

# Käyttöjärjestelmät ja systemiohjelmointi

Kurssin harjoitustyöraportti

Otso Weckström 0547090

Teemu Kauranen 051009

# Sisällysluettelo

<b>Sisällysluettelo</b>	<b>1</b>
<b>Intro</b>	<b>2</b>
<b>Projekti 1</b>	<b>2</b>
<b>Projekti 2</b>	<b>4</b>
<b>Projekti 4</b>	<b>6</b>
<b>Tehtävien GitHub Repositorio</b>	<b>10</b>

# Intro

Tehtäville on perus ohjeet alhaalle linkatun repositorion Readme-tiedostossa!

## Projekti 1

<https://github.com/TepaXD/HarkkaRepo/tree/master/Projekti1>

Tehtävässä toteutettiin koodi joka muodostaa annetusta syötteestä tiedoston, jossa syötteen rivit ovat vastakkaisessa järjestyksessä.

Esim.

Input.txt ->                      Output.txt ->

Haloo	12485
Lahoo	Kaloja
Kaloja	Lahoo
12485	Haloo

Ohjelma toimii kolmella eri tapaa annetuista komentorivi argumenteista riippuen;

Nolla argumenttia:

```
Teemu-MacBook-Pro-5:projekti1 TeemuK$ ./reverse  
give input to reverse: tämä teksti peilaantuu  
  
peilaantuu teksti tämä  
Teemu-MacBook-Pro-5:projekti1 TeemuK$ █
```

- Syöte annetaan terminaalissa, ja se tulostuu väärinpäin terminaaliin.

Yhdellä argumentilla:

- Annetusta tiedostosta (input.txt esimerkissä) luetaan syöte ja tulostetaan ruudulle.

```
5555555
4444444
3333333
2222222
1111111
```

```
Teemu-MacBook-Pro-5:projekti1 TeemuK$ ./reverse input.txt
5555555
4444444
3333333
2222222
1111111
Teemu-MacBook-Pro-5:projekti1 TeemuK$
```

Kahdella argumentilla:

- Ohjelma lukee syötteen ensimmäisestä tiedostosta ja tallentaa toiseen.

```
Teemu-MacBook-Pro-5:projekti1 TeemuK$ ./reverse input2.txt output2.txt
Teemu-MacBook-Pro-5:projekti1 TeemuK$
```

aaaaaa	fff
bbbbbbbbbbbb	eeeeee
cc	dddddddddddddddd
dddddddddddddddd	cc
eeeeee	bbbbbbbbbbbb
fff	aaaaaa

# Projekti 2

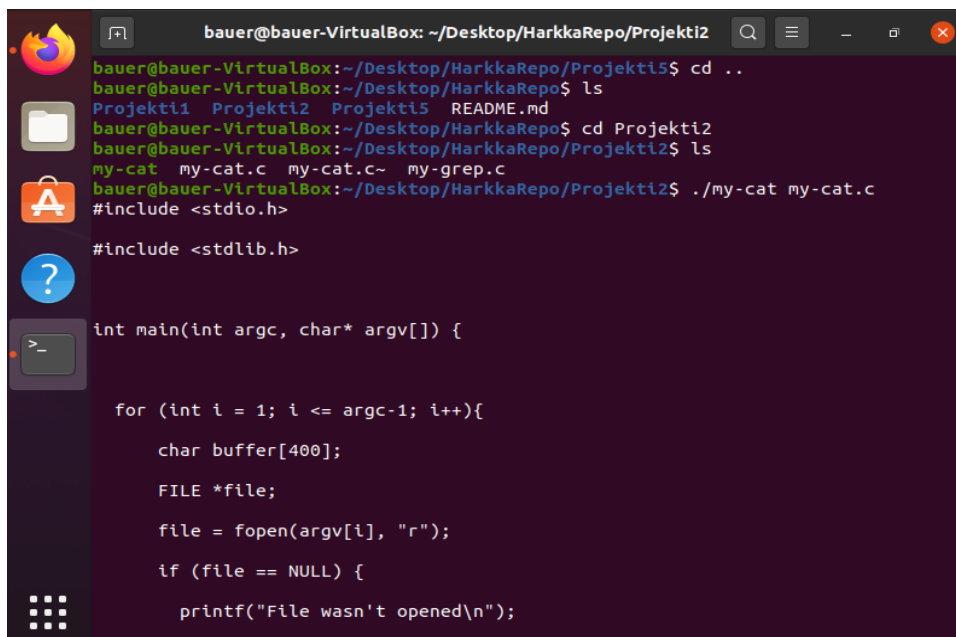
Repo: <https://github.com/TepaXD/HarkkaRepo/tree/master/Projekti2>

Tehtävä koostuu neljästä pienemmästä tehtävästä, jotka kaikki löytyy saman kansion alta.

