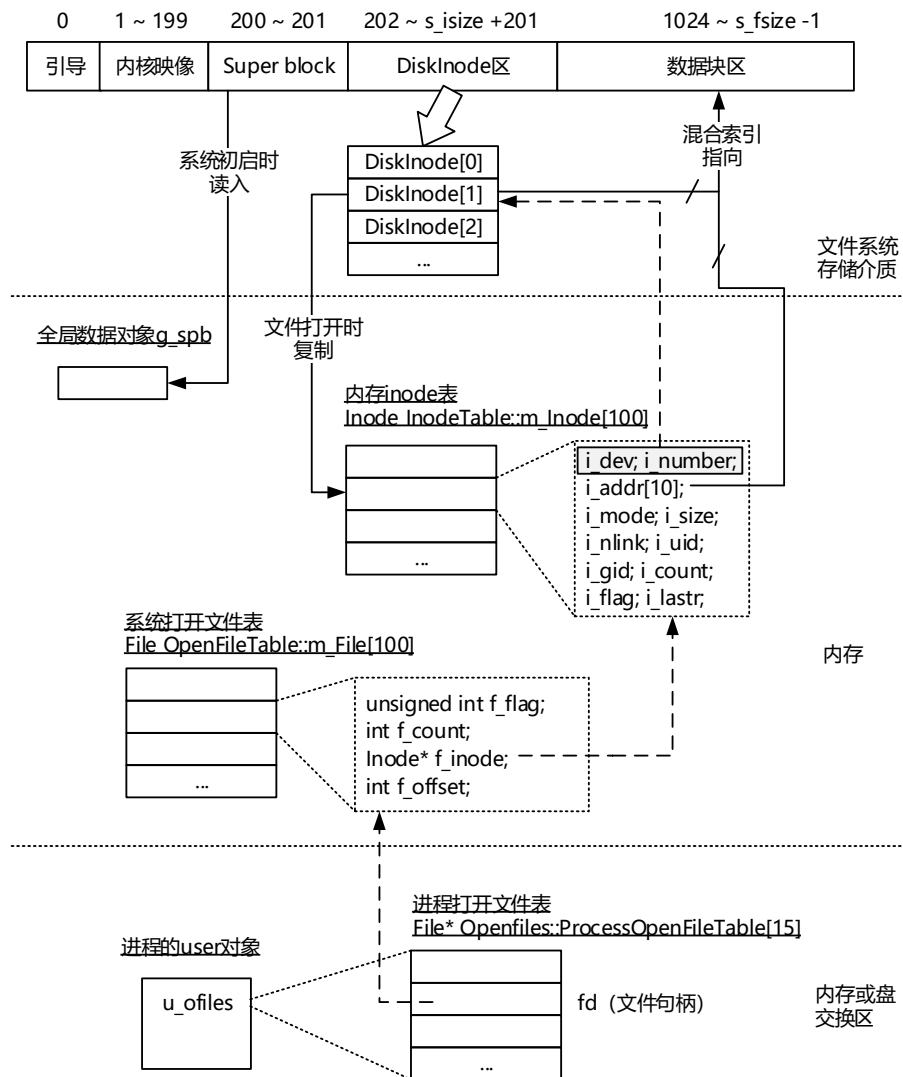


E12: UNIX V6++文件系统

参考答案与说明

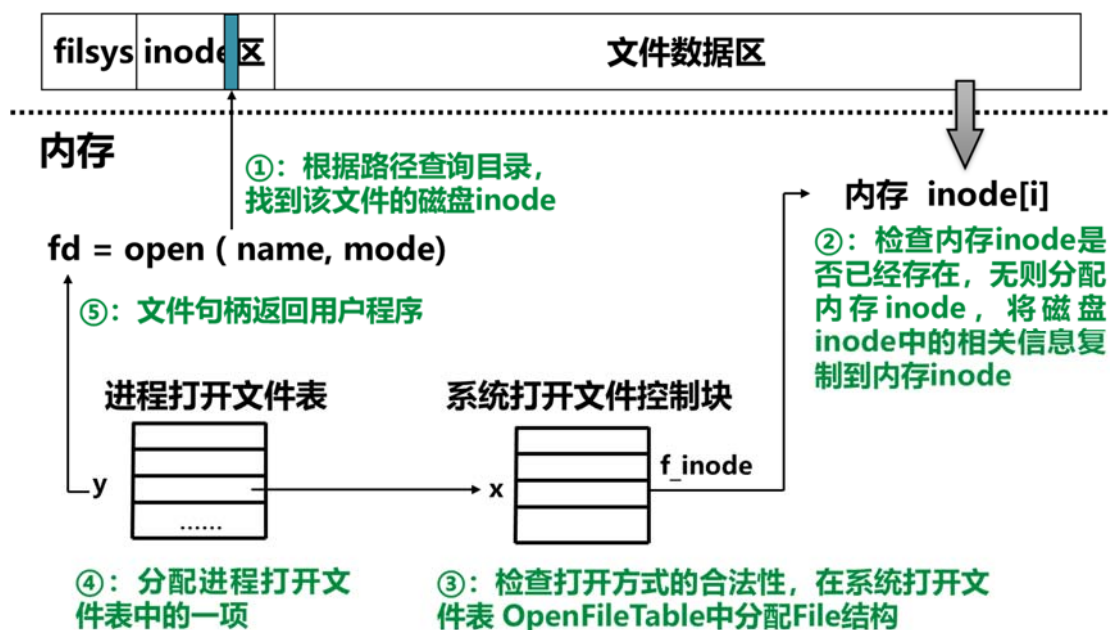
1. A B B B C A BC

2. 【参考答案】UNIX 文件系统涉及的主要数据结构及相互关系如下所示:



3. 【参考答案】文件外存索引节点 DiskInode 用于记录文件的静态信息，包括文件的地址索引结构等。DiskInode 是驻留在外存而不能直接访问的，对它们进行的查询、修改要通过内存进行，按一般方式，可将其临时调入内存，但是这样非常麻烦、费时。UNIX V6++ 文件系统可能是相当庞大的，当用户需要使用某一文件时，对 DiskInode 的访问往往是非常频繁的，按上述方式进行很不经济。文件系统的工作效率一定是十分低下的。所以，从提高系统工作效率出发，需要在内存设置一个非常精炼的文件管理机构。这一机构不应当是外存上文件管理机构的全部拷贝，而应适应于管理最近正在使用的一些文件。而且，为了系统管理和用户使用的方便，对这些文件进行存访、处理是也不再使用符号文件名，而只要求使用整型编号数。这就是设置内存 Inode 节点的目的和意义。

4. 【参考答案】



5. 【参考答案】

