aligator.db
}
#-----uprav studaka-----
uprav ()
{
urc=$1
umeno=$2
upr=$3
uroc=$4
IFS=':'
while read rc m mpr fak roc zvys; do
if [ "$rc" = "$urc" ]; then
echo "$urc:$supr $umeno:$umeno.$supr:$fakulta:$uroc:$zvys">>aligator.$$
else
echo "$rc:$m:$mpr:$fak:$roc:$zvys">>aligator.$$
fi
done<aligator.db
cp aligator.$$ aligator.db
rm aligator.$$
IFS=" "
}
#----- porovnaj dva datумы -----
porovnaj()
{
zmaz="false"
}
#-----
if [ "$#" != 0 ];then
fakulta=$1
check_fak $fakulta #----- checkne fakultu -----
if [ "$ok" = "true" ]; then
echo "fakulta $fakulta je ok"
else
echo "zla fakulta, zadajte nazov fakulty:"
while read newfak ;do
check_fak $newfak
if [ "$ok" = "true" ]; then
echo "fakulta $newfak je ok"
fakulta=$newfak
break
else
echo "zla fakulta, zadajte nazov fakulty:"
fi
done</dev/tty
fi #-----az kym nezadam spravnu fakultu---

IFS=':'
while read rc prm mpr fak roc zvysok; do #prechadzam aligatora-----
if [ "$fak" = "$fakulta" ]; then
sitam='false'
IFS=" "
while read krc zv; do

```

```

if [ "$krc" = "$rc" ]; then
sitam='true'
break
fi
done<$fakulta.txt
IFS=':'
if [ "$sitam" = "false" ]; then #v txt neni ale v aligovi ano
echo "$prm ukoncil studium na fakulte $fakulta"
datum=`date +%d%m%y`
IFS=" "
case $zvysok in
[a-zA-Z]*'@'*: '[0-9][0-9]* )
dat=`echo $zvysok | cut -d: -f2`
mail=`echo $zvysok | cut -d: -f1`
echo "dna: $dat"
porovnaj $datum $dat
if [ "$zmaz" = "false" ]; then
echo "$rc:$prm:$mpr:$fak:$roc:$mail:$datum">>ali.$$
fi;;
[0-9]*) echo "dna: $zvysok"
porovnaj $datum $zvysok
if [ "$zmaz" = "false" ]; then
echo "$rc:$prm:$mpr:$fak:$roc:$datum">>ali.$$
fi;;
*) echo "$rc:$prm:$mpr:$fak:$roc:$zvysok:$datum">>ali.$$;;
esac
IFS=':'
else
echo "$rc:$prm:$mpr:$fak:$roc:$zvysok">>ali.$$
fi
else
echo "$rc:$prm:$mpr:$fak:$roc:$zvysok">>ali.$$
fi
done <aligator.db
IFS=" "
cp ali.$$ aligator.db
rm ali.$$

while read rod_c meno priezv rocnik; do #prechadzam studakov fakulty----
check_stud $rod_c
if [ "$jetam" = "true" ]; then #ak tam uz je
echo "$meno $priezv je v aligatorovi, upravujem udaje"
uprav $rod_c $meno $priezv $rocnik
else
echo "$meno $priezv nieje v aligatorovi, pridavam ho"
pridaj $rod_c $meno $priezv $rocnik
fi
done<"$fakulta.txt"

fi #koniec jedneho parametru

```

Bindhost

Zadani

Napiste Bindhost - program, který v závislosti na obsahu /etc/hosts a /etc/named.boot vygeneruje zonové soubory pro domeny, které jsou nadefinovány v /etc/named.boot

Tedy jde o to, že v /etc/hosts jsou napsány IP adresy, k nim přiřazena jména strojů. Z těchto údajů se má vygenerovat soubor, který bude konfiguračním souborem DNS serveru, a to jak normálně (záznamy IN A a IN CNAME), tak reversní (IN PTR).
Navíc v /etc/hosts jsou jména buď celá (tedy pocitac.domena.nekde.cz) nebo jen zkrácená (například pocitac1). K těmto zkráceným jménům je nutno doplnit (jen pro naše účely, původní /etc/hosts musí zůstat nezměněn) naši vlastní doménu, která se nachází v /etc/resolv.conf na řádce, která začíná slovem domain.
Pak je třeba udělat nějaké průchody tak, aby výsledek byl korektní pro nameserver.

Na začátku každého souboru pro nameserver je SOA hlavička, ve které je uloženo serialové číslo tohoto záznamu. (tato SOA hlavička může být pro zjednodušení stejná pro všechny zonové soubory - tedy na začátku každého souboru může být napsáno \$include named.soa, například). Toto číslo je ve formátu yyyymmddxx - y je rok, m je měsíc, d je den a xx je poradové číslo změny v tom dni. Tedy je při změně nutné upravit toto serialové číslo, aby odpovídalo (pokud souhlasí datum s dnešním, zvýšit xx o jedničku, pokud nesouhlasí datum, zapsat dnešní datum a xx=01).

Příklad:

```
/etc/hosts
1.2.3.4      www.kotelnik
11.10.9.8    neco.cizidomena.cz
11.12.13.14 pocitac1 pocitac2 pocitac3 #komentar
```

```
/etc/resolv.conf
domain mojedomena.cz
```

```
/etc/named.boot
directory /etc/named
```

```
primary mojedomena.cz mydom.db
primary 13.12.11.in-addr.arpa reverse11.db
```

Vas program by měl vygenerovat do adresáře /etc/named soubor mydom.db:

```
$include named.soa
www      IN      A      1.2.3.4
kotelnik IN      CNAME   www.mojedomena.cz
pocitac1 IN      A      11.12.13.14
pocitac2 IN      CNAME   pocitac1.mojedomena.cz
pocitac3 IN      CNAME   pocitac1.mojedomena.cz
```

```
reverse11.db:
```

```
$include named.soa
14.13.12.11.in-addr.arpa      IN      PTR      pocitac1
```

a příslušným způsobem upravit named.soa

Reseni

```
#!/bin/sh
```

```
#####
```

```
domena=`cat resolf.conf | grep 'domain' | cut -d' ' -f2`;
```

```
den=`date '+%d'`;  
mesiac=`date '+%m'`;  
rok=`date '+%Y'`;  
poradc='01';
```

```
# vytvorenie hlavicky
```

```
[ -d 'temp' ] || mkdir temp;  
cd temp;  
echo $rok$mesiac$den$poradc >named.soa;
```

```
#####
```

```
vytvor_subor()  
{  
cd ..;  
subor=`cat named.boot | grep 'primary '$domena | cut -d' ' -f3`;  
cd temp || echo error;  
touch $subor;  
}  
#####
```

```
hlavicka()  
{  
echo '$include named.soa' > $subor ;  
}  
#####
```

```
prechadzaj()  
{
```

```
cd ..;
```

```
while read line  
do  
poc='1';
```

```
while slovo=`echo $line | cut -d' ' -f $poc` && [ $slovo != '' ]  
2>/dev/null  
do  
poc=`expr $poc + 1`;  
#echo $slovo;
```

```
case $slovo in  
[0-9]*) ip=`echo $slovo 2>/dev/null`;;  
) if [ $poc = '3' ]; then  
prvy=`echo $slovo 2>/dev/null`;  
echo $prvy 'IN A '$ip >>$subor;  
else dalsi=`echo $slovo 2>/dev/null` ;  
echo $dalsi 'IN CNAME '$prvy'.'$domena>>$subor;  
fi;;  
\#*) echo 'komentar '$slovo;;  
#neco.cizidomena.cz) dom=`echo $slovo | cut -d'.' -f2`;  
#  
echo 'heeeeeeej '$dom;;
```

```
esac;  
done;
```

```
done<hosts  
}
```

```
vytvor_subor;  
hlavicka;  
prechadzaj;  
cat $subor;
```

Sismail

Zadani

.A ted priklad (novy, ale jednoduchy):Mate udelat SISMAIL - tj. prihlasovani na zkousku via mail.Existuji soubory:

1. soubor, který obsahuje LOGIN, OBOR, ROCNIK.
- 2.soubor obsahuje OBOR, ROCNIK, PREDMETY (6-mistny kod) a
- 3.soubor, který obsahuje PREDMET, DATUM A CAS,MAXIMALNI pocet lidi a LIDI (loginy).

Prvni dva soubory jsou naplneny na studijnim, 3. se plni v prubehu casu. Tvar a forma souboru je libovolna, jen musi byt v nich obsazeny dane informace, napr. je dobre mit treba kazdy predmet v samostatnem souboru (tj. ne nutne jen tri soubory), coz je povoleno.

Na vstup vam prijde mail.

- 1) zjistit, od koho prisel. Je-li za radkou s FROM login

a) tvaru AAAA1AA@ (kde A je pismo a 1 je cislice), pak mate spravny login. Je-li

b) neco@..., co je obsazeno v shellovske promenne REJECT (tvaru neco mezera neco mezera...(napr. root, mailer...)), pak nic neudelte (proste to zahodite). A

c) neco libovolne jineho, poslete na tu adresu mail, jak by mel login spravne vypadat.

- 2) Zjistite, co po vas mail chce. Najdete to bud na radce SUBJECT nebo na prvni radce textu mailu (poznate tak, ze text je oddelen prazdnou radkou)Jsou nasledujici moznosti, co po vas muzou chtit:

a) LIST - vypise vsechny predmety a je-li na ne ten clovek prihlasen, tak u te zkousky i datum a cas.

b) LIST predmet - vypise termíny zkousek(datum, cas) a jejich volnou kapacitu(nebo maximalni, to uz nevím)

c) RESERVE predmet datum cas - zkontroluje, jestli neni již přihlášen (na nějaký jiný termín) a přihlásí hod

d) DELETE predmet datum a cas - odhlási hoPoznamky: je treba mit skript osetren zamkem a na kazdy prikaz se odpovida mailem s prislusnym textem akce.

Reseni

