第六章

文件管理

主要知识点

基本概念：

- 设备管理部分
  - 磁盘调度算法
- 文件管理部分
  - 文件物理结构：顺序文件、链接文件（隐式、显式）
  - 磁盘空间管理：位视图、空白文件

主要知识点：

UNIX文件系统：

- 文件系统的磁盘结构

- 文件系统的内存打开结构

主要知识点：

UNIX文件系统：

- 文件系统的磁盘结构
  - SuperBlock（1024字节）：空闲Inode栈，空闲数据盘块栈（成组链接法）

文件系统挂载成功后，所有的分配和回收都由内存SuperBlock完成，卸载时写回

- 文件系统的内存打开结构
  - SuperBlock（1024字节）：文件系统挂载时创建内存副本

主要知识点：

UNIX文件系统：

- 文件系统的磁盘结构
  - SuperBlock（1024字节）：空闲Inode栈，空闲数据盘块栈（成组链接法）
  - Inode（64字节）：三级混合索引结构，d_mode，d_size，d_nlink

文件打开成功后，所有的操作都由内存打开结构完成，关闭时写回

- 文件系统的内存打开结构
  - SuperBlock（1024字节）：文件系统挂载时创建内存副本
  - Inode（64字节）：文件打开时创建内存Inode和内存File，i_mode，i_addr，i_size，i_nlink

主要知识点：

UNIX文件系统：

- 文件系统的磁盘结构
  - SuperBlock（1024字节）：空闲Inode栈，空闲数据盘块栈（成组链接法）
  - Inode（64字节）：三级混合索引结构，d_mode，d_size，d_nlink
  - 树状目录结构：通过目录文件实现目录结构的构建，检索，删除，增加）

- 文件系统的内存打开结构
  - SuperBlock（1024字节）：文件系统挂载时创建内存副本
  - Inode（64字节）：文件打开时创建内存Inode和内存File，i_mode，i_addr，i_size，i_nlink
  - 树状目录结构：文件系统挂载时根目录文件创建内存Inode

