

CT30A3370 Käyttöjärjestelmät ja systeemiohjelmointi – Harjoitustyöprojektit

Tommi Uponen, 001134917

Elias Kukkonen, 000458717

GitHub (Projektin koodi): https://github.com/TommiUp/CT30A3370_Course_Projects

Pvm: 19.03.2025

Projekti 1

Perustiedot

- Projektin numero: 1
- Projektin nimi: Reverse
- Palautuspäivämäärä: 19.3.2025

Projektin kuvaus

Tämän projektin tavoitteena on toteuttaa yksinkertainen C-ohjelma reverse, joka lukee rivejä joko tiedostosta tai standardisyytteestä (stdin) ja tulostaa ne käänteisessä järjestyksessä joko standarditulosteeseen (stdout) tai toiseen tiedostoon. Projekti toimii harjoitusesimerkkinä erilaisista C-kielessä käytettävistä operaatioista tiedoston käsittelyssä sekä dynaamisesta muistinhallinnasta (malloc, realloc, free).

Toteutuksen dokumentaatio

Käytetyt teknologiat ja ratkaisut:

C-ohjelmointikieli

- Kääntäjä: gcc
- Kirjastot: <stdio.h>, <stdlib.h>, <string.h>, <errno.h>, <unistd.h>

Tiedostonkäsittely

- fopen() ja fclose() tiedostojen avaamiseen ja sulkemiseen
- stdin ja stdout käytetty vakiosyötteenä ja -tulosteena

Rivien luku

- getline() mahdollistaa rivien lukemisen dynaamisesti kasvavaan puskuriin
- Ei rajoitusta rivin pituudelle

Muistinhallinta

- malloc(), realloc(), ja free() rivien tallentamista ja vapauttamista varten
- Taulukon koon dynaaminen kasvattaminen

Ohjelman eri versiot:

Versio 1 (Barebones)

- Yksinkertainen toteutus: lukee joko stdin:stä tai tiedostosta (1 argumentti), tulostaa käänteisessä järjestyksessä stdout:iin.
- Käytännössä ei virhetarkistuksia (esim. fopen voi epäonnistua ilman, että ohjelma reagoi).
- Sopii ymmärtämään ohjelman ydintoiminnon (rivinluku, tallennus, käännetty tulostus).

```
home > tommi > C reverse_v1.c > ...
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <sys/types.h>
5
6  int main(int argc, char *argv[])
7  {
8      FILE *inFile = stdin; // default to stdin
9      if (argc == 2)
10     {
11         // attempt to open the file without error checking
12         inFile = fopen(argv[1], "r");
13     }
14
15     size_t capacity = 8; // initial array size
16     size_t size = 0;     // actual number of lines
17     char **lines = malloc(capacity * sizeof(char *));
18
19     char *buffer = NULL; // for getline() function
20     size_t bufsize = 0;  // for getline() function
21     ssize_t linelen;
22
23     // read lines until EOF
24     while ((linelen = getline(&buffer, &bufsize, inFile)) != -1)
25     {
26         // if the array is full, increase its size
27         if (size == capacity)
28         {
29             capacity *= 2;
30             lines = realloc(lines, capacity * sizeof(char *));
31         }
32
33         // memory allocation
34         lines[size] = malloc(linelen + 1);
35         // copy the read line
36         strncpy(lines[size], buffer, linelen + 1);
37         size++;
```

```
38     }
39
40     free(buffer); // free the buffer memory
41
42     // print the lines in reverse order
43     for (ssize_t i = size - 1; i >= 0; i--)
44     {
45         printf("%s", lines[i]);
46         free(lines[i]); // free the line memory
47     }
48     free(lines); // free the array memory
49
50     // close the file if it was opened
51     if (inFile != stdin)
52     {
53         fclose(inFile);
54     }
55
56     return 0; // program terminates
57 }
58
```

Versio 2 (Osittainen virhetarkistus)

- Käsittelee 0–2 argumenttia (syötetiedosto + tulostetiedosto).
- Tarkistaa, jos argumentteja on liikaa.
- Tarkistaa tiedoston avaamisen onnistumisen, mutta ei tarkista malloc/realloc-ongelmia eikä sitä, ovatko syöte- ja tulostetiedosto samat.
- Parempi kuin v1, mutta ei vielä täydellinen.

```

home > tommy > C reverse_v2.c > ...
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <sys/types.h>
5
6  int main(int argc, char *argv[])
7  {
8      // handling the number of commandline arguments
9      if (argc > 3)
10     {
11         fprintf(stderr, "usage: reverse <input> <output>\n");
12         return 1;
13     }
14
15     FILE *inFile = stdin; // default to stdin
16     FILE *outFile = stdout; // default to stdout
17
18     // open files according to the given arguments
19     if (argc >= 2)
20     {
21         // opening input file
22         inFile = fopen(argv[1], "r");
23         if (inFile == NULL)
24         {
25             fprintf(stderr, "cannot open file '%s'\n", argv[1]);
26             return 1;
27         }
28     }
29     if (argc == 3)
30     {
31         // opening output file
32         outFile = fopen(argv[2], "w");
33         if (outFile == NULL)
34         {
35             fprintf(stderr, "cannot open file '%s'\n", argv[2]);
36             if (inFile != stdin)
37                 fclose(inFile);

```

```

38         return 1;
39     }
40 }
41
42 // read lines into memory
43 size_t capacity = 8; // initial array size
44 size_t size = 0; // actual number of lines
45 char **lines = malloc(sizeof(char *) * capacity);
46
47 char *buffer = NULL; // for getline() function
48 size_t bufsize = 0; // for getline() function
49 ssize_t linelen;
50
51 // read lines until EOF
52 while ((linelen = getline(&buffer, &bufsize, inFile)) != -1)
53 {
54     // if the array is full, increase its size
55     if (size == capacity)
56     {
57         capacity *= 2;
58         lines = realloc(lines, sizeof(char *) * capacity);
59     }
60
61     // memory allocation
62     lines[size] = malloc(linelen + 1);
63     // copy the read line
64     strncpy(lines[size], buffer, linelen + 1);
65     size++;
66 }
67
68 free(buffer); // free the buffer memory
69
70 // print the lines in reverse order
71 for (ssize_t i = size - 1; i >= 0; i--)
72 {

```

```

73     fprintf(outFile, "%s", lines[i]);
74     free(lines[i]); // free the line memory
75 }
76 free(lines); // free the array memory
77
78 // close the file if it was opened
79 if (inFile != stdin)
80     fclose(inFile);
81 if (outFile != stdout)
82     fclose(outFile);
83
84 return 0; // program terminates
85 }
86

```

Versio 3 (Täysi toteutus)

- Toteuttaa kaikki projektin vaatimukset
- Argumenttien määrä (0–2)
- Virheilmoitus, jos tiedostoa ei voi avata
- Tarkistus, että syöte- ja tulostetiedosto eivät ole samat
- Muistinvähennys (malloc/realloc) -virheiden tarkistus

