Sormet unixiin

ITKA203 Käyttöjärjestelmät -kurssin Demo 1 kesällä 2007. Paavo Nieminen / Jyväskylän yliopiston Tietotekniikan laitos.

"Superpikaintro interaktiivisen Unix-shellin käyttöön"

1 Mistä tässä harjoitteessa on kyse

Materiaali syntyi Käyttöjärjestelmät -kesäkurssilla 2007. Kiitän kurssilaisia palautteesta. Koska tämän pitäisi olla Superpikaintro, jokainen kuudesta kohdasta pitäisi mennä keskimäärin 15 minuutissa (jotkut vähemmässä, jotkut lyhemmässä ajassa). Kesäkurssin opiskelijoilla tämän tekemiseen kului vähimmillään puoli tuntia ja enimmillään noin kaksi. Jokainen ainakin väitti saaneensa tästä jotakin uutta tietoa tai taitoa irti.

Visioni on, että tämä sekä skriptiohjelmointia käsittelevä demo 4 korvaisivat IT-tiedekunnan Käyttöjärjestelmät-kurssin lisämateriaalina aiemmin käytetyn THK:n "UNIX-käyttäjän opas" -kirjasen, josta aika on hieman ajanut ohitse.

Harjoituksen tavoitteet:

- Shellin perusidea tulee sinulle tutuksi ja komentojen antaminen selkärankaan käytännön tekemisen kautta.
- Saat hieman ohjattua käytännön kokemusta shellin käytöstä; se on tärkeää, etenkin jos se on ensimmäinen kerta ikinä.
- Näet ainakin nimeltä ja päällisin puolin kertaalleen joitain perusohjelmia, joita Unix/Linux -järjestelmiin on useimmiten asennettu; näet myös joitakin unix-maisia käytäntöjä mm. käyttäjän näkökulmasta tiedostojärjestelmään.
- Opit tietämään, miten aihetta voi itsenäisesti oppia lisää tarvittaessa ja "sitten kun on aikaa".

Sisältö

1 Mistä tässä harjoitteessa on kyse **1**

2 Merkinnät ja erikoisnäppäimet **4**

3 Varoituksia / ennen kuin aloitat **5**

3.1 Pääsäännöt; muista aina 5

3.2 Pelottavia esimerkkejä 6

3.3 Apua “paniikkitilanteissa” 6

4 Tarkennuksia: Unix/Linux/GNU/BSD/... **7**

5 Osa 1: Ota pääteyhteys THK:n koneelle **9**

5.1 Jos tämä on ensimmäinen kerta ikinä 9

6 Osa 2: Tutustu etäympäristöösi **10**

6.1 Kuka olet, missä olet 10

6.2 Keitä muita on paikalla? 11

6.3 Mitä omistat täällä? 12

6.4 Mitä täällä voi tehdä 13

7 Osa 3: Opi, miten opit lisää milloin vain **16**

8 Osa 4: Tiedostonhallintaa **18**

8.1 Käydään juuressa 18

8.2 Tehdään hakemistoja 19

9 Osa 5: Temppuja **20**

9.1 Interaktiivisuuden tehostamista 21

9.2 Useamman ohjelman käyttöä, putkittaminen 22

9.3 Tulosteiden ohjaus tiedostoon 23

9.4 Syötteiden lukeminen tiedostosta 24
9.5 Vielä pari shell-kikkaa 24
9.6 Mistä saat täydellisen opastuksen käyttämääsi shelliin 25

10 Osa 6: Pakollinen palautustehtävä **25**
10.1 Tehtävä 25
10.2 Tarkastaminen 26

11 Ennakoiva osuus: tekstieditointi ja prosessit **27**

12 Motivoiva esimerkki skripteistä **28**

13 Ihan lopuksi **31**

2 Merkinnät ja erikoisnäppäimet

- `Ctrl-R` tarkoittaa näppäinyhdistelmää, jossa painat Ctrl-napin pohjaan ja sitten kirjainta `R`.
- `^R` on usein käytetty muoto `Ctrl-R`:stä
- `Alt-R` tarkoittaa näppäinyhdistelmää, jossa painat Alt-napin pohjaan ja sitten kirjainta `R`.
- `M-r` tarkoittaa Unix-dokumentaatioissa "Meta"-näppäimen käyttöä. Yleensä nykyvehkeissä Alt-näppäimen käyttö on sama asia. Jotkut pääteohjelmat eivät osaa lähettää Altia meta-näppäimenä, mutta usein voi näppäillä `ESC` eli Escape ja sen jälkeen kirjain, ja se tarkoittaa samaa.

Jotkut komennoissa tarvittavat merkit ovat hieman yllättäviä ja harvemmin käytettyjä muualla kuin shelleissä. Lisäksi ne on joskus vaikea erottaa toisistaan paperilta tai näytöltä. Tässä on mallikappaleet vaikeammin toisistaan erottuvista erikoismerkeistä, joita shelleissä käytetään:

```
~     Tilde, "mato"; tavallisesti Alt Gr:n ja enterin viereisen
      näppäimen yhdistelmänä. Tämän jälkeen pitää painaa vielä
      välilyöntiä, että varsinainen matomerkki tulee. Syy on se,
      että sen varsinainen merkitys on olla espanjalaistyylinen
      aksenttimerkki, esim. "mañana" vai miten se meni...

^     "hattu"; tavallisesti Shiftin ja enterin viereisen näppäimen
      yhdistelmänä. Sama juttu: pitää painaa välilyöntiä tai voikin
      tulla esim ê tai î ...

"     Lainausmerkki, "double quote"

’     Heittomerkki, "single quote"

‘     gravis eli vasemmalle kallistuva aksenttimerkki, "back-tick";
      tulee usein Shiftin ja BackSpacen vieressä olevan napin
      yhdistelmänä. Tämäkin on àksènttimerkkì, joten pitää painaa
      perään välilyöntiä. Näin se on ohjelmoijan elämä tehty vai-
keaksi.
      Muista Shift, ettei tule vahingossa ´ eikä ‘

{     Kiharasulku (vai mikä lie); tulee yleensä näppäilemäl-
lä Alt Gr + 7

}     Vastaava sulku toisin päin; tulee yleensä näppäilemäl-
lä Alt Gr + 0

|     Tolppamerkki; tulee Alt Gr + vasemmasta alakulmasta, jossa mm. >
```

Muita tavallisesti käytettyjä erikoismerkkejä ovat `|<>*?#!;&` -- ne erottuvat toisistaan vähän helpommin.

3 Varoituksia / ennen kuin aloitat

Luennolla nämä samat varoitukset jo mainittiin, mutta laitetaan ne tähän itseopiskelijoita varten. Shellillä komennat tietokonetta erityisen suoraan ja tehokkaasti: Jos pyydät poistamaan tiedoston, se HÄVIÄÄ! Ja jos ylikirjoitat tiedoston, sen aiempi sisältö HÄVIÄÄ! Ja kun jotain hävisi, se EI TULE TAKAISIN! Mitään tuplavarmistuksia ei ole.

Älä pelkää kuitenkaan. Noudata ohjeita, niin kaikki menee hyvin ja on mukavaa. Päivän analogia: Ranskanperunat on tehokasta uppopaistaa rasvassa, joten uppopaisto on hyvä juttu. Vaikka sormien laittaminen kuumaan rasvaan on pieni ja helppo liike, sitä ei saa tehdä sen takia että se on vaarallista eikä ollenkaan se asia, minkä haluat rasvan avulla toteuttaa. Sama juttu shellin käytössä.

3.1 Pääsäännöt; muista aina

1. Ajattele ennen kuin kirjoitat!
    - aina on oltava täysin varma siitä, mitä komento tekee tai vähintään siitä, mitä keljua se ei varmasti ainakaan tee.
    - jos et ole varma, mitä komento tekee, lue ohjeistusta kunnes olet täysin varma.
2. Ajattele vielä kahdesti ennen kuin painat enteriä!
    - komento suoritetaan vasta sitten kun painat enteriä. Silloin on viimeinen hetki perua; enterin painamisen jälkeen väärän komennon tuhot tapahtuvat silmänräpäyksessä.
3. Älä päästä kissaa, lapsia tai vastaavaa näppäimistölle!
    - Apinalta menee käytännössä ääretön aika kirjoittaa Sheakespearen tuotanto, mutta vahinko voi tapahtua yksinkertaisesti nuolinäppäimen ja enterin painamisella peräkkäin, mikä vaikuttaa vaarallisen todennäköiseltä.
4. Varo copy-pastaamasta shelliin mitään, koska teksti tulkitaan useina komentoriveinä. Vähintään on oltava sikavarma siitä, mitä leikepöydällä on ennen kuin lähtee liimaamaan sitä komennoiksi!
5. Tee silloin tällöin varmuuskopiot korvaamattoman tärkeistä tiedostoista. (Vaikka et itse hukkaisi edes vahingossa mitään, laitevika saattaa periaatteessa tuhota tiedostosi milloin vain).

3.2 Pelottavia esimerkkejä

- jos vahingossa antaisit komennon `rm -rf *` kotihakemistossasi, kaikki tiedostosi hukkuisivat iäksi. En suosittele.
- esimerkiksi, jos vahingossa antaisit komennon `echo > gradu.doc`, tiedosto `gradu.doc` tuhoutuisi täysin. Ei kannata.
- jompikumpi edellisistä (tai joku muu pommi) saattaa lymytä leikepöydällä melkeinpä odottamassa, että liität sen shell-ikkunaasi.
- jompikumpi edellisistä (tai joku muu pommi) on saattanut juuri äsken olla tarkoituksellinen komento, ja se melkeinpä odottaa, että otat sen nuolinäppäimillä vahingossa komentohistoriasta.
- Tuplavarmistuksia ei ole, vaan shell olettaa, että tiedät, mitä haluat. Siis ole varma komennoistasi.

