Tigu lahkamas, ehk ekskursioon UNIXi maailma

Autor: Anto Veldre, 1995

Esimene osa: eelteadmisi vastase kohta.

UNIXi ajalugu algas 60-ndate aastat lõpus ning seda on kuivalt ja igavalt käsitletud pea iga arvutiõpiku alguslehekülgedel. Siinkohal vast niipalju, et grupp härrasmehi, parandamatud idealistid, otsustasid kord luua maailma parima operatsioonisüsteemi. Nagu iga ideaalse projekti puhul, tuli see ellu viia vanarauda kasutades - selleks kõlbas vana ja logisev PDP-3 tüüpi miniarvuti.

Loomulikult ei saanud nii pretensioonikat asja kirjutada üheski olemasolevas programmeerimiskeeles, niisiis otsustas üks mainitud härrasmeestest spetsiaalselt selleks otstarbeks ka uue keele võlja mõelda; kuna ta just oli lõpetanud keelega **B** , siis kiikas ta korraks tähestiku poole ning uue keele nimeks sai **C** (häälda: sii). Sama mure tekkis süsteemi enda nimevalikul. Kuna grupi kandvamad jõud olid just lõpetanud süsteemiga **Multics** (sõnast *multi* - palju), siis pidi loodav süsteem saama veelgi pretensioonikama nime. **UNIX** näib viitavat universaalsusele, unifitseeritusele ning unikaalsusele. Kõike ülejäänut saab lugeda paljudest paksudest ja tarkadest raamatutest, mis vahepeal UNIXi kohta kokku on kirjutatud.

Mõned kõige ilmsemad faktid UNIXi kohta : ta on ikka veel elus, ta areneb ühtlaselt ja tagasilöökideta, ta on haaranud arvestatava osa arvutiprogrammide turust.

Käesolev kirjutis püüab UNIXiga tutvustada eelkõige neid, kes kodus koera asemel peavad IBM-PC-tüüpi arvutit ning kes on surmkindlalt veendunud, et mistahes muid rehkendamise riistu pole kunagi olemas olnudki. Ka neid, kes täpselt teavad, et pole eales eksisteerinud mingeid muid operatsioonisüsteeme peale Microsoft DOS'i, kusjuures versiooninumbriga alates 3.3 .

Me ei paku lugejale mingeid akadeemilisi tõdesid ega teoreetilisi arutlusi. Me räägime UNIXist kui reaalsest faktist, mida pole maakeeles seni arusaadavalt lahti kirjutatud. Esimeses osas püüame vaadelda üldisi erinevusi ning mitte vaevata lugejat käsuridade süntaksiga.

Esimesed muljed UNIX'ist

Esimene kontakt UNIXiga on kindlasti äärmiselt eemaletõukav. Olenemata sellest, kas üritada sidet moodemi kaudu või siis mõnes asutuses kogemata ripakile jäetud terminaali vahendusel, näeb esimene vestlus välja umbes sedasi:

```
Carrier 2400

ULTRIX V4.1 (Rev. 52) (zeus)

login: uugu
Password:
Login incorrect
login: Uugu
Login incorrect
Password:
login: UUGU
Login incorrect
Carrier lost
```

Ent ka sellisest lühikesest dialoogist saab mõningat infot. Te saite teada, et Teie vastaseks on UNIXi lähedane sugulane nimega ULTRIX, mis jookseb vaid firma DEC miniarvutil VAX. See on umbes kuupmeetrine kast, mis ammu moest läin'd, kuid milliseid Eestisse humanitaarabiga päris mitmeid on sattunud. Võimalik, et login-rea ees näete hoopis sõna MUNIX . Ka tema on

UNIXi sugulaste hulgast, kasti suurus on ehk veidi väiksem ning masina tüübiks CADMUS. Veel on võimalik, et Te juhtute vestlema veidi kaasaegsema süsteemiga nagu SCO UNIX System V/386 (ja see käib päris tavalisel 386-masinal) ehk siis siis isegi SOLARISega, mis jookseb firma SUN tööjaamadel.

Lisaks saite Te teada, et masina nimi on "zeus". Siin pole üldsegi tegu mingi inglasliku traditsiooniga, kus suvemaja kannab uhket nime "Villa Hortensia". UNIX-masinad ajavad tihtipeale teineteisega juttu (moodemi, püsiliini või satelliidi vahendusel) ning masina nimi on üks automaatseks sideks väga vajalik parameeter. Väike valik Eesti UNIX-masinate nimedest: osiris, anubis, mars, venus, nano, piko, siil ja mp43.

Huvitav, huvitav, ütleb nüüd keskmine arvutitarbija, millest see siis tuleb, et üksainus op.-süsteem saab joosta mitmel erineva prosaga masinal? Tegelikult ongi just ühilduvus üks põhieesmärke, mis UNIXi loojad enda ette seadsid. Kui uue arvuti jaoks on olemas **C**-kompilaator, siis tuuakse kohale UNIXi mingi versiooni algtekstid ning lihtsalt kompileeritakse see kohapeal kokku. Tõsi, mõned kõige kriitilisemad lõigud tuleb siiski kirjutada assembleris, nii peame tööjõukuluks arvestama paari inimkuud. Ja jooksebki UNIX kõige erinevama raua peal...

UNIXi veel ühe omadusena on kõik algtekstid seaduslikult saadaval ning alati on võimalik neisse muudatusi teha, kui mingi parameeter Teid ei rahulda.

Veidi headest kommetest

Lükkame veidi edasi oma esimese õnnestunud sisselogimise UNIX-süsteemi (pange tähele, mitte masinasse, vaid süsteemi) ning kirjeldame mõningaid tegevusi, mida on parem UNIXi läheduses vältida. Yks esimesi küsimusi, mida Te sooviksite esitada: "Kus asub arvuti?" Tegelikult on see täiesti ükskõik. Kus ta ka ei asuks, ei nähta Teid arvuti vahetus läheduses kuigi meelsasti. Siin on tegemist puhtalt julgeolekuküsimusega. Eriti SCO UNIXi puhul, mis pesitseb tavalises 386-masinas, tekib Teil pärast töö lõppu

instinktiivne tahtmine masinal toide välja lülitada. Sellegipoolest: ärge mitte kunagi üritage UNIX-masinat voolu alt välja lülitada, kuni pole antud käsku **shutdown** või selle ekvivalenti teistes versioonides. Nagu Microsoft Windows'is nii ka UNIX'is kasutab masin mälupikendusena kõvakettal paiknevat swap-ruumi.

Lisaks sellele ei taha UNIX iga tühja asja pärast ketast kulutada, vaid hoiab muudatusi mälus, kust need siis iga minuti või poole tagant kettale viiakse. Peaks nüüd toide järsku kaduma, siis on *filesystem corrupted* ja paarikümne inimese töö sassis. Ega ilmaasjata reklaamid väida, et vaid hull võib UNIXi ilma UPSita käima lasta (UPS on maakeeli tagavaratoide akudel).

UNIXi puhul kohtate Te palju imelikke ja arusaamatuid nähtusi. Näiteks logite Te ennast sisse ühte masinasse, aga Teie faile hoitakse kusagil hoopis teises masinas. Ju siis nii on otstarbekam, see on süsteemi administraatori valdkond ning kuni Teil õnnestub normaalselt tööd teha, ei tasu uurima hakata, miks Teid füüsiliselt sellesse või teise masinasse on paigutatud. Ju need masinad on omavahel võrgus ning see tähendab UNIXi puhul midagi sootuks enamat kui tavalist DOSi käiates.

Ärge kunagi küsige, miks on UNIXi käskudel nii lollakas ja DOSist erinev süntaks! See poleks aus juba ajaloolistel kaalutlustel, sest UNIX sai lapse-east välja veel siis, kui hr. William H. Gates koolipinki nühkis. Seetõttu langeb ära ka küsimus, kes kelle pealt malli on võtnud. Veidi lihtsustades võib öelda, et DOS kujutab endast alamhulka UNIXi võimalustest pluss mõningad kõrvalmõjud mujalt. Siin ei saa läbi näiteta. Microsoft DOS'i loojad kartsid käsusüntaksi täielikult UNIXi pealt maha vehkida (Pigem kartsid nad küll AT&T firma sanktsioone) ning keerasid failinimedes kaldkriipsu tagurpidi... Kataloogilistingu saate UNIXis käsuga **ls**. See kahetäheline käsk töötab igal pool ja alati. Küll aga võib manuaali uurides välja selgitada, et konkreetsel versioonil on veel hulk võtmeid, nii näiteks **ls -a** näitab ka varjatud, punktiga algavaid faile, **ls -R** siseneb ühtlasi ka kõigisse alamkataloogidesse ning **ls -rt** (inglise k.: *reverse time*) näitab faile tagurpidi ajalises järjestuses, st

viimatiloodud fail viimasena. Kogu süntaks on kirjeldatav sellise reaga (kusjuures osa võtmeid välistavad teineteist):

```
ls [ -ACFRabcdfgilmnopqrstux ] [ names ]
```

Näide on System V manuaalist, UNIXi teistel harudel on kindlasti mõni võti erinev.

Ärge UNIXi läheduses kunagi rääkige TSR-idest. Sellist terminit siin lihtsalt ei kasutata. Ka ärge üritage UNIXit üle kavaldada sel moel, et kirjutate assemblerprogrammi katkestuse haaramiseks. Sel puudub eriline mõte, kuna UNIXi tuum (ingl.k: *kernel*) kasutab protsessori kaitstud rezhiimi.

Kuni Te pole administraatoriga sinasõbraks saanud, ärge paluge temalt flopiseadme kasutusluba. Esiteks pole paljudes UNIX-masinates MS-DOS'i ketastega midagi peale hakata, kui aga süsteem siiski lubab teist tüypi ketaste lugemist ja vastav *drive* on olemas, siis sageli puudub Teil füüsiline ligipääs flopiseadmele ning veel sagedamini pole Teile flopide lugemine lubatud. Masin teatab Teile viisakalt *Sorry, You are not authorized to ...*. UNIXi hierarhial pole demokraatiaga midagi pistmist, masinas on vaid üks mees, kes tohib teha kõike. Selle mehe nimi on **superuser** (ingl.k: ülemkasutaja) ning tema kasutajanimeks eri versioonides "root" , "system" või midagi sarnast. Väga sageli on tegemist väikese või keskmise süsteemiga, kus administraator ja superuser on üks ja seesama isik. UNIX-masinas on peale superuser'i veel mitut sorti priviligeeritud kasutajaid, kuid kardetavasti ei kuulu Teie (vähemalt esialgu) nende hulka.

Siinkohal hakkab keskmise häkkeri eneseuhkus mässama: "Kas nad peavad mind tobuks või mis? Kuidas saab masinaga tööd teha, kui mind isegi floppiaugule ligi ei lasta?! Aga kui ma tahan oma programme tuua ja viia? Kui ma tahan endale *backup*'i teha?". Siin avaldub DOSi ja UNIXi ideoloogiline vastuolu ehk kõige jõulisemalt.

Kõigepealt *backup*'ist. Iga normaalse UNIX-masina juures on inimene, kes varundamise (ebaõnnestunud vaste sõnale *backup*) eest vastutab. Enamasti käib asi siiski automaatselt. Igal õhtul mingil kellaajal lülitub sisse lindikapp või

striimer ning masin teeb ise backup'i. On kuulda olnud UNIX-masinatest, mis teevad iga päev täieliku backup'i mõnesse teise gigabaidise kettamahuga süsteemi, kulutades selleks Interneti kaudu vaid mõne minuti. Niisiis ei kao Teie failid kõvakettalt kuhugi. Enamik UNIXi tekstitöötlusprogramme ei vaevu isegi *backup*-koopiat tegema, selleks lihtsalt puudub vajadus. UNIXi käigushoidmine eeldab raua tõrgeteta tööd ning enamasti on see tingimus täidetud. Pole juhus, et UNIX on massiliselt levima hakanud just nüüd, kui protsessorite ja kõvaketaste töökindlus on ületanud teatava kriitilise piiri...

Mis floppidesse puutub, siis korralike sideliinidega süsteemis Te lihtsalt ei vaja neid. UNIXis on olemas vahendid failide kopeerimiseks ühest masinast ja süsteemist teise. Kui Teie masin on ühendatud ülemaailmse INTERNET-võrguga (ning ka Eestis paljud UNIX-masinad on seda), siis saate maailma suvalisse võrgusolevasse masinasse faile saata ja neid sealt tellida. See võimalus on olemas ka tavalisel privileegideta kasutajatel. UNIX võtab teiste masinatega sidet täiesti iseseisvalt, ilma inimese abita. Olles kord saanud vastava käsu, üritab süsteem faili transportimist ikka ja jälle, kuni see õnnestub.

Veel üks nõuanne. Kui Teie ees on IBM-PC-ühilduv arvuti ja UNIXi SCO versioon, ärge raisake aega võrgukaartide ja serverite otsimisele! Võib juhtuda, et neid polegi. UNIX on oma olemuselt ise mitmekasutajasüsteem (ingl.k *multiuser system*) ning op.-süsteemis endas leiduvad kõik vahendid paljude tarbijatega suhtlemiseks. Yhe masina puhul ei saa veel võrgust rääkida, kuigi selle taga töötavad kuni paarkümmend inimest korraga.

Protsessorist ja terminaalidest

IBM PC-ühilduva masina puhul probleeme ei teki -- prosa on 386 ning sellega saab korraga tööd teha vaid üks inimene. Imelikul kombel suudab sama protsessor toime tulla kümne ja enama inimesega, kui kasutada firma Santa Cruz Operation versiooni UNIXist (SCO System V/386). Milles on asi? Kust tulevad ülearused Mipsid ja Flopsid?

Tegelikult pole siin midagi üleloomulikku. Käivitage prooviks DOSi all mõni multimeedia tüüpi rakendus (olgu selleks kasvõi mäng nimega "DOOM"). Tehke sama veel kord ning hästi aeglases masinas (16 MHz taktiga ja kehva videokaardiga). Mängimine kaotab mõtte, sest kujutis ja heli ei tule enam õige kiirusega kätte : prosa kogu aeg kulub videokaardiga suhtlemisele ning heli-sample'ite edastamisele. Siin ongi peidus põhjus, miks 7 MHz taktiga Amiga graafika on võimsam kui 33 MHz IBM-ühilduval masinal. Amigal on spetsiaalsed kivid, mis heli- ja pildimured prosa kaelast ära võtavad. Seevastu puhta arvutusülesandega jääb Amiga kohe jänni, siin osutub määravaks protsessori jõudlus.

Kõige mustvalgemal kujul on valik selline -- kas ilus värvipilt või tegelik töö. UNIXi puhul on valik ühene -- protsessor teeb tehteid, mitte ei konstrueeri ilupilti. Kui masinas on korraga 20 kasutajat, siis statistiliselt vaid mõni neist tegeleb protsessori koormamisega (arvutamise, faili pakkimise või kompileerimisega). Ülejäänud istuvad rahumeeli ekraani ees ning kas sisestavad mingit teksti või siis loevad seda. Praktikas pannakse 386 tüüpi masina sisse näiteks 16 COM-pordiga *serial* card. Iga pordi küljes ripub üks PC-masin või terminaal. (Paljud pole osanud Windows'i all Terminal-ikooniga midagi peale hakata, vaat selleks ta ongi!).

