

Panu Aho 266300

sulautetut järjestelmät PLA-32310

Oppimispäiväkirja

|  |  |
| --- | --- |
| Tarkastaja: Mika Saari |  |
|  |  |

SISÄLLYSLUETTELO

[1. Johdanto 1](#_Toc492998912)

[2. käyttöjärjestelmät 2](#_Toc492998913)

[3. ohjelmointi 12](#_Toc492998914)

[Lähteet 18](#_Toc492998915)

LIITE 1: TUNTIKIRJANPITO

.

# Johdanto

Oppimispäiväkirja on tehty syksyllä 2017 TTY:n Porin laitoksen kurssille Sulautetut Järjestelmät (PLA-32310). Päiväkirjassa on tarkoitus käsitellään kurssin eri aihealueet sekä millaisia lähestymistapoja olen henkilökohtaisesti käyttänyt sisältääkseni kurssilla käsiteltävät asiat. Haasteita kurssin suoritukseen tuo suoritustapani, joka on etäpainotteinen itsenäinen työskentely, mutta en usko tämän kuitenkaan kasvavan ylitsepääsemättömäksi ongelmaksi.

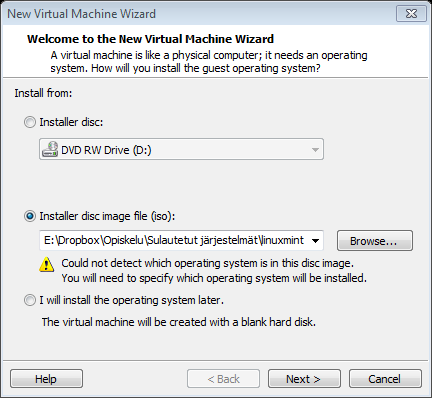
Oppimispäiväkirja on nimensä mukaisesti muodoltaan päiväkirjamainen, tarkoittaen sitä, että kerran kirjoitettuja merkintöjä ei jälkikäteen ole juurikaan palattu editoimaan. Tekstin koherenttius kärsinee tästä hieman, mutta etuna saavutetaan paras mahdollinen kuvaus nimenomaan oppimisprosessista, mihin sudenkuoppiin on pudottu ja miten niistä on (toivon mukaan) myös kavuttu ylös.

Teksti tulee päivittymään ja elämään kurssin aikana. Liitteessä 1 esitetään viiteellinen tuntikirjanpito.

# käyttöjärjestelmät

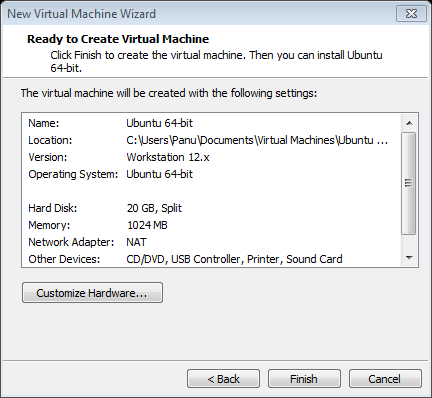
*9.9.2017 – 10.9.2017*

Tutustuttiin virtuaalikoneiden mahdollisuuksiin sekä Linux-käyttöjärjestelmän asennukseen sekä peruskäyttöön vMWare-virtualisointiympäistössä. Linux-distroksi valittiin Mint, jonka iso-levykuva ladattiin [www.linuxmint.com](http://www.linuxmint.com) sivustolta. VMwaresta asennettiin ’Workstation 12 Player’-versio, joka on ilmainen ei-kaupallisessa käytössä. Uusi virtuaalikone otettiin käyttöön ’New Virtual Machine Wizard’-velhotoiminnolla. Käyttöjärjestelmän kohdalle asetettiin em. ladattu ISO-image. Vaihtoehtona olisi ollut myös uuden virtuaalikoneen alustaminen tyhjällä kovalevyllä, mikäli oltaisiin haluttua asentaa käyttöjärjestelmä myöhemmin manuaalisesti.



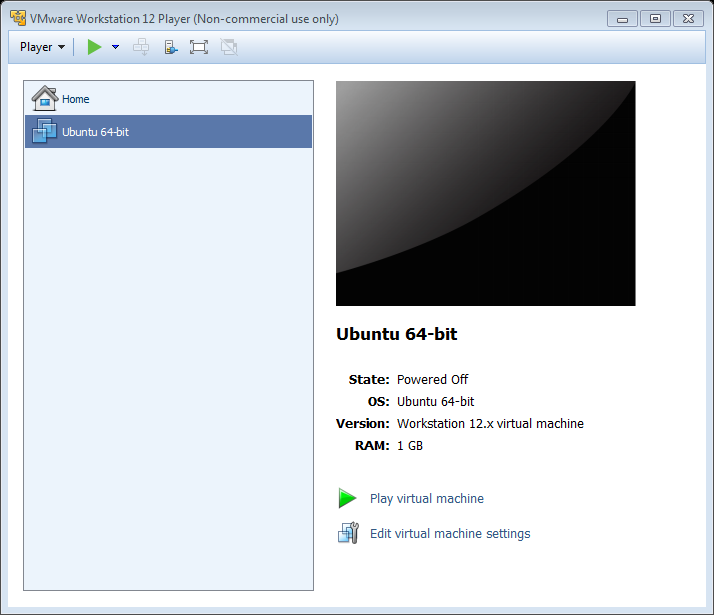
Kuva 1. Virtuaalikoneen luontityökalu

Seuraavana piti valita asennettavan Linuxin tyyppi. Listalla ei ollut valittavana Mintiä, mutta Mintin dokumentaatiosta löytyi tieto että se perustuu Ubuntun versioon 16.04. Niinpä tässä kohdassa valittiin Linux-distroksi Ubuntu 64-bit. Seuraavaksi oli paljon valintaikkunoita liittyen tallennuslokaatioihin sekä virtualisoitavan koneen kokoonpanoon, jota olisi myös halutessaan päässyt kustomoimaan. Nyt edettiin kuitenkin hyväksyen velhon ehdottamat oletusasetukset.



