GREP

Sami Peltonen

HAMK 2021

Sisällysluettelo

Sivu

1. Historiaa 3
2. Alkutoimenpiteet 4
3. Grep-peruskäyttö 7
4. Grepin edistyneempi käyttö 11
5. Komennot 20
6. Erikoismerkit 21
7. Lähteet 22
8. Tiedot.txt sisältö 23

Historiaa

Bell Labs (yrityskauppojen johdosta tunnetaan myös nimellä Nokia Bell Labs) on tutkimus- ja kehitysorganisaatio, jonka päämaja sijaitsee New Jerseyssä, Yhdysvalloissa. Alun perin puhelinteknologiaan keskittyneen Bell Labsin saavutuksiin voidaan lukea muun muassa transistorin, laserteknologian ja radioastronomian kehitystyö.

Aivan 1960 -luvun lopussa Bell Labsin työntekijät Ken Thompson ja Dennis Ritchie aloittivat Unix -käyttöjärjestelmän kehittämisen. Unix oli suhteellisen helppokäyttöinen ja kevyt ympäristö, jonka mukana tuli myös hyödyllisiä ohjelmia. Yksi niistä oli tekstinkäsittelyohjelma nimeltään ed.

Ed -ohjelmaa hallitaan yksinkertaisilla käskyillä. Käsky **1p** tulostaa tiedoston ensimmäisen rivin, **1d** poistaa sen. Käsky **1,$p** tulostaa tiedoston kaikki rivit ensimmäisestä viimeiseen, käsky **1,$d** poistaa ne.

Ed kykenee myös toimimaan säännöllisten lausekkeiden avulla. Nämä ovat käyttäjän määrittelemiä merkkijonoja, joita voidaan etsiä, muuttaa ja poistaa tiedostosta. Esimerkiksi käsky **g /koira/p** etsii tiedostostakaikkirivit, jotka sisältävät merkkijonon ”koira” ja tulostaa ne.

Ed:n toimintaa rajoitti sen kyvyttömyys käsitellä suuria tiedostokokoja. Vuonna 1974 Ken Thompson kehitti edistyneemmän komentoriviohjelman nimeltään grep, eli **g/re/p (**search **g**lobally for a **r**egular **e**xpression and **p**rint). Grep peri edeltäjältään helppokäyttöisen käskyrakenteen, mutta on lähes viisikymmentävuotisen historiansa aikana saanut yhä edistyneempiä piirteitä.

Linus Torvaldsin kehittämä Linux -käyttöjärjestelmä on rakennettu Unixin perusteiden päälle. Siksi Linuxin työkalupakista löytyy grep. Tämän oppaan tavoitteena on opettaa aloittelevalle Linux -käyttäjälle grepin käyttöä, alkaen aivan perusteista ja siirtyen askel askeleelta kohti kehittyneempiä toimintoja.

Alkutoimenpiteet

Tarvitsemme harjoituksia varten uuden kansion ja sen sisälle kaksi tekstitiedostoa. Löydät tekstitiedoston sisällön githubista: https://github.com/sivakkasutari/GREP.git tai tämän oppaan lopusta.

Linuxin työpöydältä voimme avata terminaalin painamalla **Ctrl + Alt + T**.

Luomme aivan aluksi uuden kansion nimeltä **grep** seuraavalla käskyllä:

**sudo mkdir grep**

Siirrymme kansion sisälle:

**cd grep**

Koska olemme jo hieman tutustuneet ed -tekstinkäsittelyohjelmaan, käytämme sitä luodaksemme kaksi tekstitiedostoa, joita harjoituksissa käytämme. Aivan aluksi avaamme ed-ohjelman:

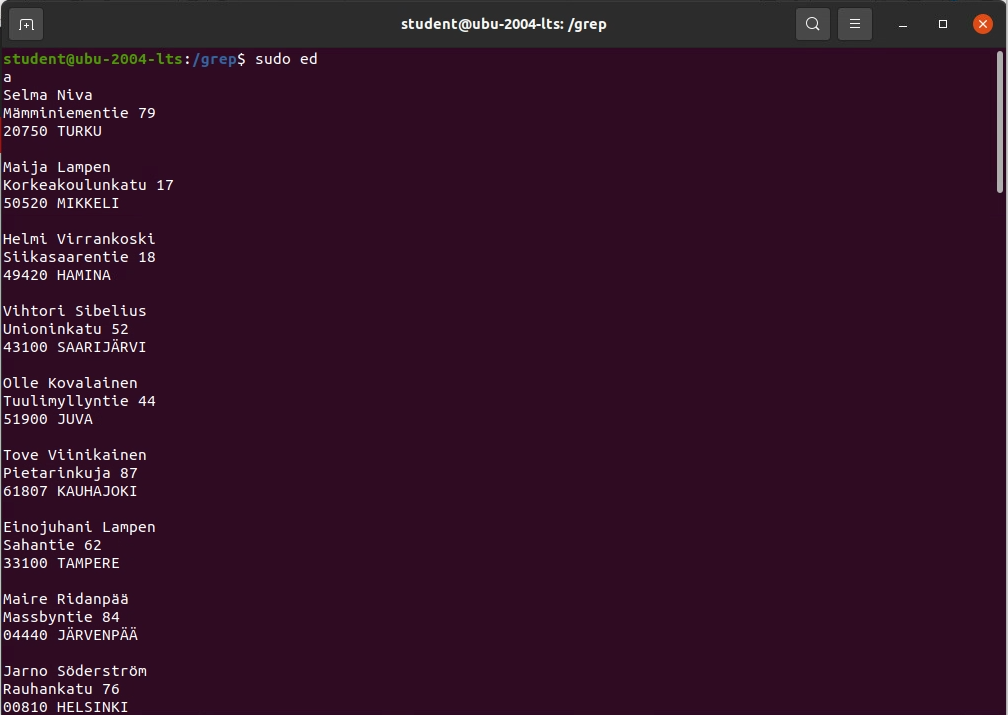
**sudo ed**

Laitamme ed:n vastaanottotilaan kirjoittamalla

**a**

ja painamalla **Enter**.

