

## Project 1: Unix Utilities

### **Määritelmä**

Projektin tarkoituksena oli rakentaa omat C-kieliset versiot neljästä unix ohjelmasta: my-cat, my-grep, my-zip ja my-unzip. My-cat lukee tiedoston ja tulostaa sen sisällön. My-grep käy tiedoston läpi rivi kerrallaan ja tulostaa rivit, jotka sisältävät käyttäjän antaman hakutermi. My-zip pakkaa tiedoston käyttämällä run-length encoding pakkausmetodia ja my-unzip purkaa pakatut tiedostot.

### **my-cat**

My-cat on yksinkertainen ohjelma, joka tarkistaa ensin, että käyttäjä on antanut ohjelman vaatiman määrän parametreja ja jos ei niin ohjelma sulkeutuu. Jos käyttäjä on antanut useamman kuin yhden tiedoston luettavaksi, käydään jokainen tiedosto läpi for loopissa, jossa tiedostot avataan *fopen*:lla lukumoodissa ja tiedostosta luetaan 100 merkkiä kerrallaan *fgets*:llä puskuriin, joka tulostetaan, kunnes tiedosto on käyty läpi, jonka jälkeen tiedosto suljetaan. Jos tiedostoa ei saada aukaistua, tulostetaan virheviesti ja ohjelma sulkeutuu. Ohessa on kuva my-cat.c tiedostosta.

## Kuva: my-cat.c

## my-grep

My-grep on hyvin samankaltainen ohjelma kuin my-cat. Se tarkastaa ensin, että käyttäjä on antanut sopivan määrän parametreja, ja jos ei ohjelma sulkeutuu. Tämän jälkeen käytetään *malloc*:ia varaamaan muistia puskurina käytettävälle *line*:lle. Jos käyttäjä ei anna input tiedostoa, niin käytetään tiedoston sijasta käyttäjän syötettä. Toteutin tämän tarkistamalla parametrien määrän, eli onko tiedosto annettu, ja lukemalla *getline*:lla puskuriin *stdin*:stä. Hakutermiä etsitään luetulta riviltä *strstr* funktiolla, joka etsii osajonoa merkkijonosta. Ne rivit, joista osajono löytyy, tulostetaan. Sama toistetaan, jos käyttäjä on antanut yhden tai useamman input tiedoston. Tiedostot käydään läpi for loopissa, avataan lukumoodiin, luetaan rivi kerrallaan ja tulostetaan rivit, joissa esiintyy käyttäjän antama hakutermi. Lopuksi tiedosto suljetaan ja vapautetaan muisti. Ohessa on kuva my-grep.c tiedostosta.

