BSEE19046 | Muhammad Saqib Niaz

BSEE19020 | Muhammad Hammad

OS Project

OS Project Report:

1. This is a multithreaded C program that reads data from a file, builds a Huffman table, divides the data into chunks, and then creates multiple threads to encode each chunk using the Huffman table. The encoded data is saved to separate file named Encoded_file.txt, and huffman table values are compressed and stored as bits according to the characters in the original file with another group of threads i.e the zip file! Finally, the program frees the allocated memory and exits.
2. The program begins by including several header files, defining a maximum number of threads to use, and declaring three structs: huffman_node, zip_file, and huffman_table. The huffman_node struct represents a node in a Huffman tree and contains four fields: left, right, freq, and c. The zip_file struct represents a file that has been compressed using Huffman encoding and contains two fields: data and size. The huffman_table variable is a global pointer to an array of huffman_node structs.
3. First the program takes the maximum number of threads and divides it into frequency and compression threads. Next, the program defines a function frequency find that will be executed by each frequency thread. This function takes a single argument of type struct with start index end index and a frequency string and returns a value of type void *. This is to avoid thread race situation by giving each thread its own area of indexes, The function begins by casting the argument to an int and using it to determine the start and end indices of the chunk of data that the thread should process. The function then allocates a buffer freq to store the the frequencies of all the characters in the original file.
4. The function then enters a loop that iterates over the chunk of data, encoding each character using the Huffman table. It does this by looking up the character in the table, following the path through the tree indicated by the left and right fields, and appending a '0' or '1' to the encoded_data buffer for each left or right child, respectively. After the loop completes, the function using the remaining writes the encoded data to a file and frees the Encoded_file.txt buffer. The function then exits.

My Code and further possible improvement and Observations:

My code is a program that reads data from a file, builds a Huffman table, and then creates multiple threads to encode the data using the Huffman table. The encoded data is then saved to separate files for each thread. The main function starts by reading the data from the input file and storing it in a dynamically allocated buffer. It then builds the Huffman table, which is also stored in a dynamically allocated buffer. The data and its size are then stored in a zip_file struct, which is also dynamically allocated. The program then creates multiple threads, each of which executes the encode_thread_function. This function takes a portion of the data and encodes it using the Huffman table. The encoded data is stored in a dynamically allocated buffer and then saved to a file. After all the threads have finished executing, the program cleans up by freeing the dynamically allocated buffers for the data, zip_file struct, and Huffman table. However, there is no error handling in the code to handle cases where the memory allocation fails or if there are any other issues with the input or execution of the program. One potential issue with the code is that the

encode_thread_function is allocating a large buffer for the encoded data, even though it is only using a small portion of it. This can lead to inefficient use of memory and potentially cause the program to run out of memory. Another issue is that the global variables huffman_table and zipped_file are being accessed and modified by multiple threads without any synchronization, which can lead to race conditions and potential data corruption. It would be recommended to use mutexes or other synchronization mechanisms to protect access to these variables. To optimize the code for heap usage, one option is to allocate the buffer for the encoded data only as large as necessary for the encoded data. This can be done by keeping track of the size of the encoded data as it is being generated and allocating the buffer to be exactly that size. This will reduce the amount of unused memory and improve the efficiency of the program. To address the issue of race conditions and potential data corruption, mutexes or other synchronization mechanisms can be used to protect access to the global variables huffman_table and zipped_file. This will ensure that multiple threads do not attempt to access or modify these variables simultaneously, which can lead to data corruption.

