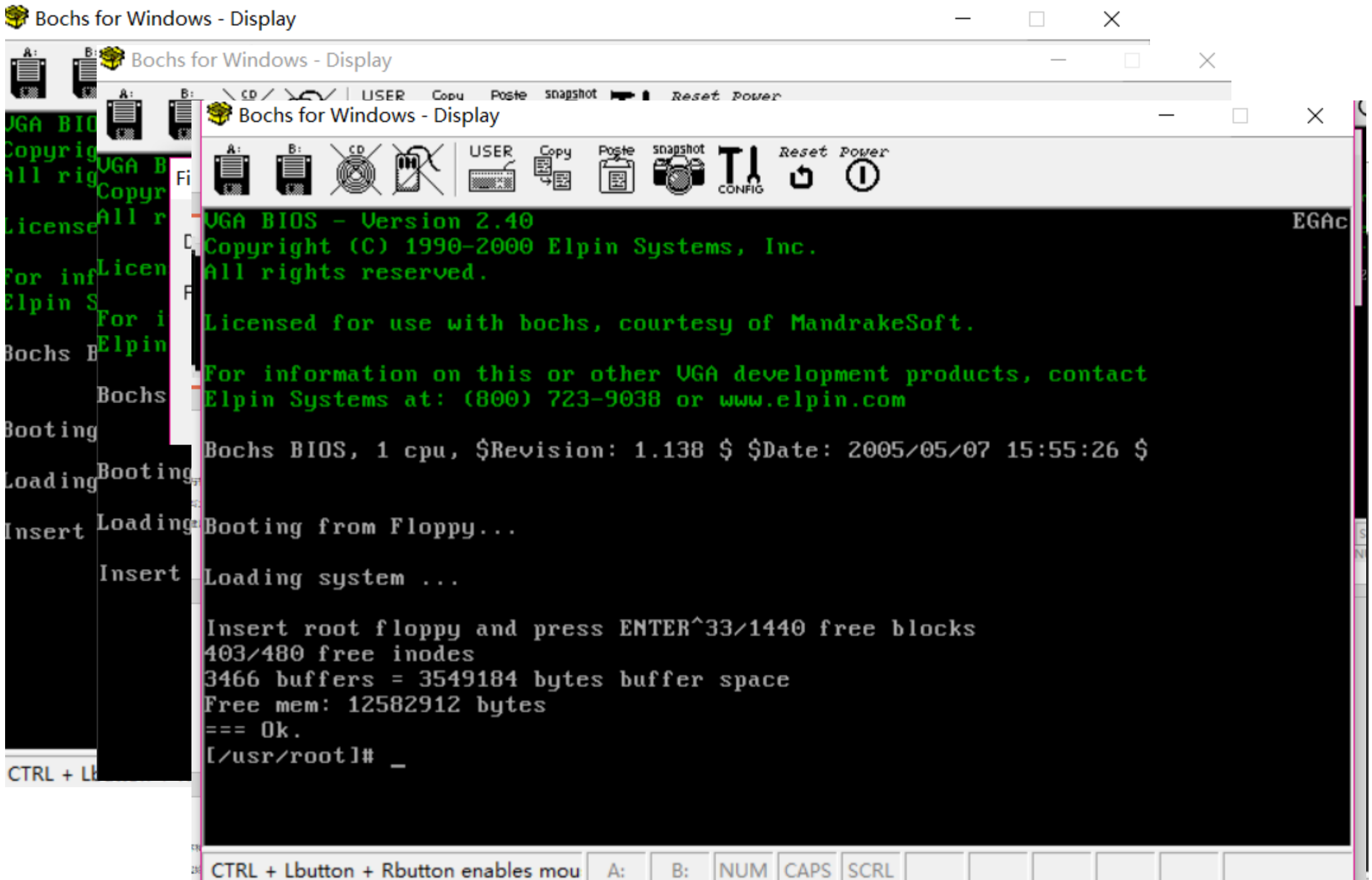
Linux操作系统运行过程可视化—文件系统

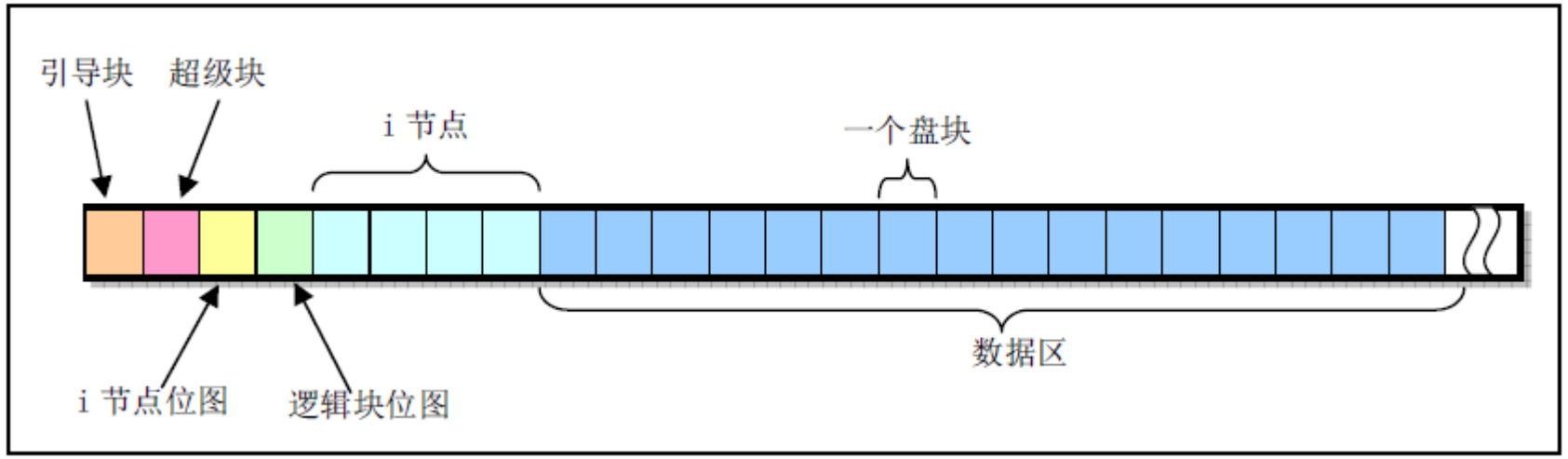
Linux

王文嵩


Bochs for Windows - Display
Bochs for Windows - Display
Bochs for Windows - Display
A:
B:
CD
USER
Copy
Poste
snapshot
Reset
Power
CONFIG
VGA BIOS - Version 2.40
Copyright (C) 1990-2000 Elpin Systems, Inc.
All rights reserved.
Licensed for use with bochs, courtesy of MandrakeSoft.
For information on this or other VGA development products, contact
Elpin Systems at: (800) 723-9038 or www.elpin.com
Bochs BIOS, 1 cpu, $Revision: 1.138 $ $Date: 2005/05/07 15:55:26 $
Booting from Floppy...
Loading system ...
Insert root floppy and press ENTER^33/1440 free blocks
403/480 free inodes
3466 buffers = 3549184 bytes buffer space
Free mem: 12582912 bytes
=== Ok.
[/usr/root]# _
CTRL + Lbutton + Rbutton enables mou
A:
B:
NUM
CAPS
SCRL

- 高速缓冲区的管理程序，对硬盘等块设备进行数据高速存取
- 文件索引节点的管理、磁盘数据块的分配和释放以及文件名与i节点的转换算法
- 对文件中数据进行读写操作，包括对字符设备、管道、块读写文件中数据的访问
- 文件的系统调用接口的实现，主要涉及文件打开、关闭、创建以及有关文件目录操作等的系统调用。

MINIX文件系统



- 硬盘的第一个扇区是主引导扇区，其中存放着硬盘引导程序和分区表信息。
- 分区表中的信息指明了硬盘上每个分区的类型、在硬盘中起始位置的参数和结束位置参数以及占用的扇区总数