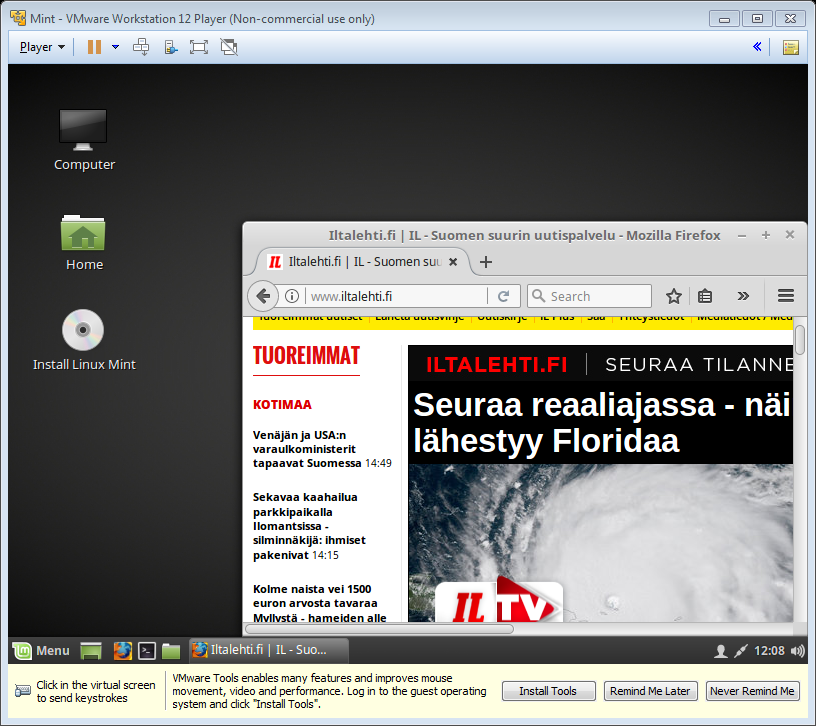
Kuva 2. Uuden virtuaalikoneen kokoonpano

Näin oli saatu virtuaalikone luotua ja oli aika pistää kone käyntiin Play virtual machine-painikkeesta.



Kuva 3. Juuri luotu virtuaalikone vMwaren home-näkymässä

Ensimmäisellä käynnistysyrityksellä saatiin pitkä virheilmoitus, jossa mm. moitittiin käyttäjää siitä, että host-järjestelmä olisi pitänyt käynnistää uudelleen vmWaren asennuksen jälkeen (Asennuksen aikana tällaista vaatimusta ei tosin esitetty). Kun uudelleenkäynnistyskään ei auttanut, ruvettiin tutkimaan tarkemmin virheilmoituksen anatomiaa. Pikaisen selvittelyn ja google-hakujen perusteella ilmeni, että ongelma koski Intelin Vt-x virtualisointiteknologiaa, joka syystä tai toisesta ei ollut isäntäkoneen Core i7-prosessorissa käytettävissä. Sen sijaan että oltaisi lähdetty säätämään hostin BIOS-asetuksia, ongelma ratkaistiin lataamalla ja asentamalla Mintistä 32-bittinen versio, joka pystyy toimimaan ilman Vt-x:ää. Tämän jälkeen virtuaalikone buuttasi ja verkkoyhteydetkin näyttivät saman tien pelaavan hienosti.

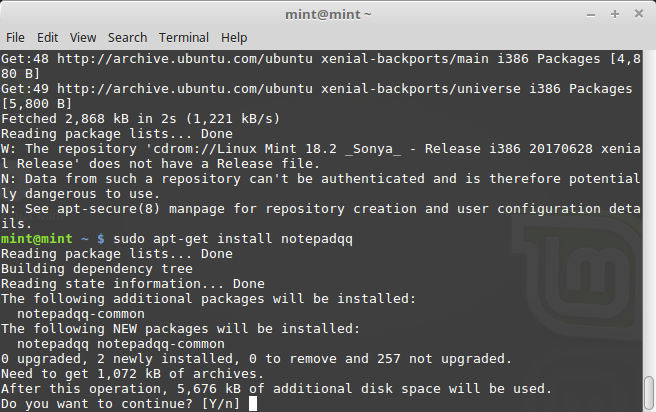


Kuva 4. Mint pyörii komeasti virtuaalikoneessa.

Tämän jälkeen oli aika hieman harjoitella Linuxin peruskäyttöä. Linuxissa ohjelmien asentaminen käy näppärästi suoraan komentoriviltä. Harjoitustehtävänä oli laatia Hello World- sovelluksia muutamilla eri ohjelmointikielillä, joten ensimmäinen mitä tarvittiin oli tietysti kunnollinen tekstieditori. Kun ensin oli saatu käyttöön suomalainen näppäimistökonfiguraatio (setxkbmap fi), virtuaaliympäristöön asennettiin notepadqq-ohjelmisto suorittamalla ohjelmiston kotisivuilta [1] saatujen ohjeiden mukaiset komennot:

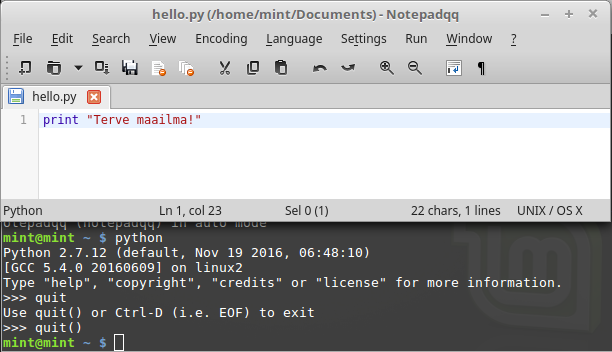
sudo add-apt-repository ppa:notepadqq-team/notepadqq  
sudo apt-get update  
sudo apt-get install notepadqq

Kuvassa 5 on lisätty ja päivitetty tarvittavat repositoriot ja ollaan viimeistelemässä asennusta.



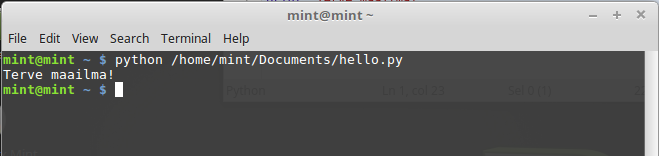
Kuva 5. Tekstieditorin asennusta terminaalin kautta

Nyt voitiin laatia juuri asennetulla tekstieditorilla yksinkertainen python-sovellus, joka tulostaa ruudulle tekstin ”Terve maailma!”. Koodi tallennettiin nimellä ’hello.py’ polkuun /home/mint/Documents. Terminaalikomennolla python saatiin selville, että järjestelmässä oli valmiina asennettuna Python version 2.7.12 eli sen asennuksesta ei tässä kohtaa tarvitsisi huolehtia.