目录搜索的过程如下:

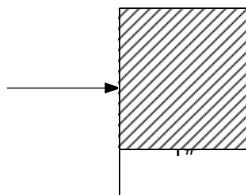
- 根据 1 号 inode (根目录文件的 Inode) 中的 d_addr 找到根目录文件在磁盘中的位置, 从 101 号盘块开始读入根目录文件;
- 在根目录文件中, 逐条记录查找文件名为 usr 的目录项, 该项显示 usr 目录文件所在的 inode 号为 6 号;
- 读入 6 号 Inode (user 文件的 Inode) 所在的盘块, 根据其中的 d_addr 找到 usr 目录文件在磁盘中的位置, 从 132 号盘块开始读入 user 目录文件;
- 在 usr 目录文件中逐条记录查找文件名为 ast 的目录项, 该项显示 ast 目录文件所在的 inode 号为 30 号;
- 读入 30 号 Inode (ast 文件的 Inode) 所在的盘块, 根据其中的 d_addr 找到 ast 目录文件在磁盘中的位置, 从 406 号盘块开始读入 ast 目录文件;
- 在 ast 目录文件中逐条记录查找文件名为 temp 的目录项, 该项显示 temp 文件所在的 inode 号为 80 号;
- 找到 80 号 Inode, 即找到 “/usr/ast/temp” 文件。

