UNIXi käsurida

Koostanud: Indrek Zolk

Dokumendi versioon: 5
Kompileerimise kuupäev: 25. september 2005. a.

Tartu Kunstigümnaasium
Tartu 2005

Sisukord

Sissejuhatus

Richard Stallmani algatusel 1970-ndate lõpus levima hakanud ideoloogia, mille kohaselt tarkvara peab olema kõigi jaoks kättesaadav koos lähtekoodiga ning vabalt modifitseeritav (seda toetav organisatsioon *Free Software Foundation*[1] loodi 1984), on tänaseks katnud avatud lähtekoodiga tarkvaraga pea kõik eluvaldkonnad. Litsenseerimistingimustes on tavaliselt (*GNU General Public Licence*[2] 1988) lubatud vabalt levitada programmi, tingimusel, et saajal on vaba ligipääs programmi lähtekoodile. Samuti on lubatud programmi lähtekoodi vabalt modifitseerida ning muudetud kujul levitada, tingimusel, et lähtekoodis on selgelt fikseeritav, kes ja mida täpselt muutnud on.

Taolise **avatud tarkvara** eelisteks on kõrgem turvalisus ja töökindlus (turvaauke saavad otsida kõik maailma programmeerijad, mitte ainult väljatöötav firma), kiiremad täiendused ja veaparandused (kuivõrd vigu parandavad entusiastid üle maailma), suurem modulaarsus (kasutajad võivad lähtekoodi „tükke“ kasutada oma programmides), kindlus „tagauste“ puudumise suhtes (kuna lähtekoodi näevad kõik) ning odavus (enamasti on avatud tarkvara suisa tasuta või minimaalsete kuludega kättesaadav), puuduseks aga väiksemad sissetulekud tarkvarafirmadele (kuna lähtekood on vabalt kättesaadav, pole paljude jaoks mõttekas toodet poest osta, vaid see ise lähtekoodist kompileerida). Siiski paistab, et läbipaistvus, odavus ja suurem potentsiaal kvaliteetseks saada on põhiargumendid, miks avatud lähtekoodiga tarkvara eelistatakse eeskätt turvakriitiliste rakenduste juures, aga viimasel ajal ka mitmetes ettevõtetes ja haridusasutustes. Tarkvaraettevõtted aga kasutavad kombineeritud lähenemist (arendusvariant avatud koodiga ja tasuta, põhivariant suletud koodiga ja tasuline) või panevad suuremat rõhku tasulistele tugiteenustele.

Nii ongi käesoleval ajal paljudes kohtades tüüplahenduseks: serverid ja mõned tööjaamad kasutavad avatud lähtekoodiga operatsioonisüsteeme ja tarkvara, töökohaarvutites on suurem osa tarkvarast suletud koodiga.

Käesolevas õppevahendis vaadeldakse tööd UNIX-tüüpi operatsioonisüsteemide käsureal. Enamik taolistest süsteemidest (Linux, FreeBSD, OpenBSD jt.) on avatud lähtekoodiga. Tuleb kohe öelda, et UNIX-tüüpi operatsioonisüsteemid ei paku kaugeltki ainult käsurea võimalusi; piisab näiteks mainida graafilist liidest X.org ning töökeskkondi Gnome ja KDE, millele leiab kaasajal väga palju avatud tarkvara. Käsurea eelisteks graafiliste keskkondade ees on aga töökindlus, väljendustäpsus ja võrgutöö peaaegu mistahes kiirusega ühenduse korral.

[1] Avatud Tarkvara Fond

[2] GNU Üldine Avalik Litsents

PEATÜKK 1

Üldpõhimõtted

1.1 Meldimine serverisse

1.1.1 Windows

1. Otsi töölaualt ikoon pealkirjaga „SSH Secure Shell Client“ ning tee sellel topeltklõps.
2. Avaneb valge aken, üleval on paar nupurida. Vajuta tühikuklahvi.
3. Sisesta esimesse kasti `home.tkug.tartu.ee` ning teise kasti oma kasutajatunnus. Vajuta nupule „Connect“.
4. Kui tuleb ette aken, kus küsitakse kinnitust, vajuta „Yes“. See dialoogiaken tuleb ainult esimest korda ning on seotud SSH protokolli võtmevahetusega.
5. Ilmuvasse parooliaknasse sisesta oma parool ja vajuta „OK“.
6. Oled jõudnud serveri käsureale. (Jäta meelde! Siit väljutakse käsurea käsu `exit` abil. Igaks juhuks proovi ühendus sulgeda ja veel kord meldida.)

1.1.2 KDE või Gnome

1. Otsi tööriistaribalt ikoon pealkirjaga „Terminal“ ning tee sellel üks hiireklõps.
2. Avaneb must aken. Sisesta käsk
   `ssh kasutajatunnus@home.tkug.tartu.ee`
   ning vajuta sisestusklahvile.
3. Kui küsitakse võtmevahetust ning valida saab `yes/no`, siis tipi `yes` ja vajuta sisestusklahvile. See dialoog tuleb ainult esimest korda.
4. Sisesta parool ja vajuta sisestusklahvile.
5. Oled jõudnud serveri käsureale. (Jäta meelde! Siit väljutakse käsurea käsu `exit` abil. Igaks juhuks proovi ühendus sulgeda ja veel kord meldida.)

1.2 Failisüsteem

1.2.1 Failisüsteemi objektide liigid

Failisüsteem on organiseeritud **kataloogide** (*directory*) ehk **kaustade** (*folder*) **kataloogipuuna**[1]. Igas failisüsteemis leidub puu juur, nn. **juurkataloog** (*root directory*), sellel **alamkataloogid** (*subdirectories*), nendel omakorda alamkataloogid jne. Alamkataloogi jaoks kõrgema taseme kataloogi nimetatakse **ülemkataloogiks** (*parent directory*). Juurkataloogi tähiseks on `/` ja ülemkataloogi tähiseks on kaks punkti `..`

UNIX-tüüpi süsteemides kettatähti ei ole[2]. Faili- ja kataloginimed on suhteliselt vabalt valitavad, suur- ja väiketähti eristatakse. Tühikuid sisaldavate nimede kasutamiseks tuleb nad paigutada jutumärkidesse. Punktiga algavad nimed tähistavad „peidetud“ faile.

Faili- ja kataloginimedes tuleks ühilduvuse huvides kasutada vaid ladina tähestiku tähti ja numbreid ning hoida nimed nii lühikesed kui võimalik. Vältida tuleb kõikvõimalikke erisümboleid, kirjavahemärke, täpitähti, muude tähestike tähti jmt.

Kõigil failidel ja kataloogidel on omanik ja juurdepääsuõigused.

Käsureal olles on teatud kataloog alati see, milles ollakse n.ö. „sees“, s.t. millest lähtuvalt toimivad kõik failioperatsioonid. Seda kataloogi nimetatakse **jooksvaks** kataloogiks (*current directory*). Jooksva kataloogi tähiseks on üks punkt `.`

Igale kasutajale määratakse kindel „lähtepunkt“ failisüsteemis, mida nimetatakse **kodukataloogiks** (*home directory*). Kodukataloogi tähiseks on `~` ning tavaliselt viitab see kataloogile `/home/kasutajanimi`.