每一级目录文件的解析都需要内存打开，修改后需要写回

主要知识点：

UNIX文件系统：

- 文件操作的实施
  - Link，Unlink
  - Creat
  - Open，Close
  - Seek，Read，Write

对目录结构的检索和修改

主要知识点：

UNIX文件系统：

- 文件操作的实施
  - Link，Unlink
  - Creat

  对目录结构的检索和修改
  - Open，Close

  文件内存打开结构的建立、共享和撤销
  - Seek，Read，Write

  文件的地址映射、通过缓存的读写过程

主要知识点：

UNIX文件系统：

- 文件操作的实施
  - Link，Unlink
  - Creat

  对目录结构的检索和修改
  - Open，Close

  文件内存打开结构的建立、共享和撤销
  - Seek，Read，Write

  文件的地址映射、通过缓存的读写过程

UNIX的设备管理系统：

- 缓存队列：分析缓存分配过程
- 基于共享缓存的读写技术：同步读，异步读，异步写，延迟写

主要知识点：

UNIX文件系统：

- 文件操作的实施
  - Link，Unlink
  - Creat
  - Open，Close
  - Seek，Read，Write

对目录结构的检索和修改

文件内存打开结构的建立、共享和撤销

文件的地址映射、通过缓存的读写过程

读：无预读（同步读当前块）
有预读（同步读当前块+异步读下一块）
写：先同步读，再异步写
异步写
延迟写

UNIX的设备管理系统：

- 缓存队列
- 基于共享缓存的读写技术：同步读，异步读，异步写，延迟写

例题

假设此时系统中只有一个进程pa在进行下列IO操作，且此时缓存中没有任何与文件Jerry相关的内容。

（1）该文件系统中的所有目录文件内容如下图所示，请绘制出其对应的目录结构。

root

.	1
bin	4
usr	6
home	78

usr

.	6
..	1
Jerry	19
temp	45
books	26

bin

.	4
..	1

home

.	78
..	1

root文件的Inode

……
d_addr[0]= 3000
d_addr[1] = 0
……

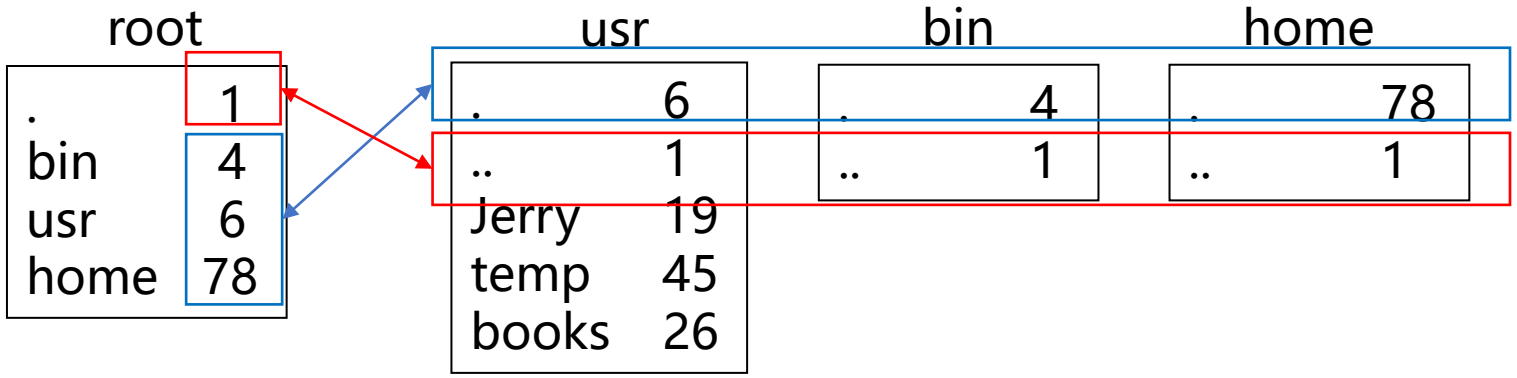
usr文件的Inode

……
d_addr[0]= 6000
d_addr[1] = 0
……

例题

假设此时系统中只有一个进程pa在进行下列IO操作，且此时缓存中没有任何与文件Jerry相关的内容。

（1）该文件系统中的所有目录文件内容如下图所示，请绘制出其对应的目录结构。

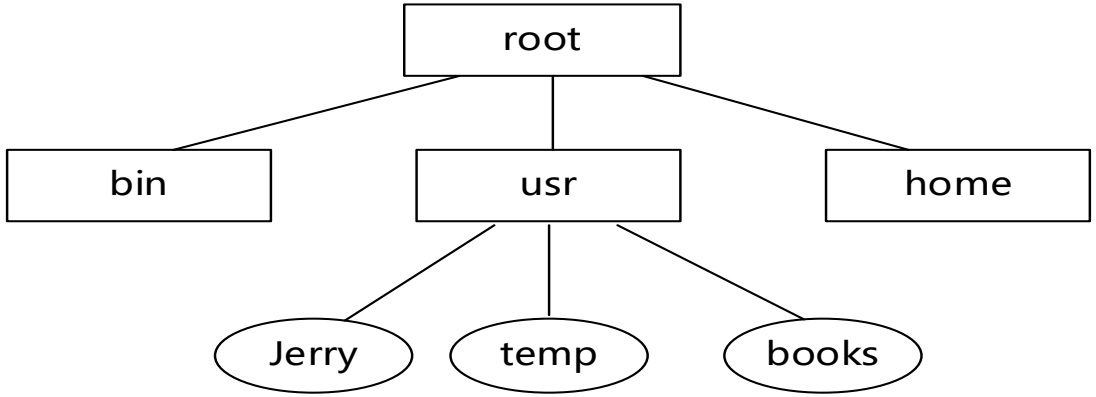


root文件的Inode

```
......
d_addr[0]= 3000
d_addr[1] = 0
......
```

usr文件的Inode

```
......
d_addr[0]= 6000
d_addr[1] = 0
......
```



例题

假设此时系统中只有一个进程pa在进行下列IO操作，且此时缓存中没有任何与文件Jerry相关的内容。