Kesäopettaja vastaa siitä, että tämän oppaan esimerkit ovat täysin turvalliset **jos ne kirjoitetaan merkistä merkkiin sillä tavoin kuin ne on annettu**. Siihen vastaaminen loppuu. Älä siis sohlaa ellet ymmärrä. Sitten kun olet varma siitä mitä sohlaat, se muuttuu jopa suotavaksi.

Nämä varoitukset on annettava aluksi. Ymmärtänet miksi. Niistä huolimatta tämän harjoituksen päätarkoitus on **vähentää pelkokerrointa** ja tehdä yksi **tehokkaan automaattisen tietojenkäsittelyn työkalu tutuksi ja turvalliseksi**. Mitään hätää ei ole, jos noudatat yllämainittuja pääsääntöjä, etenkin numeroa yksi, *"ajattele ennen kuin kirjoitat"*, ja viisi, *"pidä varmuuskopiot"*. Toivon lukijalle tietoteknisen vapautumisen, ymmärtämyksen luoman turvallisuuden sekä tehokkuudesta syntyvän riemun kokemuksia, jollaisia interaktiivisen shellin oppiminen itselleni aikoinaan tarjosi!

3.3 Apua "paniikkitilanteissa"

" Apua! Painelen näppäimiä, mutta mitään ei tapahdu shellissä "

Jotkut normaalit tilanteet voivat näyttää siltä, että shell jumittui eikä tee mitään. Alla on pieni ongelmanselvitysohje pääteyhteydelle silloin kun se "lakkaa vastaamasta". Tärkeintä on olla menemättä paniikkiin; näppäimiä ja varsinkaan enteriä ei saa hakata hullun raivolla, koska voi tulla lisätuhoja. Todennäköisesti kyseessä on ihan normaali ilmiö.

- Yleisiä syitä:
  - Graafisen työpöydän fokus muualla kuin konsoli-ikkunassa (eli pääteyhteysikkunassa)? Klikkaa konsoli-ikkunaa ja katso auttaako.
  - `Ctrl-S` painettu vahingossa? Paina `Ctrl-Q`. Jostain syystä (veikkaan että historiallisena jäänteenä) on käytössä näppäinkomento `Ctrl-S`, joka pysäyttää sinulle päin tulevan tulostuksen. Kaikki näppäinpainalluksesi kuitenkin välittyvät edelleen palvelimelle! Tulosteet jäävät jemmaan ja ne tuodaan puskurista kerralla sinulle sitten kun painat `Ctrl-Q`. Outo homma...

  - Ohjelma jumissa / ajo kestää liian kauan? Kokeile `Ctrl-C` (stop) tai `Ctrl-Z` (suspend) ja tarkista tila.
  - Oman talon verkkoyhteys poikki? Ei voi kuin odotella. Tai voi kiljua palveluntarjoajan tukipalvelunumeroon ja odotella.

- Harvinaisempia:
  - Etäkone kaatui / buutattiin muistamatta ilmoittaa käyttäjille? Todennäköisesti varmistus ajaa koneen kohta ylös. Odottele. Toivottavasti automaattinen tallennus toimi, eikä tiedostojärjestelmä vioittunut sinun kotihakemistosi kohdalla.
  - Omassa koneessa / asennuksessa vikaa? Olet oman onnesi nojassa... onnea ylläpitotaidon opiskeluun tai kilauta nörttikaverille, joka on palveluksen velkaa...
  - Etäkoneessa viallinen asennus / buginen käyttöjärjestelmä tai shelli? Yritä rinnakkaista yhteyttä koneeseen ja tarkista jumittuneen shellin tila prosessinhallintatyökaluilla.

“ Apua! Komentoni menevät läpi ja tulosteet tulevat, mutta en näe, mitä kirjoitan. ”

- Itselleni on käynyt joskus näin; en yhtään tiedä, mistä se johtuu enkä osaa korjata. Yritä kirjoittaa sokkona komento `exit` ja ottaa yhteys uudelleen. Ja jos itse tiedät, miksi merkkien kaiutus voisi mennä rikki, kerro se heti mullekin!

“ Täysi paniikki! Kaikki on jumissa ”

- Jotkut ohjelmat voi olla pirullisia käyttää ennen kuin on lukenut niiden kryptiset ohjeet. Yksi esimerkki on Vi-tekstieditori, jossa pitää osata painella järjestyksessä napit `ESC : q !` ja enter, että pääsee pois. Muitakin ehkä on.
- Joskus ohjelmat kysyvät jotakin, johon pitäisi osata vastata jotakin. Jos et ole kiinnostunut opettelemaan asiaa manuaalista, tai ei ole aikaa, kokeile seuraavia näppäilyjä, jotka useissa ohjelmissa peruuttavat aloitetun interaktiivisen toiminnon:
  - `ESC`
  - `ESC` pari kertaa peräkkäin
  - `Ctrl-G` -- tämä on aika tyypillinen “Cancel”-toiminnon näppäin
  - `Ctrl-Q` tai `Ctrl-C` -- huomaa että nämä saattavat lopettaa koko ohjelman!
- Viimeisenä keinona sulje vaan kylmästi pääteyhteysohjelma ja ota yhteys uudelleen. Normaalisti tähän pitäisi olla hyvin vähän tarvetta.

4 Tarkennuksia: Unix/Linux/GNU/BSD/...

Jos haluat vain päästä käsiksi harjoitteisiin, voit hypätä tämän luvun yli! Koen että on pakko vähän saivarrella ja ylitarkentaa sitä, mitä kirjoittajana olen osannut kertoa, ja mitä taas en oikeastaan ole edes tarkkaan pohtinut. Ettei näistä asioista tule tarpeetonta kritiikkiä...

Tämä käytännön harjoite toteutetaan Jyväskylän yliopiston Tietohallintokeskuksen suorakäyttökoneella nimeltä `jalava.cc.jyu.fi` touko-kesäkuussa vuonna 2007. Näihin aikoihin tuossa koneessa on käyttöjärjestelmäksi asennettu Fedora Core 4, joka on ns. Linux-distribuutio eli "jakelupaketti". Distribuutio sisältää Linux-käyttöjärjestelmän ytimen sekä valikoiman apu- ja sovellusohjelmia.

Käytetään siis Linuxia. Se on suomalaisen Linus Torvaldsin 1990-luvulla alullepanema käyttöjärjestelmä, joka pohjautuu vahvasti 1970-luvulla alullepantuun Unix -käyttöjärjestelmään. Molempien käyttöjärjestelmien, Unixin ja Linuxin, erilaiset versiot ja muunnokset sekä niiden jakelupaketit (huom: sekä kaupalliset että ilmaiset) elävät ja voivat hyvin vuonna 2007 ja näköpiirissä olevaan tulevaisuuteen. Esimerkiksi valtaosa nykypäivän WWW-palvelinkoneista käyttävät jotakin Unixia tai Linuxia käyttöjärjestelmänä. Se on potentiaalinen käyttöjärjestelmä tulevaisuuden kännyköihin ja muihin kannettaviin laitteisiin. Linux-osaamista arvostetaan tällä hetkellä työmarkkinoilla.

Tarkempi käyttöjärjestelmien historia ja "sukupuu" jää oman kiinnostuksen ja tiedonhaun nojaan. Käyttöjärjestelmät-kurssilla neljässä opintopisteessä on keskityttävä toisaalta hyödyllisiin nykypäivän kädentaitoihin ja toisaalta nykyaikaisen käyttöjärjestelmän toteutuksellisiin yksityiskohtiin, vaikka historian tunteminen olisi akateemisesti välttämätöntä ja se antaisi vankan perustelun moniin nykypäivän ja tulevaisuuden ilmiöihin.

Käytän tässä oppaassa usein sanoja kuten "Unixissa tai ainakin Linuxissa", koska jotkut asiat toimivat hieman eri tavoin Linuxissa kuin Unixissa, ja itselläni on varsinaista kokemusta lähinnä Linuxista. Eroja on sitä enemmän, mitä "monimutkaisemmasta" työkalusta on kyse. Perusperustyökalut, joita tässä oppaassa nähdään, ovat käsittääkseni yleismaailmallisia kaikissa Unixeissa. Kun olen asiasta mielestäni täysin varma, sanon että "Unixeissa" näin on. Silloin on toki myös Linuxeissa. Kuitenkin jossain kulkee raja siinä, mikä on Perusperustyökalu, mikä on vain perustyökalu, ja mikä ei sitten enää ole "peruskalu" ollenkaan vaan enemmänkin itsenäinen ja erillisenä toimitettava sovellus, joka ei Unixiin tai Linuxiin kuulu. Raja on vähän häilyvä, eikä hahmottuisi kuin esim. POSIX-standardia lukemalla. Myönnän, etten ole paljonkaan silmäillyt sitä.

Mitä tämä esipuhemainen höperrys tarkoitti? Sitä, että

- Tässä oppaassa mainitut työkalut ovat ihan varmasti saatavilla, ja toimivat juuri tässä kerrotulla tavalla, jos käyttöjärjestelmänä jossakin on GNU Linux ... tai ainakin sen Fedora Core -distribuutio.
- Monet niistä ovat melko varmasti saatavilla (tai ainakin pienellä vaivalla asennettavissa), jos käytät mitä tahansa muuta Unix-varianttia, joita on siis maailmalla paljon.

Ajattele tässä olevia esimerkkeinä juuri vain esimerkkeinä, ja muista että opettelet olennaisempaa asiaa kuin joku tietty komento tai sen argumenttien muoto: Opettelet interaktiivisen shellin käyttöä, sen ideaa ja mahdollisuuksia, ja monen käyttäjän järjestelmässä toimimista pääteyhteyden kautta. Se on aina samanlaista vaikka komennot tai syntaksit olisivat mitä!

