Department of Computer Science
UNIVERSITY OF COLORADO BOULDER

Design and Analysis of Operating Systems CSCI 3753

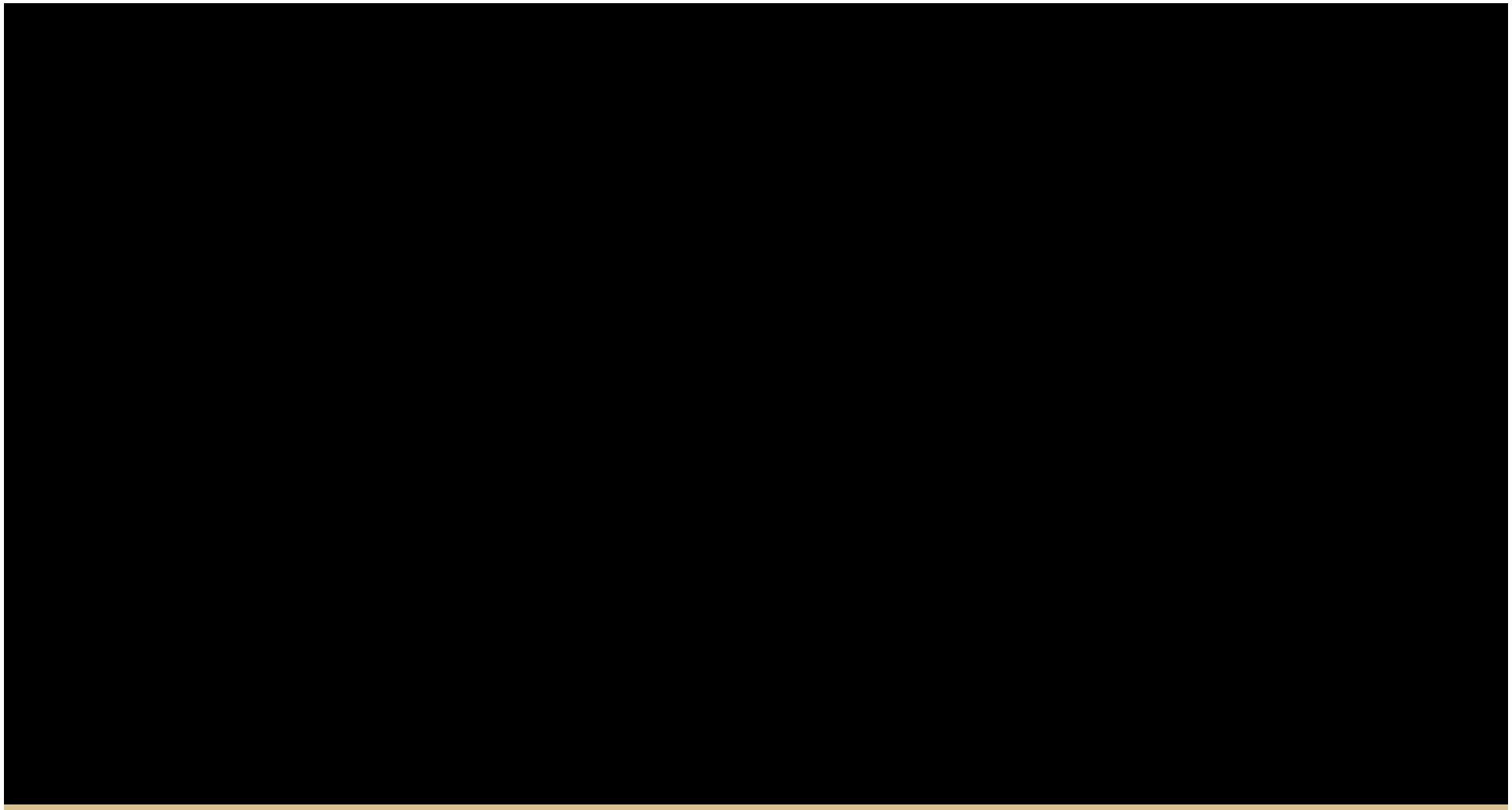
Dr. David Knox
University of Colorado Boulder

These slides adapted from materials provided by the textbook authors.