（2）如果进程pa执行了link（“/usr/Jerry”,”/Jerry1”），请绘制该操作结束后的目录结构和目录文件。

root

.	1
bin	4
usr	6
home	78
Jerry1	19

usr

.	6
..	1
Jerry	19
temp	45
books	26

bin

.	4
..	1

home

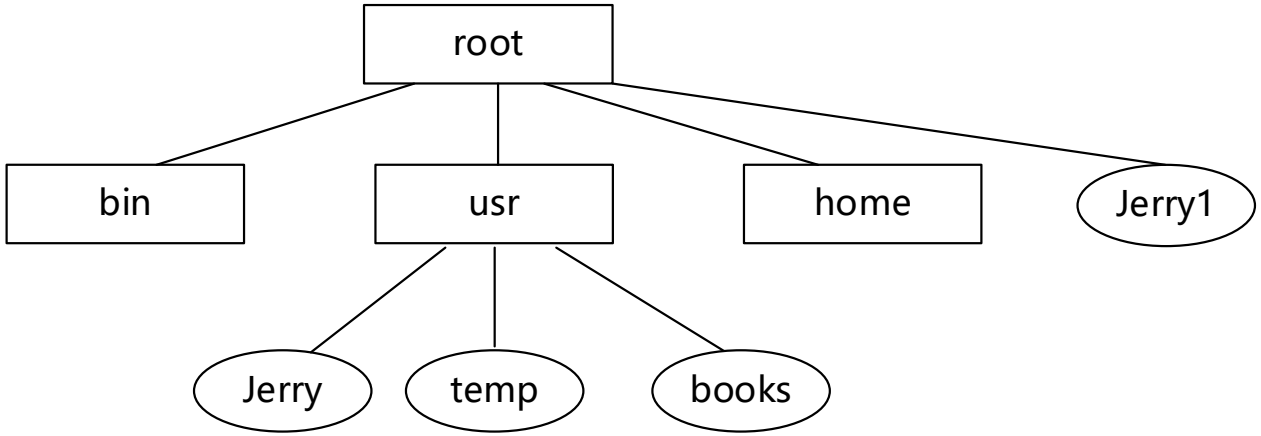
.	78
..	1

root文件的Inode

……
d_addr[0]= 3000
d_addr[1] = 0
……

usr文件的Inode

……
d_addr[0]= 6000
d_addr[1] = 0
……



例题

假设此时系统中只有一个进程pa在进行下列IO操作，且此时缓存中没有任何与文件Jerry相关的内容。

（3）如果进程pa以可读可写的方式打开“/usr/Jerry”，在此过程中，需读入__4__个盘块。

root

.	1
bin	4
usr	6
home	78
Jerry1	19

bin

.	4
..	1

home

.	78
..	1

usr

.	6
..	1
Jerry	19
temp	45
books	26

自由队列：
3000 202 6000 204

设备队列：
204 6000 202 3000

1. 根据1#Inode创建root文件的内存Inode；（文件系统挂载时以完成，这里无需IO）
2. 读入root文件，逐条记录搜索usr，找到usr目录文件的磁盘Inode号6；（同步读入root文件0#逻辑块所在盘块）
3. 根据6#Inode创建usr文件的内存Inode；（同步读入6#磁盘Inode所在盘块）
4. 读入usr文件，诸条记录搜索Jerry，找到Jerry文件的磁盘Inode号19；（同步读入usr文件0#逻辑块所在盘块）
5. 查找成功，根据19#Inode创建Jerry文件的内存Inode；（同步读入19#磁盘Inode所在盘块）
6. 返回指向该Inode的指针。

例题

假设此时系统中只有一个进程pa在进行下列IO操作，且此时缓存中没有任何与文件Jerry相关的内容。

（3）如果进程pa以可读可写的方式打开“/usr/Jerry”，在此过程中，需读入__4__个盘块。

root

.	1
bin	4
usr	6
home	78
Jerry1	19

bin

.	4
..	1

home

.	78
..	1

usr

.	6
..	1
Jerry	19
temp	45
books	26

1. 根据1#Inode创建root文件的内存Inode；（文件系统挂载时以完成，这里无需IO）
2. 读入root文件，逐条记录搜索usr，找到usr目录文件的磁盘Inode号6；（读入root文件0#逻辑块所在盘块）
3. 根据6#Inode创建usr文件的内存Inode；（读入6#磁盘Inode所在盘块）
4. 读入usr文件，诸条记录搜索Jerry，找到Jerry文件的磁盘Inode号19；（读入usr文件0#逻辑块所在盘块）
5. 查找成功，根据19#Inode创建Jerry文件的内存Inode；（读入19#磁盘Inode所在盘块）
6. 返回指向该Inode的指针。

