ATK Tähtitieteessä, osa 1: IDL

Pertti Rautiainen 17.11.1998

Revised by Heidi Korhonen and Raine Karjalainen

26. maaliskuuta 2001

Lyhyt johdatus IDL:n käyttöön Unix-ympäristössä.

Sisältö

Luku 1

Johdanto

ATK tähtitieteessä -kurssin ensimmäinen osuus käsittelee ohjelmistopakettia nimeltä IDL (Interactive Data Language). Koska opetus tapahtuu Unix-ympäristössä, niin kurssilla käsitellään lyhyesti myös Unix-käyttöjärjestelmän tärkeimpiä komentoja.

Unix- ja IDL-komennot kirjoitetaan `kirjoituskonefontilla`.

Luku 2

Yleisimpiä tähtitieteilijöiden käyttämiä ohjelmistoja

Tähtitieteilijät käyttävät tietokoneita moniin tarkoituksiin, eli muuhunkin kuin mudaamiseen. Tässä osassa mainitaan lyhyesti joitakin yleisimpiä tähtitieteilijöiden käyttämiä ohjelmia. Kaupallisten ohjelmien lisäksi on olemassa monia hyvin käyttökelpoisia ohjelmia jotka ovat ilmaisia ja niitä voi asentaa kotikoneeseenkin. Niitä löytyy etenkin Linuxille, joka on Unixin kaltainen käyttöjärjestelmä.

2.1 Editorit

Editorit ovat ohjelmia, joilla voi käsitellä ASCII-tekstitiedostoja. Tärkein editori Unix-maailmassa on emacs. Muita ovat esim. pico ja vi.

2.2 Tekstinkäsittely

Tähtitieteilijät eivät yleensä käytä Wordin kaltaisia tekstinkäsittelyohjelmia, vaan kirjoittavat editorilla LATeX-koodia (LATeX on oikeastaan "pelkällä" TeX:illä kirjoitettu makropaketti), joka sitten erikseen käännetään ja muutetaan esim. postscript-formaattiin tulostamista varten. Tähän ovat pääsyinä: 1) matemaattisen tekstin kirjoittaminen kaavaeditoria käyttämällä johtaa yleensä hoitopaikkaan suljetulla osastolla, 2) monet tieteelliset lehdet olettavat, että käsikirjoitukset toimitetaan lehden omilla LATeX-makropaketilla muotoiltuna, 3) LATeX on saatavilla käytännössä kaikkiin laiteympäristöihin ja 4) LATeX-koodi on puhdasta ASCII-tekstiä, joten se on vaivatonta lähettää sähköpostissa esim. toisissa maissa oleville yhteistyökumppaneille.

2.3 Sähköposti

Sähköpostin lukuun ja lähettämiseen on olemassa useita ohjelmia alkaen käyttöjärjestelmään kuuluvasta yksinkertaisesta mail-ohjelmasta. Useimpien mielestä sitä helppokäyttöisempi on valikkoja käyttävä pine.

Sähköpostiosoite on tyyppiä tunnus@kone.domain, esim. unski@paju.oulu.fi. Saman "domainin" sisällä voi käyttää myös muotoa tunnus@kone ja saman koneen sisällä pelkkää tunnusta.

2.4 WWW

Tärkein WWW-selain Unix-puolella on Netscape. Internet Explorer on saatavina joihinkin unixeihin. Lisäksi voi mainita tekstipohjaisen selaimen Lynxin.

WWW:n merkitys on selvässä kasvussa. Monilla lehdillä on WWW-versionsa ja monista artikkeleista on saatavissa ns. preprint-versiot ennen varsinaista julkaisua. Ja suuri määrä esim. NASA:n kuvamateriaalia on saatavana verkossa. Seuraavassa joitakin tähtitieteellisiä WWW-paikkoja:

http://oposite.stsci.edu/pubinfo/pictures.html – Hubblen kuvia
http://mars.sgi.com/mgs/index.html – Mars Global Surveorin kuvia
http://www.skypub.com/ – Sky and Telescope -lehti
http://www.kalmbach.com/astro/astronomy.html – Astronomy -lehti
http://nedwww.ipac.caltech.edu/ – laaja galaksitietokanta, kuviakin
http://stdatu.stsci.edu/dss/ – The Digitized Sky Survey
http://www.cv.nrao.edu/fits/www/astronomy.html – hyvä linkkilista

2.5 Ohjelmointi

Tärkeimmät ohjelmointikielet tähtitieteessä ovat samat kuin muussakin numeerisessa laskennassa eli Fortran (Fortran 77 tai Fortran 90) ja C/C++. Molemmista löytyy sekä ilmaisia (gcc, g77) että kaupallisia kääntäjiä.

Vähitellen kasvava osa laskennasta tapahtuu tulkkaavilla järjestelmillä, joiden käyttäminen on usein suoraviivaisempaa kuin kääntävien ohjelmointikielten käyttö. Esimerkkejä ovat IDL, Matlab ja Mathematica. Todella suuriteholaskentaa tarvittaessa on tosin edelleen käytettävä Fortrania tai C:tä.

2.6 Havaintojen käsittely ja tulosten esittäminen

Havaintojen, esim. ccd-kuvien käsittelyyn on olemassa useita erilaisia ohjelmia. Eräs tärkeimmistä ohjelmapaketeista on IRAF. Myös IDL:lle on kirjoitettu paljon havaintojen käsittelyyn sopivaa ohjelmistoa.

Tulosten (havainnot tai numeeriset mallit) graafisessa esittämisessä eli visualisoinnissa IDL on hyvin vahva yleistyökalu. Sen sijaan tavanomaiseen kuvien käsittelyyn löytyy helppokäyttöisempiäkin ohjelmia. Kuvien katseluun ja kuvien muuttamiseen formaatista toiseen sopii ohjelma nimeltä xv. Monipuolista manipulaatiota kuville voi tehdä ohjelmalla nimeltä Gimp. Siitä löytyy suuri osa vastaavista toiminnoista kuin Mac- ja Windows-maailman kuvankäsittelyohjelmista.

Luku 3

Unix tunnissa

Koska kurssilla on hyvin rajallisesti aikaa käytettävissä, niin jostakin on karsittava. Suurin karsinta kohdistuu käytettävän käyttöjärjestelmän, Unixin opetukseen. Tämän vuoksi itseopiskelu ja vaikkapa Teoreettisen fysiikan osaston järjestämien ATK-kurssien läpikäyminen on enemmän kuin suositeltavaa.

3.1 Mikä Unix on?

Unix on tietokoneen käyttöjärjestelmä, tarkemmin sanottuna monen käyttäjän moniajokäyttöjärjestelmä, mikä erottaa sen esim. MS-DOS:sta tai Windows 95:sta. Tämä tarkoittaa sitä, että Unixissa voi samalla käyttäjällä olla monta eri prosessia samaan aikaan pyörimässä ja toisaalta sitä, että koneella voi samanaikaisesti olla useita käyttäjiä. Useiden käyttäjien yhtäaikainen läsnäolo tarkoittaa sitä, että hyvin oleellinen piirre unixille ovat käyttäjätunnukset ja tunnuskohtaiset kotihakemistot. Edellisen perusteella on ilmeistä, että Unix-koneesta saa kaiken irti silloin kun kone on kytkettynä tietokoneverkkoon.