Järjestikühenduse jaoks on tavaline ühenduskiirus 9600 või siis 19200 boodi. Selle kiirusega koguneb ikka kenake hulk terminaale, enne kui 386-protsessor hätta jääb. Muide, kriitiline on pigem väljastus- kui sisestuskiirus, inimene lihtsalt ei suuda enam kui 10 märki sekundis sisse tippida. Masinais, mis on spetsiaalselt UNIXi jaoks konstrueeritud, tegeleb andmevahetusega ikka mõni spetsiaalne plaat, säilitades prosa ainult sisulise töö tarvis.

Veidi terminaalidest, kuivõrd neid kasutati seni vaid piiratud hulgal ning enamik arvutifännidest ei tunnegi selliseid riistu, kasutati neid varemalt ju vaid riigile alluvates arvutuskeskustes. Terminaal on välimuselt üpris arvuti välimusega. Sisu on tal küll tunduvalt vaesem -- pole märgata mingeid kettaseadmeid. Terminaal ühendatakse arvutiga enamasti (kuigi mitte alati) järjestikliini abil. Terminaali ainsaks ülesandeks on võtta klaviatuurilt sümboleid ja edastada neid liini, samuti võtta liinilt sümboleid ja näidata neid

ekraanil. Tegelikkuses on asi veidi keerulisem, kuivõrd headel terminaalidel on mitu kooditabelit, sisukas *setup*, piisav valik juhtkoode ning muid mugavusi.

Terminaalide puhul tuleb ilmsiks huvitav paradoks, mida on meie arenevas ühiskonnas küllaltki tähtis teada. Terminaali varguse korral ei saa varas kätte mitte üht bitti infot, kuna arvuti koos kõvakettaga asub luku, riivi ja signalisatsiooni taga. Kindlasti on lihtsam kaitsta infovarguse eest ühtainust ruumi, kui tervet maja. Ka pole sulidel terminaaliga midagi peale hakata, kui nad just UNIXit installeerida ei kavatse.

Mõnda tarbijat võivad huvitada just graafilised rakendused. UNIXil on lahendus ka selleks puhuks. Tuntud headuses kolmemõõtmelist graafikat teevad näiteks SUN-firma tööjaamad. Need baseeruvad UNIXi vennal, op.-süsteemil SOLARIS. SUNis jookseb akendussüsteem X-Windows ning sellega töötamine nõuab alustuseks vaid hiire käsitsemise oskust (hiire nuppude funktsioonid on mõneti erinevad). Muide, esimene aken, millega Teil seal tegemist tuleb, küsib Teilt eht-UNIXiliku visadusega kasutajanime ja parooli... Põhimõtteliselt võite Te küll iga UNIX-masina külge X-terminaale lylitada, kuid see koormab masinat, muutes töö võimatult aeglaseks. Samas, kui Te soovite just käsureaga töötada, avage **console** või **xterm**-nimeline aken ning tegutsege!

Kui tööjaamu saab palju, siis on mõtet hoida kokku kõvaketaste pealt ning ühendada tööjaamade külge suuremahulise kõvakettaga server, siitmaalt hakkab taas kehtima DOSist tuttav loogika. Novell'ist erinevalt on tegemist võrdväärsete UNIX-masinatega, millest mõni võib olla võimsam ja teine piiratud võimalustega, kuid kiire sidemasin (router); kolmas hoopis võimsa graafika ja heliga multimeediamasin. Siin on varjul põhjus, miks kaitseb UNIX end nii kiivalt paroolisüsteemiga. Omades ühes masinas maksimaalseid õigusi, on teatud määral võimalik ligi pääseda ka teistele masinatele (muidu ei saaks ju toimida automaatselt tegutsevad sideprogrammid). Tervet maailma hõlmav võrk ei elaks päevagi, kui ta ei kaitseks ennast paroolidega. Samal põhjusel on raskusi DOSi masina integreerimisel üldisse võrku, pole ju DOSis

kurja käe eest mingit tõsisemat kaitset (Novell ja teised võrgusüsteemid on eraldi jututeeema).

Miks just UNIX?

Viimasel ajal on nii mõnigi firma pööranud oma pilgu UNIXi suunas. Põhjusi selleks on mitmeid, ühena esimestest kindlasti andmekaitse vajadus. Edukas firma ei saa lubada, et tema andmed sattuksid konkurendi kätte. Teisena tuleb kindlasti töökindlust. Korralikult installeeritud UNIX-süsteem töötab korraliku administraatori käe all kindlasti mitu kuud ilma masinat võrgust välja lülitamata. Kolmanda põhjusena nimetatakse maksumust ning see tundub esialgu üllatav, kuna UNIX (paljas op-süsteem ilma rakendusteta) maksab oma paarkümmend tuhat krooni, masinast ja terminaalidest rääkimata.

Olukord muutub otsustavalt, kui Teie süsteemis on kümme ja enam töökohta. Täiesti töökorras tekstiterminaali võib osta 300-2000 EEK eest. Küll ei saa Te käima lasta Windows'i, aga sekretäritöö ja andmebaasidega probleeme ei teki. Ilma igasuguse serveri ja võrgutarkvarata saavad kümned inimesed korraga firma laoseisu jälgida, kliente ja kaupu registreerida, kirjavahetust pidada. Kui firmas tehakse konstruktori- või kunstnikutööd, tuleb lisaks üles seada X-terminaal. Juhul kui põhitöö ongi seotud graafikaga, siis tuleks pigem muretseda SUNi, Silicon Graphics'i või Hewlett-Packard'i tööjaamu, sellise kiiruse ja kvaliteediga graafika on PC peal kardetavasti võimatu.

Kokkuhoid algab sellest hetkest, kui Te ostate vaid ühe (olgugi et kalli) rakendusprogrammi ning see on kasutatav kogu Teie UNIX-süsteemis. Kokkuhoid jätkub, kui selgub, et UNIX-masinas on võimalik emuleerida tervet IBM-ühilduvat arvutit ning kui vaja, siis mitutki veel. Kindlasti osutub määravaks valdkond, milles tegutsetakse, kuid rusikareegel kõlab nii, et umbes 10 töökohast alates muutub UNIX-süsteem tasuvamaks.

On olemas veel üks tugev argument UNIXi kasuks. UNIX'i viirusi pole olemas. Tõsi, esineb juhtumeid, kus mõni UNIXit hästi tundev programmeerija üritab arvutisidet enda kasuks tööle panna. Nii juhtus näiteks 1988. aastal, kui aspirant Robert Morrise kuritahtlik programm viis rivist välja umbes 6000

arvutit 60 000-st, mis tollal INTERNETi kuulusid. Siiski oli tegemist vaid tööseisakuga ning andmed ei hävinud. Levis see programm vaid neid masinaid pidi, mille administraatorid ei järginud teatud julgeolekunõudeid. Kindlasti on vaja teada, et Morrise isa oli üks USA juhtivaid spetsialiste UNIXi ja paroolkaitse alal, vastavad eriteadmised sai poeg arvatavasti isa tööd jälgides.

Vastuväiteidki võib leida UNIXile üpris mitu. Kindlasti küsitakse, miks sellest UNIXist enne midagi kuulda polnud?! Vastusena sellele viitan lõigule SCO UNIX System V/386 versioon 3.2 manuaalist, kus USA föderaalseaduse kohaselt on keelatud selle versiooni müümine Afganistani, Hiinasse, Rumeeniasse, Ungarisse, Eestisse, Lätisse, Leetu, Vietnami, Kampucheasse ning veel mõnessegi riiki. Siiski, ka Vene ajal liikus siinmail UNIXi versioone, enamjaolt küll kehva riistvara peale kohandatud nuditud variante. Aga oli ka üksikuid **tõelisi** masinaid. Tore legend on seotud ühe Teaduste Akadeemia allasutusega, kuhu embargoseadust eirates õnnestus UNIX muretseda. CADMUS-tüüpi masin nimetati seal ümber MUSCAD'iks, kuigi muusika loomisega ei tegeldud tolles asutuses teps mitte. Nüüd, kui UNIX Eestis on põranda alt välja lubatud, saab teda omandada täiesti seaduslikult.

UNIXi tarkvaraga on Eestis veel teatud raskusi. Asi on selles, et UNIXi soft'i pole võimalik varastada, nagu see MS-DOSi puhul sageli sünnib. Installeerimiseks vajalik magnet- või valguskandja kannab individuaalset numbrit, ükski oma firma au eest hoolitsev administraator seda kopeerimiseks ei anna, kui see aga varaste kätte satub, pole softist kasu, kuna puuduvad salasõnad. Kui soft siiski liikuma läheb, on selle allikas seerianumbri järgi alati üheselt määratav.

UNIXi tarkvaraturg on Eestis praktiliselt täitmata. Kui leidub soovijaid, küll programmid mere tagant kohale tuuakse. Pealegi on Eesti arvutifirmadel piisavalt oskusi, et kohapeal päris korralikku softi kirjutada. Alati on võimalik arvutivõrgu vahendusel free- või shareware programme tellida. Kui Teil on soov UNIXiga tutvuda, kuid puudub raha tema ostmiseks, installeerige oma

386-masinale Linux! See on UNIXi noorem vend, mis mõnes osas UNIXist endast parem ja kiiremgi.

Teine osa

Protsessid

ehk raamatupidamise tähtsusest

UNIXi sisemuses peetakse kõige toimuva üle raudset arvestust. Vähe sellest, et masin fikseerib sisselogimise kuupäeva ja kellaaja, UNIX peab eraldi arvestust ka kõigi protsesside üle, mis süsteemis käima lastakse. Kusagil mälusopis on protsessitabel, kus kirjas kõik protsessid koos käivitamise kellaaja ja omaniku nimega. Igal protsessil on oma number (PID -- *process identificator*), mille järgi saab iga protsessi masinast üles leida. Ühemehesüsteemiga harjunud arvutisõbral on päris raske mõista, MILLEKS kõik need numbrid ja tabelid, niikuinii saab protsessor korraga vaid ühtainukest programmi täita.

Tõepoolest, protsessor suudab korraga järada vaid ühte masinkäskude järjestust. Siin aga astub mängu säärane protsessori omadus nagu kaitstud rezhiim (*protected mode*). Paarkümmend korda sekundis viiakse protsessor sundkorras kaitstud rezhiimi, kus ta vaatab läbi kogu käivitamist ootavate protsesside tabeli ning annab juhtimise mõnele järjekorras seisvatest protsessidest. Kui säärane ümberlülitamine toimub piisavalt kiiresti, siis jääb mulje, nagu lahendaks masin korraga mitut ülesannet. Mida rohkem on protsessitabelis ootavaid ülesandeid, seda aeglasemaks muutub masina töö.

Oma individuaalse PID saavad kõik masinas toimuvad protsessid, olenemata sellest, kas on tegu käsurealt käimalastud programmiga või siis keerulisema jadaga, kus üks programm kasutab oma töös teisi abiprogramme. PID suurus on tavaliselt limiteeritud arvuga 2 astmel 16 , siis algab numeratsioon taas algusest.

MS-DOSi all on programmeerija nuhtluseks programmid, mis ühel või teisel põhjusel kinni jooksevad. Tuleb ju siis masinale alglaadimine korraldada. UNIXi puhul on olukord kardinaalselt erinev. Mingi üksik protsess ei halva süsteemi tegevust. Ta lihtsalt EI SAA seda teha, sest juba järgmisel sekundikümnendikul võtab kaitstud rezhiimis tegutsev UNIXi tuum (*kernel*) tema käest juhtimise ära ning annab selle mõnele muule protsessile. Nüüd on paras aeg uurida, MILLINE protsess kinni jooksis. Käsuga **ps** saate UNIXis igal ajal ekraanile protsessitabeli ning teha järeldusi, milline on kinnijooksnud protsessi number. Teine asjakohane käsk on **kill**, millega saab maha tappa iga protsessi. Lihtsustatult võib öelda, et **kill** lihtsalt kustutab vastava nime protsessitabelist. Mõnikord on selline laustapmine ka ohtlik --- näiteks **kill -9 29587** tapab armutult ja aega viitmata protsessi, mille PID=29587 , küsimata temalt, kas ta oma failid kinni pani ja muud maised tegemised tehtud sai. Seevastu **kill -15 <PID>** tapab veidi "viisakamalt".

Isad ja emad, pojad ja tütred.

Vaadeldes protsessitabelit täies ulatuses (**ps -ef** või siis mõnedes süsteemides ka **ps -ax**), saate Te näha, et iga protsessi puhul on kirjas ka tema vanemate koordinaadid (PPID -- *Parent Process ID*). Näiteks käivitasite Te just nüüdsama käsu **ps**. See kiikas korraks protsessitabelisse, tõi tulemused ekraanile ning andis juhtimise tagasi teda sünnitanud protsessile, (antud juhul oli selleks Teie login-shell). Seda poleks saadud teha, teadmata PPID numbrit. Võimalik on ka teisesuunaline side -- oma login-shelli **kill**-käsuga tappes tapate te automaatselt ka kõik muud protsessid, mis temast pärinevad.

UNIX pakub protsesside loomisel päris huvitavaid võimalusi. Kui Teil on soov käivitada mingi pikem arvutusprogramm ja ise hoopis koju magama minna, siis andke käsk:

```
nohup proge_nimi
```

Nüüd lõigatakse protsessitabelis läbi side protsessi ja tema ema vahel. Ema tapmine ei põhjusta enam tütre surma. Te võite ennast rahulikult masinast välja logida, Teie programm aga jätkab tegevust (**nohup** -- *no hang up*). Soovi korral saab sünnitada kaksikuid, selle nähtuse nimi on fork ehk kahvel ning tüüpiliseks käsuks, mis seda nõksu kasutab -- **tee** (tee on oma nime saanud T-tähe kuju järgi):

```
write john | tee faili_nimi
```

Kui John sel hetkel masinas asub ning soovib Teile samaga vastata, saate ekraani ja klaviatuuri vahendusel juttu vesta, kusjuures kõik, mis Teie ekraanil toimub, salvestatakse täiendavalt ka faili.

Viimane trikk, mida me siinkohal käsitleme, on seotud programmi panekuga background'i . Kirjutades käsurea lõppu "&"-märgi:

```
proge_nimi&
```

lõikate jällegi läbi sideme ema- ja tütarprotsessi vahel. Te saate koheselt tagasi oma login-shelli prompti, tütarprotsess (Teile teatatakse selle number) aga jätkab iseseisvat tegutsemist. Sel võttel on mingi väline sarnasus DOSi TSR-programmidega. Ärge kunagi pange background'i protsessi, mis ekraani risustab. Häda on selles, et kui ta risustab Teie ekraani piisava kiirusega, siis ei jõua Te fikseerida selle õnnetu protsessi numbrit ning Teil jääb üle kirjutada kas **exit** või **kill -9 0**.

