

Käyttöjärjestelmät ja systeemiohjelmointi

Projekti 6: Virtuaalinen Linux-työasema kotiinkuljetuksella

Joonas Liedes

Ohjeet: Tähän dokumenttiin kirjataan seuraavaa (tämä ohje on myös kuvaus sisällysluettelolle)

1) Meta ja Testialustana toimineen koneen tiedot

Suoritin	Intel(R) Core (TM) i5-10600K CPU @ 4.10GHz
Asennettu RAM	16,0 Gt
Järjestelmätyyppi	64-bittinen käyttöjärjestelmä Windows 10 Home v 21H1
Näytönohjain	NVIDIA GeForce RTX 3070 8 Gt Ohjaimen versio: 511.79

1.2 Lähdekoodi

<https://github.com/j00lie/OSprojects>

1.3 Virtualisointi

Tässä kontekstissa virtualisoinnilla tarkoitetaan käytännössä fyysisen tietokoneen resurssien jakamista ylimääräisen abstraktiokerroksen kautta. Näitä resursseja voidaan sitten jakaa loppukäyttäjälle hieman eri tavoin riippuen virtualisoinnin toteutuksesta. Tämä mahdollistaa tietokoneresurssien tehokkaamman käytön, mikäli on tarve päästä hyödyntämään käyttöjärjestelmäkohtaisia ominaisuuksia tai tietyille järjestelmille suunnattuja sovelluksia. Tavoitteena on siis luoda ohjelmistolla toisistaan eristettyjä kokonaisuuksia samojen fyysisten resurssien puitteisiin.

Ohjelmistokerrosta, jolla järjestelmän resursseja abstrahoidaan ja tehdään jaettavaksi, kutsutaan usein Hypervisoriksi. Abstraktio voidaan toteuttaa joko suoraan raudan tasolla (tyyppi 1) tai vaihtoehtoisesti käyttöjärjestelmän kautta (tyyppi 2). Suoraan raudan tasolla tehty abstraktio on tyypillinen ratkaisu esimerkiksi palvelimissa. Sen etuina on tehokkuus ja turvallisuus, sillä abstraktion ja raudan välissä ei ole ylimääräisiä kerroksia. Virtuaalikoneita voidaan kuitenkin joutua käyttämään toisen koneen kautta. Tässä yhteydessä loppukäyttäjille suunnattu virtualisointi tapahtuukin useammin siten, että abstraktiokerros sijoittuu toisen käyttöjärjestelmän päälle sovellusohjelmana. Etuna tässä tavassa on helppokäyttöisyys ja nopea pääsy vaihtoehtoiseen käyttöjärjestelmään. Toisen käyttöjärjestelmän päällä tapahtuva

ajo tuo mukanaan väistämättä kuitenkin kompromissit tehokkuuden suhteen, kun operoidaan rajallisilla resursseilla.

Jaottelua voidaan tehdä myös virtualisoinnin tason perusteella. Täydestä virtualisoinnista puhuttaessa tarkoitetaan tilannetta, jossa muokkaamaton vieraskäyttöjärjestelmä ajetaan virtuaalikoneen simuloimalla laiteresurssilla. Tämän kaltainen ns. täysi virtualisointi voidaan jaotella edelleen toteutuksiin, jossa virtualisointi perustuu ohjelmistototeutukseen tai vaihtoehtoisesti suoraan prosessoriarkkitehtuuriin perustuviin ratkaisuihin.

Ohjelmistoavusteiset täydelliset virtualisoinnit käyttävät virtualisointiin ajonaikaista kääntämistä, kun taas prosessoriarkkitehtuuriin perustuvat ratkaisut mahdollistavat sovellusohjelman suorittamien tiettyjen käskyjen suoraan prosessorilla. Ohjelmistoavusteiset täysvirtualisoinnit toimivat edellä kuvatulla tyypin 2 hypervisorilla, jolloin siis raudan ja abstraktion välissä on isäntänä toinen käyttöjärjestelmä. Prosessoriarkkitehtuuripohjaiset ratkaisut voivat hyödyntää sekä tyypin 1, että tyypin 2 hypervisoreita. Käytössä on joskus myös ns. paravirtualisointia, mikä edellyttää muutoksia vieraskäyttöjärjestelmässä toimiakseen. Näissä tilanteissa hypervisor asennetaan raudalle ja tähän päälle asennetaan muokattu versio käyttöjärjestelmästä. Joissain tilanteissa on myös hyödyllistä käyttää yhdistelmiä täys- ja paravirtualisoinnista tehojen optimoimiseksi.

Yleisesti käytössä oleva metodi on myös käyttöjärjestelmätasolla tehtävä virtualisointi, joissa raudan päällä on yksittäinen käyttöjärjestelmä, jonka päälle on mahdollista luoda toisistaan erillä olevia säiliöitä. Olennaisena erona aiempiin metodeihin on siis virtualisointi käyttöjärjestelmän tasolla, mikä mahdollistaa isoloidut prosessit verrattuna isoituihin resursseihin. Koska toteutus perustuu yksittäiseen kerneliin, on menetelmä varsin resurssitehokas.

