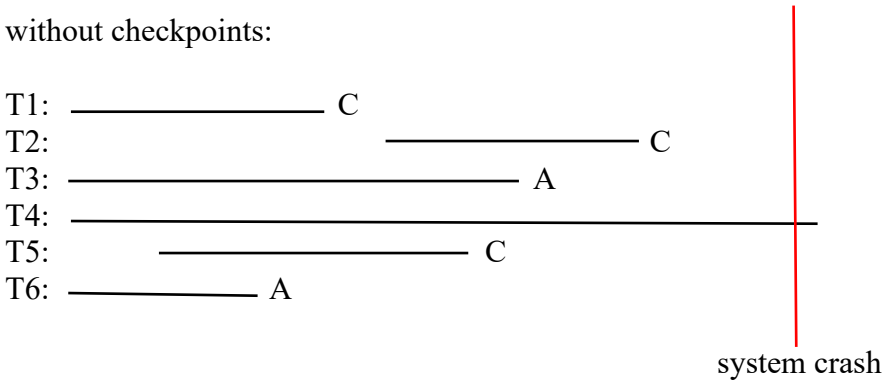
# CS1555/CS2055 Recitation 13

Objective: Practice recovery, operations on static hashing and extendible hashing

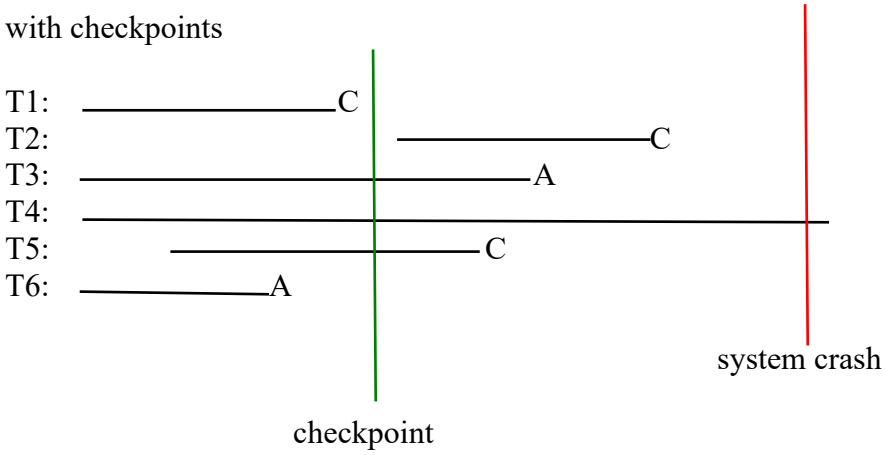
**Part 1. Recovery**

For the following transaction executions, state what the system should do when it restarts after a crash:

a. without checkpoints:




b. with checkpoints




**Part 2. Hash Files**

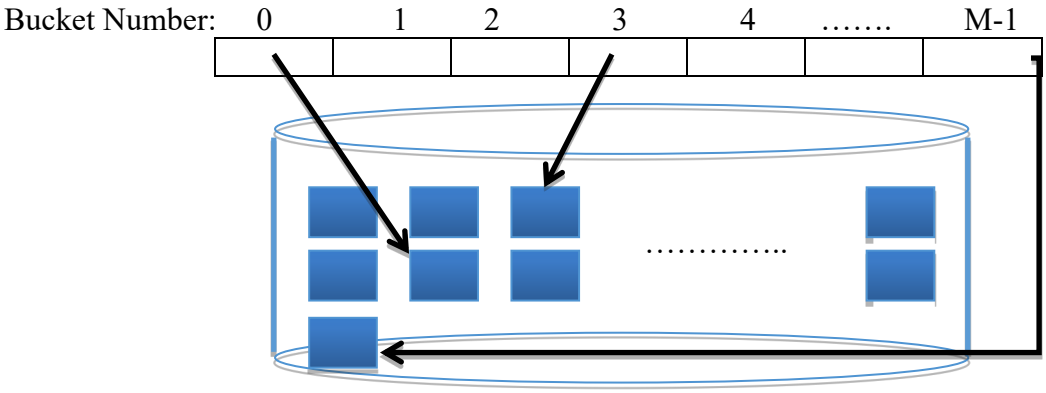
Hashing

- Convert the key of a record into an address in which the record is stored
- Search condition *must be* an equality condition on a single field, called **hash field**
- **Hash function:** a function that is applied to the hash field value of a record and yields the *address* of the disk block in which the record is stored

External Hashing for disk Files

- Target address space is made of **buckets**, each of which holds multiple records.
- A **bucket** is either one disk block or a number of contiguous disk blocks
- The *hashing function* maps a key into a relative bucket number, rather than assigning an absolute block address to the bucket

- A table maintained in the file header converts the bucket number into the address of the first disk block of the bucket
- Why target buckets and not blocks?




1) **Static hashing:**
   a) Converts the key of a record into an address in which the record is stored
   b) A fixed number of buckets, M, is allocated to match the fixed (static) Hash function range
   c) Each bucket has the same number of blocks. The number of blocks can change
   d) The collision problem is solved by using overflow buckets

Consider the following record keys: (3, 2, 1, 8, 6, 4, 14, 5, 9). Create the static hash structure, with M=4 main buckets, that will contain the provided records, using the chaining technique. Use h(k) =k mod M as a hashing function. Each bucket can hold 2 records.

2) **Dynamic hashing:**
   a) **Extendible hashing:**
      - An array of $2^d$ bucket addresses is maintained, where *d* is called **global depth** of the directory
      - The integer value corresponding to the first(high-order) *d* bits of a hash value is used as an index to the array to determine a directory entry and the address in that entry determines the bucket in which the corresponding records are stored
      - A local depth $d'$ –stored with each bucket –specifies the number of bits on which the bucket contents are based
      - The value of *d* can be increased or decreased by one at a time, thus doubling or halving the number of entries in the directory array.
      - **Doubling** is required when the bucket whose local depth $d'$ , equal with the global depth *d*, overflows

Create an extendible hash structure for these record keys: (2, 3, 4, 8, 1, 12, 9, 7). Use the 4 bit binary representation of the keys (2=0010, 3=0011, 4=0100, 8=1000, 1=0001, 12=1100, 9=1001, 7=0111). Use a bfr=2.