Eri laite- ja ohjelmistovalmistajilla on erilaisia versioita Unixista, mutta erot ovat tavallisen käyttäjän kannalta varsin pieniä. Unix on rekisteröity tavaramerkki ja tuota tavaramerkkiä käyttävien järjestelmien lisäksi on olemassa joukko Unixin kaltaisia järjestelmiä. Seuraavassa sanalla Unix tarkoitetaan molempia. Tärkeimpiä Unixeja ovat:

-Solaris/SunOS (Sun)
-AIX (IBM)
-HP-UX (HP)
-IRIX (SGI)
-Linux (vapaa käyttöjärjestelmä joka on saatavissa useille laiteympäristöille)

Käytettävän käyttöjärjestelmäversion saa näkyviin komennolla `uname -a`.

Vaikka unixia voidaan käyttää pelkässä tekstipääteyhteydessä (esim. Väylän varrella olevat päätteet), niin nykyään useimmiten käytetään apuna ikkunointijärjestelmää, yleensä X11:ta. Tämä mahdollistaa useiden yhteyksien käytön eri ikkunoissa ja monien hiirtä apunaan käyttävien ohjelmien käyttämisen. Ikkunointijärjestelmistä on useita eri variaatioita aina karusta perus X:stä sellaisiin jotka sisältävät työpöydän roskalaatikoineen.

3.2 Loggautuminen Unix-koneeseen

Jos käyttää X-päätettä tai konsolia, niin kone on alussa tilassa jossa lukee

```
login:
```

Jotta koneeseen pääsee sisälle, pitää kirjoittaa oma käyttäjätunnuksensa. Tämän jälkeen kone kysyy salasanaa. Tämän kirjoittaminen ei tulostu ruudulle. Jos kirjoittaa oikean salasanan, niin pääsee sisälle koneeseen. Tässä vaiheessa kone ajaa joukon alustusprosesseja, jotka riippuvat asetuksista. Tällainen voi olla esimerkiksi ikkunointijärjestelmän käynnistäminen.

On olemassa myös Windowsissa pyöriviä yhteysohjelmia, joista joissakin on mahdollista avata X-ikkunoita (esim. Exceed/W), mutta noiden ohjelmien toimintaa ei käsitellä tässä.

Jos käyttää esimerkiksi Väylän varrella olevia tekstipäätteitä, niin loggautumisprosessi on hieman erilainen: aluksi kirjoitetaan koneennimi, esim. `paju`. Tämän jälkeen loggautuminen jatkuu kuten edellisessä esimerkissä. Poistuminen yhteydestä tapahtuu komennolla `exit` tai `logout`. Joissakin ikkunajärjestelmissä on valikoista saatavilla poistumiskomento.

Salasanan pitää olla vaikeasti arvattava, mieluusti 8 merkkiä pitkä, jossa osa merkeistä on numeroita ja isoja kirjaimia. Useisiin koneisiin on asennettu ohjelmat, jotka yrittävät murtaa huonoja salasanoja.

Salasanan voi muuttaa toiseksi kirjoittamalla `passwd`, jonka jälkeen kone pyytää ensiksi kirjoittamaan vanhan salasanan ja sen jälkeen uuden (yleensä kone pyytää toistamaan uuden salasanan).

Usein tilanne on se, että jokin ohjelmisto on käytettävissä vain jossakin tietyssä Unix-koneessa tai jotain tiettyä Unix-konetta halutaan käyttää laskentaan sen suuren tehon vuoksi. Tällöin joudutaan ottamaan yhteys Unix-koneesta toiseen. Vanhastaan tämä tapahtui yleensä `rlogin` -komennolla tai telnet-ohjelmalla. Nuo yhteydet ovat kuitenkin tietoturvaltaan heikkoja, ja nykyään käytetään ssh-ohjelmaa joka kryptaa käytettävän viestiliikenteen.

Jos käyttäjätunnus logattavassa koneessa on sama kuin loggautumiseen käytettävässä koneessa, niin loggautuminen tapahtuu seuraavasti: `ssh kone`, minkä jälkeen kone kysyy salasanaa. Jos taas tunnus on eri, niin sitten kirjoitetaan: `ssh kone -l tunnus`, missä tunnus on käyttäjätunnus logattavassa koneessa.

Kun Unix-koneeseen ollaan loggauduttu, niin käyttäjän kirjoittamia komentoja käsittelee ns. komentotulkki (engl. shell). Unix-maailmassa on olemassa useita komentotulkkeja, yleisiä ovat mm. csh, tcsh ja bash. Kurssilla käytettävällä demo-tunnuksella komentotulkkina on tcsh.

Kun käyttäjä loggautuu koneeseen, käyttöjärjestelmä suorittaa joitakin alustustiedostoja. Tcsh:ssa (ja csh:ssa) sellaiset ovat `.login` ja `.cshrc`. Jos käyttäjä haluaa tehdä henkilökohtaisia asetuksia, niin sellaiset voi luoda omaan kotihakemistoonsa, muussa tapauksessa järjestelmä käyttää "prototyyppejä". `.cshrc`-tiedostoon voi määritellä "aliaksia", eräänlaisia lyhennelmiä komennoista, esim. demo-tunnuksella on määritelty:

```
alias ll 'ls -al'
```

Samoin `.cshrc`:ssä on määritelty tarvittavia polkuasetuksia.

3.3 Hakemistorakenne ja tiedostot

Samaan tapaan kuin muissakin käyttöjärjestelmässä, niin Unixissakin tieto sijaitsee kovalevyillä olevissa tiedostoissa. Asioiden selkeyttämiseksi käyttöjärjestelmässä on olemassa hakemistorakenne. Tämä tarkoittaa sitä, että on olemassa hakemistoja, joissa on tiedostoja tai alihakemistoja jne. Komennolla pwd näkee oman sijaintinsa hakemistorakenteessa. Esim.

```
typy(18)% pwd
/home/pertti/programs
```

Listauksen hakemiston sisällöstä saa komennolla `ls`. Pelkän ls-komennon antama informaatio on niukkaa, ja enemmän saa komennolla `ls -al`, missä komennolle ls annetaan argumentteina optiot a ja l (tästä komennosta on .cshrc-tiedostossa määritelty alias nimeltä ll). Seuraavassa esimerkki edellisen komennon vaikutuksesta:

```
typy(20)% ls -al
total 8
drwxr-xr-x   6 pertti   users        1024 Jan 28 01:24 ./
drwxr-xr-x  26 pertti   users        3072 Apr 14 17:49 ../
drwxr-xr-x   2 pertti   users        1024 Jan 28 01:24 F/
drwxr-xr-x   2 pertti   users        1024 Apr  6 19:32 c/
drwxr-xr-x   2 pertti   users        1024 Jan 17 02:37 c++/
drwxr-xr-x   3 pertti   users        1024 Feb  2 19:11 fortran/
-rwxr-xr-x   1 pertti   users        2190 Apr 14 17:57 temppi
```

Alihakemistossa programs on neljä alihakemistoa (d -kirjain tulostusrivin alussa) ja yksi tiedosto (hakemistot ./ ja ../ tarkoittavat hakemistoa jossa ollaan ja sen alapuolella olevaa hakemistoa). Komento ilmaisee myös hakemistojen ja tiedostojen omistajista, käyttöoikeuksista, koosta sekä luonti/muutosajanhetkestä. Hakemiston c sisällön saa seuraavasti:

```
typy(24)% ls -al c
total 8
drwxr-xr-x   2 pertti   users        1024 Apr  6 19:32 ./
drwxr-xr-x   6 pertti   users        1024 Apr 14 17:57 ../
-rw-r--r--   1 pertti   users          38 Feb 14  1996 Makefile
-rwxr-xr-x   1 pertti   users        4008 Apr  6 19:32 hello*
-rw-r--r--   1 pertti   users          61 Jan 17 02:34 hello.c
```

Lisätietoa komennosta ls saat kirjoittamalla `man ls`. Tämä avaa kyseistä komentoa vastaavan manuaalisivun. Vastaavat manuaalisivut ovat myös muista komennoista.