## Kuva: my-grep.c

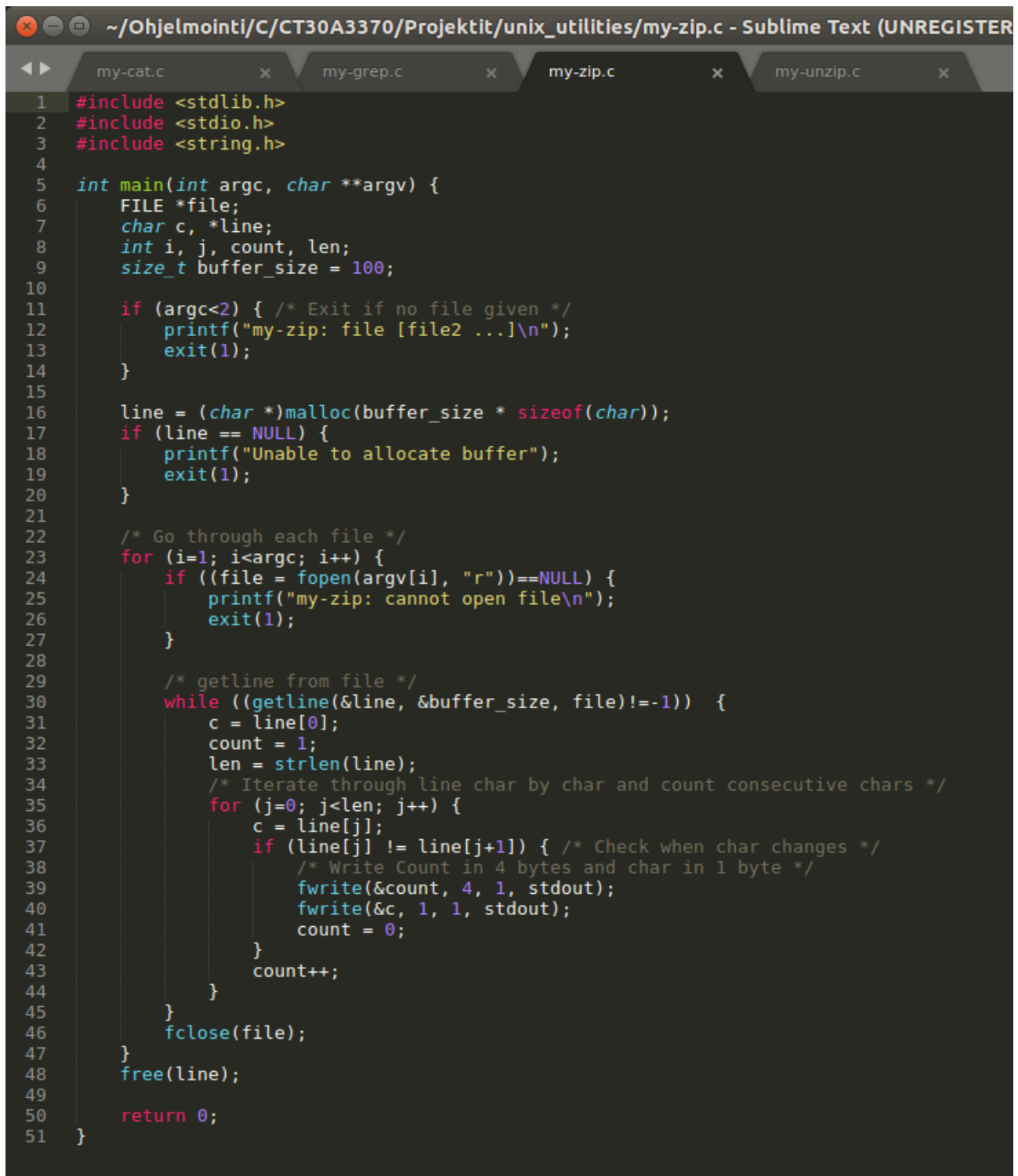
```
~/Ohjelmointi/C/CT30A3370/Projektit/unix_utilities/my-grep.c - Sublime Text (
my-cat.c x my-grep.c x my-unzip.c x my-zip.c
1  #include <stdlib.h>
2  #include <stdio.h>
3  #include <string.h>
4
5  int main(int argc, char **argv) {
6      int i;
7      char *line;
8      size_t buffer_size = 100;
9      FILE* file;
10
11     if (argc<2) { /* Exit if no arguments given */
12         printf("my-grep: searchterm [file ...]\n");
13         exit(0);
14     }
15
16     line = (char *)malloc(buffer_size * sizeof(char));
17     if (line == NULL) {
18         printf("Unable to allocate buffer");
19         exit(1);
20     }
21
22     if (argc==2) {
23         /* getline from stdin if no file given */
24         getline(&line, &buffer_size, stdin);
25         /* check line for searchterm */
26         if (strstr(line, argv[1])) {
27             printf("%s", line);
28         }
29     } else {
30         /* Go through each file */
31         for (i=2; i<argc; i++) {
32             if ((file = fopen(argv[i], "r"))==NULL) {
33                 printf("my-grep: cannot open file\n");
34                 exit(1);
35             }
36
37             /* getline from file */
38             while ((getline(&line, &buffer_size, file)!=-1)) {
39                 /* check line for searchterm */
40                 if (strstr(line, argv[1])) {
41                     printf("%s", line);
42                 }
43             }
44             printf("\n");
45
46             fclose(file);
47         }
48     }
49
50     free(line);
51     return 0;
52 }
```

## my-zip

My-zip on siis ohjelma, joka pakkaa sille annetun tiedoston run-length encodingilla, eli esittää tiedoston muodossa, jossa vuorottelevat luku ja kirjain parit esimerkiksi 10a5b6c. Luku kirjaimen edessä kertoo, kuinka monta kertaa kirjain toistuu peräkkäin ennen seuraavaa kirjainta, eli 5a2b1c olisi *aaaaabbc*. My-zip luo uuden tiedoston, jossa nämä luku ja kirjain parit ovat viiden tavun sarjoina: luku on binäärimuodossa neljän tavun suuruisena ja kirjain ASCII muodossa yhden tavun suuruisena.