## my-cat:

Ensimmäinen 2 projektin pienohjelmista, my-cat, lukee C-kielisen tiedoston ja tulostaa terminaaliin sisällön.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(int argc, char* argv[]) {
5
6     for (int i = 1; i <= argc-1; i++){
7         char buffer[400]; //Reserving memory for the contents
8         FILE *file;
9         file = fopen(argv[i], "r"); //Getting the filename from the commandline argument and opening file
10        if (file == NULL) { //Error handling incase the file doesn't work
11            printf("File wasn't opened\n");
12            exit(1);
13        }
14
15        while(fgets(buffer, 399, file) != NULL) { //Print the contents of "buffer"
16            printf("%s\n", buffer);
17        }
18        fclose(file); //close file
19    }
20    return 0;
21 }
```



The screenshot shows a terminal window titled "bauer@bauer-VirtualBox: ~/Desktop/HarkkaRepo/Projekti2". The user navigates to the directory and lists files, showing "Projekti1", "Projekti2", "Projekti5", and "README.md". They then run the command `./my-cat my-cat.c`, which outputs the C code for the my-cat program, matching the code shown in the previous block.

```
bauer@bauer-VirtualBox: ~/Desktop/HarkkaRepo/Projekti2
bauer@bauer-VirtualBox:~/Desktop/HarkkaRepo$ cd ..
bauer@bauer-VirtualBox:~/Desktop/HarkkaRepo$ ls
Projekti1  Projekti2  Projekti5  README.md
bauer@bauer-VirtualBox:~/Desktop/HarkkaRepo$ cd Projekti2
bauer@bauer-VirtualBox:~/Desktop/HarkkaRepo/Projekti2$ ls
my-cat  my-cat.c  my-cat.c~  my-grep.c
bauer@bauer-VirtualBox:~/Desktop/HarkkaRepo/Projekti2$ ./my-cat my-cat.c
#include <stdio.h>

#include <stdlib.h>

int main(int argc, char* argv[]) {

    for (int i = 1; i <= argc-1; i++){

        char buffer[400];

        FILE *file;

        file = fopen(argv[i], "r");

        if (file == NULL) {

            printf("File wasn't opened\n");
```

## my-grep:

Toinen pienohjelma, my-grep, hakee tiedostoista tai käyttäjän syötteestä annetun hakutermiin mukaisia sanoja.

Input.txt tiedosto:

```
testi
tämä on oikea testi
tätä ei tulosteta
mutta tämä testi tulostuu!
```

Hakusanana kirjain "t":

```
[Teemu-MacBook-Pro-5:grep TeemuK$ ./my-grep t input.txt
testi

tämä on oikea testi

tätä ei tulosteta

mutta tämä testi tulostuu!
Teemu-MacBook-Pro-5:grep TeemuK$
```

Hakusanana sana "testi":

```
[Teemu-MacBook-Pro-5:grep TeemuK$ ./my-grep testi input.txt
testi

tämä on oikea testi

mutta tämä testi tulostuu!
Teemu-MacBook-Pro-5:grep TeemuK$
```

## my-zip:

Kolmas pienohjelma, joka pakkaa tekstitiedoston sisältöä binäärimuotoon ja pakkaa sen toiseen tiedostoon. Esimerkiksi aaabbbccd -> 3a3b2c1d. Jos ohjelmalle syöttää enemmän kuin yhden tekstitiedoston, ne yhdistetään yhdeksi.

file.txt input:

```
aaaaaaaaaabb
```

file2.txt input:

```
ccccccdd
```

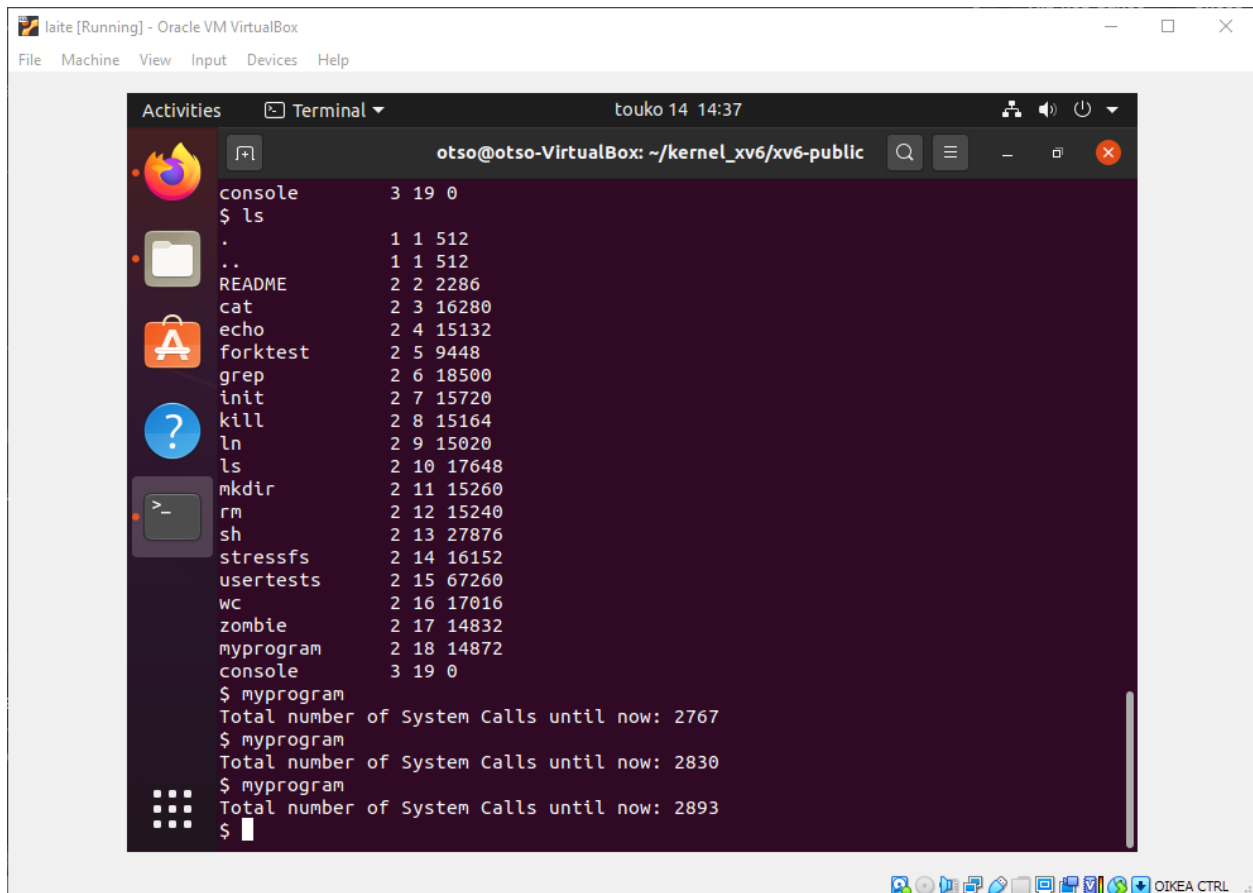
## my-unzip:

Neljäs pienohjelma, joka toimii parina my-zipin kanssa. Toimii vastakkaiseen suuntaan, eli purkaa pakatun binääritekstin ja tulostaa sen ruudulle. Text.z tulostaa molemmat tiedostot yhdessä.