Tämän oppaan lopusta löydämme sisällön, jonka voimme maalata ja kopioida hiirellä. Liitetään nyt kopioitu teksti ed -editoriin. Näkymän pitäisi olla seuraavanlainen:



Tekstiä kopioitu ed -editoriin.

Lopetamme tekstinsyötön kirjoittamalla

**.**

ja painamalla **Enter**.

Tietomme on nyt tallennettu ed:n muistiin. Luomme tekstitiedoston, johon tallennamme tiedot:

**w tiedot.txt**

Kuitataan komento painamalla **Enter**.

Kirjoitetaan vielä samat tiedot toiseen tekstitiedostoon:

**w tiedot2.txt**

Kuitataan painamalla **Enter.**

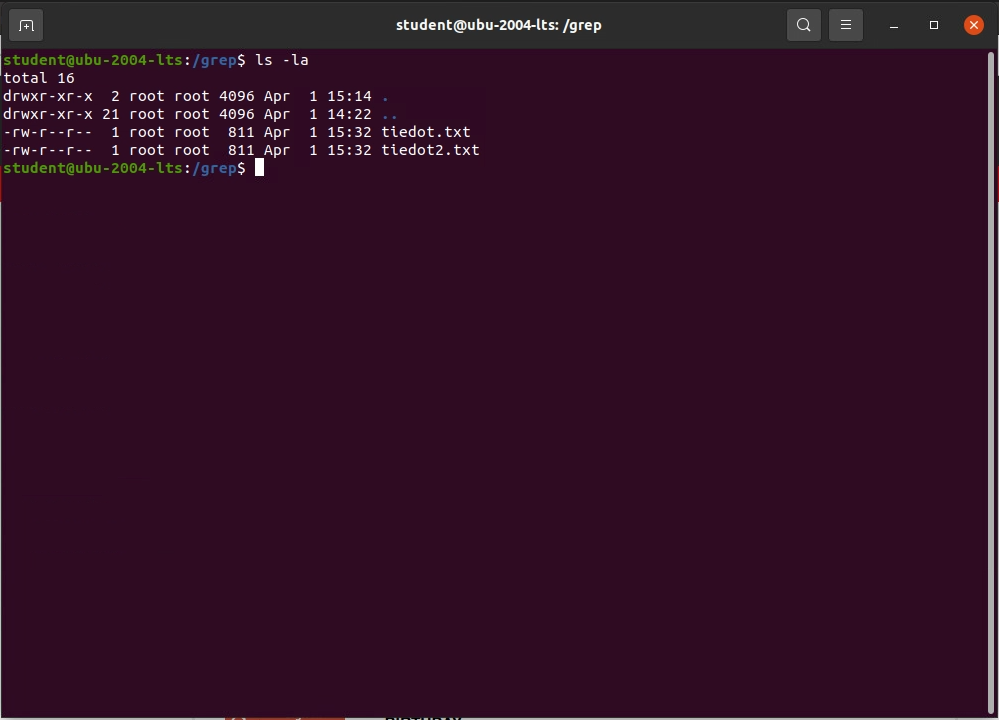
Voimme lopettaa ed:n käytön:

**q**

ja painamalla **Enter**.

Tarkistetaan, että grep -kansiosta löytyy luomamme kaksi tekstitiedostoa:

**ls -la**

****

Tiedot.txt ja tiedot2.txt löytyvät grep -kansiosta**.**

Voimme tyhjentää terminaalinäkymän:

**clear**

Grep -peruskäyttö

Teemme ensimmäisen haun:

**grep ”Risto” tiedot.txt**

Saamme yhden hakutuloksen:

**Risto Haapoja**

Tiedot.txt -tiedostosta löytyy siis yksi maininta nimestä ”Risto” ja grep tulostaa sen rivin, jolta tämä maininta löytyy. Entä jos kirjoitamme:

**grep ”risto” tiedot.txt**

Tuloksia ei näy. Tämä johtuu siitä, että grep -haut ovat oletusarvoisesti merkkikokoriippuvaisia. Jos haluamme, että grep palauttaa kaikki merkkijonoa vastaavat tulokset merkkikoosta välittämättä, voimme lisätä hakuun komennon **-i**:

**grep -i ”risto” tiedot.txt**

Tämä komento on varsin käytännöllinen, sillä on täysin mahdollista, että tiedoston luoja on tehnyt virheitä merkkikokojen suhteen.

Etsitään nyt uutta merkkijonoa:

**grep -i ”juhani” tiedot.txt**

Saamme kaksi tulosta. Grep siis etsii kaikki maininnat merkkijonosta, vaikka kyseessä olisi osa toista, pidempää merkkijonoa. Jos haluamme löytää merkkijonoa tarkalleen vastaavan tuloksen, voimme käyttää komentoa **-w**:

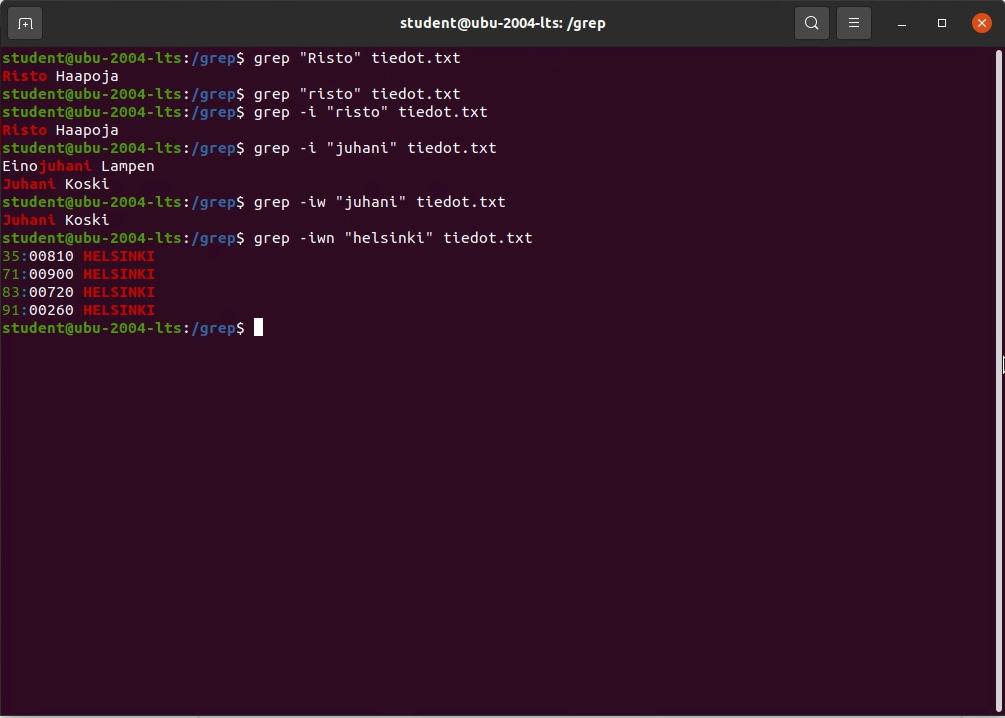
**grep -iw ”juhani” tiedot.txt**

Nyt saamme vain yhden tuloksen. Kuten näemme, uuden komennon voi lisätä edellisen perään ilman väliviivaa.

Etsitään seuraavaksi kaikki rivit, joissa mainitaan Suomen pääkaupunki. Lisäämme hakuun uuden komennon **-n**:

**grep -iwn ”helsinki” tiedot.txt**

Saamme neljä tulosta. **-n** -komennon ansiosta näemme myös rivinumerot.



Kaikki yllä mainitut komennot suoritettu.

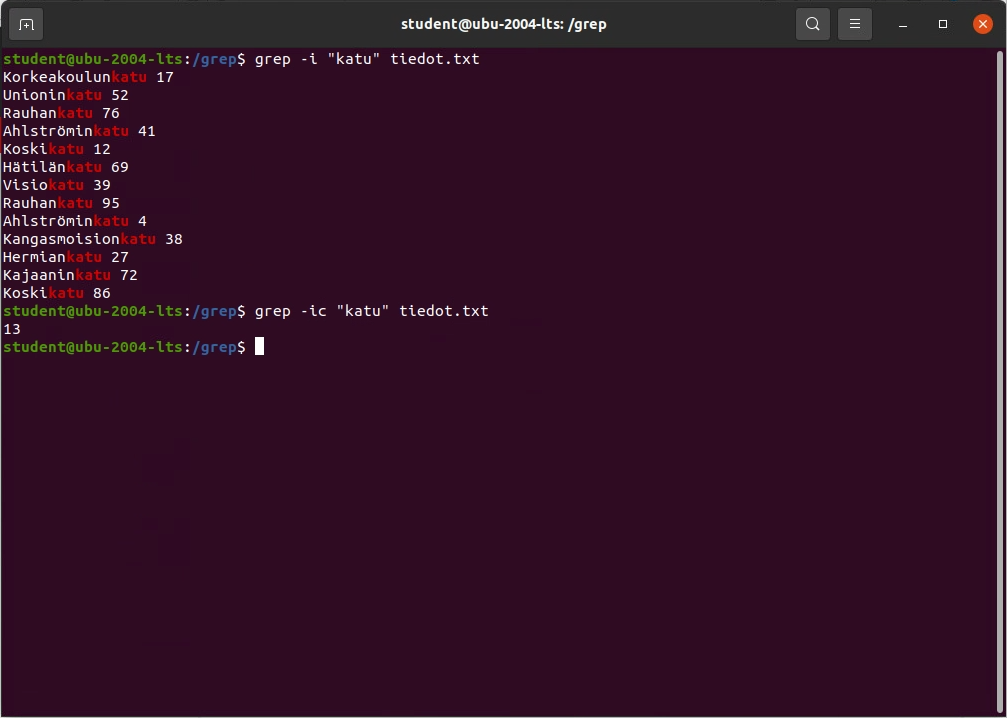
Vaikka emme olisikaan vielä nähneet tiedot.txt -tiedoston koko sisältöä, suorittamiemme hakujen perusteella voimme päätellä, että tiedosto luultavasti sisältää ihmisten nimiä ja osoitteita. Etsitään seuraavaksi sellaisia rivejä, jotka sisältävät merkkijonon ”katu”:

**grep -i ”katu” tiedot.txt**

Tuloksia on jo useampia. Käytetään **-c** -komentoa laskemaan tuloksien lukumäärä:

**grep -ic ”katu” tiedot.txt**

Grep kertoo meille, että tiedostosta löytyy kolmetoista riviä, jotka sisältävät merkkijonon ”katu”.



Ensin on etsitty ”katu” -merkkijonolla ja sen jälkeen käsketty laskemaan hakutulosten lukumäärä.

Entä jos haluamme löytää ne rivit, jotka eivät sisällä haluttua merkkijonoa? Kokeillaan seuraavaa:

**grep -iv ”katu” tiedot.txt**

Grep tulostaa tiedostosta kaikki rivit lukuun ottamatta niitä, jotka sisältävät merkkijonon ”katu”.