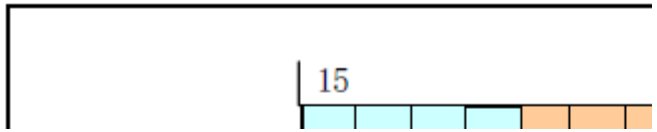
```
static struct hd_struct
{
    long start_sect;
    long nr_sects;
}

hd[5 * MAX_HD] =
{
    {0, 0},
};
```

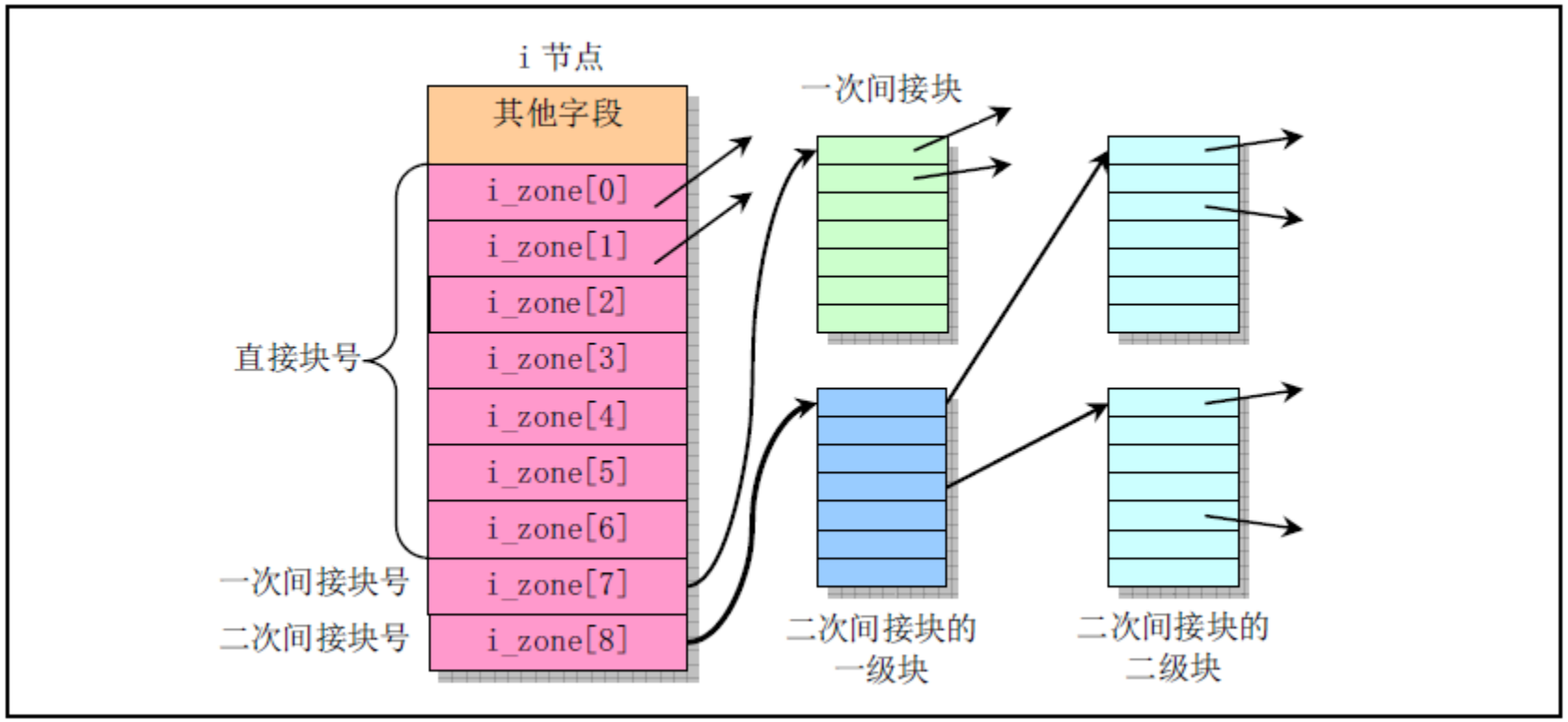
- 超级块用于存放盘设备上文件系统的一些参数，如i节点数、逻辑块数、逻辑块大小。

```
#define NR_
```