```
#!/bin/sh
REJECT="root mailer pepooo blbost test"
while read line; do          #spravim si kopiu mailu
echo $line >>mail.$$
```

```
done
#----- uklid -----
uklid()
{
rm mail.$$
exit 1
}
#-----zistim nejake info-----
while read line; do
case $line in
"From:*) from=`echo $line | sed 's/From:/'` ;;
"Subject:*) subj=`echo $line | sed 's/Subject:/'`;;
esac
done<mail.$$
#-----check login-----
pred=`echo $from | cut -d@ -f1`
for name in $REJECT ;do          #nerobim nic
if [ "$name" = "$pred" ]; then
echo "$pred je v REJECT"
uklid
fi
done
case $pred in
[a-z][a-z][a-z][a-z][a-z][0-9][a-z][a-z]) echo spravny login ideme na to;;
*) echo nespravny login,posielam mail          #nespravny login
echo "spravny login vyzerá takto: aaaaalaa" | mail -s"spravny login"
$from
uklid;;
esac
#-----zistim, co po mn echce-----
if [ "$subj" = "" ]; then
comm=`cat mail.$$ | sed '1,/^\$/d' | head -1`
else
comm=$subj
fi
#-----spracuj prikaz-----
echo "login: $login"
case $comm in
"LIST") #----- LIST -----
echo "prikaz: $comm"
IFS=:
while read predmet dat cas poc mena; do
prihlaseny='false';
IFS=" "
for m in $mena; do
if [ "$m" = "$login" ] ; then
prihlaseny='true'          # je prihlaseny
fi
done
if [ "$prihlaseny" = 'false' ]; then
echo $predmet >>list
else
echo "$predmet si prihlaseny! datum: $dat cas:$cas">>list
fi
IFS=:
done <predmety
IFS=" "
```

```
before=""          # vypis jednotlivé predmety (ak sa
opakujú, tak len raz)
```



```

while read predmet zvysook;do
    if [ "$predmet" != "$before" ]; then
        echo "$predmet $zvysook"
    fi
    before=$predmet
done<list
rm list
uklid;;

"LIST "[a-zA-Z]*) #----- LIST predmet -----
    echo "prikaz: $comm"
    predmet=`echo $comm | cut -d" " -f2`
    IFS=':'
    while read predm dat cas kap mena; do
        if [ "$predmet" = "$predm" ];then
            IFS=" "
            pocm='0'
            for m in $mena; do
                pocm=`expr $pocm + 1`
            done
            IFS=':'
            volne=`expr $kap - $pocm`
            echo "datum:$dat cas:$cas volne kapacita:$volne"
        fi
    done<predmety
    IFS=" "
uklid;;

"RESERVE"*) #----- RESERVE termin -----
    echo "prikaz: $comm"
    predmet=`echo $comm | cut -d" " -f2`|| { echo error par; uklid; }
    datum=`echo $comm | cut -d" " -f3`|| { echo error par; uklid; }
    cas=`echo $comm | cut -d" " -f4`|| { echo error par; uklid; }

IFS=':'
while read predm dat c kap mena; do #check ci u znie je prihlaseny
    if [ "$predm" = "$predmet" ]; then
        IFS=" "
        for m in $mena; do
            if [ "$m" = "$login" ];then
                echo "na $predm uz si prihlaseny!!! datum:$dat cas:$c"
                uklid
            fi
        done
    fi
done < predmety

IFS=':'
ok='false'
while read predm dat c kap mena; do #check ci dal dobry termin
    if [ "$dat" = "$datum" -a "$c" = "$cas" ];then
        ok='true'
    fi
done < predmety
if [ "$ok" = "false" ]; then
    echo "takyto termin nieje vypisany"
    uklid
fi
echo "prihlasujem na datum:$datum cas:$cas" #setko ok, mozem prihlasit

```

```

while read predm dat c k mena; do #prihlasim ho
    if [ "$predm" = "$predmet" -a "$dat" = "$datum" -a "$c" = "$cas" ]; then
        echo "$predm:$dat:$c:$k:$mena $login">>predmety.$$
    else
        echo "$predm:$dat:$c:$k:$mena">>predmety.$$
    fi
done<predmety
IFS=" "
cp predmety.$$ predmety
rm predmety.$$
uklid;;

"DELETE"*) #----- DELETE termin -----
    echo "prikaz: $comm"
    predmet=`echo $comm | cut -d" " -f2`|| { echo error par; uklid; }
    datum=`echo $comm | cut -d" " -f3`|| { echo error par; uklid; }
    cas=`echo $comm | cut -d" " -f4`|| { echo error par; uklid; }

IFS=':'
ok='false'
while read predm dat c kap mena; do #check ci dal dobry termin
    if [ "$dat" = "$datum" -a "$c" = "$cas" ];then
        ok='true'
    fi
done < predmety
if [ "$ok" = "false" ]; then
    echo "takyto termin nieje vypisany"
    uklid
fi

jetam='false'
while read predm dat c k mena; do #zistim ci tam vobec je
    if [ "$predm" = "$predmet" -a "$dat" = "$datum" -a "$c" = "$cas" ]; then
        IFS=" "
        for m in $mena; do
            if [ "$m" = "$login" ]; then
                jetam='true'
                echo "odhlasujem $login z terminu $dat $cas"
                break
            fi
        done
    fi
done<predmety
if [ "$jetam" = 'false' ]; then
    echo "na tomto termine nie ste prihlaseny!!!"
    uklid
fi
while read line; do #secko ok, mozem odhlasit
    case $line in
        "$predmet:$datum:$cas"*) echo $line | sed "s/$pred //" >>predmety.$$;;
        *) echo $line >> predmety.$$;;
    esac
done<predmety

cp predmety.$$ predmety
rm predmety.$$
uklid
;;
*) echo error command
    uklid;;
esac

```

Links

Zadani

Zadani nevím přesně, ale je to něco o tomhle:

Ma se naprogramovat script, který vypíše pro daný soubor všechny hard-linky a soft-linky (i rekurzivně) a pro adresar asi to same.

Nic víc nevím

Resení

```
#!/bin/sh
#       showlinks
#       vypise vsechny souboru, ktere odkazuji na dany soubor

TMP1="/tmp/showl1$$";
TMP2="/tmp/showl2$$";
TMP3="/tmp/showl3$$";
XTMP="/tmp/showl0$$"; #zde je ulozen seznam symlinku a kam ukazují za sebou
TMP="/tmp/showl$$";
FILE=""; # jmeno naseho fajlu ; prazdne --> byl zadán adresar
DIR=""; # adresar ve kterém je náš fajl
AROOT=""; # root filesystemu s našim fajlem
BOBR="je tu";
DEBUG=""; #neprazdna --> debug
RECURSION_LEVEL="2";

unset BOBR; #bobr odjel na dovolenou

clean()
{
    rm -f "$TMP" "$TMP1" "$TMP2" "$TMP3" "$XTMP" ;
}

get_file() #parsuje cestu vraci string za posledním '/'
{
    echo "$1" | awk -F "/" ' { printf("%s\n",$NF) } ' ;
}

get_dir() #parsuje cestu vraci vše kromě stringu za posledním '/'
{
    echo "$1" | awk -F "/" ' { for(i=1;i<NF;i++) { printf("%s/",$i); } } ' ;
}

inter_cd() #zabsolutní relativní cestu k adresari.
{
    cd "$1" || { echo "/"# ; return 1; }; #soubor nemusí existovat
    POM1=`pwd`;
    if [ "$POM1" = "/" ] ; then
        echo "/";
    else
        echo "$POM1/";
    fi;
}


```

```
get_name() #parsuje vstupní název souboru ; adresar vrací do $DIR ; soubor $FILE
{
    if ( echo "$1" |grep '^.*$' ) > /dev/null ; then
        DIR="$1";
    else
        FILE=`get_file "$1"`;
        DIR=`get_dir "$1" `;
    fi;
    if ( echo "$DIR" |grep '^\.\/.*$' ) > /dev/null ; then
        DIR=`inter_cd "$DIR"`;
    elif ( echo "$DIR" |grep '^\.\/.*$' ) > /dev/null; then
        DIR=`inter_cd "$DIR"`;
    fi;
}

deabsolut1()
{
    if [ "$1" = '/' ] ; then
        echo "$1";
    else
        echo "$1" | sed 's:^(.*)$:\1:' ;
    fi;
}

deabsolut2()
{
    if [ "$1" = '/' ] ; then
        echo "$1";
    else
        echo "$1" | sed 's:^(.*)/$:\1:' ;
    fi;
}

get_part()
{
    if ( echo "$1" | grep -v '/' ) > /dev/null ; then
        echo "$1";
    else
        echo "$1" | awk -F "/" ' { printf("%s", $1) ; } ' ;
    fi;
}

get_rest()
{
    if ( echo "$1" |grep -v '/' ) > /dev/null ;then
        echo;
    else
        echo "$1" | awk -F "/" ' { for(i=2;i<NF;i++) {
printf("%s/", $i); };
        printf("%s\n", $NF); } '
    fi;
}

is_lower()
#funkce na porovnávání dvou absolutních cest; vrací 0 je-li $1<$2
#nepracuje lexikograficky; např. pro cesty /hue/brb a /hue/ahh vrací 1
{
    XPOM1=`deabsolut1 "$1"`;
    XPOM1=`deabsolut2 "$XPOM1" `;
    XPOM2=`deabsolut1 "$2" `;
}


```