Voimme etsiä tietoa myös merkkijonon ympäriltä. Käytämme uutena komentona **-A**:ta, ja lisäämme sen perään numeron:

**grep -iA 1 ”katu” tiedot.txt**

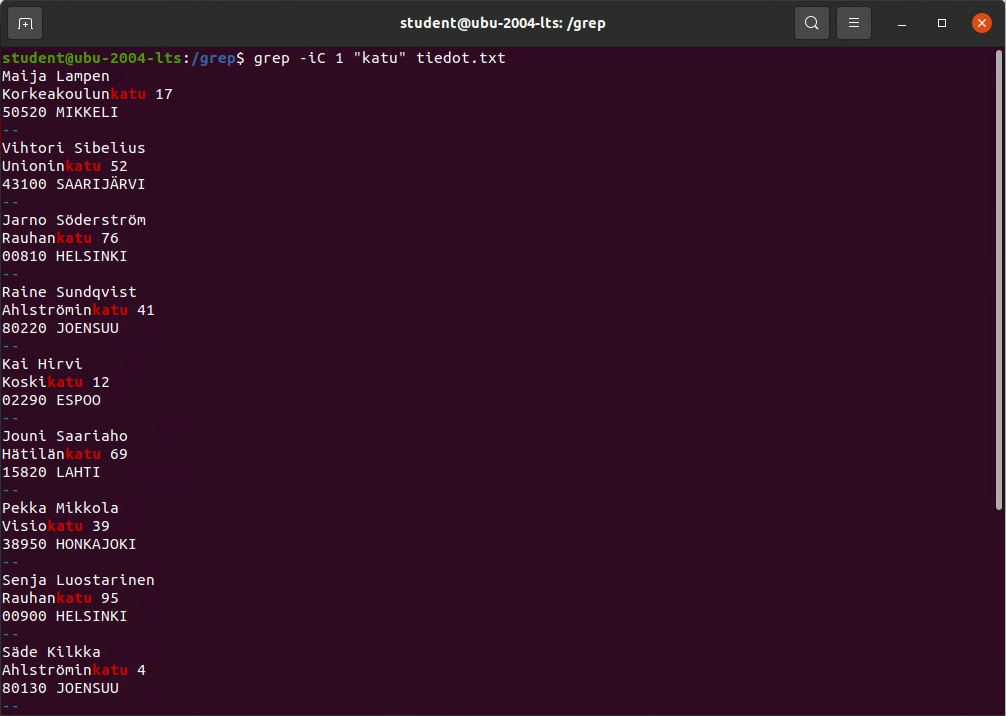
Saamme kolmetoista hakutulosta, jotka on erotettu toistaan katkoviivoilla. Grep hakee siis rivin, joka sisältää merkkijonon ”katu”, mutta lisäksi sen *jälkeen* tulevan rivin. Entä jos kirjoitamme:

**grep -iB 1 ”katu” tiedot.txt**

Näemme saman ”katu” -rivin ja lisäksi sitä *edeltävän* rivin.

Yhdistetään kaksi edellistä komentoa:

**grep -iC 1 ”katu” tiedot.txt**



Palautetaan yksi rivi merkkijonon ”katu” sisältävän rivin ympäriltä.

Grepin edistyneempi käyttö

Olemme käyttäneet grep -hauissa merkkijonoja, kuten ”katu” ja ”helsinki”. Näitä merkkijonoja kutsutaan säännöllisiksi lausekkeiksi (regular expressions). Haemme siis määrättyyn kaavaan perustuvia merkkijonoja, jotka voivat toistua tiedoston sisällä. Erikoismerkkejä käyttämällä säännöllisten lausekkeiden avulla voidaan suorittaa hyvin edistyneitä hakuja. Tutustumme niihin tässä kappaleessa.

Kysymme aluksi grepiltä, mitä tiedot.txt pitää sisällään:

**grep ”.” tiedot.txt**

Olemme käskeneet grepin etsiä ja palauttaa tiedostosta kaikki merkit, ovat ne sitten kirjaimia, numeroita tai erikoismerkkejä. Grep käy tiedoston läpi rivi riviltä, merkki merkiltä alkaen ensimmäisestä ja päätyen viimeiseen. Huomaamme, että merkki **.** (piste)edustaa kaikkia merkkejä**.** Jos haluaisimme hakea pistettä itseään, pitäisi meidän esittää asia näin:

**grep ”\.” tiedot.txt**

Seuraavaksi kirjoitamme:

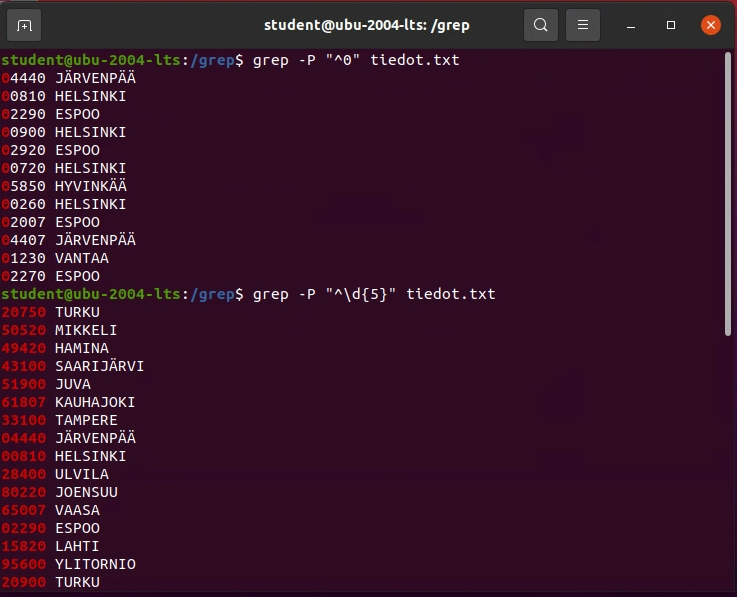
**grep -P ”^0” tiedot.txt**

Erikoismerkki **^** edustaa rivin alkua. Toisin sanoen, käskemme grepiä hakemaan rivejä, joissa rivin alkua seuraa välittömästi numero nolla. Komento **-P** ei ole tässä aivan välttämätön, mutta otamme sen nyt käyttöön, sillä joudumme vääjäämättä siihen turvautumaan, kun hakujen monimutkaisuus kasvaa. **-P** kertoo grepille, että haluamme käyttää Perl -yhteensopivia säännöllisiä lausekkeita. Näin grep osaa käyttää laajempaa ”sanavarastoa”.