5 Osa 1: Ota pääteyhteys THK:n koneelle

Tässä oppaassa oletetaan seuraavat asiat:

- Toimit Agoran mikroluokassa tai muualla, jossa käytössäsi on Windows -käyttöjärjestelmä sekä SSH-kelpoinen yhteysohjelma (esim. Secure Shell SSH Client tai PuTTY SSH Client).
- Olet Jyväskylän yliopiston opiskelija, jolloin sinulla on käyttöoikeus THK:n suorakäyttökoneille. Käyttäjätunnus ja salasana on sinulle joskus jaettu.
  Jos näin ei kohdallasi ole, ota yhteyttä, niin mietitään ratkaisua...
- Huom: Unix-koneiden käyttö täytyy aktivoida erikseen; ohjeet ovat tuossa alla.

Seuraavassa käytetään yhteysohjelmana PuTTYä. Varo tosiaan copy-pastea; hiiriklikkaukset näyttävät toimivan PuTTY-ohjelmassa "Linux/X-mäisesti" mikä voi Windows-käyttäjää ihmetyttää (eli jo maalaaminen ikkunassa tekee välittömästi kopion leikepöydälle, ja oikeanpuoleinen nappi liittää leikkeen sisällön ikään kuin se olisi kirjoitettu tekstinä!). Toinen vaihtoehto olisi Secure Shell, mutta mun mielestä se ei ole niin mukava kuin PuTTY. Tämä on makuasia.

5.1 Jos tämä on ensimmäinen kerta ikinä

Näköjään THK:n ohjeen mukaisesti sinun pitäisi ensin käydä aktivoimassa Unix-tunnus WWW-osoitteessa http://salasana.jyu.fi/ jos et ole sitä aiemmin tehnyt:

- Linkki "Salasanan vaihto ja tunnuksen tietojen päivitys"
- "Activate UNIX" -painike

Huomioi, että samassa palvelussa voi myös asettaa kaikille käyttäjille näkyviä tietoja ja valita varsinaisen interaktiivisen shell-ohjelman, jota haluaa käyttää. Itse olen laittanut "Shell" -valinnaksi `bash` eli Bourne Again Shellin, koska joskus totuin siihen ja se on kyllä tosi hyvä. Sillä tehtäneen myöhemmin tämän kurssin skriptit, mutta interaktiivisena shellinä voit käyttää myös esim. `tcsh` -nimistä shelliä. Se lienee nykyinen oletusasetus. Kaikki tämän oppaan komennot toimivat identtisesti, käytitpä kumpaa tahansa edellämainituista. Monet edistyneemmät asiat kuitenkin olisivat erilaisia riippuen valinnasta.

Sitten pitäisi olla Unix-etäyhteys käytettävissä. Seuraavaksi:

- Etsi ja käynnistä PuTTY SSH Client -ohjelma (tai muu valitsemasi SSH-asiakasohjelma)
- Kirjoita etäkoneen osoitteeksi `jalava.cc.jyu.fi` ja klikkaa "Open"
- Ensimmäisellä kerralla tulee dialogi-ikkuna, jossa kysytään, hyväksytkö etäkoneen Host Key:n. Se ei (ehkä, välttämättä, toivottavasti) tällä kertaa tarkoita, että joku vakoilisi tietoliikennettä, vaan sitä, että Jalavan isäntäavainta ei ole vielä tallennettu paikalliselle

tietokoneelle. Voinet klikata hyväksyvästi. Jos joku ilkeä yrittää napata tietoliikennettä, tämä varokeino kertoo tarkkasilmäiselle käyttäjälle siitä. Normaalisti tämä tulee vain silloin, kun isäntäkoneella on esim. niin suuri päivitys että avain vaihdetaan.

- Kirjoita käyttäjätunnus, kun kysytään `login as:`
- Kirjoita THK:n salasanasi, kun sitä kysytään eli tulostuu `password:` (huomaa, että salasanan merkkejä ei kaiuteta näkyviin päätteellesi, tietenkään)
- Jossain vaiheessa voit tutustua kirjautumisen yhteydessä mainittuun infosivuun
  - http://www.jyu.fi/erillis/atkk/ohjeet/suorakaytto/

  ... Tämä demo-ohje tosin kertoo enemmän kuin tuo sivu ainakaan nykymuodossaan.

6 Osa 2: Tutustu etäympäristöösi

Nyt tutustut shellin käyttöön kirjoittamalla komentoja, jotka tulostavat tietoja siitä tietokoneesta, johon olet yhteydessä. Jokaisen komentorivin jälkeen pitää tietysti painaa enter. Komennot ovat lyhyitä aluksi; loppua kohti tehdään pidempiä ja pidempiä, ja tehtävien lopussa olevassa skriptissä on ihan hirmu pitkiä komentoja. Mutta liikkeelle siis perusasioista.

6.1 Kuka olet, missä olet

Kuka olet?

Komenna:

```
whoami
```

Mitä tapahtui? Shell kertoi kuka sinä sen mielestä olet. Eli käyttäjätunnuksesi tulostui.

Missä olet?

Komenna:

```
uname -n
```

Millainen paikka se on?

Komenna:

```
uname -nmo
```

Tämä kertoo millainen prosessori ja minkä tyyppinen unix etäkoneessa on. Vielä lisää tietoa:

```
uname -a
```

6.2 Keitä muita on paikalla?

Komenna:

```
who
```

Koeta tulkita tulostetta. Havainnet, että `who` kertoo, mitkä muut käyttäjätunnukset ovat tällä hetkellä kirjautuneina koneelle. Se kertoo myös, milloin kukin käyttäjä on kirjautunut ja minkä nimiseltä tietokoneelta yhteys on otettu. Jokainen koneeseen otettu yhteys näkyy yhtenä rivinä. Huomaa käyttäjän näkökulma moniohjelmointiin: Eikö vaikutakin siltä kuin itse olisit ainoa joka käyttää tietokonetta, vaikka Jalavan prosessoreita käyttää niin moni muukin yhtäaikaa! Tämä kokemuksesi on yksi nykyaikaisen käyttöjärjestelmän tavoitetilanne.

Kaikki nimet tuskin mahtuivat yhteen kuvaruudulliseen. Kokeile pääteohjelmasi vieritys-ominaisuutta. Jossain kohtaa ikkunaa on luultavasti vierityspalkki, josta näet ohi vilahtaneet rivit. Voi myös olla, että esim. näppäinyhdistelmä `Shift+Page Up` antaa edellisen sivun tulosteesta ja `Shift+Page Down` seuraavan. Tulosteita tallentavan puskurin koko on rajoitettu, joten ihan vanhimmat rivit häviävät sitä mukaa kun uusia tulee.

Komenna:

```
who | less
```

Nyt jouduitkin uuteen tilanteeseen! Ensinnäkin käynnistit kaksi ohjelmaa yhden sijasta. Minkä tahansa ohjelman tulosteen voi ohjata minkä tahansa toisen ohjelman syötteeksi käyttämällä tolppamerkkiä | yllä olevalla tavalla. Asiaan palataan myöhemmässä osiossa.

Lisäksi viimeksi käynnistämäsi ohjelma `less` on itsessään interaktiivinen. Sen tarkoitus on näyttää pitkä teksti pieni pätkä kerrallaan. Voit selata tulostetta nuolinäppäimillä ylös ja alas. Less-ohjelmasta pääsee pois näppäilemällä `Q`.

Pääteyhteydellä ei siis tarvitse käyttää pelkästään shelliä tai yksinkertaisia apuohjelmia. Mitä tahansa tekstimuotoisia ohjelmia (tai siis niin sanottuja konsoliohjelmia) voidaan käynnistää ja käytellä pääteikkunassa. Ja jos etäkoneella ja omalla koneella on yhteinen ikkunointijärjestelmä, voi nopean nettiyhteyden yli kuljettaa grafiikkaakin.

Kuka on kukin?

Komenna:

```
finger
```

Tarkkaile, miten tuloste on erilainen kuin edellä `who` -komennon tulostama. Jos haluat, voit käyttää `less`-"sivuttajaohjelmaa" jälleen putken avulla:

```
finger | less
```

Valitse joku kirjautuneista käyttäjätunnuksista tai joku tuntemasi kaveri, ja katso, mitä saat tietää hänestä THK:lta. Esim. komenna:

```
finger -m nieminen
```

Vertaa seuraavaan:

```
finger nieminen
```

Kokeile myös seuraavia ja vertaile tulosteita:

```
finger liisa
```

```
finger -m liisa
```

Käytä fingeriä joka tapauksessa tunnukseen `nieminen`. Huomaat, että siellä näkyy käyttäjän itsensä asettamia tiedotuksia. Jokainen voi asettaa näitä itselleen tietyllä tapaa, mutta ei käsitellä asiaa tässä tarkemmin. (Jos jossain vaiheessa asetat itsellesi `.plan` -tiedoston, pidä tarkoin huolta hakemistojesi käyttöoikeuksista, ja ymmärrä niiden vaikutus.)

Mitä kukin tekee?

Komenna:

```
ps -ef
```

Mitä tapahtui? Komento tulosti kaikkien käyttäjien kaikki päällä olevat ohjelmat. Niitäkin on luultavasti aika monta koko ajan. Älä vielä tässä vaiheessa huoli siitä, mitä kaikki tulosteen tiedot ovat. Ne selviävät varmasti Käyttöjärjestelmät -kurssin edetessä yksi asia kerrallaan. Ensimmäinen sarake on käyttäjätunnus ja viimeinen on se komento, jolla käyttäjä käynnisti ohjelman. Huomannet, että `root` -niminen käyttäjä on käynnistänyt paljon `sshd`-ohjelmia. Itse asiassa tämä on sitä, kun käyttöjärjestelmän valtuuksin (root) toimiva SSH-palvelin on käynnistänyt login-shellejä etäyhteyksille. Sinullekin pitäisi tällä hetkellä löytyä tuollaiset rivit `ps`:n tulosteesta.