Igaks juhuks on masinasse ka piir seatud, mis ei lase protsesside arvul lõpmatusse tõusta. Harju keskmisel UNIX-süsteemil on protsessitabeli pikkuseks mõnisada rida ning ühel kasutajal pole luba käivitada üle 25

protsessi korraga. Edasi tulevad teated: *No more processes, Region Table Overflow ...*

Kest tema ümber

UNIXi keskpunktiks on tuum (*kernel*). Süsteemi üleslaskmisel lisandub poole kuni megabaidise suurusega op-süsteemile veel mitmesuguseid mäluosasid, mistõttu kogu kerneli poolt haaratav mälumaht võib ulatuda päris mitme megabaidini. Tuumaga on väga lähedalt seotud hulk draivereid, mis võimaldavad konkreetsete välisseadmetega suhelda. See osa UNIXist on täiesti individuaalne, ning lastud kokku täiesti konkreetse raua peale, ning seetõttu pole mingit mõtet üritada seda teist tüüpi raua otsas käivitada. Op-süsteemi koossseisu kuulub terve pesakond utiliite (**cp**, **rm**, **ps**, **who**...), mis annavad tuumale tõelise väärtuse. Enamikku utiliitidest saab kohapeal kokku kompileerida ning nende lähtetekst pole raua suhtes kuigi kriitiline. Utiliite on korralikus UNIXis vähemalt paarkümmend megabaiti. UNIXi, eriti SYSTEM V loosungiks on mitte leiutada jalgratast, vaid ehitada oma programm üles teiste poolt valmistatud tükkidele. Öeldakse, et üks korralik utiliit peab tegema **ainult ühte asja**, kuid peab tegema seda korralikult. Kogu selle utiliitide loomaaia kasutamiseks ja vaoshoidmiseks on igas UNIX-süsteemis programm nimega ***shell***. Shell võtab käsurealt (või tekstifailist) korraldusi ning tõlgib UNIXi tuumale arusaadavasse keelde. Elukutseline programmeerija võib ju suhelda tuumaga ka C-keelse progise ja *system call*'ide tasemel, aga enamik inimesi vajaksid nagu midagi lihtsamat. Muide, neile, kes ka käsureaga hakkama ei saa, on olemas X/Windows ja see maksab palju raha :-)

SHELL kui töökeskkond

Shell on peaaaegu seesama, mis DOSis COMMAND.COM , ehkki tema võimalused on tohutult suuremad. Tegelikult võib vaielda, kas shell ongi käsuinterpretaator või on ta hoopis programmeerimiskeel? Fakt on see, et iga kord, kui Te end UNIXisse sisse logite, sattute Te igatahes shelli prompti otsa, nii et shellist ei pääse Te mingil moel. Laisa süsadmini puhul on promptiks "%" või "$", ruudu (*root*'i) promptiks on shellist sõltumata "#". Veidike vaeva

nähes saate oma prompti palju sisukamaks kirjeldada, pistes sinna masina nime, *username*'i, kellaaja või kataloogi nime. Kui Teil parasjagu muud teha pole, võite oma prompti poole ekraani pikkuseks kasvatada ning ESC-koodidega vikerkaarevärviliseks muuta... Muide, kuna shell kujutab endast kasutajakeskkonda, mis peab konkreetse UNIXi erinevused kasutaja eest ära peitma, siis sealt on ta oma nimegi saanud: *shell* - koorik, teokarp.

Shelle on mitu erinevat nimetust, kuid algaja programmeerija ei peaks erinevusi esialgu märkamagi. Üks shell on süsteemi administraatori poolt seatud Teie login-shelliks -- so shelliks, kuhu Te satute kohe sisselogimise järel. Kui soovite, võite alati käivitada mõne teise shelli. Kirjutades shelli prompti otsa mingi korralduse ning vajutades ENTERit, annate Te oma lausekonstruktsiooni shellile hindamiseks. Kui ta leiab, et see on grammatiliselt õige, teeb ta asendused ja väärtustab muutujad ning annab siis kogu selle käsurea täitmiseks.
Näiteks, kirjutades:

proge <ENTER>

Te lihtsalt käivitate oma programmi nimega "proge";. Kirjutades

sh proge <ENTER>

määrate Te ühtlasi ära, et seda programmi käivitatakse Bourne'i Shelli (**sh**) abil. Siiski on võimalikud ka palju keerulisemad konstruktsioonid. Nagu näiteks:

```
kill -15 `ps -u root | awk '/uugetty/ && !/awk/ {print $1}'` > /dev/null
```

Sellest tuleks aru saada umbes nii, et protsessitabelist otsitakse välja kõik *root*'ile kuuluvad protsessid, leitakse rida, mis vastab käsule **uugetty**, eraldatakse **awk**-käsuga sellest reast PID-number, edastatakse argumendina **kill**-käsule ning lastakse see **kill** käima. Tulemuseks on muidugi programmi **uugetty** seiskamine. Kõik, mida sel käsul öelda on, suunatakse olematusse

(**> /dev/null**).

Vahelduseks üritage midagi sarnast teostada COMMAND.COMi vahenditega...

STDIN, STDOUT, STDERR ja toru

Kuni Teil pole erisoove, arvab shell, et kõik käsud ja korraldused tuleb võtta klaviatuurilt, kogu väljundvoog aga tuleb suunata ekraanile. Siiski saab mistahes protsessi sisendit ja väljundit ka ümber suunata, kui selleks tahtmine peaks tekkima. UNIXis on olemas säärane tore asi nagu "toru" (ingl.k. *pipe*), mille abil saab ühe programmi väljundi suunata järgmise programmi sisendisse, ilma et vahelt midagi ekraanile pudeneks. Nii näiteks käsukonveier (inglise k. *pipeline*)

```
cat yks_tekstifail | sort | tr "A" "b" | pr -30 | lp -or -depson
```

oleks täiesti võrdväärne jadaga

```
sort yks_tekstifail > teine_fail
tr "A" "b" teine_fail > kolmas_fail
pr -30 kolmas_fail > neljas_fail
lp -or -depson neljas_fail
```

(Pange tähele, faili suunatakse märgiga **>**, programmi standardsisendisse toruga **|**). Eelnenud näitest aga tuleks aru saada umbes nii, et üks_tekstifail sorditakse reakaupa, vahetatakse temas kõik suured A-tähed väikeste b-tähtedega, paigutatakse see kenasti 30 rea kaupa leheküljele ning lastakse Roman-kirjas trükki Epson-printerile. Käsk **tr** tähendaks *translate*, **pr** -- *print*, **lp** -- *line print*.

Niisiis on toru justkui voolik, mille abil saab erinevaid käske kokku ühendada. Loomulikult ei takista Teil keegi vahetulemusi failidesse salvestamast (kui Winchesteri kõrin Teile rõõmu teeb):

```
cat file | tee file |
```

```
sort | tee file | tr "A" "b" | tee file | ja nii edasi...
```

Siiski on igal programmil veel üks väljundkanal, mis lihtlabasele ümbersuunamisele ei allu. See on veavoog ehk *standard error* (**stderr**). Täiesti mõistetav, et kui konveieris midagi vussi läheb, sooviksite Te veateadet ikka ekraanil näha, mitte sordituna ja tõlgituna väljundfailis või dokumendi ridade vahel. Veavoog ilmutab mingeid teateid üksnes siis, kui Teie programm või käsurida oma tööga normaalselt hakkama ei saa.

Eriti enesekindlatele häkkeritele soovitaksin programmi suu sulgemiseks järgmist käsurida (suunab tühjusse nii **stdout** kui **stderr**):

```
sh enesekindel_proge > /dev/null 2>&1
```

Mis tähendab, et veavoog (1) suunatakse sinnasamasse, kuhu standardväljund (2). Seega siis kuhu? :-)

SHELL-script

Ei ole midagi uut siin ilma peal, shell-script on peaaegu täpselt seesama, mis *.BAT fail MS-DOSis. Selle pisikese erinevusega, et Teil on lubatud kasutada suvalisi muutujaid, tsükleid, IF-lauseid, CASE-konstruktsiooni ja palju muud, mida *batch-failid* iialgi suutma ei hakka. Nii näiteks kasutan ma UNIXi kataloogipuust ülevaate saamiseks järgmist scripti:

```
# Selle proge nimi on /u/anto/bin/tree
tyhik='.######'
algus='+-----|'
slash='/'
for i in `ls`
    do
      if test -d $i
      then (t=$tyhik$t ; export slash ; export algus ; \
      export t ; echo $slash$t$algus$i ; cd $i  ;
/u/anto/bin/tree )
      fi
    done
```

Script on rekursiivne, st kutsub iseennast välja seni, kuni kõik varjulised nurgakesed kataloogipuus on läbi käidud.

Muutuja $tyhik='######' on kasutusel põhiliselt seetõttu, et minu poolt kasutatav shell keeldub jonnakalt vahet tegemast tühikute ja tühjuse (NULL) vahel. Pärast teisendan ma **tr** käsu abil trellid kenasti tühikuteks tagasi ning saadud pilt ei erine kuigi palju sellest, mida väljastab Norton-Tree. Ettevaatust - skript sellisel kujul **ei töötle** nn *symbolic link*e.

Konfiguratsioonifailid

MS-DOSi all on tarbijal teatavasti võimalik konfigureerida oma töökeskkonda failidega AUTOEXEC.BAT ning CONFIG.SYS . Ka UNIXis on olemas sarnase otstarbega konfiguratsioonifailid. Iga kord, kui Te end sisse logite, täidetakse failis **.login** sisalduv käsujada. Lisaks sellele, iga kord kui Te uue alamshelli välja kutsute, täidetakse fail **.profile** (Bourne'i ja Korn- shelli puhul) või **.cshrc** (C-shelli puhul). Nendes failides saate ära määrata muutujad $PATH ja $EDITOR ning kirjeldada oma aliased (va Bourne'i shell, mis aliasi ei tunnista). Kui Te arvate, et kataloogi vaadatakse käsuga **dir**, mitte **ls -la**; , siis kirjutage oma **.login**isse:

```
alias dir 'ls –la'
```

Siiski hakkab see muudatus mõjuma alles järgmisest sisselogimisest alates. kui soovite kohest efekti, andke sama käsk käsurealt. Oma töökeskkonnast ülevaate saamiseks kasutaage käske **env**, **@** ja **alias**.

Märkimisväärne on veel see, et kuna iga UNIXi kasutaja sattub peale sisselogimist erinevasse kataloogi (nn kodukataloog), siis on kõigi konfiguratsioonifailid täiesti sõltumatud. Kodukataloogist võib leida muidki initsialiseerimisfaile:

```
.newsrc     -- uudissüsteemi konfigureerimiseks,
.mailrc     -- postimasina häälestamiseks,
.exrc       -- tekstiredaktorite vi ja ex taltsutamiseks,
.xDefaults  -- graafilise keskkonna initsialiseerimiseks.
```

Allpool üks näide tüüpilisest SCO UNIXi **.login**-failist:

```
set ignoreeof
set history=20
alias p lp -ob -or -oi $1
alias pg "pg -p \"---PAGER pg %d--->\" -s"
alias h history
echo "Terminaal $LOGTTY on kirjeldatud kui $TERM"
echo "Terminfo tarvitab kataloogi $TERMINFO"
alias lock lock -v
alias news "news | pg"
alias setprompt 'set prompt="`pwd` \! >"'
alias cd 'chdir \!* || setprompt'
setprompt
mesg y
```

Failisüsteem

(*Austus eraomanduse vastu*)

UNIXi failisüsteem võrdluses MS-DOSi omaga oleks umbes sama, kui võrrelda omavahel elevandi paindlikku lonti ning kolme liigendiga puujalga. UNIXi failidel on tunduvalt enam parameetreid. Faili üheks kõige olulisemaks parameetriks on tema kuuluvus. Igal failil on omanik (*owner*), kellel on selle faili suhtes eriõigused (näiteks õigus kustutada isegi, kui kustutamine on keelatud). Järgmiseks eristab UNIX omaniku sõpru. Igal UNIXi kasutajal on lisaks *username*'ile veel grupitunnus. Enamasti valivad ühisee projekti kallal töötavad inimesed endale ühe ühise grupitunnuse (näiteks **staff, research, bookkeep** või millise iganes kuni 8-tähelise nime).

Nüüd osutub võimalikuks faile valikuliselt kättesaadavaks teha -- mõned omanikule, mõned tema grupile, mõned aga kõigile soovijaile. Loomulikult saab üks kasutaja vaheldumisi võtta erinevaid grupitunnuseid nende hulgast, mis administraator talle kasutamiseks on võimaldanud.

Teine tähtis mõõde UNIXi failisüsteemis sisaldab endas infot selle kohta, kas faili on lubatud **lugeda** (**read, r**), kas tema kallal tohib kasutada salvestus- ja kustutusoperatsioone (**write, w**) ning kas seda faili tohib käivitada (**execute,**

x). Täiesti normaalsena mainigem olukorda, kus mingi shell-script on omanikule salvestamiseks kinni (et kogemata ei kustutaks), sõpradele lahti vaid lugemiseks (las kopeerivad oma koju ja lasevad seal käima), võõrastele aga üldse kinni. Sellise faili loabitid oleksid järgmised:

```
-rwxrwxrwx  Need oleksid maksimaalsed õigused
-r-xr-----  Need on vaadeldava faili loabitid
 ^^^        See on omaniku õigusi tähistav kolmik rwx (r-x)
    ^^^     See on grupi õigusi tähistav kolmik   rwx (r--)
       ^^^  See viimane kolmik kehtib ülejäänutele rwx (---)
```

Loabitte on programmeerijal eriti mugav meeles pidada arvu astmetena: r=4, w=2, x=1 . Ülaltoodud näites on esimese (omaniku-)kolmiku väärtuseks 5 (r+x), grupi kolmiku väärtuseks 4 (r) ning muidumeeste kolmiku väärtuseks 0 . Need numbrid yksteise järel näeksid välja niimoodi: **540** . Lühidalt, kui soovite, et Teie failil oleksid just sellised loabitid, nagu just esitatud näites, andke käsk:

chmod 0540 <failinimi>

UNIX hoiab iga faili juures kolme kuupäeva koos kellaajaga. Üks neist on loomise aeg (*Creation Time*) ehk tunnusbittide muutmise aeg, teine faili sisu muutmise aeg (*Modification Time*), kolmas aga *Access Time*, mis annab aimu, kuna faili viimati kasutati. Tavaline kataloogilisting esitab *Modification Time*'i, kuid eri võtmetega on võimalik kätte saada ka ülejäänud kaks daatumit. Mõned nuditud UNIXid ei erista loomise ja muutmise aegu. Failiaegu loetakse alates 1. jaanuarist 1970 kella 0:00 Greenwichi järgi ning tehakse seda sekundi täpsusega. Miks just Greenwichi järgi? Aga seepärast, et UNIX suhtleb vahel omasuguste masinatega teistest ajavööndidest ning teeb aeg-ajalt katset mitmesuguseid daatumeid kohaliku aja peale ümber rehkendada. Eesti dekreediaja tähistuseks on **EDT**, suvel on see sama mis GMT-2, talvel GMT-3.

Kataloogid

UNIX vaatleb katalooge tavaliste failidena, millel on olemas kõik needsamad loabitid. Kataloogi tunnuseks on täht "d":

```
drwxr-xr-x  1 omanik   grupp     DATE    LENGTH  FILENAME
```

See on kataloog, mis kuulub kasutajale **omanik**, kõigil teistel on keelatud sinna midagi juurde salvestada või olemasolevat modifitseerida. Kataloogi sisenemine ja failinimestiku saamine on kõigile lubatud. Kataloogi puhul saavad tähed **r**; ja **x** veidi teistsuguse tõlgenduse -- **r** tähistab õigust kataloogi listingu saamiseks, **x** aga õigust kataloogi sisenemiseks. Nii on võimalik luua olukord, kus alamkataloogis sisalduvat programmi saab käivitada üksnes see, kes *A)* teab et seal on *executable* programm; *B)* teab programmi täpset nime.