Seuraavaksi haluamme nähdä kaikki tiedostosta löytyvät rivit, jotka alkavat viisinumeroisella merkkijonolla:

**grep -P ”^\d{5}” tiedot.txt**

Ohjaamme grepin etsimään rivin alkua **^** -merkillä. Komento **\d{5}** kertoo, ettärivin alun jälkeen etsimme numeroita ( **\d** ) ja että niitä on oltava viisi kappaletta ( **{5}** ). Tulokseksi saamme postinumeroita sisältävät rivit.



Haetaan rivin aloittavia numeroita.

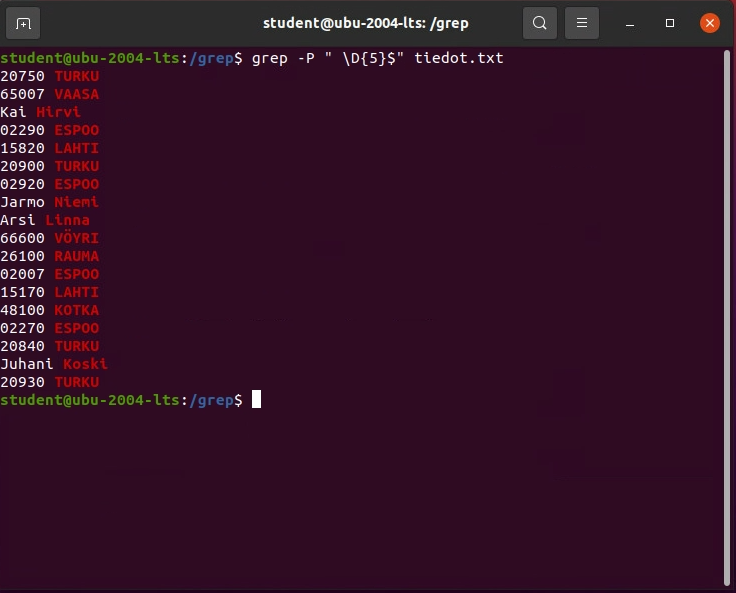
Voimme kohdistaa haun myös rivin loppuun:

**grep -P ”\d{2}$” tiedot.txt**

Haemme rivejä, jotka sisältävät kaksi numeroa ( **\d{2}** ) juuri ennen rivin loppua ( **$** ).

Jos emme halua hakea numeroita, voimme käyttää **\D** -komentoa:

**grep -P ” \D{5}$” tiedot.txt**

****

Haetaan muita kuin numeroita rivin lopusta.

Vastaako näkymäsi yllä olevaa kuvaa? Jos ei, katso käskyämme uudestaan: ensimmäisen lainausmerkin jälkeen tulee tyhjä tila, eli välilyönnin painallus. Vasta sen jälkeen tulee käsky **\D{5}$**. Grep etsii rivit, jotka päättyvät välilyöntiin ja viiteen merkkiin, jotka eivät ole numeroita.

Yllä olevassa kuvassa näemme rivejä, jotka sisältävät kaupunkien nimien lisäksi kolme henkilöä, joiden sukunimet ovat viisikirjaimisia. Entä jos haluamme hakea vain kaupunkien nimet?

Tarkoituksena on siis hakea rivejä, joiden kuusi viimeistä merkkiä ovat välilyönti ja viisi isoa kirjainta. Voimme kirjoittaa näin:

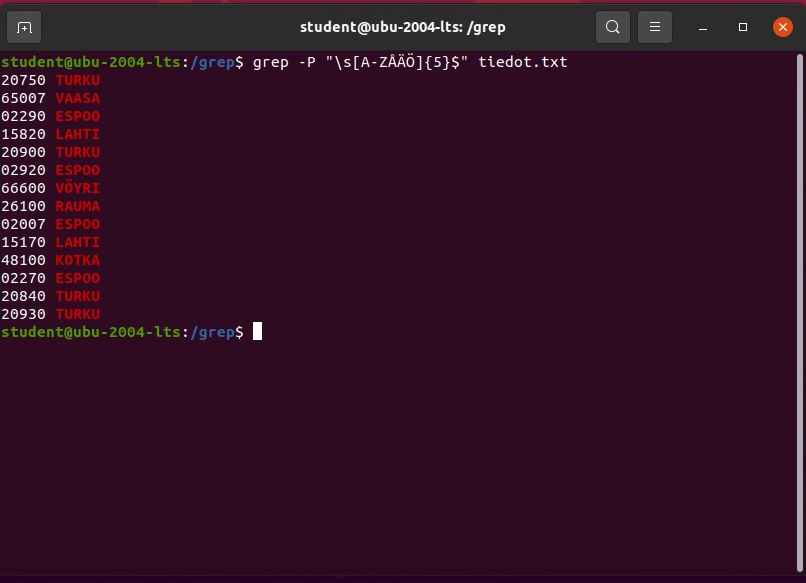
**grep -P ” [A-Z]{5}$” tiedot.txt**

Muutimme hakua niin, että käskemme grepin etsiä välilyönnin jälkeen isoja kirjaimia A:sta Z:aan ( **[A – Z]** ). Kirjaimia pitää olla tasan viisi kappaletta.

Ongelmana on nyt se, että ”Vöyri” -niminen paikkakunta ei ole tuloksessa mukana.

Valitettavasti emme voi korvata Z -kirjainta Ö:llä, jos Linuxin lokalisaatioasetukset eivät tunnista skandinaavisia kirjaimia. Pääsemme kyllä eteenpäin, kunhan laitamme hakuun skandinaaviset merkit yksi kerrallaan. Tämän lisäksi laitamme välilyönnin tilalle **\s**, joka edustaa yhtä välilyönnin tai tab -painikkeen painallusta:

**grep -P ”\s[A-ZÅÄÖ]{5}$” tiedot.txt**

****

Haku palauttaa vain kaupunkien nimet.

