# Python solution with explainations and comments

zhuyinghua1203 ★ 1190 December 20, 2015 1:49 AM 3.0K VIEWS

6

Structure is from this post

IMHO, the above solution cannot handle a few problems:

1. if there are lots of read(1) or read(2), the queue becomes increasingly large
2. the queue is extend by size of 4 in each call, and "" can be wrote into queue even it doesn't exist in file

Here is my modified algorithm with commons..

```
def __init__(self):
    self.queue = [] # global "buffer"

def read(self, buf, n):
    idx = 0

    # if queue is large enough, read from queue
    while self.queue and n > 0:
        buf[idx] = self.queue.pop(0)
        idx += 1
        n -= 1

    while n > 0:
        # read file to buf4
        buf4 = [""]*4
        l = read4(buf4)

        # if no more char in file, return
        if not l:
            return idx

        # if buf can not contain buf4, save to queue
        if l > n:
            self.queue += buf4[n:l]

        # write buf4 into buf directly
        for i in range(min(l, n)):
            buf[idx] = buf4[i]
            idx += 1
            n -= 1
    return idx
```

python