- Kaikki virheilmoitukset tulostetaan stderr-virtaukseen
- Palauttaa virhekoodin exit(1) tarpeen mukaan
- Tämä on projektin lopullinen tai täydellinen versio.

```
home > tommi > C reverse_v3.c > ...
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <errno.h>
5  #include <unistd.h>
6
7  int main(int argc, char *argv[])
8  {
9      // handling the number of commandline arguments
10     if (argc > 3)
11     {
12         fprintf(stderr, "usage: reverse <input> <output>\n");
13         exit(1);
14     }
15
16     FILE *inFile = stdin; // default to stdin
17     FILE *outFile = stdout; // default to stdout
18
19     // open files according to the given arguments
20     if (argc >= 2)
21     {
22         // opening input file
23         inFile = fopen(argv[1], "r");
24         if (inFile == NULL)
25         {
26             fprintf(stderr, "error: cannot open file '%s'\n", argv[1]);
27             exit(1);
28         }
29     }
30     if (argc == 3)
31     {
32         // check if the input and output file names are the same
33         if (strcmp(argv[1], argv[2]) == 0)
34         {
35             fprintf(stderr, "input and output file must differ\n");
36             if (inFile != stdin)
37                 fclose(inFile);
```

```
38         exit(1);
39     }
40     // opening output file
41     outFile = fopen(argv[2], "w");
42     if (outFile == NULL)
43     {
44         fprintf(stderr, "error: cannot open file '%s'\n", argv[2]);
45         if (inFile != stdin)
46             fclose(inFile);
47         exit(1);
48     }
49 }
50
51 // read lines into memory
52 size_t capacity = 8; // initial array size
53 size_t size = 0; // actual number of lines
54 char **lines = malloc(sizeof(char *) * capacity);
55 if (lines == NULL)
56 {
57     fprintf(stderr, "malloc failed\n");
58     if (inFile != stdin)
59         fclose(inFile);
60     if (outFile != stdout)
61         fclose(outFile);
62     exit(1);
63 }
64
65 char *buffer = NULL; // for getline() function
66 size_t bufsize = 0; // for getline() function
67 ssize_t linelen;
68
69 // read lines until EOF
70 while ((linelen = getline(&buffer, &bufsize, inFile)) != -1)
```

```
71 {
72     // if the array is full, increase its size
73     if (size == capacity)
74     {
75         capacity *= 2;
76         char **tmp = realloc(lines, sizeof(char *) * capacity);
77         if (tmp == NULL)
78         {
79             free(buffer);
80             fprintf(stderr, "malloc failed\n");
81             if (inFile != stdin)
82                 fclose(inFile);
83             if (outFile != stdout)
84                 fclose(outFile);
85
86             // free already allocated lines
87             for (size_t i = 0; i < size; i++)
88             {
89                 free(lines[i]);
90             }
91             free(lines);
92             exit(1);
93         }
94         lines = tmp;
95     }
96
97     // allocate memory for the line and copy it
98     lines[size] = malloc((linelen + 1) * sizeof(char));
99     if (lines[size] == NULL)
100     {
101         free(buffer);
102         fprintf(stderr, "malloc failed\n");
103         if (inFile != stdin)
104             fclose(inFile);
105         if (outFile != stdout)
```

```
106         fclose(outFile);
107
108     // free already allocated lines
109     for (size_t i = 0; i < size; i++)
110     {
111         free(lines[i]);
112     }
113     free(lines);
114     exit(1);
115 }
116 strncpy(lines[size], buffer, linelen + 1);
117 size++;
118 }
119
120 free(buffer); // free the buffer memory
121
122 // print the lines in reverse order
123 for (ssize_t i = size - 1; i >= 0; i--)
124 {
125     fprintf(outFile, "%s", lines[i]);
126     free(lines[i]); // free the line memory
127 }
128 free(lines); // free the array memory
129
130 // close the file if it was opened
131 if (inFile != stdin)
132     fclose(inFile);
133 if (outFile != stdout)
134     fclose(outFile);
135
136 return 0; // program terminates
137 }
138
```

Testaus ja tulokset

Ilman argumentteja:

- Kutsutaan `./reverse` ja syötetään rivejä stdin:in kautta (manuaalisesti). Tarkastellaan, tulostuvatko rivit käänteisessä järjestyksessä.

```
tommi@LAPTOP-EBRN4IOO:~$ gcc -o reverse_v3 reverse_v3.c -Wall -Werror
tommi@LAPTOP-EBRN4IOO:~$ echo -e "line1\nline2\nline3" | ./reverse_v3
line3
line2
line1
tommi@LAPTOP-EBRN4IOO:~$ echo $?
0
```

Yksi argumentti (syötetiedosto):

- `./reverse input.txt` tulostaa syötetiedoston sisällön käänteisessä järjestyksessä ruudulle.

```
tommi@LAPTOP-EBRN4IOO:~$ echo -e "hello\nthis\nis\na file" > test1.txt
tommi@LAPTOP-EBRN4IOO:~$ ./reverse_v3 test1.txt
a file
is
this
hello
tommi@LAPTOP-EBRN4IOO:~$ echo $?
0
```

Kaksi argumenttia (syöte + tuloste):

- `./reverse input.txt output.txt` lukee `input.txt`-tiedostosta ja kirjoittaa `output.txt`-tiedostoon käänteisessä järjestyksessä.

```
tommi@LAPTOP-EBRN4IOO:~$ ./reverse_v3 test1.txt output.txt
tommi@LAPTOP-EBRN4IOO:~$ cat output.txt
a file
is
this
hello
tommi@LAPTOP-EBRN4IOO:~$ echo $?
0
```

Virhetilanteet:

- Sama syöte- ja tulostetiedosto: `./reverse input.txt input.txt` → Virheilmoitus.
- Tiedostoa ei voi avata: `./reverse ei_ole.txt` → `error: cannot open file '...'`.
- Liian monta argumenttia: `./reverse one.txt two.txt three.txt` → `usage: reverse <input> <output>`.

- Mahdolliset muistinvarausvirheet testattuna valgrindilla.

```

tommi@LAPTOP-EBRN4IOO:~$ ./reverse_v3 input1.txt input2.txt extra.txt
usage: reverse <input> <output>
tommi@LAPTOP-EBRN4IOO:~$ echo $?
1
tommi@LAPTOP-EBRN4IOO:~$ ./reverse_v3 test1.txt test1.txt
Input and output file must differ
tommi@LAPTOP-EBRN4IOO:~$ echo $?
1
tommi@LAPTOP-EBRN4IOO:~$ ./reverse_v3 does_not_exist.txt
error: cannot open file 'does_not_exist.txt'
tommi@LAPTOP-EBRN4IOO:~$ echo $?
1

```

```

tommi@LAPTOP-EBRN4IOO:~$ valgrind --leak-check=full ./reverse_v3 input.txt
==29822== Memcheck, a memory error detector
==29822== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==29822== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==29822== Command: ./reverse_v3 input.txt
==29822==
a test
is
this
hello
==29822==
==29822== HEAP SUMMARY:
==29822==    in use at exit: 0 bytes in 0 blocks
==29822==   total heap usage: 9 allocs, 9 frees, 5,801 bytes allocated
==29822==
==29822== All heap blocks were freed -- no leaks are possible
==29822==
==29822== For lists of detected and suppressed errors, rerun with: -s
==29822== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
tommi@LAPTOP-EBRN4IOO:~$

```

Tulokset:

- Versio 1: toimii peruslogiikaltaan, mutta ei hallitse virhetilanteita tai suuria muistiongelmia.
- Versio 2: käsittelee useampia argumentteja ja tiedostovirheitä, mutta ei tee kattavia muistivirhetarkistuksia.
- Versio 3: läpäisee kaikki projektin vaatimukset (syöte==tuloste, liian monet argumentit, tiedostonavausvirheet, malloc-virheet jne.). Valgrind-analyysi ei raportoi muistivuotoja, joten tulokset ovat onnistuneet.

Lähteet ja viittaukset

fopen, getline, fclose

- Käytetty tietolähteenä tiedoston avaamiseen, lukemiseen (getline) ja sulkemiseen.
- Fopen ja fclose varten käytetty apuna <https://www.geeksforgeeks.org/what-is-the-difference-between-fopen-and-fclose-functions-in-php/>
- Getline varten käytetty apuna: <https://c-for-dummies.com/blog/?p=1112>

malloc, realloc, free

- Käytetty dynaamisen muistinhallinnan dokumentaatioon.
- Malloc, realloc ja free varten käytetty apuna <https://www.geeksforgeeks.org/dynamic-memory-allocation-in-c-using-malloc-calloc-free-and-realloc/>

Stack Overflow

- Yleisiä esimerkkejä rivinluvusta, muistinhallinnan virheiden debuggaamisesta, jne.
- Getline varten käytetty apuna <https://stackoverflow.com/questions/1744665/need-help-with-getline>
- Muistinhallintaa varten käytetty apuna <https://stackoverflow.com/questions/25116678/dynamic-memory-allocation-c-programming>
- Valgrindia varten käytetty apuna <https://stackoverflow.com/questions/5134891/how-do-i-use-valgrind-to-find-memory-leaks>

Valgrind

- Muistivuotojen ja virheiden paikallistamiseen.
- Komento: valgrind --leak-check=full ./reverse input.txt output.txt
- Valgrindia varten käytetty apuna <https://valgrind.org/docs/manual/quick-start.html>

ChatGPT o1

- Käytetty apuna koodin debuggaamisessa, mutta lopulliset ratkaisut ja testaukset on tehty manuaalisesti <https://chatgpt.com/>

Projekti 2

Perustiedot

- Projektin numero: 2
- Projektin nimi: Unix Utilities
- Palautuspäivämäärä: 19.03.2025

Projektin kuvaus

Tämän projektin tavoitteena oli luoda 4 eri funktiota, my-cat, my-grep, my-zip ja my-unzip. Näiden funktioiden ideana on matkia Linux:in valmiina tarjottavia funktioita, kuten cat, grep, zip, unzip. Funktiot oli testattu eri tiedostoilla ja eri syötteillä varmistaakseen funktioiden toimivuuden.