Seuraavaksi etsimme katuosoitteita, jotka sisältävät joko ”tie” tai ”katu” -merkkijonot. Teemme hausta mahdollisimman tarkan, jotta emme vahingossakaan saa ruudulle ylimääräistä tietoa.

Katuosoitteen sisältävän rivin rakenne on luultavasti seuraava: ensin isoja ja pieniä kirjaimia, sitten joko ”katu” tai ”tie”, tyhjää tilaa ja lopuksi numeroita. Aloitetaan kirjoittamalla:

**grep -iP ”[a-zåäö]+**

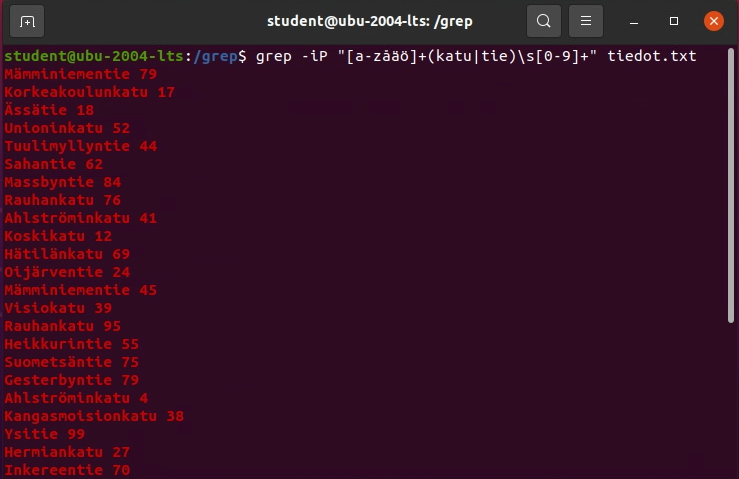
Haemme siis hakasulkujen sisällä olevia kirjaimia, yhtä tai useampaa ( **+** ), kirjainkoosta välittämättä ( **-i** ).

Osoitteen ensimmäisen osan on päätyttävä joko merkkijonoon ”katu” tai ”tie”. Niiden jälkeen tulee yksi tyhjä lyönti ( **(katu|tie)\s** ):

**grep -iP ”[a-zåäö]+(katu|tie)\s**

Lopuksi haemme rivin päättäviä numeroita ( **[0-9]** ), joita saa olla yksi tai enemmän:

**grep -iP ”[a-zåäö]+(katu|tie)\s[0-9]+” tiedot.txt**

****

Etsitty “katu” ja “tie” -päätteiset osoitteet.

Voimme laskea tuloksien määrän ( **-c** ) ja lisätä mukaan rivinumerot ( **-n** ):

**grep -icP ”[a-zåäö]+(katu|tie)\s[0-9]+” tiedot.txt**

**grep -inP ”[a-zåäö]+(katu|tie)\s[0-9]+” tiedot.txt**

Hakutulosten järjestäminen aakkosjärjestykseen käy lisäämällä haun perään **sort** -käsky:

**grep -iP ”[a-zåäö]+(katu|tie)\s[0-9]+” tiedot.txt | sort**

Lopuksi vielä yksi samantyyppinen haku.

Tiedot.txt pitää sisällään neljän henkilön sähköpostiosoitteet, jotka tiedoston luoja on sinne huolimattomuuttaan laittanut. Yritetään etsiä ne säännöllistä lauseketta käyttäen.

Sähköpostiosoite voidaan jakaa kahteen osaan: ennen ja jälkeen @ -merkin. Sallittuja merkkejä kummallakin puolella ovat yleisesti latinalaiset aakkoset ( **[A-Za-z]** ) , numerot ( **[0-9]** ) ja erikoismerkit, joista yleensä kuitenkin käytetään vain muutamaa ( **[.+\_-]** ). Suomalaisissa fi -päätteisissä tunnuksissa saa kuitenkin käyttää myös ”ääkkösiä”.

Rakennamme haun pala palalta. Aloitamme lisäämällä hakuun isot ja pienet latinalaiset aakkoset, numerot ja skandinaaviset kirjaimet:

**grep -P ”[\wåäö**

Huomaamme, että **\w** edustaa sekä isoja että pieniä latinalaisia kirjaimia ja numeroita. Skandinaaviset kirjaimet on lisättävä erikseen.

Sitten lisäämme erikoismerkit:

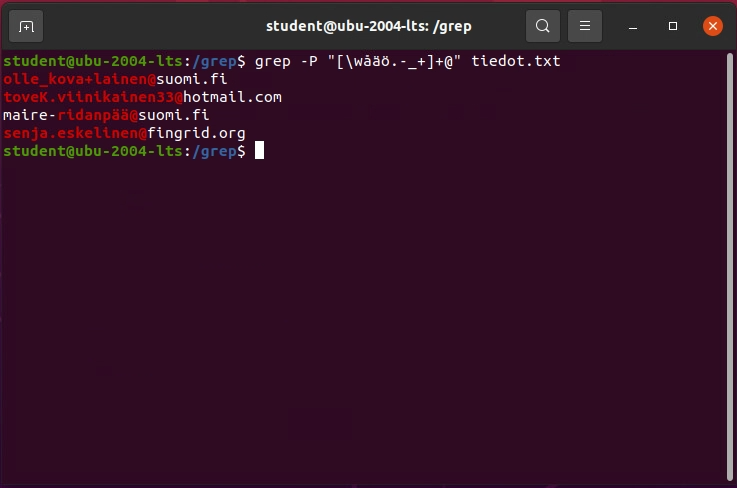
**grep -P “[\wåäö.+\_-**

Voimme nyt sulkea hakasulun. Perään lisäämme vielä **+** ja **@** -merkit. Nyt lauseen ensimmäinen puolisko on valmis.

**grep -P ”[\wåäö.-\_+]+@**

Voimme mielenkiinnosta ajaa tämän haun**:**

**grep -P ”[\wåäö.-\_+]+@” tiedot.txt**

****

Osittainen onnistuminen sähköpostiosoitteita haettaessa.

