CT30A3370_02.09.2019 Käyttöjärjestelmät ja systeemiohjelmointi

Harjoitustyö

Nikkilä Eetu 0453223

Harjoitustyö jakaantuu kolmeen osaan. Kaikki osat löytyvät repositoriosta: . Valitsin tehtävistä projektit 1, 2 ja 3. Ensimmäinen osa(Htyo_1) sisältää itse luotuja versioita unix ohjelmista cat, grep, zip ja unzip. Toinen osa(Htyo_2) sisältää yksinkertaisen Unix Shellin. Ja kolmas osa(Htyo_3) sisältää itsetehdyn järjestelmä kutsun, joka on tehty xv6 kerneliin.

Projekti 1: Unix Utilities

Tässä osassa tein ohjelmat my-cat, my-grep, my-zip ja my-unzip.

-my-cat:

Lähteenä ohjelmaan käytin pitkälti:
http://cprogrammingwithlinux.blogspot.com/2011/10/c-program-to-simulate-linux-cat-command.html

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

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

        FILE *file;
        char fname[350], ch;
        int i;

        if(argc < 2){   // Tarkistaa syntaxin
                printf("There is a syntax problem. 'Cat File File2 etc'.\n");
                return 0;
        }

        for(i=1; i<argc; i++){  //käy läpi jokaisen annetun tiedoston
                strncpy(fname, argv[i], 350);
                file=fopen(fname, "r");
                if(file == NULL){
                        printf("No file found\n");
                        return 0;
                }
                while((ch=fgetc(file)) != EOF){ //kirjoittaa tiedoston sisällön
                        putchar(ch);
                }
                fclose(file);
        }
        return 0;
}
```

Ja toiminta:

```
eetu@eetu-VirtualBox:~/kaysys/Htyo/Htyo_1$ gcc -o my-cat my-cat.c -Wall -Werror
eetu@eetu-VirtualBox:~/kaysys/Htyo/Htyo_1$ ./my-cat
There is a syntax problem. 'Cat File File2 etc'.
eetu@eetu-VirtualBox:~/kaysys/Htyo/Htyo_1$ ./my-cat fileee
No file found
eetu@eetu-VirtualBox:~/kaysys/Htyo/Htyo_1$ ls
file.z         my-cat    my-grep.c   my-zip        testtext1.txt  unzipped.txt
greptest1.txt  my-cat.c  my-unzip    my-zip.c      testtext2.txt  zip.txt
greptest2.txt  my-grep   my-unzip.c  rfe_test.txt  testtext3.txt  zip.z
eetu@eetu-VirtualBox:~/kaysys/Htyo/Htyo_1$ ./my-cat testtext.txt
No file found
eetu@eetu-VirtualBox:~/kaysys/Htyo/Htyo_1$ ./my-cat testtext1.txt
Hello, does this work?
eetu@eetu-VirtualBox:~/kaysys/Htyo/Htyo_1$ ./my-cat testtext2.txt testtext3.txt
What about when I add multiple rows
What about when I add multiple rows
What about when I add multiple rows
What about when I add multiple rows
What about when I add multiple rows
That should be there 5 times
And now some other stuff
12345

12345
12345
12345
```

Eli ohjelma kirjoittaa komentoriville onnistuneesti annetut tiedostot ja ilmoittaa, jos tiedostoa ei löydy. Virheilmoitukset toimivat mikäli tiedostoa ei ole olemassa tai syntaksi on väärä, mutta ei tarkista tiedostotyyppiä.

-my-grep:

Lähteenä:
http://codetrays.blogspot.com/2015/09/implementation-of-grep-command-in-c.html

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define BUFFER_MAX 5120

```
int main(int argc, char *argv[])
{
        FILE *file;
        char fline[BUFFER_MAX];
        int i;
        if(argc<3){     //Tarkistaa argumenttien määrä
                printf("Syntax error. 'my-grep searchable file file etc'\n");
                return 0;
        }
        for(i=2; i<argc;i++){   //tiedostot, mikäli enemmän kuin 1

                file=fopen(argv[i], "r");
                if(file==NULL){
                        printf("file not found.\n");
                        return 0;
                }

                //hakee rivit tiedostosta
                while(fscanf(file, "%[^\n]\n" , fline)!=EOF){

                        //printtaa rivit joista löytyy annettu hakusana
                        if(strstr(fline, argv[1]) != NULL){
                                printf("%s\n", fline);
                        }

                }
                fclose(file);
        }
```

Text Editor

Tämä kuitenkin aiheuttaa sen että yli 5120(yli BUFFER_MAX) merkkiset rivit rikkovat

ohjelman.