Kuva 6. Hello World-koodi tekstieditorissa sekä Pythonin versiotiedot Terminalissa

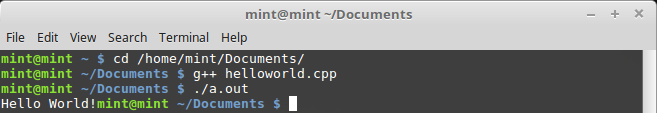
Koodi ajettiin komentoriviltä windows-ympäristöstäkin tutun oloisella python-syntaksilla:



Kuva 7. Ensimmäinen python-sovellus Linux-ympäristössä näyttää toimivan!

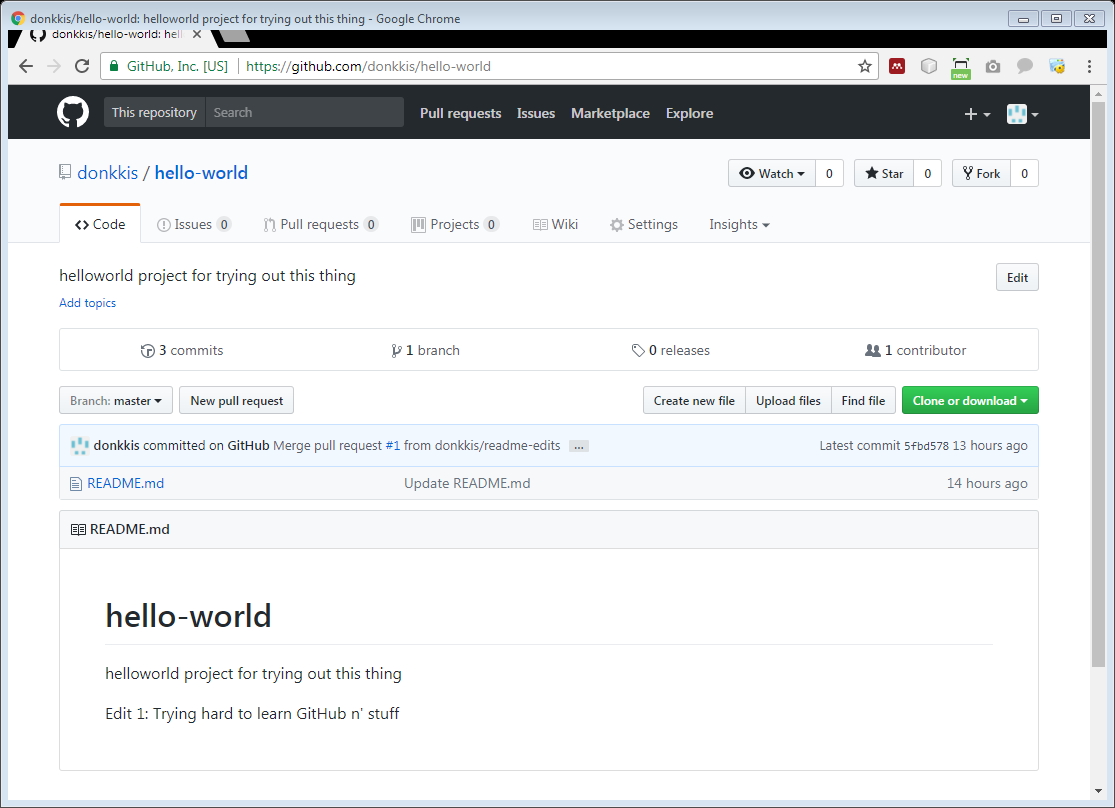
Yksi asia missä mentiin aluksi hieman vipuun oli virtuaalikoneen sammuttaminen. Jos sammutuksen tekee ’normaalilla’ tavalla käyttöjärjestelmän valikkojen kautta, on mitä ilmeisemmin seurauksena guest-järjestelmän kovalevyn tyhjentyminen. Parempi tapa on käyttää vmwaren suspend-toimintoa, jolloin seuraavalla käyttökerralla voidaan palata samaan tilaan kuin mihin viimeksi jäätiin.

Seuraavaksi oli vuorossa C++ testisovelluksen laatiminen jota varten asennettiin g++ työkalupaketti komennolla sudo apt install g++. HelloWorld-testikoodi tallennettiin jälleen Documents-hakemistoon nimellä helloworld.cpp. Tämän jälkeen koodi voitiin kääntää ja suorittaa Terminalin kautta.



Kuva 8. C++ lähdekoodin kääntäminen g++:lla sekä ajaminen

Myös GitHub oli uusi tuttavuus, joten siihen perehtyminen aloitettiin luomalla tunnukset [3] sekä harjoittelemalla GitHubin guides-sivulta löytyvä Hello World-tutoriaali, jossa tutustuttiin versionhallinnan perustoiminnallisuuksiin selainkäyttöliittymän kautta [2]. Ja eipä aikaakaan, kun oli luotu ensimmäinen repositorio sekä harjoiteltu hieman branchien, committien sekä mergejen tekemistä.