Tässä projektissa hyödynnetty VMware workstation player on toteutettu sovellusohjelmana ja edustaa ns. isännöityä virtualisointia. Käytössä on siis tyypin 2 hypervisor, joka on rautaan yhteydessä isännän käyttöjärjestelmän kautta. VMware player toteuttaa täydellisen virtualisoinnin hyödyntämällä ajonaikaista binäärikääntöä yhdessä suoraan prosessorilla suoritettavien ohjeiden kanssa. Ajonaikaista kääntöä käytetään sensitiivisten ohjeiden ajoon, siinä missä suoraan prosessorilla ajetaan ei niin kriittiset käskyt.

2) Kokeillut tehtävät ja miniprojektit (jokainen omaksi aliluvuksi X)

2.X.1 Mitkä tehtävät toteutettiin virtuaalikoneen avulla

2.X.2 Muutama kuvankaappaus työskentelystä

2.X.3 Yhteenveto siitä mitä teitte, miten toimi

2.0.1)

Asensin VMwaren Moodlen ohjeistuksen mukaisesti ongelmitta. Asennuksessa oli huomioitava koneen oma virtualisointi (Hyper-V) ja klikattava WHP-yhteensopivuuden mahdollistava kohta, olettaen että jotain tällaisia ominaisuuksia käytössä (itsellä esimerkiksi WSL2 ym.). Tätä ei ohjeissa sen enempää kuitenkaan käyty läpi, toki tämä oli asennusohjelman yhteydessä avattu. Tämän jälkeen latasin Ubuntu virtuaalikoneen kuvan ja käynnistin koneen WMwarella ohjeiden mukaisesti ongelmitta. Ohjeista poiketen konetta käynnistäessä ei kuitenkaan tullut kehotusta asentaa WMwaren Linux työkaluja.

Kaikkien alla lueteltujen tehtävien tekoon käytin virtuaaliympäristön sovelluksiin asennettua Visual Studio Codea ja sen integroitua bash-terminaalia.

2.1.1)

Viikon 8 harjoitustehtävä 1.

2.1.2)

```
student@ubuntu:~/Documents/Exercises/Viikko_8$ ls
Luento  midsummer_nights_dream.txt  palautus  T4_esitys_esimerkkikoodi.pdf  tehtavat
```

```
student@ubuntu:~/Documents/Exercises/Viikko_8$ mkdir testi
student@ubuntu:~/Documents/Exercises/Viikko_8$ ls
Luento  midsummer_nights_dream.txt  palautus  T4_esitys_esimerkkikoodi.pdf  tehtavat  testi
```

```
student@ubuntu:~/Documents/Exercises/Viikko_8$ rmdir testi
student@ubuntu:~/Documents/Exercises/Viikko_8$ ls
Luento  midsummer_nights_dream.txt  palautus  T4_esitys_esimerkkikoodi.pdf  tehtavat
```

2.1.3)

Tehtävässä oli tarkoituksena käydä läpi Unixin peruskomentoja. Tehtävän tekoon käytin itselle tutuinta VS Codea editorina, sillä pidän sen integroidusta terminaalista ja muista tutuista ominaisuuksista. Esimerkkikuvissa listataan kyseisen kansion sisältöä, luodaan uusi kansio ja sitten poistetaan kyseinen kansio. Tehtävänannossa lueteltujen perustoiminnallisuuksien toteutuksessa ei ollut ongelmia.

2.2.1)

Viikon 8 harjoitustehtävä 3.

2.2.2)

```
student@ubuntu:~/Documents/Exercises/Viikko_8$ echo $SHELL
/bin/bash
student@ubuntu:~/Documents/Exercises/Viikko_8$ echo \$SHELL
$SHELL
student@ubuntu:~/Documents/Exercises/Viikko_8$ echo \\$SHELL
\\bin/bash
student@ubuntu:~/Documents/Exercises/Viikko_8$ echo \\$SHELL
\\$SHELL
student@ubuntu:~/Documents/Exercises/Viikko_8$ echo '\$'
\$
student@ubuntu:~/Documents/Exercises/Viikko_8$ echo \\
/
student@ubuntu:~/Documents/Exercises/Viikko_8$ echo \\\
//
student@ubuntu:~/Documents/Exercises/Viikko_8$ echo "\$"
$
student@ubuntu:~/Documents/Exercises/Viikko_8$ echo '"$'
"$
student@ubuntu:~/Documents/Exercises/Viikko_8$ echo "$"
$
student@ubuntu:~/Documents/Exercises/Viikko_8$
```

2.2.3)

Tehtävässä oli tarkoituksena käydä läpi koodinvaihtomerkin (escape character) vaikutuksia tulosteeseen. Bash-komentotulkin toiminta VS Coden sisällä oli odotettua eikä virheitä tapahtunut.