Löydämme kyllä kaikki neljä sähköpostiosoitetta, mutta haku ei ole onnistunut täydellisesti. Näemme punaisella ne merkit, jotka grep on tunnistanut @ -merkkiin asti. Hämmennystä herättää kolmas osoite, jonka alkuosa on jäänyt tunnistamatta. Tämä johtuu siitä, että olemme tehneet virheen hakiessamme erikoismerkkejä: grep tulkitsee väliviivan ( **-** ) tarkoittavan ”kaikki väliltä”, aivan kuten etsiessämme kirjaimia väliltä a-z.

Aikaisemmin kirjoitimme **\.** kun halusimme etsiä kirjaimellisesti pistettä. Voimme tehdä tässä saman korjaamalla hakuamme seuraavasti:

**grep -P ”[\wåäö.\-\_+]+@” tiedot.txt**

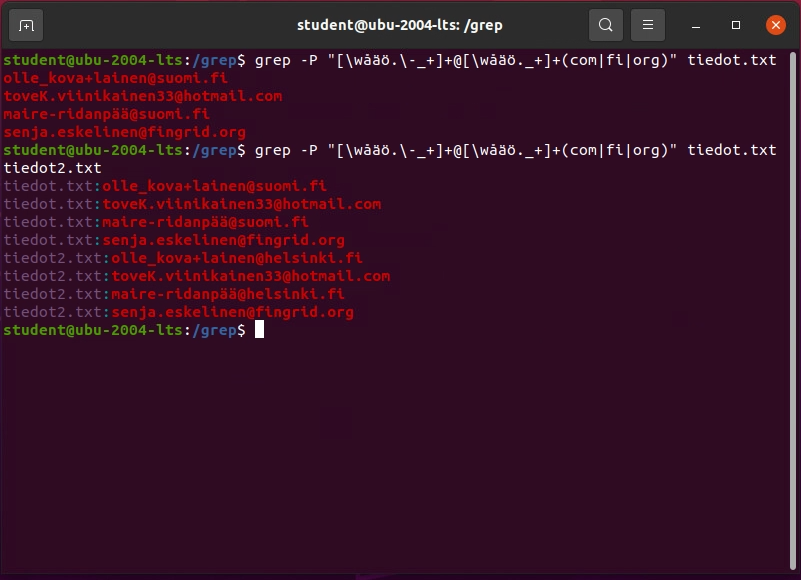
Nyt kaikki on tunnistettu @ -merkkiin asti.

Sähköpostiosoitteen jälkimmäinen osa koostuu samoista merkeistä kuin ensimmäinenkin. Aivan lopussa kerromme grepille, mitä päätteitä on osoitteen lopusta etsittävä **(com|fi|org)**:

**grep -P ”[\wåäö.\-\_+]**[**+@[\wåäö.\-\_+]+(com|fi|org)**](mailto:+@[\wåäö.\-_+%5d+(com|fi|org))**” tiedot.txt**

Jos haluamme etsiä säännöllisiä lausekkeita useammasta tiedostosta, voimme lisätä uuden edellisen perään:

**grep -P ”[\wåäö.\-\_+]**[**+@[\wåäö.\-\_+]+(com|fi|org)**](mailto:+@[\wåäö.\-_+%5d+(com|fi|org))**” tiedot.txt tiedot2.txt**

****

Sähköpostiosoitteet on haettu sekä tiedot.txt- että tiedot2.txt -tiedostosta.

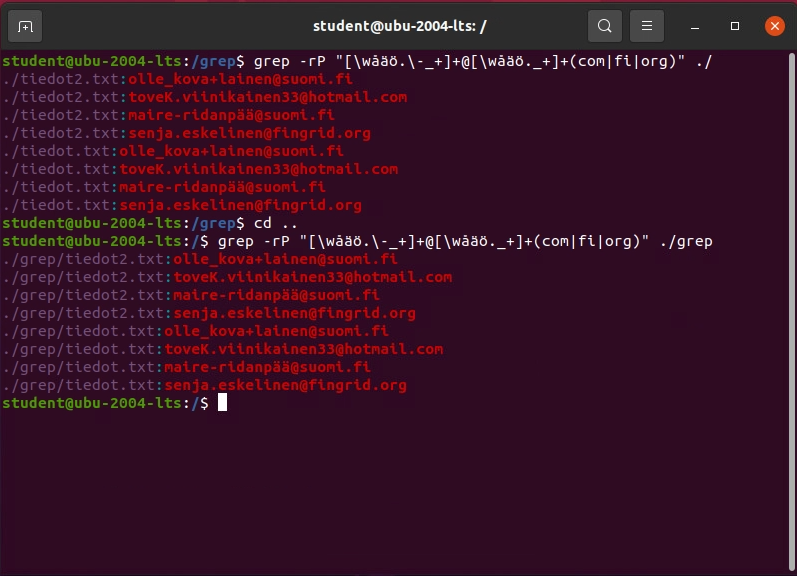
Jos tiedostojen lisääminen haun perään on liian vaivalloista, voimme muuttaa hakua seuraavasti:

**grep -rP ”[\wåäö.\-\_+]**[**+@[\wåäö.\-\_+]+(com|fi|org)**](mailto:+@[\wåäö.\-_+%5d+(com|fi|org))**” ./**

Ohjaamme grepiä käymään läpi kaikki tiedostot ( **-r** ) kansiosta, jonka sisällä olemme ( **./** ) ja myös kaikista alikansioista, jos niitä löytyy.

Jos olemme kansion ulkopuolella, voimme ohjata haun haluamaamme kansioon. Tässä esimerkissä olemme yhden tason grep -kansion yläpuolella:

**grep -rP ”[\wåäö.\-\_+]**[**+@[\wåäö.\-\_+]+(com|fi|org)**](mailto:+@[\wåäö.\-_+%5d+(com|fi|org))**” ./grep**



Haetaan useammasta kohteesta, oman kansiosijaintimme ulkopuolelta.