Kuva 9. GitHub-harjoittelua

Seuraavaksi harjoiteltiin GitHub:ssa toimimista Terminalin kautta:

//Ensimmäisellä käyttökerralla pitää konfiguroida user-tiedot

git config –-global user.email “ ” // sähköposti…

git config –-global user.name “donkkis”

//sitten itse asiaan

git clone <https://github.com/donkkis/hello-world>

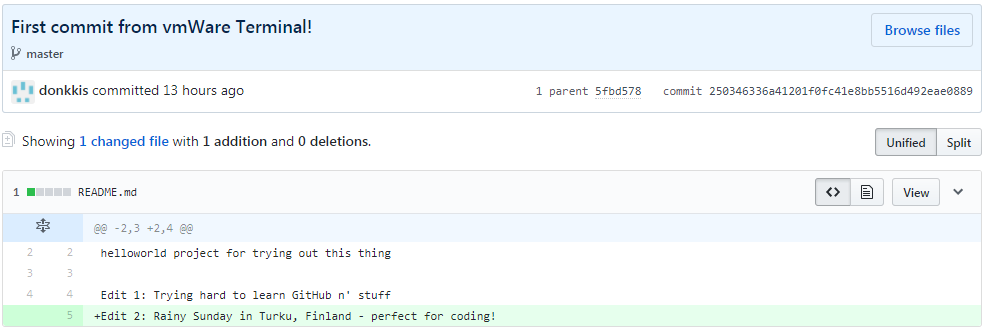
//Käytiin tekemässä muutoksia repoon, esim. README-tiedoston sisältöön…

git commit –am “First commit from vmWare Terminal!”

git push –u origin master

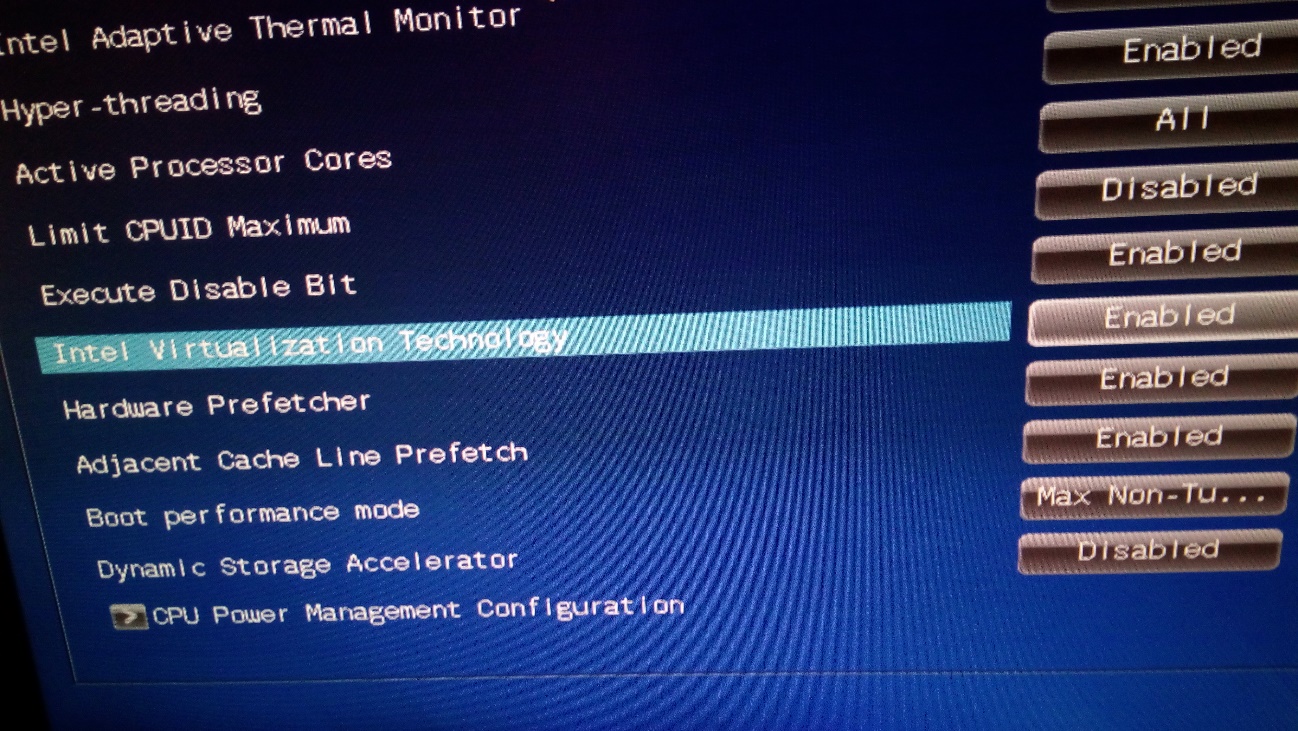
//Kysyy gh-tunnuksia, syötä käyttis ja passu kysyttäessä

Ja niinhän siinä kävi että tehdyt muutokset päivittyivät repositorioon. Vertailutoiminto web-näkymässä korostaa mukavasti commitissa tehdyt muutokset.



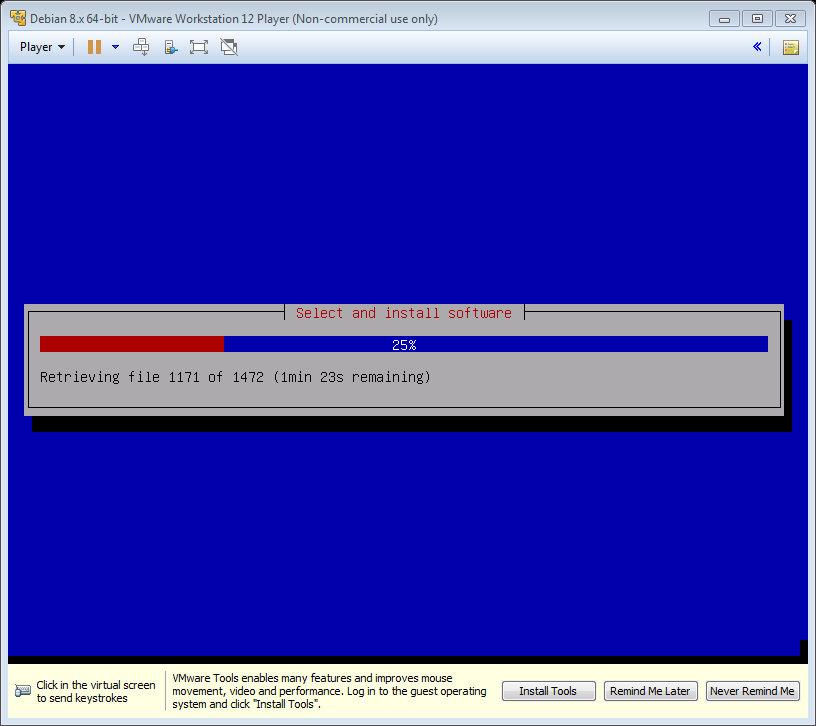
Kuva 10. Terminalista tehty commit

LAMP:n testaamista varten oli välttämätöntä saada 64-bittinen virtualisointiympäristö toimimaan, sillä kyseinen sovellus oli saatavilla vain 64-bittisenä. Suunnattiin siis tutkailemaan isäntäkoneen BIOS-asetuksia, mistä oikea täppä löytyikin lyhyen haeskelun jälkeen.