Toteutuksen dokumentaatio

Toteutus my-cat funktiolle:

My-cat on yksinkertainen ohjelma, joka lukee käyttäjän antamaan tiedoston sisällön ja tulostaa sen näytölle. Ohjelma avaa tiedoston fopen()-funktioilla, tarkistaa virheilmoitukset, lukee tiedoston rivit fgets()-funktioilla ja sulkee tiedoston. Ohjelma tukee useamman tiedoston käsittelyä.

Käytetyt teknologiat ja ratkaisu:

- Kirjastot: <stdio.h>, <string.h>, <errno.h>
- Globaali muuttuja: Buffer 2048
- Funktiot: fopen() -> tiedoston luku
- fgets() -> tiedoston lukeminen ja bufferiin sijoittaminen
- fclose() -> tiedoston sulkeminen.

```

C my-cat.c > printFileContent(const char *)
1  #include <stdio.h>
2  #include <string.h>
3  #include <errno.h>
4
5  // Inspiration for the code, and copied some of the approaches here: https://www.geeksforgeeks.org/build-your-own-cat-command-in-c-for-linux/
6  // Usage of fgets was learned from here and utilized via example: https://www.geeksforgeeks.org/fgets-function-in-c/
7
8  #define BUFFER 2048 // Define the buffer size
9
10 // Opens a file and prints the content into standard output
11 int printFileContent(const char *file)
12 {
13     FILE *files = fopen(file, "r");
14
15     if (files == NULL)
16     {
17         fprintf(stderr, "my-cat: cannot open file '%s': ", file);
18         perror("");
19         return 1;
20     }
21     // Declare buffer
22
23     char buffer[BUFFER];
24
25     // Read from file until the end, insert into buffer
26     while (fgets(buffer, BUFFER, files) != NULL)
27     {
28         // Print the line into the output
29         printf("%s", buffer);
30     }
31
32     // Close the file
33     fclose(files);
34
35     return 0;
36 }
37
38
39
40 int main(int argc, char *argv[])
41 {
42     // Check for all arguments
43     if (argc < 2)
44     {
45         return 0;
46     }
47
48     int exit_code = 0;
49
50     // Loop through all files that are given, skipping ofcourse the argv[0]
51     for (int i = 1; i < argc; i++)
52     {
53         if (printFileContent(argv[i]) != 0)
54         {
55             exit_code = 1; // If any file fails, return error status
56         }
57     }
58
59     return exit_code;
60 }

```

Testaus ja tulokset

Lukeminen suoraan tiedostosta:

- Luodaan tiedosto ja luetaan luotu tiedosto.
- Ohjelman pitäisi palauttaa 0.

```
tommi@LAPTOP-EBRN4IOO:~$ echo -e "hello\nworld\nC programming" > test1.txt
tommi@LAPTOP-EBRN4IOO:~$ ./my-cat test1.txt
hello
world
C programming
tommi@LAPTOP-EBRN4IOO:~$ echo $?
0
```

Lukeminen kahdesta tiedostosta:

- Otetaan toinen tiedosto mukaan ja luetaan 2 tiedostoa.
- Kokeillaan tiedostoja "test2a.txt" ja "test2b.txt"
- Ohjelman pitäisi palauttaa 0.

```
tommi@LAPTOP-EBRN4IOO:~$ echo -e "File 1 line 1\nFile 1 line 2" > test2a.txt
tommi@LAPTOP-EBRN4IOO:~$ echo -e "File 2 line 1\nFile 2 line 2" > test2b.txt
tommi@LAPTOP-EBRN4IOO:~$ ./my-cat test2a.txt test2b.txt
File 1 line 1
File 1 line 2
File 2 line 1
File 2 line 2
tommi@LAPTOP-EBRN4IOO:~$ echo $?
0
```

Virheelliset syötteet:

- Testataan syöttämällä virheellinen tiedosto, ohjelman pitäisi palauttaa 1.
- Testataan ilman, että syötetään tiedostoa lainkaan, ohjelman pitäisi palauttaa 0.

```
tommi@LAPTOP-EBRN4IOO:~$ ./my-cat does_not_exist.txt
my-cat: cannot open file 'does_not_exist.txt': No such file or directory
tommi@LAPTOP-EBRN4IOO:~$ echo $?
1
tommi@LAPTOP-EBRN4IOO:~$ ./my-cat test1.txt does_not_exist.txt
hello
world
C programming
my-cat: cannot open file 'does_not_exist.txt': No such file or directory
tommi@LAPTOP-EBRN4IOO:~$ echo $?
1
```

```
tommi@LAPTOP-EBRN4IOO:~$ ./my-cat
tommi@LAPTOP-EBRN4IOO:~$ echo $?
0
```

Toteutus my-grep funktiolle:

My-grep on ohjelma, joka etsii käyttäjän määrittelemän hakutermiä tiedostoista tai standardisyytöstä. Ohjelma lukee syötteen `getline()`-funktioilla, käyttää `strstr()`-funktiota hakutermiä löytämiseksi riviltä (siis standardisyyttestä tai tiedostosta) ja tulostaa löytyneet rivit käyttäjälle. Ohjelma tukee yksittäisiä ja useita tiedostoja ja palauttaa virheilmoituksia, jos on tarvetta.

Käytetyt teknologiat ja ratkaisu:

C-ohjelmointikieli

- Kirjastot: `<stdio.h>`, `<stdlib.h>`, `<string.h>`, `<unistd.h>`

Funktiot

- `getline()`: luetaan rivit, tiedostosta/standardisyyttestä, sekä käytetään dynaamisessa muistinhallinnassa eli määritellään muistin koko bufferille.
- `strstr()`: löydetään parit hakutermille.
- `free()`: vapautetaan dynaamisesti allokoitu muisti.
- `fprintf()`: tulostus
- `fopen()`: tiedoston avaaminen
- `fclose()`: tiedoston sulkeminen

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <unistd.h>
5
6  // Inspiration for the code is found here, on how to implement the grep command: https://www.quora.com/How-do-you-write-a-C-program-to-mimic-grep-command-in-Linux
7  // For the getline implementation in the code this was used: https://c-for-dummies.com/blog/?p=1112
8  // Usage of strstr was learned from here: https://www.geeksforgeeks.org/strstr-in-ccpp/
9
10 // Function to read file and search for the given searchTerm
11 void ReadAndFindFileContent(FILE *fp, const char *searchTerm)
12 {
13     char *line = NULL; // Pointer to store dynamically allocated line
14     size_t len = 0;    // Size of the allocated buffer. (getline will determine the size)
15     ssize_t nread;     // Number of character to read
16
17     // Read each line
18     // Getline will return -1 if no lines.
19     while ((nread = getline(&line, &len, fp)) != -1)
20     {
21         if (strstr(line, searchTerm) != NULL)
22         {
23             // strstr check if the searchTerm existis within the line
24             printf("%s", line); // Prints the line
25         }
26         free(line); // Free the dynamically allocated memory.
27     }
28
29 int main(int argc, char *argv[])
30 {
31     // Check if there is enough arguments
32     if (argc < 2)
33     {
34         fprintf(stderr, "my-grep: searchterm [file ...]\n");
35         return 1;
36     }
37     // Search term itself
38     const char *searchTerm = argv[1];
39     int exit_code = 0;
40     // If there is searchTerm but no file, read from standard input
41     if (argc == 2)
42     {
43         // Read from standard input.
44         ReadAndFindFileContent(stdin, searchTerm);
45     }
46     else
47     {
48         // Process each file provided on the command line.
49         for (int i = 2; i < argc; i++)
50         {
51             FILE *files = fopen(argv[i], "r");
52             if (files == NULL)
53             {
54                 fprintf(stderr, "my-grep: cannot open file '%s'\n", argv[i]);
55                 exit_code = 1; // Continue to the next file but return 1 at the end
56                 continue;
57             }
58             // Call the function to search for term
59             ReadAndFindFileContent(files, searchTerm);
60             fclose(files); // Close
61         }
62     }
63     return exit_code;
64 }
65

```

Testaus ja tulokset:

Testitiedostot:

sanat_15.txt

