1.1 Tekstinprosessointiohjelmien yleinen toimintatapa (1)

- Useimmat tekstinprosessointiohjelmat (`cat`, `grep`, ...) toimivat kahdella erilaisella tavalla.
- Jos niille annetaan tiedostojen nimiä argumenteiksi, ne lukevat syötteensä kyseisistä tiedostoista, esim. `grep misu juttu.txt`.
- Sen sijaan jos niille ei kerrota yhtäkään tiedostonimeä, ne lukevat syötteensä *vakiosyötteestä* (standard input). Komentotulkki määrää, mistä vakiosyöte varsinaisesti tulee.
- Oletuksena vakiosyöte tulee suoraan käyttäjältä. Tästä syystä esimerkiksi komento `grep foo` hakee foo-sanoja käyttäjän kirjoituksesta. Syötteen loppumisesta voi ilmoittaa painamalla `C-d`.

1.2 Virroista (1)

- Vakiosyöte on yksi esimerkki *vakiovirroista*. Virrat ovat ikään kuin kanavia, joita pitkin kulkee tekstiä. Muita vakiovirtoja ovat *vakiotuloste* ja *vakiovirhe*.
- Vakiovirtojen idea on siinä, että ohjelma voi sokkona lukea syötettä, tuottaa tulostetta ja kirjoittaa virheilmoituksia tietämättä, mistä syöte oikeasti tulee tai mihin tuloste / virheilmoitukset oikeasti menevät.
- Oletuksena vakiosyöte tulee yksinkertaisesti käyttäjältä ja vakiotuloste ja -virhe menevät käyttäjän näytölle luettaviksi. Esimerkiksi `cat -n` (ilman muita argumentteja) lukee käyttäjältä syötettä linja kerrallaan ja näyttää linjat numeroituina.

- Komentotulkin ohjaustoiminnoilla esim. syötteen voi yhtä hyvin antaa ohjelmalle tiedostosta tai vaikkapa toisen ohjelman tulosteesta. Meillä on jo käsiteltykin tulosteen ohjaus tiedostoon (>- ja >>-rakenteet).
- Tämä mahdollisuus ohjailla ohjelmien syötettä, tulostetta ja virheilmoituksia on tärkein ohjelmien yhdistelytapa Unixissa. Erityisen tärkeä on rakenne, jolla annetaan yhden ohjelman tuloste toisen syötteeksi. Tämän rakenteen nimi on putki ja sitä merkitään merkillä |.
- Kaikki seuraavien kalvojen ohjelmat osaavat lukea myös vakiosyötettä, jos niille ei anneta yhtään tiedostoa argumentiksi.

1.3 `cat`, `wc` — uudestaan

cat `[ -n ] [ -s ]` *`tied1 tied2 ...`* (CATenate)
Yhdistää peräkkäin argumentteina olevat tiedostot. Valitsimella `-n` (Numerate) numeroi rivit. Valitsimella `-s` (Squeeze) korvaa peräkkäiset tyhjät rivit yhdellä.

wc `[ -l ] [ -w ] [ -c ] [ -L ]` *`tied1 tied2 ...`* (Word Count)
Laskee rivit, sanat ja merkit argumenttitiedostoista ja näyttää tuloksen. Valitsimilla `-l` (Lines), `-w` (Words) ja `-c` (Characters) kertoo *vain* linjojen, sanojen tai merkkien määrän (vastaavasti). Valitsimella `-L` (Line) kertoo pisimmän rivin pituuden.

1.4 `grep` — uudestaan

grep `[ -C` *n* `] [ -v ] [ -x ] [ -i ] [ -o ]` *sana tied1 tied2 ...*
(Global RegExp Print)
Etsii sanaa argumenttitiedostoista. Oletuksena tulostaa linjat, joilla kyseinen sana löytyy. Valitsimella `-C` (Context) tulostaa myös löydöslinjojen ympäriltä pari linjaa. Valitsimella `-v` (?) tulostaa kaikki linjat, joista sanaa *ei* löydy. Valitsimella `-x` (eXact) tulostaa vain ne linjat, joilla on haettu sana eikä mitään muuta. Valitsimella `-i` (Insensitive) ei kiinnitä huomiota suurten ja pienten kirjainten eroon. Valitsimella `-o` (Only) tulostaa vain sen kohdan, joka täsmäsi (ei koko linjaa).

grep `-r` *sana hakemisto* (Global RegExp Print / Recursive)
Etsii sanaa *rekursiivisesti* kaikista hakemiston sisältämistä tiedostoista.