Ohjelma alkaa taas tarkistamalla, että sille on annettu sopiva määrä parametreja, jonka jälkeen varataan puskurille muistia. Sitten käydään tiedostot läpi yksi kerrallaan for loopissa. Tiedostot avataan lukumoodissa ja luetaan rivi kerrallaan puskuriin käyttäen *getline* funktiota. Jokainen rivi käydään läpi merkki kerrallaan for loopissa, jossa kasvatetaan laskuria, niin kauan kuin sama merkki toistuu peräkkäin. Kun vertailu huomaa, että seuraava merkki merkkijonossa on eri kuin nykyinen merkki, kirjoitetaan laskurin arvo *stdout*:iin neljän tavun suuruisena binäärimuodossa käyttäen *fwrite* funktiota, sama toistetaan nykyiselle merkille yhden tavun suuruisena ja laskuri resetoidaan siirryttäessä uuteen merkkiin. Lopuksi tiedosto suljetaan ja muisti vapautetaan. Ohessa on kuva my-zip.c tiedostosta.

## Kuva: my-zip.c



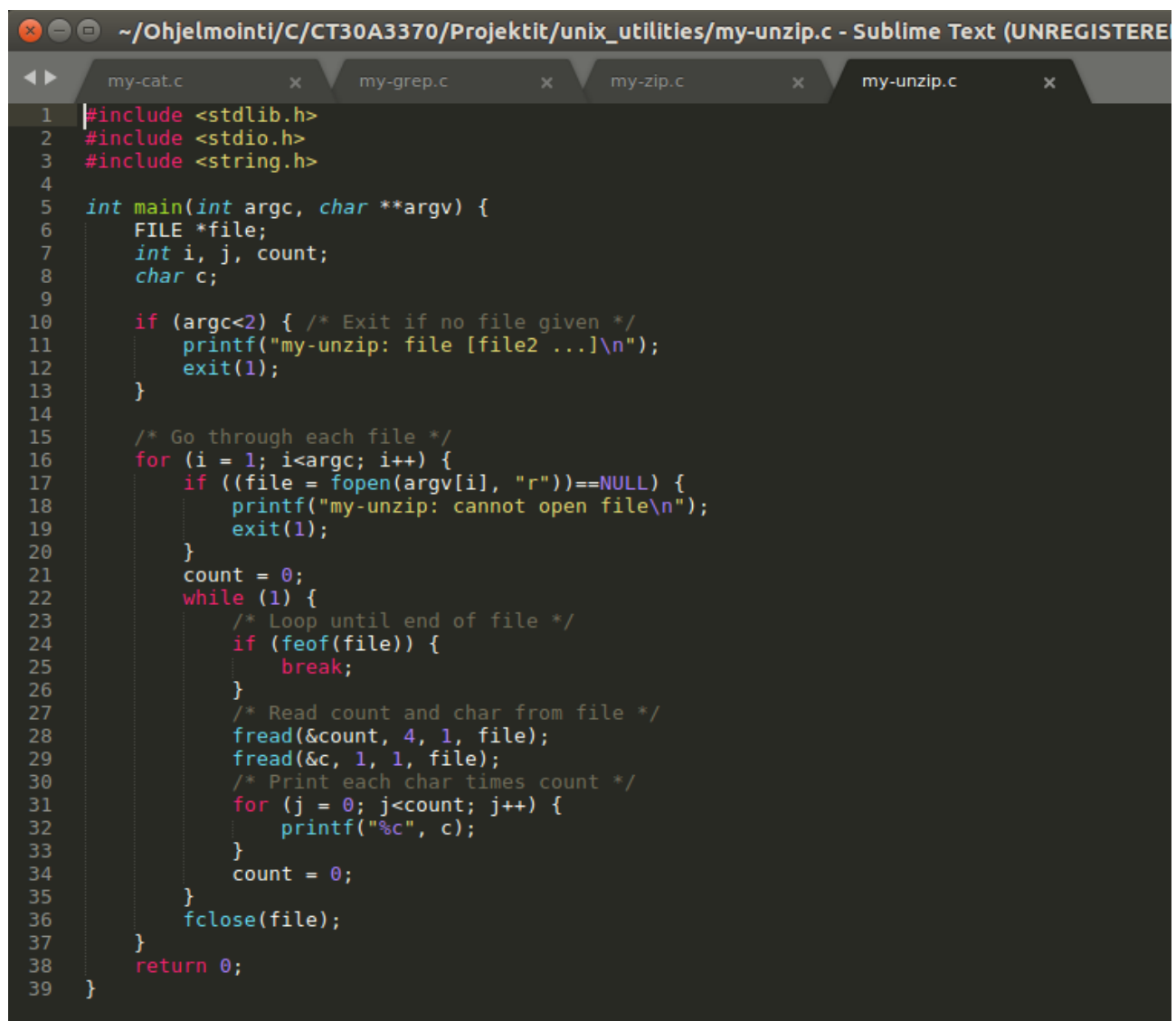
```
1 #include <stdlib.h>
2 #include <stdio.h>
3 #include <string.h>
4
5 int main(int argc, char **argv) {
6     FILE *file;
7     char c, *line;
8     int i, j, count, len;
9     size_t buffer_size = 100;
10
11     if (argc<2) { /* Exit if no file given */
12         printf("my-zip: file [file2 ...]\n");
13         exit(1);
14     }
15
16     line = (char *)malloc(buffer_size * sizeof(char));
17     if (line == NULL) {
18         printf("Unable to allocate buffer");
19         exit(1);
20     }
21
22     /* Go through each file */
23     for (i=1; i<argc; i++) {
24         if ((file = fopen(argv[i], "r"))==NULL) {
25             printf("my-zip: cannot open file\n");
26             exit(1);
27         }
28
29         /* getline from file */
30         while ((getline(&line, &buffer_size, file)!=-1)) {
31             c = line[0];
32             count = 1;
33             len = strlen(line);
34             /* Iterate through line char by char and count consecutive chars */
35             for (j=0; j<len; j++) {
36                 c = line[j];
37                 if (line[j] != line[j+1]) { /* Check when char changes */
38                     /* Write Count in 4 bytes and char in 1 byte */
39                     fwrite(&count, 4, 1, stdout);
40                     fwrite(&c, 1, 1, stdout);
41                     count = 0;
42                 }
43                 count++;
44             }
45             fclose(file);
46         }
47         free(line);
48     }
49     return 0;
50 }
51 }
```