Komennot

**-A + numero** = palauttaa haetun merkkijonon sisältävän rivin, sekä halutun määrän sen jälkeen tulevia rivejä

**-B + numero** = palauttaa haetun merkkijonon sisältävän rivin sekä halutun määrän sitä edeltäviä rivejä

**-C + numero** = palauttaa haetun merkkijonon sisältävän rivin sekä halutun määrän rivejä sen ympäriltä

**-i** = palauttaa rivit, jotka sisältävät haetun merkkijonon, merkkikoosta välittämättä

**-c** = palauttaa haetun merkkijonon sisältävien rivien lukumäärän

**-n** = palauttaa rivit, jotka sisältävät haetun merkkijonon ja lisäksi rivinumerot tiedoston sisällä

**-P** = otetaan käyttöön Perl -yhteensopivat säännölliset lausekkeet

**-r** = palauttaa tuloksia alikansioiden sisällä olevista tiedostoista

**-v** = palauttaa kaikki rivit, jotka eivät sisällä haluttua merkkijonoa

**-w** = palauttaa rivit, jotka sisältävät tarkalleen merkkijonoa vastaavat tulokset

Erikoismerkit

**\d** = kaikki numerot

**\D** = kaikki muut merkit paitsi numerot

**\s** = tyhjä tila

**\w** = kaikki merkit välilä A-Z ja a-z sekä numerot

Lähteet

Bell Labs:

<https://fi.wikipedia.org/wiki/Nokia_Bell_Labs>

Ed:

<https://www.howtoforge.com/linux-ed-command/>

Satunnaiset nimet:

<https://www.fakenamegenerator.com>

Grep-perusteet:

<https://www.youtube.com/watch?v=VGgTmxXp7xQ>

Säännölliset lausekkeet wiki: <https://www.linux.fi/wiki/S%C3%A4%C3%A4nn%C3%B6llinen_lauseke#Operaattorit>

Säännölliset lausekkeet perusteet:

<https://www.youtube.com/watch?v=sa-TUpSx1JA>

Sähköpostiosoitteet:

<https://help.returnpath.com/hc/en-us/articles/220560587-What-are-the-rules-for-email-address-syntax->

ja

<https://www.traficom.fi/fi/viestinta/fi-verkkotunnukset/kansallisten-merkkien-kayttaminen-verkkotunnuksessa>

Tiedot.txt sisältö

Selma Niva

Mämminiementie 79

20750 TURKU

Maija Lampen

Korkeakoulunkatu 17

50520 MIKKELI

Helmi Virrankoski

Ässätie 18

49420 HAMINA

Vihtori Sibelius

Unioninkatu 52

43100 SAARIJÄRVI

Olle Kovalainen

Tuulimyllyntie 44

51900 JUVA.

Tove Viinikainen

Pietarinkuja 87

61807 KAUHAJOKI

Einojuhani Lampen

Sahantie 62

33100 TAMPERE

Maire Ridanpää

Massbyntie 84

04440 JÄRVENPÄÄ

Jarno Söderström

Rauhankatu 76

00810 HELSINKI

Senja Eskelinen

Ilmalankuja 36

28400 ULVILA

Raine Sundqvist

Ahlströminkatu 41

80220 JOENSUU

Kalevi Meriluoto

Rörgrunsvägen 69

65007 VAASA

Kai Hirvi

Koskikatu 12

02290 ESPOO

Jouni Saariaho

Hätilänkatu 69

15820 LAHTI

Tommy Pappila

Oijärventie 24

95600 YLITORNIO

Anna-Liisa Grönholm

Mämminiementie 45

20900 TURKU

Pekka Mikkola

Visiokatu 39

38950 HONKAJOKI

Senja Luostarinen

Rauhankatu 95

00900 HELSINKI

Josefiina Hynninen

Itätuulenkuja 79

02920 ESPOO

Jarmo Niemi

Heikkurintie 55

62900 ALAJÄRVI

Arsi Linna

Suometsäntie 75

00720 HELSINKI

Ari Salonen

Gesterbyntie 79

05850 HYVINKÄÄ

Paula Lenkkeri

Norra Esplanaden 18

00260 HELSINKI

Säde Kilkka

Ahlströminkatu 4

80130 JOENSUU

Jaska Vanhanen

Kangasmoisionkatu 38

40620 JYVÄSKYLÄ

Arvi Lindberg

Hirvenpolku 72

66600 VÖYRI

Riku Aaltonen

Ysitie 99

33480 YLÖJÄRVI

Anu Kinnunen

Hermiankatu 27

34800 VIRRAT

Ensio Kapanen

Inkereentie 70

26100 RAUMA

Sebastian Sinervo

Kajaaninkatu 72

02007 ESPOO

Ari Lax

Kauhajärventie 35

62200 KAUHAVA

[olle\_kova+lainen@suomi.fi](mailto:olle_kova+lainen@suomi.fi)

[toveK.viinikainen33@hotmail.com](mailto:toveK.viinikainen33@hotmail.com)

[maire-ridanpää@suomi.fi](mailto:maire-ridanpää@suomi.fi)

senja.eskelinen@fingrid.org

Jyrki Heikkilä

Tawastintie 9

15170 LAHTI

Risto Haapoja

Ysitie 72

33840 TAMPERE

Rita Nuorela

Massbyntie 31

04407 JÄRVENPÄÄ

Kaj Styrman

Pesolantie 2

85200 ALAVIESKA

Arja Timonen

Keskiortentie 28

22340 GETA

Samsa Pohjonen

Kaarrostie 43

01230 VANTAA

Paavo Kauppinen

Vanhamaantie 32

48100 KOTKA

Justiina Wiljakainen

Koskikatu 86

02270 ESPOO

Arvi Harila

Mämminiementie 5

20840 TURKU

Juhani Koski

Haapahuhdantie 11

20930 TURKU

Åsa Myllyrinne

Keimolanpolku 34

22130 ÅBO