2.3.1)

Viikon 8 harjoitustehtävä 4.

2.3.2)

The image shows a Visual Studio Code window titled "script.bash - Viikko_8 - Visual Studio Code". The Explorer sidebar on the left shows a project named "VIKKO_8" with files: "Luento", "midsummer_nights...", "palautus", "script.bash" (selected), "T4_esitys_esimerkki...", and "tehtavat". The script "script.bash" contains the following code:

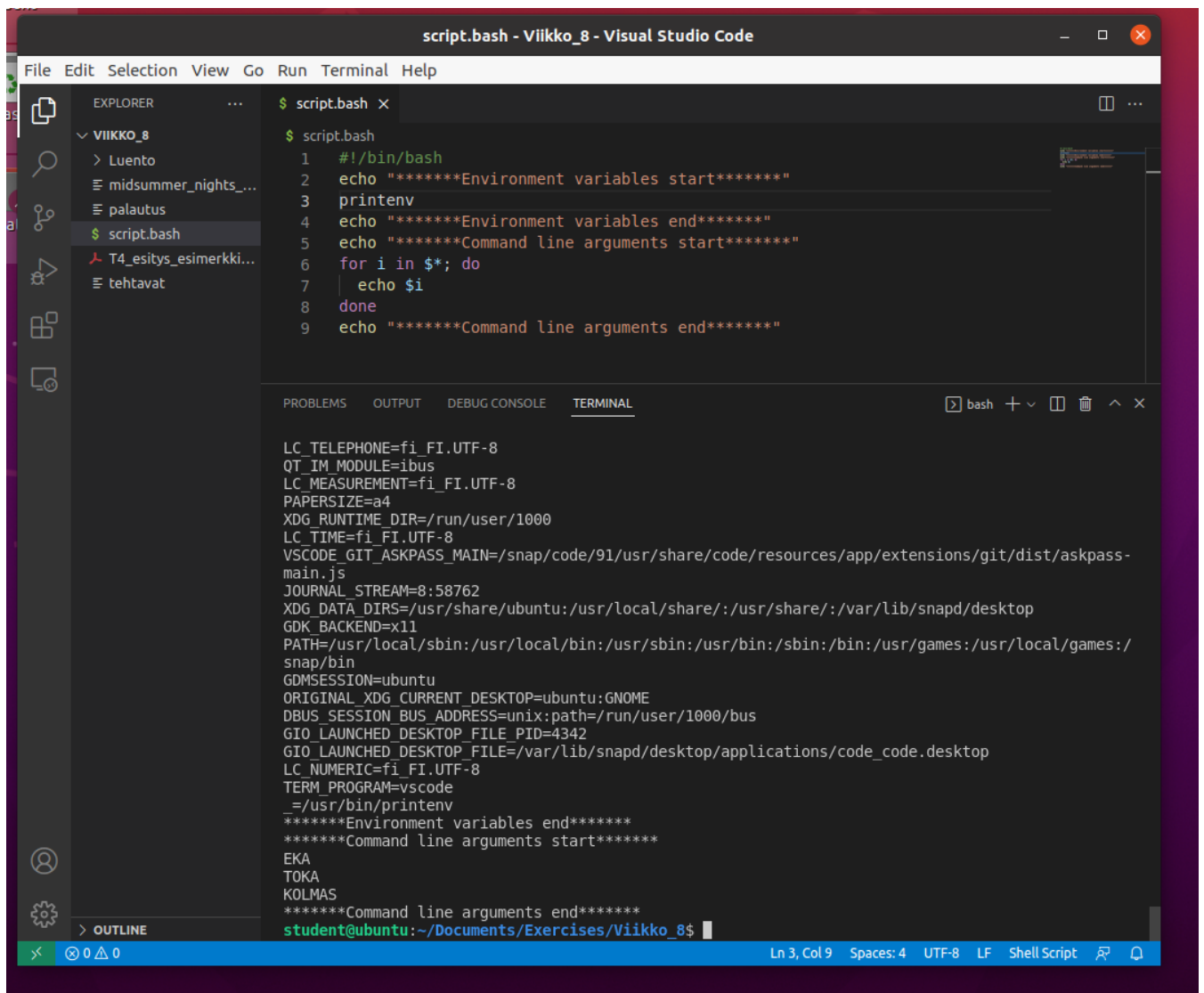
```
$ script.bash
1  #!/bin/bash
2  echo "hello world t. $USER"
3  echo "Files in working directory $PWD"
4  ls # list files
5  echo "We just used variables $USER and $PWD"
```

The TERMINAL panel at the bottom shows the execution of the script:

```
Files in working directory /home/student/Documents/Exercises/Viikko_8
student@ubuntu:~/Documents/Exercises/Viikko_8$ ./script.bash
hello world t. student
Files in working directory /home/student/Documents/Exercises/Viikko_8
Luento      palautus      T4_esitys_esimerkkikoodi.pdf
midsummer_nights_dream.txt  script.bash  tehtavat
'We just used variables student and /home/student/Documents/Exercises/Viikko_8'
student@ubuntu:~/Documents/Exercises/Viikko_8$
```

The status bar at the bottom indicates "Ln 5, Col 45", "Spaces: 4", "UTF-8", "LF", and "Shell Script".