如果进程已将”/usr”设置为当前工作目录，可以用文件名”Jerry”打开该文件，需读入_____个盘块。

例题

假设此时系统中只有一个进程pa在进行下列IO操作，且此时缓存中没有任何与文件Jerry相关的内容。

（3）如果进程pa以可读可写的方式打开“/usr/Jerry”，在此过程中，需读入__4__个盘块。

root

.	1
bin	4
usr	6
home	78
Jerry1	19

bin

.	4
..	1

home

.	78
..	1

usr

.	6
..	1
Jerry	19
temp	45
books	26

1. 根据1#Inode创建root文件的内存Inode；（文件系统挂载时以完成，这里无需IO）
2. 读入root文件，逐条记录搜索usr，找到usr目录文件的磁盘Inode号6；（读入root文件0#逻辑块所在盘块）
3. 根据6#Inode创建usr文件的内存Inode；（读入6#磁盘Inode所在盘块）
4. 读入usr文件，诸条记录搜索Jerry，找到Jerry文件的磁盘Inode号19；（读入usr文件0#逻辑块所在盘块）
5. 查找成功，根据19#Inode创建Jerry文件的内存Inode；（读入19#磁盘Inode所在盘块）
6. 返回指向该Inode的指针。

如果进程已将”/usr”设置为当前工作目录，可以用文件名”Jerry”打开该文件，需读入__2__个盘块。

例题

假设此时系统中只有一个进程pa在进行下列IO操作，且此时缓存中没有任何与文件Jerry相关的内容。

（4）如果Jerry文件长度为4500字节，系统需要为其分配多少磁盘存储资源？填写文件Jerry的d_addr数组并画出其混合索引结构，分配给该文件的盘块号可以自行指定。