```
eetu@eetu-VirtualBox:~/kaysys/Htyo/Htyo_1$ gcc -o my-grep my-grep.c -Wall -Werr
or
eetu@eetu-VirtualBox:~/kaysys/Htyo/Htyo_1$ ls
file.z         my-cat    my-grep.c   my-zip        testtext1.txt  unzipped.txt
greptest1.txt  my-cat.c  my-unzip    my-zip.c      testtext2.txt  zip.txt
greptest2.txt  my-grep   my-unzip.c  rfe_test.txt  testtext3.txt  zip.z
eetu@eetu-VirtualBox:~/kaysys/Htyo/Htyo_1$ ./my-grep foo greptest1.txt
foo
foo foo
eetu@eetu-VirtualBox:~/kaysys/Htyo/Htyo_1$ ./my-grep foo testtext1.txt
eetu@eetu-VirtualBox:~/kaysys/Htyo/Htyo_1$ ./my-grep foo greptest2.txt
asdimoaaaaaaaaaaaaaaaasshhhhhhhhhhhhwyaaaaaaaaaaaaaaaaaaaaaaaaaagddddddddddddddd
dwadoesfoom   sjiw foo Foo Foo foo
Segmentation fault (core dumped)
eetu@eetu-VirtualBox:~/kaysys/Htyo/Htyo_1$ yli 5120 merkkinen rivi :(
```

Tyhjä rivi:

```
eetu@eetu-VirtualBox:~/kaysys/Htyo/Htyo_1$ ./my-grep \n  greptest1.txt
eetu@eetu-VirtualBox:~/kaysys/Htyo/Htyo_1$
```

-my-zip ja my-unzip

lähteenä pääosin:
https://stackoverflow.com/questions/42193703/convert-text-file-into-binary-file-in-c
zip:

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

        FILE *fin, *fout;
        int count;
        char c1, c2;
        fin=fopen(argv[1], "r");
        fout=fopen(argv[2], "wb");      //tiedostot
        fread(&c1, sizeof(char), 1, fin);

        while(!feof(fin)){
                count=0;
                do{
                        fread(&c2, sizeof(char), 1, fin);       //rle pyöritys
                        count++;
                }while(c1 == c2 && c2 != EOF && count < 255);

                fwrite(&count, sizeof(int), 1, fout);
                fwrite(&c1, sizeof(char), 1, fout); //tiedostoon kirjoitus
                c1 = c2;
        }

        fclose(fin);
        fclose(fout);
        return 0;
}
```

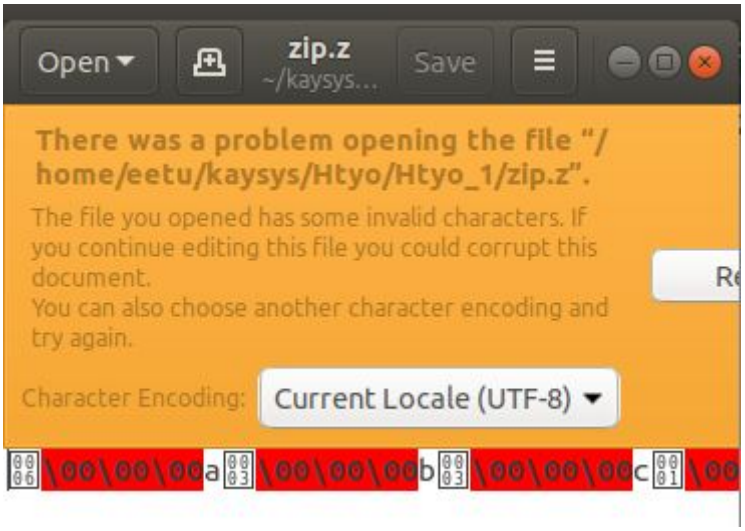
unzip:

```c
int main(int argc, char *argv[]){
        FILE *fin, *fout;

        fin=fopen(argv[1], "rb");       //tiedostot
        fout=fopen(argv[2], "w");

        char c;
        int count, i;

        while(!feof(fin)){
                fread(&count, sizeof(int), 1, fin);     //koodin rle purkaminen
                fread(&c, sizeof(char), 1, fin);
                for(i = 0; i < count; i++){
                        fwrite(&c, sizeof(char), 1, fout);      //tiedostoon kirjoitus
                }
        }


        fclose(fin);
        fclose(fout);
        return 0;
}
```

