Harsh Patel
AU1940184

Que-1

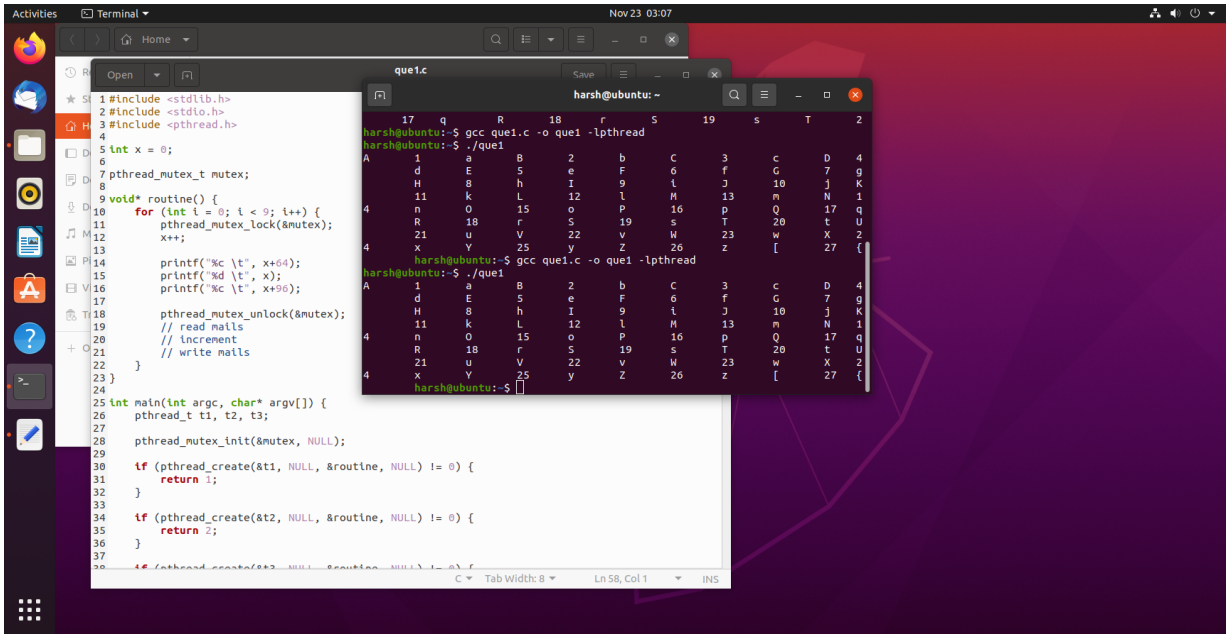
(b)

In the main() function we have created a variable called thread_id, which type is pthread_t, we are using this variable to identify threads in the system. We are calling pthread_crate() after declaring 3 thread_id for creating three threads. We are calling mutex to synchronize all the threads.

We have a number of processes and these processes require a number of resources in any Operating System. In this code 3 threads are using the same variable x. They are reading the variable and then updating the value to variable and then finally writing the data in the memory.

We can see that after doing some operation the process will have to read the value of x, then increment the value of x by 1 and last write the value of x in the memory. Then, we have all the three threads that need to be executed. Foe the synchronization all three threads we have udes mutex lock in the routine function by using that we have confirmed that not more than one thread is using the value of x togather and then we have printed the x's ascii value and integer value . Then we unlocked the mutex so that other thread can make chnages too.

Output:



Que-2

(b)

Buddy Algorithm for memory Allocation

The buddy memory allocation is a technique for memory allocation that divides memory into partitions and tries to satisfy a memory request. This algorithm makes use of memory by splitting it into halves and satisfies requirements.

Compared to the modern memory allocation techniques buddy memory allocation is easy to implement, and has much less hardware requirements. Also, compared to the other memory allocation techniques like dynamic allocation it has little internal fragmentation, and has little overhead trying to do compaction of memory.