```

XPOM2=`deabsolut2 "$XPOM2"`;
if [ "$XPOM1" = "$XPOM2" ]; then
    return 1;
fi;
XPOM3=`get_part "$XPOM1" `;
XPOM4=`get_part "$XPOM2" `;
while [ "$XPOM4" = "$XPOM3" ]; do
    XPOM1=`get_rest "$XPOM1" `;
    XPOM2=`get_rest "$XPOM2" `;
    XPOM3=`get_part "$XPOM1" `;
    XPOM4=`get_part "$XPOM2" `;
done;
if [ -z "$XPOM4" ]; then
    return 1;
elif [ -z "$XPOM3" ]; then
    return 0;
else
    return 1;
fi;
}

get_root()
{
    df | awk ' { if( NR != 1 ) { printf("%s\n", $NF) >> "'$TMP1'" } } ' ;
    > "$TMP2";
    POM1=`deabsolut2 "$DIR"`;
    while read x ; do
        if [ "$x" = "$POM1" ]; then
            AROOT="$x";
            return 0;
        elif is_lower "$x" "$POM1"; then
            echo "$x" >> "$TMP2";
        fi;
    done < "$TMP1";
    POM1="/";
    while read x ; do
        if is_lower "$POM1" "$x" ; then
            POM1="$x";
        fi;
    done < "$TMP2";
    AROOT="$POM1";
    return 0;
}

get_hardlinks()
{
    if [ -d "$DIR$FILE" ]; then
        INUM=`ls -id "$DIR$FILE" | sed 's/^\([0-9][0-9]*\) .*$/\1/'`;
    else
        INUM=`ls -i "$DIR$FILE" | sed 's/^\([0-9][0-9]*\) .*$/\1/'`;
    fi;
    cd /;
    find "$AROOT" -xdev -inum "$INUM" 2> /dev/null > "$TMP1";
    if [ -n "$BOBR" ]; then
        echo "Mas tu ty hardlinky bobre ;-] ";
        cat "$TMP1";
    fi;
}

get_allsofts()

```

```

{
    > "$XTMP";
    > "$TMP3";
    POM1="$FILE";
    POM2="$DIR";
    > "$TMP2" ;
    # cat linky > "$TMP2";
    # cat > "$TMP2" << _EOF_
    #/tmp/soft01
    #/tmp/...dir/soft02
    #/tmp/soft02
    #/tmp/...dir/soft03
    #_EOF_
    find / -type l -print 2> /dev/null > "$TMP2";
    while read x; do
        POM3=`ls -l "$x" | sed 's/^. * -> \(.*\)$/\1/'`;
        if [ -n "$DEBUG" ]; then
            echo "softlink $x";
            echo "ukazuje na: $POM3 ";
        fi;
        POM4=`get_dir "$x"`;
        cd "$POM4";
        get_name "$POM3";
        if [ -z "$DIR" ]; then
            DIR="$POM4";
        elif echo "$DIR" | grep -v '^. *$' > /dev/null ; then
            DIR="$POM4$DIR";
        fi;
        if [ "$DIR" = '/' ]; then
            if [ -n "$DEBUG" ]; then
                echo "tento symlink ukazuje na neexistující soubor "
            fi;
            continue; #softlinky co nikam neukazují
        else
            POM3="$DIR$FILE";
            POM3=`deabsolut2 "$POM3"`;
            echo "$POM3" >> "$XTMP";
            echo "$POM3" >> "$TMP3"; #tohle se v dalsim jevi jako
            zbytecne
            echo "$x" >> "$XTMP";
            if [ -n "$DEBUG" ]; then
                echo "$POM3 ";
            fi;
            echo "$x";
            fi;
        fi;
    done < "$TMP2";
    FILE="$POM1";
    DIR="$POM2";
    if [ -n "$DEBUG" ]; then
        cat "$XTMP" > ~/shell/links/xtmp ;
    fi;
    if [ -n "$BOBR" ]; then
        echo "Dneska se s tim bobre nepareme ";
    fi;
    if [ -n "$DEBUG" ]; then
        cat "$XTMP";
        clean;
        exit 0;
    fi;
}

```

```

get_softlinks()
{
    #nyni jsou ve "$TMP1" hardlinky ve "$TMP2" nic ve "$TMP3" prelozene
softlinky
    # v "$XTMP" je seznam softlinku a kam ukazuji
    count="1"; #rekurzi osetrim omezenim pruchodu cyklem --> limitujici
    ok="no"; #pro testovani jestli byly nalezeny jeste dalsi symlinky
    > "$TMP2";
    > "$TMP3";
    while [ `cat "$TMP1" | wc -l ` -gt "0" ]; do #overime jestli sme uz
na konci
        if [ -n "$DEBUG" ]; then
            echo "v souboru s hardlinkama je: ";
            cat "$TMP1";
            echo "_EOF_";
        fi;
        while read x; do          #nacteme jmeno zpracovavaneho fajlu na
ktery vedou syml.
            while read y; read z ; do #do y nacteme jmeno
softlinku
                # do z nacteme kam ukazuje
softlink
                    if [ "$x" = "$y" ]; then #ukazuje na nas fajl
                        echo "$z";          #vypiseme ho
                        ok="yes";
                        echo "$z" >> "$TMP2" ;
                        echo "$y" >> "$TMP3";
                        echo "$z" >> "$TMP3";
                        if [ -n "$DEBUG" ]; then
                            echo "symlink $z nam
ukazuje na fajl $x .je predan dal "
                                fi;
                        elif is_lower "$y" "$x"; then #ukazuje na
podadresar napr.
                            #pro fajl /home/hue/brbla softlink /tmp/brb ukazuje na /home/hue
                                POM1="$XPOM2"; #proto je sem ulozen
string brbla
                                    echo "$z/$POM1" ; #fajl /tmp/brb/brbla
je vypsán
                                        echo "$z/$POM1" >> "$TMP2";
                                #/tmp/brb/brbla je predan pro dalsi zprac.
                                    echo "$y" >> "$TMP3";
                                    echo "$z" >> "$TMP3";
                                    ok="yes";
                                    if [ -n "$DEBUG" ]; then
                                        echo "symlink $z nam ukazuje na
$y coz je podadresar cesty $x";
                                            echo "Bude zarazen dale do
slosovani";
                                                fi;
                                            else #symlink nema zadny vztah k hledanemu
fajlu; bude opet testovan
                                                if [ -d "$y" ]; then
                                                    echo "$y" >> "$TMP3"; #v dalsi
smycce
                                                        echo "$z" >> "$TMP3";
                                                        if [ -n "$DEBUG" ]; then
                                                            echo "symlink $z ukazuje
na adresar $y. bude zarazen do slosovani";
                                                                fi;
                                                                elif [ -n "$DEBUG" ]; then

```

```

        echo "symlink $z ukazuje na
soubor $y a je naprd";
            fi;
        done < "$XTMP";
        done < "$TMP1";
        mv "$TMP2" "$TMP1"; #fajly na sva mista ;=]
        mv "$TMP3" "$XTMP";
        > "$TMP2";
        > "$TMP3";
        if [ "$count" -gt "$RECURSION_LEVEL" ]; then
            echo "Recursion level reached. Terminating. ";
            break;
        else
            count=`expr "$count" + "1" `;
        fi;
        if [ -n "$DEBUG" ]; then
            echo "soubor s hardlinkama je: ";
            cat "$TMP1";
            echo "_EOF_";
            echo "soubor se symlinkama pro parsovani je: ";
            cat "$XTMP";
            echo "_EOF_";
            echo "uroven rekurze je $count";
        fi;
        if [ "$ok" = "no" ]; then
            exit 0;
        else
            ok="no";
        fi;
        done;
    }

usage()
{
    echo "usage: $sh [-r RECURSION_LEVEL ] file ";
}

trap ' clean; exit 0; ' 1 2 3 15;

sh="$0";

if [ "$#" -eq "0" ]; then
    usage;
    exit 0;
fi;

if [ "$1" = "-r" ]; then
    shift;
    if echo "$1" |grep '^[0-9][0-9]*$' > /dev/null ;then
        RECURSION_LEVEL="$1";
        shift;
    else
        echo "Bad number of recursion level. " ;
        exit 1;
    fi;
fi;

if [ "$#" -eq "0" ]; then
    usage;
    exit 0;

```

```
fi;

get_name "$1";
get_root ;
#cat > "$TMP1" <<_EOF_
#/tmp/hardlink
#_EOF_
get_hardlinks ;
echo "Hardlinks:";
cat "$TMP1";
echo "Softlinks:";
get_allsofts ;
#cat xtmp > "$XTMP";
get_softlinks > "$TMP";
cat "$TMP" | sort | uniq; #aby v tom nebyl bordel
clean;
exit 0;