Teeks (*path*) nimetatakse kirjeldust mingi failisüsteemi objekti poole pöördumiseks. Kataloogide vahel on tees kaldkriips `/`. **Absoluutne tee** algab juurkataloogist, **relatiivne tee** aga jooksvast kataloogist.

Näiteks:
`/usr/local/lib` (absoluutne)
`~/Mail` (kodukataloogist lähtuv)
`./bin` (jooksvast kataloogist alla)
`../../programs` (kaks korda ülespoole)

Failid jaotuvad **tekstifailideks** (*text*) ja **kahendfailideks** (*binary*). Tekstifaile suudab inimene põhimõtteliselt vahetult lugeda, kahendfailid sisaldavad ainult arvutile arusaadavaid märke.

Seadmed on erilised failid `/dev` kataloogis. Seadmefailide abil toimuvad välisseadmete sisend-väljundoperatsioonid.

Näiteks `/dev/audio` — heliseade; sinna info saatmisel jõuab see helikaardile.

[1] 1. ja 2. peatüki teooria-osa materjal on suures osas pärit veebiaadressilt
`http://math.ut.ee/~jaanus/tjkursus/`

[2] Seadmetele ligipääsuks kasutatakse kataloogis `/dev` asuvaid „seadmefaile“, näiteks disketiseadet tähistab tavaliselt `/dev/floppy` ja laserplaadiseadet `/dev/cdrom` jne.

Viidad (*link*) annavad võimaluse ühe objekti poole pöörduda mitmest kohast. Viidad jaotatakse „kõvadeks“ (*hard*) ja „pehmeteks“ ehk sümbolviitadeks (*soft link*). „Kõvad“ viidad toimivad failisüsteemi blokkide, sümbolviidad aga failinimede järgi.

Ülesanne 1. Lahenda ülesanne veebilehelt
`http://www.cs.ut.ee/~taremaa/tjpraks/harjutus/`

1.2.2 Failisüsteemi objektide juurdepääsuõigused

UNIXi failisüsteemis kasutatakse järgmisi põhiõigusi:

- `r` *read*: objekti lugemine
- `w` *write*: objekti muutmine
- `x` *execute*: käsufaili või programmi käivitamine, faili otsimine kataloogist (lehitsemine)

Näiteks kui kataloogi jaoks pole lugemisõigust, aga on otsimisõigus, siis saab selle kataloogi failidele ligi, kui teada faili täpset nime.

Need kolm õigust määratakse iga taseme jaoks (kasutaja – *user*, grupp – *group* ning teised – *others*).

Õiguste näitamiseks on kasutusel 10-märgiline esitus, õiguste muutmiseks saab kasutada kaheksandkoodi ning sümbolesitust. Kõik nimetatud viisid on kokku võetud järgmises tabelis:

`-r--------` `400` `u+r` — omaniku lugemisõigus
`--w-------` `200` `u+w` — omaniku kirjutamisõigus
`---x------` `100` `u+x` — omaniku käivitamisõigus
`----r-----` `040` `g+r` — grupi lugemisõigus
`-----w----` `020` `g+w` — grupi kirjutamisõigus
`------x---` `010` `g+x` — grupi käivitamisõigus
`-------r--` `004` `o+r` — teiste lugemisõigus
`--------w-` `002` `o+w` — teiste kirjutamisõigus
`---------x` `001` `o+x` — teiste käivitamisõigus

Tegeliku komplekti saamiseks tuleb sellest tabelist liita vajalikud read. Näiteks:

`drwxr-x---` on kataloog, millele omanikul on kõik õigused, grupil on lugemise ja otsimise õigus, teistel pole mingeid õigusi. Vastav kaheksandkood on `750`.

Esimene märk näitab faili tüüpi (`d` on kataloog, `-` tavaline fail, `l` viit, `b` või `c` seade jne).

1.2.3 Erisümbolite moodustamine

Reavahetuse (*newline*) tähiseks on `\n`. Tabulaatorisümboli (*tab*) tähiseks on `\t`.

Paljudel juhtudel on mitmesugustel sümbolitel (`* . $ | ^ "` jne.) eritähendus, mistõttu „päris“ tärni, punkti jne. saamiseks tuleb vastav sümbol **varjestada** langjoonega. Teisi sõnu, eritähenduseta kujud on vastavalt `\* \. \$ \| \^ \"` jne.

PEATÜKK 2

Levinumaid UNIXi käsurea käske

2.1 Käsu üldkuju

Iga käsurea käsk koosneb kolmest osast:

käsunimi lipud parameetrid

Käsunimi on käsku tähistav sõna. Näiteks: `ls`, `cp`, `rm` jne. *Lippude* abil täpsustatakse käsu tähendust, *parameetrite* abil öeldakse, millele seda käsku rakendada.

Lipud jagatakse **lühikesteks** (näiteks `-c`) ja **pikkadeks** (näiteks `--create`). Näide:

```
ls -l /usr
```

Siin `ls` on käsunimi, `-l` on (lühike) lipp ja `/usr` on parameeter.

Käskude **üldkujusid** loetakse järgmiselt:

1) tavakeelsete tekstide asemel kirjutatakse vajalikud suurused. Näiteks *failinimi* asemele kirjutatakse faili nimi, millega tegeldakse;
2) nurksulgudes avaldisi peetakse mittekohustuslikeks, kui aga taolisi avaldisi kasutatakse, siis ilma nurksulgudeta;
3) püstkriips `[a | b]` või `{a | b}` tähistab välistavat valikut, s.t. valida tuleb kas `a` või `b`, aga mitte mõlemad.

Käske täidetakse nii, et sisestatakse käsk ja vajutatakse sisestusklahvile. Kui käsu täitmine on pooleli (või blokeerunud), on võimalik käsu täitmine peatada juhtklahvi ja C vajutamisega.

2.1.1 Sageli esinevad lipud

`-l`	*long*, pikk listinguvorm
`-f` *failinimi*	sisendfail
`-o` *failinimi*	väljundfail (*output*)

`-i` *interactive*, töö dialoogis; mõnikord *ignore*
`-f` *force*, nõusolekut mitte küsida
`-R` või `-r` *recursive*, rakendada rekursiivselt alampuule (s.t. järjest kõigile alamkataloogidele); mõnikord *read* (lugemine)
`-a` *all*, kõik (ka need, mida vaikimisi ei vaadelda)

2.2 Käsud juhendite vaatamiseks

- `man` — käsu kohta täisinfo saamine. Juhendid on organiseeritud peatükkide kaupa.

