

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:

```
4054 #define ROOTINO 1 // root i number
```

```
4113 #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

```
4162 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
<input type="checkbox"/>	0	15	Sarah
	16	120	Avraham
	32	0	????
	48	17	Rivka
	64	100	Itzhak
	80	200	Lea
	96	112	Rachel
	112	90	Yaacov
	128		

“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		

`readi(dp,&de,0,sizeof(de));`

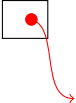
“open” (dp, "Moshe")



	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		

`readi(dp,&de,16,sizeof(de));`


“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		

`readi(dp,&de,32,sizeof(de));`


“open” (dp, “Moshe”)



	innum	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		

`readi(dp,&de,48,sizeof(de));`


“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		

`readi(dp,&de,64,sizeof(de));`


“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		

`readi(dp,&de,80,sizeof(de));`


“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		

`readi(dp,&de,96,sizeof(de));`


“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		


`readi(dp,&de,112,sizeof(de));`

“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		

“open” (dp, “Moshe”)



A diagram consisting of a small square box with a red dot inside. A red arrow originates from the red dot and points downwards and to the right, ending at the first row of the table (index 0).

	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		


return (0)

“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		

“open”(dp, " Rivka")

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

readi(dp,&de,0,sizeof(de));

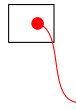
“open”(dp, " Rivka")



	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		

`readi(dp,&de,16,sizeof(de));`

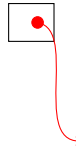
“open”(dp, " Rivka")



	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		

readi(dp,&de,32,sizeof(de));

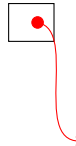
“open”(dp, " Rivka")



	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		

readi(dp,&de,48,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		

return(iget(dp →dev,17));

dirlookup(dp, name, off) logic

- The routine 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)

```
5611 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”)

“open” (dp, “ancestors/Rivka”)

dp1 = “open” (dp, “ancestors”);

“open”(dp, “ancestors/Rivka”)

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

“open” (“ancestors/Rivka”);

“open”(“ancestors/Rivka”);

dp = “open”(myproc()—>cwd,“ancestors”);

“open”(“ancestors/Rivka”);

```
dp = “open”(myproc()->cwd, “ancestors”);  
ip = “open”(dp, “Rivka”);
```


“open” (“ /ancestors/Rivka”);

“open”(“/ancestors/Rivka”);

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

“open”(“/ancestors/Rivka”);

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

`"open" ("dir1/.../dirn/file")`