6.3 Mitä omistat täällä?

Komenna:

```
pwd
```

Jos mitään ihmeellistä ei ole tapahtunut, olet yhä kotihakemistossasi, johon sisäänkirjoittautumisen jälkeen shellissä päästään (tai ehkä jossain on joku asetus, jolla voi määrätä shellin

aloituksen eri paikkaan kuin kotihakemisto, en tiedä, koska koskaan en ole tarvinnut sellaista ominaisuutta). Komento `pwd` kertoo nykyisen oleskeluhakemiston. Se on jotain vastaavaa kuin `/autohome/home3/363/nieminen`.

Huomaa, että Unixissa hakemistonimissä on hakemistojen erottimena kauttaviiva `/`. Windowsissa vastaavaa tarkoitusta toimittaa kenoviiva `\`.

Unixin tiedostojärjestelmän "juuri" on hakemisto `/` eli pelkkä kauttaviiva. Koko hakemistorakenne jäsentyy siitä alkaen. Tällainen rakenne on suoraviivaisempi ja monen mielestä loogisempi kuin esim. Windowsin rakenne, jossa on eri juuret eli "levyasemille" eli `A:`, `B:`, `C:` jne.

Komenna:

```
ls
```

Pitäisi tulostua kotihakemistosi sisältö. Huomaa, että se on sama kuin Windowsin `U:` -asemasi. Kumpikin hakemistopolku viittaa samaan paikkaan jaetulla verkkolevyllä. Fyysinen kovalevy voidaan siis jakaa ihan eri käyttöjärjestelmien ja tietokoneiden välillä! Ja voit olla kirjautuneena useaan tietokoneeseen, joissa kaikissa näet yhden ja saman verkkolevyn sisällön ikään kuin se olisi yksi paikallisen koneen levy.

6.4 Mitä täällä voi tehdä

Ensinnäkin voit huutaa ja katsoa, miten kaiku vastaa. Komenna:

```
echo Miten kaiku vastaa
```

Eli komento `echo` ei tee muuta kuin tulostaa omat argumenttinsa. Se on erityisen hyödyllinen esim. skripteissä, joiden halutaan ehkä tulostavan jotakin. Skriptin suorittama "kaiku" päätyy nimittäin esimerkiksi käyttäjälle, joka suoritti skriptin. Tai se voi tallentua lokitiedostoon (kohtapuoleen katsotaan, miten echon tai minkä tahansa ohjelman tulosteet voi kirjoittaa tiedostoihin; olet jo alustavasti nähnyt, miten ne voidaan ohjata toisen ohjelman syötteiksi putkella).

Tekstitiedostojakin täällä voi tulostaa päätteelle. Komenna:

```
cat /etc/group
```

Eli komento `cat` on sellainen, että kun sille antaa argumenttina tiedoston nimen, se lukee tiedoston ja tulostaa sen sisällön. (Varsinainen käyttö on "katenointi" eli usean tiedoston tulostaminen peräkkäin; siitä nimi `cat`.)

Esimerkin tiedosto `/etc/group` löytyy usein Unix-asennuksista (ehkä jopa aina? pitäisi tarkistaa standardi...); se on tietyn muotoinen tekstitiedosto, joka kertoo koneelle määritellyt käyttäjäryhmät.

Eli tällaisia peruskomentoja kuin `echo` ja `cat` löytyy... Ja pidempi teksti olisi ehkä parempi lukea interaktiivisella `less`-ohjelmalla kuten edellä tehtiin. Putken sijasta tiedoston voi kertoa lessille argumenttina:

```
less /etc/group
```

(Lessistä tosiaan pääsee pois painamalla `Q` ...)

Mitäs muuta etäkäyttö-Unixilla voi tehdä?

Komenna:

```
ls /bin/
```

Tulostui noin 90 kappaletta apuohjelmia. Tutki listaa; löydätkö edellä käytettyjen komentojen nimiä? Hakemisto `/bin` on tyypillisesti Unixissa perusapuohjelmien paikka. Siitä löytyvät asiat ovat melko varmasti käytössä kaikissa unixeissa. Ja melko varmasti ne löytyvät kaikkialla nimenomaan hakemistosta `/bin`. Muistanet, mistä kaikki suoritettavat ohjelmat tulevat: ne on jossain vaiheessa käännetty lähdekoodista konekieliseksi binääritiedostoksi. `bin`-nimisissä hakemistoissa on usein sellaisia ja lisäksi tulkattavia ohjelmia kuten shell-skriptejä. Jokainen voi tehdä itselleen oman kotihakemiston alle kokoelman omia ajettavia ohjelmia ja skriptejä (`/home/kotihakemistosi/bin`).

Sitten laajennetaan tajuntaa... pidä tuolista kiinni, kun komennat seuraavan.

Komenna:

```
ls /usr/bin/
```

Tulostui noin 2600 kappaletta sovellusohjelmia. Niillä voi tehdä kaikenlaista, voin kertoa. Useille Linux-distribuutioille on saatavilla yli 5000 valmiiksi paketoitua ohjelmaa, joilla voi tehdä kaikenlaista. Tietohallintokeskuksemme sedät ja tädit ovat valinneet jonkinlaisen harkinnan perusteella Jalavaan asennettavaksi juuri nuo siellä olevat 2600 ohjelmaa. Jos niitä olisi pari tuhatta enemmän, saattaisi alkaa “Dependency hell” -niminen ongelma, jossa ohjelmien asennukset ovat keskenään ristiriitaisia. (Ne kun on rakennettu kerrosmaisesti toistensa päälle virtuaalikonehierarkiaksi, ja sitten joku onneton menee ja muuttaa rajapintaa kun kerroksesta tulee uusi versio!) Ongelma saattaa korjautua ajan kanssa, kunhan ohjelmistojen toteuttajat saavat kehitettyä parempia standardeja ja paketointityökaluja. Tietysti jo 5000 ohjelmalla on aika monta eri kehittäjää, ja kommunikaatiossa on selviä rajoituksia.

Turha varmaan yrittää käydä läpi kaikkia noita ohjelmia. Ennemmin kannattaa ehkä keksiä, mitä haluaisit tietokoneella tehdä, ja sitten etsiä Internetistä jollain hakusanoilla tieto siitä, minkä nimisellä ohjelmalla se voidaan Linuxissa tehdä. Tai kysy kokeneemmalta Linux-käyttäjältä vinkkiä. Vaikka onhan mahdollista tutustua mielenkiintoisen nimisiin ohjelmiin sattumanvaraisestikin. Seuraavassa osuudessa katsotaan, miten käyttöohjeet ovat shellissä aina lähellä. Ennen sitä kerrataan kuitenkin havainnot shellistä tähän asti:

- Shell antaa sinun kirjoittaa komennon ja odottaa että painat enter-näppäintä.
- Sitten shell suorittaa komennon. Usein se on jonkun ohjelmatiedoston nimi ja ohjelmalle välitettävä argumenttilista.
- Argumenteista muodostuu siis lista. Eri argumentit erotetaan toisistaan välilyönnillä.

  Esimerkiksi Javalla tehty ohjelma vastaanottaa argumentit `main`-metodin `String[] args` parametrissa.
- Jos yhteen argumenttiin haluaa sisällyttää välilyönnin, se on mahdollista, mutta se pitää koodata eri tavoin, koska muuten argumentit tulkitaan useina erillisinä (koska välilyönti on erotinmerkki). Pari mahdollista tapaa:

  ```
  jokukomento argum1 "argum2 jossa on välilyöntejä" argum3

  jokukomento argum1 'argum2 jossa on välilyöntejä' argum3

  jokukomento argum1 argum2\ jossa\ on\ välejä argum3
  ```

  Eli argumentti lainausmerkkien sisälle tai välilyönnin koodaus kenoviivalla + välilyönnillä eli "`\ ‘‘". (Lainausmerkit ‘‘`" ja ' tarkoittavat hieman eri asiaa! Selvitä asia jostain lähteestä myöhemmin...). Kaikki ylläolevat komennot siirtävät kolme argumenttia ohjelmalle. Esim. komento:

  ```
  java JokuLuokka risto "aliisa lassi" pena
  ```

  aiheuttaa Java-ohjelman JokuLuokka alkuun tilanteen, jossa pätee:

  ```
  args[0].equals("risto") == true
  args[1].equals("aliisa lassi") == true
  args[2].equals("pena") == true
  ```

  Ja edelleen, jos argumentissa on oltava " niin se pitää koodata \" -- jos pitää olla ' niin se pitää koodata \' ja samoin tietyt shellin erityismerkit. Puhutaan "escape sequencestä", olisiko suomeksi "eskeippimerkki", jolla "paetaan" yhden merkin ajaksi normaaleista tulkintasäännöistä.

  Tämä on sama periaate kuin useissa ohjelmointikielissä esim. rivinvaihdon tai tabulaattorimerkin koodaus vakiomerkkijonossa (rivinvaihto `\n`, tabulaattori `\t`).
- Ohjelman ns. "valitsimet" (*options*) ovat Unix-maailmassa usein argumentteja, joiden muoto on `-v` tai `--verbose` eli niissä on tavuviiva ja kirjain tai vaihtoehtoisesti kaksi tavuviivaa ja kuvaavampi sana. Yksikirjaimisia argumentteja saa useimmiten yhdistää peräkkäin, esim. `-xvzf` paitsi jos ne edellyttävät että seuraava argumentti on esim. tiedostonimi, tyyliin `-o outputfilename`. Argumenttien käsittely on täysin ohjelmasta riippuvaa, ja ohjelman käyttöohjeisiin täytyy tutustua ennen kuin lähtee sitä käyttämään.