`man` *`käsunimi`*	info käsu kohta
`man` *`võtmesõna`*	info võtmesõna järgi
`man -f` *`failinimi`*	info süsteemifaili kohta
`man -s` *`ptknr käsunimi`*	info käsu kohta koos juhendi peatüki nr. äranäitamisega

- `whatis` *`käsunimi`* — käsu kohta lühiinfo saamine
- `apropos` *`võtmesõna`* — info otsimine võtmesõna järgi
- `whereis` *`käsunimi`* — käsuga seotud juhendifailide otsimine

2.3 Käsud tööks kataloogide ja failidega

- `ls` — kataloogi loetelu näitamine

  `ls [-alrtuRd] [` *`kataloog`* `... ]`

  `-a` *all*, näidatakse kõik (ka peidetud) failid
  `-l` *long*, kuvatakse pikk listinguvorm
  `-r` *reverse*, sorteeritakse tagurpidi järjekorras
  `-t` *time*, sorteeritakse viimase muutmise aja järgi
  `-u` *used*, sorteeritakse viimase kasutamise aja järgi
  `-R` *recursive*, rekursiivselt üle kogu alampuu (s.t. järjest kõigile alamkataloogidele)
  `-d` *directory*, listingus kataloogide nimed nende sisu asemel

- `pwd` — jooksva kataloogi nime näitamine (*print working directory*)
- `cd [`*`kataloog`*`]` — jooksva kataloogi muutmine (ilma parameetrita minnakse kodukataloogi)
- `cat` *`failinimi`* — faili väljastamine
- `more` *`failinimi`* — faili kuvamine peatustega

tühik – uus lehekülg	`q` – väljuda käsureale
reavahetus – uus rida	*/sõna* – otsimine
`b` – lehekülje võrra tagasi	`h` – abiinfo

- `mkdir` *`kataloog`* — uue kataloogi loomine (eeldab vähemalt `-wx` õigusi ülemkataloogis)
- `rmdir` *`kataloog`* — kataloogi kustutamine (eemaldab ainult tühja kataloogi; mittetühja kataloogi eemaldamiseks `rm -r`)
- `cp` *`objekt sihtkoht`* — faili või kataloogi kopeerimine
- `mv` *`objekt sihtkoht`* — faili või kataloogi tõstmine
- `rm [-rfi]` *`objekt`* — faili(de) või kataloogi(de) kustutamine (`-r` eemaldab koos kõigi failide ja alamkataloogidega ehk *rekursiivselt*: OHTLIK!!!)
- `ln [-s]` *`objekt [viidanimi]`* — viida loomine; `-s` loob sümbolviida; kui *`viidanimi`* puudub, luuakse viit jooksvasse kataloogi sama nimega mis *`objekt`*
- `which` *`käsunimi`* — käsu asukoha täpsustamine (millises kataloogis paikneb)
- `chmod` *`õigused objekt`* — failile või kataloogile soovitud õiguste andmine

`u`	*user*	`+`	anda õigus	`r`	*read*
`g`	*group*	`-`	võtta õigus ära	`w`	*write*
`o`	*others*			`x`	*execute*
`a`	*all* (sama, mis `ugo`)				

  Näiteks
  `chmod a+rx lecture` — objektile `lecture` antakse juurde kõigi jaoks lugemis- ja käivitamisõigus
  `chmod 750 lecture` — objektile `lecture` antakse täpselt õigused `rwxr-x---`
- `joe` *`failinimi`* — tekstifaili redigeerimine
- `nano` *`failinimi`* — tekstifaili redigeerimine
- `pico` *`failinimi`* — tekstifaili redigeerimine

2.3.1 Metamärgid

Enne käsu täitmist asendab käsuinterpretaator kõik käsureal olevad erisümbolid vastavalt nende tähendusele.

Metamärgid `* ? [ ]` on kasutusel failinimede genereerimiseks:

***** väljendab mistahes märgijada (s.h. tühja)

? väljendab mistahes (täpselt üht) märki

[märgid] lubab valida ühe nurksulgudes olevatest märkidest

Ülesanne 2. Loo jooksvasse kataloogi failid `kiri1 kiri2 kiri3 kiri4 kiri10 Ptk4.txt ptk5.txt .login`

Tegelikult piisab luua üks neist failidest ja see kopeerida ülejäänuteks.

Vaata failide loetelu erinevate lippude abil.

Millised failid vastavad järgnevatele metaavaldistele?

```
*
kiri?
kiri*
kiri[2-4]
[Pp]tk*
*.*
.*
```

Ülesanne 3.

a) Kuidas saada suurtähtedega algavate failide loetelu?

b) Kuidas saada numbreid sisaldavate nimedega failide loetelu?

c) Kuidas teada saada, millises kataloogis paikneb käsk `lynx`?

d) Loo oma kodukataloogi alamkataloog `unixi asjad` ning kopeeri sellesse failid laiendiga `.txt` kataloogist `/home/zolki/public_html/unix05s/`

e) Lehitse neid tekste käsu `more` abil.

f) Muuda need failid loetavaks kõigile, aga kirjutatavaks ainult iseendale.

g) Muuda kataloog `unixi asjad` kirjutatavaks grupile (tunni lõppedes võid tagasi muuta).

h) Kuidas kustutada eelpoolloodud faile ja katalooge (ei pea kustutama, kui ei taha)?

i) Loo sümbolviit kataloogile `/home/zolki/public_html/unix05s/`

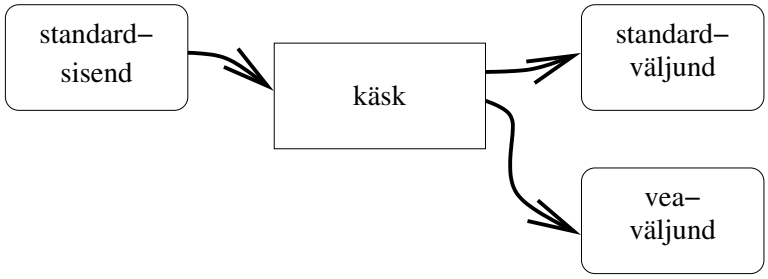
2.4 Sisend ja väljundid

Üldjuhul töötleb käsk mingeid sisendandmeid ning väljastab tulemuse ja võimalikud veateated väljunditesse.

Standardsisendiks (0, stdin) on vaikimisi klaviatuur, **standardväljundiks** (1, stdout) ja **veaväljundiks** (2, stderr) vaikimisi käsuaken. Sisendit ja väljundeid saab ümber suunata.

Käsk võib oma tööks vajaliku informatsiooni saada faili asemel standardsisendist. Näiteks kui soovime kuvada kõigi serveris leiduvate kataloogide loetelu pikas vormingus, kirjutame

`ls -lR /`. Kui aga soovime seda kuvada lehekülgede kaupa (ehk „anda ette“ käsule `more`), kirjutame `ls -lR / | more`
Siin käsk `ls -lR /` paiskab tulemuse standardväljundisse, aga see on toru abil suunatud käsule `more` standardsisendisse.
Täpsemalt kirjeldab käskude sisendi- ja väljundiga opereerimist järgmine skeem ja loetelu:



- `>` või `1>` — suunab standardväljundi faili
- `|` — suunab standardväljundi järgmise käsu standardsisendiks (nn. konveier, toru ehk *pipe*)
- `>>` — suunab standardväljundi olemasolevale failile lõppu juurde
- `2>` — suunab veaväljundi faili

Ettevaatlik tuleb olla operaatorite `>` ja `>>` segiajamisel. Operaator `>` kustutab sihtfaili vana sisu, `>>` aga jätab vana sisu alles.

Vabalt valitud teksti suunamisel on vaja käsku `echo`.

`echo` *`tekst`* — saadab antud *teksti* väljundisse.