```
1 sana;sanaluokka;taso;aikaleima (dd.mm.yyyy HH:MM:SS)
2 about;5;1;27-01-2022 04:42:47
3 about;4;1;19-06-2022 04:56:52
4 above;5;1;09-05-2022 13:45:07
5 above;4;1;10-03-2022 23:00:21
6 few;7;1;15-03-2022 09:56:18
7 able;3;2;10-03-2022 16:48:28
8 abroad;4;2;26-04-2022 12:11:19
9 absolutely;4;3;30-05-2022 07:20:56
10 academic;3;3;04-03-2022 06:05:20
11 absolute;3;4;26-05-2022 14:16:07
12 academic;1;4;14-07-2022 08:26:54
13 programming;1;4;18-01-2022 16:28:25
14 worthy;3;5;31-10-2022 03:17:43
15 yield;1;5;04-05-2022 05:19:39
16 yield;2;5;30-06-2022 00:19:23
```

sanat.txt

```
1 sana;sanaluokka;taso;aikaleima (dd.mm.yyyy HH:MM:SS)
2 about;5;1;27-01-2022 04:42:47
3 about;4;1;19-06-2022 04:56:52
4 above;5;1;09-05-2022 13:45:07
5 above;4;1;10-03-2022 23:00:21
6 across;5;1;18-06-2022 09:22:12
7 across;4;1;03-06-2022 01:54:27
8 action;1;1;05-11-2022 16:57:53
9 activity;1;1;22-10-2022 20:37:05
10 actor;1;1;23-10-2022 12:49:33
11 actress;1;1;07-09-2022 19:49:14
12 add;2;1;24-10-2022 15:34:22
13 address;1;1;19-11-2022 12:01:03
14 adult;1;1;18-05-2022 11:56:16
15 advice;1;1;05-12-2022 17:56:31
16 afraid;3;1;13-12-2022 19:30:14
17 after;5;1;11-05-2022 22:12:50
18 afternoon;1;1;26-09-2022 19:05:48
19 again;4;1;18-11-2022 09:44:52
20 age;1;1;11-06-2022 22:19:19
21 ago;4;1;26-05-2022 15:40:22
22 agree;2;1;12-12-2022 14:15:58
23 air;1;1;18-03-2022 02:21:52
24 airport;1;1;28-07-2022 08:32:39
25 all;7;1;26-06-2022 20:14:36
26 all;8;1;23-09-2022 04:12:47
27 also;4;1;12-09-2022 03:16:35
28 always;4;1;28-08-2022 18:45:31
29 amazing;3;1;09-07-2022 03:17:11
30 and;9;1;06-05-2022 12:16:29
31 angry;3;1;28-11-2022 07:18:36
32 animal;1;1;16-11-2022 10:46:50
33 another;7;1;25-06-2022 03:15:18
34 another;8;1;21-01-2022 05:59:55
35 answer;1;1;21-06-2022 06:53:48
36 answer;2;1;21-06-2022 17:21:10
37 any;7;1;24-02-2022 21:07:05
38 any;8;1;20-06-2022 00:42:16
39 anyone;8;1;04-08-2022 03:27:01
40 anything;8;1;31-05-2022 08:46:44
41 apartment;1;1;29-10-2022 00:58:21
42 apple;1;1;12-03-2022 16:20:52
43 april;1;1;14-06-2022 07:51:09
44 area;1;1;23-12-2022 05:08:59
45 arm;1;1;17-04-2022 20:18:44
46 around;5;1;19-03-2022 19:52:57
47 around;4;1;25-07-2022 16:33:41
48 arrive;2;1;18-06-2022 05:45:00
49 art;1;1;22-06-2022 19:00:12
50 article;1;1;18-09-2022 03:17:16
```

Lukeminen suoraan tiedostosta:

- Luetaan yksi testitiedostoista

```
• tommy@LAPTOP-EBRN4IOO:~$ ./my-grep about sanat_15.txt
about;5;1;27-01-2022 04:42:47
about;4;1;19-06-2022 04:56:52
• tommy@LAPTOP-EBRN4IOO:~$ echo $?
0
```

Lukeminen kahdesta tiedostosta:

- Otetaan toinen tiedosto mukaan ja luetaan 2 tiedostoa.

```
• tommy@LAPTOP-EBRN4IOO:~$ ./my-grep about sanat_15.txt sanat.txt
about;5;1;27-01-2022 04:42:47
about;4;1;19-06-2022 04:56:52
about;5;1;27-01-2022 04:42:47
about;4;1;19-06-2022 04:56:52
• tommy@LAPTOP-EBRN4IOO:~$ echo $?
0
```

Standardisyytteestä lukeminen:

- Luetaan standardisyytteestä

```
• tommy@LAPTOP-EBRN4IOO:~$ cat sanat_15.txt | ./my-grep about
about;5;1;27-01-2022 04:42:47
about;4;1;19-06-2022 04:56:52
• tommy@LAPTOP-EBRN4IOO:~$ echo $?
0
```

Virhetilanteet:

- Testataan syöttämällä virheellinen tiedosto, ohjelman pitäisi palauttaa 1.
- Testataan ilman, että syötetään tiedostoa lainkaan, ohjelman pitäisi palauttaa 1.

```
• tommy@LAPTOP-EBRN4IOO:~$ ./my-grep word does_not_exist.txt
my-grep: cannot open file 'does_not_exist.txt'
• tommy@LAPTOP-EBRN4IOO:~$ echo $?
1
```

```
• tommy@LAPTOP-EBRN4IOO:~$ ./my-grep
my-grep: searchterm [file ...]
• tommy@LAPTOP-EBRN4IOO:~$ echo $?
1
```

Toteutus my-zip funktiolle:

My-zip on tiedostojen pakkausohjelma, joka käyttää run-length-encoding (RLE)-menetelmää.

Lukee tiedostot merkki merkiltä, laskee peräkkäisten merkkien lukumäärän ja kirjoittaa pakatun datan binaarimuodossa: 4 tavua luku, sitten yksi merkki. Ohjelma käsittelee yksittäisiä ja monia tiedostoja. Jos on monta tiedostoa, ohjelma käsittelee niitä yhtenä isona tiedostona.

Käytetyt teknologiat ja ratkaisu:

Kirjastot:

- <stdio.h>, <Stdlib.h>

Funktiot:

- fgetc() käytetty tiedoston lukemiseen merkki merkiltä.
- fwrite() tiedostoon kirjoittaminen
- fopen, fclose tiedoston avaaminen/sulkeminen.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  //Run length encoding was learned from this: https://www.geeksforgeeks.org/run-length-encoding/
5  // For the fgetc, https://www.geeksforgeeks.org/fgetc-function-in-c/
6  // Found how to write into the file: https://www.geeksforgeeks.org/fwrite-in-c/
7
8  //Helper function, reads a file character by character and updates the run length encoding state
9  void processFile(FILE *fp, int *runCount, int *currentChar, int *firstChar) {
10     int c; //Variable to hold the current character
11
12     //Loop through file
13     while ((c = fgetc(fp)) != EOF) {
14         if (*firstChar) {
15             //First character to be processed
16             //set the character and set run to 1
17             *currentChar = c;
18             *runCount = 1;
19             *firstChar = 0; //Mark that at least one character is processed
20         } else {
21             if (c == *currentChar) {
22                 //If c is same as current character increment the run count
23                 (*runCount)++;
24             } else {
25
26                 //When new character is spotted
27                 //Write the current run
28                 // Write the current run as a 4-byte integer and
29                 //and current char as a single byte
30                 fwrite(runCount, sizeof(int), 1, stdout);
31                 fwrite(currentChar, sizeof(char), 1, stdout);
32
33                 // Start a new run.
34                 *currentChar = c;
35                 *runCount = 1;
36             }
37         }
38     }
39 }
```



```

int main(int argc, char *argv[]) {
    //Check that file exist
    if (argc < 2) {
        fprintf(stderr, "my-zip: file1 [file2 ...]\n");
        return 1;
    }

    //Run length encoding
    int runCount = 0;    // Count of consecutive occurrences.
    int currentChar = 0; // The current character.
    int firstChar = 1;   // Flag: 1 means no character processed yet.
    int ErrorOccured = 0;

    // Process each file as a continuous stream.
    //Files are processed as one stream of text
    for (int i = 1; i < argc; i++) {
        FILE *fp = fopen(argv[i], "r"); //Text mode
        if (fp == NULL) {
            fprintf(stderr, "my-zip: cannot open file %s\n", argv[i]);
            ErrorOccured = 1;
            continue;
        }

        //Process the file
        processFile(fp, &runCount, &currentChar, &firstChar);
        fclose(fp);
    }

    // After processing all files, flush out the final run (if any).

    //Flush out the final run. This will ensure that last run is written.
    if (!firstChar) {
        fwrite(&runCount, sizeof(int), 1, stdout);
        fwrite(&currentChar, sizeof(char), 1, stdout);
    }

    return ErrorOccured ? 1 : 0;
}

