CSCE 313-200
Introduction to Computer Systems
Spring 2022

Practice II
Dmitri Loguinov
Texas A&M University

March 8, 2022
• Request buffer allocated once per thread:

```c
#define MAX_BATCH 10000
// set up initial buffer to hold header + MAX_BATCH rooms
char *request = new char [...];
CommandRobotHeader *crh = (...) request;
DWORD *roomArray = (...) (crh + 1);
```

• Then, batch-mode pop works as following:

```c
int nPopped = Q[cur].pop (roomArray, MAX_BATCH);
// compute msg size based on nPopped
pipe.SendMsg (request, requestSize);
```

• BFS queue class – needs to be written from scratch
  - Encapsulates a buffer with two offsets: head & tail
• Use a private heap inside the queue class
  - HeapCreate(), HeapAlloc(), HeapFree() instead of new/delete
Homework #2

• Simplified queue without concurrent push/pop
  - Push moves tail by batch size
  - Pop moves head similarly

• When buffer overflows, what operations are needed to double the queue size?

```c
// double queue size
size <<= 1;
buf = HeapReAlloc (heap, HEAP_NO_SERIALIZE,
                   buf, size);
```

• Simplest is to use HeapReAlloc()
  - If realloc is not in place, the function copies your data
**Homework #2**

- **Hash tables**
  - 4B bits in a 512-MB buffer represent all possible nodes
  - InterlockedBitTestAndSet to access the bits
  - LONG array of $2^{32}/32 = 2^{27}$ words (each word is 4 bytes)
  - Make sure to memset to zero during initialization

- **Given room ID x, what is the offset and bit # in array?**
  - Offset = $x >> 5$ (equivalent to $x / 32$)
  - Bit = $x & 0x1F$ (equivalent to $x \% 32$)

- **Extra credit**: devise a method to interlock less frequently when the number of unique rooms drops close to 0%
  - One line of code
Homework #2

- General structure, gets you ~260 sec runtime on ts

```c
char *request = new char
    [sizeof(CommandRobotHeader) +
     MAX_BATCH * sizeof(DWORD)];
CommandRobotHeader *crh =
    (CommandRobotHeader*)request;
crh->command = MOVE;
DWORD *rooms = (DWORD *) (cr + 1);
while (true) {
    if (quit) // flag set? 
        break;
    int batch = 0;
    CS.lock(); // PC 3.4
    if (Q[cur].sizeQ > 0) {
        batch = Q[cur].pop (rooms, MAX_BATCH);
        activeThreads ++;
        // other stats go here
    }
    CS.unlock();
    if (batch == 0) { // got nothing from Q?
        Sleep (100);
        continue;
    }
pipe.SendMsg (...); // send request[]
pipe.RecvMsg (...); // read response
while (rooms left in response) {
    DWORD ID = ... // get next room
    DWORD offset = ...
    DWORD bit = ...
    if (InterlockedBitTestSet (hashTable + offset, bit) == 0)
        localQ.push (ID);
}
CS.lock();
// batch-pop all elements from 
// localQ into Q[cur^1]
activeThreads --;
if (this BFS level is over)
    if (next level empty)
        quit = true;
    else
        cur ^= 1;
    CS.unlock();
}
```

- Target delay is below 130 sec on P30