Zip.c
Valgrind:

```
valgrind --track-origins=yes --leak-check=full ./zip tozip.txt 10 zip.bin
==2622== Memcheck, a memory error detector
==2622== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==2622== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==2622== Command: ./zip tozip.txt 10 zip.bin
==2622==
File zip.bin already exists
Do you want to overwrite its contents? <y/n>
y
YES
OVERWRITING FILE!

You have asked for 10 number of threads!
Threads will be divided into frequency finder threads and Compression Threads

Your TARGET FILE is:       | tozip.txt |
Your OUTPUT FILE will be:          | zip.bin |

          **This is the data in your file***
----------------------------------------
Hello! how are you?
This is the final semester project of OS Course.
And our group members->
          --bsee19020--Muhammad Hammad
          --bsee19046--Muhammad Saqib Niaz

We hope you like our project!

Please do give additional remarks about the project


,./12345.';'][\[_+-=!@#$%^&&*()
```

This CHUNK 1: from 0, to 55 belongs to thread no 0
This CHUNK 2: from 55, to 110 belongs to thread no 1
This CHUNK 3: from 110, to 165 belongs to thread no 2
This CHUNK 4: from 165, to 220 belongs to thread no 3
This CHUNK 5: from 220, to 274 belongs to thread no 4

--Char--||Huffman code

---r----||0000
---h----||00010
---j----||000110
---q----||00011100
---^----||00011101
---k----||0001111
--- ----||001
---a----||0100
---d----||01010
---(----||01011000
---W----||01011001
---3----||01011010
---\----||01011011
---$----||01011100
---*----||01011101
---_----||01011110
---5----||01011111
---o----||0110
---'----||0111000
---/----||01110010
---w----||01110011
--->----||01110100
---?----||01110101
---;----||01110110
---=----||01110111
---M----||0111100
--- ----||0111101
---H----||0111110
---y----||0111111
---9----||1000000
---f----||1000001
---S----||1000010
---2----||1000011
---N----||10001000
---z----||10001001
---@----||10001010
---A----||10001011
---C----||10001100
---v----||10001101
---&----||1000111

---]----||10010000
---)----||10010001
---4----||1001001
---[----||1001010
---O----||10010110
---P----||10010111
---t----||10011
---6----||10100000
---+----||10100001
---%----||10100010
---,----||10100011
---g----||1010010
---T----||10100110
---#----||10100111
---i----||10101
---u----||10110
---p----||101110
---b----||101111
---m----||11000
---s----||11001
--------||11010

----||11011
---l----||111000
---1----||1110010
---.----||1110011
---n----||1110100
---c----||1110101
---!----||1110110
---0----||1110111
---e----||1111

Previous encoder file deleted successfully
Your output ZIP FILE: | zip.bin | is created succesfully!
==2622==
==2622== HEAP SUMMARY:
==2622== in use at exit: 1,680 bytes in 70 blocks
==2622== total heap usage: 180 allocs, 110 frees, 57,025 bytes allocated
==2622==
==2622== 1,680 bytes in 70 blocks are definitely lost in loss record 1 of 1
==2622== at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==2622== by 0x109B3E: newNode (HUFF.h:38)
==2622== by 0x10A1E2: createAndBuildMinHeap (HUFF.h:178)
==2622== by 0x10A247: buildHuffmanTree (HUFF.h:187)
==2622== by 0x10A4DF: HuffmanCodes (HUFF.h:224)
==2622== by 0x10A617: HUFF (HUFF.h:244)
==2622== by 0x10B1A2: main (zip.c:187)
==2622==
==2622== LEAK SUMMARY:
==2622== definitely lost: 1,680 bytes in 70 blocks
==2622== indirectly lost: 0 bytes in 0 blocks

```
==2622==      possibly lost: 0 bytes in 0 blocks
==2622==    still reachable: 0 bytes in 0 blocks
==2622==         suppressed: 0 bytes in 0 blocks
==2622==
==2622== For lists of detected and suppressed errors, rerun with: -s
==2622== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

Zip.c

Helgrind:

```
valgrind --tool=helgrind ./zip tozip.txt 10 zip.bin
==2600== Helgrind, a thread error detector
==2600== Copyright (C) 2007-2017, and GNU GPL'd, by OpenWorks LLP et al.
==2600== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==2600== Command: ./zip tozip.txt 10 zip.bin
==2600==
File zip.bin already exists
Do you want to overwrite its contents? <y/n>
y
YES
OVERWRITING FILE!

