Kesäkurssi 22.5.-27.7.2007

Opettaja: Paavo Nieminen <nieminen@jyu.fi>

Luento 3

24.5.2007

Ohjelmointi ja ohjelmointikieli — suoritus ja konekieli

Aiheet:

- Pääteyhteys ja interaktiivisen shellin käytön idea; bash, tcsh, MS-DOS komentokehote
- Shellin käytöstä: pääpiirteet, mahdollisuudet, vaarat

- Java-ohjelman kääntäminen ja ajaminen komentoriviltä
- C-ohjelman kääntäminen ja ajaminen komentoriviltä
- Tiedostoformaateista ja tiedon tallentamisesta

ENNEN KUIN ALOITAT shellin käytön . . .

Muista vaarat:

- Komennat tietokonetta hyvin suoraan ja tehokkaasti!

→ jos pyydät poistamaan tiedoston, se HÄVIÄÄ!

→ jos ylikirjoitat tiedoston, aiempi sisältö HÄVIÄÄ!
(ja kun jotain hävisi, se EI TULE TAKAISIN!

- Esim. pieni käsky ”rm -rf .” voi tuhota sinulta kaiken.
Ja ”echo > gradu.tex” olisi varsin paha myös!)
- Ns. Roskakoria tai muita tuplavarmistuksia ei ole.

Vältä vaarat:

- Ajattele ennen kuin kirjoitat !
- Ajattele vielä kerran ennen kuin painat enteriä!
- Älä päästä kissaa, lapsia tai vastaavaa näppäimistölle.
- Varmuuskopioi tärkeät tiedot usein. (Joka tapauksessa, käytitpä shelliä tai mitä tahansa!!!)

Kertaus: ENNEN KUIN ALOITAT shellin käytön . . .

Muista:

- Ajattele ennen kuin kirjoitat !
- Ajattele vielä kerran ennen kuin painat enteriä!
- Ymmärrä komentosi etukäteen.
- Jos et ymmärrä, lue manuaalia kunnes ymmärrät.
- Älä päästä kissaa tai vastaavaa näppäimistölle.
- Copy-pastessa on vaaransa; tiedä, mitä pastaat, jos pastaat suoraan interaktiiviseen shelliin!

Älä panikoi!

Älä panikoi:

- Jos olet tarkkana ja tiedät, mitä teet, ei ole hätää
- Nykyiset shellit ja apuohjelmat ovat kehittyneitä, turvallisia ja käyttäjäystävällisiä (eri asia kuin graafinen ja klikkailtava . . .)
- Shellin ja pääteyhteyden osaaminen mahdollistaa paljon käteviä juttuja.
- Edelliset varoitukset olivat vähän sama juttu kuin että älä aja autolla täyttä vauhtia seinään vaikka se olisi helppoa ja vaivatonta kyseisellä välineellä.

Ja sitten vaan käyttämään shelliä!

Mistä on kyse:

- Kirjoitat komennon, ja painat enteriä . . .
- Shell tekee, mitä komensit (useimmiten suorittaa pyytämäsi apuohjelman), ja näet komentosi aiheuttamat tulosteet sitä mukaa kun niitä tulee.
- Näin tietokoneita käytetiin jo silloin kun ei ollut monitoreja (näppis oli kirjoituskonemainen, ja tulosteet hakkautuivat printterillä paperille).
- Näin tietokoneita on varsin kätevä käyttää nykyäänkin.
- ”Vain vähän erilaista” kuin hiirellä klikkaaminen;

Pääteyhteys ja pääteyhteysohjelma

Mistä on kyse:

- Otat yhteyden tietokoneeseen, johon sinulla on käyttäjätunnus
- Etäkone käynnistää sinua varten shellin. Shell-ohjelma toimii siis usein kaukana siitä, missä istut.
- Pääteyhteysohjelma, kuten Secure Shell tai Putty, toimii omalla tietokoneellasi. Se vain välittää Internetin yli näppäinpainallukset ja tulosteet (ja näyttää ne sinulle)
- SSH2-protokollassa tietoliikenne on salattua, minkä etäkoneen serveriohjelma ja oma pääteyhteysohjelmasi (client) hoitavat.

Miten tästä eteenpäin

Demot:

- Demoissa tehdään kädestä-pitäen-harjoitteita, joissa käydään läpi pääteyhteyden ja interaktiivisen shellin käyttö sekä useita tyypillisiä Unixeista tai ainakin Linuxeista löytyviä apuohjelmia
- Jossain vaiheessa siirrytään tallentamaan komentoja peräkkäin tekstitiedostoon eli tekemään shell-skriptejä
- Ihan vähän katsotaan shellien ohjelmointiominaisuuksia eli muuttujia ja kontrollirakenteita.