Teatud avalikel kataloogidel seisab **x**-tähe asemel hoopis **T**-täht. See märgib kitsendust, kus kataloogi sees saab faili maha kustutada üksnes tema omanik, väitku siis loabitid mida tahes.

Iga UNIX-süsteemi administraator on seadnud teatavad vaikimisi väärtused, millised bitid seatakse vastloodud kataloogidele, (kui kasutada käsku **mkdir**, tavaliselt drwxr-xr-x) ning vastloodud failidele. Kui teile *sysadmin*'i valik ei meeldi, seadke omaenda mängureeglid käsuga **umask**, näiteks **umask 022** failis **.login** on vajalik, kui Te ei soovi oma failide kallal näha kedagi peale iseenda.

Failisüsteemi võimalustest

UNIXi failinime pikkus pole põhimõtteliselt piiratud. Enamik realisatsioone võimaldavad kuni 255-märgilisi failinimesid. Nuditud versioonidel on nime pikkuseks 14 märki, väidetavalt on olemas süsteeme, mis lubavad isegi tuhandeid märke sisaldavaid nimesid (kuigi, kas sel on erilist mõtet). MS-

DOSi seisukohalt on vaja märgata, et failinimes tohib kasutada nii suuri kui väikesi tähti, kusjuures süsteem peab selliseid nimesid erinevateks. UNIX ei tunne laiendi mõistet. Te võite ju lisada oma faili lõppu **.EXE** , UNIXil pole sellest sooja ega külma. Temal on vaid üks käivitamise kriteerium -- **x**-biti olemasolu. Kui Te seate **x**-biti püsti oma armastuskirjale, siis saab seda käivitada koos kõigi sellest tulenevate tagajärgedega. Allpool näide, milline võib olla failinimi UNIXis:

```
total 8
drwxrwxrwt   2 mmdf     mmdf          32 Jun 04  1992 :saved
-rwsr-x--x   1 666      59810          0 Apr 07 12:08
EXECUTABLE
-rw-------   1 root     other          7 Apr 07 12:07 See on
üks experimentaalne_FAILINIMI-mis.DeMoNsTrEeRiB! UNIX'i
failisysteemi omadusi,.<>?~$%^&*()_+
-rw-------   1 root     other          0 Apr 07 12:10 dirlist
-rw-------   1 loll     group        284 Apr 04 14:43 loll
-rw-------   1 root     other          0 Apr 07 11:59 root
-rw-------   1 sys      sys            0 Mar 31 21:30 sys
-rw-------   1 uucp     uucp         602 Apr 06 23:45 uucp
```

Siiski on sümboleid, mille toppimist failinimedesse peaks vältima. Esmajoones on need kaldkriips **/** -- UNIX kasutab kataloginimede vahel just sedapidi kaldkriipsu -- ning tärn *****. Kustutades faili nimega ***** võib juhtuda midagi hoopis hullemat :-) . Samasuguseid sekeldusi tekib miinusmärgi puhul. Pole midagi lihtsamat, kui luua fail nimega **-r** , kuid eks katsuge sellest lahti saada! Shell tõlgendab failinime **-r** käsu **rm** võtmena ning nõuab sinna taha veel &umul;hte failinime...

UNIX vaatleb failidena ka välisseadmeid. Kõik sellised seadmed alates ketastest ja lõpetades terminaalidega, on koondatud kataloogi nimega **/dev**. (Tavalise device-faili loabittide ees seisab täht **c** või **b**.) Prooviks kirjutame sõbra terminaaliekraanile sõgedusi käsuga:

```
echo "Tere Vasja!" > /dev/ttyA
```

(seda muidugi eeldusel, et see fail on Teile kirjutamiseks lahti). Firma SUN tääjaamadel on kasutusel UNIXi versioon, mis peegeldab **/proc** kataloogi isegi masinas toimuvad protsessid. Vabalt hangitavas Linuxis on **/proc**

failisüsteem kenasti olemas. Soovi korral võite minna **/proc** kataloogi ning näha kõiki protsesse koos omaniku nime ja loabittidega.

Eraldi kokkulepe on seotud failidega, mille nime esisümboliks on punkt (**.**). Tavaline kataloogilisting (**ls** , **ls -li**) sääraseid faile ei näita. See on mugav viis varjata tavakasutaja eest faile nimega **.** (mis tähistab sedasama kataloogi, kus Te parasjagu olete) ning **..**, (mis tähistab ülemist, nn *parent*-kataloogi). Ka enamik konfiguratsioonifaile varjavad endid punkti abil, neid saab nähtavale tuua käsuga **ls -la**.

Võrreldes MS-DOSiga

MS-DOSi failistruktuur on parasjagu puine sellest küljest, et Te saate ühte kataloogi luua vaid piiratud hulga faile (konkreetne arv oleneb ketta tüübist). 720 KByte flopide puhul on maksimumiks veidi üle 100 faili. Rohkem pole failitabelis lihtsalt ruumi, kavaldage palju tahate! Samasugune lugu on faili maksimaalse lubatud pikkusega. Ketta tüübist olenevalt lõpeb FAT varem või hiljem füüsiliselt otsa.

UNIXi failisüsteemis on need piirangud suhteliselt kaugemale liigutatud. UNIX peab eraldi arvestust vabade blokkide (*Free Data Blocks*) ning vabade ***i-node*'ide** (*Free i-node List*). Termin *i-node* [i:noud] on midagi täiesti spetsiifilist, millele ei julge isegi eestikeelset vastet pakkuda. See oleks vast iga faili isiklik FAT, kui tohib termineid nii vabalt ümber sõnastada.

Oletame, et Te plaanite luua hästi pika faili. UNIX eraldab Teile ühe *i-node*'i vabade *i-node*'ide nimekirjast. Iga *i-node* sisaldab infot faili omaniku, loabittide ja kõige muu säärase kohta. Sellele järgneb koht, kuhu salvestatakse esimese andmebloki numbrid (midagi FATi taolist). Bloki suuruseks on enamasti 1024 baiti, ehkki pole mingeid takistusi loomaks 512 või 2048-baidiste blokkidega failisüsteeme. Niisiis saab sel viisil maha märkida faile, mille maht on kuni 10240 baiti (10 blokki a' 1024 baiti).

i-node'i 11. positsioonile salvestatakse viit (*pointer*) blokile, mis sisaldab

omakorda 1024pointerit andmeblokkidele. Sellise "teist järku" bloki kasutamine võimaldab viiteid kuni 10+256 andmeblokile a' 1024 baiti, nii et kokku saab maha märkidakuni 272384 baidi pikkusi faile. Kui ka sellest ei piisa, siis vastava *i-node* 12. positsioonile salvestatakse pointer, mis osutab "e;kolmandat järku blokile"e;, tolles on omakorda viited 256 "teist järku" bloki kohta. Taoline mehhaanika võimaldab ära mahutada faile, mille pikkuseks on 1024*(10+256+256^3)=67.381.248 baiti. Ning lõpuks, kui ka sellest väheks jääb, siis sama *i-node* 13. positsioonil paiknev "neljandat järku" pointer teeb võimalikuks 1024*(10+256+256^2+256^3)=17.247.250.432 baiti pika faili salvestamise. Sellest 7 Gigabaidist peaks piisama ka kõige nõudlikumale häkkerile.

Analoogselt toimub kataloogide manipuleerimine. Näiteks SCO UNIX eraldab kataloogile vaikimisi ruumi 32 sissekande tegemiseks (sealhulgas failid **.** ja **..**, mis on vajalikud süsteemi enda tööks). Kui sellest ei piisa, eraldatakse täiendav teist järku blokk ja sissekannete (failinimede) arv kataloogis tõuseb kuni 288-ni. Põhimõtteliselt pole takistusi ka kolmandat ja neljandat järku blokkide hõivamiseks. Ometi ei maksaks erilise vajaduseta sääraseid mammutfaile ja katalooge luua. Igasuguste otsimiskäskude töö aeglustub ekspotentsiaalselt iga järgmist järku bloki jaoks. Ka ei pruugiks põhjuseta kasutada ülipikki failinimesid. Vaadake juuresolevat listingut (saadud käsuga **hd .**) ning te näete, kuidas lohisev failinimi ruumi raiskab.

```
0000 3a 18 2e 00 00 00 00 00  00 00 00 00 00 00 00 00  :...............
0010 e7 03 2e 2e 00 00 00 00  00 00 00 00 00 00 00 00  ................
0020 3b 18 3a 73 61 76 65 64  00 00 00 00 00 00 00 00  ;.:saved........
0030 f2 19 75 75 63 70 00 00  00 00 00 00 00 00 00 00  ..uucp..........
0040 f1 19 72 6f 6f 74 00 00  00 00 00 00 00 00 00 00  ..root..........
0050 93 1a 73 79 73 00 00 00  00 00 00 00 00 00 00 00  ..süs...........
0060 4a 1a 6c 6f 6c 6c 00 00  00 00 00 00 00 00 00 00  J.loll..........
0070 b9 18 45 58 45 43 55 54  41 42 4c 45 00 00 00 00  ..EXECUTABLE....
0080 ff ff 53 65 65 20 6f 6e  20 79 6b 73 20 65 78 70   ..See on üks exp
0090 ff ff 65 72 69 6d 65 6e  74 61 61 6c 6e 65 5f 46  ..erimentaalne_F
00a0 ff ff 41 49 4c 49 4e 49  4d 49 2d 6d 69 73 2e 44  ..AILINIMI-mis.D
00b0 ff ff 65 4d 6f 4e 73 54  72 45 65 52 69 42 21 20  ..eMoNsTrEeRiB! 
00c0 ff ff 55 4e 49 58 27 69  20 66 61 69 6c 69 73 79   ..UNIX'i failisy
00d0 ff ff 73 74 65 65 6d 69  20 6f 6d 61 64 75 73 69   ..steemi omadusi
00e0 ff ff 2c 2e 3c 3e 3f 7e  24 25 5e 26 2a 28 29 5f  ..,.<>?~$%^&*()_
00f0 7a 1a 2b 00 00 00 00 00  00 00 00 00 00 00 00 00  z.+.............
0100 ad 18 64 69 72 6c 69 73  74 00 00 00 00 00 00 00  ..dirlist.......
0110 25 1a 68 65 78 6c 69 73  74 00 00 00 00 00 00 00  %.hexlist.......
0120
```

UNIXi failisüsteemi ühe pisipuudusena võib märkida seda, et temas puuduvad vahendid kõvaketta kasutuse optimeerimiseks (et blokid paikneksid järjest). Samuti on probleeme kasutatud blokkide tagastamisega. Kui Te olete oma kodukataloogi üheainsa korra üle mainitud lävede kasvatanud (30 või 288), siis jääbki töökiirus selles kataloogis madalamale astmele, olgugi et seda põhjustanud failid on ammu kustutatud. Sellise probleemi vastu aitab *backup*'i tegemine ning failide taastamine *backup*-magnetlindilt. Korralikus UNIX-süsteemis tehakse *backup*'i sagedamini, kui kõvaketas väga ära jõutakse jupitada, nii polegi keegi kunagi hakanud Norton SpeedDisk'i -taolisi programme UNIXi jaoks tarvitama.

Mis läind, see läind

UNIXis pole mingit võimalust kustutatud faile tagasi saada. Kui süsteemis tegutseb paarkümmend kasutajat, siis antakse vabanenud blokid küllalt kiiresti järgmisele soovijale, nii et mõni sekund hiljem on Teie õnnetult kustutatud andmed juba üle kirjutatud. Siiski pakutakse suure kettamahuga süsteemidele kavalat lahendust, kus kustutamise asemel antakse failile mingi punktiga algav nimi ning hoitakse siis sellise olekus päev või kaks, et äpu kasutaja jõuaks oma faili tagasi küsida. Siingi päästab õigeaegne *backup*. Kui Te kogemata midagi maha kustutate, saab päev vanad failid alati magnetlindilt taastada. Muide, *backup* on tehtav täiesti automaatselt öisel kellaajal, tuleb vaid õige lindikassett vastavasse auku pista.

Kolmas osa

Linkimine

Ehk Must Maagia

Kindlasti olete tähele pannud, et mistahes fail teie kõvakettal kipub ikka olema seal, kus teda ei vajata. Loomulik asi, et töötamiseks teete endale tagavarakoopia sobivamasse kataloogi. Vahel veel mitugi. Kettaruum aga muudkui kulub ja kulub...

UNIX sisaldab endas suurepärast failide linkimise võimalust. Te ei pea enam failist füüsilist koopiat valmistama, piisab väikesest vihjest kataloogis. Ühel failil võib olla palju linke, näiteks 25 või enam. Huvitav on see, et kui ükskõik millist neist tavalisena näivatest failidest muuta, siis muutuvad sünkroonselt ka ülejäänud 24.

Kui teil aga tekib tahtmine oma infost üldse lahti saada, siis tuleb järjest kustutada kõik 25 viidet sellele failile.Sealjuures pole enam võimalik kindlaks teha, milline neist oli esimene ja "õige". Kettale on fail ikka jäädvustatud ühes ja ainsas eksemplaris, ainult et tänu linkimisele paistab ta olevat paljudes kataloogides korraga.

Kataloogilistingut uurides tähistab linkide arvu faili omaniku nime ees seisev

arv. Kui tekib mingeid kahtlusi, siis saab listingut nõuda sellisel kujul, mis sisaldab ka inode'i numbri (**ls -li**). Kui kahe faili ees seisab sama inode'i number, siis pole kahtlust, et on rakendatud linkimiskäsku.

Linkimiskäsku esineb kahe nime all -- üsna haruldane **link** ning palju sagedasem **ln**.

Mõned programmimeistrid on linkimisest lausa teaduse teinud. Tõime kusagilt Internetist kohale GNU pakkija gzip algtekstid (muide, GNU - Gnu is Not Unix, kuid gzip pakib kuni 2 korda tõhusamalt kui UNIXi muud pakkimisprogrammid). Kompileerimisel aga ei tekkinud kusagilt lahtipakkijat nimega gunzip, vaid see tuli tekitada käsuga **link gzip gunzip**.

Tundub küllalt müstilisena see, et üks ja sama programm saab täita kahte eri funktsiooni, olenevalt nimest, mida ta parasjagu kannab. Abi saab C-keele õpiku sellest kohast, mis väidab, et igale C funktsioonile edastatakse nullinda parameetrina tema enda nimi.

Tegelikult on ju nii kinni- kui lahtipakkimisalgoritmis suur osa ühiseid koodilõike ning linkimine annab tõhusa ruumikokkuhoiu.

Ülalöeldu puudutas linkimise lihtsamat varianti nimega *hard link*. Hard lingi puhul peab viidatav fail alati kohal olema ning ka samas failisüsteemis paiknema. Soft link (ehk *symbolic link*) on veidi vabam viide. Kindlasti olete näinud mõnda DOS'i mängu (Retal näiteks), mis nõuab installeerimiseks C-ketta peakataloogi. Kui UNIX'i all säärane asi peaks juhtuma, kirjutatakse

ln -s /mingi/hoopis/teine/kataloog /nõutav.nimi

ning probleem ongi lahendatud. Ka kõige nõudlikum programm jääb säärase pettusega rahule. Soft linkidel on see omapära, et saab viidata ka teises failisüsteemis paiknevale failile ja kataloogile ning seejuures ei pea viidatav fail ilmtingimata eksisteerima.