  (Windows-maailmassa usein valitsimet ovat yksikirjaimisia ja alkavat kauttaviivalla.)

- ”Interaktiivinen shell“ tarkoittaa tätä, että editoit aina komennon kerrallaan interaktiivisesti. Toinen tapa käyttää shelliä on skriptit, joissa shell suorittaa peräkkäin komentoja, jotka on kirjoitettu ohjelmaksi - puhutaan shell-ohjelmoinnista tai skriptien tekemisestä. Shell voidaan siinä mielessä ajatella tulkattavaksi ohjelmointiympäristöksi, vaikka se on ohjelmointimielessä heppoinen oikeisiin ohjelmointikieliin verrattuna.

7 Osa 3: Opi, miten opit lisää milloin vain

Komenna:

```
man ls
```

Tämä näyttää apuohjelman `ls` manuaalin eli käyttöohjeen. Perusasetuksilla manuaali näyttää pitkän tekstin käyttämällä `less`-ohjelmaa, johon tutustuit edellisessä kohdassa. Muistanet, että tekstiä voi selata ylös ja alas nuolinäppäimillä, ja less-ohjelmasta pääsee pois painamalla `Q`-näppäintä. Kokeile.

Ohjelman `ls` man-sivun eli käyttöohjeen alku sisältää tällaista:

```
NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List information about the FILEs (the current directory by
       default).  Sort entries alphabetically if none of -cftuSUX
       nor --sort.

       Mandatory arguments to long options are mandatory for short
       options too.

       -a, --all
              do not hide entries starting with .

       -A, --almost-all
              do not list implied . and ..

       --author
              print the author of each file

       ...
```

Ja tällaisiapa man-sivut yleensä ovat. Niissä kerrotaan ohjelman nimi ja tarkoitus (NAME). Sitten kerrotaan tapa, jolla ohjelmaa saa komentaa (SYNOPSIS); eli komennon jälkeen voi yleensä antaa tietyssä järjestyksessä tietynlaisia argumentteja, joista osa voi olla pakollisia ja osa voi olla mahdollisia. Ei-pakolliset on kirjoitettu hakasulkeisiin. Tässä tapauksessa komennon perään voi laittaa tarkentimia OPTION jotka ovat siis vapaaehtoisia, koska [OPTION] on hakasulkeissa. Tarkentimien jälkeen voi olla tiedoston tai hakemiston nimi FILE, joka siis sekin on vapaaehtoinen. Molempia voi olla monta. Näiden jälkeen DESCRIPTION -osa kertoo tarkemmin, mitä ohjelma tekee. Ja sitten alkaa tarkentimien kuvaus. Yleensä ohjelmilla on mahdollisia tarkentimia aika paljon.

Selaa huvikseen pari man-sivua, esim. käyttöohjeet edellä kokeilluista komennoista:

```
man pwd

man finger
```

Vielä enemmän tietoa Jalavaan asennetuista ohjelmista tarjoaa GNU -ohjelmistoissa yleisesti käytetty `info`-järjestelmä. Kokeile:

```
info info
```

Info opastaa nyt infon käyttämistä. Info on itse asiassa interaktiivinen, tekstimuotoinen hypertekstijärjestelmä, jonka sisältöihin kannattaa tutustua joskus kun on aikaa. Info-lukijan oma intro saadaan päälle painamalla `H`. Ja tästähän se opas alkaisi:

```
1.2 How to use Info
===================

You are talking to the program Info, for reading documentation.
```

Monet tekstimuotoiset Unix-sovellukset ovat varsin hyvin dokumentoituja ja käyttäjäystävällisiä -- se ystävällisyys pitää vain hahmottaa kovan kuoren alta, joka siitä tekstimuotoisuudesta syntyy...

Melkein kaikesta on joko man-sivu tai info-dokumentaatio. Ja hyvän komentoriviohjelman tunnistaa mm. siitä, että sitä voi komentaa esim. näin:

```
rm --help
```

Silloin ohjelma itse tulostaa pienen perusavustuksen. Huomaa muuten, että tiedoston poistamisväline `rm` ilmoittaa avustuksessa: "*Note that ... it is usually possible to recover the contents*". Tämä tarkoittaa, että vieraan valtion agentti voi varastaa kovalevyn hyvin pian tiedoston poiston jälkeen ja tietyin keinoin palauttaa ainakin osia tiedostosta. Tietoturvamielessä kannattaisi kirjoittaa ensin arkaluontoisen datan päälle jotain merkityksetöntä tauhkaa. Tämän tekisi tiedoston silpomisväline `shred`.

Eli kiteytetään vielä:

- Tiettyyn Unix-koneeseen asennetut ohjelmat löytyvät usein näillä tai vastaavilla komennoilla:

```
ls /bin/
ls /usr/bin/
ls /usr/local/bin
```

- Jokaisesta ohjelmasta saa tietoja (Internetistä, painetuista manuaaleista, ja sen lisäksi paikanpäällä...) komentamalla:

```
man ohjelmannimi
ohjelmannimi --help     # tai -h tai -? ...
```

8 Osa 4: Tiedostonhallintaa

8.1 Käydään juuressa

Komenna:

```
cd ..

ls

pwd
```

Mitä tapahtui? Siirryit hakemistopuussa yhtä askelta ylöspäin. Äsken olit Unix-kotihakemistossasi (joka siis Windowsissa näkyy U: -asemana). Nyt oletkin siinä hakemistossa, joka sisältää kotihakemistosi. Shell pitää kirjaa työhakemistosta ja `pwd` näyttää, mikä se tällä hetkellä on.

Komenna:

```
cd /
```

Mikäs nyt on työhakemisto? Komenna:

```
ls
```

Näyttääkö jotakuinkin tältä:

```
adminhome    autowww  home     ithome2     mailhome  proc     sys
autohome     bin      home1    ithome3     media     root     tmp
automail     boot     home2    lib         misc      sbin     usr
automisc     dev      home3    lib64       mnt       selinux  var
autowebhome  etc      ithome1  lost+found  opt       srv      wwwhome
```

Kyseessä on Linux-koneen juurihakemisto, jonka alla kaikki on. Näillä hakemistoilla on useimmissa unix/linux -asennuksissa nämä samat nimet, ja niillä kullakin on erityismerkitys. Hakemistot `/bin` ja `/usr/bin` tulivatkin tutuiksi aiemmin. Esimerkkejä muista hakemistoista: `/sbin` sisältää järjestelmänhallintaan liittyviä ohjelmia, joihin on asiaa vain pääkäyttäjällä eli "rootilla". `/lib` ja `/usr/lib` sisältävät aliohjelmakirjastoja, joita voidaan linkittää ohjelmiin. `/lib64` näköjään sisältää aidosti 64-bittisiä kirjastoja. `/home` on yleensä se, missä kotihakemistot sijaitsevat. Yliopistolla on niin paljon porukkaa, että kotipäähakemistojakin on monta, eli näyttää olevan `home1`, `home2` ja `home3`. Hakemistossa `/boot` on käyttöjärjestelmäytimen käännetty konekielikoodi, joka käynnistyksen yhteydessä ladataan. `/var` sisältää mm. lokitietoja. `/etc` sisältää eri ohjelmien asetuksia. `/tmp` sisältää ohjelmien väliaikaisia tiedostoja. Eksoottisempia hakemistoja ovat `/proc` ja `/dev`. Ensiksi mainitussa on ajonaikaisten prosessien tietoja ja jälkimmäisen kautta pääsee käsiksi laitteisiin. Unixissa (ja "perillisessä" Linuxissa) tiedostojärjestelmän kautta hallitaan muutakin kuin tiedostoja! Tai siis kaikki asiat näyttäytyvät tiedostoina tiedostojärjestelmän ja hakemistorakenteen kautta.

Unixin hakemistorakenne juuresta alkaen on monen mielestä looginen ja selkeä. Fyysisten kovalevyjen eri tiedostojärjestelmät, verkkolevyt ja jopa ohjelmat ja laitteetkin löytyvät nätisti omilta paikoiltaan, ja kaikki ovat saman juurihakemiston `/` alla.

Löydätkö kotiin? Eli muistatko ulkoa kotihakemistosi nimen, jos nyt pitäisi mennä esim. niinku meikäläinen:

```
cd home3
cd 363
cd nieminen
```

Ei hätää, jos et muista. Komenna:

```
cd
```

Kokeile esim. `pwd`-ohjelmalla, että olet taas kotihakemistossasi. `cd` -komento ilman parametreja siirtää kotihakemistoon.

Itse asiassa `cd` ilman parametreja on sama kuin komento:

```
cd $HOME
```

missä `$HOME` on ns. ympäristömuuttuja (*environment variable*). Mutta ympäristömuuttujista lisää varmaan jossain myöhemmässä harjoituksessa.

8.2 Tehdään hakemistoja

Nyt tehdään verkkolevyllesi hakemistot Käyttöjärjestelmät-kurssin kaikkia demoja varten. Saat valita minkä tahansa hakemistonimen; esimerkissä on `kj07kesa` koska mun mielestä se

on kiva nimi hakemistolle. Ääkkösiä ja erikoismerkkejä kannattaa välttää tiedostonimissä aina, koska niiden koodauksen standardit laahaavat perässä nykyistenkin järjestelmien välillä, ja maailmalla on käytössä laitteita, joiden tiedostojärjestelmät ovat peräisin vuosikymmenten takaa. Välilyöntejä en suosittele, koska ne on Unixissa vähän hankalia (vaikka ei sen kummemmin kuin että shellissä pitää kirjoittaa kenoviiva välilyönnin eteen...).

Komenna:

```
mkdir kj07kesa
```

Sitten mene juuri luotuun hakemistoon:

```
cd kj07kesa
```

Jos haluat varmistua hakemiston vaihtumisesta, voit käyttää taas `pwd`-ohjelmaa...