# [--- EOF
```

Uudecode (skladacka)

Zadani

Naprogramujte ???

From: "Tomas Kezes"

Sent: Fri, 22 Sep 2000 17:10:27 +0200

```
...
mas nejake uudecode, co zere vstupny subor v tvare
begin file_name prava
    zakodovany text, ktory neobsahuje begin ani end
end
a ten ti subor rozkoduje
```

```
mas z $MAIL vybrat vsetky subory v takomto tvare
a dekodovat ich
subory v $mail vyzeraju:
begin file_name prava cislo_suboru/pocet_suborov
.....
end
teda ich treba pozliepat
```

```
-neprichadzaju v spravnom poradí
-ak nepride niektora cast, tak sa pise chyba
-a ak subor, co sa poskladal uz existuje, tak
sa opytat, ci ho prepises
```

to je tak asi vsio

Reseni

```
#!/bin/sh
```

```
MAIL='zblepty zblepty
begin testy prava 2/2
prvnitest - druha radka
end
```

nejaky bordel

```
begin testy prava 1/2
prvnitest - prvni radka
end
```

dalsi bordel

```
begin testchyby prava 4/5
zblepty
end
```

,

```
yes="yes"
yesno() {
    aswer=""
    while [ "" = "" ] ; do
        case "$answer" in
            [yY][eE][sS] )
                yes='yes'
                break
                ;;
            [nN][oO] )
                yes='no'
                break
                ;;
            * )
                echo '[yes/no]' > /dev/tty
                read answer < /dev/tty
                ;;
        esac
    done
}

# roztrhani vstupu z promene MAIL
{
while read begin filename prava index ; do
if [ "$begin" = "begin" -a "$filename" != "" ] ; then
    index=`echo "$index" | tr '/' '.'`
    echo "$index" >> "${filename}.mainindex"
    echo "$prava" > "${filename}.prava"
    {
        while read line ; do
            if [ "$line" = 'end' ] ; then
                break
            fi
            echo "$line"
        done
    } > ${filename}.index.${index}
fi
done
} << MAIL_END
$MAIL
MAIL_END
```

```

# srovnani radku v mainindexech, kontrola vyskytu vsech radku a sesypani do
jednoho souboru
ls -l *.mainindex | {
    while read file ; do
        sort -n -t\ . -k1 "$file" > "${file}.s"
        rm -f "$file"

        # kontrola usporadani - asu udelame awkem
        awktest=`awk '
BEGIN { id = 1; last = 0; FS="."; ok = 1; }
{
    if ( $1 != id ) {
        ok = 0;
        exit;
    }
    last = $2;
    id++;
}
END {
    if ( last + 1 != id ) ok = 0;
    if ( ok == 0) printf("0");
    else printf("%d", last);
}
' < "${file}.s" `

        ofile=`echo "$file" | sed "s/./mainindex$/"`
        if [ "$awktest" != "0" ] ; then
            #sliti souboru dohromady
            echo -n "begin $ofile " > "${file}.uu"
            cat "${ofile}.prava" >> ${file}.uu

            ID=1
            while [ $ID -le $awktest ] ; do
                cat "${ofile}.index.${ID}.${awktest}" >>
${file}.uu

                ID=`expr $ID + 1`
            done

            #test, zda existuje
            if [ -f "$ofile" ] ; then
                echo "$ofile uz ekzistuje. Ma se prepsat ?"
                yesno
                if [ "$yes" = "no" ] ; then
                    break
                fi
            fi

            echo "end" >> "${file}.uu"

            # dekodovani
            # uudecode "${file}.uu"
            # cat "${file}.uu"
        else
            echo "NEPRISLI VSECHNY CASTI K SOUBORU $ofile"
        fi

        # smazani tempovych souboru
        rm -f ${file}* ${ofile}.index.* ${ofile}.prava
    done
}

```

Touringuv stroj

Zadani

Neformální definice Touringova stroje (prevzato z KSP):

Turinguv stroj sestává z pásky a oídící jednotky. Páska Turingova stroje má jen jeden konec, a to levý (doprava je nekonečná) a je rozdělena na políčka.

Na každém políčku se nachází právě jeden znak z abecedy Σ (to je nějaká konečná množina, o které navíc víme, že obsahuje znak Λ). Nad páskou se pohybuje hlava stroje, která děm okamžiku je nad právě jedním políčkem.

Oídící jednotka stroje je v každém okamžiku v jednom stavu ze stavové množiny Q (opět nějaká konečná množina) a rozhoduje se podle pochoďové funkce $f(q, z)$, která pro každou kombinaci stavu q a znaku z , který je zrovna pod hlavou, dává uspořádanou trojici (q', z', m) , po čemž je stav, do kterého oídící jednotka půjde v dalším kroku, z' znak, kterým bude nahrazen znak z umístěný pod hlavou a koneční m je buďto L nebo R podle toho, zda se má hlava posunout doleva nebo doprava.

Výpočet Turingova stroje probíhá takto: na počátku je hlava nad nejlevějším políčkem, na počátku pásky jsou uložena vstupní data (zbytek pásky je vyplněn symboly Λ) a oídící jednotka je ve stavu q_0 . V každém kroku výpočtu se Turinguv stroj podívá, co říká funkce f o kombinaci aktuálního stavu a znaku pod hlavou, načež znak nahradí, půjde do nového stavu a posune hlavu v určeném směru. Takto pracuje do té doby, než narazí hlavou do levého okraje pásky, čímž výpočet končí a páska obsahuje výstup stroje.

Formálně:

Turinguv stroj je uspořádána petice $(\Sigma, Q, \delta, q_0, F)$, kde Σ je konečná množina symbolů (abeceda) obsahující symbol Λ , Q je konečná množina stavů, q_0 prvek Q je počáteční stav stroje, $\delta: \Sigma \times Q \rightarrow \Sigma \times Q \times \{L, R, \cdot\}$ je přechodová funkce stroje a F podmnožina Q je množina koncových stavů.

Výpočet se pak nadefinuje indukcí jako posloupnost kroků začínajících v q_0 a řízených přechodovou funkcí (viz výše).

Pod UNIXem bych to asi celé programoval tak, že bych napsal sedový skript, který prepíše zadání Turingova stroje na jiný sedový skript :)

Reseni

```
#!/bin/sh
```

```
# předpokládá $1 - input soubor, $2 - funkce $3 - počáteční stav
```

```
if [ "$1" = '' -o "$2" = '' -o "$3" = '' ] ; then
    echo 'Použití : turing.sh <input> <funkce> <počáteční stav> ;
    počáteční stav pro testování dejte na 1'
    exit 0
fi
```

```
# aby jsme si neznepokojili původní soubor - to by byla škoda
cp "$1" "$1".$$
```

```
line=1
file="$1".$$
func="$2"
stav="$3"
maxlines=`cat "$file" | wc -l`
```

```
while [ "$line" -gt 0 ] ; do
```

```
    echo "Stav : $stav Radka : $line MaxRadka : $maxlines"
```

```
    # načteme tu "správnou radku"
    znak=`head -"$line" "$file" | tail -1`
    f=`grep "^${stav}:${znak}:" "$func"`
```

```
    if [ "$f" = '' ] ; then
        echo "NEDEFINOVANÁ FUNKCE : Stav : $stav, Znak : $znak"
        break;
    fi
```

```
    novyznak=`echo "$f" | cut -d: -f3`
    novystav=`echo "$f" | cut -d: -f4`
    posun=`echo "$f" | cut -d: -f5`
```

```
#    echo "ECHO : $znak, $f $posun, $stav"
```

```
    # zasáhne ED a upraví "stavovou pásku"
    ed "$file" > /dev/null 2>&1 << ED_END
$line s/${znak}/${novyznak}/
wq
ED_END
```

```
    # změní stav a kdyžtak prodloužíme náš soubor
    line=`expr "$line" "${posun}" 1`
    echo " before test : $line = $posun; $maxlines"
    if [ "$line" -gt "$maxlines" ] ; then
        echo "S" >> "$file"
        maxlines="$line"
    fi
    stav="$novystav"
```

```
done
```

```
echo "----- Výstup ze stroje -----"
cat "$file"
```

```
rm -f "$file"
```


Placeni na koleji

Zadani

Más dva soubory
jména: rod. c. jméno příjmení
platby:cena za leden cena za únor
 rodný c. kolik placeno za leden kolik za únor ...

a má vypsát lidicky, který maj zaplaceno za danej mesic vypsát serazený podle český abecedy (vcetne "ch") a jeste kdyz más den v mesici mesi nez 5, tak staci minulej mesic zaplaceney to samé platí s koncem mesice(ktorej je prominlivej)

jo a na zaplacení ti staci polovina částky uvedená v souboru platby na daný mesic.

Reseni