	字段名称	数据类型
出现在盘上和内存中的字段	s_ninodes	short
	s_nzones	short
	s_imap_blocks	short
	s_zmap_blocks	short
	s_firstdatazone	short
	s_log_zone_size	short
	s_max_size	long
	s_magic	short
仅在内存中使用的字段	s_imap[8]	buffer_head
	s_zmap[8]	buffer_head
	s_dev	short
	s_isup	m_inode *
	s_imount	m_inode *
	s_time	long
	s_wait	task_struct
	s_lock	char
	s_rd_only	char
	s_dirt	char

```
242: void mount_root(void)
243: {
244:     int i,free;
245:     struct super_block * p;
246:     struct m_inode * mi;
247:
248:     if (32 != sizeof (struct d_inode))
249:         panic("bad i-node size");
250:     for(i=0;i<NR_FILE;i++)
251:         file_table[i].f_count=0;
252:     if (MAJOR(ROOT_DEV) == 2) {
253:         printk("Insert root floppy and press ENTER");
254:         wait_for_keypress();
255:     }
256:     for(p = &super_block[0] ; p < &super_block[NR_SUPER] ; p++) {
257:         p->s_dev = 0;
258:         p->s_lock = 0;
259:         p->s_wait = NULL;
260:     }
261:     if (!(p=read_super(ROOT_DEV)))
262:         panic("Unable to mount root");
263:     if (!(mi=iget(ROOT_DEV,ROOT_INO)))
264:         panic("Unable to read root i-node");
265:     mi->i_count += 3 ;  /* NOTE! it is logically used 4 times, not 1 */
266:     p->s_isup = p->s_imount = mi;
267:     current->pwd = mi;
268:     current->root = mi;
269:     free=0;
270:     i=p->s_nzones;
271:     while (-- i >= 0)
272:         if (!set_bit(i&8191,p->s_zmap[i>>13]->b_data))
273:             free++;
274:     printk("%d/%d free blocks\n\r",free,p->s_nzones);
275:     free=0;
276:     i=p->s_ninodes+1;
277:     while (-- i >= 0)
278:         if (!set_bit(i&8191,p->s_imap[i>>13]->b_data))
279:             free++;
280:     printk("%d/%d free inodes\n\r",free,p->s_ninodes);
281: } « end mount_root »
```





字段名称	数据类型	说明
i_mode	short	文件的类型和属性（rwx 位）
i_uid	short	文件宿主的用户 id
		节）
		1970.1.1:0 时算起，秒）
		id
		多个文件目录项指向该 i 节点）
		盘上逻辑块号数组。其中：
		6]是直接块号；
		间接块号；
		（双重）间接块号。
		的意思，可译成区块或逻辑块。
		文件名的 i 节点，其 zone[0]中
		件名所指设备的设备号。
		的进程。
		改时间。
		备号。
		次数，0 表示空闲。
i_lock	char	i 节点被锁定标志。
i_dirt	char	i 节点已被修改（脏）标志。
i_pipe	char	i 节点用作管道标志。
i_mount	char	i 节点安装了其他文件系统标志。
i_seek	char	搜索标志（lseek 操作时）。
i_update	char	i 节点已更新标志。

- 逻辑位图最多使用8块缓冲块（s_zmap[8]）buffer 1024B，一个buffer可表示8192个盘块，故共65536个盘块，最大块设备容量是64MB
- 一个盘块可表示8192个i节点的使用状况
- 文件中的数据是放在磁盘块的数据区中的，而一个文件名则通过对应的i节点与数据磁盘块相联系，这些盘块的号码就存放在i节点的逻辑块数组i_zone[]中。0-6直接块7：512 8：512*512
- /dev/设备文件，不占用磁盘数据区中的数据盘块，i节点仅保存定义设备的属性和设备号。Zone[0]设备号
- 两个扇区作为一个数据块来处理，称为磁盘块或盘块。长度与高速缓冲区的缓冲块长度相同

```
[/usr/root]# ls
a.out    hello.c
[/usr/root]# mkdir demo
[/usr/root]# ls
a.out    demo     hello.c
[/usr/root]# cd demo
[/usr/root/demo]# ls
[/usr/root/demo]# touch hello.c
[/usr/root/demo]# cat hello.c
[/usr/root/demo]# ls
hello.c
[/usr/root/demo]# _
```

```c
#include<stdio.h>
int main()
{
  printf("hello,world!\n");
  return 0;
}
```

```
[/usr/root/demo]# rm hello.c
[/usr/root/demo]# cd ..
[/usr/root]# rm demo
rm: demo: is a directory
[/usr/root]# rm -r demo
```