Komenna seuraavasti:

```
mkdir demo{1,2,3,4}
```

Tarkista tulema komentamalla:

```
ls
```

Mitä tapahtui? Interaktiivisissa shelleissä on tiettyjä tehokäyttöominaisuuksia. Yksi esimerkki on tämä, jossa komentoriville kirjoitettu teksti `jotakin{muuta,puuta,juu}` korvautuu ennen komennon suorittamista yhdistelmällä `jotakinmuuta jotakinpuuta jotakinjuu`. Kun tästä tulee refleksi, muuttuu aika monen asian tekeminen nopeammaksi.

Kokeile:

```
echo {aa,bee}{1,2,3}{X,Y}
```

Jännää, eikö. Jätetään ”asia hautumaan“ ja siirrytään seuraavaan.

9 Osa 5: Temppuja

Eri shellit toimivat monella tapaa erilaisesti, mutta jotkut asiat ovat yhteisiä monille niistä. Ainakin seuraavien ominaisuuksien pitäisi toimia ainakin sekä `bash` että `tcsh` -shelleissä. Vastaavaa on käytettävissä kaikissa järkevissä shelleissä, mutta mm. näppäinyhdistelmät voivat olla ihan erilaisia.

9.1 Interaktiivisuuden tehostamista

Pitkien rimpsujen kirjoittaminen on hidasta (verrattuna esim. klikkaamiseen hiirellä), mutta näppäimistön käyttö sinänsä on nopeata, jos näppäinpainallusten määrää saadaan rajoitettua.

Kokeilepa seuraavaa:

- Paina nuolta ylöspäin ja katso mitä tapahtui! (Saat edellisen komennon, jonka voit antaa uudelleen ihan vaan painamalla enteriä). Komennon toistaminen vaatii siis vain kaksi näppäinpainallusta.
- Painelepa nuolta ylöspäin useampia kertoja. Saat aina edellisen ja edellisen komennon.
- Nuoli alaspäin selaa tietysti toiseen suuntaan.

Tämän nimi on *komentohistoria* ja se on kätevä juttu. Jos haluat uudelleen komennon, jonka teit pari askelta sitten, se löytyy muutamalla painalluksella. Sitten katsotaan rivin editointia:

- Selaa historiasta tuo äskeinen komento:

  ```
  echo {aa,bee}{1,2,3}{X,Y}
  ```

  Nuolilla vasemmalle ja oikealle voit siirtää kursoria komentorivillä vanhan komennon päälle. Muuta se tuollaiseksi siten, että vaihdat vain `X`-kirjaimen `Z`-kirjaimeksi:

  ```
  echo {aa,bee}{1,2,3}{Z,Y}
  ```

  Suorita.

  Suorita vielä komentohistoriaa käyttämällä nämä vähän eri komennot:

  ```
  echo {aa,bee}{1,2,3}{W,Y}

  echo {aa,bee}{1,2,3}{kuu,Y}
  ```

- Katsotaanpa lisää näppäinpainallusten välttämistä. Ota historiasta vielä tuo edellinen, melko pitkä, komentorivi editoitavaksi. Kokeile näppäimiä `Ctrl-A` ja `Ctrl-E`. Rivin alkuun ja loppuun hyppiminen on nopeaa!

Komentorivin editointi on tehty nykyisissä shelleissä monella tapaa tehokkaaksi. Tässä vielä yksi tapa:

- Kirjoita tyhjän rivin alkuun `less /u` ja paina sen jälkeen tabulaattoria eli sarkainnäppäintä. Huomannet, että shell osaa tutkia hakemistorakennetta ja täydentää sinne hakemiston tai tiedoston nimen aina siihen asti kuin se on yksikäsitteinen. Tässä tapauksessa komentorivilläsi pitäisi olla edellisten kahdeksan näppäilyn jälkeen teksti `less /usr/` Jatka kirjoittamalla perään `i` eli `less /usr/i` ja paina taas tabulaattoria. Nyt rivillä on `less /usr/include/`. Näppäile `oc`, tabulaattori kaksi kertaa, `ve` ja vielä kerran tabulaattori. Rivillä on nyt seitsemäntoista näppäinpainalluksen jälkeen komento:

```
less /usr/include/octave-2.9.8/octave/version.h
```

Kun tästä tulee refleksi, on tiedostojen löytäminen erittäin syvistäkin hakemistorakenteista helppoa ja nopeaa interaktiivisen shellin avulla.

Voit myös suorittaa tuon viimeisimmän komennon, jos haluat: Less-ohjelmallahan voi selata tekstitiedostoja nuolinäppäimillä interaktiivisesti. Ja `Q` lopettaa sen. (Octave, jonka lähdekoodista otsikkotiedosto `version.h` on, on avoimeen lähdekoodiin perustuva matemaattinen laskentaohjelma, vähän Matlabin tyyppinen.)

9.2 Useamman ohjelman käyttöä, putkittaminen

Kokeile seuraavia perusesimerkkejä Unixin ideologiasta ”paljon pieniä ohjelmia, jotka tekevät yhden asian hyvin, ja joita voi yhdistellä monimutkaisempien ongelmien ratkaisemiseksi“. Niissä käytetään aiemmin `less`-ohjelman yhteydessä nähtyä putkittamista:

```
who | sort
```

Who-ohjelman tuloste menee sortille, joka lajittelee rivit aakkosjärjestykseen. Kokeile sitten seuraavaa:

```
who | cut -c-8 | sort | uniq
```

Who-ohjelman tuloste menee tässä cutille, joka leikkaa kahdeksannen merkin kohdalta aina rivin poikki; sitten leikatut rivit menevät cutilta sortille ja lopulta ohjelmalle `uniq` joka jättää peräkkäisistä samoista riveistä vain yhden jäljelle. Lopputuloksena on aakkosjärjestyksessä kaikki käyttäjät vain yhteen kertaan listattuna, vaikka heillä olisi useita rinnakkaisia istuntoja auki. Vielä yksi:

```
who | cut -c-8 | sort | uniq | wc -l
```

Tässä vielä `uniq`-ohjelman tulos putkitetaan `wc`:lle, joka laskee rivien määrän (tosiaan viimeinen merkki komennossa on ällä eikä ykkönen). Lopputuloksena on shell-komento, joka laskee konetta käyttävien henkilöiden määrän.

Huom 1: Pitkät putkitukset ohjelmalta toiselle ovat tuhlailevaisia, koska jokainen ohjelma on käynnissä yhtäaikaa. Jokainen niistä vaatii käyttöjärjestelmän resursseja, joiden rajallisuuteen tällä kurssilla tutustutaan piakkoin. Putkia on kuitenkin periaatteessa mahdollista tehdä, ja yllä olevat olivat esimerkkejä siitä, miten ne toimivat. Ne ovat käteviä, mutta ylenmääräistä putkittamista on hyvä pyrkiä välttämään oikeissa shell-skripteissä.

Huom 2: Putki on synkronoitu tuottaja-kuluttaja -toteutus, jonka Unix-järjestelmä tarjoaa palveluna. Tähänkin syvennytään kurssilla myöhemmin. Ohjelmissa nähdään standardi syöttövirta ja standardi tulostusvirta. Mistä putki tulee sisään tai mihin se vie ulostulosta, on määrättävissä ulkopuolelta, esim. shellin komentoriviltä.

9.3 Tulosteiden ohjaus tiedostoon

Putkien käytön lisäksi on hyötyä tietää, miten shell-komennoissa pääsee käsiksi tiedostoihin. Varmistu taas ensin, että olet siellä äsken tekemässäsi `demo1` -hakemistossa, jos olet tehnyt välillä kokeiluja.

Kokeile ensin grep-ohjelman toimintaa:

```
grep Section /etc/X11/xorg.conf
```

Näin komennettu grep etsi toisena argumenttina annetusta X-ikkunointipalvelimen konfigurointitiedostosta kaikki rivit, joilla esiintyy ensimmäisenä parametrina annettu merkkijono ”Section“. Grep on todella hyödyllinen väline isojen tekstimassojen selaamiseen.

Komenna:

```
grep Section /etc/X11/xorg.conf > sektionit.txt
```

Mitään ei tulostunut näytölle. Syy on se, että väkäsellä `>` pyysit ohjaamaan `grep` -ohjelman tulosteen tiedostoon, jonka nimi on `sektionit.txt`. VARO VAARAA taas: Väkäsellä ohjattaessa tiedoston mahdollinen aiempi sisältö tuhoutuu ja se korvataan uudella. Totea että tiedosto `sektionit.txt` on syntynyt. Komenna:

```
ls -l
```

Näet mm. jokaisen tiedoston viimeisen muutosajankohdan ja pituuden tavuina. Totea tiedoston sisältö:

```
cat sektionit.txt
```

Ohjelma `cat` tosiaan tulostaa argumenttina annetun tiedoston. Putkia voi yhdistää ja lopputuloksen voi sijoittaa tiedostoon. Kokeile ja havainnoi:

```
cat sektionit.txt | grep Device > deviset.txt

cat deviset.txt

grep Section /etc/X11/xorg.conf | grep Input > inputit.txt

cat inputit.txt
```

Huomaa, että `grep` -ohjelma osaa toimia joko standardisyöttövirrasta tai lukemalla argumenttina annetun tiedoston.

