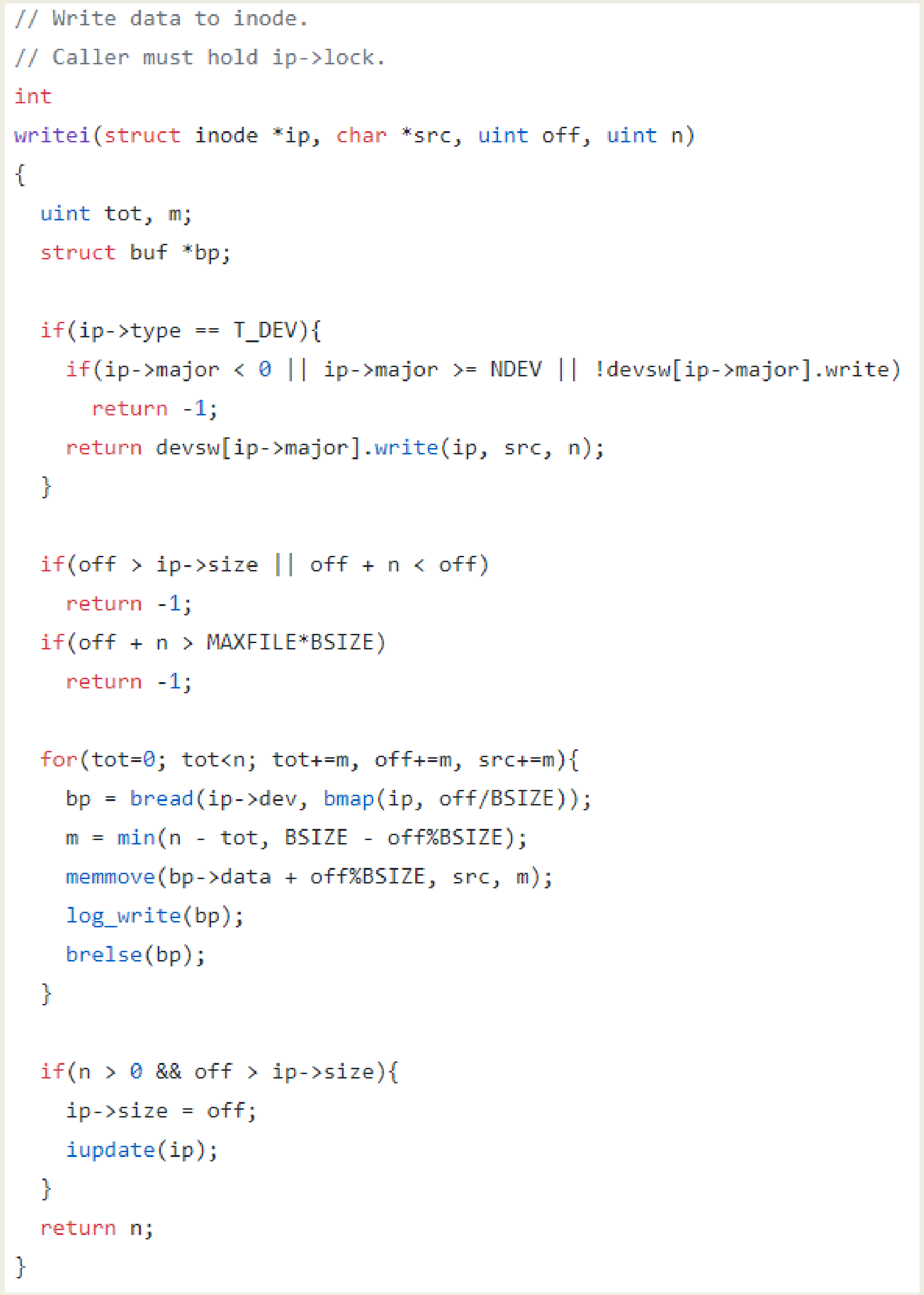
XV6代码分析

王轩阳、龚力维、肖士湘、袁琦(未参加)组

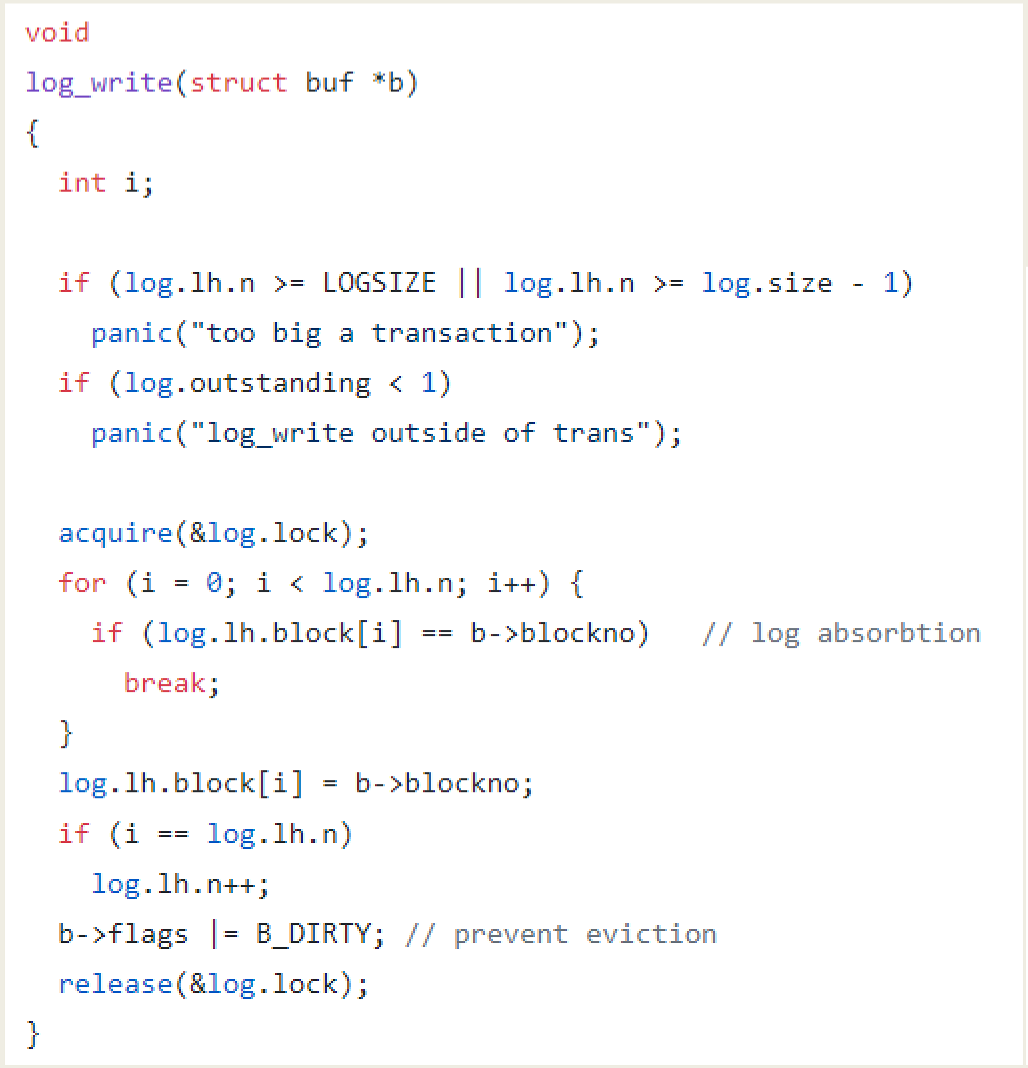
writei 函数中调用了 log_write 函数，它的功能是什么？

writei函数



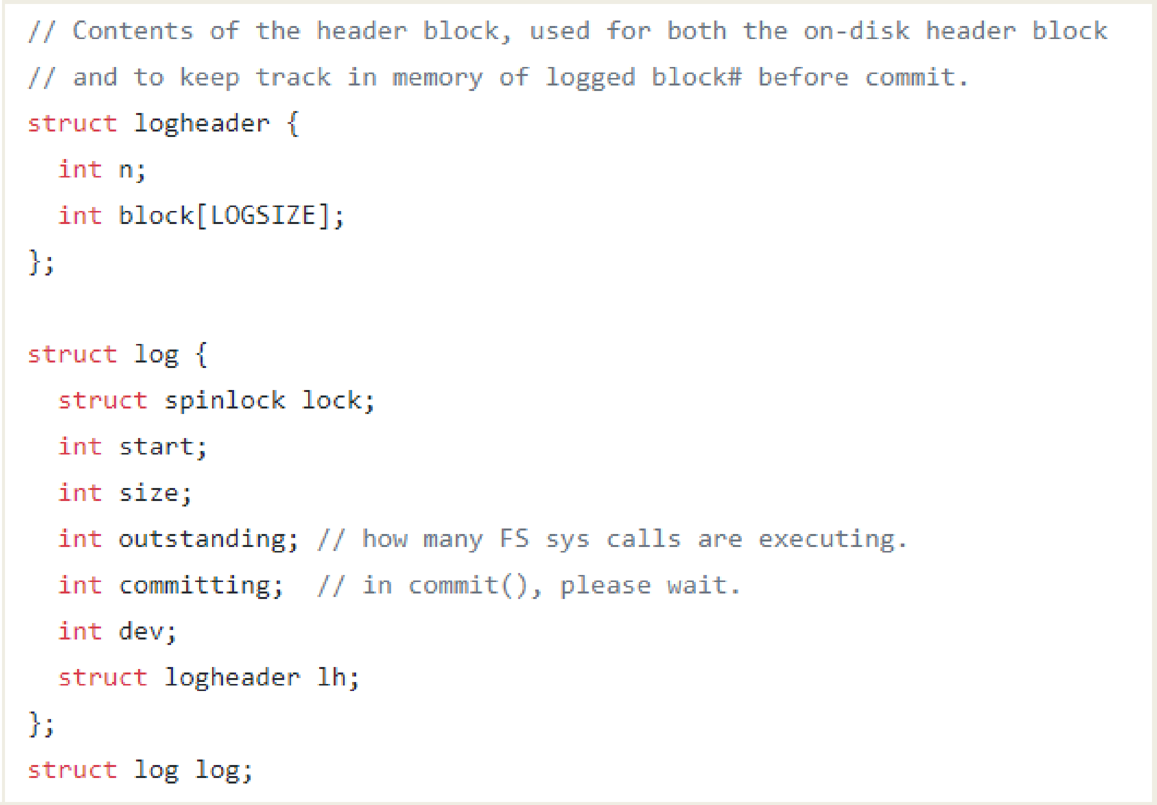
```c
// Write data to inode.
// Caller must hold ip->lock.
int
writei(struct inode *ip, char *src, uint off, uint n)
{
  uint tot, m;
  struct buf *bp;

  if(ip->type == T_DEV){
    if(ip->major < 0 || ip->major >= NDEV || !devsw[ip->major].write)
      return -1;
    return devsw[ip->major].write(ip, src, n);
  }

  if(off > ip->size || off + n < off)
    return -1;
  if(off + n > MAXFILE*BSIZE)
    return -1;