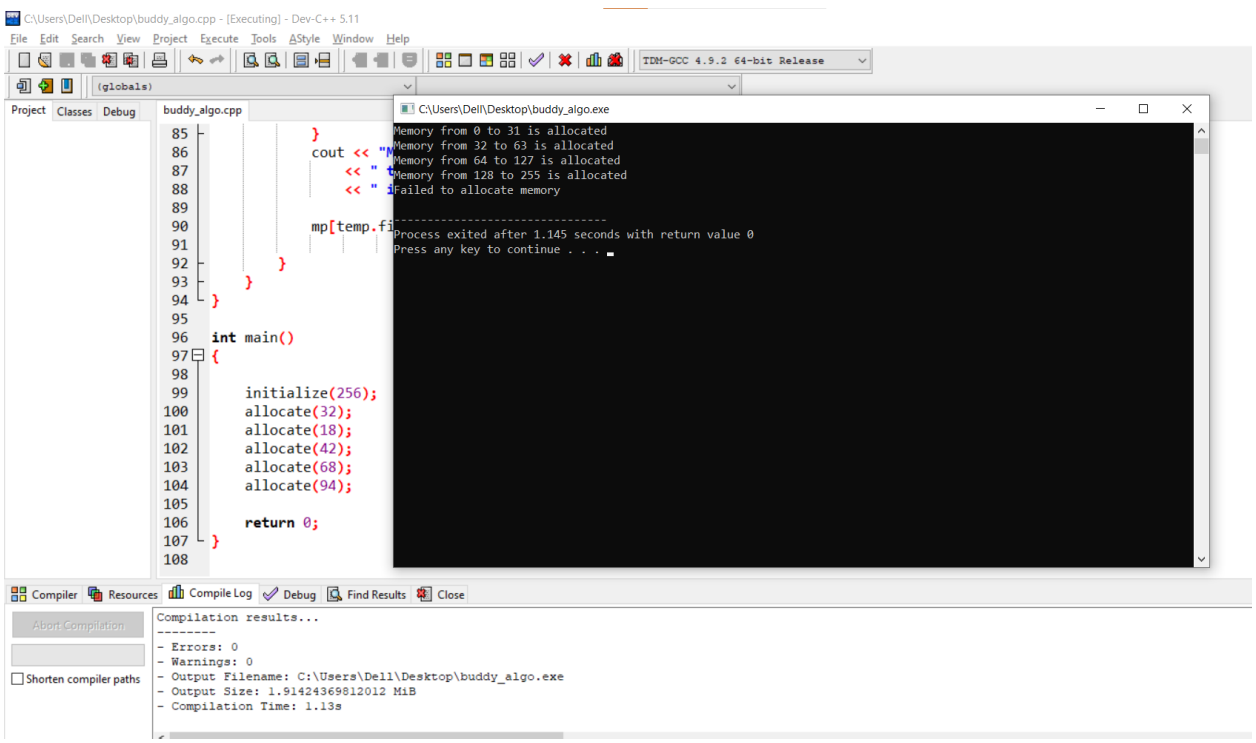
In buddy memory allocation there is memory wasted because the memory requested is a little larger than a small block, but a lot smaller than a large block. (ex. If a program requests 65K memory, then memory allocated is 128K so 63K memory is wasted).

The buddy memory allocation technique allocates memory in powers of 2, i.e 2^x, where x is an integer. Whenever a request for memory allocation comes, we are looking for the smallest block bigger than it and if such a block is found on the remaining free list then allocation is done. If not found then we traverse the free list upward until we find the bigger block. We keep splitting it

into two blocks. One for adding the next free list and another to traverse down the tree till we reach the target and return the request memory block to the user. If no allocation is possible, we return null.