Näiteks
`ls > listing` — käsu `ls` täitmise tulemus suunatakse faili `listing` (faili vana sisu hävib)
`echo Viimane rida! >> listing` — faili `listing` lõppu suunatakse tekst `Viimane rida!`
`ls -al | more` — jooksva kataloogi kõik failid pikas listingus lehekülgede kaupa
`ls >> listingud` — käsu `ls` täitmise tulemus suunatakse faili `listingud` lõppu juurde
`find * 2> /dev/null` — käsu `find *` täitmisel tekkida võivad veateated kaotatakse (suunatakse nn. *tühiseadmele* `/dev/null`)

Ülesanne 4.

a) Suuna käsu `ps -A` (näidata kõiki protsesse) väljund faili `ps-log`.

b) Anna käsk, mis väljastab faili `ps-log` standardväljundisse ning suuna tulemus käsule `more`. Näed väljundit lehekülgede kaupa.

c) Mine juurkataloogi, anna käsk `find` ning suuna standardväljund seadmele `/dev/null`. Suuna veaväljund oma kodukataloogi faili `find-errs`.

2.5 Ridade otsimine failist

2.5.1 Regulaaravaldise mõiste

Regulaaravaldis (*regular expression*) kirjeldab mustrit sõnetöötluses (näiteks otsimis- ja asendusoperatsioonides). Regulaaravaldise all mõistetakse teatavat üldistatud üleskirjutust, mis jagab kõik sõned kahte klassi: need, mis **klapivad** (*match*) selle regulaaravaldisega, ja need, mis ei klapi.

`.` punkt — tähistab suvalist märki, välja arvatud uuele reale üleminekut (*newline*)
`^` katus — sõne või rea algus
`$` dollar — sõne või rea lõpp

Kõiki erimärke (ülal loetutele on neid lisaks veel) tuleb varjestada langjoonega, kui soovitakse nende eritähendust kaotada.

Mõned näited:

`v.i` kolmest märgist koosnev kombinatsioon, esimene on `v`, teine suvaline, mis pole reavahetus, kolmas täht `i`. seega klapivad `või`, `v i`, `võimas`, `koorevõipakend`, `viin`, ei klapi aga `vi`, `vali`
`^h` `h` rea alguses
`^http` `http` rea alguses
`Ots$` `Ots` rea lõpus
`^$` tühi rida, algusele peab järgnema kohe lõpp
`^\.$` täpselt ühest punktist koosnev rida

Eriline tähendus on nurksulgudes `[ ]` avaldisel. Selles toodud märkide loetelu tähendab, et sellesse kohta sobib täpselt üks ükskõik milline loetelu märk. Miinusmärk – nurksulgude sees märgib ära, et sobib kogu vahemik.

Näited:

`[arh]` klapib, kui reas on märgid `a` või `r` või `h`
`[a-z]` klapib, kui reas on suvaline väiketäht (`a`-st kuni `z`-ni)
`[a-zA-Z]` klapib, kui reas on mingi täht

2.5.2 Käsk `grep`

`grep [-`*`lipud`*`] '`*`regexp`*`' [`*`failinimi`*` ... ]`
otsib tekstifaili(de)st read, mis klapivad antud regulaaravaldisega *`regexp`*.

`-v` vastupidine, s.t. väljastada read, mis EI klapi regulaaravaldisega
`-c` ridade asemel näidata ainult nende arv
`-n` näidata ka reanumbreid
`-i` *ignore case* — lugeda suur- ja väiketähed samaks

Näited:

`grep -i 'Peeter' aruanne*` — väljastada kõigist failidest, mille nimi algab `aruanne`, kõik read, milles esineb `Peeter`, `peeTER` jne.

`grep -c '\.' tekst` — väljastada faili `tekst` punkti sisaldavate ridade arv (NB! mida teeb `grep -c '.' tekst` ?)

Ülesanne 5. Too näiteid sõnedest, mis klapivad antud regulaaravaldisega:

Regulaaravaldis	Näited
`^.....`	
`[Tt]he`	
`[bmBM]other`	
`[09]A`	
`\^`	
`^\^`	

Ülesanne 6. Otsi failist `/etc/passwd` välja oma kasutajakirje. Kuidas leida sellest failist kõik kasutajad, kelle kasutajanimes või nimes esineks `salu`? Kuidas leida sellest failist kõik kasutajad, kelle kasutajatunnus algab sõnega `ann`?

Ülesanne 7. Koosta tekstifail, mille ridade alguses ja lõpus esineb märke `$`, `^`, `.`, milles on tühje ridu ja sarnaseid ridu.

a) Kuidas käsu `grep` abil leida read, mille esimene märk on `$`?

b) Kuidas käsu `grep` abil leida read, mille viimane märk on `^`?

c) Kuidas käsu `grep` abil leida read, milles esineb punkt `.`?

d) Mil viisil saab leida failis tühjade (*empty*) ridade arvu?

Ülesanne 8. (*Edinburghi ülikool*) Kopeeri endale fail `/home/zolki/public_html/unix05s/mix.txt` ning proovi järele käsud veebilehelt `http://www.ling.ed.ac.uk/facilities/help/unix/workbook.shtml#mix` Failis on segamini läinud luuletus, toiduainete loend ja rahvatarkused, mis tuleks korda teha. Moodusta kolm faili `poem.txt`, `list.txt` ja `sayings.txt` veebilehel saadud parima tulemuse põhjal.

2.6 Failide otsimine

Failisüsteemi objektide otsimiseks kasutatakse käsku `find`.

`find` *`kataloogid tingimused`*

Parameetriga *`kataloogid`* määratakse kataloog(id), millest otsida tingimusi rahuldavaid faile. Meil peab olema nendes kataloogides lehitsemisõigus.

Tingimuste loetelu:

`-name` *`failinimi`*	konkreetse nimega
`-name` *`'metaavaldis'`*	metamärkides (`*`, `?`, nurksulud) antud nimega
`-type [fdbcl]`	tüüp: `f` – tavaline fail, `d` – kataloog, `b` ja `c` – seadmed, `l` – viit
`-perm` *`kaheksandkood`*	õigused
`-user` *`nimi`*	omanik
`-size` *`arv`*`c`	suurus baitides
`-mtime` *`arv`*	sisumuutuse aeg (*modification time*) *`arv`* päeva tagasi
`-exec` *`käsk`* `\;`	täita käsk, jooksva faili nimeks kasutada `{}`
`-ok` *`käsk`* `\;`	täita käsk loa küsimisega, jooksva faili nimeks kasutada `{}`

Arvuliste argumentide tõlgendus:

`+n` rohkem kui `n` `-n` vähem kui `n` `n` täpselt `n`

Näited:

`find . -name 'a*'` — väljastada jooksvast kataloogist ja altpoolt kõik objektid, mille nimi algab tähega `a`

`find / -perm 666` — väljastada alates juurkataloogist kõik objektid, mille õiguste komplekt on `rw-rw-rw`

`find / -size +10000c -mtime +5 -exec rm {} \;` — alates juurkataloogist kustutada loa küsimisega kõik objektid, mis on suuremad kui 10000 baiti ja mille sisu pole viimase 5 päeva jooksul muudetud

Ülesanne 9.

a) Leia käsku `find` kasutades failid, mida on muudetud viimase nädala jooksul.