```
#!/bin/sh
```

```
day=`date '+%-d'`  
month=`date '+%-m'`  
year=`date '+%Y'`
```

```
echo "$day $month $year"
```

```
#***** zda se ma pocitat i predchozi mesic  
before="$month"  
if [ $day -le 5 ] ; then  
    before=`expr "$month" - 1`  
    if [ $before -eq 0 ] ; then  
        before=12  
    fi  
fi
```

```
echo "before = $before"
```

```
#***** zda se ma pocitat i nasledny mesic  
mod4=`expr "$year" % 4`  
mod100=`expr "$year" % 100`  
mod1000=`expr "$year" % 1000`
```

```
echo "mod4 = $mod4 mod100 = $mod100 mod1000 = $mod1000"
```

```
days='31:28:31:30:31:30:31:31:30:31:30:31'  
if [ "$mod4" = 0 -a \( "$mod100" != 0 -o "$mod1000" = 0 \) ] ; then  
    days='31:29:31:30:31:30:31:31:30:31:30:31'  
fi
```

```
maxday=`echo "$days" | cut -d: -f"$month"`
```

```
echo "maxday = $maxday"
```

```
diff=`expr "$maxday" - "$day"`  
after="$month"  
if [ "$diff" -le 5 ] ; then  
    after=`expr "$month" + 1`  
    if [ $after -eq 13 ] ; then  
        before=1  
    fi  
fi
```

```
echo "after = $after"
```

```
#----- funkce, zda nas clovek prosel  
# 1. parametr - kolik ma platit, 2. v kterem mesici, 3. lina s clovekem
```

```
OK=0
```

```
zaplatil() {  
    field=`expr $2 + 1`  
    platil=`echo "$3" | cut -d: -f"$field"`  
    #echo "platil = $platil maplatit = $1" > /dev/tty  
    if [ $platil -ge $1 ] ; then  
        #        echo zaplaceno > /dev/tty  
        OK=1  
    fi  
}
```

```
#***** zacina hledani
```

```
{  
    # nacteme z prvnioho radku zajimave ceny-----  
    read ceny  
    a=`echo "$ceny" | cut -d: -f"$before"`  
    b=`echo "$ceny" | cut -d: -f"$month"`  
    c=`echo "$ceny" | cut -d: -f"$after"`  
    a=`expr $a / 2`  
    b=`expr $b / 2`  
    c=`expr $c / 2`
```

```
echo "a = $a b = $b c = $c" > /dev/tty
```

```
# prohlidneme vsechny  
while read line ; do
```

```
    OK=0  
    zaplatil "$a" "$before" "$line"  
    zaplatil "$b" "$month" "$line"  
    zaplatil "$c" "$after" "$line"  
    if [ $OK = 1 ] ; then  
        rc=`echo "$line" | cut -d: -f1`  
        grep "^$rc" jmena | cut -d: -f2  
    fi
```

```
done
```

```
# vystup posleme do specialniho sortu na trideni i podle ch :-)  
} < platby | sed 's/^\([Cc][h]\)/hzzzzzzzzzz\1/' | sort | sed  
's/^hzzzzzzzzzz//'
```

Strom odkazu na html serveru

Zadani

na www serveru jsou vselijak provazane html soubory a nasim ukolem bylo vypsat strom tech odkazu. Napriklad takto:

```
0 index.html
1 welcome.html
2 contents.html
1 goodbye.html
index.html -> welcome.html, goodbye.html
welcome.html -> contents.html
```

Odkazem je samozrejme cokoliv mezi "", tzn. absolutni cesta, relativni cesta (vcetne ./ ../), popr. i adresar. V pripade adresare najit prvni soubor z shellovske promenne INDICES (jmena souboru oddelena mezerami), který existuje na dane ceste.

Na ustnim jsem vysvetlil, co který kus kodu dela. Meli jsme to napsat pruchodem do sirky, nicmene ja to prochazel do hloubky a k tomu akorat rekl, ze jsem ho osalil, zamyslel se a po chvilce premysleni rekl "Tak dobre." a napsal mi zapocet.

Reseni

```
#!/bin/sh
prejdi()
{
stranka=$1
echo $stranka>>presiel
```

```
odk=`cat $stranka | grep -e '<a href=' | sed 's/<a href="//'|sed 's/"//' |
sed 's~</a>~~`
```

```
echo -n "$stranka --> "
echo $odk
echo " "
```

```
for s in $odk; do
pres='false'
case $s in
'./'*) s=`echo $s | sed 's~/~~g`;;
esac
```

```
while read l;do
if [ "$l" = "$s" ]; then
pres='true'
fi
done<presiel
if [ "$pres" = 'false' ]; then
```

```
prejdi $s
fi
done
}
```

```
prejdi index.html
```

```
#----- ukli -----
rm presiel
```

Tar

Zadani

Naprogramujte program tar, který umi "zabalit" dane soubory do jednoho. Syntaxe programu by mela byt nasledujici:

```
tar [t|x|c|f <soubor> [soubory ...]
```

kde jednotlivé parametry znamenaji nasledujici:

```
t .....
    vypise obsah (tedy jmena souboru a adresaru) "zapakovanych" v
    souboru.
x .....
    "rozbali" soubor do aktualniho adresare
c .....
    "zapakuje" dalsi uvedene soubory a adresare (cely jejich podstrom)
    do souboru
```

parametr f je z historickych duvodu povinny

program by mel po rozbaleni zachovat u souboru i adresaru skupinu a vlastnika, cas posledni modifikace (nikoliv u linku) a linky (soft, hard pouze v ramci jednoho adresare). Uzitecny je take prikaz uuencode, který na vstup prichoji soubor predela na vystup ve formatu

```
begin
....
...
....
end
```

kde jsou pismena velke abecedy a nejake znaky (v tomto pripade se tedy nebude komprimovat, ale zvetsovat), a prikaz uuencode, který udela pravy opak prikazu uuencode. Je mozne take vymyslet jiny zpusob.

Jesti jste to nepochopili, tak by vysledny soubor mel vypadat nejak takto:

```
<informace o souboru>
(napr. jako vypis ls : -rw-r--r-- user group date+time name (popr. link) )
begin
....
...
end
<informace o dalsim souboru>
begin
...
...
atd.
```

Tar - Reseni

```
#!/bin/sh

numrights() {
    total=0
    if echo "$1" | grep "^r.*" > /dev/null ; then
        total=4
    fi

    if echo "$1" | grep "^w.*" > /dev/null ; then
        total=`expr "$total" + 2`
    fi

    if echo "$1" | grep "^..x.*" > /dev/null ; then
        total=`expr "$total" + 1`
    fi

    echo $total
}

# ulozi informace o souboru
savedata() {
    ls -l -d "$1" | {
        read prava a user group size mesic den cas jmeno
        if expr "$prava" : "l.*$" > /dev/null ; then
            jmeno=`echo "$jmeno" | sed 's/ -> .*$//'`
        fi

        prava=`echo "$prava" | sed 's/^.//'`

        # tady se prava musi prevest na cislo!!!
        pravaN=`numrights "$prava"`
        prava=`echo "$prava" | sed "s/^...//"`
        pravaN="$pravaN"`numrights "$pravap"`
        prava=`echo "$prava" | sed "s/^...//"`
        pravaN="$pravaN"`numrights "$pravap"`

        ##### | pozor - cas se rozhodi na 2
        pole !!!
        echo "$pravaN:$user:$group:$mesic:$den:$cas:$jmeno"

    }
}

# vyzvedne informace o souboru - bere na vstup to, co udelalo save data a
modifikuje vlastnosti souboru
# predpoklada, ze uz jsme v aktualnim adresari u souboru
loaddata() {
    OIFS="$IFS"
    IFS=":"
    {
        read typ prava user group mesic den cash casm jmeno
        ldpath=`getfilepath "$jmeno"`
        ldfile=`getfile "$ldpath"`
        chown "$user" "$ldfile"
        chgrp "$group" "$ldfile"
        chmod "$prava" "$ldfile"
    }
```

```
        # tady to je blbe, ale uz to nestiham...
        touch -d "$mesic $den ${cash}:${casm}"
    }
    IFS="$OIFS"
}

# jako vstup slouzi infolina ze savedata
getfilepath() {
    echo $1 | sed 's/^\([^:]*\)\\{8\\}/'
#echo $1
}

# jako vstup slouzi vystup z getfilename
getdir() {
    echo $1 | sed 's#[^/]*$##'
}

getfile() {
    echo $1 | sed 's#^.*\/\([^/]*\)#$\1#'
}

# zpracovani prepincu a nacteni jmena archivu

action="$1"
archiv="$2"
shift 2

#echo "$action $archiv"

if [ "$action" = '' -o "$archiv" = '' ] ; then
    echo 'Ma se to pouzivat tar [t|x|c] <soubor> [soubory...]'
    exit 0
fi

# struktura pakovaneho souboru - jmeno vctne cesty na zacatku
# typ:skupina:vlastnik:cas:jmeno
#typ - dir      - dal uz pak nic neni - jen se vytvori soubor
#   - file      - nasleduje begin.....end
#   - symlink   - radek s jmenem souboru, kam link ukazuje
#   - hardlink  - radek s jmenem souboru, kam link ukazuje