Shellit hanskassa!

- Pääteyhteys ja shell on nyt esitelty, joten päästään täysipainoisesti käyttämään niitä luentoesimerkeissä!
- Notta kädet rasvaan taas

Ohjelmointi ja ohjelmointikieli

Ohjelmointi on . . .

- osa ohjelmistotyötä, suunnittelua, toteutusta ym.
  mutta rajataan tässä yhteydessä tarkemmin . . .
- Lähdekoodin tekemistä eli tekstin editointia
- Ohjelmointikieliä on tuhansia, esim. Algol, Basic, C,
  C++, C#, D, Eiffel, Fortran, . . . , Java, . . .
- Parhaassa tapauksessa kieli valitaan tilanteen mukaan;
  kukin soveltuu joihinkin ohjelmiin paremmin kuin toisiin
- Vähimmillään tarvitaan tekstieditori
  kuten ConText, Emacs tai Nano
- Yleensä tehdään ohjelmakehitysympäristöllä eli IDEllä
  kuten Eclipse, NetBeans, JBuilder, Delphi, . . .

Suoritus ja konekieli

Ohjelman suoritus on . . .

- Sitä kun prosessori noutaa ja suorittaa konekieltä eli bittijonoksi koodattuja käskyjä peräjälkeen
- Mitä bittejä konekielikäskyissä on, riippuu alla olevasta prosessorista, kuten Intel 8086, 286, 386, 486, Pentium, Itanium, Motorola 68000, 68030, 68040, MIPS, ARM, Cell, Sparc, . . . , . . .
- Suorittava kone voi olla myös virtuaalinen: Java Virtual Machine muuntaa ”lennosta” oman konekielensä käskyjä alla olevan fyysisen prosessorin käskyiksi.
- (Ja toki voidaan ohjelmoida tulkattavalla kielellä, jossa virtuaalikone muuntaa lähdekoodin rivi kerrallaan natiiviksi. Tällaista virtuaalikonetta sanotaan tulkiksi)

Kääntäminen kielissä kuten Java tai C

Kääntäjä on ohjelma, joka:

- Lukee lähdekoodin ja ”ymmärtää” sen, jos kielioppia on noudatettu oikein.
- Tuottaa jonkun prosessoriarkkitehtuurin mukaista konekieltä.
- Java-kielen kääntäjä tekee aina JVM-koodia
- C-kielen kääntäjä tekee esim. Intel 386-koodia

Javan ja C:n vertailua

Java

- Kääntäjä tulostaa JVM:n tavukoodia
- Suoritetaan JVM:n päällä
- Toimii kaikilla laitteistoilla joille on toteutettu JVM ja tarvittavat kirjastot; kirjastoissa oltava mukana natiivia koodia.
- Sama käännös käy aina

C

- Kääntäjä tulostaa tietyn prosessorin koodia, ”natiivi ohjelma”
- Suoritetaan fyysisellä prosessorilla
- Toimii kaikilla laitteistoilla joille on toteutettu C-kääntäjä ja tarvittavat kirjastot
- Pitää kääntää erikseen eri prosessoreille

→ Java on kätevä, koska ohjelmasta tarvitsee jaella vain yhtä käännöstä, joka toimii useissa eri laitteistoissa.

→ C:llä voi tehdä ohjelmia useampaan (melkein mihin tahansa) ympäristöön, koska kääntäjiä on ehditty tehdä monelle prosessorille.

→ C:n kirjastot eri ympäristöille ovat kuitenkin suppeammat, joten Javalla voi tehdä isompia ohjelmia helpommin.

• Ohjelmointi Javalla ja C:llä on syvällisesti erilaista; toinen on oliokieli ja toinen imperatiivinen (rakenteinen).

• Käyttöjärjestelmät (ja vaikkapa Java virtuaalikoneet) tehdään usein C:llä sen yleismaailmallisuuden, tehokkuuden ja laiteläheisyyden vuoksi.

Huomautus!

Tällä kurssilla nähdään C-kieltä hyvin minimaalisesti. Jotta opit sen oikeasti, on syytä lukea paljon lisää. Sudenkuoppia on paljon (kuten Javassa ja kaikissa kielissä). C-kielessä ohjelmoijalla on vastuu kaikesta — ei ole Java-alustaa paapomassa.