Uuden alihakemiston voi luoda komennolla mkdir olettaen, että käyttäjällä on kirjoitusoikeus kyseiseen paikkaan hakemistorakenteessa. Esim.

```
typy(25)% mkdir puppu
typy(26)% ls -al
total 12
drwxr-xr-x   7 pertti   users        1024 Apr 14 18:03 ./
drwxr-xr-x  26 pertti   users        3072 Apr 14 17:49 ../
drwxr-xr-x   2 pertti   users        1024 Jan 28 01:24 F/
drwxr-xr-x   2 pertti   users        1024 Apr  6 19:32 c/
drwxr-xr-x   2 pertti   users        1024 Jan 17 02:37 c++/
drwxr-xr-x   3 pertti   users        1024 Feb  2 19:11 fortran/
drwxr-xr-x   2 pertti   users        1024 Apr 14 18:03 puppu/
-rwxr-xr-x   1 pertti   users        2190 Apr 14 17:57 temppi
```

Tyhjän hakemiston voi tuhota komennolla rmdir, esim. `rmdir puppu`.

Yksittäisen tiedoston voi tuhota komennolla rm, esim. `rm temppi`.

Yleensä jokaisella käyttäjällä on oma kotihakemisto, esim. /home/pertti. Omaan kotihakemistoonsa käyttäjä voi luoda uusia hakemistoja ja tiedostoja sekä tuhota niitä. Tämän lisäksi on voitu määritellä muita paikkoja hakemistorakenteessa joihin käyttäjällä on luku- tai kirjoitusoikeus.

Hakemistorakenteessa on myös paikkoja, joihin tavallisella käyttäjällä ei ole kirjoittamisoikeutta. Näitä ovat esimerkiksi kaikki sellaiset paikat jotka sisältävät käyttöjärjestelmään liittyviä tiedostoja.

Hakemistosta toiseen pääsee käyttämällä komentoa cd. Esim. edellä olisi päässyt hakemistosta /home/pertti/programs hakemistoon /home/pertti/programs/fortran komennolla `cd fortran`. Vastaavasti kotihakemistosta kyseiseen hakemistoon olisi päässyt komennolla `cd programs/fortran`. Kotihakemistoon pääsee pelkällä cd komennolla. Hakemistorakenteen “juureen” pääsee komennolla `cd /`.

3.4 Muita tärkeitä Unix-komentoja

Seuraavassa esitellään parikymmentä tärkeää Unix-komentoa.

Tiedoston nimen voi muuttaa toiseksi käyttäen mv komentoa: `mv tiedosto1 tiedosto2`.

Tiedoston voi kopioida toiselle nimelle cp komennolla: `cp tiedosto1 tiedosto2`. Tämän jälkeen on olemassa kaksi identtistä tiedostoa. Jos nimellä `tiedosto2` on jo ollut tiedosto, niin se tuhoutuu em. kopiointiprosessissa.

Tiedoston kopioimiseen koneesta toiseen oli tapana käyttää komentoa rcp tai ottaa ftp-yhteys. Tietoturvallisuussyistä noita tulisi nykyään välttää ja käyttää scp ohjelmaa:

`scp tiedosto tunnus@kone:hakemisto` -kopioidaan tiedosto toisen koneen hakemistoon.

`scp tunnus@kone:hakemisto/tiedosto` -kopioidaan toisen koneen hakemistosta tiedosto siihen hakemistoon, jossa ollaan.

Komennolla cat voi tulostaa tiedoston sisällön: `cat tiedosto`.

Koska tiedostot ovat usein pitkiä, edellinen komento ei ole aina kovin kätevä. Komento more toimii paremmin: `more tiedosto`. Tämä tulostaa ensiksi ruudullisen tiedoston alusta. Lisää saa joko ruutu (välilyönti) tai rivi (Enter) kerrallaan. Moniin järjestelmiin on asennettu vieläkin kehittyneempi komento less, jolla tiedostoa voi selata molempiin suuntiin.

Kahden tiedoston väliset erot voi tulostaa komennolla diff: `diff tiedosto1 tiedosto2`.

Haluttua merkkijonoa voi etsiä tiedostosta komennolla grep: `grep hassu tiedosto` tulostaa tiedostosta ne rivit, jotka sisältävät merkkijonon hassu. Optiolla -i aiheuttaa sen, että isot ja pienet kirjaimet samaistetaan. Optiolla -v tulostetaan puolestaan ne rivit, jotka eivät sisällä annettua merkkijonoa.

Komento sort asettaa tiedoston riveittäin aakkosjärjestykseen.

Normaalisti edellisten komentojen tulostukset ohjautuvat ruudulle. Ne voi myös ohjata tiedostoon käyttämällä > -merkkiä, esim. `sort tiedosto1 > tiedosto2` tulostaa tiedoston tiedosto1 riveittäin aakkostettuna tiedostoon2.

Usein halutaan tulostaa tiedosto printterille. Tämä tapahtuu komennolla lpr. Option voidaan antaa printterin nimi, esim. `lpr -Pturbops tiedosto1` tulostaa tiedoston printterille nimeltä turbops. Yleensä jokin printteri on määriteltu oletusarvoksi ja sille tulostettaessa printterin nimeä ei tarvitse erikseen mainita. Printterin tulostusjonon näkee komennolla `lpq`. Jokaisella tulostuksella on oma numeronsa. Tulostuksen voi hävittää tulostusjonosta komennolla `lprm nro`, missä nro viittaa lpq:lla määritettyyn tulostuksen numeroon.

On hyvin tärkeää, että printtereille lähetetään vain sellaista materiaalia, jonka ne voivat tulostaa! Esimerkiksi postscript-kirjoittimelle ei saa lähettää kuvatiedostoja jpeg-formaatissa, vaan ne on ensiksi muunnettava esimerkiksi xv-ohjelmalla postscript-formaattiin. Muussa tapauksessa voi tulostua monta sivua erikoismerkkejä sisältävää kökköä.

Suuria tiedostoja, joita ei tarvita kovin usein voi kompressoida tiiviimpään muotoon. Kaikissa unix-järjestelmässä on komento compress: `compress tiedosto` kompressoi tiedoston sisällön ja luo tiiviimmän tiedoston nimellä tiedosto.Z. Komennolla `uncompress tiedosto.Z` tiedosto palautuu alkuperäiseen muotoon. Useissa järjestelmässä on myös pakkaussuhteeltaan tehokkaampi gnuzip, jonka pakkaus ja purkukomennot ovat gzip ja gunzip.