# a jdeme na to.....
case "$action" in
    tf )
        # vypsani obsahu To by snad melo jit :-)
        while read line ; do
            getfilepath "$line"
            case "$line" in
                file:* )
                    while [ "$line" != "end" ] ; do
                        read line
                    done
                    ;;
                symlink:* | hardlink:* )
                    read line
                    ;;
            esac
        done < "$archiv"
    ;;
```

```

xf )
adir=`pwd`
while read line ; do
    cd "$adir"
    filep=`getfilepath "$line"`
    file=`getfile "$filep"`
    cd `getdir "$filep"`
    case "$line" in
        dir:* )
            #      mkdir $file
            echo "$file"
            ;;
        file:* )
            {
                tmp=""
                while [ "$tmp" != "end" ] ; do
                    read tmp
                    echo $tmp
                done
            } | uudecode -o $file.tmp
            rm -f $file.tmp
            echo "$file"
            ;;
        symlink:* )
            read zdroj
            #      ln -s "$zdroj" "$file"
            echo "$file"
            ;;
        hardlink:* )
            read zdroj
            #      ln "$zdroj" "$file"
            echo "$file"
            ;;
    esac
    loaddata "$line"
done < "$sarchiv"
;;

cf )
find @$ > list.$$
# roztrideni na adresare, softlinky a soubory
while read file ; do
    if [ -d "$file" ] ; then
        echo "$file" >> dirs.$$
    else
        if [ -L $file ] ; then
            echo "$file" >> links.$$
        else
            ls -i "$file" >> unknownn.$$
        fi
    fi
done < list.$$

#rozdeleni na soubory a hardlinky
sort unknownn.$$ -t\  -n -k1 > unknowns.$$
rm -f unknownn.$$ list.$$

lastID=""
lastFile=""
while read ID file ; do

```

```

    if [ "$lastID" = "$ID" ] ; then
        echo "$file" >> hard.$$
        echo "$lastFile" >> hard.$$
    else
        echo "$file" >> file.$$
    fi
    lastID="$ID"
    lastFile="$file"
done < unknowns.$$

rm -f unknowns.$$

# mame to hezky rozdelene, takze zaciname pakovat - asi od adresaru,
soubory, softlinky a hardlinky
dir=`pwd`

{
    while read file ; do
        echo -n "dir:"
        savedata "$file"
    done < dirs.$$

    while read file ; do
        echo -n "file:"
        savedata "$file"
        cat "$file" | uuencode "$file"
    done < file.$$

    while read file ; do
        echo -n "symlink:"
        savedata "$file"
        ls -l "$file" | sed "s/^.* -> //"
    done < links.$$

    while read file ; do
        echo -n "hardlink:"
        savedata "$file"
        read hlink
        echo "$hlink"
    done < hard.$$
} > "$sarchiv"

rm -f hard.$$ links.$$ file.$$ dirs.$$

;;

esac

```

Quota

Zadani

Mate 2 soubory quota.usr a quota.grp ve kterých je

```
jmeno/uid:kvota na domovsky adresar:kvota total
```

v . grp je jmeno nebo GID:dtto;

kvoty jsou na velikost v blocích/bytech
Mate zkontrolovat jestli nekterý uživatel (paktizuje je v quota.usr tak to jsou jeho limity, jestli ne tak podle grup kde je tak nejmenší z nich) nepřekracuje limity, paktizuje jo tak pošlat mějlem kdo-kolik je třeba si dát pozor na hardlinky, simlinky a adresare (ty mají taky nějakou velikost)

Takže quota.usr vypadá třeba takto:
Tomas:20000:15000
14586:20000:15000

Reseni

```
#!/bin/sh
# ----- check par -----
if [ "$#" = 0 -o "$#" -ge 3 ]; then
    echo error par
    exit 1
elif [ "$#" = 1 ]; then
    cislo=$1
    user=`grep "$cislo" /etc/passwd | cut -d: -f5`
    else
        user="$1 $2"
        cislo=`grep "$user" /etc/passwd | cut -d: -f3`
    fi
home=`grep "$cislo" /etc/passwd | cut -d: -f6`
#----- zratam velkost home dir-----
velkost_dir()
{
    ls -alR $home 2>/dev/null>>vypis
    cat vypis | tr -s " ">>vyp
    velkost='0'
    while read line; do
        case $line in
            [dlf-][rwx-][rwx-][rwx-]*)
                vel=`echo $line | cut -d" " -f5`
                velkost=`expr $velkost + $vel`
            esac
        done<vyp
        rm vypis
        rm vyp
    }
}
```

```
#-----prejdeme najprv *.usr -----
```

```
IFS=:
while read meno adr total; do
    if [ "$meno" = "$user" -o "$meno" = "$cislo" ]; then #nasiel som ho v .usr
        velkost_dir
        if [ "$velkost" -ge "$adr" ]; then
            echo "uživatel $user prekročil kvotu $adr -$velkost-"
        else
            echo "uživatel $user nepřekročil kvotu $adr -$velkost-"
        fi
    fi
    exit
fi
done<quota.usr
IFS=" "
#-----tak teda prejdeme *.grp-----
gid=`grep "$cislo" /etc/passwd | cut -d: -f4` #jedno cislo
gname=`grep ":$gid:" /etc/group | cut -d: -f1` #a meno
login=`grep "$cislo" /etc/passwd | cut -d: -f1`
grep "$login" /etc/group | cut -d: -f1,3|sed 's:/ /'>>grupy #ostatne cisla
echo "$gid $gname">>grupy
#----- prejdeme vsetky grupy a najdem minimum-----
velkost_dir
min="1000000000000"
while read c m; do #prechadzam subor skupin
    if { kvota=`grep "$m" quota.grp && grep "$c" quota.grp` } then
        if [ "$kvota" -le "$min" ]; then
            min=$kvota
        fi
    fi
done<grupy
rm grupy
if [ "$velkost" -ge "$min" ]; then
    echo "uživatel $user prekročil kvotu $min -$velkost-"
else
    echo "uživatel $user nepřekročil kvotu $min -$velkost-"
fi
```

Showhint

Zadani

Zadani bylo udelat skript showhint, ktorej dostal jako vstup retezec nebo nic a mel vypsats vsechny spustitelny soubory (podle \$PATH) retezcem zacinajici. Zkratka neco jako dela tabulator pod bashem. Rozdil je v tom, ze se maji kontrolovat prava (pro uzivatele, prim. skupinu a sekundarni skupinu) podle toho jestli je to bin nebo skript. Bylo to docela lehky, reseni se mi veslo ani ne na dve stranky.

Reseni

```
#!/bin/sh
#-----check par-----
if [ "$#" -ge 2 ]; then
    echo error par
    exit 1
fi
if [ "$#" = 1 ]; then
    vzor=$1 #hladane slovo
else
    vzor=" "
fi #----- testovacia funkcia -----
testuj() #ci je to spustitelny subor
{
    adr=$1
    prava=`ls -l $adr | tr -s " " |cut -c4`
    if [ "$prava" = "x" ]; then
        return="0"
    else
        return="5"
    fi
}
#-----
IFS=':' #prehladame vsetky adr z $PATH
for cesta in $PATH; do
    find $cesta -type f -name "$vzor*" 2>/dev/null >>vypis
done
IFS=" "
while read line; do
    if testuj $line; then #ak je to spustitelny subor
        echo $line | sed 's/.*\\/(.*)$/\\1/'>>final
    fi
done<vypis
sort -u final #vypisem bez zdvojeni
#----- uklid-----
rm vypis
rm final
```

Maly skript z veľkého (killall)

Zadani

Caute, dneskajsie zadanie z unixu: Mame 1 skript vytvorit a to tak aby po spusteni sa:

- 1.) vas spytal na cestu kde chcete jeden mensi skript(ktory musi byt nejak ulozeny v tom nasom skripte!) nainstalovat, musite zistit ci dany adresar existuje,ma zapisovacie prava, a ak tam dany subor je tak porovnat datum ak je novsi tak nechat inak tam dat nas
- 2.) zistit aky system pouzivat(UnixV/FreeBSD), bude sa hodit neskoro
- 3.) datum a cas zachovat povodneho skriptu

ten "maly skript by mal"

- 1.) sa spustat prikazom killall [-cislo] ...
to ... je bud wildcart, alebo regexp, zalezi od vas co si vyberiete, to cislo moze nemusi
- 2.) ma zakilovat, resp poslat signal(=tomu cislu) tany procesom, ktorych meno sa zhoduje s danym regexpom/wildcartom.
- 3.) hmmm to je asi vsetko

Takze hinty:

1. do PATHU cestu pridať,
2. nastaviť prístupové práva!!(kvôli tomu som asi aj presiel, ako jeden z malá som to mal)
3. ten subor si uložte napr. pomocou #.... a potom to nejak sed alebo cim tam supnut
4. na vypis mien procesu použiť ps, ale pozor meno nie je vždy v rovnakom stĺpci, a je medzi nimi ľubovoľný počet medzier, t.j. neda sa použiť cut,ale vedeli sme že vždy to je v stĺpci, ktorý je označený v hlavičke ako COMMAND, robí to slo napr. awk, sed, alebo pozrieť OPTIONY a tam sa to vypisovanie da nastaviť.
5. touch -r subor subor1, nastaviť čas subor1, presne taký aký má čas subor
6. ten čas sa da zistiť findom, ci je novsi -newer
7. zistiť ci je to V/BSD, ide napr. aj pomocou ps, bud ps -ef ps -ax, ak si dobre pamätám, jedno ide tam druhé ide tam a roznych iných fint sa tam dalo použiť

Reseni