6. 【参考答案】

由问题描述可知, Jerry 文件大小为 750 个字节, 即: 该文件包含 2 个逻辑块, 第 1 块为满块, 第 2 块 238 个字节。

(1) `seek(fd, 500, 0);`

文件的读写指针被定位在第 500 个字节, 如下图所示。

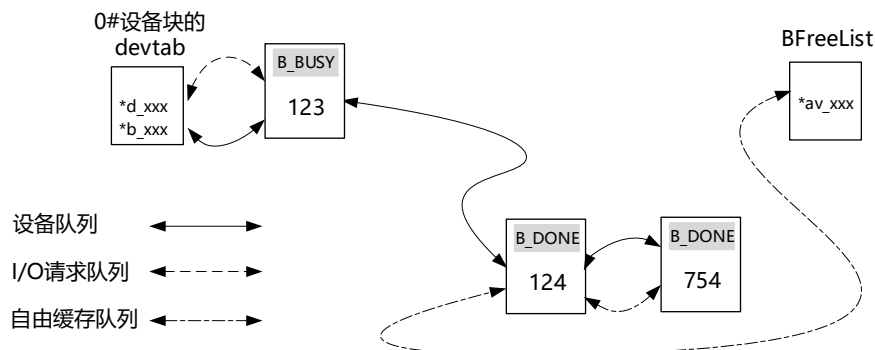


(2) `int count = read (fd, data, 300);`

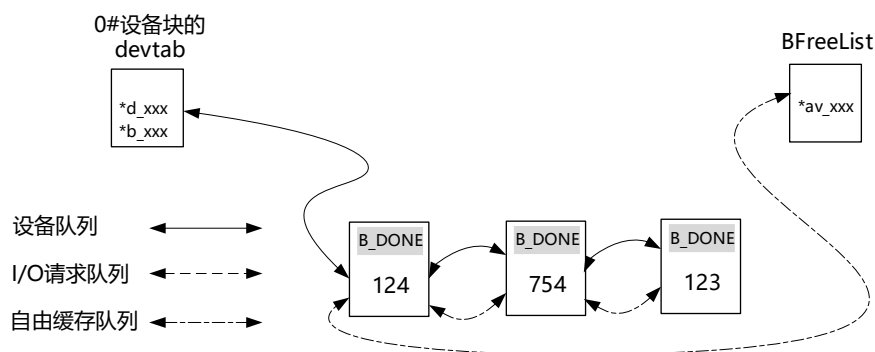
该读操作将分两次盘块读操作完成。

第一次：

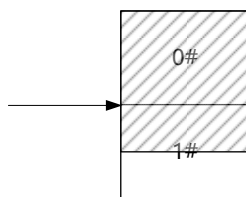
- 通过当前文件读写指针的位置 500 计算得到需读取的逻辑块号为 0 ($500/512=0$)；
- 通过 `i_addr` 查询 Jerry 文件 0 号逻辑块所在的物理盘块号为 123；
- 对该物理盘块申请一个缓存，启动 I/O 之后，进程睡眠等待；