Järjestelmän yleisen levytilan käytön näkee komennolla `df`. Yksittäisen hakemiston taas komennolla `du`.

Yksittäisiä prosesseja voidaan käynnistää myös taustalle käyttämällä & -merkkiä, jolloin niiden käynnistämiseen käytetty xterminaali pystyy edelleen vastaanottamaan komentoja, esim. `emacs tiedosto &`. Jos haluaa muuttaa jo pyörivän prosessin taustaprosessiksi, niin se tapahtuu painamalla ensiksi prosessin käynnistäneessä xterminaalissa Ctrl-Z ja kirjoittamalla sitten `bg`. Muutos toiseen suuntaan tapahtuu komennolla `fg`.

Käynnissä olevat prosessit näkee komennolla ps. Esim.

```
typy(52)% ps
  PID TTY STAT  TIME COMMAND
 7647   1 S    0:00 -tcsh
22223   1 S    0:00 sh /usr/X11/bin/startx
```

```
22224   1 S    0:00 xinit /usr/X11R6/lib/X11/xinit/xinitrc --
22227   1 S    0:03 fvwm95-2
22237   1 S    0:00 xbiff
22241  p0 S    0:00 -tcsh
23055  p0 S    0:01 emacs atk_tahti.tex
23448  p3 S    0:00 -tcsh
23652  p0 R    0:00 ps
```

Kunkin rivin ensimmäinen numero antaa prosessin numeron. Komennolla ps on useita erilaisia argumentteja, joiden merkitys vaihtelee hieman eri unix-varianteissa. Kannattaa tarkistaa niiden merkitys manuaalisivulta.

Jos pyörivä prosessi, esim. jumiin jäänyt xterminaali halutaan keskeyttää resurssien vapauttamiseksi, niin se tapahtuu komennolla kill: `kill nro` tappaa prosessin, jonka numero nro on saatu komennolla ps. Joskus prosessi on "sitkeähenkinen" ja tarvitaan järeämpää tappokalustoa, eli komentoa kill -9.

Eri prosessien käyttämä "tehonkulululutus" näkyy komennolla `top`. Sen näyttötilasta pääsee pois kirjoittamalla q.

Komento `who` listaa koneeseen loggautuneet käyttäjät. Sama käyttäjä voi esiintyä useampia kertoja jos hänellä on useampia yhteyksiä päällä.

Komennolla `finger tunnus` näkee tietoja käyttäjästä, esim. sen milloin hän on loggautunut sisään ja kauanko yhteys on ollut käyttämättömänä. Finger komennon voi kohdistaa myös muihin koneisiin: `finger @paju` tulostaa listan koneeseen paju loggautuneista käyttäjistä. Samoin `finger tunnus@paju` antaa tietoa yhden käyttäjän loggautumisesta. Jotkut koneet eivät vastaa ulkopuolelta tuleviin fingerointeihin tietoturvallisuussyistä.

Komennolla talk voi avata keskusteluyhteyden toiseen käyttäjään, esim. `talk tunnus`. Tämän jälkeen talk-pyyntö välittyy toiselle käyttäjälle ja tämä voi valita vastaako pyyntöön vai ei.

Komento `date` antaa päiväyksen ja kellon ajan. Komento `cal` puolestaan kuluvan kuun kalenterin. Komento `cal 1998` antaa puolestaan koko vuoden 1998 kalenterin.

Edellä on kuvattu vain pieni osa Unix-komennoista. Lisätietoja haluavat voivat etsiä niitä kirjoista tai verkossa olevista oppaista. Kohtuullisen hyvä suomenkielinen unix-opas on saatavissa paikasta http://www.hut.fi/TKK/ATK/oppaat/unix/. Se on kirjoitettu Teknillisen korkeakoulun olosuhteisiin, eikä ole kaikilta osiltaan sovellettavissa Oulun yliopiston ATK-järjestelmiin.

Luku 4

IDL

IDL on tulkattava ohjelmointikieli, jonka vahvuuksia ovat:

-IDL on rakenteellinen ohjelmointikieli, jota voidaan käyttää interaktiivisesti ja jolla voidaan luoda helppokäyttöisiä aliohjelmia ja sovelluksia.

-IDL:n operaattorit ja funktiot vaikuttavat kokonaisiin taulukoihin, jolloin esimerkiksi silmukoiden tarve on vähäistä ja ohjelmat voidaan kirjoittaa muotoon, joka on sekä lyhyt, että havainnollinen.

-IDL:ssä on suuri määrä rutiineja datan esittämiseksi graafisessa muodossa, samoin suuri määrä numeerisia rutiineja.

-IDL hallitsee useita tiedostoformaatteja.

-IDL voi kutsua FORTRANilla tai C:llä kirjoitettuja rutiineja.

-IDL toimii hyvin monissa laiteympäristöissä: useissa Unix-varianteissa, Windowsissa ja MacOS:ssä. IDL koodi on hyvin siirrettävää ympäristöstä toiseen, jos välttää käyttöjärjestelmäkutsuja, niin koodi toimii yleensä sellaisenaan ja käyttöjärjestelmäkutsutkin voi rakentaa sellaiseen muotoon, että eri käyttöjärjestelmissä suoritetaan eri osiot ohjelmista. Toisaalta IDL-ohjelmien toimivuus edellyttää sitä, että koneeseen on asennettuna IDL, binäärikoodia ei nimittäin ole mahdollista luoda.

4.1 IDL:n perusominaisuuksia

IDL käynnistetään Unix-koneessa kirjoittamalla komentorivillä `idl`. Jos ei avaa ikkunoita, niin IDL:lää voi käyttää myös tekstipääteyhteydessä.

Kun IDL on käynnistynyt, niin komentoja pääsee antamaan IDL:n komentoriville:

```
IDL>
```

4.1.1 Muuttujatyypit

Kuten monissa muissakin ohjelmointikielissä, niin IDL:ssäkin on olemassa erilaisia muuttujatyyppejä. IDL:n muuttujatyypit ovat:

Byte: Etumerkittömiä kokonaislukuarvoja välillä $[0, 255]$.

Integer: Kokonaisluvut väillä $[-32768, +32768]$.

Long: Kokonaisluvut välillä $[-2^{31}, 2^{31} - 1]$

Floating-Point: Yksinkertaisen tarkkuuden liukuluvut, itseisarvot välillä $[10^{-38}, 10^{38}]$. Kuusi merkitsevää numeroa.

Double-Precision: Kaksinkertaisen tarkkuuden liukuluvut, 16 merkitsevää numeroa.

Complex: Reaaliosa-imaginääriosa pari liukulukuja.

Double-Precision Complex: Sama kaksinkertaisen tarkkuuden liukuluvuilla.

String: Merkkijono, 0 - 32767 merkkiä.

Undefined: Aiheuttaa virheilmoituksen jos käytetään operandina.

IDL:ää voi käyttää "taskulaskimena" print-komennon avulla:

```
IDL> print,5.5+8.3
      13.8000
```

Muuttujien luominen IDL:ssä on helppoa, esim.

```
IDL> a=5.5
IDL> b=8.3
IDL> c=a+b
IDL> print,c
      13.8000
IDL> print,a+b
      13.8000
```

Kuten viimeisestä komennosta käy ilmi, IDL:n komennoille ja ohjelmille voidaan syöttää argumentteina myös laskutoimituksia!