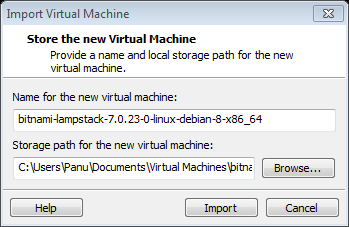
Kuva 11. 64-bittisen virtualisoinnin enablointi isäntäkoneen BIOSista

LAMP ei kuitenkaan ottanut toimiakseen 64-bittiselläkään Mintillä, joten seuraavaksi kokeiltiin alustaa uusi virtuaalikone debian 9.1.0 käyttöjärjestelmällä. Debianin asennus oli huomattavasti pidempi prosessi kuin Mintin ja valintoja piti tehdä yllinkyllin. Pakkaus oli myös ilmeisesti huomattavasti Mintiä laajempi sillä virtuaalikoneessa suoritettu asennusohjelma haeskeli ohjelmistokirjastoja verkon yli hyvän tovin. Samalla kun asennettiin uusi käyttöjärjestelmä, käyttökokemuksen parantamiseksi virtuaalikoneeseen lisättiin myös hieman muistia sekä kahden suoritinytimen käyttömahdollisuus.

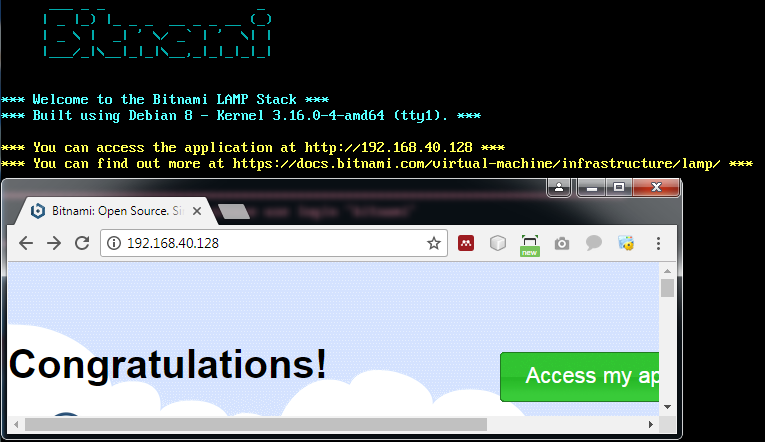


Kuva 12. Debian asentuu virtuaalikoneeseen

LAMPia ei vieläkään saatu toimimaan vaivalla asennetussa debian-ympäristössä. Tässä vaiheessa luettiin ohjeet ja havaittiin, että LAMP:n sivuilta ladattu Ova-tiedostoon on ilmeisesti paketoitu kokonainen valmis virtuaalikone, joka sisältää mm. käyttöjärjestelmän sekä apache-palvelimen käyttämät sovellukset esiasennettuna. Ova-virtuaalikone voidaan ottaa käyttöön vmWaren Import Virtual Machine-toiminnolla.

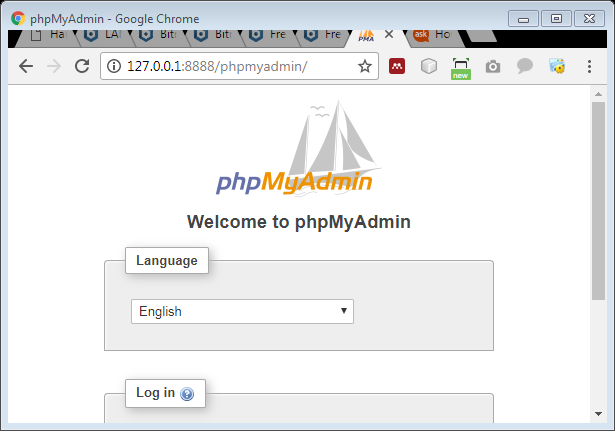


Oletusasetuksista poiketen virtuaalikoneen verkkokokoonpano säädettiin tilaan ’Host only’, koska virtuaaliserverin toimintaa haluttiin tässä tapauksessa testata nimenomaan isäntäkoneella. Näiden säätöjen jälkeen serveri saatiinkin pystyyn ja yhteyskin näytti toimivan.



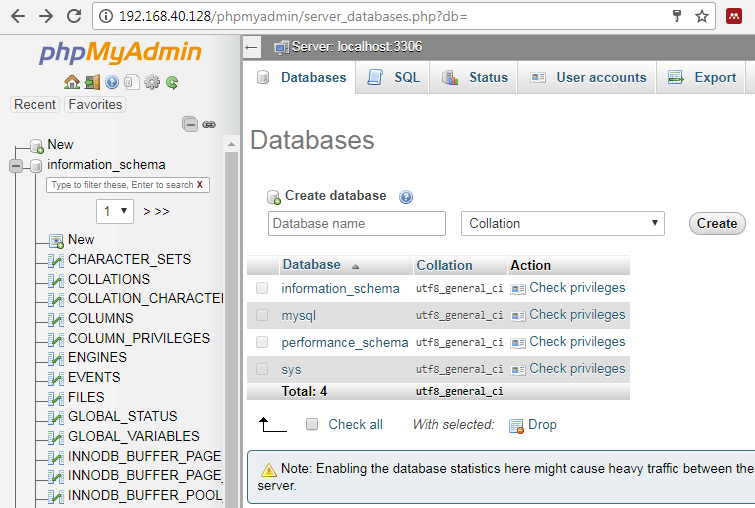
Kuva 13. LAMP-serveri virtuaalikoneessa sekä isäntäjärjestelmän selainikkuna, jolla onnistuneesti otetty yhteys serveriin

Seuraavaksi kokeiltiin phpMyAdminin toimintaa. Oletuksena phpmyadmin hyväksyy vain lokaalit yhetydenotot, joten ensin piti luoda Puttylla isäntäkoneesta virtuaaliserveriin SSH-tunnelointi noudattaen lähteissä [4] ja [5] annettuja ohjeita. Samalla tuli harjoiteltua debianin tiedostohierarkiassa liikkumista sekä tekstitiedostojen editointia komentorivin kautta ja ainakin kommennot ls, edit sekä view tulivat tutuiksi.