```
dp = myproc() -> cwd;  
for (i = 1; i ≤ n; i++)  
    dp = "open"(dp, "diri");  
ip = "open"(dp, "file");
```

skipelem(path,name) examples

/	d	i	r	/	d	i	r	2	/	d	i	r	3	/	f	i	l	e	
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	--

path

name

return value

skipelem(path,name) examples

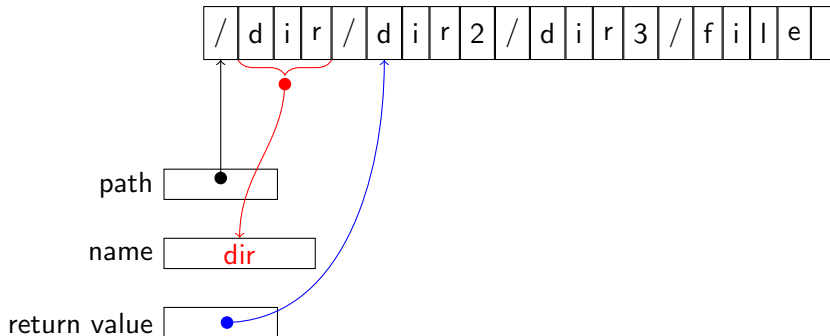
/	d	i	r	/	d	i	r	2	/	d	i	r	3	/	f	i	l	e
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

path

name

return value

skipelem(path,name) examples



skipelem(path,name) examples

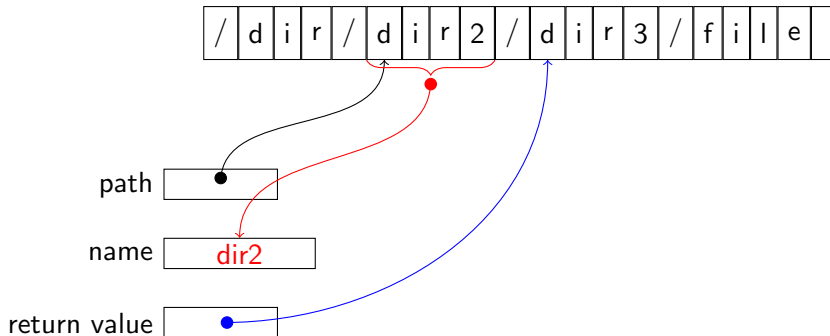
/	d	i	r	/	d	i	r	2	/	d	i	r	3	/	f	i	l	e	
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	--

path

name

return value

skipelem(path,name) examples



skipelem(path,name) examples

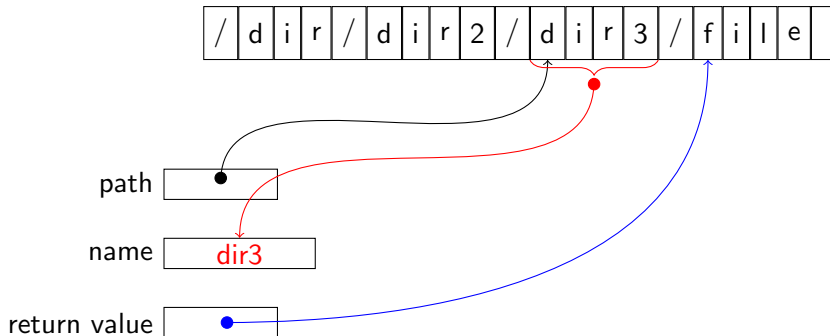
/	d	i	r	/	d	i	r	2	/	d	i	r	3	/	f	i	l	e
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

path

name

return value

skipelem(path,name) examples



skipelem(path,name) examples

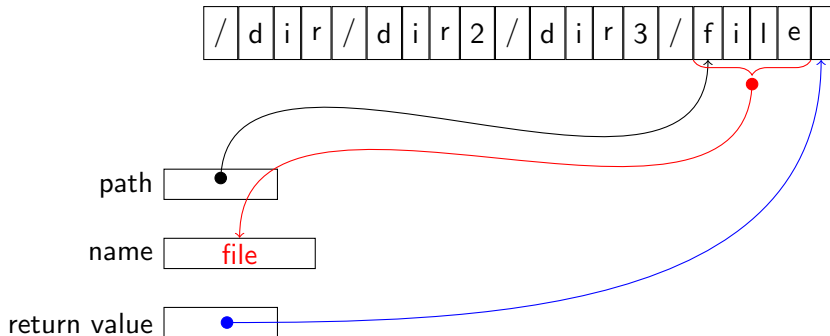
/	d	i	r	/	d	i	r	2	/	d	i	r	3	/	f	i	l	e
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