You have asked for 10 number of threads!
Threads will be divided into frequency finder threads and Compression Threads

Your TARGET FILE is:       | tozip.txt |
Your OUTPUT FILE will be:          | zip.bin |

          **This is the data in your file***
----------------------------------------
Hello! how are you?
This is the final semester project of OS Course.
And our group members->
          --bsee19020--Muhammad Hammad
          --bsee19046--Muhammad Saqib Niaz

We hope you like our project!

Please do give additional remarks about the project


,./12345.';'][\[_+-=!@#$%^&&*()

----------------------------------------
This CHUNK 1:   from 0, to 55 belongs to thread no 0
This CHUNK 2:   from 55, to 110 belongs to thread no 1
This CHUNK 3:   from 110, to 165 belongs to thread no 2
This CHUNK 4:   from 165, to 220 belongs to thread no 3
This CHUNK 5:   from 220, to 274 belongs to thread no 4

----------------------
--Char--||Huffman code
```

---r----||0000
---h----||00010
---j----||000110
---q----||00011100
---^----||00011101
---k----||0001111
--- ----||001
---a----||0100
---d----||01010
---(----||01011000
---W----||01011001
---3----||01011010
---\----||01011011
---$----||01011100
---*----||01011101
---_----||01011110
---5----||01011111
---o----||0110
---'----||0111000
---/----||01110010
---w----||01110011
--->----||01110100
---?----||01110101
---;----||01110110
---=----||01110111
---M----||0111100
--- ----||0111101
---H----||0111110
---y----||0111111
---9----||1000000
---f----||1000001
---S----||1000010
---2----||1000011
---N----||10001000
---z----||10001001
---@----||10001010
---A----||10001011
---C----||10001100
---v----||10001101
---&----||1000111
---]----||10010000
---)----||10010001
---4----||1001001
---[----||1001010
---O----||10010110
---P----||10010111
---t----||10011
---6----||10100000
---+----||10100001
---%----||10100010

---,----||10100011
---g----||1010010
---T----||10100110
---#----||10100111
---i----||10101
---u----||10110
---p----||101110
---b----||101111
---m----||11000
---s----||11001
--------||11010

----||11011
---l----||111000
---1----||1110010
---.----||1110011
---n----||1110100
---c----||1110101
---!----||1110110
---0----||1110111
---e----||1111

Previous encoder file deleted successfully
Your output ZIP FILE: | zip.bin | is created succesfully!
==2600==
==2600== Use --history-level=approx or =none to gain increased speed, at
==2600== the cost of reduced accuracy of conflicting-access information
==2600== For lists of detected and suppressed errors, rerun with: -s
==2600== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)

UNZIP.c
Valgrind:

valgrind --track-origins=yes --leak-check=full ./unzip zip.bin 10 unzipped.txt
==2638== Memcheck, a memory error detector
==2638== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==2638== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==2638== Command: ./unzip zip.bin 10 unzipped.txt
==2638==
Your Selected ZIP FILE is: zip.bin
File zip.bin Exists!
File unzipped.txt already exists
Do you want to overwrite its contents? <y/n>
Your uncompressed OUTPUT FILE will be: unzipped.txt
y
YES
OVERWRITING FILE!

Binary string from zipfile

01111101111111000111000011011101100010001001100111001100101000000111100101111101
10101100111010111011101001100001010101110010011010111001001100110001011110011000
01101011110100010011100000111001111111000111111001100111111000000110111000000110
00110111111101011001100101101000001001100101101000010001100011000110101100000110
11111111001111011100010111110100010100010110101100000000110100100000001101011010
00011100011111100010111111110000110011101001110100110110111101110101101010111111
01111111111110010100000001110111100001111101111101011010011110010110000100100110
11000010001010001011111001001100011000010001010110110111101110101101010111110011
11111111100101000000111011110010011010000011010110100111100101100001001001100011
00001000101000110000100100000111001010110111100110001000101010100100010011101111