```
Teemu-MacBook-Pro-5:zip TeemuK$ ./my-zip file.txt file2.txt > file.z
Teemu-MacBook-Pro-5:zip TeemuK$ ./my-unzip file.z
aaaaaaaaaabbccccccdd
Teemu-MacBook-Pro-5:zip TeemuK$
```

## Projekti 4

Ohjelma myprogram.c tulostaa sen hetkisten tehtyjen järjestelmäkutsujen määrän konsoliin.



Git clone <https://github.com/mit-pdos/xv6-public>

Tiedostot joihin tehty muutoksia

### **Syscall.h**

Lisää SYS\_readcount listaan

### **Defs.h**

Lisää int readcount(void) //proc.c alapuolelle

### **User.h**

Lisää int readcount(void)

### **Sysproc.c**

Int total\_calls

### **usys.S**

SYSCALL(readcount)

### **Syscall.c**

Lisää extern int sys\_readcount(void)

ja

[SYS\_readcount] sys\_readcount



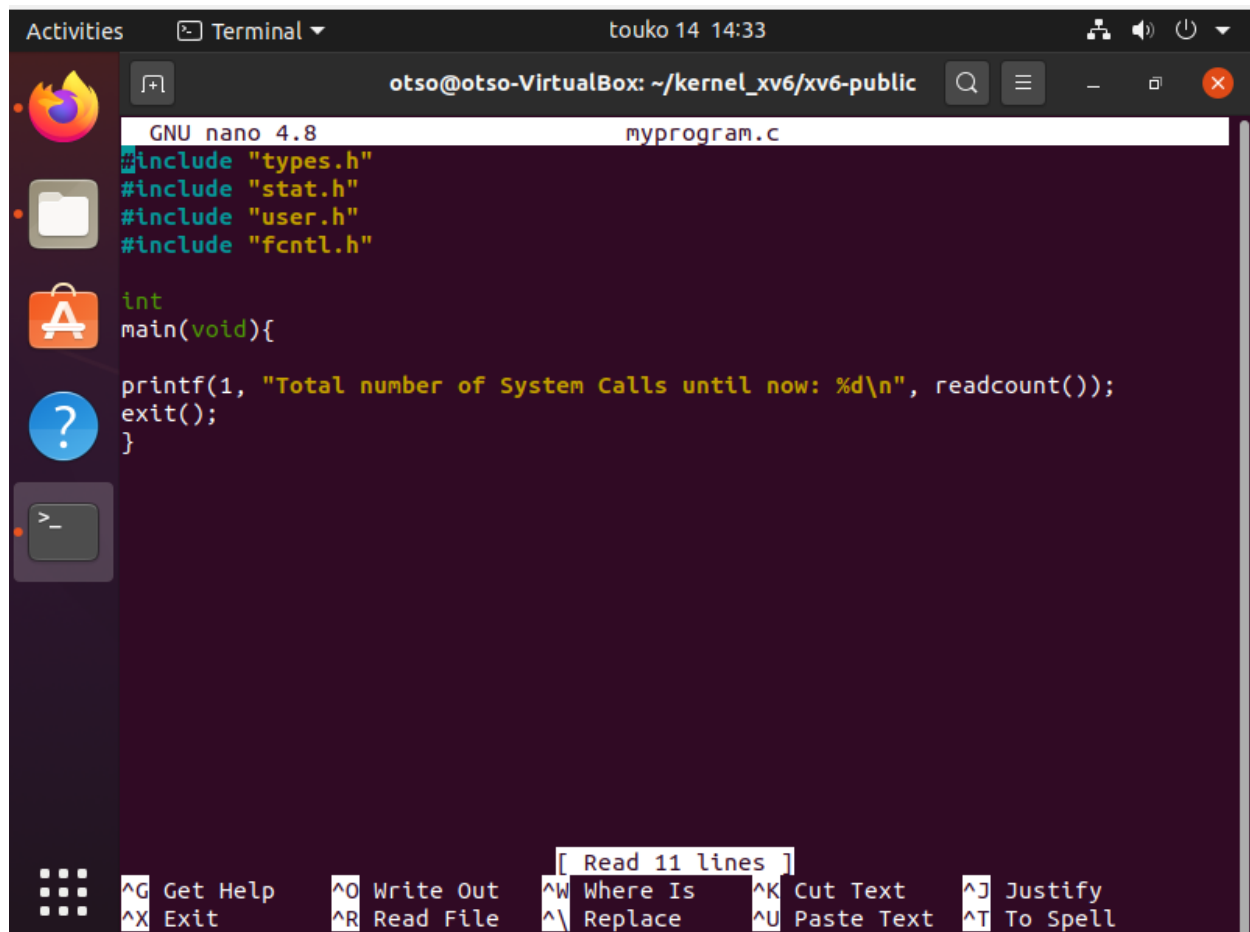
```

void
syscall(void)
{
    total_calls++;
    int num;
}

```

Countteri

## Makefile



The screenshot shows a terminal window titled "otso@otso-VirtualBox: ~/kernel\_xv6/xv6-public" with the time "touko 14 14:33". The window contains the GNU nano 4.8 editor editing a file named "myprogram.c". The code in the editor is as follows:

```

#include "types.h"
#include "stat.h"
#include "user.h"
#include "fcntl.h"