  for(tot=0; tot<n; tot+=m, off+=m, src+=m){
    bp = bread(ip->dev, bmap(ip, off/BSIZE));
    m = min(n - tot, BSIZE - off%BSIZE);
    memmove(bp->data + off%BSIZE, src, m);
    log_write(bp);
    brelse(bp);
  }

  if(n > 0 && off > ip->size){
    ip->size = off;
    iupdate(ip);
  }
  return n;
}
```

log_write函数



```
void
log_write(struct buf *b)
{
  int i;

  if (log.lh.n >= LOGSIZE || log.lh.n >= log.size - 1)
    panic("too big a transaction");
  if (log.outstanding < 1)
    panic("log_write outside of trans");

  acquire(&log.lock);
  for (i = 0; i < log.lh.n; i++) {
    if (log.lh.block[i] == b->blockno)   // log absorbtion
      break;
  }
  log.lh.block[i] = b->blockno;
  if (i == log.lh.n)
    log.lh.n++;
  b->flags |= B_DIRTY; // prevent eviction
  release(&log.lock);
}
```

log定义



```
// Contents of the header block, used for both the on-disk header block
// and to keep track in memory of logged block# before commit.
struct logheader {
  int n;
  int block[LOGSIZE];
};

struct log {
  struct spinlock lock;
  int start;
  int size;
  int outstanding; // how many FS sys calls are executing.
  int committing;  // in commit(), please wait.
  int dev;
  struct logheader lh;
};
struct log log;
```