```
#!/bin/sh
##zaciatok
##-----check par -----
#if [ "$#" -ge 3 -o "$#" = 0 ]; then
#   echo error par
#   exit 1
#fi
#if [ "$#" = 1 ]; then
#   vyraz=$1
#else
#   cislo=$1
#   vyraz=$2
#fi #-----
#ps |tr -s " "| sed 's/[ ]*\([a-zA-Z]*\)/\1/'>vypis
#read line<vypis   #zistim ktory stlpec je cmd
#poc='1'
#for par in $line; do
```

```
# if [ "$par" = "CMD" ]; then
#   break
# fi
#poc=`expr $poc + 1`
#done #-----
#sed 'ld' vypis>vyp #zmazem prvý nepotrebný riadok
#while read line; do #prechádzam vypis ps
#   num=`echo $line | cut -d" " -f1`
#   name=`echo $line | cut -d" " -f$poc`
#   if expr $name : $vyraz >/dev/null ; then
#       echo "$name macuje s $vyraz, killujem ho!!!"
#       if [ "$#" = 1 ]; then
#           kill $num
#       else
#           kill $cislo $num
#       fi
#   fi
#done<vyp
##----uklid---
#rm vypis
#rm vyp
##koniec
sed -n '/^##zaciatok/,/^##koniec/p' main.sh| sed 's/#!/>kil
chmod +x kil
echo "zadajte adresar kam mam uložiť killera"
read adresar
if [ -d "$adresar" ]; then #test ci existuje
    if [ -w "$adresar" ]; then #a ci ma w prava
        echo adresar $adresar má prístupové práva
    else
        echo nemá prístupové práva pre $adresar
        rm kil
        exit 1
    fi
else
    echo robím adresar $adresar
    mkdir $adresar
fi

if [ -f "$adresar/kill_all" ]; then
    echo "killer už existuje, porovnam čas"
    f=`find $adresar -name "kill_all" -newer main.sh`
    if [ "$f" = "$adresar/kill_all" ]; then
        echo je novsi, nechám ho tak
    else
        echo je starsi prepisem ho
        rm "$adresar/kill_all"
        mv "kil" "$adresar/kill_all"
        touch -r main.sh "$adresar/kill_all" #zachovám povodný čas veľkeho
skriptu
    fi
else
    echo "killer tu není..."
    mv "kil" "$adresar/kill_all"
    touch -r main.sh "$adresar/kill_all" #zachovám povodný čas
fi
PATH="$h/$adresar:$PATH" #pripisem cestu do PATH
```

Listserver

Zadani

Nase zadani bylo napsat listserver, neboli skript, do ktereho se rourou sypou dosle maily (to se da nastavit souborem ~/.forward, coz ale neni nas problem :) a který je rozesila clenum mailovych konferenci u nej zaregistrovanych, a zaroven zajistuje zakladni spravu tech konferenci.

Konference ma jmeno, a ma taky spravce. Do konference muzou prispirvat bud jen clenove, nebo kdokoliv, a novi clenove se bud muzou pridavat sami, nebo je musi potvrdit spravce. Taky ma archiv prispevku.

Kdyz prijde mail na adresu jmeno_konference@nas_server, rozesle se vsem clenum konference (pokud to dovoluje nastaveni konference, eventualne identita odesilatele), a zaroven se prida do archivu prispevku.

```
Kdyz prijde mail na adresu listserv@nas_server, musime poslouchat, co se
pise v jeho tele:
subscribe [-h] jmeno_konference
    prida odesilatele do zadane konference. pokud je uzavrena, s pridanim se
    ceká na potvrzeni spravce konference. pokud je tam -h, prida se clovek
jako
    skryty clen - tj. chodi mu prispevky, ale neobjevuje se ve vypisech clenu
unsubscribe {jmeno_konference|*} [adresa]
    odhlasi adresu, nebo odesilatele (neni-li uvedena) bud ze zadane, nebo ze
vsech konferenci
lists
    posle seznam konferenci
view jmeno_konference
    posle seznam neskrytych clenu konference
approve jmeno_konference adresa
    smi posilat jen spravce konference, odsouhlasí tim kandidata do uzavrene
konference
index jmeno_konference
    posle archiv konference
get jmeno_konference id_prispevku
    posle zadany prispevek. format id, stejne jako veskere dalsi datove
reprezentace, je na nas.
```

listserver by mel odpovidat na dosle maily (potvrzeni pridani, odebrani, a podobne), a mel by byt schopny prezít i sulinske mailove adresy v polich >From a To typu: Oskar Schindler schindleruv@seznam.cz

schindleruv@seznam.cz (Oskar Schindler)

To se pak rozlisuje podle posledního znaku. Taky by to melo fakovat lidi, kteri se zapisuji do konfery, kde uz jsou, nebo do neexistujici a podobne. Skript by mel mít nějak osetrene zamykani dat v dobe prepisu, aby se dve instance navzajem nepomlatily.

Reseni

```
#!/bin/sh
```

```
# Listserv
#
# Chybi kontroly ve stylu existuje zadana konference apod.
# Necht si ctenar domysli.
# Diskove adresy by mely byt v uvozovkach.. :o)
```

```
init()      #inicializuje promenne
{
prichozi_mail=/tmp/listserv
dir=/home/luckyboy/listserv/      #adresa listservu
}
```

```
GetDataFromMail()      #Zjisti data z mailu
{
    head=`cat prichozi_mail | sed '/^$/, $d`
    body=`cat prichozi_mail | sed '1,/^$/d`
    command=`echo $body | head -n 1`
    from=`echo $head | grep '[fF]rom: '`
    IFS=" "
    echo $from | read x from
    to=`echo $to | grep '[tT]o: '` | cut -d" " -f2`
    if echo $from | grep '>'; then
        #radek ma tvar Libor Forst <libor@forst.cz>
        IFS=" "
        echo $from | read jmeno prijmeni prichozi_adresa
        prichozi_adresa=`echo $prichozi_adresa | sed 's/[<, >]//g`
    else
        #radek ma tvar libor@forst.cz (Libor Forst)
        IFS=" "
        echo $from | read prichozi_adresa jmeno
        jmeno=`echo $jmeno | sed 's/[ (,)]//g`
        echo $jmeno | read jmeno prijmeni
    fi
}
```

```
subscribe() #parametry: [-h] jmeno_konference
{
if [ "$#" = 2 ] then ;
    jmeno_konference="$2"
    hidden=true
else
    jmeno_konference="$1"
    hidden=false
fi
#zkusi, zda se da do konference volne zapisovat
povoleni=`cat ${dir}/conf/${jmeno_konference}/options | grep authorize |
cut -d"=" -f2`
if [ "$povoleni" = "false" ] ; then
    echo "${prichozi_adresa}:${jmeno}:${prijmeni}:${hidden}" >>
${dir}/conf/${jmeno_konference}/users
    echo "Subscription succesfull" | mail $prichozi_adresa
else
    echo "Administrators approve needed" | mail $prichozi_adresa
    echo "${prichozi_adresa}:${jmeno}:${prijmeni}:${hidden}:disabeled"
>> ${dir}/conf/${jmeno_konference}/users
    adresa_spravce=`cat ${dir}/conf/${jmeno_konference}/options | grep
'^admin' | cut -d"=" -f2`
    echo "Needed approvet to add ${prichozi_adresa} $jmeno $prijmeni
into $jmeno_konference" | mail $adresa_spravce
}
```

```
lists() #posle seznam konferenci
{
. cd ${dir}/conf
ls | mail $prichozi_adresa
}
```



```

view() #posle seznam neskrytych ucastniku konference, parametr jmeno
konference
{
if [ "$#" != "1" ] ; then
    echo "Wrong args" | mail $prichozi_adresa
    rm -f prichozi_mail
}
    exit 1
fi
. cd ${dir}/conf/$1
cat users | grep 'false$' | cut -d":" -f1,2,3 | sed 's:/ /g' | sort | mail
$prichozi_adresa
}

approve() #schvaleni pridani uzivatele, parametry jmeno_konference a adresa
{
if [ "$#" != "2" ] ; then
    echo "Wrong args" | mail $prichozi_adresa
    rm -f prichozi_mail
    exit 1
else
    spravce=`cut ${dir}/conf/$1/options | grep '^admin' | cut -d"=" -f2`
    if [ "$spravce" = "$prichozi_adresa" ] ; then
        ed ${dir}/conf/$1/users <<- ED_END
        g/^$2/ s/:disabeled$//;
        wq
        ED_END
        echo "Approvement succesfull" | mail $prichozi_adresa
    else
        echo "You're not an admin of this conferention." | mail
$prichozi_adresa
        rm -f prichozi_mail
        exit 1
    fi
}

index() #posle archiv konference, parametr je jmeno konference
{
if [ "$#" != "1" ] ; then
    echo "Wrong args" | mail $prichozi_adresa
    rm -f prichozi_mail
    exit 1
fi

ls "${dir}/conf/$1/archiv" | mail $prichozi_adresa
}

get() #odesle archiv konference
    #parametry jmeno_konference a id_prispevku
{
if [ "$#" != "2" ] ; then
    echo "Wrong args" | mail $prichozi_adresa
    rm -f prichozi_mail
    exit 1
fi