A close-up of the terminal output shows the command `student@ubuntu:~/Documents/Exercises/Viikko_8$./script.bash` and its output: `EKA TOKA KOLMAS`.



The screenshot shows the Visual Studio Code interface with a file explorer on the left containing a directory named 'VIKKO_8' with files 'Luento', 'midsummer_nights...', 'palautus', 'script.bash', 'T4_esitys_esimerkki...', and 'tehtavat'. The 'script.bash' file is open in the editor, showing a script that prints environment variables and command line arguments. The terminal at the bottom shows the output of running the script, displaying various system environment variables and the command line arguments 'EKA', 'TOKA', and 'KOLMAS'.

```
script.bash - Viikko_8 - Visual Studio Code
File Edit Selection View Go Run Terminal Help

EXPLORER
  VIKKO_8
    Luento
    midsummer_nights...
    palautus
    script.bash
    T4_esitys_esimerkki...
    tehtavat

$ script.bash
$ script.bash
1  #!/bin/bash
2  echo "*****Environment variables start*****"
3  printenv
4  echo "*****Environment variables end*****"
5  echo "*****Command line arguments start*****"
6  for i in $*; do
7    echo $i
8  done
9  echo "*****Command line arguments end*****"

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
bash
LC_TELEPHONE=fi_FI.UTF-8
QT_IM_MODULE=ibus
LC_MEASUREMENT=fi_FI.UTF-8
PAPERSIZE=a4
XDG_RUNTIME_DIR=/run/user/1000
LC_TIME=fi_FI.UTF-8
VSCODE_GIT_ASKPASS_MAIN=/snap/code/91/usr/share/code/resources/app/extensions/git/dist/askpass-main.js
JOURNAL_STREAM=8:58762
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share/:/usr/share/:/var/lib/snapd/desktop
GDK_BACKEND=x11
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
GDMSESSION=ubuntu
ORIGINAL_XDG_CURRENT_DESKTOP=ubuntu:GNOME
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
GIO_LAUNCHED_DESKTOP_FILE_PID=4342
GIO_LAUNCHED_DESKTOP_FILE=/var/lib/snapd/desktop/applications/code_code.desktop
LC_NUMERIC=fi_FI.UTF-8
TERM_PROGRAM=vscode
_/usr/bin/printenv
*****Environment variables end*****
*****Command line arguments start*****
EKA
TOKA
KOLMAS
*****Command line arguments end*****
student@ubuntu:~/Documents/Exercises/Viikko_8$
```

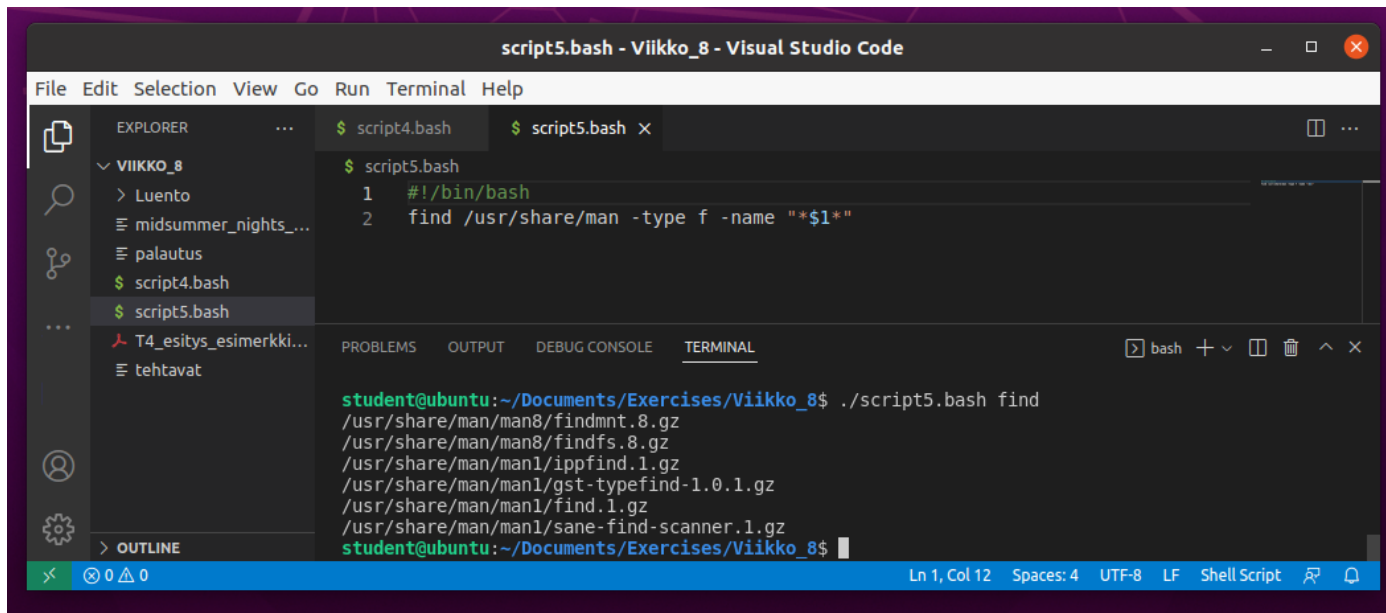
2.3.3)