COLORADO

Classical Problems of Synchronization

- Bounded-Buffer Problem
- Readers and Writers Problem
- Dining-Philosophers Problem



Reader/Writer Problem

- *N* tasks want to write to a shared file
- *M* different tasks want to read from same shared file
- Must synchronize access
- Condition 1: a writer must have exclusive access
- Condition 2: Readers can share access with other readers

Reader/Writer problem is similar, but different, to Bounded Buffer problem

	Reader / Writer	Bounded Buffer
# tasks	*M* readers *N* writers	1 or more producers, 1 or more consumers
Amount of data	One shared data object	*D* data objects stored in a shared buffer
Exclusion	A writer must exclude all other writer and readers A reader allows other readers, but excludes writers	Any producer excludes all other tasks Any consumer excludes all other tasks

Modified Conditions

- Condition 1: a writer must have exclusive access
- Condition 2: Readers can share access with other readers
  - No reader is denied access unless a writer has control
  - A writer cannot be kept waiting indefinitely by readers arriving after the writer

- **Problem**:
  first reader grabs lock, preventing other readers & writers
- Solution:
  only the first reader will grab the lock, and the last reader will release the lock
- **Problem**:
  readcount++ and *readcount–* lead to race conditions
- Solution:
  surround access to *readcount* with a mutex

- **Problem**:

  This solution could starve pending writers

Current solution gives precedence to readers

- new readers can keep arriving while any one reader holds the write lock, the writer can starve waiting until the last reader is finished

Add another mutex and counter

- Instead, allow a pending writer to block future reads
- Once first writer grabs readBlock,
  - any number of writers can come through while the first reader is blocked on readBlock
  - and subsequent readers are blocked on writePending
  - So, behavior is that a writer can block not just new readers, but also some earlier readers
  - Note now that readers can be starved!

```
int readCount = 0, writeCount = 0;
semaphore mutex = 1, mutex2 = 1;
semaphore readBlock = 1, writeBlock =
1,
          writePending = 1;

writer() {
  while(TRUE) {

    P(mutex2);
      writeCount++;
      if(writeCount == 1)
        P(readBlock);
    V(mutex2);

    P(writeBlock);
      // writing
    V(writeBlock);

    P(mutex2)
      writeCount--;
      if(writeCount == 0)
        V(readBlock);
    V(mutex2);
  }
}
```

```
reader() {
  while(TRUE) {

    P(writePending);
      P(readBlock);
        P(mutex1);
          readCount++;
          if(readCount == 1)
            P(writeBlock);
        V(mutex1);
      V(readBlock);
    V(writePending);

     // reading

    P(mutex1);
      readCount--;
      if(readCount == 0)
        V(writeBlock);
    V(mutex1);
  }
}
```

- We can simplify the algorithm
- This is starvation-free solution

Modified Conditions

- Condition 1: a writer must have exclusive access
- Condition 2: Readers can share access with other readers
  - No reader is denied access unless a writer has control
  - A writer cannot be kept waiting indefinitely by readers arriving after the writer

Starvation free solution

```
Semaphore writeblock=1, mutex=1, readblock=1
int readcount = 0
```

Writer:

```
while(1) {
  wait(readblock)
    wait(writeblock);
      // writing
    signal(writeblock);
  signal(readblock)
}
```

Reader:

```
while(1) {
    wait(readblock)
      wait(mutex)
      readcount++;
      if (readcount==1)
         wait(writeblock);
      signal(mutex)
    signal(readblock)

       // reading

    wait(mutex)
    readcount--;
    if (readcount==0)
       signal(writeblock);
    signal(mutex)
}
```

Modified Conditions

- Condition 1: a writer must have exclusive access
- Condition 2: Readers can share access with other readers
  - No reader is denied access unless a writer has control
  - A writer cannot be kept waiting indefinitely by readers arriving after the writer

Department of Computer Science
UNIVERSITY OF COLORADO BOULDER

Design and Analysis of Operating Systems CSCI 3753

Dr. David Knox
University of Colorado Boulder

These slides adapted from materials provided by the textbook authors.