```

Testaus ja tulokset:

Testaan yksittäinen tiedosto:

- Luodaan tiedosto, katsotaan tiedostonkoko ja katsotaan koko uudestaan zippauksen jälkeen.
- Kuten huomataan testistä, tiedostonkoko pienenee.

```

• tommy@LAPTOP-EBRN4IOO:~$ gcc -o my-zip my-zip.c -Wall -Werror
• tommy@LAPTOP-EBRN4IOO:~$ echo "aaaaaaaaaaaaaaaaabbbbbbbbbbcccccccccccccccdrrrrrrrrrrrrrrrrr" > MyZipTest.txt
• tommy@LAPTOP-EBRN4IOO:~$ ls -l MyZipTest.txt && cat MyZipTest.txt
-rw-r--r-- 1 tommy tommy 74 Mar 19 12:55 MyZipTest.txt
aaaaaaaaaaaaaaaaabbbbbbbbbbcccccccccccccccdrrrrrrrrrrrrrrrrr
• tommy@LAPTOP-EBRN4IOO:~$ ./my-zip MyZipTest.txt > MyZipTestingZip.z
• tommy@LAPTOP-EBRN4IOO:~$ ls -l MyZipTestingZip.z
-rw-r--r-- 1 tommy tommy 25 Mar 19 12:55 MyZipTestingZip.z
• tommy@LAPTOP-EBRN4IOO:~$ cat MyZipTestingZip.z
abcd
• tommy@LAPTOP-EBRN4IOO:~$ hexdump -C MyZipTestingZip.z
00000000 15 00 00 00 61 10 00 00 00 62 12 00 00 00 63 12 |...a....b....c.|
00000010 00 00 00 64 01 00 00 00 0a                        [...d.....]
00000019
• tommy@LAPTOP-EBRN4IOO:~$ echo $?
0

```

Testataan useampaa tiedostoa:

- Luodaan toinen tiedosto ja zipataan molemmat tiedostot yhteen pakettiin.
- Kuten huomataan, yhdistetyn tiedoston koko on suurempi, mutta silti pienempi kuin yksittäinen tiedosto (esimerkiksi MyZipTest.txt)

```

• tommyi@LAPTOP-EBRN4IOO:~$ echo "aaaaaaaaaaaaaaaaabbbbbbbbbbbbbbccccccccccccccccccdddddcccccccccccccccc" > MyZipTest2.txt
  -l MyZipTest2.txt && cat MyZipTest2.txt
• tommyi@LAPTOP-EBRN4IOO:~$ ls -l MyZipTest2.txt && cat MyZipTest2.txt
-rw-r--r-- 1 tommyi tommyi 74 Mar 19 12:57 MyZipTest2.txt
aaaaaaaaaaaaaaaaabbbbbbbbbbbbbbccccccccccccccccccdddddcccccccccccccccc
• tommyi@LAPTOP-EBRN4IOO:~$ ./my-zip MyZipTest.txt MyZipTest2.txt > MyZipTwoFiles.z
• tommyi@LAPTOP-EBRN4IOO:~$ ls -l MyZipTwoFiles.z && cat MyZipTwoFiles.z
-rw-r--r-- 1 tommyi tommyi 50 Mar 19 12:58 MyZipTwoFiles.z
abcd
abcd
• tommyi@LAPTOP-EBRN4IOO:~$ hexdump -C MyZipTwoFiles.z
00000000  15 00 00 00 61 10 00 00 00 62 12 00 00 00 63 12 |...a...b...c.|
00000010  00 00 00 64 01 00 00 00 0a 15 00 00 00 61 10 00 |...d.....a..|
00000020  00 00 62 12 00 00 00 63 12 00 00 00 64 01 00 00 |..b....c...d...|
00000030  00 0a                                     |..|
00000032
• tommyi@LAPTOP-EBRN4IOO:~$ echo $?
0

```

Tyhjällä tiedostolla testaaminen:

- Kokeillaan zipata tyhjä tiedosto.

```

• elias@DESKTOP-C6U5EQQ:~$ touch TyhjaZip.txt
• elias@DESKTOP-C6U5EQQ:~$ ll TyhjaZip.txt
-rw-r--r-- 1 elias elias 0 Mar 13 16:21 TyhjaZip.txt
• elias@DESKTOP-C6U5EQQ:~$ ./my-zip TyhjaZip.txt > TyhjaZip.z
• elias@DESKTOP-C6U5EQQ:~$ ll TyhjaZip.z
-rw-r--r-- 1 elias elias 0 Mar 13 16:23 TyhjaZip.z
• elias@DESKTOP-C6U5EQQ:~$

```

Virhetilanteet:

- Testataan syöttämällä virheellinen tiedosto
- Testataan ilman, että syötetään tiedostoa lainkaan.

```
tommi@LAPTOP-EBRN4IOO:~$ ./my-zip EIOLEOLEMASSA.txt > EIOLEPAKATTU.z
my-zip: cannot open file EIOLEOLEMASSA.txt
tommi@LAPTOP-EBRN4IOO:~$ ./my-zip
my-zip: file1 [file2 ...]
tommi@LAPTOP-EBRN4IOO:~$ echo $?
1
tommi@LAPTOP-EBRN4IOO:~$
```

Toteutus my-unzip funktiolle:

My-unzip on tiedostojen purkuohjelma, joka palauttaa my-zip -ohjelmalla pakatun datan alkuperäiseen muotoon. Se lukee binääritiedoston 5 tavun lohkoina, joissa on 4 tavua lukuja ja 1 tavu merkkiä, ja tulostaa ne. Ohjelma voi lukea yksittäisiä tai useampia tiedostoja.


Käytetyt teknologiat ja ratkaisu:

Kirjastot:

- <stdio.h>, <stdlib.h>

Funktiot:

- fread(): tiedoston lukemiseen.
- putchar(): käytetty tiedoston kirjoittamiseen.

```
C my-unzip.c >  decompressFile(FILE *)
1  #include <stdio.h>
2  #include <stdlib.h>
3
4
5  //To decompress the file, this guide for fread was used: https://www.geeksforgeeks.org/fread-function-in-c/
6  //To print out, found out about this function (putchar) from here: https://www.geeksforgeeks.org/putchar-function-in-c/
7
8
9  // decompressFile reads a compressed file from files, then writes out the decompressed text.
10 void decompressFile(FILE *files) {
11     int count;    // Run length (4 bytes)
12     char ch;      // The character (1 byte)
13
14     // Loop through 5 byte blocks
15     // fread will return the number of elements that have been read
16     while (fread(&count, sizeof(int), 1, files) == 1) {
17         //Read the associated character
18         if (fread(&ch, sizeof(char), 1, files) != 1) {
19             break; //exit the loop, if not possible to read
20         }
21         // Output the character 'count' times.
22         for (int i = 0; i < count; i++) {
23             putchar(ch);
24         }
25     }
26 }
27
28
29
30 int main(int argc, char *argv[]) {
31     if (argc < 2) {
32         fprintf(stderr, "my-unzip: file1 [file2 ...]\n");
33         return 1;
34     }
35
36     int ErrorOccured=0;
37
38     // Process each file.
39     for (int i = 1; i < argc; i++) {
40         FILE *files = fopen(argv[i], "rb"); // Open in binary mode.
41         if (files == NULL) {
42             fprintf(stderr, "my-unzip: cannot open file %s\n", argv[i]);
43             ErrorOccured=1;
44             continue;
45         }
46         //Decompress the file
47         decompressFile(files);
48         fclose(files);
49     }
50
51     return ErrorOccured ? 1 : 0;
52 }
```

Testaus ja tulokset:

Yksittäisen tiedoston purku:

- Kokeillaan purkaa yksittäinen tiedosto, joka pakattiin my-zipin avulla.

