CS418—Operating Systems

Lecture 6

Memory Management

Textbook: Operating Systems
by William Stallings

1. Why buddy system?

- In real systems, a page is usually of size 2^a — a power of 2.
- A combination of dynamic partition and paging.
- Allocation algorithm
  - 1. `Take a free block` B_i.
  - 2. `If the job requests more than 50% of` B_i, `allocate` B_i `to the job. Otherwise break` B_i `into two blocks` B_{i1}, B_{i2} `with equal size and proceed recursively on` B_{i1}.
- Deallocation algorithm
  - 1. `Take a block` B_i `to be released.`
  - 2. `If the buddy of` B_i `(i.e., adjacent to` B_i `and has the same size as` B_i`),` B_j, `is free, then combine` B_i, B_j `into a free block` B_k `with twice of the size.`
  - 3. `Recurse on` B_k.

2. Hash Tables

- How to put N items in a table of size $M(N \leq M)$?
  - 1. Sequential search — $O(M)$ time in the worst case.
  - 2. Associative search — $O(1)$ time, but needs special hardware.
  - 3. Binary search — $O(\log M)$ time, but adding new items is hard.
- Hashing: An element with key k is mapped to slot $h(k)$ in the table.
- How do we handle the situation when two elements are mapped to the same slot (collision)?
  - 1. Using an overflow area.
  - 2. Without using an overflow area.

3. Hashing with chaining

- Insert: If collision occurs, insert the element in an overflow area.
- Search: If slot $h(k)$ is empty, report that no element exists. Otherwise, search slot $h(k)$ then possibly the corresponding slots in the overflow area.
- Deletion: Just mark 'NIL'.
- Can you think of some good hash function?

4. Hashing without overflow area

- Insert: To perform insertion using this method, we successively examine (or probe) the hash table until we find an empty slot for the element.
- Search: Similar to Insert.
- Deletion: You can't just mark 'NIL', mark 'DELETED' instead. Why?
- Linear hashing
- Double hashing
- **End of Memory Management.** $\Box$