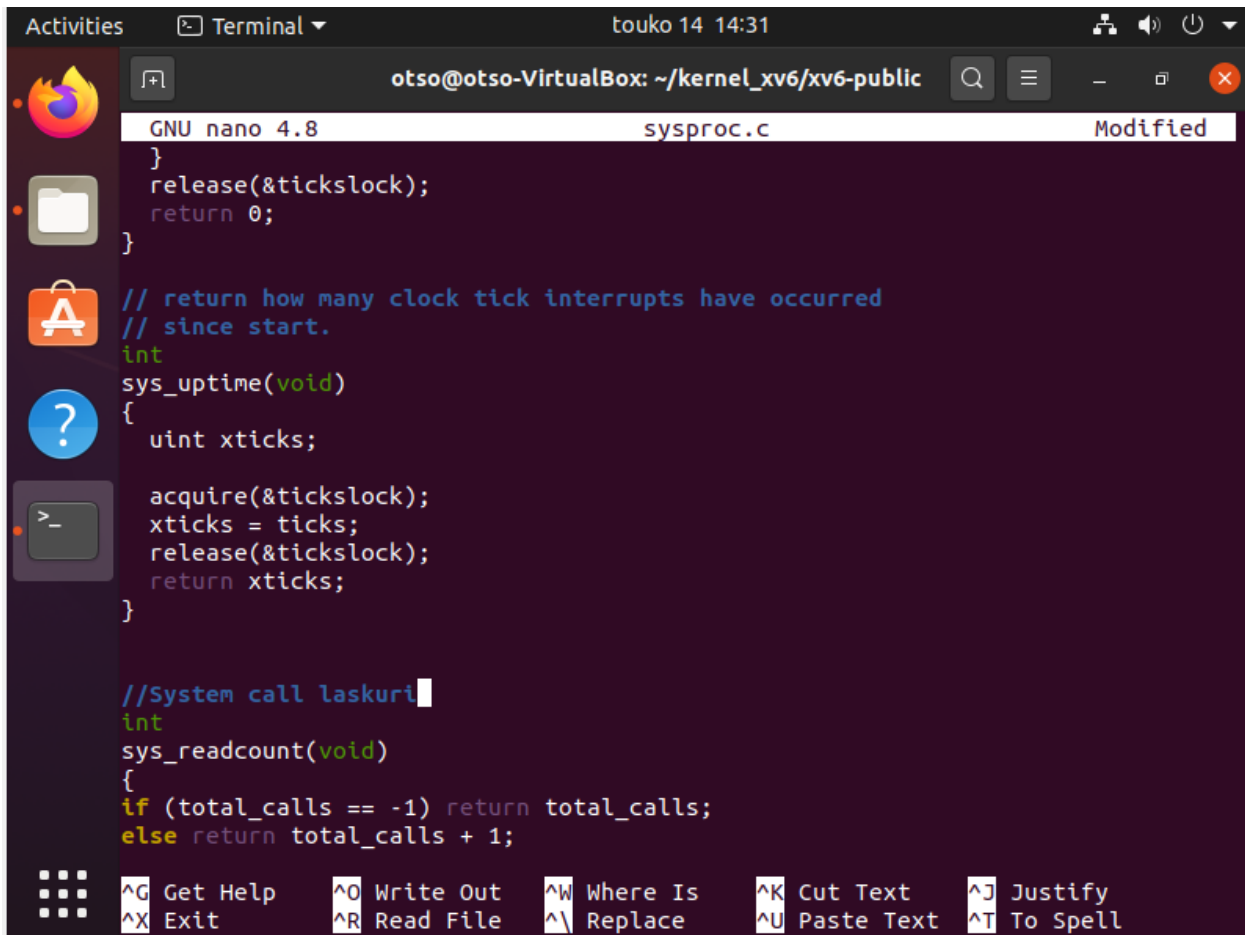
int
main(void){

    printf(1, "Total number of System Calls until now: %d\n", readcount());
    exit();
}

```

At the bottom of the terminal, a status bar displays various keyboard shortcuts: ^G Get Help, ^O Write Out, ^W Where Is, ^K Cut Text, ^J Justify, ^X Exit, ^R Read File, ^\ Replace, ^U Paste Text, and ^T To Spell. A small box above the status bar indicates "[ Read 11 lines ]".

Yksinkertainen C-ohjelma joka kutsuu readcount() ja tulostaa returnin.



```
GNU nano 4.8 sysproc.c Modified
}
release(&tickslock);
return 0;
}

// return how many clock tick interrupts have occurred
// since start.
int
sys_uptime(void)
{
    uint xticks;

    acquire(&tickslock);
    xticks = ticks;
    release(&tickslock);
    return xticks;
}

//System call laskuri
int
sys_readcount(void)
{
    if (total_calls == -1) return total_calls;
    else return total_calls + 1;
}

^G Get Help      ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify
^X Exit          ^R Read File    ^_ Replace      ^U Paste Text   ^T To Spell
```

readcount() palauttaa total\_calls integerin, joka on määritelty sysproc.c tiedostossa ja johon lisätään yksi joka kerta kun systeemi tekee system callin.

<https://gist.github.com/bridgesign/e932115f1d58c7e763e6e443500c6561>

<https://www.youtube.com/watch?v=qZ16UodDlz8>

<https://www.youtube.com/watch?v=vR6z2QGcoo8&t=781s>

# Tehtävien GitHub Repositorio

<https://github.com/TepaXD/HarkkaRepo>