Väkäsellä ohjaus siis korvaa tiedoston aiemman sisällön. Tiedoston peräänkin voidaan kyllä jatkaa; silloin pitää käyttää kahta väkästä peräkkäin >> eli kokeile vaikkapa seuraavaa; havainnoi tulokset:

```
echo "Pari tiedostoa nipussa:"  >  nippu.txt

echo -------------"  >>  nippu.txt

cat deviset.txt >> nippu.txt

echo -------------"  >>  nippu.txt

cat inputit.txt >> nippu.txt
```

9.4 Syötteiden lukeminen tiedostosta

Tiedostosta voidaan myös lukea syöte ohjelman standardisyöttövirtaan; silloin käytetään väkästä toisin päin eli <. Kokeile:

```
less < nippu.txt

wc -l < nippu.txt
```

Luettu syöte voi aloittaa putkiketjun:

```
sort < sektionit.txt | uniq | wc -l
```

Silloin se kirjoitetaan ensimmäisen ohjelman perään, ja ensimmäinen putkimerkki on tämän kokonaisuuden jälkeen.

9.5 Vielä pari shell-kikkaa

Komenna:

```
echo Oletko ‘whoami‘ vai kuka
```

Eli shell suorittaa ”backtick“ -merkkien välissä olevan komennon ensin ja sijoittaa sen antaman tulosteen komentoriviin, joka sitten suoritetaan. Tämä on usein hyödyllinen kikka skripteissä - toimii tietysti lähinnä ohjelmille, jotka tulostavat vain vähän. Komento `echo` olikin varmaan jo aiemmasta kohdasta tuttu.

Huomaa, että ”baktickien“ välissä voi olla täydellinen shell-komento, jossa voi siis olla vaikkapa putkitus:

```
echo Paikalla ‘who | cut -c-8 | sort | uniq | wc -l‘ kayttajaa
```

Näitä voi olla komennossa monta, kuten seuraavassa esimerkissä. Se ei mahdu yhdelle riville paperitulostetta, joten siinä käytetään tyypillistä rivinjatkomenettelyä kenoviivalla: voit näppäillä enterin välittömästi kenoviivan \ jälkeen kertoaksesi shellille, että aiot jatkaa vielä komentoa seuraavalle riville, tai sitten voit jättää kenoviivat pois ja laittaa koko komennon samalle rivillle. Mieluiten tietysti otat komentohistoriasta edellisen rivin ja muokkaat siitä vain alkupuolen uusiksi:

```
echo ‘whoami‘ laski ‘date‘  \
  ‘who | cut -c-8 | sort | uniq | wc -l‘ kayttajaa
```

Ja niin edelleen. Tällaisia tietovirtojen ohjauksia voi tehdä shellissä. Etköhän jo osaa yhdistellä niitä ihan hyvin.

9.6 Mistä saat täydellisen opastuksen käyttämääsi shelliin

Jos et vielä arvannut, niin paljastan suuren salaisuuden:

```
man tcsh
```

```
man bash
```

Ja niin edelleen (`ksh`, `zsh`). Shellien manuaalit ovat pitkiä (koska ohjelmissa on paljon ominaisuuksia selitettävänä) ja aluksi ehkä hiukan vaikealukuisia (koska ne on tehty hyödyttämään tehokäyttäjää tehokkaalla tavalla). Ne kertovat kaikki kikat, joita kullakin shellillä voi käyttää, sekä myös kyseisen shellin skriptiohjelmoinnin mahdollisuudet.

Shelleistä on myös määrättömän paljon hyviä johdantoja Internetissä. Niistä osa on kirjoitettu myös vasta-alkajille -- eli ei muuta kuin hakukone käyttöön, jos asia alkaa kiinnostaa enemmän.

10 Osa 6: Pakollinen palautustehtävä

10.1 Tehtävä

Käytä yksinomaan pääteyhteyttä `jalava.cc.jyu.fi`-koneeseen ja interaktiivista shelliä komento kerrallaan. Käytä tässä demossa käsiteltyjä ominaisuuksia (ml. nämä: `| ‘komento‘ < > >>` tarpeen vaatimalla tavalla). Tuota uusi tiedosto alla olevan ohjeen mukaan. Jos menee pipariksi jossain välissä, aloita koostaminen ensimmäisestä askeleesta.

1. Tee aiemmin luotuun demohakemistoosi `demo1` tiedosto nimeltä `vastaus.txt`, jossa on täsmälleen yksi rivi tekstiä; rivillä pitää lukea `Hei.`

2. Jatka samaa tiedostoa niin, että toisella rivillä lukee työskentelyhakemistosi siten kuin `pwd` -komento sen tulostaa

3. Nouda `wget`-ohjelmalla nettisivu http://www.cc.jyu.fi/~nieminen/kj07kesa/demo1nouto.php
   Ohje:

   ```
   wget http://www.cc.jyu.fi/~nieminen/kj07kesa/demo1nouto.php
   ```

   Huom: Jos tämä kohta epäonnistui aiemmin, eli demo1nouto.php ei ole oikealla tavoin noudettu, huomioi, että `wget` ei ylikirjoita työhakemistossa olevaa samannimistä tiedostoa, vaan se nimeää noudetun uuden kopion lisäämällä tiedostonimen perään `.N` missä N on juokseva numerointi. Ensimmäisellä kertaa noudettu HTML-sivu on `demo1nouto.php` ja toiseksi noudettu on `demo1nouto.php.1` ja sitten `demo2nouto.php.2` ja niin edelleen. (Joissain ohjelmissa siis on tuplavarmistuksia ylikirjoittamisen estämiseksi, vaikka perustyökaluissa useimmiten ei ole)

4. Jatka edellä tehtyä tiedostoa `vastaus.txt` siten, että aiempien rivien perässä on täsmälleen äsken noudetun `demo1nouto.php`:n sisältö

5. Tarkista, että tiedosto näyttää suurin piirtein tältä:

   ```
   Hei.
   /autohome/home3/363/nieminen/kj07kesa/demot/demo1
   <html>
     <head />
     <body>
        <p>
           Demo1. Noudettu 2007-05-25T09:16:35+03:00
           Muistithan noutaa jalava.cc.jyu.fi:sta wget -
   ohjelmalla!
        </p>
     </body>
   </html>
   ```

6. Lähetä tiedosto `mail` -ohjelmalla seuraten tarkoin alla olevaa komentosarjaa:

   ```
   mail -s "eka vastaukseni" nieminen@jyu.fi < vastaus.txt
   ```

Jos `mail`-ohjelma ei tulostanut mitään, se luultavasti teki sen mitä pyysitkin; toivottavasti pyysit oikeita asioita. Voit kokeilla lähettää omaan sähköpostiosoitteeseesi. Muista, että sähköposti on epävarma kommunikointitapa: ainoa tapa tietää viestin perillemenosta on, että saat vastauspostin. Oikein palautetusta demosta tulee jossain vaiheessa merkintä Korppiin. Jos ei sitä viikkojen kuluessa ilmaannu, voit alkaa kysellä perään.

10.2 Tarkastaminen

Teitpä demon paikan päällä tai itsenäisesti, mulla on pari kikkaa joilla pyrin varmistamaan, että olet ihan itse tehnyt täsmälleen ohjeiden mukaisen vastauksen. Muun muassa seuraavat täytyy toteutua:

- viestin muodon tulee tietyin eri tavoin vastata sitä, joka syntyy THK:n koneella shellistä annetuista komennoista, joita tehtävässä pyydettiin käyttämään.
- sähköpostin tulee olla lähetetty unixin `mail` -apuohjelmalla THK:n Linux-suorakäyttökoneelta Jalava (tai identtisesti näyttää siltä aina mail-clienteissa näkymättömiä rfc822 -headereita myöten)
- otsikon pitää olla täsmälleen ”eka vastaukseni“ (muuten automaattinen tarkastusohjelmani ei löydä sitä ollenkaan!)

Eli kaverin vastauksen kopiointi ja muokkaus ensiksi mieleen tulevin tavoin jää varsin helposti haaviin.

11 Ennakoiva osuus: tekstieditointi ja prosessit

Tämä lyhyt osuus tarjoaa katsauksen yhteen helppoon tekstieditoriohjelmaan, joka ainakin Linuxeista usein löytyy. Lisäksi saadaan yksi käyttäjän näkökulma *prosessinhallinnan* käsitteeseen, joka on keskeisimpiä, kun puhutaan nykyaikaisista käyttöjärjestelmistä. Tämä on suositeltava pikkuhomma tehdä, jos turhautumiskynnyksesi ei ole ylittynyt tähän astisissa tehtävissä.

Komenna:

```
nano
```

Käynnistit juuri tekstieditorin nimeltä GNU nano. Sitä on aika helppo käyttää, mutta se ei ole kovin monipuolinen.

Kirjoita jotain.

Näppäile `M-z` eli jos yhteysohjelmassasi toimii Alt, niin se on `Alt-Z` tai jos ei toimi, niin paina ensin `ESC` ja sitten `Z`. Pitäisi saada näkymään alalaidassa teksti ”Suspend enabled“.

Näppäile `Ctrl-Z`

Mitä tapahtui? Olet taas shellissä. Komenna:

```
ps
```

Näet, että nano-ohjelma on yhä päällä, mutta se on pysäytetty (suspended). Voit antaa komentoja shellissä ja käyttää muita ohjelmia. Pysäyttämäsi `nano` pysyy taustalla siinä tilassa, jossa sen pysäytit. Tämä siis on sitä käyttäjän näkökulmaa prosessinhallintaan, jonka teoriaa luennoilla aletaan käydä läpi toisella kurssiviikolla.

Komenna:

```
fg
```

Nyt viimeisin pysäytetty ohjelma tuli taas ”etualalle“ eli ”foreground“. Nano on aika itsensäselittävä ohjelma. Sen voi lopettaa kokonaan näppäilemällä `Ctrl-X`. Tämä kuten muut näppäinkomennot näkyvät jatkuvasti Nano-ikkunan alalaidassa muodossa `^X`.