4500/512 = 8，4500%512 = 404

该文件共需占用9个数据盘块，1个索引盘块。

64个字节的一个磁盘Inode节点

32个字节的一个目录项

d_addr数组：

0	4000
1	4001
2	4002
3	4003
4	4004
5	4005
6	4006
7	0
8	0
9	0

4006#盘块

0	4007
1	4008
2	4009
3	0
4	0
5	0
…	…
127	0

例题

假设此时系统中只有一个进程pa在进行下列IO操作，且此时缓存中没有任何与文件Jerry相关的内容。

（5）pa进程在成功打开Jerry文件后，如果分别独立进行下面的读操作，请回答：

① 访问偏移量是0的字节，此时是顺序读还是随机读？ 顺序读
需要读入 2 个盘块。

文件刚打开时，i_lastr = -1，此时读0#逻辑块，满足预读条件，所以将同步读入4000#盘块，异步读入4001号盘块

② 访问偏移量是2500的字节，此时是顺序读还是随机读？ 随机读
需要读入 1 个盘块。紧接着访问偏移量是3000的字节，
需要读入 1 个盘块。

2500 / 512 = 4，读4#逻辑块，直接索引，不预读，所以只需同步读入4004#盘块

3000 / 512 = 5，读5#逻辑块，虽然满足预读条件，但是读入6#逻辑块需要额外读入索引块，所以放弃预读。因此，只需同步读入4005#盘块

i_addr数组：

0	4000
1	4001
2	4002
3	4003
4	4004
5	4005
6	4006
7	0
8	0
9	0

4006#盘块

0	4007
1	4008
2	4009
3	0
4	0
5	0
…	…
127	0

例题

假设此时系统中只有一个进程pa在进行下列IO操作，且此时缓存中没有任何与文件Jerry相关的内容。

（5）pa进程在成功打开Jerry文件后，如果分别独立进行下面的读操作，请回答：

③ 访问偏移量是3500的字节，需要读入__2__个盘块。紧接着访问偏移量是3750的字节，需要读入__2__个盘块。

3500 / 512 = 6，读6#逻辑块，一次间接索引，不预读，所以需同步读入4006#索引块，再读入4007#数据块

3750 / 512 = 7，读7#逻辑块，满足预读条件，且索引块缓存命中，不需额外读入。因此，需同步读入4008#盘块，异步读入4009#数据块

i_addr数组：

0	4000
1	4001
2	4002
3	4003
4	4004
5	4005
6	4006
7	0
8	0
9	0

4006#盘块

0	4007
1	4008
2	4009
3	0
4	0
5	0
…	…
127	0

例题

假设此时系统中只有一个进程pa在进行下列IO操作，且此时缓存中没有任何与文件Jerry相关的内容。

（6）当前内存SuperBlock中的空闲Inode栈和空闲盘块栈的内容如下图所示：

磁盘Inode栈

s_ninode：85
s_inode[0]：12
……
s_inode[84]：20

空闲盘块栈

s_nfree：100
s_free[0]：1000
……
……
s_free[99]：5000

i_addr数组：

0	4000
1	4001
2	4002
3	4003
4	4004
5	4005
6	4006
7	0
8	0
9	0

4006#盘块

0	4007
1	4008
2	4009
3	0
4	0
5	0
…	…
127	0

如果pa从文件结束位置开始，继续向该文件中写入500个字节，会有哪些数据结构发生变化？

为该文件分配一个新的盘块

例题

假设此时系统中只有一个进程pa在进行下列IO操作，且此时缓存中没有任何与文件Jerry相关的内容。

（6）当前内存SuperBlock中的空闲Inode栈和空闲盘块栈的内容如下图所示：

磁盘Inode栈

s_ninode：85 s_inode[0]：12 …… s_inode[84]：20

空闲盘块栈

s_nfree：99 s_free[0]：1000 …… …… s_free[99]：

i_addr数组：

0	4000
1	4001
2	4002
3	4003
4	4004
5	4005
6	4006
7	0
8	0
9	0

4006#盘块

0	4007
1	4008
2	4009
3	5000
4	0
5	0
…	…
127	0

如果pa从文件结束位置开始，继续向该文件中写入500个字节，会有哪些数据结构发生变化?