1.5 Uusia tekstinprosessointikomentoja

head `[ -n ]` *tied1 tied2 ...* (HEAD of file)

tail `[ -n ]` *tied1 tied2 ...* (TAIL of file)

Näyttää n ensimmäistä (`head`) tai viimeistä (`tail`) riviä tiedostoista. Oletus 10 riviä.

sort `[ -n ] [ -r ] [ -k n ]` *tied1 tied2 ...* (SORT)

Järjestää argumenttitiedostojen rivit (oletuksena aakkosjärjestykseen). Valitsin `-n` (Numeric) tuottaa numeerisen järjestyksen; valitsin `-r` (Reverse) kääntää järjestyksen päinvastaiseksi; valitsimella `-k` (Key) voi valita, monennenko sanan mukaan järjestetään.

uniq `[ -c ] [ -d ]` *`tied`* (UNIQue lines)
Näyttää argumenttitiedoston siten, että peräkkäiset samat linjat on poistettu. Valitsimella `-c` (Count) kertoo lisäksi, kuinka monta kertaa mikin linja oli syötteessä. Valitsimella `-d` (Duplicates) kertoo *vain* ne linjat, joita oli useampi kappale.

cut `[ -d` *`merkki`* `] -f` *`n tied1 tied2 ...`* (CUT fields)
Näyttää argumenttitiedostot siten, että jokaisesta linjasta näytetään vain *n*:s kenttä (Field). Kenttäerottimena käytetään *merkki*:a (Delimiter), joka on oletuksena sarkain (tabulaattori).

diff `[ -u ]` *`tied1 tied2`* (DIFFerence)
Näyttää erot *tied1*- ja *tied2*-tiedostojen välillä.

Date: 2004/01/03 00:39:01

1.6 `tr` — kielitieteilijän perustyökalu

Tämä on yksinkertaisuudessaan hämmentävän monipuolinen ohjelma. `tr` osaa lukea syötettä vain vakiosyötteestä, sille ei voi antaa argumentiksi tiedostonimiä.

`tr` *`mistä mihin`* (TRanslitterate)
Muuntaa syötteen *mistä*-merkit vastaaviksi *mihin*-merkeiksi. Esimerkiksi `tr ab uo` muuttaa kaikki a:t u:ksi ja b:t o:ksi. Merkkijoukoissa voi olla myös väliviivalla merkittyjä merkkivälejä, esim. `tr A-M a-m` muuntaa kaikki isot kirjaimet A:n ja M:n väliltä pieniksi.

`tr` `-d [ -c ]` *`merkkejä`* (TRanslitterate / Delete)
Poistaa syötteestä annetut merkit. Valitsimella `-c` (Complement) poistaa kaikki merkit paitsi annetut merkit. Myös tämä ymmärtää merkkivälejä.

tr -s *merkkejä* (TRanslitterate / Squeeze)
Jos syötteestä löytyy useampia peräkkäisiä kappaleita jotain merkkiä, joka kuuluu annettuun merkkijoukkoon, ne korvataan yhdellä.

`tr`:n vipusia voi myös yhdistää. Esimerkiksi `tr -sc A-Za-z ?` korvaa kaikki yhden tai useamman merkin ei-kirjainjaksot kysymysmerkillä.

Tärkeimpiä `tr`:n käytöistä lienee sanojen paneminen omille riveilleen. Se onnistuu komennolla

```
tr " " \\012
```

joka siis muuttaa välilyönnit rivinvaihdoiksi. Tähän hipsujen käyttöön tutustutaan tuonnempana.

1.7 Tositoimiin: prosessointikomennot

- Ota jokin isohko tiedosto raakamateriaaliksesi. Jos sinulla ei ole sellaista, voit esim. hakea jonkin WWW-sivun komennolla `wget`, jos se löytyy järjestelmästä, tai kirjoittaa `emacs`-ohjelmalla jonkin tekstin.
- Etsi tiedostosta erilaisia sanoja sisältäviä linjoja `grep`-komennolla. Esim. `grep -i and tiedosto.txt`. Kokeile `grep`-komennon eri valitsimien vaikutusta lopputulokseen.
- Kokeile erilaisia merkkivaihdoksia ja -poistoja `tr`-komennolla. Esim. `tr uio üíå <tiedosto.txt`.
- Katso, mitä tiedostossa `/etc/passwd` on. Vertaa komennon `cut -d: -f1 /etc/passwd` tulokseen. Kokeile hakea muitakin kenttiä.