Siiski ei maksaks teha ristlinke ega ka linkida kataloogi iseendaga, missugune tegevus mõned lihtsameelsed utiliidid kinni jooksutab.

Viimase moeröögatusena mainiksin nn hüperlinke, mis pole küll pelgalt UNIXi pärisosa, kuid nad moodustavad loogilise jätku eelnenule. Mida teha siis, kui fail, mida te soovite linkida, asub hoopis teises masinas (ja võimalik, et isegi teisel mandril)? Kui te soovite oma dokumendis viidata kellegi teise autori elektroonselt kättesaadavale tekstile, siis teatud programmid lubavad kasutada hüperlinke. Üks Harju keskmine hüperlink näeks välja selline: http://www.eenet.ee/index.html . See tähendab, et kui masin tekstis selle kohani jõuab, tiritakse arvutivõrku pidi masinast www.eenet.ee kohale fail nimega index.html (**html** - *HyperText Macro Language*). Loomulikult eeldab hüperlinkide kasutamine juba hoopis uut tüüpi tehnoloogiat (Interneti otseühendust ja hüperlinkide keelt mõistvaid programme).

Failisüsteem

DOSi puhul saab ühele kõvakettale luua mitu partitsiooni (veel üks kohmakas eestikeelne vaste, originaalis *partition*). Esialgu arvati, et kõvaketast on mõtet jaotada kuni neljaks osaks (*primary partitions*), siis aga asi arenes ning tekkis mõiste *extended partition* . UNIXi puhul vastaks igasugustele partitsioonidele paremini hoopis failisüsteemi mõiste, tundubki, et mikrode ajalugu on need kaks mõistet lootustult sassi ajanud. Lihtsama UNIXi kõvakettal on niisiis vaid kaks objekti -- *root file system* ning *swap partition*. Esimene sisaldab kõiki süsteemi üleslaskmiseks elulisi programme, teine on kasutusel virtuaalse mälupikendusena. Veidi keerulisemates süsteemides lisandub nendele eraldi failisüsteem süsteemi kasutajatele (user file system, mis traditsiooniliselt paigutatakse kataloogi **/u** või **/wrk** . Eraldatud failisüsteemide mõte on selles, et kui voolukatkestuse või administraatori vea tõttu mõni neist kannatada saab, siis on võimalik ülejäänud failisüsteemidega ikka edasi töötada. Pole ülearune lisada, et PC masina puhul on võimalik samale kõvakettale luua ka DOSi failisüsteem (*partition* :-)). Spetsiaalsete käskudega saab sealseid faile kasutada ning enamasti isegi masinat DOSi alt bootida, kui tõesti enam ilma hakkama ei saa.

UNIX-süsteemi bootprotsess on palju paindlikum, kui DOSil. Boot-prompti taha on võimalik kirjutada, milline failisüsteem võtta root'iks, kui suurt mäluosa kasutada ning palju muud. Kui UNIX edukalt üles läheb, vaadatakse, milliseid failisüsteeme on vaja täiendavalt kättesaadavaks teha, kontrollitakse nende korrasolekut ning mounditakse root-failisüsteemi külge. UNIXi seisukohast on peaaegu ükskõik, millisel füüsilisel kujul mingi failisüsteem on esindatud, kas paikneb see ühel paljudest kõvaketastest, floppil, CD-ROMil või hoopis teisel pool maakera asuvas Internetiga ühendatud masinas.

Mount it!

Nüüd aga mountimisest. See termin on pärit vanade vahetatavate kõvaketaste ajajärgust. Mitte alati pole vindimahtu sadades megades mõõdetud. Veel 15 aastat tagasi läks iga suurema masina juurde vaja paari näitsikut, kes lugesid paberlindilt teateid stiilis "Pista sisse kõvaketas nr. 37" ning tegid, mida kästud. Kui siis nõutud kettaplokk õnnelikult vastavasse auku oli tõstetud ja mootori tuurid üles võetud, võiski masinale teatada, millise virtuaalse kettaosaga on tegemist. Kui näiteks sellel füüsilisel kettaplokil leidusid programmid, mille koht oli kataloogis **/usr/bin**, siis "maunditi" kõvaketas kataloogi **/usr/bin** külge, seevastu andmed võidi "mauntida" kusagile **/clients/data** kanti. Selge ka, et korraga mahtus kettaauku istuma ikka vaid üksainuke kettaplokk. Seepärast, kui /usr/bin oli kasutuses, siis /clients/data kataloogilisting sai sel ajal näidata vaid totaalset tühjust ja vastupidi.

Ajad läksid edasi, vindid muutusid 1000 korda väiksemaks, mount'imine kui käsk ja termin aga jäi. UNIXi puhul pole mingeid raskusi paigutada teine kõvaketas või mõni selle failisüsteemidest esimese alamkataloogi nii, et jääb mulje katkematust vindiruumist. Nagu just mainitud, saab kõvakettale külge mountida ka flopi, CD-ROM'i või muud kraami. Kasutajale tundub, nagu tegutseks ta ikka oma ainsal ja armsal kõvakettal. Raamatupidaja võib oma strateegilisi andmeid lausa flopil hoida, vajadusel siis flopit mõnesse sobivasse alamkataloogi mountides. Töö lõppedes võetakse ketas koju kaasa

ning mingeid turvaprobleeme ei saa tekkida.

Tõelisel häkkeril pole nüüd raske aru saada, et ka kõvaketast saab flopi külge monteerida, kui selleks vajadus peaks tekkima. Ning loomulikult on võimalik oma vindi külge "kinnitada" võõra masina kataloog, kui teil mõlemal ikka Interneti ots toas ning too soovitav kataloog ka krabamiseks lahti. Taolist asja nimetetakse **NFS** (Network File System), käsusüntaks aga käib umbes sedasi:

mount -t nfs kadri.ut.ee:/export/ftp/pub /home/mina/FTP

Sellest hetkest alates tekib minu isiklikku kataloogi nimega FTP kõik seesama, mis sisaldub Tartus "kadri"-nimelise masina vastavas kataloogis. Loomulikult on failide kohalolek vaid virtuaalne ning nende kättesaamise kiirus sõltub suuresti ka kasutatava Interneti ühenduse jämedusest.

Lõpetuseks veel sedapalju, et oleks ilus alati anda peale töö lõppemist käsk **umount**. Vastasel korral riskite te oma failisüsteemi tervisega.

Deemonid ja tumedad jõud

UNIXist rääkides võib sageli kuulda väljendeid "UNIX teeb", "UNIX saadab, võtab, paneb, loob, kustutab, avab, sulgeb" jne. Umbes nagu oleks tegu elusa olendiga, kellel on oma nimi (UNIX-masinal ongi), keha, vaim ja isiklik hingeelu. UNIXi sisemust uurides ei leia te hingeelust jälgegi, küll aga avastate viis kuni viisteist deemonit, kes igaüks oma kitsas liinis asju ajab.

Kõige arusaadavam on kindlasti printerideemoni **lpd** olemasolu. Isegi M$ Windows'i all on võimalik korraga trükkida ja muud tööd teha. UNIXi **lpd** (mõnes süsteemis **lpsched**) on sedavõrd võimekam, et kord printerisse suunatud kirjatükk Teie edasist tegutsemist ei sega. Kui teil on masina küljes mitu printerit, siis saate nende vahel koormust jagada. Kõige huvitavam on ehk see, et trükkimiskäsk ei aegu. Teie printerideemon suhtub mõistvalt

igasugustesse riistariketesse, isegi kui vool kaob ja teie masin alles paari päeva pärast taas töökorda saab, jätkab lpd samast kohast, kuhu ta pooleli jäi.

Teine tähtis deemon on **cron**, mis vastutab automaatsete operatsioonide eest. Cron ei tunnista sekundeid. Üks kord iga täisminuti jooksul ärkab cron üles ning vaatab, kas spetsiaalsetes ajatabelites (*/usr/spool/cron/crontabs/**) leidub mõni käsk, mille kellaaeg klapib deemoni ärkamisajaga. Kui nii, siis läheb see käsk täitmisele. Crontab on eriti mugav keeruliste süsteemihaldamise käskude töölelaskmiseks ajal, mil see masina kasutajaid ei sega. Tavaliselt tehakse öösiti kokkuvõtted masina koormatusest ööpäeva lõikes, toimetatakse lindile **backup**, eemaldatakse /tmp kataloogist vanad failid jne. Põhimõtteliselt võib igal kasutajal olla oma isiklik crontab.

Sellega pole deemonite loetelu kaugeltki ammendunud. Küll eri nimede all (enamasti **sendmail**), kuid kindlasti igas UNIX-masinas elab postideemon. Selle ülesanne on näiteks iga 10 minuti järel kontrollida, kas kusagile pole takerdunud saatmata posti. Postideemon teeb järjekordse katse seismajäänud kirju minema saata, kui see aga ei õnnestu (ühendus katkenud vms), siis heidab järgmiseks 10-ks minutiks magama. Ühtlasi teeb postideemon sissekandeid vastavatesse failidesse, nii on alati võimalik kindlaks teha, miks post ei liigu või kes kuipalju kirju on saanud.

Tavaliselt on UNIXis käimas 1-2 deemonit, mis (või hoopis kes :-)) kõik vähegi olulised sündmused vastavatesse failidesse üles tähendavad (näiteks **syslogd**) ja **logger** (**klogd**). Kui masinas juhtubki midagi imelikku, siis on selle põhjusi alati võimalik hiljem uurida, lugedes faili **/usr/adm/messages** ja teisi samalaadseid. Tüüpilisi teated: ketas täis, avariiboot kell 00:15.

Eraldi deemonite seltskond on seotud Internetiga. Kui Teie arvuti on ühendatud ülemaailmsesse võrku, siis peab ta mõistma rahvusvahelist keelt. Pealikuks on deemon nimega **inetd**, kes(mis) korraldab sideseansse kui selliseid ning annab siis juhtimise üle mõnele deemonile, mille teenust parasjagu nõutakse:

- ftpd (failitranspordideemon, lubab faile ühest masinast teise nihutada),
- talkd (korraldab ekraanil vestlusseansi, kusjuures osalised võivad asuda eri mandritel),
- pcnfsd (korraldab UNIXi failisüsteemi kasutamist tavaliste PC'de poolt),

httpd (hüperteksti deemon, saadab tellitud hüperlinkide järgi faile) jne.

Ka lasub deemonitel püha kohustus hoida ohjes tumedaid jõude, kes UNIX-masina kallal oma jõudu proovivad. Võite kindlad olla, et iga soovimatu tegevuse jaoks leidub vähemalt üks deemon, kes tumedale jõule risti ette astub ning süsteemiadministraatorile ettekande koostab :-) .

Infoallikatest

Paljudel võib tekkida küsimus - kust hankida täiendavaid teadmisi UNIXi kohta? Ühest vastust sellele pole: kõik oleneb teie keeleoskusest, rahakoti paksusest ning ligipääsust töötavale süsteemile, mille kallal saab kätt harjutada. Päris korralikud UNIX-võrgud on olemas Tartu Ülikoolis, Tallinna Tehnikaülikoolis ning ka Pedagoogikaülikoolis. Kõigis neis kohtades toimib ka Interneti ühendus, mis teeb teile kättesaadavaks väga paljudes masinates leiduva info.

UNIX on jõudnud ka paljudesse tõsise tööga tegelevatesse asutustesse, alustades Eesti Pangast ning lõpetades Tallinna veepuhastusjaamaga. Seega on tõenäosus omaenda töö- või õppimiskohas UNIXi otsa komistada küllalt suur.

Suureks abiks võivad olla ka raamatud. Seni kõige põhjalikumaid eestikeelseid juhtnööre leiate EMI loengukonspektide sarjast, Loengumapist nr.1-28, autoriks T.Kadarpit ja ilmumisaastaks 1994. Ingliskeelsetest raamatutest tasub eelistada O'Reilly kirjastuse raamatuid sarjast Nutshell, mille hind pole kahjuks küll enamikele eestimaalastele taskukohane. Väga kasulik oleks sirvida raamatuid "UNIX. The Book" tema loojate Ritchie ja Kernighani sulest (olemas ka venekeelses tõlkes) ning "Introducing UNIX, System V", autoriteks Morgan ja McGilton.

Kõigist neist raamatutest on ka allakirjutanu aastate vältel malli võtnud. UNIXi käskude lühispikri leiate ajakirja "A&A" 1993. aasta juunikuu numbrist, täielikuma variandi loodetavasti ka järgmisest **.EXE**st.

Siiski tuleb kõige tõhusamaks ja kiiremaks abivahendiks tunnistada Internet. Spetsiaalselt algajate jaoks levitatakse vastuseid korduvalt küsitud küsimustele (KKK, inglise keeles FAQ - *frequently asked questions*). Kõrvuti kõigi teiste teemadega on vastused olemas ka UNIXi asjus tekkinud probleemidele, igakuiselt uuendatava versiooni leiate archie, veronica'i ja teiste võrguotsimisvahendite abil. Kindlasti on kasulik lugeda ka UNIXiteemalisi uudisgruppe Internetis.

Neljas osa

UNIXi 20 käsku

Kui suur on teadmistepagas, millega võib iseseisvalt UNIXit uurima asuda? Kindlasti on vaja teada üht-teist UNIXi ideoloogiast ja ülesehitusest. Samuti aga on olemas mingi minimaalne hulk käske, milleta lihtsalt pole võimalik UNIX-masinas tegutseda ega edasisi teadmisi omandada. Võimalik, et allpool toodud valik pole just kõige optimaalsem, kuid igatahes annab see võimaluse ka päris algajatele. Algul mõtlesin toime tulla kõigest 10 käsuga (kõlaks ju hästi -- UNIXi 10 käsku!), kuid asi ei tahtnud õnnestuda. Tundub, et 20 käsku on siiski miinimum, ja nende selgeksõppimisest tasubki alustada, enne kui masina juurde astuda.

0 login

login on üks neist vähestest programmidest, mida läbimata UNIX-masinasse lihtsalt sisse ei saa. Kui masin arvab, et ta on nõus Sind sisse lubama, viskab ta ekraanile sõna **login**. Sina pead selle logini järele kirjutama oma kasutajanime (*username*). Mõnikord, eriti väiksemates süsteemides, ongi kasutajanimeks sinu eesnimi. Suuremates asutustes, kus arvutisüsteemi kasutajaid rohkem, koguneb ka kindlasti mitu ühe ja sama nimega isikut. Et nende vahel kuidagi vahet teha, lisatakse eesnimele perekonnanime esitäht

(**peeter_k**), mõni number (**2mari**) või kasutatakse hoopis sügavama sisuga tähekombinatsioone nagu **magpie**, **tiritamm** või **sipsik**.