Muuttujan tyypin voi tarkistaa help-komennolla:

```
IDL> help,b
B               FLOAT     =        8.30000
```

Muuttuja B on siis yksinkertaisen tarkkuuden liukuluku. Huom! IDL:ssä pienet ja isot kirjaimet ovat samanarvoisia.

Muuttujatyypeillä tehtävistä laskutoimituksista kannattaa huomata se, että kokonaislukujen laskutoimituksista tuloksena on kokonaislukuja jne. Sekatoimituksissa tulos on "tarkimman" tekijän tyyppiä, esim. kokonaisluku + liukuluku = liukuluku. Seuraava esimerkki havainnollistaa eroja:

```
IDL> print,3/2
       1
IDL> print,3/2.0
      1.50000
```

"Yhteenlaskua" voi tehdä myös merkkijonoilla, mutta muut laskutoimitukset ovat kiellettyjä:

```
IDL> jono1='Yhteenlaskua '
IDL> jono2='merkkijonoilla'
IDL> print,jono1+jono2
Yhteenlaskua merkkijonoilla
IDL> print,jono1-jono2
% Operation illegal with strings.
% Execution halted at:  $MAIN$
```

Merkkijonojen käsittelyyn on omat komentonsa. Niistä enemmän Heikki Salon kirjoittamassa monisteessa "ATK Tähtitieteessä: IDL:n perusteita".

4.1.2 Matemaattiset operaatiot

IDL:n matemaattiset operaatiot:

$\hat{}$ potenssiin korotus
$*, /$ kerto- ja jakolasku
mod modulo
$+, -$ yhteen- ja vähennyslasku
$<, >$ pienempi kuin, suurempi kuin

4.1.3 Matemaattiset funktiot

IDL sisältää suuren joukon matemaattisia funktioita. Yksittäisten lukujen lisäksi argumentit voivat olla taulukoita.

$sin(x), cos(x), tan(x)$, missä x radiaaneina
$asin(x), acos(x), atan(x)$
$atan(y, x)$ palauttaa kulman α, jolle $y = sin(\alpha)$ ja $x = cos(\alpha)$.
$sinh(x), cosh(x), tanh(x)$
$exp(x), sqrt(x), abs(x)$ eksponenttifunktio, neliöjuuri ja itseisarvo
$alog(x), alog10(x)$ luonnollinen ja 10-kantainen logaritmi, jne.

4.1.4 Taulukot

Edellisistä muuttujatyypeistä voidaan helposti muodostaa taulukoita.

`x=bytarr(100)` luo 100 alkioisen byte-taulukon, arvot alustettu nolliksi.

`a=intarr(100)` luo 100 alkioisen kokonaislukutaulukon, jonka alkiot on alustettu nolliksi. Vastaavasti `lonarr` luo kokonaislukutaulukon, jonka luvut ovat "pitkiä" kokonaislukuja.

`b=fltarr(100)` luo 100 alkioisen liukulukutaulukon, arvot alustettu nolliksi. Ja vastaavasti kaksinkertaiselle tarkkuudelle on komento `dblarr`.

Myös merkkijonotaulukoita voi luoda: `lista=strarr(100)`. Alkiot on alustettu tyhjiksi merkkijonoiksi.

Määriteltyjen taulukoiden arvoja voidaan muuttaa esimerkiksi antamalla taulukon tiettyjä arvoja näppäimistöltä, esim.

```
IDL> b=fltarr(100)
IDL> b(10)=5.5
```

muuttaa määritellyn liukulukutaulukon alkion numero 10 arvoksi 5.5. Huom, taulukon ensimmäisen alkion indeksi on IDL:ssä 0! Toisin sanoen edellisen esimerkin taulukon b viimeisen alkion indeksi on 99, ei 100.

Usein halutaan luoda taulukkoja, joiden arvo muuttuu säännöllisesti. Tämä tapahtuu seuraavasti:

`a=indgen(100)` luo kokonaislukutaulukon, alkioiden arvot 1-99. Vastaavasti `lindgen`.

`b=findgen(100)` luo liukulukutaulukon, arvot 0.0-99.0. Vastaavasti `dindgen`.

Edellisten esimerkkien taulukot ovat olleet yksiulotteisia. Useampiulotteisten taulukoiden luonti on triviaalia, esim. `a=fltarr(200,200)` luo kaksiulotteisen taulukon jossa on yhteensä 40 000 alkiota.

Taulukon voi luoda myös seuraavasti komentoriviltä:

```
IDL> x=[0.5,-0.1,8.6,11.0,-7.2,-4.6,0.77,-18.1]
IDL> help,x
X               FLOAT     = Array(8)
```

Eli esimerkissä luotiin kahdeksanalkioinen liukulukutaulukko.

Jos taulukon alkiot haluaa järjestää suuruusjärjestykseen, niin siinä voi käyttää apuna `sort`-käskyä. Se toimii eri tavalla kuin esimerkiksi Unixin sort, sillä se ei järjestä itsessään taulukkoa, vaan palauttaa taulukon suuruusjärjestystä vastaavat indeksit (pienimmästä suurimpaan):

```
IDL> ind=sort(x)
IDL> print,ind
           7           4           5           1           0
           6           2           3
```

Jos nyt halutaan sitten muokata taulukko suuruusjärjestykseen, niin käytetään äsken määriteltyä indeksitaulukkoa: `x=x(ind)`. Tämän olisi voinut tehdä myös suoraan kirjoittamalla `x=x(sort(x))`. Jos taas taulukon haluaa järjestää kääntäen suurimmasta pienimpään, niin voi käyttää edellisten komentojen jälkeen `reverse` -käskyä: `x=reverse(x)`.

Taulukon maksimi- ja minimiarvot saadaan komennoilla `max` ja `max`:

```
IDL> print,max(x),min(x)
      11.0000      -18.1000
```

Joskus taulukoista halutaan "merkitä" niitä alkioita, jotka täyttävät tietyn ehdon. Sen voi tehdä käyttämällä `where`-komentoa:

```
IDL> ind2=where(x>0.0)
IDL> y=x(ind2)
IDL> print,y
     0.500000      0.770000       8.60000       11.0000
```

Edellisessä esimerkissä merkitään indeksillä ind2 ne taulukon x alkiot, jotka ovat suurempia kuin 0.0. Tämän jälkeen määritellään taulukko y, joka sisältää juuri nuo alkiot. Vastaavia indeksointeja on usein kätevää käyttää esim. tulostamisen yhteydessä.

4.2 Datan graafinen esittäminen

IDL tarjoaa useita erilaisia mahdollisuuksia esittää dataa graafisessa muodossa. Voidaan piirtää käyriä, kontuurikäyriä, pintoja, kuvia jne. Graafinen esitys voidaan tehdä kuvaruudulla erilliseen ikkunaan tai kirjoittaa vaikkapa postscript-tiedostoon printtaamista varten.