- I/O 结束的中断处理中，会将进程叫醒，进程上台后，将该缓冲区中 500 到 511 字节的内容复制到 `data` 数组；
- 释放缓存；

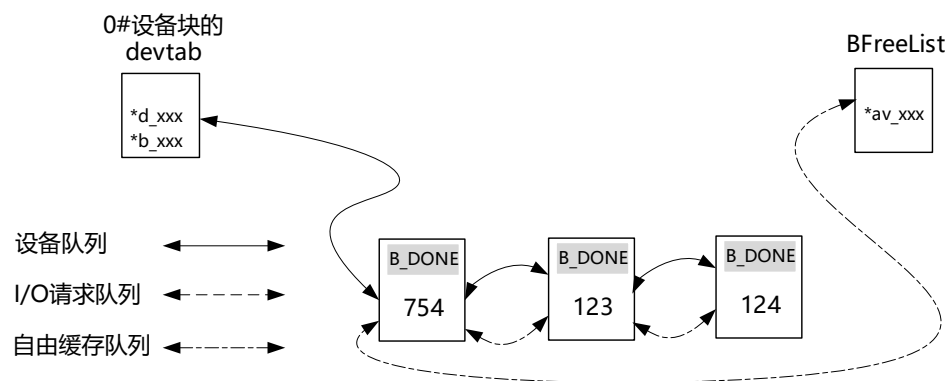


- 修改文件读写指针为 512。

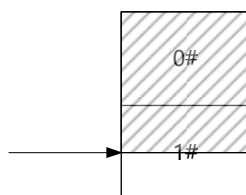


第二次：

- 通过当前文件读写指针的位置 512 计算得到需读取的逻辑块号为 1 ($512/512=1$)；
- 通过 `i_addr` 查询 Jerry 文件 1 号逻辑块所在的物理盘块号 124；
- 对该物理盘块申请一个缓存，发现 124 号盘块的缓存在设备队列中，且有 `B_DONE` 标志，即该缓存中的内容可以直接使用；
- 因为文件到该缓存中的第 237 字节就结束了，所以将该缓冲区中 0 到 237 字节的内容复制到 `data` 数组；
- 释放缓存；



- 修改文件读写指针为 750。

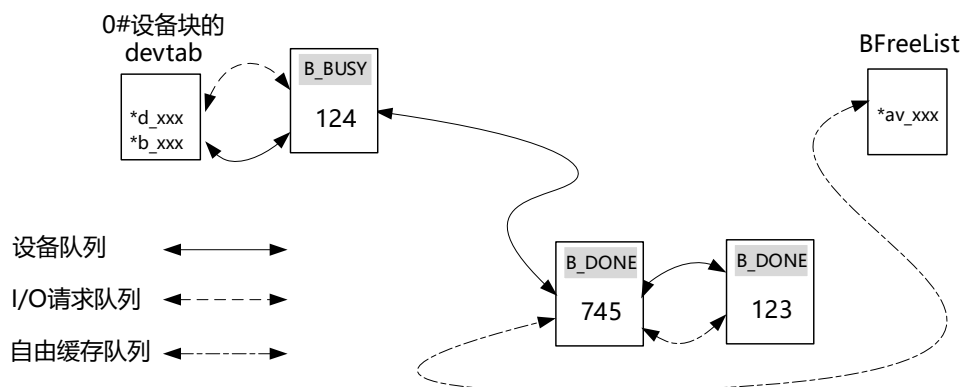
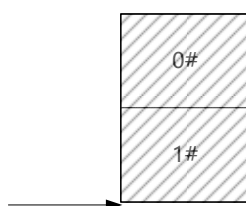


(3) write(fd, data, 300);

该写操作将分两次盘块写操作完成。

第一次：

- 通过当前文件读写指针的位置 750 计算得到需写的逻辑块号为 1 ($750/512=1$)，块内偏移地址为：238，本次需写入的字节数为 274；
- 通过 i_addr 查询 Jerry 文件 1 号逻辑块所在的物理盘块号 124；
- 因为 $274 < 512$ ，即：不是写完整的一块，所以需将 124 号盘块读入；
- 对该物理盘块申请一个缓存，发现 124 号盘块的缓存在设备队列中，且有 B_DONE 标志，即该缓存中的内容可以直接使用；
- 将 data 数组的前 274 个字节写入该缓存的 238 到 511 字节；
- 修改读写指针为 1024，因为正好一块写满，所以执行异步写操作将缓存内容写入磁盘；