- Näytä `/etc/passwd` käyttäjätunnusten mukaan aakkostettuna komennolla `sort /etc/passwd` ja nimien mukaan aakkostettuna komennolla `sort -t: -k5 /etc/passwd`.
- Kokeile `sort`-komennon eri valitsimien vaikutusta lopputulokseen.
- Anna komento `grep 'a.*i'` ja kirjoittele erilaisia sanoja kokeillaksesi, mihin `grep` tarttuu.
- Anna pelkkä komento `cat -n` ja kirjoittele kaikenlaista katsoaksesi, mitä se tekee.
- Tee `cut`-komennon eri valitsimilla raakamateriaalistasi erilaisia listittyjä versioita.

2.1 Tekstinprosessointiohjelmien yleinen toimintatapa (2)

- Erityisesti tekstinprosessointiohjelmat, mutta usein Unixin muutkin ohjelmat, on suunniteltu toimimaan tarvittaessa *filttereinä*: ne ottavat vakiosyötteestä jotain, tekevät sille jonkin muunnoksen ja antavat tuloksen vakiotulosteeseen.
- Filttereistä pystyy rakentelemaan yhdistelemällä hyvinkin monimutkaisia kokonaisuuksia.
- Useimmat tekstinprosessointiohjelmat suostuvat lukemaan syötettä sekä joistain tiedostoista että vakiosyötteestä. Tällöin vakiosyötettä merkitään tiedostonimellä `-`. Esimerkiksi komento `diff makkara -` vertailee eroja `makkara`n ja vakiosyötteen välillä.

2.2 Virroista (2)

- Vakiosyötteen ja -tulosteen pointti on siis se, että ne sallivat ohjelman toimia filtterinä, muunnoksena tiedolle, josta ei tiedetä, mistä se tulee, eikä mihin se menee.
- Vakiovirheen pointti on siinä, etteivät virheilmoitukset sotkeutuisi muuhun tulosteeseen, vaan ne voivat tulla suoraan käyttäjän nähtäviksi tai ohjata esim. tiedostoon tarkasteltaviksi.
- Vakiovirheen voi ohjata tiedostoon rakenteella `2> tiedosto`, joka liitetään komennon perään samoin kuin >-ohjaus. Vakiosyötteen voi ohjata tiedostosta rakenteella `< tiedosto`.

2.3 Putket

- Putkilla filtterikomennoista voi rakennella pitkiä "tuotantolinjoja", joissa jokainen komento tekee oman hommansa ja lähettää tuloksensa taas eteenpäin seuraavalle komennolle.
- Putkia komentojen välille muodostetaan kirjoittamalla kaksi komentoa erotettuna putkimerkillä (`|`).
- Esim. `grep jukkeli foo.txt | cat -n` etsii `foo.txt`-tiedostosta jukkeli-sanaa ja lähettää tuloksen `cat`:lle numeroitavaksi.
- Putkimerkkiä kohden komentotulkki tekee siis kolme asiaa: perustaa putken, ohjaa edellisen käskyn tulosteen putkeen, ja ohjaa seuraavan käskyn syötteen putkesta.

3.1 Putkikomentojen lukeminen

⋆ Esimerkki:

```
cat -n a.html | grep href | head -30 | tail -1
```

Tämä komento näyttää `a.html`-tiedoston 30. linkin sekä rivin numeron, jolla kyseinen linkki sijaitsee. Mutta miten se toimii?

- `cat -n` lisää rivinumerot linjojen eteen.
- `grep href` etsii rivit, joilla esiintyy sana href, ts. rivit joilla on linkkejä. (Jos rivillä on useampi kuin kaksi linkkiä, ohjelmamme ei toimi oikein. Linkit pitää oikeasti rivittää `tr`:lla ensin.)
- `head -30` jättää tästä jäljelle vain 30 ensimmäistä riviä.
- `tail -1` poimii näistä vain viimeisen rivin, ts. `grep`:n tulosteen 30. rivin.

Putkikomentojen ymmärtämisessä on usein hyödyllistä nähdä, millaista tekstiä kulkee minkäkin kahden komennon välillä. Tämä onnistuu siten, että kirjoittaa putkilinjan komento kerrallaan ja katsoa aina, millaista tulostetta syntyy missäkin vaiheessa.

Sen sijaan halutun putkilinjan muodostaminen onkin vaikeampi homma: se edellyttää avointa ajattelua, erilaisten ratkaisumahdollisuuksien etsimistä. Yksi hyväksi havaittu tapa tehdä tätä on lisäillä putkilinjaan yksi komento kerrallaan sen mukaan, mikä "näyttää" vievän meitä lähemmäs ratkaisua. Tämä tietysti edellyttää intuitiota...

Joitain nyrkkisääntöjä on: kun halutaan valita tiettyjä linjoja (*selektio*), `grep` on yleensä oikea vastaus; kun halutaan valita linjoista tietty osa (*projektio*), `cut` on yleensä oikea vastaus; ja kun halutaan muuttaa tiedon rakennetta, `tr` on yleensä oikea vastaus.

