Nimi:_____ Op.num:_____

Ohjelmointityökalut

Ohjelmistokehitys on avoimen lähdekoodin aatteen lähde, joten ei tule suurena yllätyksenä että Linux-jakeluissa on mukana kattavat kehitystyökalut.

- sudo apt-get install build-essential git gitk devscripts

Ikävä kyllä tämä harkkaohje kirjoitettiin koneella jossa oli jo satoja ohjelmointikirjastoja asennettuna, joten tästä puuttuu varmaan tulevien esimerkkien tarvitsemia kirjastoja. Näiden setviminen on osa harjoitusta. Tämän harjoituksen tekeminen ei tarvi ohjelmointitaitoja, mutta hyvinkin paljon päättelyä ja uuden omaksumista.

C ja Git, oma esimerkkiprojekti

1. Konfiguroidaan ensin Gitille nimesi ja email-osoitteesi. Nämä tekevät lähinnä logien lukemisesta helpompaa.

   - *git config –global user.name "Oma Nimi"*
   - *git config –global user.email oma.nimi@aalto.fi*

   Git tallentaa nämä kotihakemistoosi /.gitconfig tiedostoon. Voit syöttää tiedot sinne myös tekstieditorilla.

2. Tehdään oma projekti! Luo hakemisto nimeltä "oma" ja siirry sinne: *mkdir oma ; cd oma*

3. Luodaan uusi git-repositorio: *git init*

4. Luodaan uusi lähdekooditiedosto: *nano helloworld.c* Kirjoita seuraavat sisään luomaasi tiedostoon:

   ```
   /* Hello World program */
   #include <stdio.h>
   int main(void) {
           printf("Hello World\n");
           return 0;
   }
   ```

   Varo kirjoitusvirheitä. Ohjelmointikielet ovat yleensäkin täysin armottomia syntaksivirheiden suhteen. C-ohjelmointi ei kuulu tämän harjoituksen tai kurssin piiriin, vaan sitä saatte *ELEC-A1100 C-ohjelmoinnin peruskurssilla.*

5. Käännä kokeeksi ohjelma komennolla: *gcc -o helloworld -Wall helloworld.c* (-o määrittelee ulostulotiedoston ja -Wall pyytää kääntäjältä raportoimaan kaikki varoitukset)

6. Suorita kääntämäsi ohjelma: *./helloworld* ("./" nimen alussa tarkoittaa että tarkoitat juuri tässä kansiossa olevaa ohjelmaa. Muutoin saisit varoituksen siitä että komentoa ei löydy polulta.)

7. Kommitoidaan tekemäsi kooditiedosto projektiin: *git add helloworld.c git commit -m "Uusi kooditiedosto"* (-m "kommenti" on tapa lisätä lyhyt kommentti kommitointiin. Jos sen jättää pois, git käynnistää tekstieditorin jotta voit kirjoittaa logientryn.)

8. Näet tekemäsi muutoksen logista komennolla: *git log*

9. Hienon ohjelmasi tuloste saa kuitenkin välimerkkifetisistit raivostumaan. Korjaa "Hello World" viesti heidän vaatimaan muotoon "Hello, World!" tekstieditorilla.

10. Näet nyt tekemäsi muutoksen komennoilla *git status* ja *git diff*.

11. Kommitoi muutoksesi: *git commit -m "Välimerkit" helloworld.c*

12. Lisätään projektiin Makefile, kääntämisen helpottamiseksi. Luo tekstieditorilla tiedosto "Makefile" ja lisää sinne seuraava sisältö:

```
helloworld: helloworld.c
        gcc -o helloworld -Wall helloworld.c
```

13. Käännä nyt muokkaamasi versio komennolla: *make*

14. Kommitoidaan Makefile: *git add Makefile*
git commit -m "Lisätään Makefile"

15. Oikeastaan kun asiaa vähän harkitaan, välimerkkifetistien vaatimukset eivät saa rajoittaa akateemista vapauttamme. Perutaan heidän vaatima muutos! Käytetään gitin versionhallintavoimaa tämän tekemiseksi. Katso *git log* komennolla "Välimerkit" kommitin tunniste (se heksanumerosotku sanan commit jälkeen) ja suorita komento: *git revert heksanumerosotku* Git avaa tekstieditorin jolla voit tehdä logiviestin, mutta oletus on riittävän hyvä, joten sulje editori.