文件名目录项

- 在文件系统的一个目录中，其中所有文件名信息对应的目录项存储在目录文件名文件的数据块中。
- 文件系统根目录下的所有文件名信息则保存在指定i节点（即1号i节点）的数据块中。

```
36: #define NAME_LEN 14
37: #define ROOT_INO 1
38:
```

```
157: struct dir_entry {
158:     unsigned short inode;
159:     char name[NAME_LEN];
160: };
```

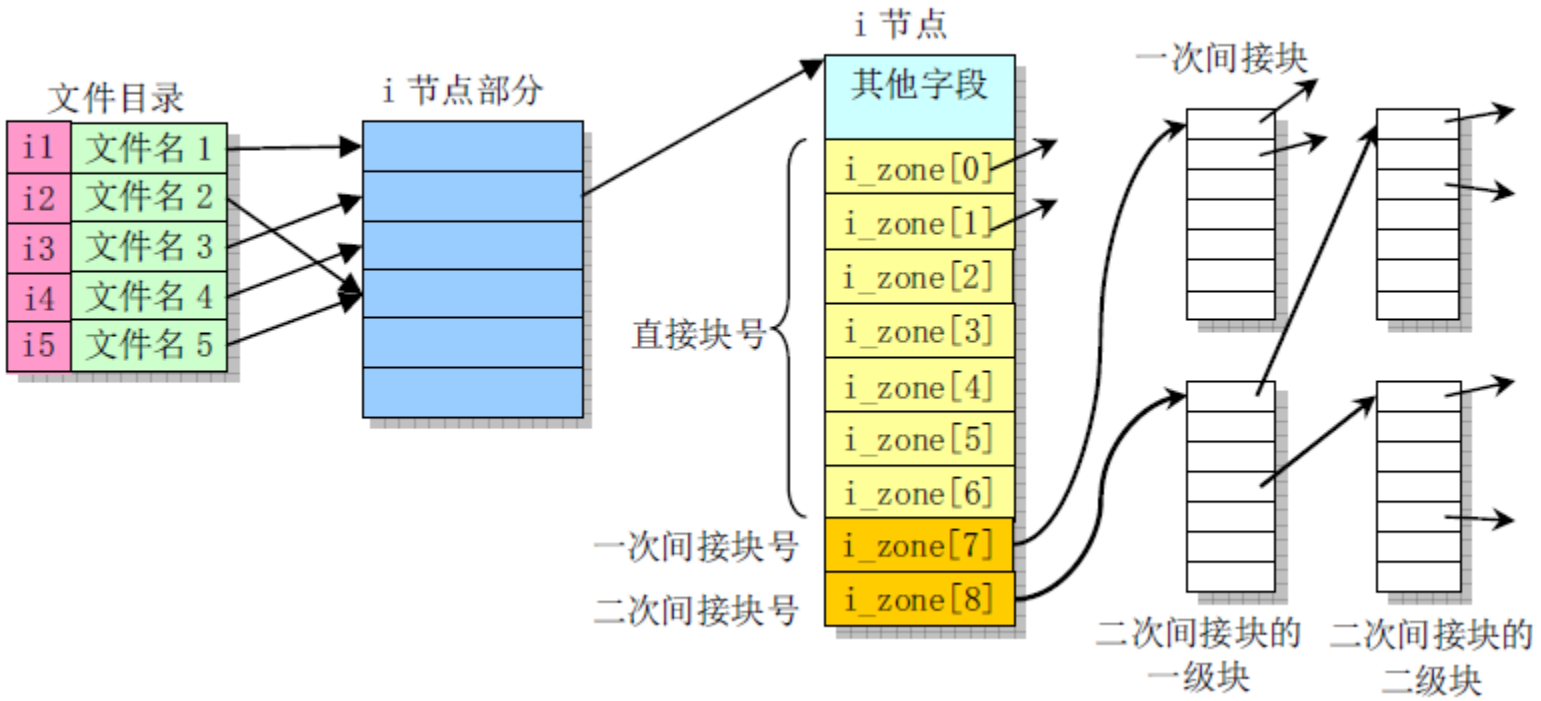
```
[/usr/root]# touch qwertyuiopasdf
[/usr/root]# ls
a.out           hello.c         qwertyuiopasdf
[/usr/root]# touch qwertyuiopasdfp
[/usr/root]# ls
a.out           hello.c         qwertyuiopasdf
```