[illegible]

Kahden tai useamman tiedoston purku:

- Kokeillaan purkaa my-zipin avulla pakattu paketti, jossa on kaksi tiedostoa yhdistettynä

```
elias@DESKTOP-C6U5EQQ:~$ ./my-unzip MyZipTwoFiles.z  
aaaaaaaaabbbbbbbbbbccccccccccccccccccddddd  
aaaaaaaaaaaaaaaaaaaaabbbbbbbbbbccccccccccccccccccddddd
```

Virhetilanteet:

- Testataan syöttämällä virheellinen tiedosto
- Testataan ilman, että syötetään tiedostoa lainkaan.

```
ⓧ elias@DESKTOP-C6U5EQQ:~$ ./my-unzip EIOLEOLEMASSA.z
my-unzip: cannot open file EIOLEOLEMASSA.z
ⓧ elias@DESKTOP-C6U5EQQ:~$
```

```
ⓧ elias@DESKTOP-C6U5EQQ:~$ ./my-unzip
my-unzip: file1 [file2 ...]
```

Lähteet ja viittaukset

my-cat

- Funktionaalisuutta täältä: <https://www.geeksforgeeks.org/build-your-own-cat-command-in-c-for-linux/>
- fgets() varten käytetty apuna <https://www.geeksforgeeks.org/fgets-function-in-c/>
- ChatGPT o1 Käytetty apuna koodin debuggaamisessa, mutta lopulliset ratkaisut ja testaukset on tehty manuaalisesti <https://chatgpt.com/>
-

my-grep

- Esimerkki grep komennosta: <https://www.quora.com/How-do-you-write-a-C-program-to-mimic-grep-command-in-Linux>
- getline() varten käytetty apuna <https://c-for-dummies.com/blog/?p=1112>
- strstr() varten käytetty apuna <https://www.geeksforgeeks.org/strstr-in-cpp/>
- ChatGPT o1 Käytetty apuna koodin debuggaamisessa, mutta lopulliset ratkaisut ja testaukset on tehty manuaalisesti <https://chatgpt.com/>

my-zip

- Run Length funktiota käytetty hyödyksi ja muokattu: <https://www.geeksforgeeks.org/run-length-encoding/>
- ChatGPT o1 Käytetty apuna koodin debuggaamisessa, mutta lopulliset ratkaisut ja testaukset on tehty manuaalisesti <https://chatgpt.com/>

My-unzip

- fwrite() varten käytetty apuna <https://www.geeksforgeeks.org/fwrite-in-c/>
- fread() varten käytetty apuna <https://www.geeksforgeeks.org/fread-function-in-c/>
- putchar() varten käytetty apuna <https://www.geeksforgeeks.org/putchar-function-in-c/>
- ChatGPT o1 Käytetty apuna koodin debuggaamisessa, mutta lopulliset ratkaisut ja testaukset on tehty manuaalisesti <https://chatgpt.com/>

Projekti 3

Perustiedot

- Projektin numero: 5
- Projektin nimi: Parallel Zip
- Palautuspäivämäärä: 19.3.2025

Projektin kuvaus

Projektin tavoitteena oli luoda pzip työkalu, joka pystyisi toteuttamaan rinnakkaisen pakkausohjelman hyödyntäen RLE:ta. Ohjelma lukee yhden tai useamman tiedoston, yhdistää niiden sisällön muistiin ja jakaa sen useisiin osiin monisäikeistä käsittelyä varten. Jokainen säie käsittelee omaan osuutensa rinnakkain, minkä jälkeen tulokset yhdistetään ja kirjoitetaan binäärimuodossa. Ohjelma käyttää POSIX-säikeitä (pthreads) ja muisitiosoitettuja tiedostoja (mmap) tehokkuuden parantamiseksi.

Toteutuksen dokumentaatio

Käytetyt teknologiat ja ratkaisu:

Kirjastot:

- `<stdio.h>`, `<stdlib.h>`, `<sys/mman.h>`, `<sys/stat.h>`, `<fcntl.h>`, `<unistd.h>`, `<pthread.h>`, `<string.h>`, `<sys/sysinfo.h>`.

Funktiot:

- `fstat()`, metadataa tiedostoista, josta koko on tärkein. Koolla määritetään, paljon dataa täytyy käsitellä
- `mmap()`: Mappaa tiedoston muistialueelle. Siis tiedoston sisältö voidaan lukea taulukosta.
- `munmap()`: Poistaa `mmap()` -funktiolla varattua muistia.
- `pthread_create()`: Käynnistää uuden säikeen kutsuvassa prosessissa
- `pthread_join()`: Odottaa, että säikeen suoritus valmistuu.

- `get_nprocs()`: Palauttaa järjestelmän käytettävissä olevien prosessoreiden määrä

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <sys/mman.h>
4 #include <sys/stat.h>
5 #include <fcntl.h>
6 #include <unistd.h>
7 #include <pthread.h>
8 #include <string.h>
9 #include <sys/sysinfo.h>
10
11
12 //A lot of reference and code implementation constraints was used from this repository: https://github.com/Saggarwal9/Parallel-ZIP/blob/master/pzip.c
13
14
15 typedef struct {
16     char *data;          // Pointer to input data
17     size_t start;        // Start of data segment
18     size_t end;          // End of data segment
19     int *counts;         // Dynamic array to store counts of the RLE
20     char *chars;         // Dynamic array to store corresponding characters (RLE)
21     int num_runs;        // Number of runs produced by this thread
22 } thread_arg_t;

```

```

/*
Run function that compresses a segment of the concatenated data.
Run-Length Encoding overview: https://www.geeksforgeeks.org/run-length-encoding/
POSIX threads: https://man7.org/linux/man-pages/man3/pthread\_create.3.html
*/
void* compress_segment(void *arg) {
    //Initializing variables
    thread_arg_t *targ = (thread_arg_t*) arg;
    size_t i = targ->start;

    // Print thread ID and segment boundaries.
    fprintf(stderr, "Thread %lu: processing segment [%zu, %zu)\n",
        (unsigned long)pthread_self(), targ->start, targ->end);

    //If there is nothing to process, exit
    if (i >= targ->end) {
        targ->num_runs = 0;
        return NULL;
    }

    //Allocate memory for the RLE results
    int capacity = 16;
    targ->counts = malloc(capacity * sizeof(int)); //Allocates memory to store RLE results
    targ->chars = malloc(capacity * sizeof(char)); //Allocates memory to store RLE results
    targ->num_runs = 0; //From start is zero

    //Assume that one character appears
    int run_count = 1;

    //Current character
    char current_char = targ->data[i];
    //Loops via each character.
    for (i = targ->start + 1; i < targ->end; i++) {
        char c = targ->data[i];

        //If matches the previous, increase the count
        if (c == current_char) {
            run_count++;

            //If not store the previous segment
        } else {
            if (targ->num_runs >= capacity) {
                capacity *= 2;
                targ->counts = realloc(targ->counts, capacity * sizeof(int)); //Dynamically store
                targ->chars = realloc(targ->chars, capacity * sizeof(char)); //Dynamically store
            }

            //Store the run-length data, in the arrays
            targ->counts[targ->num_runs] = run_count;
            targ->chars[targ->num_runs] = current_char;
            targ->num_runs++; //New RLE entry is added
            current_char = c; //Start tracking new sequence
            run_count = 1; //Reset the run_count
        }
    }
}

```



```

// Store the final run, if num_run will exceed the capacity, it will mean that arrays are full --> allocate more memory
if (targ->num_runs >= capacity) {
    capacity++; //Increase the capacity
    targ->counts = realloc(targ->counts, capacity * sizeof(int)); //Expand the array
    targ->chars = realloc(targ->chars, capacity * sizeof(char)); //Expand the array
}

//Storing new run,
targ->counts[targ->num_runs] = run_count; //Storing number of times character has appeared in the sequence
targ->chars[targ->num_runs] = current_char; //Index starts at 0, ensure that new run is stored in the correct position
targ->num_runs++; // Run is stored, increase the counter --> next run stored in the next available slot

// Print thread completion with run count.
fprintf(stderr, "Thread %lu: finished with %d runs\n",
        (unsigned long)pthread_self(), targ->num_runs);

return NULL;
}

```

```

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

    //Check command-line for enough arguments (files are needed)
    if (argc < 2) {
        fprintf(stderr, "pzip: file1 [file2 ...]\n");
        exit(1);
    }
    // Compute total size for all input files.
    size_t total_size = 0; //Hold all file data here, from single or multiple files