b) Leia oma kodukataloogist failid, mis on suuremad kui 2 kB.

c) Tee oma kodukataloogi paar nullsuurusega faili (`touch failinimi`). Anna käsku `find` kasutades korraldus nullsuurusega failide õigusteks anda 644.

d) Kustuta käsku `find` kasutades kõik failid, mille suurus on 0 (NB! kustutamisel on loa küsimine mõistlik!).

e) Leia kataloogist `/var` kõik failid, mida on muudetud täna, eile, 2, 3, jne. päeva tagasi. Vaatle nende failide listingut pikas vormingus (`ls -ld`).

f) Otsi oma kodukataloogist üles kõik failid, mille laiendiks on `.txt`. Vaatle nende failide listingut pikas vormingus (`ls -ld`).

g) Otsi oma kodukataloogist üles kõik failid, mille nimi sisaldab Sinu eesnime.

h) Otsi oma kodukataloogist üles kõik kataloogid.

i) Leia kataloogist `/usr/bin` failid, mis on õiguste komplektiga 755. Väljasta nende listing pikas vormingus (`ls -ld`) lehekülgede kaupa (s.t. suuna tulemus toru abil käsule `more` standardsisendisse).

2.7 Arhiveerimine ja pakkimine

2.7.1 Arhivaator `tar`

`tar` — failidest ja kataloogidest arhiivi moodustamine (*tape archiver*)

`tar [-ctx] -f arhiivi_nimi arhiveeritavad_failid`

`-c`	arhiivi loomine
`-t`	arhiivi listingu vaatamine
`-x`	dearhiveerimine
`-f arhiivi_nimi`	arhiivifaili nime määramine

Näiteks:

`tar -c -f arhiiv.tar *`	kõik failid jooksvas kataloogis ja selle alamkataloogides arhiveeritakse faili `arhiiv.tar`
`tar -t -f arhiiv.tar`	vaatame, millised failid on arhiivis `arhiiv.tar`
`tar -x -f arhiiv.tar`	dearhiveerib arhiivi `arhiiv.tar` jooksvasse kataloogi

2.7.2 Pakkijad `zip`, GNU `zip` ja `bzip2`

NB! Pakkijad GNU `zip` ja `bzip2` pakivad iga faili eraldi. Seda seetõttu, et kasutajalt eeldatakse failide eelnevat arhiveerimist `tar` vmt. utiliidi abil.

Pakkija	Kokkupakkimine	Pakitud faili vaatamine	Lahtipakkimine
`zip`	`zip pakitud.zip tekst*`	`unzip -l pakitud.zip`	`unzip pakitud.zip`
`gzip`	`gzip tekst.txt`	—	`gunzip tekst.txt.gz`
`bzip2`	`bzip2 tekst.txt`	—	`bunzip2 tekst.txt.bz2`

Ülesanne 10. Arhiveeri kõik oma failid. Paki saadud arhiivi kokku ja lahti erinevate pakkijate abil. Võrdle saadud failide suurusi. Milline pakkija on parim? Võimalikud pakkimisviisid on `zip`, `tar+zip`, `tar+gzip`, `tar+bzip2`. Moodusta uus kataloog, kuhu dearhiveeri oma failid. Kas säilisid loomiskuupäevad, õigused jmt?

2.8 Protsessihaldus

2.8.1 Üldpõhimõtted

Sarnaselt failisüsteemile moodustavad ka protsessid hierarhia — alamprotsessid, ülemprotsess, juurprotsess jne.

Käsu täitmisel käivitatakse alamprotsess ning üldjuhul ülemprotsess „uinub“ ja jääb ootama alamprotsessi lõppemist.

Igal protsessil süsteemis on unikaalne identifikaator (PID — *process identification number*).

2.8.2 Käsud tööks protsessidega

- `ps` — protsesside loetelu näitamine
- `top` — interaktiivne protsessihaldur
- `kill -signaal PID` — tappa protsess nr. *PID* etteantud *signaaliga* (signaal `-9` tähistab „kindlat tapmist“)

2.9 Teave süsteemi kohta

- `date` — kuupäeva ja kellaaja väljastamine
- `cal [kuu] aasta` — terve aasta või ühe kuu kalendri väljastamine
- `du` — kataloogi ja alamkataloogide suuruste väljastamine
- `df` — süsteemi kõigi ketaste vaba kettaruumi väljastamine
- `who` — väljastatakse, kes ja kust on masinas
- `w` — väljastatakse, kes on masinas ja mida teeb
- `finger kasutaja` — kasutaja infokirje väljastamine
- `quota` — kasutaja kettakvoodi väljastamine

Ülesanne 11.

a) Leia Sulle kuuluvate protsesside identifikaatorid (PID).

b) Uuri käsu `ps` lippusid juhendi abil (`man`). Kuidas näha mingile kindlale kasutajale kuuluvate protsesside loetelu? Kuidas näha kõigi protsesside loetelu?

c) Ava kolm terminaliakent. Esimeses käivita top ja lülita sisse ainult oma protsesside näitamine. Teises käivita koorik `sh`. Jälgi esimese akna `top`-ist tulemust. Käivita kolmandas aknas postiprogramm `pine`. Ava veel neljas aken ja tapa kõik käivitatud protsessid järgemööda, saates neile esiteks signaali `-15`, kui aga ei sulgu, siis `-9`.

d) Väljasta jooksva kuu kalender. Väljasta selle aasta kalender. Väljasta tuleva aasta jaanuari kalender ja suuna see faili.

e) Otsi käsu `finger` abil üles õp. Astrid Madseni meiliaadress (TKUG serveris). Mis on selle inimese pärisnimi, kelle kasutajanimi on `seldon`? Selgita välja oma koolikaaslaste kasutajainfot käsu `finger` abil.

f) Mille poolest erinevad käskude `finger` ja `who` (ilma parameetriteta) väljundid?

2.10 Töö tekstifailidega

2.10.1 Sorteerimine

`sort [-fnru] [failinimi] ...`

- `-f` — lugeda väiketähed suurtähtedega võrdseks
- `-n` — tõlgendada sorteeritavaid ridu arvudena
- `-r` — tagurpidi sorteerimine
- `-u` — mitte väljastada korduvaid ridu

Sorteeritav(ad) fail(id) antakse ette failinime(de)ga. Vaikimisi sorteeritakse ridu tekstina.

Näide: sorteerime protsessid nende identifikaatori (PID) järgi tagurpidi järjestuses:
`ps -A | sort -n -r`
(protsessi identifikaator on arv!)

2.10.2 Algus- ja lõpuosa väljastamine

`head` — faili algusosa väljastamine
`tail` — faili lõpuosa väljastamine

`head [ -n arv ] failinimi`
`tail [ -n arv ] failinimi`

Väljastab etteantud arvu `arv` (vaikimisi 10) ridu etteantud faili algusest (`head`) või lõpust (`tail`).

2.10.3 Failide võrdlemine

`diff failinimi1 failinimi2` — erinevuste leidmine. Esimese faili unikaalsed read tähistatakse märgiga < rea alguses, teise omad märgiga >.

2.10.4 Ridade, sõnade ja sümbolite arv failis

`wc [ -clw ] failinimi`

Tagastatakse ridade, sõnade või sümbolite arvu failis.