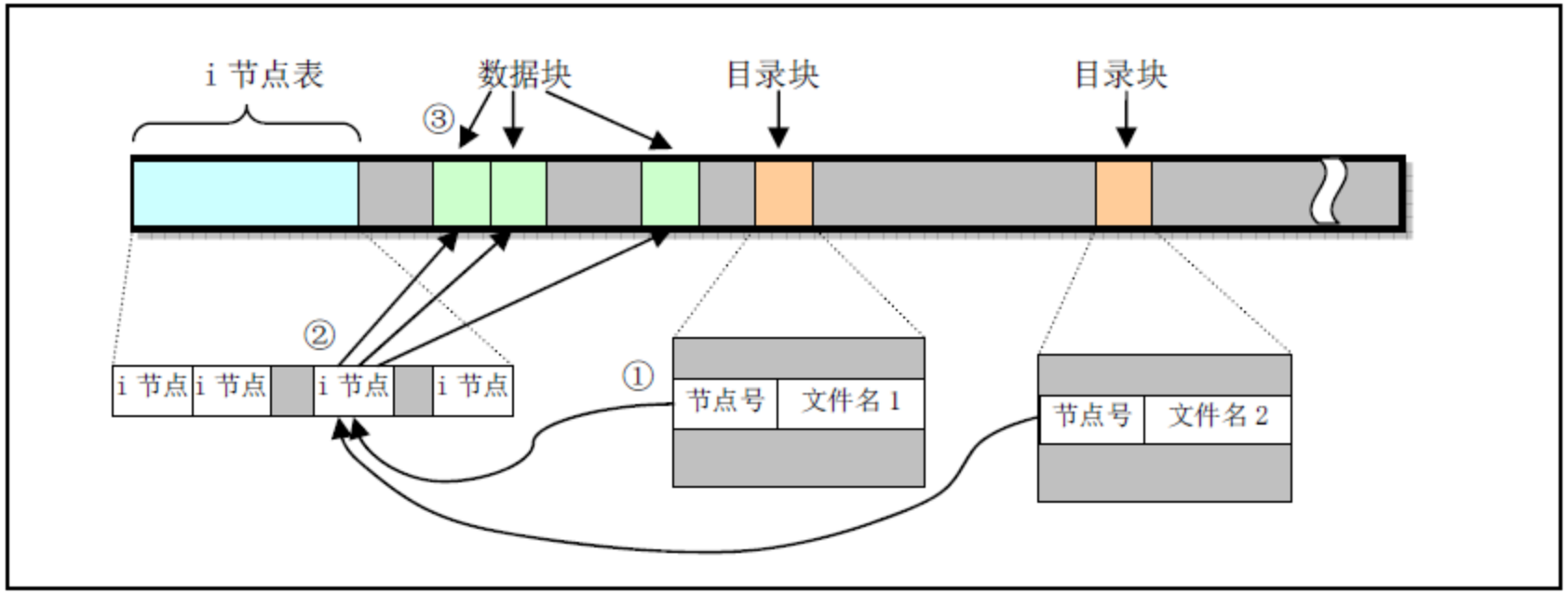
- 每个目录项只包括一个长度为14B的文件名字字符串和该文件名所对应的2B的i节点号。一个逻辑磁盘块可以存放1024/16=64个目录项。