    //Iterate
    for (int i = 1; i < argc; i++) {
        //Opens the file, with O_RDONLY
        int fd = open(argv[i], O_RDONLY);

        //Check if opening is possible
        if (fd < 0) {
            perror("pzip: cannot open file");
            exit(1);
        }
        struct stat sb;
        if (fstat(fd, &sb) < 0) { //Fetches the file's metadata
            perror("pzip: fstat error"); //Error
            exit(1);
        }

        //Adds the size to total_size
        total_size += sb.st_size;
        close(fd); //Closes the file
    }

    // Allocate a big buffer and copy all files into it.

    char *big_buffer = malloc(total_size); //Large memory block, large enough to hold all file contents
    if (!big_buffer) { //Error with malloc
        perror("pzip: malloc failed");
        exit(1);
    }

    //Load files into the previously allocated memory block
    size_t offset = 0;

```

```

for (int i = 1; i < argc; i++) {
    int fd = open(argv[i], O_RDONLY);
    if (fd < 0) {
        perror("pzip: cannot open file");
        exit(1);
    }
    struct stat sb;
    if (fstat(fd, &sb) < 0) {
        perror("pzip: fstat error");
        exit(1);
    }
    size_t fsize = sb.st_size;
    // Maps the file into memory using mmap()
    // PROT_READ --> Allows reading
    // MAP_PRIVATE --> Changes are not visible to other processes
    char *filedata = mmap(NULL, fsize, PROT_READ, MAP_PRIVATE, fd, 0);
    if (filedata == MAP_FAILED) {
        //Error
        perror("pzip: mmap failed");
        exit(1);
    }
    // Copy file contents into big_buffer.
    memcpy(big_buffer + offset, filedata, fsize); //Copying
    offset += fsize;

    munmap(filedata, fsize); //Unmaps the file, after copying is done
    close(fd);
}

//Determine number of threads based on available processors.
int num_threads = get_nprocs();
if (num_threads > total_size)
    num_threads = total_size; // Do not create more threads than bytes.

// Create threads to process segments of big_buffer.
pthread_t *threads = malloc(num_threads * sizeof(pthread_t)); //Allocate the memory for thread IDs
thread_arg_t *targs = malloc(num_threads * sizeof(thread_arg_t)); //Allocate memory for thread arguments
size_t segment_size = total_size / num_threads; //Dividing buffer into equal-sized segments, for parallel processing
//Creating threads
for (int i = 0; i < num_threads; i++) {
    targs[i].data = big_buffer; //Assign data pointers
    targs[i].start = i * segment_size; //Calculate the start index for the thread
    targs[i].end = (i == num_threads - 1) ? total_size : (i + 1) * segment_size; //Calculate the end index of the thread
    targs[i].counts = NULL; //Initialize the result storage
    targs[i].chars = NULL; // Initialize the result storage
    targs[i].num_runs = 0; // Initialize the result storage
    pthread_create(&threads[i], NULL, compress_segment, &targs[i]); //Create the thread
}

// Thread synchronization with pthread_join
for (int i = 0; i < num_threads; i++) {
    pthread_join(threads[i], NULL); //pthread_join blocks execution until corresponding thread completes
    //NULL means that return value from compress_segment is not returning (store the result in the targs[i])
}

// Allocate memory for merging.
//This block is merging the results from RLE, from multiple threads
int merged_capacity = 128; //Initially 128, can grow dynamically
int merged_count = 0; //Track of number stored in runs
int *merged_counts = malloc(merged_capacity * sizeof(int)); //Run lengths
char *merged_chars = malloc(merged_capacity * sizeof(char)); //Characters

```

```

//Iterate over threads
for (int i = 0; i < num_threads; i++) { //Iteration over each thread's result
    for (int j = 0; j < targs[i].num_runs; j++) { //Iteration over each RLE result inside the thread
        int count = targs[i].counts[j]; //Extract the count and characters
        char ch = targs[i].chars[j];
        // If previous run exists and characters match, combine them.
        if (merged_count > 0 && merged_chars[merged_count - 1] == ch) { //Check if last character matches ch, if true, merges the runs by adding count to the last stored run
            merged_counts[merged_count - 1] += count; //if true, merges the runs by adding count to the last stored run
        }
        //If previous character is different, a new run is stored
        else {
            //Check if enough memory
            if (merged_count == merged_capacity) {
                merged_capacity *= 2; //Expand memory
                merged_counts = realloc(merged_counts, merged_capacity * sizeof(int)); //Expand memory
                merged_chars = realloc(merged_chars, merged_capacity * sizeof(char)); //Expand memory
            }
            //New run is stored
            merged_counts[merged_count] = count;
            merged_chars[merged_count] = ch;
            merged_count++; //Track number of stored runs
        }
    }
}

//Free allocated memory, which were allocated by threads
free(targs[i].counts);
free(targs[i].chars);
}

// Write merged runs as binary output: each run is a 4-byte integer followed by a 1-byte character.
for (int i = 0; i < merged_count; i++) {
    fwrite(&merged_counts[i], sizeof(int), 1, stdout);
    fwrite(&merged_chars[i], sizeof(char), 1, stdout);
}

// Free all allocated memory
free(merged_counts);
free(merged_chars);
free(threads);
free(targs);
free(big_buffer);

return 0;
}

```

Testaus ja tulokset:

Laajan tiedoston pakkaus:

- Pakataan 100 MB tiedosto

```
elias@DESKTOP-C6U5EQQ:~$ gcc -o my-pzip my-pzip.c -Wall -Werror -pthread
```

```
elias@DESKTOP-C6U5EQQ:~$ head -c 100000000 </dev/zero | tr '\0' 'A' > testCompressFile.txt
elias@DESKTOP-C6U5EQQ:~$ ll testCompressFile.txt
-rw-r--r-- 1 elias elias 100000000 Mar 17 14:14 testCompressFile.txt
elias@DESKTOP-C6U5EQQ:~$ ./my-pzip testCompressFile.txt > CompressedFile.z
Thread 140216532260608: processing segment [0, 5000000)
Thread 140216515475200: processing segment [10000000, 15000000)
Thread 140216523867904: processing segment [5000000, 10000000)
Thread 140216507082496: processing segment [15000000, 20000000)
Thread 140216498689792: processing segment [20000000, 25000000)
Thread 140216473511680: processing segment [35000000, 40000000)
Thread 140216490297088: processing segment [25000000, 30000000)
Thread 140216119588608: processing segment [45000000, 50000000)
Thread 140215985370880: processing segment [50000000, 55000000)
Thread 140216094410496: processing segment [65000000, 70000000)
Thread 140216102803200: processing segment [60000000, 65000000)
Thread 140216086017792: processing segment [70000000, 75000000)
Thread 140216111195904: processing segment [55000000, 60000000)
Thread 140216077625088: processing segment [75000000, 80000000)
Thread 140216069232384: processing segment [80000000, 85000000)
Thread 140215834367744: processing segment [90000000, 95000000)
Thread 140215842760448: processing segment [85000000, 90000000)
Thread 140215825975040: processing segment [95000000, 100000000)
Thread 140216465118976: processing segment [40000000, 45000000)
Thread 140216481904384: processing segment [30000000, 35000000)
Thread 140216523867904: finished with 1 runs
Thread 140216507082496: finished with 1 runs
Thread 140216498689792: finished with 1 runs
Thread 140216515475200: finished with 1 runs
Thread 140216532260608: finished with 1 runs
Thread 140216473511680: finished with 1 runs
Thread 140216077625088: finished with 1 runs
Thread 140216481904384: finished with 1 runs
Thread 140215842760448: finished with 1 runs
Thread 140215834367744: finished with 1 runs
Thread 140216119588608: finished with 1 runs
Thread 140215825975040: finished with 1 runs
Thread 140216490297088: finished with 1 runs
Thread 140216111195904: finished with 1 runs
Thread 140216465118976: finished with 1 runs
Thread 140216069232384: finished with 1 runs
Thread 140216094410496: finished with 1 runs
Thread 140215985370880: finished with 1 runs
Thread 140216102803200: finished with 1 runs
Thread 140216086017792: finished with 1 runs
elias@DESKTOP-C6U5EQQ:~$ ll CompressedFile.z
-rw-r--r-- 1 elias elias 5 Mar 17 14:15 CompressedFile.z
```