Molemmat kääntyvät:

```
eetu@eetu-VirtualBox:~/kaysys/Htyo/Htyo_1$ gcc -o my-zip my-zip.c -Wall -Werror
eetu@eetu-VirtualBox:~/kaysys/Htyo/Htyo_1$ gcc -o my-unzip my-unzip.c -Wall -We
rror
eetu@eetu-VirtualBox:~/kaysys/Htyo/Htyo_1$ ./my-zip zip.txt zip.z
eetu@eetu-VirtualBox:~/kaysys/Htyo/Htyo_1$ ./my-unzip zip.z unzip.txt
eetu@eetu-VirtualBox:~/kaysys/Htyo/Htyo_1$
```

toiminta:



Eli zip.z ei pysty avaamaan, mutta zip.txt ja unzip.txt olivat samat.

Projekti 2: Unix Shell

Tässä osassa tein simple shellin nimeltä wish. Lähteenä pääosin:
https://danrl.com/blog/2018/how-to-write-a-tiny-shell-in-c/

-my-wish.c

#include <unistd.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sys/wait.h>
#define EXITCMD “exit”

```
int main(void) {
    for (;;) {
        char in[256] = { 0x0 };     //alustaa komentorivi listan
        char *ptr = in;             //pointteri nykyiseen komentoon
        char *args[99] = { NULL }; //alustaa listan

        printf("wish >");       //promptin kirjoitus

        fgets(in, 256, stdin); //ottaa inputin

        if (*ptr == '\n')           //ohittaa tyhjän inputin
            continue;
        //muuttaa inputin argumentti listaksi
        for (int i = 0; i < sizeof(args) && *ptr; ptr++) {
            if (*ptr == ' ') continue;
            if (*ptr == '\n') break;    //kunnes rivinvaihto katkaisee
            for (args[i++] = ptr; *ptr && *ptr != ' ' && *ptr != '\n'; ptr++);
            *ptr = '\0';
        }

        if (strcmp(EXITCMD, args[0]) == 0) return 0; //jos input on "exit"
sulkee shellin

        if (fork() == 0) exit(execvp(args[0], args)); //ajaa annetun argumentin
        wait(NULL);//varmistaa loppumisen
    }
}
```

```
eetu@eetu-VirtualBox:~/kaysys/Htyo/Htyo_2$ ./wish
wish >ls
myShell.c  myShell_loop.c  mywish.c  test2  wish
wish >ls -a
.  ..  myShell.c  myShell_loop.c  mywish.c  test2  wish
wish >mkdir test
wish >ls
myShell.c  myShell_loop.c  mywish.c  test  test2  wish
wish >rmdir test
wish >ls
myShell.c  myShell_loop.c  mywish.c  test2  wish
wish >rm test2
rm: cannot remove 'test2': Is a directory
wish >cat mywish.c
/*How to write a tiny shell in C*/
#include <unistd.h>
#include <stdio.h>
```

```
wish >grep wish mywish.c
                printf("wish >");               //promptin kirjoitus
wish >gedit newfile
```

gedit avautui, mutta jumitti wishin sulkemisen jälkeenkin.

Projekti 3: Kernel Hacking

Tässä osassa tein järjestelmäkutsun rcount() xv6 kerneliin, joka laskee read() järjestelmäkutsujen määrän.
Lähteenä käytin pääosin: https://www.youtube.com/watch?v=21SVYiKhcwM, https://www.youtube.com/watch?v=vR6z2QGcoo8 ja https://stackoverflow.com/questions/8021774/how-do-i-add-a-system-call-utility-in-xv6 .