Tehtävässä käytiin ensin läpi luentokalvojen bash-skriptauksen esimerkkejä. Ensimmäisessä kuvakaappauksessa yksinkertainen koodi, joka tulostaa käyttäjän sekä sen hetkisen kansion. Tehtävän toisessa osassa oli tarkoituksena tehdä skripti, joka tulostaa omat parametrinsa. Skriptin ajo toisessa kuvakaappauksessa, jossa 3 kpl komentoriviargumentteja. Näillä ei tässä sinänsä vaikutusta ohjelman toimintaan pois lukien niiden tulostus. Lisäksi on tulostettu tehtävänannon mukaisesti ympäristömuuttujat kokonaisuudessaan käyttäen printenv-komentoa. Tehtävä suoritettiin aiempien esimerkkien tapaan VS Codea ja integroitua bash-komentotulkkia hyödyntäen. Suorituksessa ei ilmennyt ongelmia.

2.4.1)

Viikon 8 harjoitustehtävä 5.

2.4.2)



```
script5.bash - Viikko_8 - Visual Studio Code
File Edit Selection View Go Run Terminal Help

EXPLORER
  VIKKO_8
    Luento
    midsummer_nights_...
    palautus
    script4.bash
    script5.bash
    T4_esitys_esimerkki...
    tehtavat

OUTLINE
  > OUTLINE

script5.bash
1  #!/bin/bash
2  find /usr/share/man -type f -name "$1"

TERMINAL
student@ubuntu:~/Documents/Exercises/Viikko_8$ ./script5.bash find
/usr/share/man/man8/findmnt.8.gz
/usr/share/man/man8/findfs.8.gz
/usr/share/man/man1/ippfind.1.gz
/usr/share/man/man1/gst-typefind-1.0.1.gz
/usr/share/man/man1/find.1.gz
/usr/share/man/man1/sane-find-scanner.1.gz
student@ubuntu:~/Documents/Exercises/Viikko_8$
```

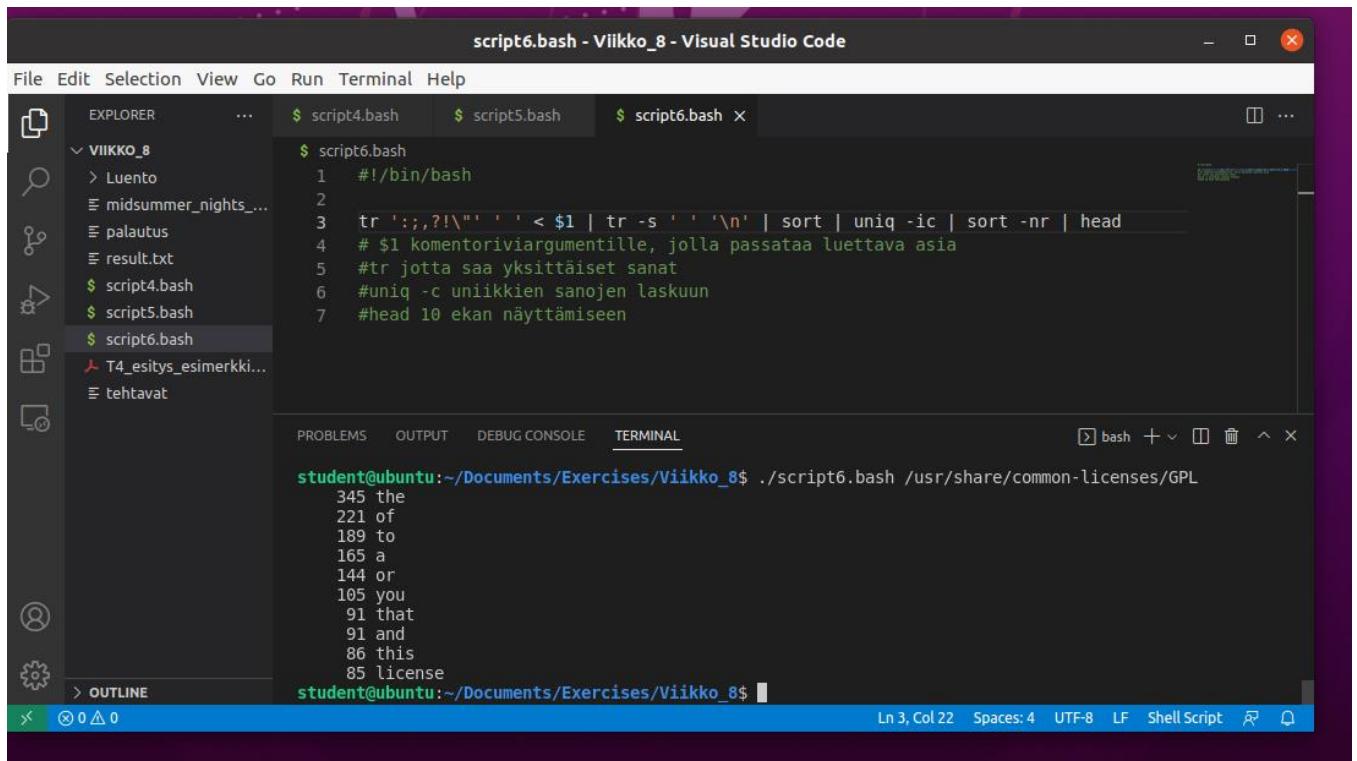
2.4.3)