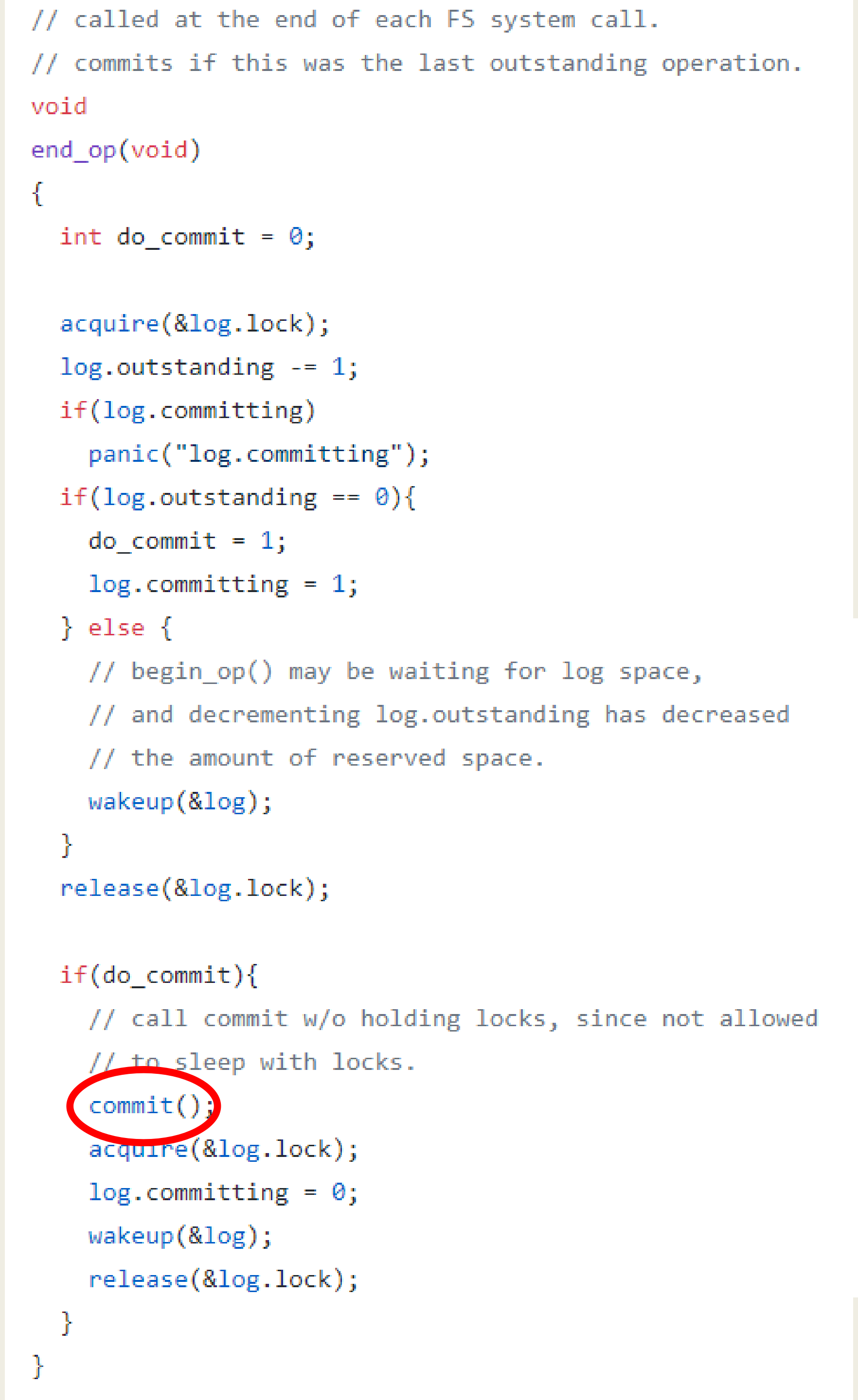
阅读end_op函数代码，介绍end_op函数何时会被调用，end_op函数的功能是什么？