Let us see how the algorithm proceeds by tracking a memory block of size 256 KB. Initially, the free list is: {}, {}, {}, {}, {}, {}, {}, { (0, 256) }

Output:



Buddy Algorithm for Memory Deallocation

We are doing allocation by the usage of free lists but for the deallocation we will maintain an extra data structure - a map with the starting address of the segment as key and size of the segment as value and update it whenever an allocation request comes. Now, whenever a deallocation request comes, first we are checking map if the request is valid or not, then if the request is valid we will add the block to the free list tracking block of their sizes. Then we will search the free list and then merge the block and place them in the free list above them.

Let us see how the algorithm proceeds by tracking a memory block of size 128 KB. Initially, the free list is: {}, {}, {}, {}, {}, {}, {}, { (0, 127) }

Output:

C:\Users\Dell\Desktop\buddy_algo_deallocation.cpp - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

Project Classes Debug | buddy_algo.cpp | buddy_algo_deallocation.cpp

```
            free_list[x].size() - 1);
            break;
        }
    }

    // Remove the key existence from map
    mp.erase(id);
}

// Driver code
int main()
{

    initialize(128);
    allocate(16);
    allocate(32);
    allocate(64);
    allocate(16);
    deallocate(0);
    deallocate(9);
    deallocate(32);
    deallocate(16);

    return 0;
```

C:\Users\Dell\Desktop\buddy_algo_deallocation.exe

```
Memory from 0 to 15 is allocated
Memory from 32 to 63 is allocated
Memory from 64 to 127 is allocated
Memory from 16 to 31 is allocated
Memory block from 0 to 15 freed
Sorry, invalid free request
Memory block from 32 to 63 freed
Memory block from 16 to 31 freed
Coalescing of blocks starting at 0 and 16 was done

--------------------------------
Process exited after 5.601 seconds with return value 0
Press any key to continue . . .
```

Compiler | Resources | Compile Log | Debug | Find Results | Close

Abort Compilation

Shorten compiler paths

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Dell\Desktop\buddy_algo_deallocation.exe
- Output Size: 1.95053672790527 MiB
- Compilation Time: 1.05s
```

Que-3

C:\Users\Dell\Desktop\que3.c - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

Project Classes Debug | buddy_algo.cpp | buddy_algo_deallocation.cpp | que3.c | q3.c

```
    for(i = 0; i < 5; i++) {
        pthread_create(&pro[i], NULL, (void
    }

    for(i = 0; i < 5; i++) {
        pthread_create(&con[i], NULL, (void
    }

    for(i = 0; i < 5; i++) {
        pthread_join(pro[i], NULL);
    }

    for(i = 0; i < 5; i++) {
        pthread_join(con[i], NULL);
    }

    pthread_mutex_destroy(&mutex);
    sem_destroy(&Empty);
    sem_destroy(&full);


    return 0;
```

C:\Users\Dell\Desktop\que3.exe

```
Producer 1: Insert Item 41 at 0
Producer 5: Insert Item 41 at 1
Producer 3: Insert Item 41 at 2
Producer 1: Insert Item 18467 at 3
Producer 4: Insert Item 41 at 4
Consumer 1: Remove Item 41 from 0
Consumer 2: Remove Item 41 from 1
Consumer 2: Remove Item 41 from 2
Consumer 2: Remove Item 18467 from 3
Consumer 3: Remove Item 41 from 4
Producer 2: Insert Item 41 at 0
Producer 5: Insert Item 18467 at 1
Producer 3: Insert Item 18467 at 2
Producer 1: Insert Item 6334 at 3
Producer 4: Insert Item 18467 at 4
Consumer 3: Remove Item 41 from 0
Consumer 1: Remove Item 18467 from 1
Consumer 4: Remove Item 18467 from 2
Consumer 5: Remove Item 6334 from 3
Consumer 2: Remove Item 18467 from 4
```

Compiler | Resources | Compile Log | Debug | Find Results | Close

Abort Compilation

Shorten compiler paths

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Dell\Desktop\que3.exe
- Output Size: 186.890625 KiB
- Compilation Time: 0.34s
```