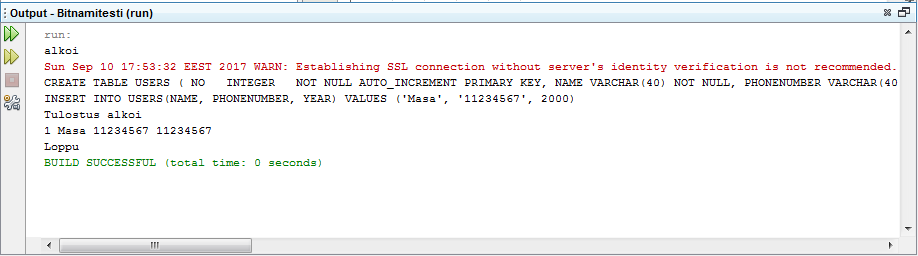
Kuva 14. phpmyadmin kirjautuminen SSH-tunneloinilla

Tässä vaiheessa luettiin jälleen ohjeet, ja käytiin muokkaamassa tiedostoa /opt/bitnami/apps/phpmyadmin/conf/httpd-app.conf siten, että yhteys phpmyadminiin saadaan myös ilman ssh-tunnelointia. Tiedostosta vaihdettiin rivi Require Local muotoon Require all granted [6]. Kun palvelin oli käynnistetty uudestaan komennolla sudo ./ctlscript.sh restart, phpmyadminiin pääsi käsiksi myös ilman ssh-yhteyttä polusta 192.168.40.128/phpmyadmin. Pitkähkön päänraapimisen jälkeen arvattiin myös oikeat oletustunnukset (root/bitnami) ja päästiin vihdoin tutkimaan phpmyadminin näkymiä, missä oli joitakin järjestelmän ilmeisesti automaattisesti generoimia tietokantoja valmiina.

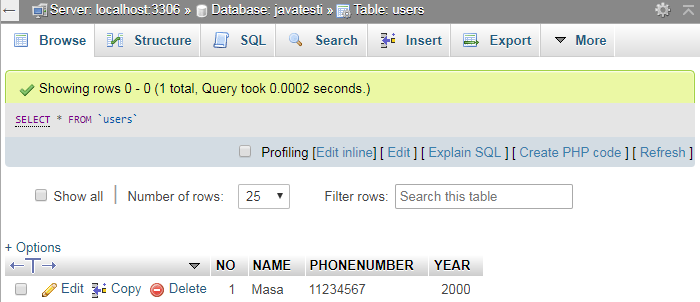


Kuva 15. Tietokantojen hallintanäkymä phpmyadminissa

Luotiin seuraavaksi phpmyadminin toiminnoilla uusi tietokanka ’javatesti’ sekä liitettiin tähän käyttäjä ’testaaja’. Suoritettiin komento portinavauskomento sudo afq allow 3306 sekä kommentoitiin tiedostosta /opt/bitnami/mysq/my.cnf rivi ’bind-address’ jotta mahdollistetaan tietokannan käyttö etänä. Suoritettiin valmiina saatu esimerkkikoodi [7] tietokannan käsittelyn demonstroimiseksi. Kaikesta päätellen koodi sai yhteyden kantaan ja taulujen muokkaus onnistui kuten pitikin.



Kuva 16. Testikoodin suoritus netbeansissa...

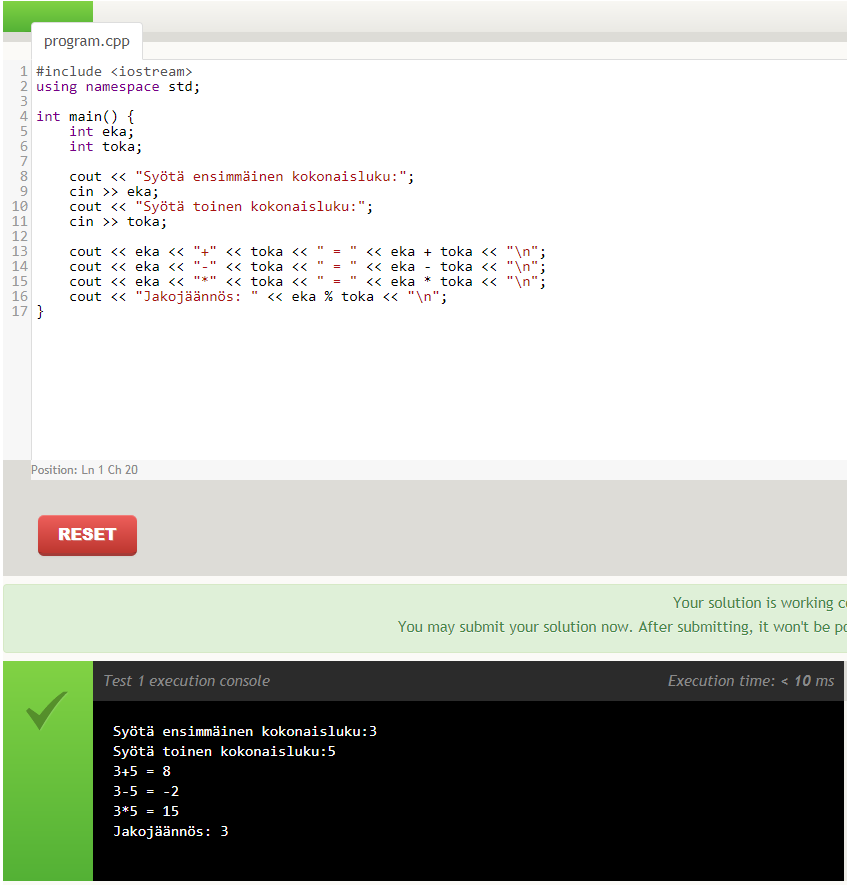


Kuva 17. ...ja tulosten tarkastus phpmyadminin kautta

# ohjelmointi

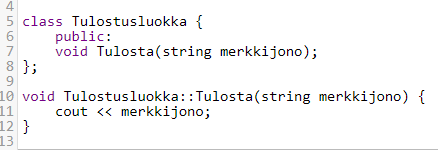
*11.9.2017 – 12.9.2017*

Osiossa tehtiin C++ sekä Python-ohjelmointi –sekä visailuharjoituksia viope-alustalla. Tehtävissä ratkaistavat ongelmat olivat melko helppoja sekä aiemmilta ohjelmointikursseilta tuttuja, joten pääpaino harjoituksella oli kyseisten kielten erityispiirteiden opettelussa. Pythonista oli kirjoittajalla jonkin verran aiempaa kokemusta, C++:sta ei käytännössä lainkaan.



