xv6©-rev10
(Copyright Frans Kaashoek, Robert Morris, and Russ Cox.)

Name Layer II

Carmi Merimovich

Tel-Aviv Academic College

January 20, 2017

Routines exposed in the name system are:

1. namei(path): returns inode* from path.
2. create(path,type,major,minor): returns inode* for created path
3. dirlookup(dp, name).

There are several more of course.

- Basically they all handle folders and inode numbers.
- Remember inode is identified by a device number and inode number pair.

directory structure

- An inode of type T_DIR is a directory.
- A directory is a sequence of records of the form:

```
#define ROOTINO 1 // root i number

#define DIRSIZ 14

struct dirent {
  ushort inum;
  char   name[DIRSIZ];
};
```

- The records are in no specific order.
- If inum is zero then the entry is not used.

inum

- inum is inode number.
- The inode number leads to the disk inode from which the memory inode is built.

inode struct

```c
struct inode {
 uint dev;  // Device number
 uint inum; // Inode number
 int  ref;  // Reference count
 struct sleeplock lock;
 int  valid;

 short type; // copy of disk inode
 short major;
 short minor;
 short nlink;
 uint  size;
 uint  addrs[NDIRECT+1];
};
```

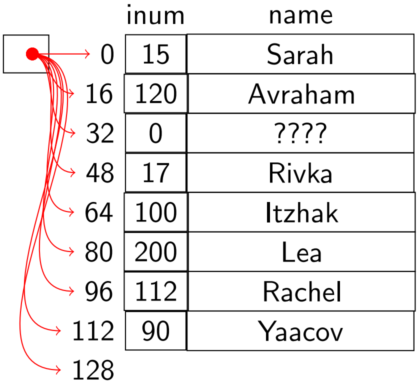
First aim: namei().

We need dirlookup(), skipelem(), and iget() beforehand.

“opening” a file

- “opening a file” means getting a (memory) inode from a path.
- Let us work step by step.
- Assume dp points to a T_DIR inode, and we need to open the file Moshe in it.

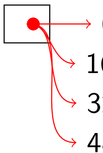
"open" (dp, " Moshe")



	inum	name
0	15	Sarah
16	120	Avraham
32	0	????
48	17	Rivka
64	100	Itzhak
80	200	Lea
96	112	Rachel
112	90	Yaacov
128		

```
read(dp, &de, [illegible], sizeof(de));
```

“open” (dp, ” Rivka”)



	inum	name
0	15	Sarah
16	120	Avraham
32	0	????
48	17	Rivka
64	100	Itzhak
80	200	Lea
96	112	Rachel
112	90	Yaacov
128		

```
read(dp, &de, sizeof(de));
```

dirlookup(dp, name, off) logic

- The routines looks for the entry name in the directory dp.
- If the name is found it returns inode * of the inum in the record is returned.
- If off is not null then *off receive the position of the record in the directory.

dirlookup(dp,name,off)

```
struct inode *dirlookup(struct inode *dp, char *name,
                                        uint *poff) {
 uint inum;
 struct dirent de;
 if (dp->type != T_DIR) panic("dirlookup␣not␣DIR");
 for (off = 0; off < dp->size; off += sizeof(de)) {
  if (readi(dp,(char*)&de,off,sizeof(de))!=sizeof(de
   panic("dirlink␣read");
  if (de.inum == 0) continue;
  if (namecmp(name, de.name) == 0) {
   if (poff) *poff = off;
   inum = de.inum;
   return iget(dp->dev, inum);
  }
 }
 return 0;
}
```

“open”(dp,”ancestors/Rivka”)

```
dp1 = “open”(dp,”ancestors”);
ip = “open”(dp1,”Rivka”);
```

“open”(”ancestors/Rivka”);

$\texttt{dp} = \text{“}\texttt{open}\text{”}(\texttt{myproc()}{-}{>}\texttt{cwd},\text{”}\texttt{ancestors}\text{”});$

$\texttt{ip} = \text{“}\texttt{open}\text{”}(\texttt{dp},\text{”}\texttt{Rivka}\text{”});$

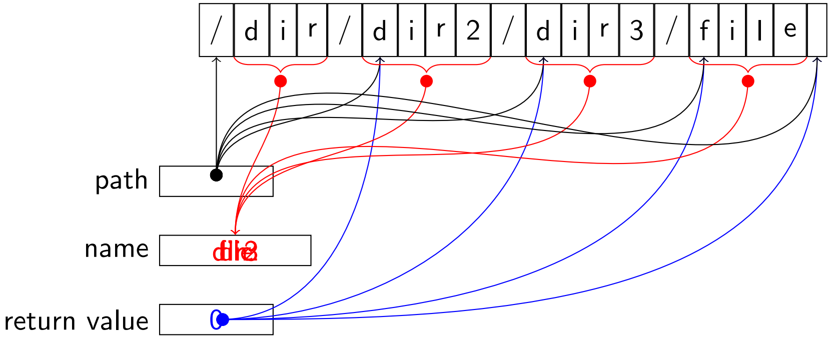
“open”(”/ancestors/Rivka”);

```
dp = “open”(iget(ROOTDEV, ROOTINO), ”ancestors”);
ip = “open”(dp, ”Rivka”);
```