Ikkunan avaaminen tapahtuu komennolla `window`. Komennolle voidaan antaa useita erilaisia avainsanoja ja parametrejä. Esimerkiksi komento

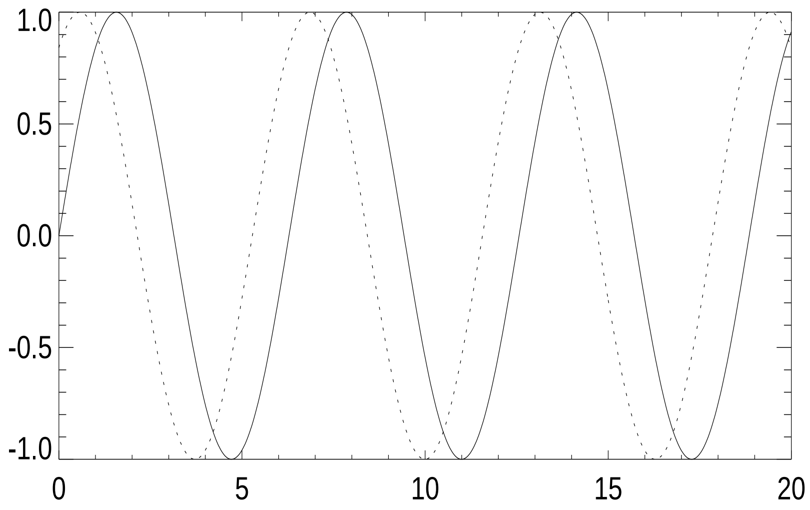
```
IDL> window,/free,xsize=700,ysize=700
```

avaa ikkunan, jonka koko on 700x700 pikseliä. Avainsana `/free` tarkoittaa sitä, että IDL etsii ikkunalle vapaan ikkunaindeksin, eikä hävitä jotain olemassaolevaa ikkunaa. Joissakin käyttöympäristöissä on myös syytä käyttää avainsanaa `retain=2`, joka liittyy ikkunan sisällöstä huolehtimiseen. Jos avainsanaa ei näissä ympäristöissä käytä, niin ikkunaan tulostettu kuva katoaa jos toista ikkunaa käyttää ikkunan päällä.

Ikkunan voi tuhota komennolla `wdelete,nro` missä nro on tuhottavan ikkunan indeksin ilmaiseva numero. Jos näytöllä on useita ikkunoita, niin halutun ikkunan voi aktivoida tulostamista varten komennolla `wset,nro`.

4.2.1 PLOT ja OPLOT

Yksiulotteisten taulukoiden sisältöä voidaan esittää komennolla `plot`. Luodaan aluksi taulukko, joka sisältää x:n arvot välillä 0-20 ja lasketaan sin(x), ja tulostetaan funktion y=sin(x) kuvaaja:



Kuva 4.1: Sini-plotti

```
IDL> x=findgen(1001)/50.0
IDL> y=sin(x)
IDL> plot,x,y
```

Usein halutaan tulostaa useita kuvaajia samaan ikkunaan. Tämä tapahtuu siten, että ensimmäinen käyrä tulostetaan samaan tapaan kuin edellä ja muut komennolla `oplot`. Käyrien erottamiseksi toisistaan, ne on syytä piirtää erilaisella viivalla. Tämä tapahtuu käyttämällä `linestyle` avainsanaa:

```
IDL> y2=sin(x+1)
IDL> oplot,x,y2,linestyle=1
```

Mikä tulostaa funktion y2=sin(x+1) kuvaajan pisteviivoituksella samaan kuvaan kuin edellä. Jos viivan piirtotyyliä ei anna, niin IDL käyttää oletusarvoa 0 = yhtenäinen viiva.

Joskus kun esimerkiksi halutaan plotata yksittäisiä mittauspisteitä joita on harvassa, on pisteitä yhdistävän käyrän sijasta mielekkäämpää plotata yksittäisiä pisteitä. Tämä tapahtuu käyttämällä `plot`- ja `oplot`-käskyissä `psym` avainsanaa. Esim. `psym=3` piirtää yksittäisiä pisteitä, `psym=2` tähtiä. Oletusarvo 0 vastaa taas pisteitä yhdistävää viivaa.

Piirrettäviin kuviin ja niiden akseleille voidaan antaa otsikkoja käyttämällä avainsanoja `title, xtitle` ja `ytitle`: `plot,x,y,title='Y=SIN(X),xtitle='X',ytitle='Y'`.

IDL skaalaa plottausalueen automaattisesti syötetyn datan perusteella. Itse plottausalueen voi määrittää käyttämällä avainsanoja `xrange` ja `yrange`:

```
plot,x,y,xrange=[0,1],yrange=[0,1.5].
```

Avainsanat voidaan kirjoittaa myös lyhyemmin `xr` ja `yr`. Vaikka plottausalueen antaisikin edellä kuvatulla tavalla, niin IDL:llä on edelleen taipumus tehdä joskus esteettiseksi kokemiaan ratkaisuja ja poiketa hieman annetuista rajoista. IDL;n saa pakotettua käyttämään täsmälleen annettuja rajoja asettamalla plottauskäskyn yhteydessä

```
plot,x,y,xr=[0,1.05],yr=[0,1.55],xstyle=1,ystyle=1.
```

Useat plottauskäskyjen yhteydessä toimivat avainsanat toimivat myös muiden graafiseen esittämiseen liittyvien komentojen, kuten `contour` kanssa.

Plottaukseen liittyy joitakin systeemimuuttujia, tällainen on esimerkiksi `!p.multi`, jota voidaan käyttää useamman kuvan plottaamiseen samaa ikkunaan.

`!p.multi=[0,3,2]` määrittelee ikkunan siten, että siihen voi plotata kuusi kuvaa, vaakasarakkeita on kolme ja pystysarakkeita kaksi. Myös useita plottauksen avainsanoista voidaan määrittää systeemimuuttijina jolloin niiden antaminen `plot`-käskyn yhteydessä on tarpeetonta, esim. `!x.range=[0,1.5]` tai `!y.title='Y'`.

Plottaamisessa on myös mahdollista käyttää värejä. Tällöin on määriteltävä väripaletti. Tästä enemmän harjoituksissa.

4.2.2 Kaksiulotteisen datan graafinen esittäminen

Usein data on kaksi- tai useampiulotteista ja `plot`-käskyllä siitä voidaan kerrallaan tutkia vain hyvin kapeaa siivua.

Yksi esimerkki kaksiulotteisesta datasta ovat harmaasävykuvat (värikuvissa saatetaan käyttää useampia ulottuvuuksia väripaletin luomiseen). Esimerkiksi JPEG-formaatissa oleva harmaasävykuva voidaan lukea IDL:ään sisään käyttäen tarvittavaa lukurutiinia. Tämän jälkeen voidaan `help`-käskyllä tutkia kuvan koko ja avataan sen kokoinen ikkuna. Lopuksi näytetään kuva `tvscl`-komennolla:

```
IDL> read_jpeg,'Mars95.jpg',kuva
IDL> help,kuva
KUVA            BYTE      = Array(600, 675)
IDL> window,/free,xs=600,ys=675
IDL> tvscl,kuva
```

Värikuvien näyttäminen tapahtuu hieman eri tavalla ja tällöin on määriteltävä RGB-taulukot, joiden avulla värit saadaan kohdalleen. Jos kuvia haluaa vain katsella, niin silloin on usein viisaampaa käyttää xv:tä.