Tehtävän tarkoituksena oli tehdä skripti, jolla voi hakea komentoriviargumentin määrittämää sivua tai sivuja ohjekirjasta. Yllä \$1 viittaa komentoriviargumenttiin, jolla passataan hakusana skriptille. Tässä tapauksessa haettiin kaikki sivut, joissa jollain tapaa 'find' mainittuna. Suoritus tapahtui aiempaan tapaan VS Codea ja sen bash-terminaalia hyödyntäen, eikä tähän liittynyt ongelmia.

2.5.1)

Viikon 8 harjoitustehtävä 6.

2.5.2)



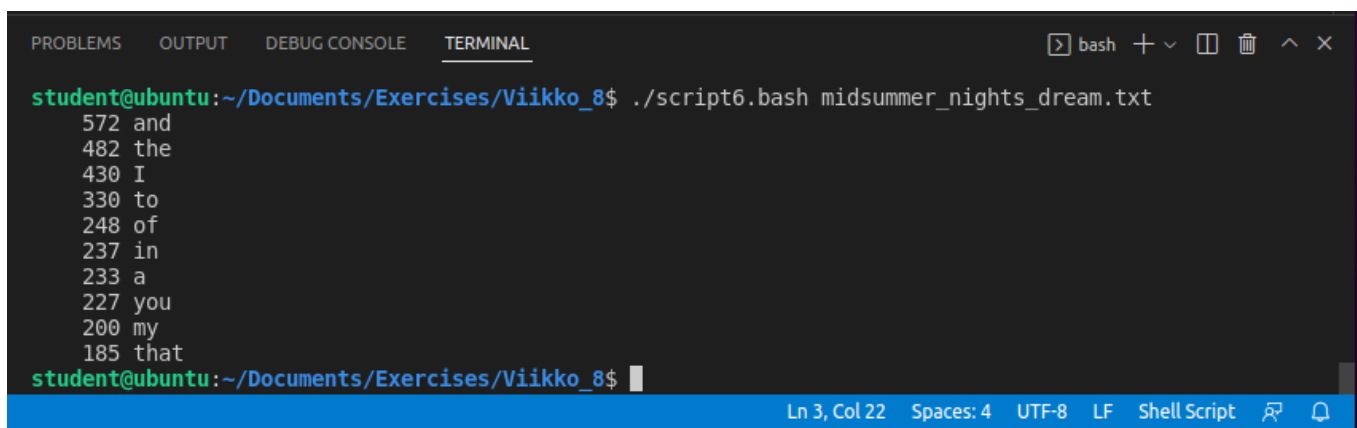
```
script6.bash - Viikko_8 - Visual Studio Code
File Edit Selection View Go Run Terminal Help

EXPLORER
  VIKKO_8
    > Luento
    midsummer_nights_...
    palautus
    result.txt
    script4.bash
    script5.bash
    script6.bash
    T4_esitys_esimerkki...
    tehtavat

OUTLINE

$ script6.bash
1  #!/bin/bash
2
3  tr '.,;?!\"' ' ' < $1 | tr -s ' ' '\n' | sort | uniq -ic | sort -nr | head
4  # $1 komentoriviargumentille, jolla passataa luettava asia
5  #tr jotta saa yksittäiset sanat
6  #uniq -c uniikkien sanojen laskuun
7  #head 10 ekan näyttämiseen

student@ubuntu:~/Documents/Exercises/Viikko_8$ ./script6.bash /usr/share/common-licenses/GPL
345 the
221 of
189 to
165 a
144 or
105 you
91 that
91 and
86 this
85 license
student@ubuntu:~/Documents/Exercises/Viikko_8$
```