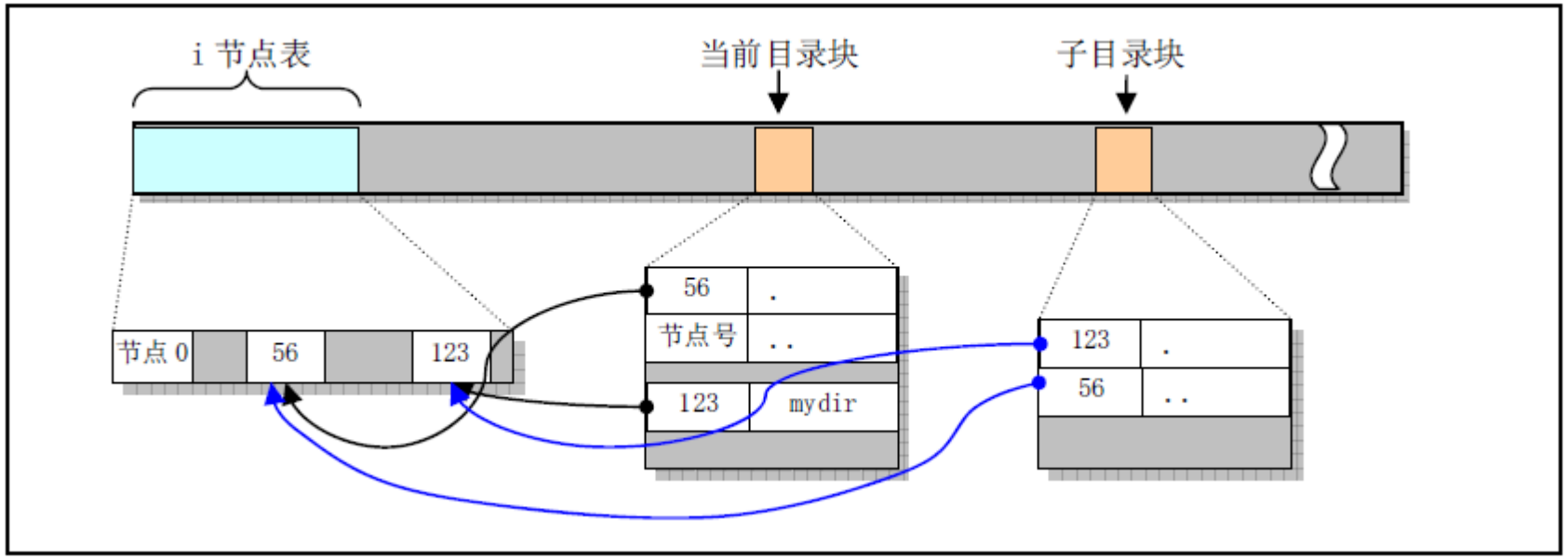
有关文件的其它信息则被保存在该i节点号指定的i节点结构中

- 该结构主要包括文件访问属性、宿主、长度、访问保存时间以及所在磁盘块等信息

- 打开一个文件时，文件系统根据给定的文件名找到其i节点号，通过其对应i节点信息找到文件所在的磁盘块位置。



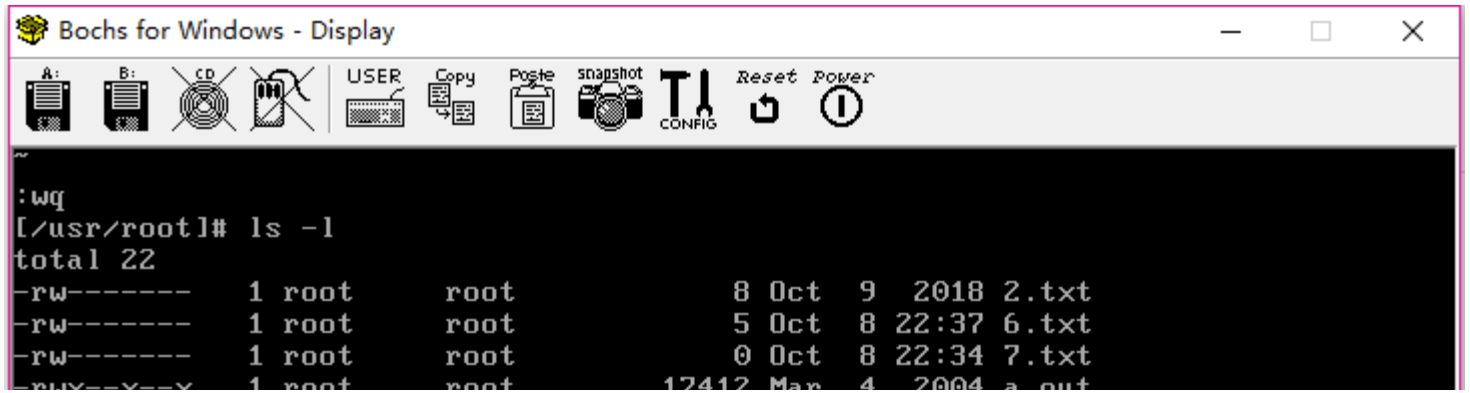

i 节点表
数据块
目录块
目录块
③
②
①
i 节点
i 节点
i 节点
i 节点
节点号
文件名 1
节点号
文件名 2


i 节点表
当前目录块
子目录块
节点 0
56
123
56
.
节点号
..
123
mydir
123
.
56
..

硬链接（Hard Link）

```
[/usr/root]# ls -l
total 21
-rw-------   1 root     root
-rwx--x--x   1 root     root
drwx--x--x   3 root     root
-rw-------   1 root     4096
-rw-------   1 root     root
[/usr/root]# cp -l 1.txt 2.txt
[/usr/root]# ls -l
total 22
-rw-------   2 root     root
-rw-------   2 root     root
-rwx--x--x   1 root     root
drwx--x--x   3 root     root
-rw-------   1 root     4096
-rw-------   1 root     root
```

```
[/usr/root]# more 2.txt
kldjalk
[/usr/root]# ls -li
total 22
    75 -rw-------   2 root     root            8 Oct  9  2018 1.txt
    75 -rw-------   2 root     root            8 Oct  9  2018 2.txt
    66 -rwx--x--x   1 root     root        17412 Mar  4  2004 a.out
    53 drwx--x--x   3 root     root           64 Oct  8  2018 demo
    69 -rw-------   1 root     4096           74 Mar  4  2004 hello.c