11010110011111001000100110101110111100101111110110101100011110001010100011111111
01011010110000000110111000000110000110111111101011001111101101101111011100101111
10001111010011001111100101010011000110100101010110001101111110010100010100101010
11001110101011011101000100111000001000011111100001000000000111111001001010010111
01101011010011001100110001011110011011100000001100001101111111010110011110111101
10111010001111100110111001011100101000011010110101001001010111111111001101110000
10110011100010010000100101001011011100101001011110101000011101001110111111011010
0101010100111010111001010001000011101100011110001110101110101011000100100000001

Your uncompressed DATA:

Hello! how are you?
This is the final semester project of OS Course.
And our group members->
	--bsee19020--Muhammad Hammad
	--bsee19046--Muhammad Saqib Niaz

We hope you like our project!

Please do give additional remarks about the project

,./12345.';']]\[_+-=!@#$%^&&*(]

==2638==
==2638== HEAP SUMMARY:
==2638==   in use at exit: 0 bytes in 0 blocks
==2638==   total heap usage: 198 allocs, 198 frees, 34,636 bytes allocated
==2638==
==2638== All heap blocks were freed -- no leaks are possible
==2638==
==2638== For lists of detected and suppressed errors, rerun with: -s
==2638== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)

UNZIP.c
Helgrind:

valgrind --tool=helgrind ./unzip zip.bin 10 unzipped.txt
==2652== Helgrind, a thread error detector
==2652== Copyright (C) 2007-2017, and GNU GPL'd, by OpenWorks LLP et al.
==2652== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==2652== Command: ./unzip zip.bin 10 unzipped.txt
==2652==

Your Selected ZIP FILE is: zip.bin
File zip.bin Exists!
File unzipped.txt already exists
Do you want to overwrite its contents? <y/n>
Your uncompressed OUTPUT FILE will be: unzipped.txt
y
YES
OVERWRITING FILE!

Binary string from zipfile

011111011111110001110000110111011000100010011001110011001010000001111001011111101
101011001110101110111010011000010101011100100110101110010011001100001011110011000
011010111101000100111000001110011111110001111110011001111110000001101110000001100
001101111111010110011001011010000010011001011010000100011000110001101011000001100
111111110011110111000101111101000101000101110101100000001101001000000110101101011
000111000111111000101111111100001100111010011101001101101111011101011010101111110
011111111111100101000000111011110000111110111110101101001111001011000010010011000
110000100010100010111110010011000110000100010101101101111011101011010101111110011
111111111100101000000111011110010011010000011010110100111100101100001001001100011
000010001010001100001001000001110010101101111001100010001010101001000100111011110
110101100111110010001001101011101110010111111010110100011110001010100011111111 0
010110101100000001101110000001100001101111111010110011111011011011110111001011111
100011110100110011111001010100110001101001010101100011011111001010001010010101010
110011101010110111010001001110000010000111111000010000000001111110010010100101111
011010110100110011001100010111100110111000000110000110111111101011001111011110111
101110100011111001101110010111001010000110101101010010010101111111100110111000011
101100111000100100001001010010111011100101001011110101000011101001110111111011010 0
010101010011101011100101000100001110110001111000111010111010101100010010000000 1

Your uncompressed DATA:

Hello! how are you?
This is the final semester project of OS Course.
And our group members->
	--bsee19020--Muhammad Hammad
	--bsee19046--Muhammad Saqib Niaz

We hope you like our project!

Please do give additional remarks about the project

,./12345.';'][\[_+-=!@#$%^&&*(]

==2652==
==2652== Use --history-level=approx or =none to gain increased speed, at
==2652== the cost of reduced accuracy of conflicting-access information
==2652== For lists of detected and suppressed errors, rerun with: -s
==2652== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)