Ülesanne 12.

a) Loe ära sõnad mingis tekstifailis. Kuidas mõista `wc` poolt antud vastust? Loe dokumentatsioonist (`man`), mida need numbrid tähendavad.

b) Kopeeri endale fail `/home/zolki/public_html/unix05s/arvud.txt`. Vaata, mis seal sees on ja uuri käsu `sort` käitumist:

```
sort -n arvud.txt
sort -nr arvud.txt
sort arvud.txt
```

Suuna sorteeritud read uude faili.

c) Kopeeri endale fail `/home/zolki/public_html/unix05s/inimesed.txt`. Sorteeri see õigetpidi ja tagurpidi.

d) Sorteeri käsu `w` väljund.

e) Sorteeri käsu `du` väljund failimahtude kahanemise / kasvamise järjekorda (kui Sinu kataloogis pole piisavalt alamkatalooge, siis vaatle mõne muu kataloogi failimahtusid).

f) Tee selline fail, mille ridadel oleks ees reanumbrid. Näiteks
`grep -nh '.' inimesed.txt > proov`
(leida failist `inimesed.txt` kõik read, kus on vähemalt 1 suvaline sümbol)
Väljasta sellest 3 esimest rida, 5 viimast rida, 40 viimasest reast 18 esimest.

g) Tee siiani kasutuses olnud failist koopia: `cp proov proov2`, muuda koopias midagi ja võrdle seejärel käsu `diff` abil.

2.11 Võrgukäsud

Tänapäeval kasutab suur osa Internetti ühendatud arvutitest **TCP/IP**[1] **protokolli**[2]. Selles on igale võrguliidesele antud 4-baidine **IP-aadress**. Näiteks kooli serveri `home.tkug.tartu.ee` IP-aadress on `193.40.61.180`. Reeglina on IP-aadressid ülemaailmselt unikaalsed. Aadress `127.0.0.1` on nn. **siseaadress**, s.t. tähistab konkreetset arvutit ennast. Samuti on kokku lepitud terve rida **privaataadresse**, näiteks aadressid algustega `192.168.`, `10.0.0.` jne. Privaataadresse kasutatakse ainult sisevõrkudes, mis pole otseselt välisele Internetile kättesaadavad.

Kui infopakett liigub Internetis (võrkude võrk) ühest arvutist teise, liigub ta lähtearvuti alamvõrgust järgmisse alamvõrku, sellest omakorda järgmisse jne. kuni jõuab lõpuks sihtarvuti alamvõrku ja seal sihtarvutisse. Alamvõrkude vahel toimuva liikluse eest vastutavad **ruuterid**. Iga arvuti peab teadma vähemalt ühe ruuteri aadressi.

Inimestel on raske meeles pidada suurt hulka numbrilisi aadresse, seetõttu on võrgus olevatele masinatele osutamiseks vaja nimesid. Protsessi, kus nime põhjal leitakse IP-aadress või vastupidi, nimetatakse **nimelahenduseks**. Nimepäringute esitamiseks peab iga arvuti teadma vähemalt ühe **nimeserveri** aadressi. Arvutite nimed on jagatud hierarhiliselt **doomeniteks**, need omakorda alamdoomeniteks. Näiteks nimi `home.tkug.tartu.ee.` tähendab masinat `home` juurdoomeni `.` alamdoomeni `ee` alamdoomeni `tartu` alamdoomenis `tkug`.

[1] *Transmission Control Protocol / Internet Protocol*

[2] Protokoll on määratlus, kuidas erinevad arvutid omavahel suhtlema peavad.

Levinud võrgukäske:

- `ping` *`sihtarvuti`* — saadab „prooviks“ 56-baidiseid infopakette *sihtarvutisse*, et kontrollida ühenduse kvaliteeti. Näiteks `ping www.neti.ee`
- `traceroute` *`sihtarvuti`* — püüab kuvada ruutinguteekonda *sihtarvutisse*. Näiteks `traceroute www.sun.com`
- `host` *`nimi_või_number`* — teostab nimelahenduse. Näiteks `host 212.7.7.34`

2.12 Rakendusprogrammid käsureal

Käesoleva õppevahendi piiratud maht ei võimalda käsitleda väikestki osa UNIXi käsurea rakendusprogrammidest. Siiski,

- e-posti klientidena kasutatakse tihti programme `pine` ja `mutt`
- tekstitoimetitena levivad näiteks `nano`, `joe`, `vi`, ...
- veebilehitsejatena tarvitatakse `lynx` ja `links`
- failide allalaadimiseks on käsk `wget`
- käsurea sessiooni algatamiseks on `ssh` (krüpteeritud) ja `telnet` (krüpteerimata)
- failide ülekande sessiooni algatamiseks on `sftp` (krüpteeritud) ja `ftp` (krüpteerimata)
- failide kopeerimiseks arvutite vahel on käsk `scp`

2.13 Tekstitoimeti `vi`

Toimeti on standardiseeritud töötama kõigil UNIX-laadsetel operatsioonisüsteemidel, on töökindel ka aeglastel liinidel ja piiratud terminalidel.

Režiimid

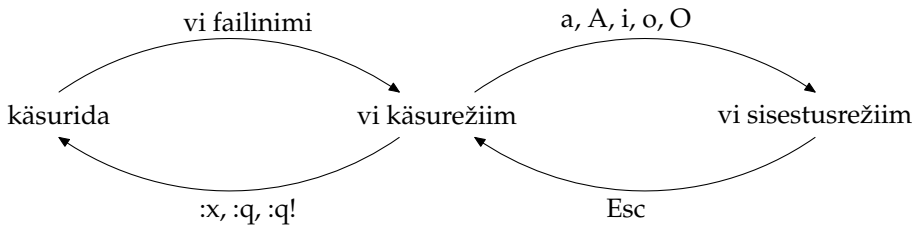
- käsurežiim toimetile käskude andmiseks (osad käsud kooloniga, osad ilma)
- sisestusrežiim teksti sisestamiseks

Käsud

- `:r` *`failinimi`* — lugeda fail jooksva rea järele
- `:f` — anda info käsiloleva faili ja rea kohta
- `:w` *`failinimi`* — kirjutada tulemus faili (kui avati koos failinimega, siis ei pea enam failinime andma)

- `:w! failinimi` — kirjutada olemasolev fail üle
- `:q` — väljumine
- `:x` — väljumine salvestamisega
- `:q!` — väljumine salvestamiseta
- `l` või → — liikumine sümboli võrra paremale
- `h` või ← — liikumine sümboli võrra vasakule
- `k` või ↑ — liikumine rea võrra üles
- `j` või ↓ — liikumine rea võrra alla
- `/regexp` — otsimine jooksvast reast alla
- `?regexp` — otsimine jooksvast reast üles
- `:%s/regexp/asendus/g` — asendab regulaaravaldisele *regexp* vastavad osad asendussõnega *asendus* (`%` — kogu failis, `/g` — kõik esinemised)
- `a` — alustada sisestamist kursori järele
- `i` — alustada sisestamist kursori ette
- `o` — alustada uue reaga kursorist allpool
- `O` — alustada uue reaga kursorist ülalpool
- `A` — alustada sisestamist jooksva rea lõppu
- `R` — siseneda ülekirjutamisrežiimi
- `rc` — asendada kursori all olev märk märgiga *c*
- `x` — kustutada kursori all olev märk
- `dd` — kustutada kursori all olev rida
- `u` — undo, tühistada viimane parandus
- *n käsk* — käsukordus, *n* on naturaalarv; näiteks kustutada 5 rida: `5 dd`
- `Esc` — sisestusrežiimist väljumine tagasi käsurežiimi