Mõnes süsteemis saab login-nime vabalt valida, teistes (eriti õppeasutustes) määrab selle süsteemi administraator Sinu näokuju või matriklinumbri järgi. Login-nimeks sobivad 4-8 tähe pikkused tähekombinatsioonid. Lühemaid lubatakse ehk vaid eriti lähedalseisvatele isikutele ning pikemaid ei suuda iga UNIX-süsteem välja kannatada. Mõnikord sisaldub login-nimes ka sidekriips või allkriips, kuid ei maksaks endale sellist nime tahta -- Sa võid sattuda häälestamata klaviatuuri taha, kus on vastavaid märke väga raske leida.

login-käsk kannab meil numbrit 0 seepärast, et tavalist tööd tehes polegi eales vajadust seda ise käima lasta. Masin käivitab logini automaatselt, kui mõne liini pealt midagi sisestama hakatakse.

1 passwd

passwd on käsk oma parooli vahetamiseks. Kui juhtub, et Sul oma parooliga igav hakkab või keegi selle kogemata teada saab, kirjuta käsurealt **passwd** ning Sulle antakse võimalus see täheühend uue vastu vahetada. Sõnade *Old password* järele sisesta oma vana parool (see on garantiiks, et Sinu äraolekul keegi teine Sinu parooli vahetada ei saa). Seejärel tuleb Sul 2 korda järjest sisestada tähekombinatsioon, mida Sa endale uueks parooliks soovid võtta. Kui Sul õnnestub 2 korda järjest sama täheühend sisestada, siis ongi vahetus toimunud. On masinaid, kus ei lasta paroole vabalt valida ning kasutajat sunnitakse valima juhuslikult genereeritud paroolide hulgast, ***quilaqreew*** on säärase parooli musternäidis. Mõned masinad on nii targad, et nõuavad parooli sees isegi numbreid, suuri tähti või kirjavahemärke.

Keeruline parool kaitseb Sinu enda huve. Ära kunagi pane parooliks oma nime, auto- või telefoninumbrit, ega ka mõnda muud sõna, mida saab sõnastikust leida. Asi selles, et kui masina üldise *security*'ga lood kehvad, siis saab pahasoovija sealt paroolifaili välja varastada. Paroolid on seal küll krüptitud (kodeeritud) kujul, näide: ***Oi5uwQ2Rq5vG***, ning lahtikodeerimine

võimatult raske. Kuid paroolimurdjal on veel teinegi tee -- ta hangib endale mitte lahti- vaid kinnikodeerimisprogrammi. Siis hakkab ta järjest nimesid, numbreid ja sõnastikust sõnu kodeerima, ise iga kord võrreldes, kas sai tulemuseks ***Oi5uwQ2Rq5vG*** või veel mitte.

Tavalise PC peal pole keeruline teha 100-1000 kodeerimist sekundis. Rehkenda nüüd ise, kui kaua kulub kogu sõnaraamatu läbikontrollimiseks ... Veel kiiremini läheb asi siis, kui muukija teab sinu koera nime, lemmikansamblit ja telefoninumbrit.

Seepärast vali parooliks midagi sellist, millest Sa vaid ise aru saad, kas mõne lause sõnade esitähed (***KAikjotja*** --- Kui Arno isaga koolimajja jõudis, olid tunnid juba alanud) või siis midagi hoopis isiklikku (***ksMmease*** -- kuule sina Maie, ma ei armasta sind enam). Ning ära jaga oma paroole välja --- kuulugu need samasse väärtuskategooriasse kui habemeajamismasin, hambahari, tüdruk ning ELT-kaart.

2 ls

ls -- *list*. ls-käsk on algaja UNIXikasutaja kannatlikkuse proovikivi. Seda käsku on võimalik anda nii mitmes eri variandis, et kõik nad niikuinii meelda ei jää. Lihtne **ls** näitab faili- ja kataloginimesid ühekaupa real.

ls -la näitab varjatud faile (**.** , **..** , **.login** , **.profile**).

ls -CF esitab failinimed tulpadena, **ls -t** ajalises järjekorras, **ls -R** rekursiivselt kõigis alamkataloogides jne. Kuna sellist lohisevat käsku nagu **ls -la** on raske sisestada, on paljudes masinates kas algselt või siis süsteemiadministraatori armust olemas käsud **l** (ls -l), **lf** (ls -CF) või isegi **lR** (ls -lR).

ls käsu väljund on tunduvalt keerulisem, kui DOSi dir käsul. Infot saab nii faili nime, pikkuse kui loomise (modifitseerimise, staatuse muutmise) aja kohta. Eraldi reas on omaniku nimi ja tema grupitunnus. Päris esimeses tulbas seisavad andmed faili staatuse ja kättesaadavuse kohta. Rida

```
-rw-r-----   1 anto    ise       312 Aug  8 08:30 minu.isiklik.file
```

iseloomustab faili nimega *minu.isiklik.file*, pikkusega *312* baiti. Seda faili on viimast korda muudetud 8. augustil *sel* aastal (muidu esitataks aastaarv). Selle faili omanikuks on *anto*, tema grupitunnuseks faili loomise hetkel oli *ise*. Fail on antole lahti lugemiseks ja kirjutamiseks (ka kustutamiseks), lahti *ise*-grupi liikmetele lugemiseks ning kinni kõigile ülejäänutele.

Arv 1 teises veerus märgib linkide puudumist.

Teine näide:

```
drwxrwxr-x   2 root     wheel        1024 Aug 30  1993 pub
```

Kataloog (d-täht rea alguses) nimega *pub* kuulub ruudule (root). Tema saab sinna salvestada (w) ja siseneda, *wheel* grupi tunnust omavad isikud saavad samuti tolles kataloogis kõike teha, kuid ülejäänutel pole lubatud seal midagi kustutada (puudub w). Kataloogi puhul tähistab 1024 kataloogi enda suurust, pikkade nimede ja hulga failide puhul on see arv suurem. Arv 2 viitab linkide hulgale.

ls puhul tuleb teada paari kavalat trikki. Et näha, mis sisaldub kataloogis */etc*, ei pruugi üldsegi sinna kohale minna. Piisab käsust **lf /etc** :

```
...
ftpusers                msgs/                   termcap
gateways                mtab                    ttys
gettydefs               mtools                  utmp@
group                   named.boot              wtmp@
zyzyy*
```

Kaldkriips nime taga osutab kataloogile, tärn tähistab käivitatavat programmi, **@** märgib mõnedes süsteemides linki. Kui Sind aga huvitavad loabitid kataloogil, aga mitte tema sisu, anna käsk **ls -lad**.

3 pwd

pwd -- *print working directory*. Kord masinasse sisse saanud, tahad Sa arvatavasti teada, kus kataloogis Sa asud. Tegelikult on süsteemiadministraatori teha, millise kataloogi ta sinu koduks kirjeldab. Sageli on selleks midagi */u/home/anto* taolist. Anna käsurealt **pwd** ning veendu selles! Sama käsku saad kasutada alati, kui oled kataloogide rägastikku ära eksinud.

4 cd

cd -- *change directory*. Selle käsuga saad kataloogi vahetada. Loomulikult peab cd'le järgnema vastava koha nimetus, kuhu sa minna tahad (pole ju mõtet öelda lihtsalt -- mine! --- kui võib öelda: mine metsa! Kui Sa **cd** järelt sihtkoha nime ära unustad, siis teeb UNIX järelduse, et Sa oled juba piisavalt purjus ning viib Su sinna, kus on joodiku koht -- koju.

Näiteid:

```
cd /usr/home/anto/texts

cd ../../../etc/rc.d  (mine kolm kataloogi ülespoole ja
siis
                       teisest kohast alla tagasi)

cd .                  ( jääd sinnasamasse kus Sa oled)

cd /usr/local/bin
```

Märka, et kataloginime saab anda mitmel kujul. Kui see algab kaldkriipsuga, siis hakatakse nime lugema *root*-kataloogist **/** (vastab DOS'i C-ketta peakataloogile :-)). Kui nime ees kaldkriipsu ei ole, siis hakatakse koha rehkendamisega pihta sealtsamast kataloogist, kus Sa hetkel asud.

5 mkdir

mkdir -- *make directory*. Teeb sedasama, mida Norton Commanderi **F7** DOS'i all. **mkdir** järel peab alati käima nimi, mida Sa tahad oma kataloogile anda. Katalooge on võimalik tekitada ka karjakaupa: **mkdir q w e r t y u i o p** . Vali oma kataloogidele mõistlikud nimed, seda enam, et UNIXi failinime pikkus on jupi maad pikem kui DOSis. Tee vahet suurtel ja väikestel tähtedel. Märkidest on lubatud kõik ASCII koodid peale ***** ja */* , ehkki ka miinusmärgiga võib teatud raskusi ette tulla. Võta endale reegliks, et üheski Sinu kataloogis poleks faile (ja alamkatalooge) rohkem kui ühele ekraanitäiele ära mahub.

6 man

man -- *manual*. UNIXis on help-käsk reserveeritud hoopis millegi muu jaoks -- abi Sa helpi küsides igatahes ei saa. Kasuta käsku **man**. Kõige kurvem on see, et paljudes masinates pole isegi loetelu kõigist käskudest, mille kohta on üldse helpi võimalik saada. Sõjakavalusena võid minna kataloogi */usr/man* ning kaeda mõnesse sealsesse alamkataloogi, kus pakitud/pakkimata helpfaile hoitakse. Eralda failinimest kõik, mis tuleb peale punkti ning küsi iga asja kohta man'i. Teine ja üldsegi mitte ohutu võimalus on lasta järjepannu käima kõik */bin* , */usr/bin* ja */usr/local/bin* kataloogides leiduvad käsud ning vaadata, mis juhtub :-) .

Paljudel käskudel on kombeks enda kohta natuke rääkida, kui nende taha mõni absoluutselt vale võti sokutada, näiteks --- (kolm miinust). Ka on mõt kontrollida, ehk on masinas olemas jutukamaid käske nagu GNU projekti **info** (**xinfo**) või **apropos** (tahab endale argumendiks mõnda võtmesõna).

7 file

Edasi tahad Sa kindlasti teada, milleks UNIXis seda või teist faili vaja läheb. Küllalt mõistliku, kuigi mitte alati õige vastuse annab Sulle käsk **file**. Sa võid küsida **file *** ning saada umbes sellise vastuse:

```
ESATT2:                 MMDF mailer spool file
Txt:                    directory
```

```
a.out:                Linux/i386 demand paged executable
not stripped
krisprog.c:           c program text
lynx_bookmarks.html: English text
mail:                 directory
passwd:               ascii text
word_dca:             Microsoft a.out separate pure
segmented word-swapped
                      386 executable
üksfile.txt:          empty
```

Arusaadavalt ei maksa loota, et masin mõne faili *Estonian text*'iks tunnistab. Varjatud failide kohta saab infot vaid erikonstruktsiooniga **file .*** :

```
.lessrc:                      ascii text (with escape
sequences)
.login:                       ascii text
.lynxrc:                      English text
.mosaic-global-history:       ascii text
.mosaic-personal-annotations: directory
```

8 finger, w või who

Vali ise, millise neist kolmest käsust Sa oma relvastusse võtad. Kõik nad annavad teavet isikute kohta, kes hetkel masinas tööd teevad. **finger** on neist ehk kõige universaalsem, lubades infot küsida ka nende kasutajate kohta, keda hetkel masinas pole. Küsi **finger sidorov** ning Sa saad infot kõigi *sidorovide* kohta, keda see masin tunneb. **fingeri** suurepäraseks omaduseks on see, et ta töötab ka Internetis kauge maa tagant. Küsi **finger @panix.com**, või **finger anto@siil.edu.ee** ning vaata, mis juhtub.

w näitab isegi, millega see või teine isik masinas tegeleb. Tema puuduseks või vastupidi -- eeliseks -- on see, et administraator saab tema kasutamist piirata. Osades süsteemides saab teiste tegevuse järele nuhkida vaid see, kel on olemas teatud grupitunnus või *authorization* -- (näiteks *mem*).

who väljund on lühike ja lakooniline -- see käsk sobib kasutamiseks programmisiseselt.

9 cat, less, pg või more

Kõik need on programmid, mille abil on võimalik tekstifaili sisu ekraanile tuua. DOSi analoogiks oleks käsk type. **cat** -- *(con)catenation* lihtsalt löristab faili üle ekraani nii et pikema faili puhul kaob algus silmist. **cat *** on eriti mugav moodus pisikeste failijupikeste kokkuliitmiseks (eriti kui väljund faili suunata). **more** on küllalt sarnane DOSi analoogse prorammiga, mis võimaldab faili ka ekraanitäite kaupa vaadata. **pg** on kasutusel vaid mõnes UNIXi versioonis ning tema juhtimine erineb veidi *more*'i omast.

Kõige täiuslikumaks tuleb ehk siiski lugeda programmi **less**, mis näiteks Linux'i peal võimaldab isegi ESC-koode vaadelda, lisaks veel ka faili sujuvalt reakaupa üles-alla kerida. Ära unusta **pg** või **more** taha failinime(sid) lisamast! Kui Sinu käsutuses olevas **more** versioonis puudub argumendi olemasolu kontroll, siis hakkab see programm Sulle neidsamu tähti viskama, mida Sa ise klaviatuurilt sisse tipid...

Täpsemalt loe (inglise keeles) **man more** või **man less** käsuga!

10 vi ja emacs või joe ning pico

Olles õppinud tekstifaile **vaatama**, tuleb Sul kindlasti tahtmine ka neid muuta ja ise luua. Kõigepealt pead Sa uurima, missugused ASCII-tekstiredaktorid on Sinu süsteemis olemas. Tegelikult leidub vaid üks tekstiredaktor, mis leidub absoluutse kindlusega mistahes UNIX-süsteemis -- **vi**. Ning just **vi** on oma loogika poolest algajale täiesti arusaamatu. Ometi pole **vi** eest pääsu. Sa pead teda oskama vähemasti niipalju, et suudad parandada ja kohandada mõne teise (Sinu arvates parema) tekstieditori algtekste kuni need Sinu masinas kokku kompileeruvad. Alles siis saad Sa mõnda muud redaktorit kasutama hakata --- olgu selleks võimas ja paindlik **emacs** või lihtsakoelisemad **joe** ja **pico** (eriti viimane on Eestis väga levinud).

vi puhul on äärmiselt tähtis, et Sa tema taha ka faili nime kirjutaksid, mida Sa redigeerida kavatsed. Vastasel korral saad Sa veateate *No current filename*

ning Sul tuleb **vi**'st väljuda kirurgiliselt **<ESC> <ESC> :q!** abil.

vi on eriti huvitav selle poolest, et temasse sisenedes ei saa Sa kohe oma teksti kallale asuda. Selle asemel ootab ta Sinult mingit ühetähelist käsku (näiteks **a**, **i** või **r** **a**=*append*, **i**=*insert*, **r**=*replace*). Nii et kui Sa **vi**'sse sisenemise järel vajutad 2 **a**-tähte järjest, siis esimest tõlgendatakse kui käsku *append mode* ning teist kui faili sisestatavat tähte. Alles nüüd tohid Sa oma teksti sisse tippida. **vi** ei tee Sinu eest reavahetusi, Sa pead ikka ise oma ENTERit vajutama.