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
bash + - [ ] [ ] ^ x

student@ubuntu:~/Documents/Exercises/Viikko_8$ ./script6.bash midsummer_nights_dream.txt
572 and
482 the
430 I
330 to
248 of
237 in
233 a
227 you
200 my
185 that
student@ubuntu:~/Documents/Exercises/Viikko_8$
```

2.5.3)

Tehtävänanto oli kirjoittaa skripti selvittämään tekstisyytteen 10 yleisintä sanaa ja niiden lukumäärät. Skriptin teknisessä toteutuksessa tai ajossa ei ollut virtuaalikoneeseen tai ympäristöön liittyviä ongelmia, vaikkakin GPL-tekstin kohdalla sanojen lukumäärät joiltain osin poikkeavat esimerkivastauksesta. Tästä syyttäminen on kuitenkin todennäköisesti allekirjoittaneen skriptiä, eikä niinkään alustaa.

2.6.1)

Viikon 9 harjoitustehtävä 1.

2.6.2)

```
C t1.c x C t1_func.c M makefile C t1_func.h
C t1.c > main()
1
2 #include "t1_func.h"
3
4 int main(){
5     calculate(1,2);
6     return 0;
7 }
```

```
C t1.c C t1_func.c x M makefile C t1_func.h
C t1_func.c > ...
1 #include "t1_func.h"
2
3 void calculate(int num1, int num2){
4     int result = num1 + num2;
5     printf("Result of sum: %d\n", result);
6 }
```

```
C t1.c C t1_func.c M makefile x C t1_func.h
M makefile
1 t1: t1.o t1_func.o
2 gcc t1.o t1_func.o -o t1
3 t1.o: t1.c t1_func.h
4 gcc -c t1.c
5 t1_func.o: t1_func.c t1_func.h
6 gcc -c t1_func.c
```

```
student@ubuntu:~/Documents/Exercises/Viikko_9$ make
gcc -c t1.c
gcc -c t1_func.c
gcc t1.o t1_func.o -o t1
```

```
student@ubuntu:~/Documents/Exercises/Viikko_9$ ./t1
Result of sum: 3
```

2.6.3)

Tehtävässä harjoiteltiin makefilen käyttöä ja tarkasteltiin tiedostoihin tehtävien muutoksen vaikutuksia kääntäjän toimintaan. Muutokset otsikkotiedostoon t1_func.h vaikuttavat molempiin lähdekooditiedostoihin, sillä ne ovat molemmat riippuvaisia tästä. Sen sijaan muutokset lähdekooditiedostoihin vaikuttavat eivät johda molempien tiedoston uudelleen kääntämiseen. Aiempien tehtävien kaltaisesti tämänkin tehtävän tein Vs Codea ja integroitua bash-terminaalia käyttäen. Mitään ongelmia ei tehtävän tekoon liittynyt.

2.7.1)

Viikon 9 harjoitustehtävä 2.

2.7.2)

```
C t2.c  X
C t2.c > main(int, char *[], char *[])
1  #include <stdio.h>
2
3  int main(int argc, char *argv[], char * envp[]){
4
5      printf("***Komentorivin parametrit alkaa***\n");
6      for (int j = 0; argv[j] != NULL; j++){
7          printf("\n%s\n",argv[j]);
8      }
9      printf("\n***Komentorivin parametrit loppuu***\n");
10     printf("***Ympäristomuuttujat alkaa***\n");
11     for (int i = 0; envp[i] != NULL; i++)
12         printf("\n%s", envp[i]);
13
14     printf("\n***Ympäristomuuttujat loppuu***\n");
15     return 0;
16 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
student@ubuntu:~/Documents/Exercises/Viikko_9$ gcc t2.c -o t2 -std=c99
student@ubuntu:~/Documents/Exercises/Viikko_9$ ./t2 TOKA KOLMAS
***Komentorivin parametrit alkaa***