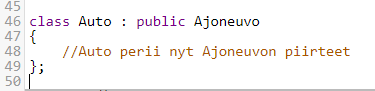
Kuva 18. Koodausta viope-ympäristössä, jossa automaattitestit tarkistavat ratkaisun oikeellisuuden.

C++:n osalta kenties suurimpana oppimiskokemuksena oli Javaan nähden huomattavan erilainen luokkien käsittely. Ainakin kyseisen viope-kurssin esimerkeissä vallitsevana käytäntönä oli, että luokan esittelyä seuraavien aaltosulkeiden sisään kirjoitetaan attribuuteista ja metodeista vain abstraktit esittelyt, ja metodien toteutukset annetaan muualla koodissa syntaksilla LuokanNimi::MetodinNimi(parametrit){}. Ja mikäli toteutus on annettu, on muistettava antaa myös esittely tai muuten ohjelma ei toimi odotetusti.



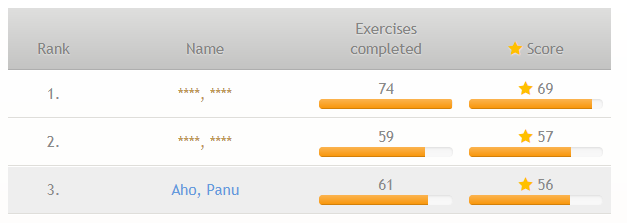
Kuva 19. Luokkien käyttö C++:ssa

Periyttäessä luokkia Javasta tutun extends-avainsanan korvaa kaksoispiste ja public –avainsana:



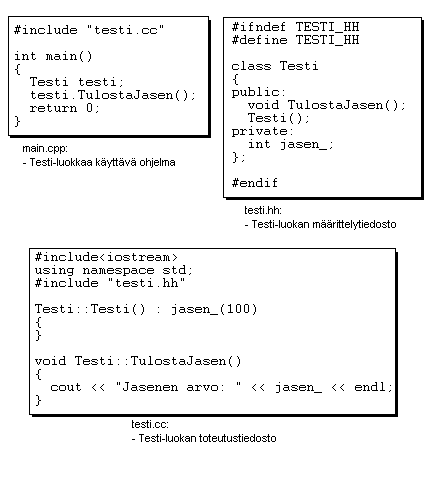
Kuva 20. Luokkien periyttämissyntaksi

C++:n peruskurssista suoritettiin yhteensä 61/74 harjoitusta, joista saatiin kerättyä kasaan 56 pistettä (joihinkin quiz-kysymyksiin ei pystynyt viopen bugaamisen takia vastaamaan!).



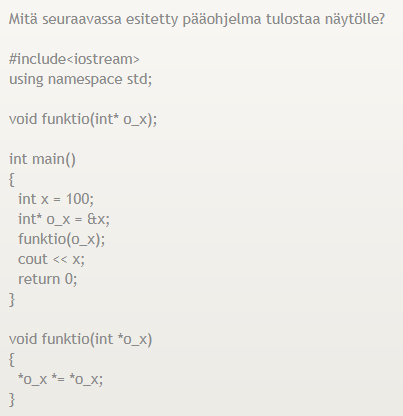
Kuva 21. Yhteenveto suoritetuista harjoituksista, C++ kurssi

Harjoituksia jatkettiin C++ oliokurssin parissa, kiinnittäen nyt enemmän huomiota myös Viopessa tarjoiltuun kirjalliseen materiaaliin. Aiemmin tehty luokkien määrittelyn ja toteutuksen erottamiseen liittyvä huomio osoittautui paikkansa pitäväksi. Luokan attribuutit (jäsenmuuttuja) sekä metodit (jäsenfunktiot) esitetään tyypillisesti omassa määrittely (otsikko) tiedostossaan sekä jäsenfunktioiden toteutukset omassaan. Jäsenmuuttujien nimeämisessä on mitä ilmeisimmin vakiintuneena käytäntönä päättää nimet alaviivaan. Jäsenmuuttujille suositeltu näkyvyysmääre private noudattelee Javan konventiota. [8]



Kuva 22. Luokkien määrittelyparadigma C++:ssa [8]

Java-maailmasta siirtyvälle osoittimien ja viitemuuttujien käyttö on uutta. Viitemuuttujat sisältävät viittauksen muuttujan muistiosoitteeseen, ja näin aliohjelmat voivat suoraan käsitellä muuttujia siten, että muutokset välittyvät myös aliohjelman kutsujalle. Opiskeltiin myös dynaamista muistinhallintaa (muistin varaaminen new-avainsanalla ja vapauttaminen deletellä). Kun osoitinmuuttujia ei tarvita, ne kannattaa alustaa NULL-arvoilla virheviittausten välttämiseksi. Näiltäkin kommervenkeiltä Java-koodaaja on välttynyt automaattisen roskienkeruun ansiosta.



Kuva 23. Tulostaa "10000". Viitemuuttujan avulla funktion tekemät muutokset välittyvät funktion kutsujalle ilman erillistä return-lausetta [8]

Opittiin myös C++:n eri oliotyypeistä. Siteerataan tässä viopen loistavaa tiivistelmää aiheesta:

*”Automaattinen olio*

*Olio on ohjelmalohkon paikallinen olio (vrt. paikalliset muuttujat). Kun olion muodostanut ohjelmalohko päättyy, olio hävitetään ja sille varattu keskusmuistialue vapautuu*

*Dynaaminen olio*

*Oliolle varataan ajonaikaisesti luokan määrittelyn mukainen tila muistista new-operaattorilla. Ohjelmassa muodostuneiden olioiden määrä voi vaihdella eri ajokerroilla. Oliolle varattu muistitila vapautetaan delete-operaattorilla, jonka vuoksi olio voi "säilyä hengissä", vaikka ohjelmalohko, jossa se muodostettiin päättyisikin*