Kuitenkin päädyin hieman erikoiseen ja ehkä hieman vaikeaankin ratkaisuun.

sysproc.c ohjelmassa vain referoin ohjelman

```
int
sys_rcount(void)
{
        return rcount();
}
```

Sitten nämä useat headerit ja muut maininnat
defs.h

```
void            yield(void);
int             rcount(void);
```

usys.S

```
SYSCALL(uptime)
SYSCALL(rcount)
```

syscall.h

```
#define SYS_close  21
#define SYS_rcount 22
```

Lisäsin järjestelmäkutsulle sijainnin

syscall.c

```
[SYS_close]   sys_close,
[SYS_rcount]  sys_rcount,
};
```

```
extern int sys_uptime(void);
extern int sys_rcount(void);
```

muokkasin sys_read komentoa sisltämään laskurin countReads;

sysfile.c

```
int
sys_read(void)
{
  struct file *f;
  int n;
  char *p;
  extern int countReads;
  countReads++;

  if(argfd(0, 0, &f) < 0 || argint(2, &n) < 0 || argptr(1,
&p, n) < 0)
    return -1;
  return fileread(f, p, n);
}
```

Lisäsin proc.c ohjelman sisällön kun en saanut sitä toimimaan rcount.c

proc.c

```
int
rcount()
{
        cprintf("There has been %d read systemcalls since
last startup.\n", countReads);
        return 22;
}
```

Referoin omassa ohjelmassaan edellistä ja ajan sen siinä

rcount.c

```c
#include "types.h"
#include "stat.h"
#include "user.h"
#include "syscall.h"

int
main(int argc, char *argv[])
{
        rcount();
        exit();
}
```

Ja lopuksi lisätään ne kernelin makefileen
Makefile

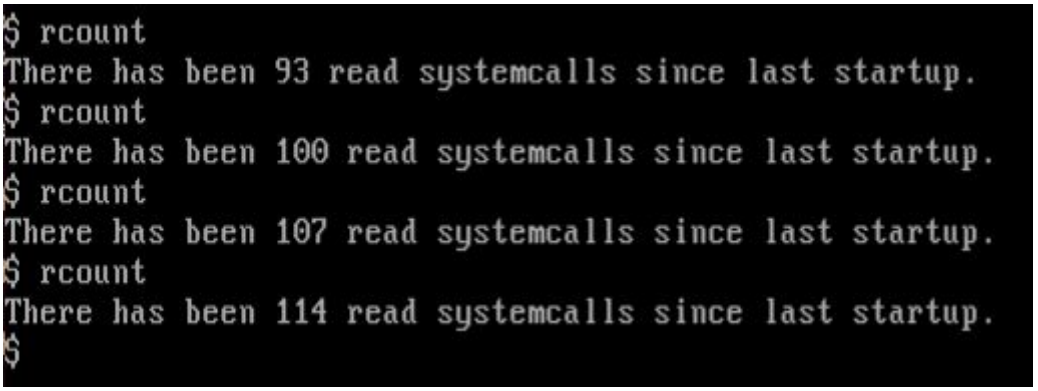
```
EXTRA=\
        mkfs.c ulib.c user.h cat.c echo.c forktest.c grep.
kill.c\
        ln.c ls.c mkdir.c rm.c stressfs.c usertests.c wc.c
rcount.c zombie.c\
        printf.c umalloc.c\
        README dot-bochsrc *.pl toc.* runoff runoff1
runoff.list\
```

```
JPROGS=\
        _cat\
        _echo\
        _forktest\
        _grep\
        _init\
        _kill\
        _ln\
        _ls\
        _mkdir\
        _rm\
        _sh\
        _stressfs\
        _usertests\
        _wc\
        _rcount\
        _zombie\
```

En usko tämän olevan järkevä ratkaisu, mutta kun lopulta pääsin toimivaan niin en halunnut alkaa rikkoa sitä enää. Vähintäänkin on paljon ylimääräistä.

Toiminta:

```
rm              2 12 12772
sh              2 13 23260
stressfs        2 14 13440
usertests       2 15 56376
wc              2 16 14192
rcount          2 17 12476
zombie          2 18 12436
console         3 19 0
testdir         1 20 32
$ rcount
There has been 43 read systemcalls since last startup.
$
```



LÄHDELISTA

-kurssimateriaali
-http://cprogrammingwithlinux.blogspot.com/2011/10/c-program-to-simulate-linux-cat-command.html
-http://codetrays.blogspot.com/2015/09/implementation-of-grep-command-in-c.html
-https://stackoverflow.com/questions/42193703/convert-text-file-into-binary-file-in-c
-https://danrl.com/blog/2018/how-to-write-a-tiny-shell-in-c/
-https://www.youtube.com/watch?v=21SVYiKhcwM,
-https://www.youtube.com/watch?v=vR6z2QGcoo8
-https://stackoverflow.com/questions/8021774/how-do-i-add-a-system-call-utility-in-xv6