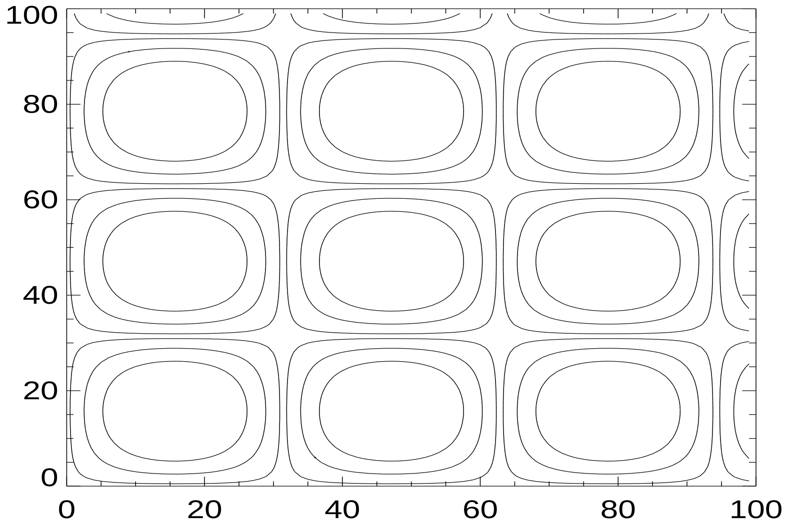
Luodaan seuraavaksi keinotekoinen kaksiulotteinen data:

```
IDL> x=findgen(100)/10.0
IDL> y=sin(x)
IDL> z=y#y
```

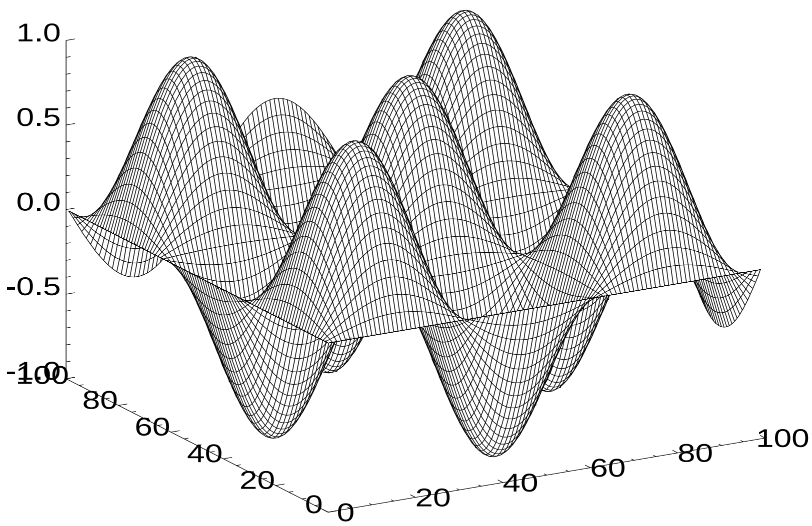
Tätä dataa voidaan havainnollistaa esim. kontuurikäyrillä tai pinnoilla. Tulostetaan esimerkiksi kontuurikäyrät:

```
IDL> contour,z,levels=[-0.5,-0.25,-0.05,0.05,0.25,0.5]
```

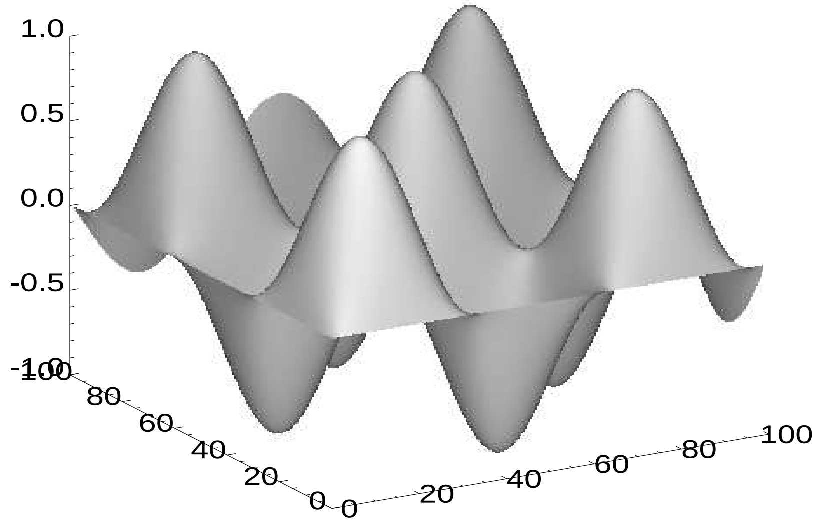
Pintojen tulostamiseen IDL:ssä on useampia komentoja. `surface`-komennolla saadaan "kanaverkko"-pinta:



Kuva 4.2: Kontuurikäyrät



Kuva 4.3: Pinta `surface,z`-käskyllä tulostettuna



Kuva 4.4: Varjostettu pinta

Varjostettu pinta puolestaan saadaan seuraavasti:

```
IDL> shade_surf,z
IDL> shade_surf,z,ax=70
```

Jälkimmäinen komento tulostaa pinnan eri suunnasta kuin ensimmäinen komento.

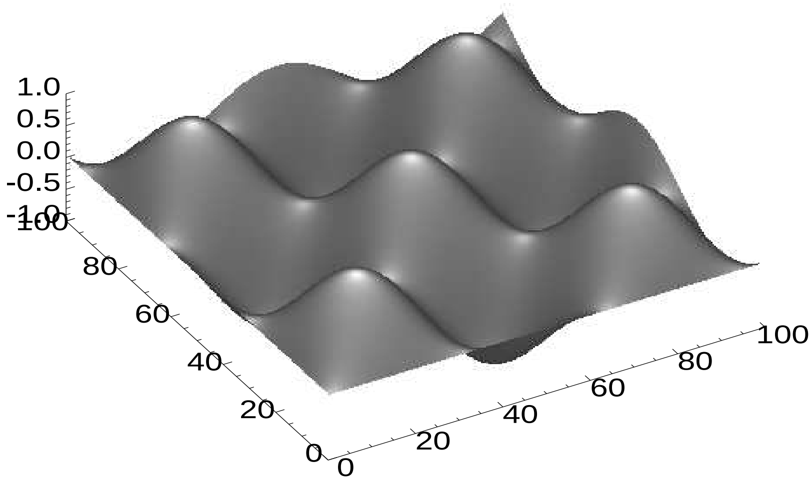
Pinnoista ja kuvista saa usein enemmän irti jos käyttää harmaasävyjen sijaan väripaletteja. Väripaletin voi valita seuraavasti:
`loadct,3`
mikä ottaa käyttöön väripaletin nro 3. On myös valikkopohjainen ohjelma nimeltä `xloadct`, jolla voi valita paletteja ja vaikuttaa värien skaalauksiin jne.

4.3 Ohjelmat, aliohjelmat ja funktiot

Edellä on käsitelty IDL:n interaktiivista käyttöä. On ilmeistä, että monimutkaisten operaatioiden suorittaminen interaktiivisesti ei ole kovin käytännöllistä. Tätä varten on olemassa pääohjelmat, aliohjelmat ja funktiot. Samoin usein tehtävistä toiminnoista kannattaa kirjoittaa ohjelmat, joita voi näppärästi kutsua komentoriviltä käsin. Ohjelmien ja aliohjelmien kirjoittamisesta ja kutsumisesta enemmän Heikki Salon kirjoittamassa monisteessa. Omien ohjelmien kirjoittamista harjoitellaan harjoituksissa.