## my-unzip

My unzip purkaa my-zipin pakkaaman tiedoston ja tulostaa sen sisällön selkokieლისenä. Aluksi testataan taas, että ohjelmalle on annettu sopiva määrä parametreja, jonka jälkeen

tiedostot käydään läpi yksi kerrallaan for loopissa ja avataan lukumoodissa. Tiedostoa käydään läpi while loopissa, josta irrottaudutaan, kun *feof* funktio palauttaa nollasta eroavan arvon merkiksi siitä, että on päästy tiedoston loppuun. Tiedostosta luetaan käyttäen, *fread* funktiota ja koska tiedetään, missä muodossa tiedostoon on kirjoitettu, voidaan sieltä lukea neljän tavun suuruinen luku binäärimuodossa ja sen jälkeen merkki yhden tavun suuruisena. Sitten merkki tulostetaan for loopissa luvun määrittämän kierroksen ajan. Näin pakkaus saadaan purettua ja tulostettua selkokieliseksi. Lopuksi tiedosto suljetaan. Ohessa on kuva my-unzip.c tiedostosta.

### Kuva: my-unzip.c



```
1 #include <stdlib.h>
2 #include <stdio.h>
3 #include <string.h>
4
5 int main(int argc, char **argv) {
6     FILE *file;
7     int i, j, count;
8     char c;
9
10    if (argc<2) { /* Exit if no file given */
11        printf("my-unzip: file [file2 ...]\n");
12        exit(1);
13    }
14
15    /* Go through each file */
16    for (i = 1; i<argc; i++) {
17        if ((file = fopen(argv[i], "r"))==NULL) {
18            printf("my-unzip: cannot open file\n");
19            exit(1);
20        }
21        count = 0;
22        while (1) {
23            /* Loop until end of file */
24            if (feof(file)) {
25                break;
26            }
27            /* Read count and char from file */
28            fread(&count, 4, 1, file);
29            fread(&c, 1, 1, file);
30            /* Print each char times count */
31            for (j = 0; j<count; j++) {
32                printf("%c", c);
33            }
34            count = 0;
35        }
36        fclose(file);
37    }
38    return 0;
39 }
```