Skemaatiliselt



PEATÜKK 3

Automatiseeritud tekstitöötluse keel awk

3.1 Programmeerimise mudel

awki[1] skript võib koosneda kolmest osast:

ALGUSOSA — vastavad käsud täidetakse üks kord kõige alguses
TÖÖOSA — vastavad käsud rakendatakse sisendi igale reale
LÕPUOSA — vastavad käsud täidetakse üks kord kõige lõpus

Neid osi märgitakse awki skriptis järgnevalt:

```
BEGIN {algusosa käsud }
{tööosa käsud }
END {lõpuosa käsud }
```

Iga käsu järel seisab reeglina semikoolon.

3.2 awk käivitamine

awki käske saab anda käsurealt või koondades nad ühte faili. Sellist tekstifaili nimetatakse awk**i skriptiks**.

Toome näite, kuidas kahes tulbas ja tühikutega eraldatud sisendandmetel tulpade järjekord ära vahetada. Olgu sisendfail nimed järgmine:

```
Miima   1959
Priit   1963
Mart    1940
Peeter  1988
```

[1]Selle peatüki teooria-osa tekst pärineb suures osas veebiaadressilt http://kuutorvaja.eenet.ee/programmeerimine/awk/awk.html.

Käsurealt toimub tulpade vahetamine selliselt:

```
awk '{ print $2, $1; }' nimed
1959 Miima
1963 Priit
1940 Mart
1988 Peeter
```

awki skript nimega `vaheta.tulbad.awk`, mis vahetab tulbad, on selline:

```
{ print $2, $1; }
```

Skript käivitatakse nii:

```
awk -f vaheta.tulbad.awk nimed
```

Heaks kombeks on panna skripti laiendiks `.awk`, viitena keelele, milles programmeeritakse.

3.3 Muutuja ja avaldis

awkis ei ole vaja muutujaid enne kasutamist deklareerida ning puuduvad muutujate tüübid. Sõltuvalt kontekstist käsitletakse muutujat kas sõne või arvuna. Muutuja nimi peab algama tähega ja võib sisaldada peale tähtede ka numbreid ja alakriipse.

Toome näite muutujate kasutamisest:

```
BEGIN {
  x = 15;
  y = 2;
  w = "Tartu";
  z = "linn";
  x_korda_y = x * y;
  print x "x " y " = " x_korda_y;
  tartu_linn = w z;
  print w " ja " z " on " tartu_linn;
}
```

- skript sisaldab vaid algusosa
- omistatakse väärtused neljale muutujale: x, y, w, z
- sooritatakse tehe arvudega ja sõnedega ning trükitakse vastavad tulemused

Käsku `print` me oleme juba tarvitanud. Tema järele kirjutatakse jutumärkidesse teksti ja ilma jutumärkideta muutujad, näiteks $nimi$ ja x:

```
print "Teie nimi on " nimi " ja te olete " x " aastat vana.";
```

3.4 Valikulause ja tingimused

Toome näite valikulausest, kus skript genereerib juhusliku arvu ja teatab, milline see arv on:

```
BEGIN {
  srand();            #lähtestab juhuarvude generaatori
  x = rand();         #saab uue juhuarvu piirkonnast [0, 1)
  if ( x < 0.5 ) {
    print x " on väiksem poolest";
  } else {
    print x " on pool või rohkem";
  }
}
```

Tingimusi seatakse võrdlustehte abil:

- $<$ — väiksem
- $>$ — suurem
- $==$ — võrdne (NB! KAKS võrdusmärki)
- $!=$ — mittevõrdne
- $<=$ — väiksem-võrdne
- $>=$ — suurem-võrdne

Võrdlustehete tulemustega saab sooritada loogilisi tehteid:

- ! — eitus
- && — ja
- || — või

Näiteks kirjutame skripti, mis „mõtleb“ juhusliku arvu x ja väljastab, kas see rahuldab tingimust $0.2 < x < 0.8$:

```
BEGIN {
  srand();
  x = rand ();
  if ( (x > 0.2) && (x < 0.8) ) {
    print " 0.2 < " x " < 0.8";
  }
}
```

3.5 Kirje ja väli

`awk` käsitleb sisendit struktureerituna. Sisend on jaotatud **kirjeteks**; vaikimisi on kirjete eraldaja reavahetus `\n`. Kirje on omakorda jaotatud **väljadeks**; vaikimisi on väljade eraldaja tabulatsioonimärk `\t` või tühik.

Programmi tööosas saab väljade poole saab pöörduda muutujate `$1`, `$2`, `$3`, `$4` jne abil. Muutuja `$0` väärtuseks on terve kirje.

Sõltuvalt välja sisust saab teha nendega tehteid. Toome näite, kus leiame tulpadena esitatud lähteandmete ridade summad. Sisendfailiks olgu `arvud`:

```
1 3 4 5
4 6 2 4
1 5 9 4
2 4 5 5
```

ning skript `leia.summad.awk` on seesugune:

```
BEGIN { print "Leiame ridade- ja kogusumma"; }
{
  summa = $1 + $2 + $3 + $4;
                    # leiame konkreetse rea arvude summa
  print $1 " + " $2 " + " $3 " + " $4 " = " summa;
  kogusumma = kogusumma + summa;
                    # liidame rea summa juurde kogusummale
}
END { print "Kogusumma = " kogusumma; }
```

Siin on esitatud kolm võimalikku skripti osa: algusosa, tööosa ja lõpuosa, kusjuures neile osadele vastavad käsud kirjutatakse looksulgudesse. Ühes osas kasutatud muutajatel on teises osas sama väärtus (nt $kogusumma$).

Käivitame skripti:

```
awk -f leia.summad.awk arvud
Leiame ridade- ja kogusumma
1 + 3 + 4 + 5 = 13
4 + 6 + 2 + 4 = 16
1 + 5 + 9 + 4 = 19
2 + 4 + 5 + 5 = 16
Kogusumma on 64
```

3.5.1 Kirje- ja väljaeraldusmärkide ümbermuutmine

Mõnel juhul on vajalik sisendandmeid kirjeteks ja väljadeks jagavaid märke muuta. Vastavalt saab muuta ka väljundis kasutatavaid kirje- ja väljaeraldusmärke.