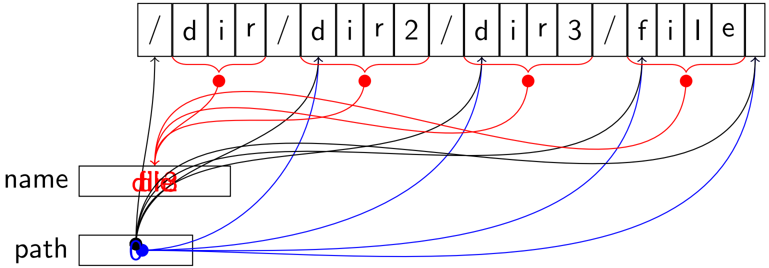
“open”(”dir1/ ⋯ /dirn/file”)

```
$\texttt{dp} = \texttt{myproc()}- > \texttt{cwd};$
$\texttt{for}\ (\texttt{i} = 1; \texttt{i} \le \texttt{n}; \texttt{i}++)$
    $\texttt{dp} = “\texttt{open}''(\texttt{dp}, ''\texttt{diri}'');$
$\texttt{ip} = “\texttt{open}''(\texttt{dp}, ''\texttt{file}'');$
```

skipelem(path,name) examples



path=skipelem(path,name) examples



skipelem()

```
static char *skipelem(char *path, char *name) {
 int len;

 while(*path == '/') path++;
 if (*path == 0) return 0;
 char *s = path;
 while(*path != '/' && *path != 0) path++;
 len = path - s;
 if (len >= DIRSIZ)
  memmove(name, s, DIRSIZ);
 else {
  memmove(name, s, len);
  name[len] = 0;
 }
 while(*path == '/') path++;
 return path;
}
```

namei() is just namex()

```
struct inode *namei(char *path) {
 char name[DIRSIZ];
 return namex(path, 0, name);
}
```

namex(path,nameiparent,name)

- If nameiparent is zero, the inode corresponding to the file named path is returned.
- if nameparent is nonzero, the inode corresponding to the path with the last element is opened. The last string is returned in name.
- The inode is not locked.

namex()

```
static struct inode *namex(char *path,
                           int nameiparent,
                           char *name) {
  struct inode *ip, *next;

  if (*path == '/')
    ip = iget(ROOTDEV, ROOTINO);
  else
    ip = idup(myproc()->cwd);
```

namex() (2)

```
while ((path = skipelem(path, name)) != 0) {
  ilock(ip);
  if (ip->type != T_DIR) {
    iunlockput(ip);
    return 0;
  }
  if (nameiparent && *path == '\0') {
    iunlock(ip);
    return ip;
  }
  if ((next = dirlookup(ip, name, 0)) == 0) {
    iunlockput(ip);
    return 0;
  }
  iunlockput(ip);
  ip = next;
}
```

namex() (3)

```
  if (nameiparent) {
    iput(ip);
    return 0;
  }
  return ip;
}
```

We aim now at create()

Beforehand we need nameiparent(), dirlink().

nameiparent() is namex()

```
struct inode *nameiparent(char *path, char *name) {
 return namex(path, 1, name);
}
```

dirlink(dp, name, inum)

- A record pointing name to inum is added to directory dp.

dirlink()

```
int dirlink(struct inode *dp, char *name, uint inum) {
 int off;
 struct dirent de;
 struct inode *ip;
 if ((ip = dirlookup(dp, name, 0)) != 0) {
  iput(ip);
  return -1;
 }
 for (off = 0; off < dp->size; off += sizeof(de)) {
  if (readi(dp, (char*)&de, off, sizeof(de)) != sizeof(de
    panic("dirlink␣read");
  if (de.inum == 0) break;
 }

 strncpy(de.name, name, DIRSIZ);
 de.inum = inum;
 if (writei(dp, (char*)&de, off, sizeof(de)) != sizeof(de
  panic("dirlink");
 return 0;
```

More facts about directories

The first two records of a directory are always as follows:

- .: self pointing.
- ..: pointing to the parent directory.
- The root (/) is an exception. The second record is also self pointing.

create (1)

```
6357 static struct inode *create(char *path, short type,
                              short major, short minor) {
  uint off;
  struct inode *ip, *dp;
  char name[DIRSIZ];

  if((dp = nameiparent(path, name)) == 0)
   return 0;
  ilock(dp);

  if((ip = dirlookup(dp, name, &off)) != 0){
   iunlockput(dp);
   ilock(ip);
   if(type == T_FILE && ip->type == T_FILE)
    return ip;
   iunlockput(ip);
   return 0;
  }
```

create (2)

```
6376    if ((ip = ialloc(dp->dev, type)) == 0)
         panic("create:␣ialloc");

        ilock(ip);
        ip->major = major;
        ip->minor = minor;
        ip->nlink = 1;
        iupdate(ip);

        if (type == T_DIR) { // Create . and .. entries.
         dp->nlink++; // for ".."
         iupdate(dp);
        if (dirlink(ip, ".", ip->inum) < 0 ||
                 dirlink(ip, "..", dp->inum) < 0)
         panic("create␣dots");
        }
```

create (3)

```
  if (dirlink(dp, name, ip->inum) < 0)
    panic("create:␣dirlink");

  iunlockput(dp);

  return ip;
}
```