介绍Linux系统内核文件Cache管理机制（3） PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/144/2021_2022__E4_BB_8B_E7_BB_8DLinu_c103_144560.htm 4、文件Cache的预读和替换 Linux内核中文件预读算法的具体过程是这样的：对于每个文件的第一个读请求，系统读入所请求的页面并读入紧随其后的少数几个页面(不少于一个页面，通常是三个页面)，这时的预读称为同步预读。对于第二次读请求，如果所读页面不在Cache中，即不在前次预读的group中，则表明文件访问不是顺序访问，系统继续采用同步预读；如果所读页面在Cache中，则表明前次预读命中，操作系统把预读group扩大一倍，并让底层文件系统读入group中剩下尚不在Cache中的文件数据块，这时的预读称为异步预读。无论第二次读请求是否命中，系统都要更新当前预读group的大小。此外，系统中定义了一个window，它包括前一次预读的group和本次预读的group。任何接下来的读请求都会处于两种情况之一：第一种情况是所请求的页面处于预读window中，这时继续进行异步预读并更新相应的window和group；第二种情况是所请求的页面处于预读window之外，这时系统就要进行同步预读并重置相应的window和group。图5是Linux内核预读机制的一个示意图，其中a是某次读操作之前的情况，b是读操作所请求页面不在window中的情况，而c是读操作所请求页面在window中的情况。 Linux内核中文件Cache替换的具体过程是这样的：刚刚分配的Cache项链入到inactive_list头部，并将其状态设置为active，当内存不够需要回收Cache时，系统首先从尾部开始反向扫描active_list并将状态不是referenced的项链入

到inactive_list的头部，然后系统反向扫描inactive_list，如果所扫描的项的处于合适的状态就回收该项，直到回收了足够数目的Cache项。Cache替换算法如图6的算法描述伪码所示。Mark_Accessed(b) {if b.state==(UNACTIVE amp. UNREFERENCE)b.state = REFERENCEelse if b.state == (UNACTIVE amp. REFERENCE) {b.state = (ACTIVE amp. UNREFERENCE)Add X to tail of active_list } else if b.state == (ACTIVE amp. UNREFERENCE) b.state = (ACTIVE amp. REFERENCE)}Reclaim(){if active_list not empty and scan_num图6 Linux的Cache替换算法描述 