path

name

return value

skipelem(path,name) examples



skipelem(path,name) examples

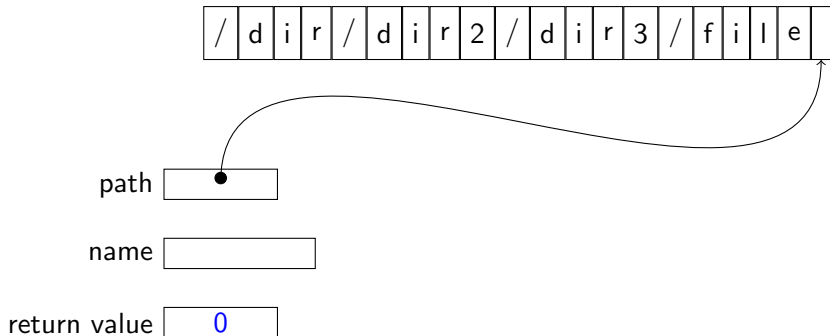
/	d	i	r	/	d	i	r	2	/	d	i	r	3	/	f	i	l	e	
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	--

path

name

return value

skipelem(path,name) examples



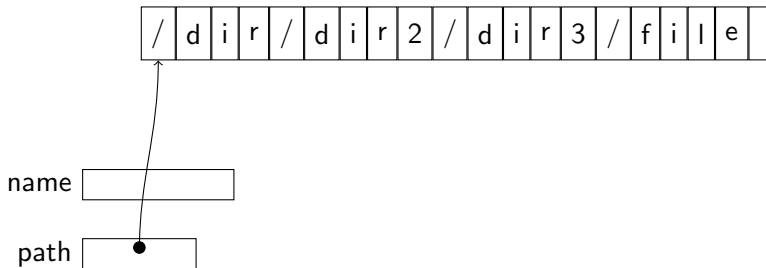
path=skipelem(path,name) examples

/	d	i	r	/	d	i	r	2	/	d	i	r	3	/	f	i	l	e	
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	--

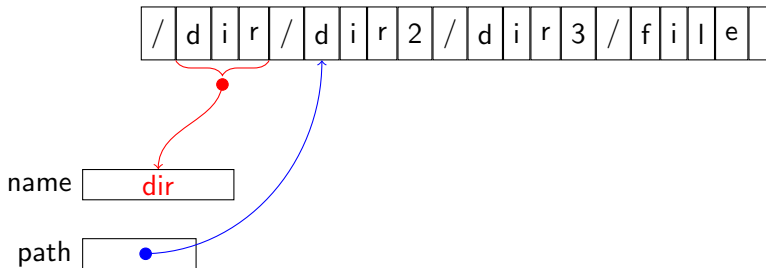
name

path

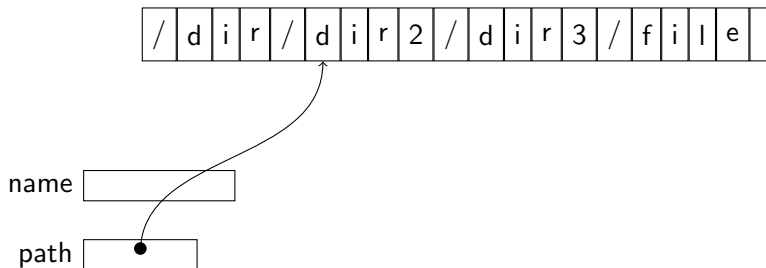
path=skipelem(path,name) examples



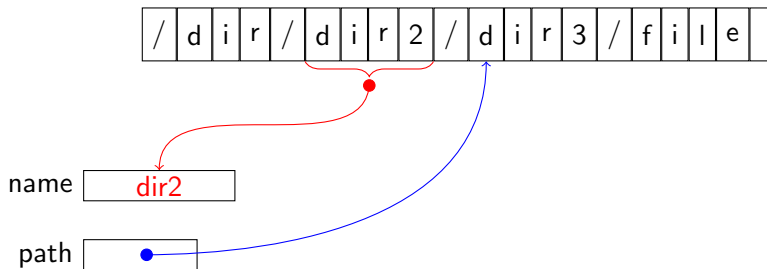
path=skipelem(path,name) examples



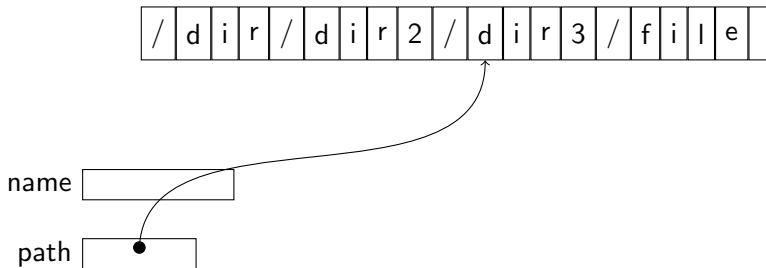
path=skipelem(path,name) examples



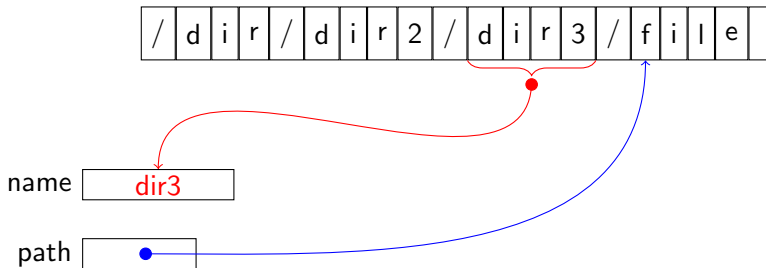
path=skipelem(path,name) examples



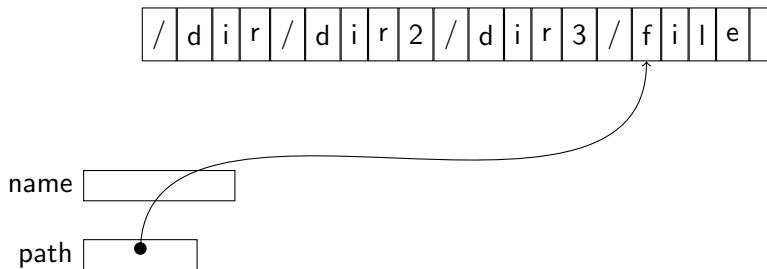
path=skipelem(path,name) examples



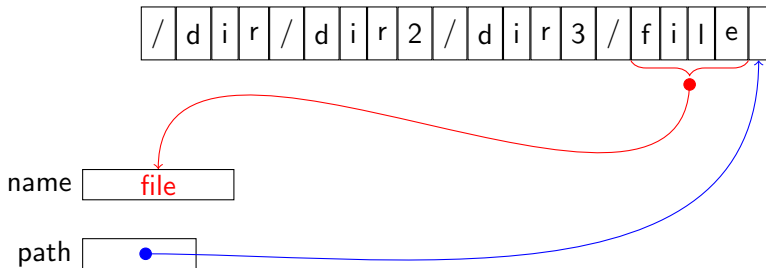
path=skipelem(path,name) examples



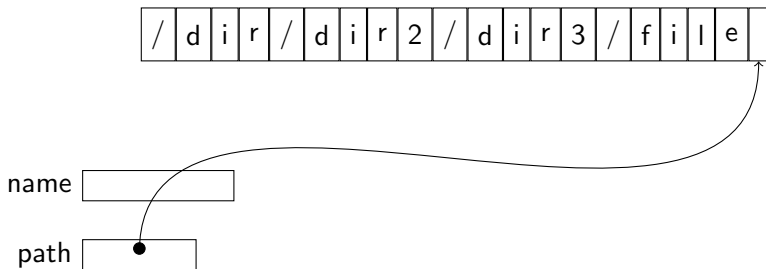
path=skipelem(path,name) examples



path=skipelem(path,name) examples



path=skipelem(path,name) examples



path=skipelem(path,name) examples

/	d	i	r	/	d	i	r	2	/	d	i	r	3	/	f	i	l	e	
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	--

name

path

skipelem()

```
5715 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()

```
5790 struct inode *namei(char *path) {  
    char name[DIRSIZ];  
    return namex(path, 0, name);  
}
```

namex(path,nameparent,name)

- If nameparent 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()

```
5755 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)

3764

```
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)

```
5782  if (nameiparent) {  
        iput(ip);  
        return 0;  
    }  
    return ip;  
}
```


We aim now at `create()`
Beforehand we need `nameiparent()`, `dirlink()`.

nameiparent() is namex()

```
5801 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()

```
5652 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)

```
6393  if (dirlink(dp, name, ip->inum) < 0)
        panic("create: _dirlink");

        iunlockput(dp);

    return ip;
}
```