12 Motivoiva esimerkki skripteistä

Jos haluat nähdä, millainen yksinkertaisimmillaan on skripti, ja mitä hauskaa ja hupsua sellaisella esimerkiksi voisi tehdä, voit kokeilla seuraavaa. Jos olet käyttänyt tehtäviin jo yli puolitoista tuntia, voit ehkä jättää tämän tällä kertaa ja keskittyä muihin asioihin. Paitsi jos kiinnostaa ja haluat käyttää enemmän aikaa kuin kesäope on mitoittanut. Joka tapauksessa tämä kuuluu kurssin sisältöön, ja samaan skriptiin palataan skriptidemossa.

HUOM: Varo VAARAA! Normaalisti sinun ei ikinä pidä ajaa skriptejä tai muitakaan ohjelmia, ellet ole varma siitä, mitä ne tekevät. Kesäopettaja vastaa henkilökohtaisella selkänahallaan siitä, että seuraava on tässä tapauksessa turvallista. ”Joku vaan ohjelma“ jostakin voi olla ns. *troijalainen*, eli esim. ohjelma, joka yrittää vallata käyttäjätunnuksesi itselleen. Jos ihan itse suoritat troijalaisen, se on tosi paha voi voi... Lähdekoodina julkaistuissa ohjelmissa (ja skripteissä) on se hyvä puoli, että voit tarkastaa niiden toiminnan tekstimuodossa ennen kuin käynnistät mitään. Muunmuassa tätä varten skriptejä pitää osata jonkun verran ymmärtää.

Okei. Varmistu taas, että olet käyttöjärjestelmäkurssin ensimmäistä demoa varten tekemässäsi hakemistossa. Seuraavat komennot tekevät työhakemistoon paljon kaikenlaisia uusia tiedostoja!

Hae paketti:

```
wget http://www.cc.jyu.fi/~nieminen/kj07kesa/demo1.zip
```

Avaa paketti työhakemistoon:

```
unzip demo1.zip
```

Katso mitä tiedostoja siellä nyt on:

```
ls
```

Käännä luentoesimerkkinäkin ollut C-ohjelma seuraavasti:

```
gcc -o bytes_to_bits byt*.c
```

Tässä kohtaa vielä niksi: Useimmat shellit osaavat täydentää tiedostonimiä. Jos olet varma, että hakemistossasi on täsmälleen yksi tiedosto, joka alkaa vaikkapa `byt` ja joka loppuu `.c` voit korvata keskiosan jokerimerkillä `*`. Tässä on se vaara, että olisi montakin tiedostoa, jotka vastaisivat maskia `byt*.c` -- silloin tuo maski korvautuisi jonolla, jossa on peräkkäin kaikkien

niiden tiedostojen nimet välilyönneillä erotettuna. Shell-komento suoritetaan aina vasta sen jälkeen kun on tehty tällainen potentiaalisten tiedostonimien jokerimerkkitäydennys. `*` vastaa mitä tahansa monen merkin yhdistelmää, `?` mitä tahansa tasan yhtä merkkiä.

Tällä kertaa (jos olet noudattanut ohjeita) tilanne on, että `byt*.c` on yksikäsitteisesti `bytes_to_bits.c` ja voit edellämainitulla tavalla vähentää näppäilyn määrää. Myös `-o` -tarkentimen parametri `bytes_to_bits` olisi saatavilla näppäilemällä esim. `by`, tabulaattori, BackSpace, Backspace. (Eli shellin täydennys antaisi `‘bytes_to_bits.c` josta ottaisit pois `.c`:n. Kokemuksen kautta tällaisesta tulee refleksi, ja shellin käyttö on lähes yhtä nopeaa kuin ajatus. Itse asiassa huomaan joskus odottavani ajatusten tulemista paljon kauemmin kuin niiden toteuttaminen shellissä kestää... Sen sijaan graafinen tiedostojen selaus hiirellä klikkaamalla kestää pitkään ja myös useimmiten katkaisee ajatuksenkin täysin...)

Sitten aja skripti:

```
./skriptiesim.sh
```

Sait parin sekunnin ajaksi käyttöösi Jalavan prosessorin käytännössä 100-prosenttisesti. Miltä tuntui? Kohteliasta on, että ohjelmasi eivät pitkiä aikoja kerrallaan kuluttaisi prosessoriaikaa. Hyvä tilanne olisi, että yhteiskäyttöisellä, ”kaikkiin tarkoituksiin ja kaikille“ tarkoitetulla koneella, suurin piirtein jatkuvasti prosessorilla olisi ”löysiä“ käytettävissä. Tarkkaile, mitä muut tekevät ja ole kohtelias. Se on vähän niinkuin mikä tahansa ympäristö, jossa on paljon porukkaa läsnä.

Katso tuloksia:

```
ls -lt | head
```

Ylläolevassa hakemistolistaus kertoo kaikki tiedot `-l` ja se lajitellaan viimeisen muutosajan mukaan `-t` -- siis yhdistettynä argumenttina `ls` -ohjelmalle kirjoitetaan `-lt`. Nyt ollaan kiinnostuneita vain kaikkein uusimmista tiedostoista, joten putkitetaan ohjelmalle `head` joka katkaisee tulostuksen muutaman ensimmäisen rivin jälkeen. Tiedostoista kerrotut ominaisuudet tulevat tutuksi, kun kurssilla käydään läpi Unix-tiedostojärjestelmää.

Skripti ilmeisesti teki työhakemistoon uusia kuvatiedostoja (JPG ja PNG -formaateissa). Katso kuvat läpi esim. Windows-työkaluilla ”`U:` -asemalta“ (tekstimuotoisen pääteyhteyden kautta kuvia on luultavasti vaikeahko hahmottaa).

Tutki skriptiä tekstieditorilla. Olen laittanut siihen kommenttiriveinä joitain selityksiä. Tarkoitus on demonstroida varsinkin seuraavia asioita:

- skriptin idea yksinkertaisimmillaan: suoritetaan komennot peräkkäin aina kun edellinen on valmis (”eräajo“)
- komentoriviltä tekstimuodossa käytettävän ohjelman tulosteiden ei tarvitse olla tekstimuotoisia, vaan ne voivat olla esim. grafiikkaa tai multimediaa

Seuraavat yksityiskohdat löytyvät skriptistä sovellettuna:

- unix-tiedostojärjestelmässä on laitehakemisto `/dev/`, jossa on fyysisten laitteiden lisäksi virtuaalisia laitteita.
- ohjelman tulosteen putkittaminen prosessilta toiselle syötteeksi `|`
- tiedoston sisällön ohjaaminen syötteeksi prosessille `<`
- ohjelman tulosteen ohjaaminen tiedostoksi (overwrite) `>`
- muutamia apuohjelmia (isoja ja pieniä; perinteisiä ja erikoisempia)
- pitkiä argumenttilistoja ja komennon jakaminen usealle tekstiriville kenoviivasyntaksilla eli \ rivin lopussa

Jos kiinnostaa, voit tehdä itsellesi oman Käyttöjärjestelmät-logon seuraavin tavoin:

- Etsi skriptistä muutettavia kohtia, muuta ne, ja aja uudelleen.
- Ensimmäisen ajon jälkeen voit toiminnan nopeuttamiseksi kommentoida pois alkupuolen, joka ottaa satunnaiset bitit taustakuvaksi. Nehän on jo tallennettu tiedostoon.
- Helppoja muutettavia ovat esim. logon teksti, Plasmaefektin värimaailma, taustakuvan tummuus.
- Taustakuvaksi voi ottaa minkä tahansa (mutta skripti on suunniteltu kuvalle, joka on kaksi kertaa niin leveä kuin korkea).

Oikeasti skripti olisi tietysti hienostuneempi: sille pitäisi voida esim. antaa taustakuva komentoriviltä argumenttina, sen pitäisi tehdä itselleen väliaikaishakemisto `/tmp/` -hakemiston alle ja huolehtia, että varmasti kyseinen hakemisto tietoineen poistetaan suorituksen lopuksi riippumatta siitä, onnistuivatko operaatiot vai tuliko niissä joku virhe. Eli skriptinkin tekemiseen liittyisi Ohjelmointi 1:ltä tuttuja huomioitavia asioita. Tuo, mitä käpistelet, on diiba-daaba-esimerkki, jonka tarkoitus kiteytyy tähän:

- olet nähnyt skriptin
- ymmärrät eräajon idean
- ymmärrät että skriptiä voi melko helposti muokata hieman
- unix-skriptit usein käyttävät useita apuohjelmia -- eli vaikka ne pyörivät shellissä, joka on välittömästi käyttöjärjestelmän rajapinnan päällä, ne voivat hyödyntää välillisesti laajoja apukirjastoja, jotka käyttävät allaan jo monia virtuaalikonehierarkian kerroksia.

Puristinen skripti käyttäisi vain standardeja kaikkialta löytyviä apuohjelmia. Eli jotakuinkin niitä, jotka GNU-järjestelmissä esiteltäisiin seuraavan komennon antamalla sivulla:

```
info coreutils
```

Tyypillisesti shell-skriptillä ei tietenkään tehdä kuvia, vaan esimerkiksi ohjelmistoasennuksen konfigurointia, varmuuskopiointia, järjestelmädiagnostiikkaa tai vastaavaa. Ajoitetut tehtävät ovat tyypillinen skriptien käyttötarkoitus, eli joku skripti voidaan ajaa vaikka joka päivä klo 04 aamulla; se voi tehdä ylläpidollisia asioita, jotka on tehtävä työajan ulkopuolella -- siis parasta tehdä ne automaattisesti ja toivoa, että skripti aina toimii vaikkei kukaan valvo sitä vierestä.

13 Ihan lopuksi

Kun olet saanut asiasi hoidettua etäyhteydellä, voit komentaa shell-ohjelman lopettamaan itsensä komennolla:

```
exit
```

Shellin suoritus loppuu, ja palvelin havaitsee tämän ja sulkee pääteyhteyden. Olet poistunut järjestelmästä; tervemenoa takaisin milloin vain :).