./t2

TOKA

KOLMAS

***Komentorivin parametrit loppuu***
***Ympäristomuuttujat alkaa***

SHELL=/bin/bash
SESSION_MANAGER=local/ubuntu:~/tmp/.ICE-unix/1621,unix/ubuntu:~/tmp/.ICE-unix/1621
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
TERM_PROGRAM_VERSION=1.65.2
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
APPLICATION_INSIGHTS_NO_DIAGNOSTIC_CHANNEL=true
LC_ADDRESS=fi_FI.UTF-8
GNOME_SHELL_SESSION_MODE=ubuntu
LC_NAME=fi_FI.UTF-8
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
LC_MONETARY=fi_FI.UTF-8
SSH_AGENT_PID=1586
BAMF_DESKTOP_FILE_HINT=/var/lib/snapd/desktop/applications/code_code.desktop
GTK_MODULES=gail:atk-bridge
PWD=/home/student/Documents/Exercises/Viikko_9
GSETTINGS_SCHEMA_DIR=/snap/code/91/usr/share/glib-2.0/schemas
XDG_SESSION_DESKTOP=ubuntu
```

2.7.3)

Tehtävänanto oli kirjoittaa C-ohjelma, joka tulostaa komentoriviargumenttinsa ja ympäristömuuttujat. Asiaa googlella tutkittuakin selvisi, että gcc-kääntäjä tukee main-metodissa kolmatta parametria, jolla pääsee käsiksi ympäristömuuttujiin. Tämän jälkeen em. Tulostus onkin varsin suoraviivaista. Tehtävän teko tapahtui käyttäen Vs Codea ja integroitua bash-terminaalia. Ongelmia virtuaaliympäristöön liittyen ei esiintynyt.

2.8.1)

Harjoitustyöprojekti 1. Toteutin projektin alun perin WSL 2:n avulla, mutta käytin testaamiseen myös tämän projektin virtuaalikonetta sillä halusin varmistua toimivuudesta 'puhtaassa' Linux-ympäristössä.

2.8.2)

```
student@ubuntu:~/Documents/Exercises/Harjoitustyoprojekti/p2$ gcc test.c -o reverse -Wall -Werror -std=c99
student@ubuntu:~/Documents/Exercises/Harjoitustyoprojekti/p2$ ./reverse test.txt
file
test

a

is

this

student@ubuntu:~/Documents/Exercises/Harjoitustyoprojekti/p2$ valgrind ./reverse test.txt
==3525== Memcheck, a memory error detector
==3525== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==3525== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==3525== Command: ./reverse test.txt
==3525==
file
test

a

is

this

==3525==
==3525== HEAP SUMMARY:
==3525==   in use at exit: 0 bytes in 0 blocks
==3525==   total heap usage: 14 allocs, 14 frees, 5,816 bytes allocated
==3525==
==3525== All heap blocks were freed -- no leaks are possible
==3525==
==3525== For lists of detected and suppressed errors, rerun with: -s
==3525== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

2.8.3)

Varsinaisen koodauksen tein WSL2:n avulla, mutta testauksessa käytin myös tätä Ubuntu-ympäristöä. Latasin lähdekoodin Githubin kautta, käänsin ohjelman ja ajoin testiajot. Lisäksi ajoin ohjelman valgrindin läpi muistivuotojen varalta. Ympäristöön liittyviä ongelmia ei tullut testauksen aikana esiin. Tarkempi dokumentaatio saatavilla dokumentaation alussa annetun Github-linkin kautta.

2.9.1)

Harjoitustyöprojekti 2. Tein kaikki kolme (my-cat, my-grep, my-zip&unzip) tämän projektin osatyötä annetulla Ubuntu-imagella.

2.9.2)

```
student@ubuntu:~/Documents/Exercises/Harjoitustyoprojekti/p2$ gcc my-cat.c -o my-cat -Wall -Werror
student@ubuntu:~/Documents/Exercises/Harjoitustyoprojekti/p2$ ./my-cat test.txt
this
is
a
test
file
student@ubuntu:~/Documents/Exercises/Harjoitustyoprojekti/p2$ ./my-cat test.txt test2.txt
this
is
a
test
file
file
Second
test file
begins here
student@ubuntu:~/Documents/Exercises/Harjoitustyoprojekti/p2$ ./my-cat
student@ubuntu:~/Documents/Exercises/Harjoitustyoprojekti/p2$
```

```
student@ubuntu:~/Documents/Exercises/Harjoitustyoprojekti/p2$ valgrind ./my-cat test.txt test2.txt
==3931== Memcheck, a memory error detector
==3931== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==3931== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==3931== Command: ./my-cat test.txt test2.txt
==3931==
this
is
a
test
file
Second
test file
begins here
==3931==
==3931== HEAP SUMMARY:
==3931==    in use at exit: 0 bytes in 0 blocks
==3931==   total heap usage: 5 allocs, 5 frees, 10,160 bytes allocated
==3931==
==3931== All heap blocks were freed -- no leaks are possible
==3931==
==3931== For lists of detected and suppressed errors, rerun with: -s
==3931== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
student@ubuntu:~/Documents/Exercises/Harjoitustyoprojekti/p2$
```


[illegible]

Laitteiston tiedot	Samat kuin dokumentaation alussa kuvattu
Kuvaus virheestä	Hiiren scrollaustoiminto ei toimi välillä lainkaan, ja toimiessaan todella takkuileva. Tapahtuu sekä verkkosivuilla, että koodieditorissa (VSCode).

Virheen toistaminen	Virhe toistuu jatkuvasti hiirellä scrollatessa.
Virheilmoitus	Ei virheilmoitusta
Ratkaisu	Kun VMwaren ajaa järjestelmänvalvojana ongelmaa ei esiinny yhtä häiritsevästi, joskaan scrollaustoiminta ei aivan yhtä sujuvaa ole kuin isäntäkoneen puolella.