16. Käännä nyt ohjelma *make* komennolla ja verifioi että se toimii *./helloworld*.

17. Tarkastele miltä gitin tietokanta näyttää komennolla: *gitk*

Makefile on joustava mekanismi kääntää isojakin projekteja. Lisäksi se ei ole sidottu varsinaisesti ohjelmointikieliin, vaan sitä voidaan käyttää moninaisiin asioihin (esimerkiksi kääntää Latex-dokumentteja). Sääntöjen formaatti on:

```
kohde: lähdetiedostot
       komennot joilla lähteistä tehdään kohde
```

Hitusen laajempi esimerkki Makefileistä löytyy sivulta: `http://www.cs.colby.edu/maxwell/courses/tutorials/maketutor/` Ja varsinainen täysi dokumentointi: `https://www.gnu.org/software/make/manual/make.html`
Git on hajautettu versionhallintajärjestelmä. Sillä voi helposti seurata mitä muutoksia koodiin on tehty. Hajautetulla tarkoitetaan että se toimii myös jos useampi eri henkilö tekee muutoksia yhtäaikaa. Jos muutokset tulevat eri osiin koodia, ne yhdistyvät automaattisesti. Hankalammat tapaukset vaativat ihmisapua siivoamisessa.
Git-projektin dokumentointi: `http://git-scm.com/documentation`

Muiden lähdekoodit

Useimmin joudutaan käyttämään jonkun muun kirjoittamaa lähdekoodia, joten tutustutaan miten niitä käännetään.

1. Hae xbill-ohjelman (`http://www.xbill.org/`) lähdekoodit: *wget http://www.xbill.org/download/xbill-2.1.tar.gz*

2. Pura paketti: *tar xvzf xbill-2.1.tar.gz*

3. xbill käyttää autoconf-mekanismia konfigurointiinsa, tämän tunnistaa "configure"-nimisen ohjelman läsnäolosta. Suorita tämä ohjelma: *./configure*

4. Kokeile kääntää ohjelma komennolla: *make*

5. Ikävä kyllä ohjelman lähdekoodissa on bugi, ja sieltä puuttuu vihje linkata libXpm-kirjasto mukaan. Korjaa tämä avaamalla "Makefile" ja lisäämällä LIBS-sanalla alkavalle riville listan loppuun "-lXpm".

6. Käännä ohjelma nyt (*make*) ja suorita se: *./xbill*

Sama uudestaan

7. xbill tulee myös Debianin mukana. Sen lähdekoodit voi ladata komennolla: *apt-get source xbill*
8. siirry xbill-2.1 -hakemistoon ja suorita: *debuild*
9. Jos vilkaiset hakemistoon "debian/patches" löydät Debian-projektin alkuperäiseen jakeluun tekemät muutokset, kuten edellätekemämme libXpm-korjauksen.

Prosessin pitäisi tehdä ylempään hakemistoon deb-paketti jonka voit asentaa (*dpkg -i xbill_2.1-8_amd64.deb*), mutta samalla antaa varoitus että sinulta puuttuu GPG-allekirjoitusavain. (GPG-avaimet ovat osa Debianin pakettien ehjyystarkistuksia, mitä ei tietenkään voida tehdä, kun emme ole paketin virallisia ylläpitäjiä.) Debian-paketointi on hieman isompi kokonaisuus niellä, mutta jos haluatte kurkistaa jäniksenkoloon, on Debian-projektilla massiiviset määrät ohjeita osoitteessa: `https://wiki.debian.org/Packaging`

Git+CMake esimerkki

Esimerkki vähän isomman projektin hakemisesta ja kääntämisestä.
`https://github.com/raceintospace/raceintospace`

1. Kloonaa projekti: *git clone https://github.com/raceintospace/raceintospace.git*
2. Asenna ohjelman tarvitsemat kirjastot: *sudo apt-get install cmake libsdl-dev libboost-dev libpng-dev libjsoncpp-dev libogg-dev libvorbis-dev libtheora-dev libprotobuf-dev protobuf-compiler*
3. Tehdään kansio jossa käännöstyö tehdään (ns. out-of-directory-build):
   mkdir raceintospace-build
   cd raceintospace-build
   cmake ../raceintospace
   make

   Ohjelman pitäisi kääntyä ongelmitta. Kääntäjä tulostaa useita varoituksia, mutta ne eivät varsinaisesti estä toimintaa.
4. Jotta voisimme asentaa ohjelman /usr/local hakemistoon, lisää itsesi joko "staff"-ryhmään (muista että sinun täytyy logata ulos ja sisään jotta ryhmäoikeusmuutokset tulevat voimaan). Vaihtoehtoisesti voit suorittaa seuraavat komennot *sudo*-komennon avulla.
5. Kun ohjelma on käännetty, jos kiinnostaa kokeilla (se on vanha avaruusohjelma-peli), sen voi asentaa komennolla: *make install*
6. Jos haluatte ohjelman mäkeen, se on asentunut kansioon /usr/local/share/raceintospace ja /usr/local/bin/race
   rm -rf /usr/local/bin/raceintospace /usr/local/share/raceintospace