Kui Sa nüüd mingil hetkel tunned, et tekst sai valmis, siis vajuta kindlasti ja tervelt 2 korda ESC klahvile. See viskab Su käsurezhiimi tagasi ning Sa saad ühetäheliste korraldustega oma teksti parandama asuda. Taas kehtivad käsud **a**, **i** ja **r**. Tegelikult on tähekorraldusi palju rohkem, kuid Sina pead alustuseks teadma vaid järgmisi:

```
a     - lisada uut teksti kursori järele
i     - pista teksti vahele kursori kohalt
r     - vahetada ära kursori kohal olev täht
        (ise ütled, mille vastu)
R     - kukutab Su pidevasse ülekirjutusmoodi, millest
saab
        välja vaid kaht ESC kasutades.
x     - kustutab kursori all oleva märgi. Ära unusta,
        et enne pead Sa 2 ESC abil käsurezhiimi tulema,
        sest muidu lähevad Su x-tähed
        kenasti faili ning kustutamisest pole juttugi.
dd    - kustutada terve rida, millel kursor parasjagu
asub.
P     - tuua ekraanile tagasi viimati kustutatu.
Loomulikult
        on võimalik miski tekst ühest kohast kustutada
        ning siis teises kohas jõlle vahele pista.
```

Kursori liigutamiseks ja tekstis ringiseiklemiseks on lisaks nooltele olemas ka terve hulk klahvikombinatsioone, neist mõnda on isegi kasulik teada:

```
$     - viib kursori rea lõppu.
0     - viib kursori rea algusse.
G     - faili lõppu
1G    - faili algusse.
```

Kui Sul **vi**'st isu täis sai, siis pääsed välja sellise käsujärjestusega:

```
ESC ESC ZZ  (2 escape'i, 2 suurt Zorrot).
```

vi on ehitatud kukile ühele hoopis ürgaegsemale editorile nimega **ed** (ka **ex**). Seepärast tunneb ta ka ed'i käske:

```
                  :q!  - väljuda muudatusi salvestamata.
                  :w!  - salvestada vahevariant.
:w! üksfilee.txt - salvestada seis vastava nimega faili
```

Kui Sa nüüd arvad, et saad kuidagi hakkama ka ilma **vi**'ta, siis on Sul õigus ning ei ole kah. **vi** valdamine UNIXi gurule on nagu aadlikule vapp. Ka ühegi UNIX-süsteemi administraator ei saa hakkama **vi**'d oskamata. Kui Sa kavatsed UNIX-masinat kasutada vaid postivahetuseks ja Internetti pidi hulkumiseks, siis ehk õnnestub Sul mingi aeg hakkama saada ka ilma **vi**'ta. Täiendavat (ka eestikeelset) infot **vi** kohta leiad Internetist.

Veidi ka teistest ASCII tekstiredaktoritest. **emacs**it on olemas kahes versioonis -- kohutavalt suurte võimalustega täispakett ning veidi pisem MicroEmacs, mille algtekstid mahuvad ehk isegi ühele floppile. Emacs on rohkem kui paljalt editor -- maxiversioonis sisalduvad isegi programmeerimiskeele LISP vahendid. Microemacs'il on olemas sisemine help, selle kallale saad märgijadaga **ESC A** ja seejärel sõna, mille kohta Sa helpi soovid saada. **ESC A A <ENTER>** toob nähtavale kogu sisemise help-menüü. Harjutamiseks võid MicroEmacs'i esialgu DOSi jaoks kokku kompileerida.

joe ja **pico** on veidi WordStar'i sarnased töövahendid algaja tekstirikkuja tarvis. Juba esileheküljel sisaldub viide help'ile, nii et siinkohal ei hakka seda dubleerima. Algajatele eriti sobiv riist on just **pico**, millel saab ekraani pooleks jagada ning help'i kogu aeg silme ees hoida. Kui oled endale kohase editori välja valinud, siis ära unusta seda kirjeldamast oma konfiguratsioonifailides (**setenv EDITOR /usr/local/bin/pico**).

11 mail, Mail, elm ja pine

Üks lõbusamaid tegevusi UNIXis on kirjade saatmine masina piires ja isegi väljapoole. UNIXiga käib kaasas lihtne postiprogrammike nimega **mail**. Et kirjutada Jürile elektronkiri, anna käsk **mail jyri**. Küsimusele *Subject?* vasta mõne sõnaga (lühikokkuvõte kirja sisust). Edasi toksi sisse oma läkituse tekst. Ärasaatmiseks anna uue rea esimese sümbolina **CTRL+D** (*End Of Transmission* sümbol on UNIXis faili või sisestuse lõputunnus) ehk siis **~.i** (tilde-punkt).

Ühetäheliste sisekäskude kirjelduse saad küsimärgi ja ENTERi abil või tilde-küsimärk (**~?**) abil juhul, kui Sa oled juba kirjutama asunud. Mõnes masinas võib leida sarnase mailiprogrammi, mille nimi algab hoopis suure tähega (**Mail**).

Siiski ei soovitaks oma kirjavahetuses **mail**'i tasemele jääda, vaid kasutada palju täiuslikumaid postiprogramme nimedega **elm** ja **pine**. Neid käivitades ilmub pisike help kohe ekraanile. Mugavaks teeb asja just see, et menüüd pidi nooltega liikudes saab kiirest vajalikku toimingut esile kutsuda. **elm**'i ja **pine** kasutamine on niivõrd lihtne ja intuitiivne, et alustamiseks abi praktiliselt vaja ei lähe. **pine** on ehk veidi täiuslikum kui Elm, ehhki täisväärtuslikku tööd saab teha mõlemaga.

Elektronkirjade puhul pole eriliseks kunstiks mitte nende lugemine ja kirjutamine, vaid hea toon -- netikett (*netiquette*) ning arusaamine mitmesuguste päiste (headers) otstarbest. Viisakusreeglid võib kokku võtta lühidalt -- ära kirjuta ligemesele seda, mida Sa oma kirjakastis näha ei sooviks. Arvesta, et on inimesi, kes oma töö iseloomu tõttu on sunnitud lugema kuni paarsada kirja päevas. Enne kirjakirjutamist mõtle, kas ehk ei saaks kuidagi hakkama ilma posti saatmata.

Päistest vaid niipalju:

```
From anto@peak.edu.ee Wed Nov 16 12:50:27 1994
Return-Path: anto@peak.edu.ee
Received: from anubis.kbfi.ee (anubis.kbfi.ee
[192.121.251.14])
 by ogalik.edu.ee (8.6.9/8.6.9) with SMTP id MAA00415 for
 <anto@ogalik.edu.ee>; Wed, 16 Nov 1994 12:50:22 +0200
Received: from peak.edu.ee by anubis.kbfi.ee with smtp
        (Smail3.1.28.1 #5) id m0r7hmF-0003oHC;
        Wed, 16 Nov 94 12:39 EET
Received: by peak.edu.ee (5.0/SMI-SVR4)
        id AA03221; Wed, 16 Nov 1994 12:41:27 --200
Date: Wed, 16 Nov 1994 12:41:27 --200
From: anto@peak.edu.ee (Anto Veldre)
Message-Id: <9411161041.AA03221@peak.edu.ee>
X-Within-Url:
http://www.cl.cam.ac.uk/users/iwj10/linux-
faq/section7.html
To: anto@ogalik.edu.ee
Subject: section7.html
content-length: 6562
Status: RO
```

Nagu näha, võib kirja päisest välja lugeda kõik selle kirja valmimise ja kirjutamise asjaolud, üht-teist isegi postiedastusprogrammi ning kasutatava sideliini kohta. Kui tekib vähimgi huvi saadud kirja autentsuse või saatja kohta, vaata alati päiseid!

12 ps

ps -- *processes*. Käsuga **ps** saad Sa näha ülevaadet protsessidest, mis antud hetkel masina jõudu röövivad. Käsul **ps** on arutu hulk võtmeid, mille kasutamine annab täiendavat infot protsesside omanike, kulutatud masinaaja jms sellise kohta.

```
  PID TTY STAT  TIME COMMAND
   43  ?  S     0:01 /usr/sbin/inetd
   45  ?  S     0:00 /usr/sbin/lpd
   50  ?  S     0:03 sendmail: accepting connections
 1739 v02 S     0:01 -tcsh
 2074 v02 S     0:00 vi tigu4
```

Nii näiteks kirjutan ma praegu Linuxi teise multiscreen'i peal neid ridu ning vastava protsessi numbriks (**PID**) on 2074. On süsteeme, mis lasevad vaadelda küll enda, kuid mitte teiste kasutajate protsesse. Oskus oma protsesside kohta ID numbrit nõuda kulub ära kohe, kui mõni protsess rappa

läheb.

13 kill

kill käsuga saab protsessile harakiri teha. **kill -9 1739** peale oleks minu töö faili kallal lõppenud ning avariikoopiat tuleks otsida kusagilt masina sügavustest. Mis tähendab, et iga lolli käsu peale ei pea veel masinat rebootima, piisab vildaka protsessi tapmisest.

14 lock

Oletame, et Sul on vaja oma terminaali juurest hetkeks ära käia, kuid pole viitsimist end päris välja logida. Siinkohal aitab käsk **lock**:

```
siil:/u/anto 2 >lock
Password:
Re-enter password:
terminal locked by anto 0 minutes ago
```

Küll ei tohiks Sa **lock**'i jaoks oma tavalist parooli kasutada. Kasuta lukustusparoolina mingit sõna, mis Sul hästi meelde jääb. Tagasi tulles anna **<ENTER>see_sõna<ENTER>** ning lukk tuleb lahti. Kui Sa lukustusparooli ära unustad, siis tuleb end teise terminaali pealt maha killida. Iga uue masina peal tasub enne kontrollida, kas ehk CTRL+Z või muu katkestusklahvi abil lukust ebaseaduslikult välja ei saa...

Tapvalt kurb, et nii väärtuslik käsk näib olevat vaid SCO UNIXil... Tõsi küll, **xlock** on see-eest olemas praktiliselt kõigil **X/Windows**i versioonidel.

15 date

See on käsk, mille abil saad teada kellaaja ja kuupäeva. Erinevalt DOSist saad Sa ise kellaaega parandada vaid siis, kui Sul on administraatori õigused. **EET** tähendab, et tegemist on Eesti ajaga (*Eastern Europe Time*).

```
Sat Nov 19 16:36:50 EET 1994
```

16 cal

cal -- kalender. See käsk räägib enda eest ise:

```
siil:/u/anto 1 >cal Jan 2005


   January 2005

 S  M Tu  W Th  F  S
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
```

17 bc

bc - *basic calculator*. Milleks Sulle **bc**, kui Sul taskus firma Texas Instruments programmeeritav kalkulaator? Kuid lase ka sellel käsul ise enda vajalikkust tõestada:

```
anto@ogalik:~>bc
bc 1.02 (Mar 3, 92) Copyright (C) 1991, 1992 Free
Software Foundation, Inc.
017509175415471+324653254624576-13146532546326
329015897493721
scale=16
1346426572457536.32546543265432/94325236541235.3457235234
14.2742983938230644
a=324687
b=75431
a+b+25
400143
quit
```

18 cc

cc --- *C compiler*. UNIXi standardsel C'l pole omaette graafilist keskkonda. Seepärast tuleb ka kompileerimiskäsk anda käsurealt. Prooviks kompileeri kokku üks mitte-midagi-tegev C-keelne programm:

```
echo "main(){}" > c.c ; cc c.c -o proge ; proge
```

ning püüa aru saada, miks tuleb saadud *proge* on nii pikk :-). Mõnedes masinates on peale **cc** või selle asemel saadaval ka GNU projekti raames levitatav **gcc** kompilaator. Yldiselt on maitse asi, millist kompilaatorit eelistada.

19 make

Vähegi suuremad programmipaketid sisaldavad suurel hulgal faile laienditega ***.c**, ***.h** ja ***.S** laiendiga faile, mis kõik tuleb läbi vaadata, objektfailideks ***.o** kokku kompileerida ning siis linkimiseks nimetatava tegevuse abil masinkoodis programmiks ühendada. Kogu see keeruline tegevus, nagu ka asjaosalised failid, on kirjeldatud ühes **Makefile** nimelises tekstifailis, mis paketiga kaasas käib. **Makefile**'s muudatusi tehes saad algtekste oma masina ja kataloogistruktuuri jaoks paika sobitada.

make tähtsaim omadus seisneb selles, et kui mõnda paketi koostisse kuuluvat faili parandada, siis **make** kompileerib ümber vaid need osad paketist, mida muudatused puudutasid. Kokkuvõttes - suur ajavõit.

make-käsul on suur hulk võtmeid, millest enamikku, tõsi küll, kunagi ei vajata. Sageli tundub **Makefile** sisu täiesti abrakadabrana, kuid **make** käsku käivitades läheb programm miskipärast ühegi veata kokku. Äratundmise hõlbustamiseks üks pisike lõik Linuxi kerneli **Makefile**'st:

```
CFLAGS=-Wall -Wstrict-prototypes -O2 -fomit-frame-pointer
-pipe
include arch/$(ARCH)/Makefile
ARCHIVES        =kernel/kernel.o mm/mm.o fs/fs.o
net/net.o ipc/ipc.o
FILESYSTEMS     =fs/filesystems.a
DRIVERS         =drivers/block/block.a \
                 drivers/char/char.a \
                 drivers/net/net.a \
```

Kogu kunst UNIXi all uusi programme tekitada seisnebki nende leidmises Internetist, kohaletoomises ja korralikus kokkulaskmises. Räägitakse, et

paremad mehed suudavad ka ise koodi kirjutada, mitte ainult võõraid programme kokku kompileerida :-)

20 exit ehk siis logout

20 käsku on läbi vaadatud. Töö on tehtud, peale UNIXi kasutamist on vaja UNIXist ka välja tulla. Enamasti kõlbab selleks käsk **exit** või sageli ka selle analood **CTRL+D**. Siiski, kui Sa oled käsurealt veel mõne shelli käivitanud, siis aitab **exit** Sind vaid ühe taseme võrra tagasi. Kindlam on kirjutada **logout**, mis alati kohe süsteemist välja viskab. Terminaali juurest tohib rahuliku südamega lahkuda alles siis, kui ekraanile on ilmunud uus login-prompt.

Viies osa

1 Arhiveerimisest

PC kasutaja reeglina ei erista pakkimis- ja arhiveerimisprogramme. Üksainuke utiliit nimega **pkzip.exe** või **arj.exe** teeb tema eest ära kogu töö. UNIXi all on arhiveerise ning pakkimise vahel selge vahe sisse tehtud.

Arhiveerimise all mõistetakse paljudest failidest ja kataloogidest ühe kompaktse tüki moodustamist. Arhiivi päises säilitatakse info üksikute failide pikkuste ja loabittide kohta, nii et seda on taas võimalik tükkideks lahti võtta. Ühe tükina on failikogumit kergem teise kohta tõsta või kettale jäädvustada.