end_op

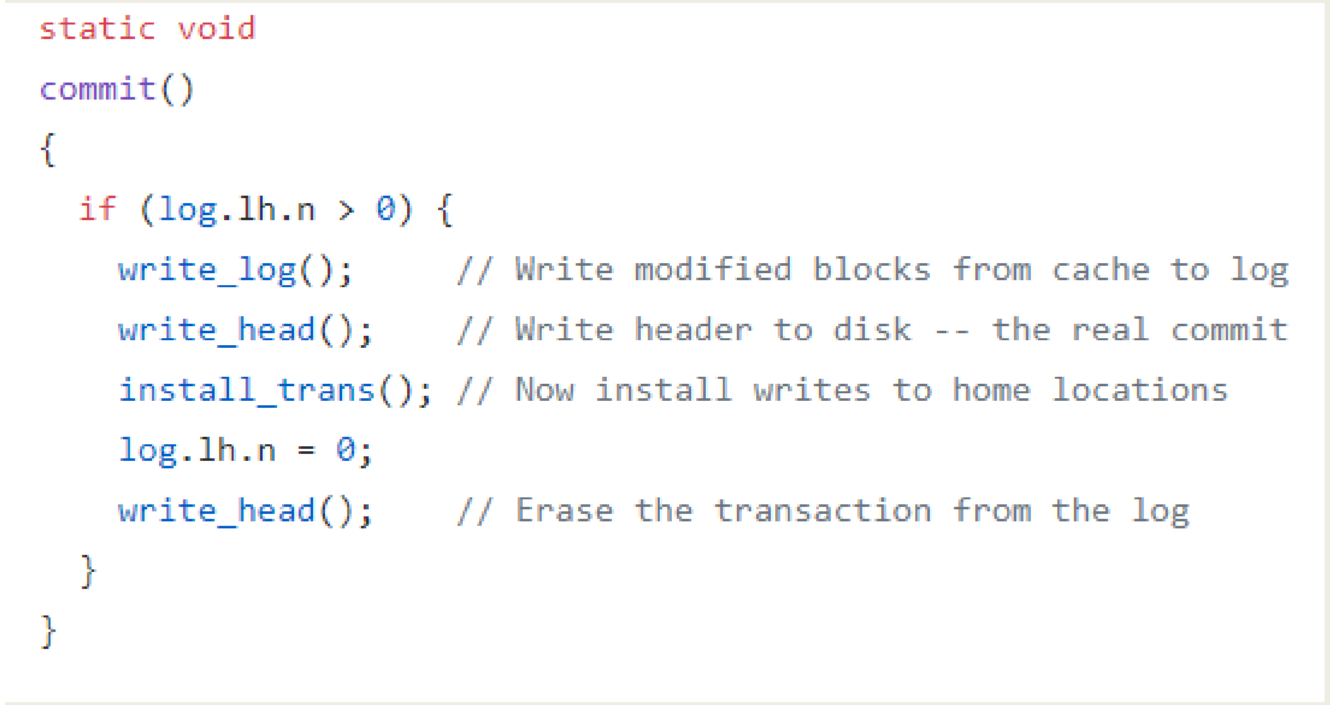
```c
// called at the end of each FS system call.
// commits if this was the last outstanding operation.
void
end_op(void)
{
  int do_commit = 0;

  acquire(&log.lock);
  log.outstanding -= 1;
  if(log.committing)
    panic("log.committing");
  if(log.outstanding == 0){
    do_commit = 1;
    log.committing = 1;
  } else {
    // begin_op() may be waiting for log space,
    // and decrementing log.outstanding has decreased
    // the amount of reserved space.
    wakeup(&log);
  }
  release(&log.lock);

  if(do_commit){
    // call commit w/o holding locks, since not allowed
    // to sleep with locks.
    commit();
    acquire(&log.lock);
    log.committing = 0;
    wakeup(&log);
    release(&log.lock);
  }
}
```

end_op函数调用了commit函数，commit函数的流程是什么，每个语句分别完成了什么功能?（介绍相关调用函数）

commit



```
// called at the end of each FS system call.
// commits if this was the last outstanding operation.
void
end_op(void)
{
  int do_commit = 0;

  acquire(&log.lock);
  log.outstanding -= 1;
  if(log.committing)
    panic("log.committing");
  if(log.outstanding == 0){
    do_commit = 1;
    log.committing = 1;
  } else {
    // begin_op() may be waiting for log space,
    // and decrementing log.outstanding has decreased
    // the amount of reserved space.
    wakeup(&log);
  }
  release(&log.lock);

  if(do_commit){
    // call commit w/o holding locks, since not allowed
    // to sleep with locks.
    commit();
    acquire(&log.lock);
    log.committing = 0;
    wakeup(&log);
    release(&log.lock);
  }
}
```



```
static void
commit()
{
  if (log.lh.n > 0) {
    write_log();     // Write modified blocks from cache to log
    write_head();    // Write header to disk -- the real commit
    install_trans(); // Now install writes to home locations
    log.lh.n = 0;
    write_head();    // Erase the transaction from the log
  }
}
```

阅读 *initlog* 函 数
log 的 初 始 化 是 何 时 完 成 的
里 面 调 用 了*recover_from_log* 函数，为什么要调用它，它的作用是什么？

inilog

```c
void
initlog(int dev)
{
  if (sizeof(struct logheader) >= BSIZE)
    panic("initlog: too big logheader");

  struct superblock sb;
  initlock(&log.lock, "log");
  readsb(dev, &sb);
  log.start = sb.logstart;
  log.size = sb.nlog;
  log.dev = dev;
  recover_from_log();
}
```

recover_from_log

```c
static void
recover_from_log(void)
{
  read_head();
  install_trans(); // if committed, copy from log to disk
  log.lh.n = 0;
  write_head(); // clear the log
}
```

（进阶题）linux 代码中实现了哪些日志文件系统，和 xv6 中的日志管理
有什么不同?

祝各位老师、同学元旦快乐