cat "${dir}/conf/$1/archiv/$2" | mail $prichozi_adresa
}

unsubscribe() #parametry: {jmeno_konference | *}
{

```

```

if [ "$1" = "*" ] ; then
    find dir | grep 'users$' | sed '/^${prichozi_adresa}:d'
else
    ed ${dir}/conf/$1/users <<- ED_END
    g/^${prichozi_adresa}:/ d
    wq
    ED_END

echo "Unsubscription succesfull" | mail $prichozi_adresa
}

#----- zacatek skriptu -----
while [ -f prichozi_mail ] ; do {sleep 1} ; done
init
>prichozi_mail
. cd dir
GetDataFromMail

if [ "$to" = "listserv" -o "$to" = "Listserv" -o "$to" = "LISTSERV" ] ;
then
    #prisly prikazy

    set $command
    case $0 in
        '[s,S]ubscribe'|'SUBSCRIBE' ) subscribe $* ;;
        '[u,U]nsubscribe'|'UNSUBSCRIBE' ) unsubscribe $* ;;
        '[l,L]ists'|'LISTS' ) lists ;;
        '[v,V]iew'|'VIEW' ) view $1 ;;
        '[a,A]pprove'|'APPROVE' ) approve $* ;;
        '[i,I]ndex'|'INDEX' ) index $1 ;;
        '[g,G]et'|'GET' ) get $* ;;
        * ) { echo "error. Option unknown"; exit 1 } ;;
    esac
else
    #prisel prispevek
    omezeni=`cat ${dir}/conf/${to}/options | grep
'^add_allowed_only_to_users=' | cut -d"=" -f2`
    if grep '^${prichozi_adresa}: ' ${dir}/conf/${to}/users ; then
        clen=true
    else
        clen=false
    fi

    if [ "$Somezeni" = "true" -a "$clen" = false ] ; then
        echo "You haven't permission to add prispevky" | mail
$prichozi_adresa
        rm -f prichozi_mail
        exit 1
    fi

    index=`ls ${dir}/conf/${to}/archiv | sort -n | tail -n 1`
    index=`eval $index + 1`
    echo $body > ${dir}/conf/${to}/archiv/${index}

    echo "prispevek uspesne pridán" | mail $prichozi_adresa
    for each address in `cat ${dir}/conf/${to}/users | cut -d":" -f1 | {
        while read emailova_adresa ; do
            [ "$prichozi_adresa" = "$emailova_adresa" ] && continue ||
echo -n "$emailova_adresa " } `
        do echo "$body" | mail "$address"
    }
fi
rm -f prichozi_mail ; exit 0

```

Archiv

Zadani

napiste program archiv. program sa vola dvoma sposobmi:

1. archiv n (n je cislo od 0 - 9)
n predstavuje tzv. uroven archivace.

Pri "0" sa archivuje cely file system,
pri "1" sa archivuju iba tie subory, ktore boli zmenene
od poslednej archivacie urovne 0 alebo 1
(podla toho, ktora je mladsia, tj bola provedena naposledy), pri
archivacii stupna
"2" sa archivuju iba tie, kt. boli zmenene od poslednej 2, 1 alebo 0 (takze
3 nas nezaujima), atd...
Kratko pred archivaciou sa vytlaci hlaska
"Vloz do mechaniky magneticku pasku cislo x.y" , kde x je cislo
urovne a y cislo
pasky od 1 po cislo_ktore_je_ uvedene_v_configuraku menom archiv.cfg.
Uzivatel ma nakoniec potvrdit ENTEROM.

Samotnu archiváciu zaisti ista, blizsie nespecifikovana procedura
"cpio" ktorej to staci
nasypat na vstup a ona to na tie pasky nahra. Takze tak.

2. druhy sposob volania programu je:
archiv datum soubor1 [soubor2...(potencialne az velmi vela:)]
ulohou je vypisat, na ktorych paskach je dany subor, archivovany niekedy po
dany_datum az po dnes. Tot vsjo.

Takze riesenie Forstik nacrtol (samozrejme po pisomke) zhruba take, ze vzdy
urobit timestamp niekam,
aby sme vedeli key bola robena archivacia, ktorej urovne a ktore subory
boli archivovane
(aby sme mohli vyhľadavat - lze prochazet pro kazdy subor zhora dolu). Jo a
jeste rikal ze prý je mu
to velmi luto, ze musel uz druhykrat vyhodit viac ludi nez pustit, pretoze
"skuska" prebiehala tak,
ze styria odisli v priebehu prvej hodinky, po vyhodnoteni to dvom hned
zapisal, styroch rovno vyhodil
a z vysnych styroch si pozval na pokec.

Zaver: nieje to take strasne tazke, hlavne riadit sa zasadou VELA KODU -
MALO SLOHU, aj na to niekteri
dnes doplatili, dalej co najmenej syntaktickych chyb, ako napr
if [\$n < 4] (ma byt [\$n -lt 4]), dosledne sed-ovat, vsetky esac, fi a
done atd.
Vela stastia!!!

Reseni

```
#!/bin/sh
# Program Archiv
# V adresari /archives maji soubory tvar
#          StupenArchivace_Den_Mesic_Rok_CisloPasky

init()
{
    conf='/home/luckyboy/archiv/archiv.cfg'
    archives='/home/luckyboy/archives'
    den=`date +%d`
    mesic=`date +%m`
    rok=`date +%Y`
    nejmladsi_archivace=''
    cislo_pasky=''
}

najdi_nejmladsi_archivaci() # maximalni uroven je parametr
{
    i="1"
    find archives * 2>/dev/null | {
        while read soubor ; do
            if [ "$1" -lq "`echo $soubor | cut -d"_" -f1`" ] ; then
                if [ "$i" -eq "1" ] ; then
                    nejmladsi_archivace=$soubor
                    i="2"
                fi
            else
                if [ $soubor -nt $nejmladsi_archivace ] ; then
                    nejmladsi_archivace=$soubor
                fi
            fi
        done
    }
}
```

```
##### ----- archivace -----
# jedyiny parametr je stupen archivace 0-9
archivace()
{
    cislo_pasky=`head -1 "$conf"`
    cislo_pasky=`expr $cislo_pasky + 1`

    if [ "$1" -eg "0" ] ; then
        echo "The whole filesystem will be archived!!"
        echo "Stupen archivace: ${1}, Cislo pasky: ${cislo_pasky}"
        echo "Proceed? (press ENTER or type 'n' to cancel)"
        read enter < /dev/tty
        if [ "$enter" = "" ] then ;
            find / 2>/dev/null | tee
            "${archives}/0_${den}_${mesic}_${rok}_${cislo_pasky}" #|cpio
                                #postara se o archivaci
            echo "Operation succesfull"
            echo $cislo_pasky > conf
        else
            echo "Operation canceled"
            exit 0
        fi
    else
        najdi_nejmladsi_archivaci $1
        echo "Proceed with archiving??"
        echo "Stupen archivace: ${1}, Cislo pasky: ${cislo_pasky}"
        echo "(press ENTER or type 'n' to cancel)"
        read enter < /dev/tty
        if [ "$enter" = "" ] then ;
            find -newer nejmladsi_archivace / * 2>/dev/null | tee
            "${archives}/0_${den}_${mesic}_${rok}_${cislo_pasky}" #|cpio
            echo "Operation succesfull"                                #postara se o
        archivaci
            echo $cislo_pasky > conf
        else
            echo "Operation canceled"
        fi
    fi
}
}
```

```
##### ----- vypis -----
# predpoklada parametry {datum} {soubor1} soubor2 soubor3 ...
# datum se predpoklada ve tvaru dd.mm.yyyy
# soubory maji plnou ardesu (napr. '/home/hehe/auticka.jpg')
vypis()
{
    IFS="."
    echo $1 | read den mesic rok
    shift 1
    IFS=" "
        # vytvorime si pomocny soubor a nastavime mu modifikacni
        # cas na pozadovane datum. S timto souborem srovnavame
        # ve findu.

    touch -m ${rok}${mesic}${den}0000.00 /tmp/pomocny_soubor
        # tvar datumu je YYYYMMDDhhmm.ss

    for file in $* ; do
        if [ "$file" -nt "/tmp/pomocny_soubor" ] ; then
            find -newer "/tmp/pomocny_soubor" "$archives" * 2>/dev/null | {
                IFS=" "
                read soubor
                if grep "$file" "$soubor" ; then
                    IFS=" "
                    echo $file | read fstupen fden fmesic frok fpaska
                    echo "soubor $soubor"
                    echo "byl archivovan ${fden}.${fmesic}.${frok}"
                    echo "archivace stupne: ${fstupen}, paska: ${paska}"
                fi
            }
            IFS=" "
        fi
    done
    rm -f /tmp/pomocny_soubor
}
}
```

#####----- begin of the script

```
init
if [ "$#" -eq "0" ] ; then
    echo "parameters error"
    exit 1
fi
if grep '^[0-9]${1}' $1 ; then
    archivace "$1"
fi
if [ "$#" -eq "1" ] ; then
    echo "parameters error"
    exit 1
else
    vypis $*
fi
exit 0
```