Muutamia C-ohjelmoinnin yksityiskohtia
Erityisesti verrattuna Javaan

Olioita ei ole!

Luokkia, perintää ja tiedonpiilotusta ei ole!

Sen sijaan on ”rakenteita” eli struktuureja, ja erikseen aliohjelmia, joilla rakenteita käsitellään.

Millään rakenteella ei itsellään ole toiminnallisuutta eli metodeja.

Tietorakenteet ovat primitiivisempiä: esim. taulukko on se kun varataan tietokoneen muistista peräkkäisiä muistipaikkoja niin paljon että tietty määrä alkioita mahtuu.

Merkkijono on merkeistä koostuva taulukko eikä mitään muuta.

Taulukon tai merkkijonon maksimipituus on aina kiinnitetty; se ei kasva automaattisesti tarpeen mukaan.

Primitiivityyppisten muuttujien varsinainen koko bitteinä riippuu alustasta! (Javassa niiden tulee olla samat kaikissa JVM-toteutuksissa)

Rakenteisiin, primitiivialkioihin ja aliohjelmiin voidaan viitata osoittimen (eli muistipaikan osoitteen!) avulla.

Kukaan tai mikään ei vahdi osoittimilla sohimista: C:llä saa helposti aikaan tietoturva-aukkoja, käyttöjärjestelmäytimen kaatumisia ja muita katastrofeja!

Jos ohjelma viittaa väärään muistipaikkaan ja kirjoittaa sinne kuulumattomia asioita, niin se on voi voi ...

→ C-ohjelmointi vaatii tarkkuutta, koska huolimaton ohjelma ei ole vain ärsyttävä (kuten Java-ohjelman ”kaatuminen” poikkeukseen) vaan mahdollisesti katastrofaalinen (esim. tietokone alttiiksi tietoturvahyökkäykselle)

Mutta mm. juuri näiden ”paapomisominaisuuksien” puuttuminen mahdollistaa käännetyille (hyvin tehdyille) C-ohjelmille valtavan nopeuden ja suoraviivaisuuden.

Ohjelmoijan vastuu on suurempi.

Moduulijako tehdään C:ssä eri tavoin: Toiminnallisesti samankaltaisia aliohjelmia voidaan kirjoittaa samaan tiedostoon moduuliksi, ja moduulille pitää lisäksi tehdä ns. otsikkotiedosto, joka kuuluttaa rajapinnan (eli tarjotut tietorakenteet ja aliohjelmat) moduulin käyttäjille.

(Vertaa Java, jossa moduulijako syntyy siitä, että kukin luokka muodostaa oman kokonaisuutensa omassa lähdekooditiedostossaan).

Kukin moduuli voidaan kääntää erikseen, jolloin se pitää linkittää sen palveluita käyttävän ohjelman kanssa joko käännösvaiheessa tai ajonaikaisesti (dynaaminen linkitys).

C:ssä ennen käännöstä tehdään esikäsittely, jossa lähdekoodia muutetaan tiettyjen esikäsittelijäohjeiden mukaisesti (lähdekoodissa korvataan makroja eri tekstillä ohjeiden mukaisesti; C-lähdekoodissa on siis vähän niinkuin kahta kieltä vuoron perään, makrokieltä esikäsittelijälle ja varsinainen C-kieli kääntäjälle).

Sekavaahan se kaikki on (Ckavaa?), ja mm. sen takia on tehty uusia ohjelmointikieliä sinänsä kaikenpätevän C-kielen kehittämisen jälkeenkin. Täydellistä kieltä ei voi olla; se on varma.

— Siinä ensimmäiset mieleentulevat erot, jotka ainakin pitää ymmärtää ennen kuin voi ymmärtää C-ohjelmointia Java-lähtökohdista.

C-kielen syntaksi on lähellä Javaa, joten siinä ei pitäisi olla yllätyksiä. Avainsanojakin C:ssä on tosi vähän.

C-kielellä ohjelmointia opitaan tarkemmin demossa 2, joka on lähinnä sitä asiaa varten. Asia tarkentuu myös loppukurssin ajan luentoesimerkeissä ja harjoitustyössä.

— Paljon on muutakin erilaista, kuten roskien keruun puuttuminen dynaamisen muistinvarauksen yhteydessä ym. mutta yllä olevilla ehkä pärjää Käyttöjärjestelmät-kurssilla, jossa tarvitsee ymmärtää korkeintaan parikymmentä koodiriviä kerrallaan. Jos haluat tehdä C:llä enemmän myöhemmin, lue joku kirja!