4.4 Ohjelmakirjastot

IDL:llä on kirjoitettu ohjelmakirjastoja eri tarkoituksiin. Niissä olevat aliohjelmat ja funktiot ovat käytettävissä, jos kyseinen kirjasto sisältyy IDL-hakupolkuun. Tämän kurssin



Kuva 4.5: Varjostettu pinta toisesta suunnasta

harjoituksissa tutustutaan joihinkin tähtitieteellisten ohjelmakirjastojen ohjelmiin.

Luku 5

Emacs

Tässä käsitellään Emacsia sen verran kuin on välttämätöntä tämän kurssin kannalta. Kokonaisuudessaan Emacs on tavattoman laaja ohjelma, jota voidaan käyttää myös sähköpostin lähettämiseen ja uutisryhmien lukemiseen. Enemmän tietoa Emacsista saat esimerkiksi Emacsiin sisäänrakennetusta oppaasta, jonka saa esiin kirjoittamalla Ctrl-h t.

Kuvassa 1 on esitetty tätä dokumenttia kirjoitettaessa käytetty Emacs-ikkuna.

Editoitavissa olevaa tiedostoa sanotaan puskuriksi. On mahdollista käyttää useita puskureita samanaikaisesti, mutta tätä mahdollisuutta ei käsitellä tässä yhteydessä.

Jos esimerkiksi haluaa luoda tai käsitellä jo olemassa olevaa tiedostoa nimeltä tiedosto.txt, niin se tapahtuu komennolla `emacs tiedosto.txt &`. Tällöin Emacs avaa kyseisen tiedoston erilliseen ikkunaan taustaprosessiksi ja se xtermi, jolta komento on annettu ottaa edelleen vastaan komentoja. Jos on tekstipääteyhteydessä, niin Emacs "valtaa" tekstipäätteen. Joillakin asetuksilla Emacs ei suostu edes käynnistymään tekstipääteyhteyttä, koska se yrittää avata erillisen ikkunan, mikä ei onnistu ilman ikkunointijärjestelmää. Tällöin voi avata Emacsin komennolla `emacs -nw`, jolloin editori käyttää ko. tekstipääteyhteyttä.

Emacsissa komentoja annetaan erilaisilla näppäinyhdistelmillä. Uudemmissa emacsversioissa osa komennoista on saatavana valikosta. Pienen opettelun jälkeen näppäinyhdistelmät ovat tosin nopeampia käyttää. Näppäinyhdistelmissä on yleensä mukana joko kontrolli-näppäin Ctrl, jota pidetään komennon ajan pohjassa ja Esc-näppäin, jota painetaan ja vapautetaan ennen komennon jatko-osan kirjoittamista. Seuraavassa tärkeimpiä Emacs-toimintoja. Eri näppäimet ovat erotettu tavuviivalla.

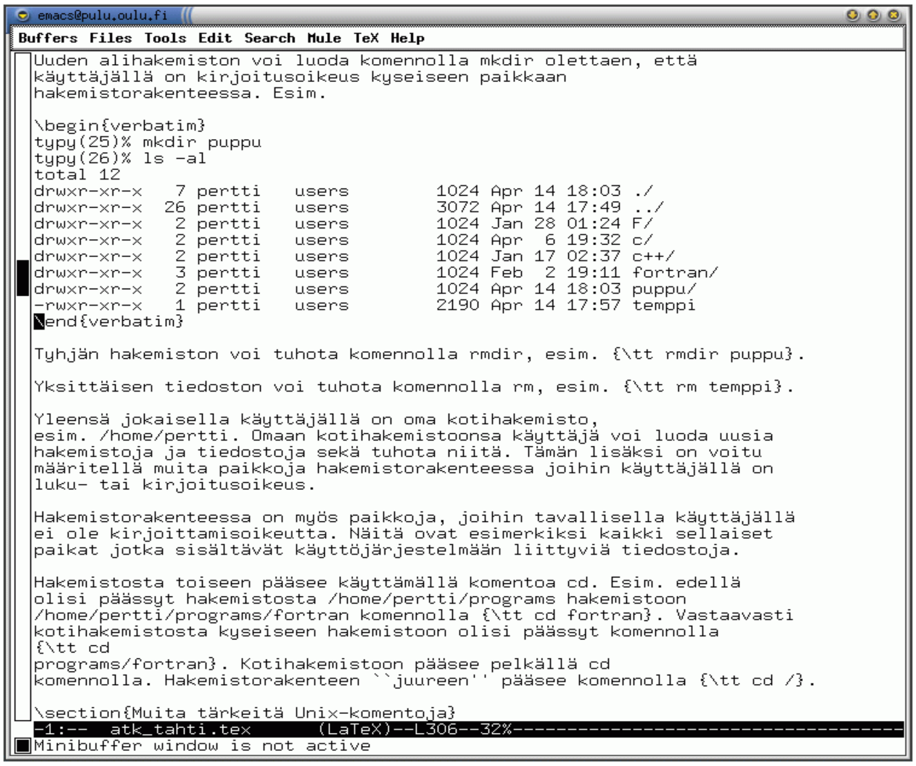
Tiedoston talletus: Ctrl-x-Ctrl-s (CTRL-näppäintä voi pitää koko komennon ajan pohjassa!)

Poistuminen Emacsista: Ctrl-x-Ctrl-c (Emacs kysyy talletetaanko tiedosto muutoksineen).

Merkkijonon etsiminen: Ctrl-s (pääsee tilaan, jossa ohjelma kysyy merkkijonon)

Siirtyminen dokumentissa paikasta toiseen tapahtuu nuoli näppäimillä. Nuolinäppäimet eivät toimi kaikissa tekstipääteyhteyksissä, jolloin on käytettävä komentoja

Ctrl-n -seuraava rivi
Ctrl-p -edellinen rivi



Kuva 5.1: Emacs-ikkuna

Ctrl-f -eteenpäin rivillä
Ctrl-b -taaksepäin rivillä.

Lisäksi on olemassa seuraavat siirtokomennot:

Ctrl-a -siirtyminen rivin alkuun
Ctrl-e -siirtyminen rivin loppuun.

Laajoissa dokumenteissa joutuu usein siirtymään niin pitkiä matkoja, että nuolinäppäinten käyttö ei ole kätevää. Tällöin voi käyttää ruutu kerrallaan siirtymistä:

Ctrl-v -ruutu eteenpäin
Esc-v -ruutu taaksepäin.

Lisäksi voidaan siirtyä:

Esc-< -siirtyminen puskurin alkuun
Esc-> -siirtyminen puskurin loppuun.

Komennolla Esc-x pääsee tilaan, jossa voi kirjoittaa useita erilaisia komentoja. Seuraavassa niistä tärkeimpiä.

`replace-string` -ohjelma kysyy korvattavan ja korvaavan merkkijonon. Muutokset kohdistuvat koko tekstiin.

`goto-line` -ohjelma kysyy mille riville mennään.