为该文件分配一个新的盘块，内存SuperBlock改变；
内存Inode中的i_size修改为5000。

例题

假设此时系统中只有一个进程pa在进行下列IO操作，且此时缓存中没有任何与文件Jerry相关的内容。

（6）当前内存SuperBlock中的空闲Inode栈和空闲盘块栈的内容如下图所示：

磁盘Inode栈

s_ninode：85 s_inode[0]：12 …… s_inode[84]：20

空闲盘块栈

s_nfree：99 s_free[0]：1000 …… …… s_free[99]：

i_addr数组：

0	4000
1	4001
2	4002
3	4003
4	4004
5	4005
6	4006
7	0
8	0
9	0

4006#盘块

0	4007
1	4008
2	4009
3	5000
4	0
5	0
…	…
127	0

如果pa从文件结束位置开始，继续向该文件中写入500个字节，会有哪些数据结构发生变化？其后，如果pa立即关闭该文件，会有哪些数据不一致的情况？

写8#逻辑块：写完，异步写回4009#盘块（可能会在关闭文件后完成）

写9#逻辑块：修改了4006#索引块所在缓存，延迟写；
修改了5000#数据块所在缓存，延迟写。

关闭文件后：修改的内存Inode写回磁盘。

何时写回磁盘不一定。所以关闭文件时，索引盘块和新的数据盘块的内容是不对的。

例题

假设此时系统中只有一个进程pa在进行下列IO操作，且此时缓存中没有任何与文件Jerry相关的内容。

（6）当前内存SuperBlock中的空闲Inode栈和空闲盘块栈的内容如下图所示：

磁盘Inode栈

s_ninode：85 s_inode[0]：12 …… s_inode[84]：20

空闲盘块栈

s_nfree：99 s_free[0]：1000 …… …… s_free[99]：

i_addr数组：

0	4000
1	4001
2	4002
3	4003
4	4004
5	4005
6	4006
7	0
8	0
9	0

4006#盘块

0	4007
1	4008
2	4009
3	5000
4	0
5	0
…	…
127	0

如果后续有进程再读取该文件，这些数据不一致会出错么？

如果再打开时，延迟写的缓存已经写回磁盘，则数据恢复一致，不会出错。

如果再打开时，延迟写的缓存还未写回磁盘，则后续的读写操作缓存会命中，不会出错。

如果还未来得及写回，系统掉电，再打开，会出错。

定时将延迟写的缓存写回磁盘

例题

假设此时系统中只有一个进程pa在进行下列IO操作，且此时缓存中没有任何与文件Jerry相关的内容。

（7）如果进程pa在关闭该文件后，执行unlink(“/usr/Jerry”)操作：

磁盘Inode栈

s_ninode：85
s_inode[0]：12
……
s_inode[84]：20

空闲盘块栈

s_nfree：99
s_free[0]：1000
……
……
s_free[99]：

修改目录文件，因为还有一条路径，无需删除物理文件

root

.	1
bin	4
usr	6
home	78
Jerry1	19

bin

.	4
..	1

home

.	78
..	1

usr

.	6
..	1
~~Jerry~~	~~19~~
temp	45
books	26

i_addr数组：

0	4000
1	4001
2	4002
3	4003
4	4004
5	4005
6	4006
7	0
8	0
9	0

4006#盘块

0	4007
1	4008
2	4009
3	5000
4	0
5	0
…	…
127	0

例题

假设此时系统中只有一个进程pa在进行下列IO操作，且此时缓存中没有任何与文件Jerry相关的内容。

（8）如果进程pa继续执行unlink(“/Jerry1”)操作：

磁盘Inode栈

s_ninode：85
s_inode[0]：12
……
s_inode[84]：20

空闲盘块栈

s_nfree：99
s_free[0]：1000
……
……
s_free[99]：

i_addr数组：

0	4000
1	4001
2	4002
3	4003
4	4004
5	4005
6	4006
7	0
8	0
9	0

4006#盘块

0	4007
1	4008
2	4009
3	5000
4	0
5	0
…	…
127	0

修改目录文件

root

.	1
bin	4
usr	6
home	78
~~Jerry1~~	~~19~~

bin

.	4
..	1

home

.	78
..	1

usr

.	6
..	1
temp	45
books	26

例题

假设此时系统中只有一个进程pa在进行下列IO操作，且此时缓存中没有任何与文件Jerry相关的内容。

（8）如果进程pa继续执行unlink(“/Jerry1”)操作：

磁盘Inode栈

s_ninode：85 s_inode[0]：12 …… s_inode[84]：20

空闲盘块栈

s_nfree：99 s_free[0]：1000 …… …… s_free[99]：

i_addr数组：

0	4000
1	4001
2	4002
3	4003
4	4004
5	4005
6	4006
7	0
8	0
9	0

4006#盘块

0	4007
1	4008
2	4009
3	5000
4	0
5	0
…	…
127	0

需要物理删除该文件。释放该文件占用的所有盘块号（按分配相反的顺序）

空闲盘块栈

s_nfree：100 s_free[0]：1000 …… …… s_free[99]：5000

空闲盘块栈

s_nfree：10 s_free[0]：4009 …… s_free[8]：4001 s_free[9]：4000

4009#盘块的前101个字

100 1000 …… …… 5000

例题

假设此时系统中只有一个进程pa在进行下列IO操作，且此时缓存中没有任何与文件Jerry相关的内容。

（8）如果进程pa继续执行unlink(“/Jerry1”)操作：

磁盘Inode栈

s_ninode：85
s_inode[0]：12
……
s_inode[84]：20

空闲盘块栈

s_nfree：99
s_free[0]：1000
……
……
s_free[99]：

i_addr数组：

0	4000
1	4001
2	4002
3	4003
4	4004
5	4005
6	4006
7	0
8	0
9	0

4006#盘块

0	4007
1	4008
2	4009
3	5000
4	0
5	0
…	…
127	0

需要物理删除该文件。释放该文件占用的所有盘块号（按分配相反的顺序）；释放该文件占用的Inode

磁盘Inode栈

s_ninode：86
s_inode[0]：12
……
s_inode[84]：20
s_inode[84]：19

例题

假设此时系统中只有一个进程pa在进行下列IO操作，且此时缓存中没有任何与文件Jerry相关的内容。

（9）如果进程继续创建一个新文件（“/usr/Tom”），并向其中写入“Hello World!”：

分配磁盘Inode：

磁盘Inode栈

s_ninode：85
s_inode[0]：12
……
s_inode[84]：20

修改目录文件

root

.	1
bin	4
usr	6
home	78

bin

.	4
..	1

home

.	78
..	1

usr

.	6
..	1
Tom	19
temp	45
books	26

例题

假设此时系统中只有一个进程pa在进行下列IO操作，且此时缓存中没有任何与文件Jerry相关的内容。

（9）如果进程继续创建一个新文件（“/usr/Tom”），并向其中写入“Hello World!”：

分配空闲盘块：

空闲盘块栈

s_nfree：9
s_free[0]：4009
……
s_free[8]：4001

4009#盘块的前101个字

100
1000
……
……
5000

i_addr数组：

0	4000
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0

修改索引结构：