*Staattinen olio*

*Olio on joko globaali, tai static-määreellä paikalliseksi määritelty. Oliolle varataan luokan määrittelyn mukainen tila pääohjelman käynnistyessä. Oliolle varattu muistitila vapautuu pääohjelman päättyessä.” [8]*

Harjoiteltiin parametrillinen ja parametriton konstruktori sekä tuhoaja:

//Parametriton konstruktori

Piste::Piste() : x\_(1), y\_(1) {}

//Parametrillinen konstruktori

Piste::Piste(int x, int y) : x\_(x), y\_(y) {}

//Tuhoaja

Piste::~Piste() {

cout << "Tuhoaja suoritettu ja olio tuhottu.";

}

Viopen C++ -kurssilla ei tullut vastaan Javaan verrattavia kehittyneitä Collection-tyyppisiä säiliöluokkia. Todennäköisesti niitä kuitenkin löytyy, jos asiaa lähtisi kaivelemaan. Oliosäilöinä käytettiin nyt taulukoita, joiden koko pitää olla etukäteen päätetty. Esim. luotaessa luokka Kenneli, voitaisiin päättää Koirien maksimimääräksi 50. Taulukko alustetaan Kenneliä luotaessa esim. NULL-arvoilla. Lisättäessä tai poistettaessa Koiria täytyy pitää huoli uuden Koiran sijoittamisesta oikeaan indeksiin, ylläpitää Koirien lukumäärää erillisessä apumuuttujassa jne. Melko hankalaa, mutta kenties tämä on hinta joka joudutaan maksamaan laiteläheisyydestä ja suorituskyvystä.

23: class Kenneli

24: {

//määrittelyjä...

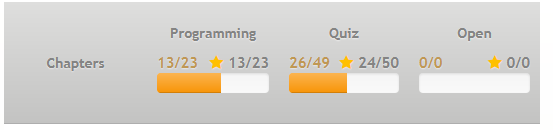
30: private:

31: Koira\* koirat\_[50];

32: int koirien\_lkm\_;

33: };

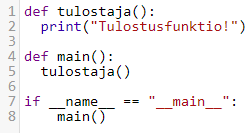
C++ oliokurssin harjoituksista suoritettiin tässä vaiheessa noin puolet, ennen kuin jatkettiin eteenpäin.



Kuva 24. Yhteenveto tehdyistä harjoituksista, C++ oliokurssi

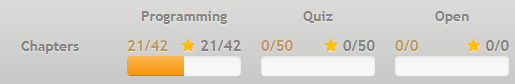
Jatkettiin seuraavaksi Python-harjoitusten merkeissä. Olennaisimmat havainnot:

* Python3 tulostussyntaksi erilainen 2.x nähden, print() käyttäytyy nyt funktiokutsun tavoin ja voidaan parametroida hauskoilla tavoilla
* raw\_input on nyt pelkkä input
* Merkkijonojen käsittely on helppoa ja hauskaa:
  + Substringejä saadaan mjono[alku:loppu+1] syntaksilla, indeksointi lähtien nollasta
  + Koko merkkijonon kääntäminen mjono[::-1]
  + Jne...
* Muuttujilla ei edelleenkään vahvaa tyypitystä
* Ohjausrakenteet tehdään edelleen sisentämällä (välilyönti, ei tab), aaltosulkeita ei käytetä
* True ja False avainsanat isolla
* Main metodin kutsutapa ohjelman aloittamiseksi on melko eksoottinen:



* Oletusparametri funktiolle määrittelyllä def function(param = ”default”)
* Listat erittäin tehokas tapa säilöä ja käsitellä tietoa

Python3-ohjelmointiharjoituksista suoritettiin 21/42 eli 50 %. Harjoituksia voidaan jatkaa vielä myöhemmin, jos aikaa jää ja/tai tähän koetaan tarvetta. Visailuharjoituksiin ei nyt kulutettu aikaa sillä asiat tuntuivat olevan melko hyvin hallinnassa.



Kuva 25. Yhteenveto tehdyistä Python-harjoituksista viopessa

Liite 1

Viitteellinen tuntikirjanpito tähän astisista tehtävistä:

|  |  |  |  |
| --- | --- | --- | --- |
| Pvm | Tunnit | Osio(t) | Tehtävät |
| 9.9.2017 | 4 | Käyttöjärjestelmät | Virtuaalikoneet, Linux, Python, C++ |
| 10.9.2017 | 8 | Käyttöjärjestelmät, Ohjelmointi | GitHub perusteet + linux integraatio, Virtualisointi rautatasolla, Linux Terminal, BitNami LAMP, SSH, mySQL/phpMyAdmin |
| 11.9.2017 | 4 | Ohjelmointi | C++ perus |
| 12.9.2017 | 6 | Ohjelmointi | C++ oliot, Python 3 |

Lähteet

1. NotepadQQ-ohjelmiston kotisivut. Viitattu 10.9.2017 <http://notepadqq.altervista.org/wp/download/>
2. GitHub Guides Hello World. Viitattu 10.9.2017 <https://guides.github.com/activities/hello-world/>
3. GitHub-tili. Viitattu 10.9.2017 <https://github.com/donkkis/>
4. Bitnami PhpMyAdmin. Bitnami Docs. Viitattu 10.9.2017 <https://docs.bitnami.com/virtual-machine/components/phpmyadmin/?utm_source=bitnami&utm_medium=virtualmachine&utm_campaign=Virtual%2BMachine>
5. Frequently asked questions for virtual machines. Bitnami Docs. Viitattu 10.9.2017 <https://docs.bitnami.com/virtual-machine/faq/#connecting-with-an-ssh-client-on-windows>
6. Verkkokeskustelu Bitnami communityssä. Viitattu 10.9.2017 <https://community.bitnami.com/t/accessing-phpmyadmin-remotely-i-get-this-security-error-for-security-reasons-this-url-is-only-accesible-using-localhost-127-0-0-1-as-the-hostname/6718>
7. Embedded With LAMP. Käyttäjän miksa007 projekti GitHub-palvelussa. Viitattu 10.9.2017 <https://github.com/miksa007/Embedded_with_LAMP>
8. Viope. Olio-ohjelmoinnin perusteet C++ -kielellä. Kurssimateriaali.