Date: 2004/01/03 00:39:01

3.2 Tyypillisiä linjan loppuun liitettäviä komentoja

On monia komentoja, jotka on selkeästi suunniteltu putkilinjan lopettimiksi. Näistä ylivoimaisesti yleisin on `less`.

- Lisää komennon loppuun `| less` jos haluat selailla tulosta edestakaisin.
- `| lpr`, `| mpage` tai `| a2ps` on kätevä tapa tulostaa lopputulos.
- `| mail tyyppi@jossain` lähettää tuloksen sähköpostiviestinä.
- `| head` on näppärä tapa katsoa vain vähän, minkänäköistä tulosta putkilinja tuottaa.

3.3 Välihuomatutus: lainaaminen

- Kuten huomattu, komentolinjat koostuvat jonosta sanoja.
- Joskus kuitenkin "sanan" pitää sisältää välilyöntejä, jos esimerkiksi nimeää tiedoston, jonka nimessä on välilyönti. Samoin erilaiset maagiset merkit, kuten `<`, `>`, `|` ja `*`, halutaan joskus antaa komennolle sellaisinaan argumentiksi.
- Sekalaisesta merkkipötköstä voi tehdä yhden "sanan" panemalla sen *hipsuihin* (`"` tai `'`).
- Esimerkiksi merkkijonoa `hui sentään` voi etsiä komennolla `grep 'hui sentään' tiedosto`.

3.4 Putkiesimerkkejä (1)

```
⋆ ls | wc -l
```

Laskee, kuinka monta tiedostoa on oletushakemistossa. (Kun `ls` tulostaa muualle kuin näytölle, se tulostaa yhden tiedoston per rivi.)

```
⋆ history | grep '|' | less
```

Antaa selailla, mitä putkikomentoja on antanut.

```
⋆ cut -d: -f5 /etc/passwd | sort
```

Luettelee koneen käyttäjät aakkosjärjestyksessä.

```
⋆ echo testausta | mail -s koe hj@iki.fi
```

Lähettää `hj@iki.fi`:lle viestin otsikolla "koe" joka sisältää vain sanan "testausta".

3.5 Putkiesimerkkejä (2)

```
tr -s " " \\012 <aine.txt | sort | uniq | fmt | less
```

Näyttää aakkosellisen listan tiedoston aine.txt sanoista.

- `tr` muuntaa välilyönnit rivinvaihdoiksi ja korvaa samalla useammat peräkkäiset rivinvaihdot yhdellä (jottei tule tyhjiä rivejä).
- `sort` järjestää sanat aakkosjärjestykseen ja siirtää ne samalla peräkkäin `uniq`:a varten.
- `uniq` jättää vain eroavat sanat jäljelle eli sanan toistot poistetaan.
- `fmt` uudelleenrivittää syötteen enintään 74-merkkisiksi riveiksi.
- `less` sallii tarkastella tulosta mukavasti.

3.6 Putkiesimerkkejä (3)

```
history | grep " cd " | sort -k 3 | mpage -8 -Plp
```

Tulostaa käytetyt `cd`-käskyt järjestettynä sen mukaan, mihin tiedostoon niillä on menty, 8 sivua yhdellä paperilla.

- `history` antaa käytetyt käskyt.
- `grep` valitsee niistä vain ne, joissa on sana `cd`.
- `sort` järjestelee ne kolmannen sanan mukaan (1. sana on historianumero, toinen cd-komento).
- `mpage` tekee niistä postscriptia (tiedostomuoto, jota tulostimet ymmärtävät) ja lähettää sen tulostimelle `lp`.

3.7 Tositoimiin: putket

- Näissäkin tarvitset jonkin raakamateriaalitiedoston...
- Rivitä tiedoston sanat `tr`-komennolla:

  ```
  tr " " \\012 <tiedosto.txt
  ```

- Lisää tämän perään erilaisia putkeksi erilaisia `tr`-komentoja, joilla saat poistetuksi sanoista suurien ja pienten kirjainten erot, välimerkit, ja niin edelleen.
- Kokeile myös tutkia sanojen vokaalirakennetta suodattamalla niistä konsonantit pois: `... | tr -dc aeiouyääö`
- Lisää tähän vielä perään `... | sort | uniq -c` katsoaksesi, mitkä ovat yleisiä vokaalirakenteita.
- Kokeile putkiesimerkeissä esitettyjä komentoja siten, että lisäät putkilinjaan elementin kerrallaan ja katselet tulosta joka välissä.