Tuntuim UNIXi arhiveerimisprogramm on muidugi **tar** - *the **T**ape **AR**chiver*. **tar**il on suur hulk võtmeid, GNU **tar**il näiteks on võtmeid üle 50. Tegelikus elus läheb neist sageli vaja vaid kolme võtmekombinatsiooni:

tar -cvf arhiiv.tar /ladu
Luua (**c** - *create*) arhiivifail (**f** - *file*) nimega *arhiiv.tar* (*.tar laiend pole kohustuslik, kuid viisakas oleks see DOSiga ühilduvuse huvides siiski lisada). Arhiivi sisuks võtta kõik failid, mis paiknevad kataloogis */ladu*. Arhiveerimise käigus rääkida kasutajale, mis toimub (**v**- *verbose*). Pakkimata arhiiv tuleb loomulikult mahult suurem, kui temas sisalduvad üksikud failid kokku.

tar -tvf arhiiv.tar
Kontrollida (**t** - *test*), kas arhiiv ikka on terviklik ja korras.

tar -xvf arhiiv.tar
Lasta arhiiv taas algosakesteks laiali (**x** - *e**X**tract*)

UNIXi all saab arhiveerimisprogramm vahetult ka flopi- ehk lindiseadet kasutada. SCO UNIXi all töötab järgne näide (teistel UNIXitel on kettaseadmete nimed ehk veidi erinevad):

tar -cvfb /dev/rfd1135ds18 18 fff1 fff2
raw mode'is kirjutada 1,44 MByte kettaseadmele (**fd** - *floppy disk* 1, **135** dpi, **ds** - *double sided*, **18** *sectors*) arhiiv blokisuurusega (**b** - *block*) **18**, kusjuures arhiiv koosnegu failidest *fff1* ja *fff2*.

Loomulikult peab ketas eelnevalt UNIXi formaadis olema. Peaaegu sama süntaksiga saab teha backuppi kogu winchesterist lindiseadmele (**/dev/rt0** - *raw tape* 0).

Nagu igal heal programmil, on ka **tar**il pisikesi puudusi. *tar* ei tarvitse säilitada kataloogide loabitte, samasi on need failide puhul alati õiged. Seepärast tuleb kataloogide omanikud ja loabitid alati üle kontrollida ja taastada.

Ilma vajaduseta ei tohi kunagi **root**-nime all tariga arhiive lahti teha, ammugi veel root-kataloogis (*/*). **tar** ruudu nime alt käivitatuna kirjutab kettal üle mistahes samanimelise faili, hävitades Sinu töö või isegi UNIXi kerneli. Loomulikult on võtmeid, millega seda vältida, kuid milleks asja liiga keeruliseks ajada.

root-nime all tegutsedes tuleb silmas pidada ka seda, et **tar** püüab arhiivi lahti kruvides taastada failid esialgsete omanike nimele. Tuues teisest masinast arhiivifaile, riskid alati neid olematu kasutaja omaks lahti pakkida (Sinu masinas ehk polegi kasutajat, kelle **UID** (***U**ser **ID**entificator*) number oleks 3958 vms ;-)).

UNIXi arhiveerimisprogramme on teisigi, **backup** ja **cpio** on tüüpilised näited, kuid neid kasutavad enamasti vaid süsteemiadministraatorid, **ar** on rohkem progejate rida. Arhiveerimisprogrammide eriti huvitavatest omadustest veel niipalju, et mõnedki UNIXid suudavad bootida isegi arhiivilindilt. Süsteemi *crashi* või kettaseadme rikke puhul on sellest omadusest vägagi abi.

2 Kokku- ja lahtipakkimisest

Pakkimise lühikursus UNIXile on tõega lühike:

```
kinni           lahti           laiend
-----------------------------------------------
pack        unpack            .z
compress    uncompress  .Z
gzip        gunzip            .gz
                        (väga vanadel versioonidel ka .z)
```

tariga kokkupandud arhiivi lahtipakkimiseks võid Sa kasutada suvalist neist kolmest käsust. Parima kompressiooni annab kindlasti **gzip**. Niisiis, et lahti pakkida pakitud arhiivi nimega ***Yx_arhiiv.tar.Z***, kasuta tema kallal kõigepealt käsku **uncompress**, tulemus kontrolli igaks juhuksi üle **tar -t** abil ning alles siis, õiges kataloogis asudes, päästa arhiiv lahti käsuga **tar -xvf**.

pack ja **compress** on olemas igas UNIXi distributsioonis. **gzip** seevastu on loodud GNU projekti raames ning seepärast leiad teda valmiskujul ehk vaid Linuxist. Kuid keegi ei takista Sul GNU zipi algtekste kohale toomast ja oma masinas kokku kompileerimast. Kompressiooniastmelt on **gzip** standardvahenditest muidugi kõvasti üle. Ka on lihtne leida **gzipi** DOSiversiooni, kui seda vaja peaks minema. GNU projektist veel niipalju, et käsk:

```
tar –xvfz arhiiv.tgz
```

toimetab korraga nii dearhiveerimist kui ka lahtipakkimist, olles selles mõttes DOSi pakkerite analoog.

Kui juba DOSist juttu, siis on Sul võimalus võrgust kohale tuua ka **unzip**i ja **unarj**i algtekstid. Need programmijupikesed võimaldavad DOSi ***.zip** ja ***.arj**

faile UNIXi all lahti pakkida. Kokkupakkimist vabas versioonis ei sisaldu, seda tööd pead ikka DOSi all edasi tegema.

3 Äraspidi pakkimine

Arvutivõrkudes on sageli probleeme sidekanalitega, mis pole kõigi 256 ASCII-koodi jaoks läbipaistvad. Väga sageli jäävad sidekanalisse kinni **CTRL+Q** (kood 17) ja **CTRL+S** (kood 19), veel sagedamini **DEL** (koodiga 127). Ka muid kontrollkoode edastavad sidekanalid üpris suvaliselt.

Teine mure on täpitähtedega. Paraku on arvutivõrgud alguse saanud just neis riikides, kus ASCII-tabeli teist poolt eriti ei vajatud. Nii olid enamik sideliinegi vanasti 7-bitised. Vahepeal saadi siiski onu Sami maal aru, et ka Euroopas on toodetel turgu ning enamikele uutele UNIXitele on 8nda-biti *support* vägisi sisse keevitatud. Sellele vaatamata, kui tahad olla absoluutselt kindel, et mõni eriti antiikne masin Sinu kirjapilti moonutama ei hakka, siis vältige ASCII koode 0-31 ning 127-255.

UNIXi (ja ka DOSi, Maci jne) jaoks on päris mitu programmi, mis sellist äraspidi kodeerimist teostavad. **uuencode**, **binhex** ning **base64** on selle perekonna tüüpilised esindajad. Kui Sa kasutad faili kallal käsku **uuencode** (*unix-to-unix encode*), siis saadud fail on küll originaalist pikem, kuid sisaldab see-eest vaid ASCII märke vahemikus SPACE - Z. Sellist faili sideliine pidi edastades pole kanali läbipaistvusega muret. Niisiis on tegemist justkui pakkimisele vastupidise tegevusega.

Teisel pool otsas tuleb fail **uuencode**ga paarilise programmi **uudecode'i** abil normaalkujule viia.

Macimaailmas väga tuntud programmid **binhex** ning **base64** on omadustelt palju efektiivsemad, kuid paljudes süsteemides neid lihtsalt pole. Kui tahate kindla peale mängida, siis kasutage faili saatmisel ikka **uuencode/uudecode** paari. FTP-arhiividest postiga *stuffi* tellides töödeldaksegi faile vaikimisi just uuencode'i abil.

4 Filtrid

Definitsioon: *filter* on programm, mis kasutab sisendina standardsisendit (vaikimisi klaviatuur), väljundina standardväljundit (vaikimisi ekraan) ning vahepeal toimetab andmevoo muundamisi miskite kindlate reeglite põhjal.

```
sisendvoog -> töötlemine  -> väljund
                  ^
                  |
               reeglid
```

Just põhimõte, et enamik UNIXi utiliite on võimelised filtrina talitlema, teeb UNIXist avatud ja universaalse süsteemi. Käsureas võib kasutada mitut filtrit järjepanu:

```
käsk | filter1 | filter2 > filee
```

Ka tohib sisendi ja väljundina kasutada omaenda faile:

```
filter3 < minu.in > minu.out
```

5 Stringitöötlus

Väga kiiresti ja korralikult töötab stringiotsimisproge nimega grep. Kirjutades:

```
grep 386 Miskifilee
```

saad Sa väljundis kõik read vastavatest failidest, millistes sisaldub sõna **386** (siinkohal vaadeldakse seda sõnana, mitte numbrina). **grep -v 386 Miskifilee** näitab, vastupidi, ridu, milles sõna **386** *ei sisaldu*. **grep -c** loetleb *386*-sõna sisaldavaid ridu, **grep -l 386 *** lappab läbi kõik kataloogis leiduvad failid ning nimetab neid ridu, milles sõna *386* sisaldus. Kui **grep**ile on vaja ette anda tühikut sisaldav string, siis peab selle panema jutumärkidesse.

grep koos **find**iga on üks võimsamaid relvasid UNIXis üldse. Kui sageli juhtub, et Sa oled unustanud faili nime, kuid mäletad mõnda fraasi sellest failist. Siinkohal aitab käsk:

```
find / -type f -exec -grep -l "Miski tekst" {} \;
```

Otsitakse juurikast (*/*) alates faile, mis on tõega ikka *fil*ed, (**-type f**), mitte kataloogid, semaforid, I/O riistade peegeldused */dev* kataloogis ega muu spetsiifiline kraam. Kui leitakse normaalne fail (neid aga on enamik), siis iga sellise faili kallal (**{} \;**) täidetakse (**-exec** - *execute*) käsk **grep**. Failist otsitakse etteantud teksti ning kui leitakse, väljastab **grep -l** kasutajale paljastatud faili nime. Semikoolon **;** märgib **-exec** alamkäsu lõppu ning *backslash* **** (ehk äraspidi-munajas kaldkriips :-)) on tarvitusel semikooloni varjestamiseks C-shelli eest.

Peale lihtsa **grep**i pakutakse UNIXis vahel veel eriotstarbelisi **grep**pe (:-)) nagu **egrep** (e-*enhanced*) ja **fgrep** (f-*fast*). Igaüks valib siis endale meelepärase kiiruse ja võimalustega versiooni.

6 Regular Expressions

Regulaaravaldised on tavalise malli edasiarendus. Iga DOSikasutaja teab, mida tähendab kustutamisekäsu sabas seisev ***.***. UNIXi puhul ei käi mall mitte ainult failinimede kohta. Kuna käsuandja ja käsutäitja vahel seisab alati **shell**, siis on regulaaravaldised kasutatavad igasuguses stringitöötluses (failinimede genereerimine on vaid üks erijuhtusid).

```
*      - ükskõik mitu (0-  ) eelnevat ASCII koodi
.      - mistahes märk peale reavahetuse (NB!! UNIXi
           reavahetuse märgiks on Line Feed koodiga 10)
           Ka tasub teada, et failinimede genereerimisel kasutab
           shell samaks otstarbeks küsimärki.
[9aHZ]       - üksainuke märk, mis sisaldub etteantud valikus.
           Antud juhul on niisiis valida tervelt 4 võimaluse
           hulgast (number 9, väike a ja suured H ning Z).
[a-f] - üks täht vahemikus a-z (a, b, c, d, e, f niisiis).
^      - tähistab rea algust ( nii et ^A pole mitte CTRL-A,
           nagu muidu UNIXi kirjanduses tavaks, vaid hoopis uue rea
           alguses paiknev suur A-täht.
i$     - tähistab i-tähte vahetult enne reavahetust.
```

On olemas tunduvalt tunderikkamaid konstruktsioone, nagu

```
[:ctrl:]     - mistahes CTRL-märk
z*           - z-tähtesid koguses null kuni palju;
```

ning ka terve reeglite süsteem, mis võimaldab defineerida sortimisel b-tähe ettepoole kui a-tähe, kui mingi maa keeles selleks vajadus peaks tekkima. Kõige selle kohta võib vaba aja olemasolul ise lugeda manuaalist käskudega **man ed** ja **man locale**.

*So **RTFM**, Read This F_ing (sorry, Fine) Manual*!

Küsid, et mida me nende avaldistega peale hakkame? Näiteid:

Leida kõik failid, mille nimi algab tähtedega **a** või **g** või **k**.

ls [agk]*

Otsida, kuhu võis pahategija endale **setUID** root-shelli maha jätta (tüüpiline ülesanne häkkeri ja süsadmini vahelises igaveses sõjas).

find / -type f -user root -exec \ ls -la {} \; | egrep '^-..s'

Siia on ilmselt vajalik lisada paar selgitust. Kõigepealt **setUID** programmi põhiomadus - teda käivitades saate programmi täitmise ajaks **root**-kasutaja õigused. Sellise proge loabittides on omaniku kohal asuv **x** (*execute*) asendatud **s**-tähega (*setUID*). Programm ise aga ei tee muud kui otsib (**find**) alates juurkataloogist (*/*) normaalseid faile (**-type f** - *file*) mille omanikuks (**-user**) on root. Kui sääraseid faile leitakse, siis täidetakse (**-exec**) igaühe (**{}**) kohta käsk **ls -la** (**-a** sest siis on näha ka varjatud failid). **\;** märgib **-exec** - alamkäsu skoobi lõppu. Kõik, mida **find** sel moel väljastas, lastakse läbi toru **egrep**i sisendisse, otsides avaldist *'^-..s'*. Teisisõnu, leitakse failid, mille loabitid vastavad mallile **-..s.....**, näiteks **-rws--x--x** . Äraspidi kaldkriipsud (\) rea lõpus on kasutusel pika rea poolitamiseks (iluvõte).

Ning kohe ka küsimus - kuidas lahendada sama ülesannet üksnes **find** käsu vahenditega?

7 Stream Editor – *sed*

sed on tekstieditor, mis töötab automaatrezhiimis ning etteantud reeglite kohaselt. **sed** talitleb nagu klassikaline *filter* .

8 Kehvast sidest

Paljud tarbijad ripuvad Interneti küljes UUCP protokolliga. Tahaks nagu megabaidiseid faile nihutada, aga ei saa. Meie telefoniliinide kvaliteet on täpselt nii hea, et side katkeb iga paarisaja kilobaidi tagant (ja maakohas veel kiireminigi). UUCP protokolli puhul ei saa kahjuks faili tirimist keskpaigast jätkata. Järeldus on selge -- fail tuleb enne ärasaatmist juppideks teha.

Enne tükeldamist on meil vaja teada saada ridade arv failis. Seda teeme käsuga **wc**:

```
anto%>wc Miskifilee

3620  19224 142182 Miskifilee
```

ehk 3620 rida, 19224 sõna ja 142182 märki.

Järgmiseks kasutame utiliiti nimega **split**, mis ongi mõeldud suurte (teksti)-failide tükeldamiseks.

```
split -800 Miskifilee
```

jupitab faili 800-realisteks osadeks nimedega *xaa*, *xab*, *xac*, *xad* jne. Need saab nüüd postiga ohutult üle pumbata ning sihtkohas taas juppidest kokku ehitada (on olemas säärane vabavara (*freeware*) nagu **uuexplode**, mis kirjapäistega pikitud **uuencode**'itud juppidest taas originaali kokku paneb).

Vaatleme nüüd skripti , mis jupid kenasti minema saadab (töötab SCO ja Linuxi all, Solarise puhul kirjuta **mail**i asemel **mailx**):

```
#!/bin/sh
# Käivitada ikka
# Bourne'i shelliga!
#
for i in `ls x??`
do
echo $i
mail -s "Selle osa nimi on $i" \
anto@siil.edu.ee < $i
sleep 5
done
#
```

Tagurpidi-kaldkriips on siin taas kasutusel pika rea poolitamiseks, Viiesekundilise ajalise viite (*sleep 5*) panime juurde selleks, et pisema masina postiprogramm üle ei koormuks.