Seda saab teha näiteks defineerides skripti algusosas üle vastavate muutujate väärtused:

- `FS` — sisendi väljade eraldaja (*field separator*)
- `RS` — sisendi kirjete eraldaja (*record separator*)
- `OFS` — väljundi väljade eraldaja (*output field separator*)
- `ORS` — väljundi kirjete eraldaja (*output record separator*)

Lisaks saab kasutada `awk`-iga seotud muutujaid:

- `FILENAME` — sisendfaili nimi
- `NF` — käesoleva kirje väljade arv (*number of fields*)
- `NR` — kirje absoluutne järjekorra number (*number of the record*)

Toome näite nende muutujate kasutamise kohta. Algandmed on sellised:

```
Priit
Elva
Jüri 15
tel. 15 123

Mart
Jüri
Elva 15
tel. 12 3 15

Peeter
Tartu
Telefoni 5
tel. 12 34 56
```

Nad soovitakse esitada nii, et ühte linna puutuv oleks ühel real, read oleks nummerdatud ning kõige lõppu kirjutataks algandmete faili nimi:

```
1:Priit:Elva:Jüri 15:tel. 15 123
2:Mart:Jüri:Elva 15:tel. 12 3 15
3:Peeter:Tartu:Telefoni 5:tel. 12 34 56
andmefail: andmed
```

Sellise töö teeb ära järgmine skript:

```
BEGIN { FS="\n"; RS="\n\n"; OFS=":"; ORS="\n"; }
{ print NR, $1, $2, $3, $4; }
END { OFS=" "; print "andmefail:", FILENAME; }
```

3.6 Adresseerimine

Regulaaravaldistega saab määrata, millistele sisendi ridadele tööosa käsud rakenduvad. Toome näite, kus tegeldakse vaid Tartu linna temperatuuride keskmistamisega. Algandmed failis `temperatuurid` on sellised:

```
linn kp 6:00 12:00 18:00 24:00
Tartu 19/4/2000 13.6 15.1 15.8 14.0
```

```
Valga 19/4/2000 11.6 12.1 12.8 11.0
Tartu 20/4/2000 14.6 18.4 13.5 13.3
Valga 20/4/2000 13.4 15.7 15.1 14.3
Tartu 21/4/2000 16.6 21.1 19.3 17.1
Valga 21/4/2000 15.7 19.4 17.6 15.5
Tartu 22/4/2000 15.6 23.6 22.6 15.4
Valga 22/4/2000 13.5 16.2 14.7 15.3
Tartu 23/4/2000 17.3 19.4 21.4 12.3
Valga 23/4/2000 18.2 16.9 13.8 15.2
```

Skripti `tartu.keskm.temp.awk` tööosa tegeleb vaid nende andmeridadega, mis klapivad regulaaravaldisega `/Tartu/`:

```
BEGIN {
  print "Tartu õhutemperatuuri ööpäevane keskmine";
  print "Teostati neli mõõtmist: 6:00 12:00 18:00 24:00\n\n";
  print "koht kuupäev mõõtmistulemus";
}
/Tartu/ {
  keskmine = ( $3 + $4 + $5 + $6 ) / 4;
  print $1, $2, keskmine;
}
END { print "\nMõõtmisi teostasid Priit ja Mart\n"; }
```

Käivitamisel saame järgmise tulemuse:

```
awk -f tartu.keskm.temp.awk temperatuurid
Tartu õhutemperatuuri ööpäevane keskmine
Teostati neli mõõtmist: 6:00 12:00 18:00 24:00
koht kuupäev mõõtmistulemus
Tartu 19/4/2000 14.62
Tartu 20/4/2000 14.95
Tartu 21/4/2000 18.52
Tartu 22/4/2000 19.30
Tartu 23/4/2000 17.60
Mõõtmisi teostasid Priit ja Mart
```

3.7 `awk` harjutusülesandeid

Ülesanne 13. Koosta tekstifail `raamatud.txt`, kus on tabulaatoriga eraldatud järgmise sisuga väljad:
raamatu autor, raamatu pealkiri, ilmumisaasta, väljaandmise koht, väljaandja
Kasuta käsku `awk` järgmiste ülesannete lahendamiseks:

a) Kirjuta käsureale korraldus, mis kuvab ekraanile antud faili, kusjuures iga rea järel on üks tühi rida.

b) Kirjuta käsureale korraldus, mis kuvab igast reast esimese välja.

c) Kirjuta käsureale korraldus, mis kuvab väljad järjekorras: väli 4, väli 3, väli 2, väli 1.

d) Kirjuta käsureale korraldus, mis kuvab faili, lisades tema ette pealkirja `Dokument`.

e) Kirjuta käsureale korraldus, mis kuvab igat rida 3 korda.

f) Kirjuta käsureale korraldus, mis kuvab teise välja, kui selle rea kolmanda välja väärtus on suurem kui 1980.

Ülesanne 14. Malle peab oma tuttavate kohta taskumärkmikku
`/home/zolki/public_html/unix05s/taskumarkmik.txt`,
kus on sellised väljad:

1. eesnimi
2. sünnikuupäev kujul aaaa-kk-pp
3. elukoht
4. mobiiltelefoni number rahvusvahelisel kujul, puudumise korral 0
5. kui palju see inimene Mallele võlgu on
6. e-posti aadress

Lahenda Malle jaoks järgmised ülesanded:

a) Ees ootab sõit Riiga. Milliste tuttavate juures leiaks Malle öömaja? Leia nende nimed ja e-posti aadressid.

b) Mallel tekkis soov saata kõigile tuttavatele tavapostiga kiri ning vaja on välja trükkida nimed ja aadressid. Oletame, et piisab väljade *eesnimi* ja *elukoht* sisust, mis peaksid asuma teineteise kohal ehk umbes nii:

```
Mari
Tartu

Jüri
Tartu
...
```

c) Mallel on vaja kiiresti võlad kätte saada. Kõigile võlglastele on tarvis saata kiri tekstiga:

```
Kallis tuttava nimi
Oled mulle võlgu võla suurus krooni. Palun maksa tagasi!
```

Eriti viisakas oleks, kui iga kirja alguses enne pöördumist oleks veel inimese nimi ja aadress (antud juhul siis linna nimi).

Ülesanne 15. Kasuta käsku `awk` järgmiste ülesannete lahendamiseks, võttes sisendiks faili `/etc/passwd`:

a) Kirjuta käsureale korraldus, mis annab sama tulemuse kui `head -n 20 failinimi`.

b) Kirjuta käsureale korraldus, mis kuvab sisendi kaheksanda rea.

c) Kirjuta käsureale korraldus, mis kuvab sisendi read nr. 5 kuni nr. 56.

d) Kirjuta käsureale korraldus, mis kuvab sisendi kolmandal väljal olevate suuruste summa.

e) Kirjuta käsureale korraldus, mis kuvab iga rea väljade arvu.

f) Kirjuta käsureale korraldus, mis kuvab sisendi ridade arvu (nagu käsk `wc -l`).

g) Kirjuta käsureale korraldus, mis kuvab iga rea viimase välja.

h) Kirjuta käsureale korraldus, mis kuvab iga rea tagant kolmanda välja.

i) Kirjuta käsureale korraldus, mis annab sisendi kolmandal väljal olevate suuruste keskmise.

j) Kirjuta käsureale korraldus, mis kuvab sisendi 5., 6., 7., 8. ja 9. rea.

k) Kirjuta käsureale korraldus, mis kuvab sisendi kõik read, välja arvatud 5., 6., 7,. 8. ja 9. rida.

Ülesanne 16. Jätkame Malle teemat (vt. ülesanne 14).

a) Kui palju Mallele kokku võlgu ollakse?

b) Kes on kõige rohkem võlgu?