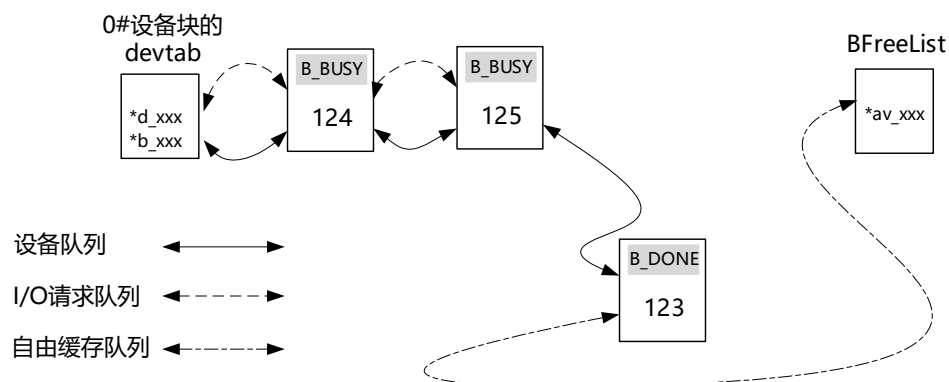


第二次：

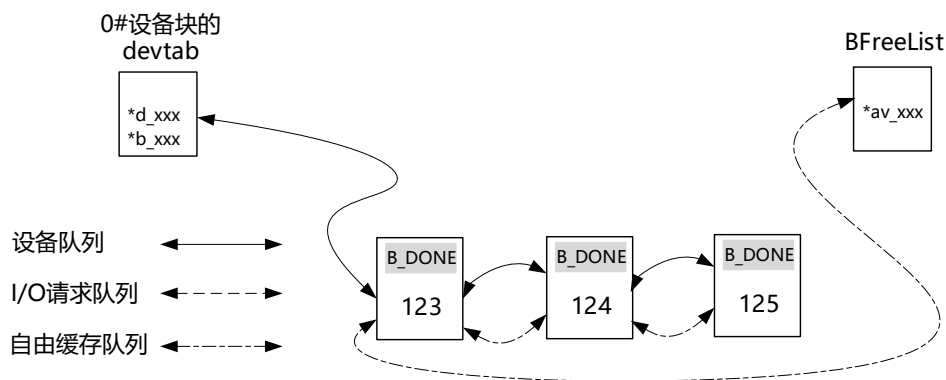
- 通过当前文件读写指针的位置1024计算得到需写的逻辑块号为2($1024/512=2$)，块内偏移地址为：0，本次需写入的字节数为26；
- 通过 `i_addr` 查询 Jerry 文件 2 号逻辑块所在的物理盘块号 0，即：一个新块，所以首先申请一个新的盘块（干净的盘块），假设分配的盘块号为 125，将该盘块号登记入 Jerry 文件的 `i_addr` 数组；

	i_addr
0	123
1	124
2	125
3	0
4	0
5	0
6	0
7	0
8	0
9	0

- 因为 $26 < 512$ ，即：不是写完整的一块，所以需将 125 号盘块读入；
- 对该物理盘块申请一个缓存，启动 I/O 之后，进程睡眠等待；

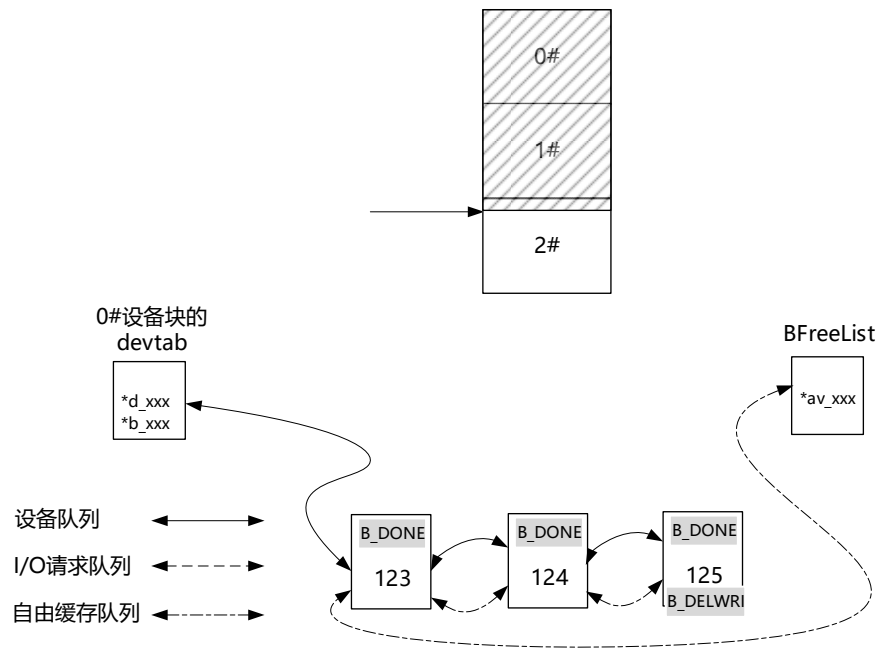


- I/O 结束的中断处理中，会将进程叫醒；



- 进程上台后，将 data 数组的后 26 个字节写入该缓存的前 26 字节；
- 修改读写指针为 1050，因为一块未写满，所以将该缓存添加 B_DELWRI 标志后

直接释放；



- 修改文件长度为 1050。