    49 -rw-------   1 root     root            0 Oct  8 11:43 qwertyuiopasdf
[/usr/root]# more 1.txt
kldjalk
[/usr/root]# rm -f 1.txt
[/usr/root]# more 2.txt
kldjalk
[/usr/root]# ls -l
total 21
-rw-------   1 root     root            8 Oct  9  2018 2.txt
-rwx--x--x   1 root     root        17412 Mar  4  2004 a.out
drwx--x--x   3 root     root           64 Oct  8  2018 demo
-rw-------   1 root     4096           74 Mar  4  2004 hello.c
-rw-------   1 root     root            0 Oct  8 11:43 qwertyuiopasdf
```




```
[/]# hexdump .
0000000 0001 002e 0000 0000 0000 0000 0000 0000      // .
0000010 0001 2e2e 0000 0000 0000 0000 0000 0000      // ..
0000020 0002 6962 006e 0000 0000 0000 0000 0000      // bin
0000030 0003 6564 0076 0000 0000 0000 0000 0000      // dev
0000040 0004 7465 0063 0000 0000 0000 0000 0000      // etc
0000050 0005 7375 0072 0000 0000 0000 0000 0000      // usr
0000060 0115 6e6d 0074 0000 0000 0000 0000 0000      // mnt
0000070 0036 6d74 0070 0000 0000 0000 0000 0000      // tmp
0000080 0000 6962 2e6e 656e 0077 0000 0000 0000      // 空闲，未使用。
0000090 0052 6d69 6761 0065 0000 0000 0000 0000      // image
00000a0 007b 6176 0072 0000 0000 0000 0000 0000      // var
00000b0
```

- 高速缓冲区


Thank you !