```
elias@DESKTOP-C6U5EQQ:~$ hexdump -C CompressedFile.z
00000000  00 e1 f5 05 41 |...A|
00000005
```

Laajan tiedoston purku:

- Kokeillaan purkaa pakattu iso tiedosto

```
• elias@DESKTOP-C6U5EQQ:~$ ./my-unzip CompressedFile.z > UnzipCompressedFile.txt
```

```
• elias@DESKTOP-C6U5EQQ:~$ ll UnzipCompressedFile.txt  
-rw-r--r-- 1 elias elias 100000000 Mar 17 14:19 UnzipCompressedFile.txt
```

Kahden laajan tiedoston pakkaaminen:

- Kokeillaan pakata kaksi isoa tiedostoa samanaikaisesti

```
• elias@DESKTOP-C6U5EQQ:~$ head -c 5000000 </dev/zero | tr '\0' 'AAAAABBBBBBCCCCCCCC' > largeFile1.txt  
• elias@DESKTOP-C6U5EQQ:~$ ll largeFile1.txt  
-rw-r--r-- 1 elias elias 50000000 Mar 17 15:05 largeFile1.txt  
• elias@DESKTOP-C6U5EQQ:~$ head -c 50000000 </dev/zero | tr '\0' 'XXXXXXXXYYYYZZZZZZZZ' > largeFile2.txt  
• elias@DESKTOP-C6U5EQQ:~$ ll largeFile2.txt  
-rw-r--r-- 1 elias elias 50000000 Mar 17 15:05 largeFile2.txt  
• elias@DESKTOP-C6U5EQQ:~$ ./my-pzip largeFile1.txt largeFile2.txt > CombinedFiles.z  
Thread 140541791184640: processing segment [0, 5000000)  
Thread 140541782791936: processing segment [5000000, 10000000)  
Thread 140541766006528: processing segment [15000000, 20000000)  
Thread 140541774399232: processing segment [10000000, 15000000)  
Thread 140541623396096: processing segment [20000000, 25000000)  
Thread 140541749221120: processing segment [30000000, 35000000)  
Thread 140541757613824: processing segment [25000000, 30000000)  
Thread 140541740828416: processing segment [35000000, 40000000)  
Thread 140541664687872: processing segment [40000000, 45000000)  
Thread 140541647902464: processing segment [50000000, 55000000)  
Thread 140541656295168: processing segment [45000000, 50000000)  
Thread 140541615003392: processing segment [60000000, 65000000)  
Thread 140541639509760: processing segment [55000000, 60000000)  
Thread 140541606610688: processing segment [65000000, 70000000)  
Thread 140540859381504: processing segment [70000000, 75000000)  
Thread 140540842596096: processing segment [80000000, 85000000)  
Thread 140540825810688: processing segment [90000000, 95000000)  
Thread 140540834203392: processing segment [85000000, 90000000)  
Thread 140540850988800: processing segment [75000000, 80000000)  
Thread 140540817417984: processing segment [95000000, 100000000)  
Thread 140541774399232: finished with 1 runs  
Thread 140541766006528: finished with 1 runs  
Thread 140541664687872: finished with 1 runs  
Thread 140541782791936: finished with 1 runs  
Thread 140541791184640: finished with 1 runs  
Thread 140541623396096: finished with 1 runs  
Thread 140541757613824: finished with 1 runs  
Thread 140541749221120: finished with 1 runs  
Thread 140541647902464: finished with 1 runs  
Thread 140541615003392: finished with 1 runs  
Thread 140541656295168: finished with 1 runs  
Thread 140541639509760: finished with 1 runs  
Thread 140540842596096: finished with 1 runs  
Thread 140541606610688: finished with 1 runs  
Thread 140541740828416: finished with 1 runs  
Thread 140540859381504: finished with 1 runs  
Thread 140540850988800: finished with 1 runs  
Thread 140540817417984: finished with 1 runs  
Thread 140540834203392: finished with 1 runs  
Thread 140540825810688: finished with 1 runs  
• elias@DESKTOP-C6U5EQQ:~$ ll CombinedFiles.z  
-rw-r--r-- 1 elias elias 10 Mar 17 15:06 CombinedFiles.z
```

```

• elias@DESKTOP-C6U5EQQ:~$ hexdump -C CombinedFiles.z
00000000  80 f0 fa 02 41 80 f0 fa  02 58          |....A....X|
0000000a

```

Kahden laajan tiedoston purku:

- Kokeillaan purkaa edellä pakatun kahden tiedoston pakkaus

```

• elias@DESKTOP-C6U5EQQ:~$ ./my-unzip CombinedFiles.z > CombinedFilesDecompressed.txt
• elias@DESKTOP-C6U5EQQ:~$ cat largeFile1.txt largeFile2.txt | diff - CombinedFilesDecompressed.txt
• elias@DESKTOP-C6U5EQQ:~$ ll CombinedFilesDecompressed.txt
-rw-r--r-- 1 elias elias 100000000 Mar 17 15:08 CombinedFilesDecompressed.txt

```

Virhetilanteet:

- Testataan syöttämällä virheellinen tiedosto.
- Testataan ilman, että syötetään tiedostoa lainkaan.

```

⊗ elias@DESKTOP-C6U5EQQ:~$ ./my-pzip EIOLEOLEMASSA.txt > Kokeilu.z
pzip: cannot open file: No such file or directory

```

```

⊗ elias@DESKTOP-C6U5EQQ:~$ ./my-pzip
pzip: file1 [file2 ...]

```

Lähteet ja viittaukset

- Koodin rakentamisessa seurattu tätä esimerkkikoodia:
<https://github.com/Saggarwal9/Parallel-ZIP/blob/master/pzip.c>
- RLE encoding: <https://www.geeksforgeeks.org/run-length-encoding/>
- POSIX ja threads: https://man7.org/linux/man-pages/man3/pthread_create.3.html
- Mmapin käyttö: <https://www.geeksforgeeks.org/memory-mapping/>
- Get_nproc() käyttö: https://man7.org/linux/man-pages/man3/get_nprocs.3.html
- Pthread_join() käyttö: https://man7.org/linux/man-pages/man3/pthread_join.3.html
- Fstat() käyttö: <https://pubs.opengroup.org/onlinepubs/00969699/functions/fstat.html>
- Munmap() käyttö:
<https://pubs.opengroup.org/onlinepubs/000095399/functions/munmap.html>

- ChatGPT o1 Käytetty apuna koodin debuggaamisessa, mutta lopulliset ratkaisut ja testaukset on tehty manuaalisesti <https://chatgpt.com/>

Projekti 1 pistetoteutus

Kategoria	Maksimi pisteet	Toteutuuko?
Ohjelma kääntyy ja toimii ainakin jollain tasolla	1	Kyllä
Dokumentaatio on asianmukainen ja repository on saatavilla	0 (pakollinen)	Kyllä
Dokumentaatio on yksityiskohtainen ja koodi on kommentoitu hyvin	1	Kyllä
Kaikki lisäoletukset on käsitelty ja virheet dokumentoitu	2	Kyllä
Yhteensä:	5 / 5	5 / 5

Projekti 2 pistetoteutus

Kategoria	Maksimi pisteet	Toteutuuko?
Ohjelmat kääntyvät ja toimivat ainakin jollain tasolla	1	Kyllä
Dokumentaatio on asianmukainen ja repository on saatavilla	0 (pakollinen)	Kyllä
Dokumentaatio on yksityiskohtainen ja koodi on kommentoitu hyvin	1	Kyllä

My-cat toimii ohjeiden mukaan	1	Kyllä
My-grep toimii ohjeiden mukaan	1	Kyllä
My-zip ja my-unzip toimivat ohjeiden mukaan	1	Kyllä
Yhteensä:	5 / 5	5 / 5

Projekti 3 pistetoteutus

Kategoria	Maksimi pisteet	Toteutuuko?
Ohjelma kääntyy ja toimii ainakin jollain tasolla	1	Kyllä
Dokumentaatio on asianmukainen ja repository on saatavilla	0 (pakollinen)	Kyllä
Dokumentaatio on yksityiskohtainen ja koodi on kommentoitu hyvin	1	Kyllä
Parallel zipping useille tiedostoille toimii	1	Kyllä
Parallel unzipping toimii (valinnainen)	1 (ei tehty)	Ei tehty
Yhteensä:	5 / 5	4 / 5