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

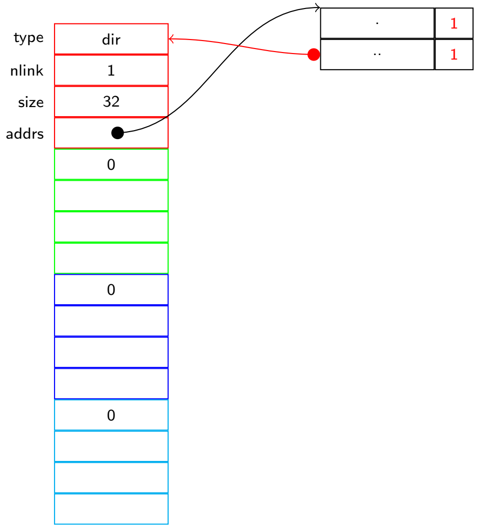
Name Layer I

Carmi Merimovich

Tel-Aviv Academic College

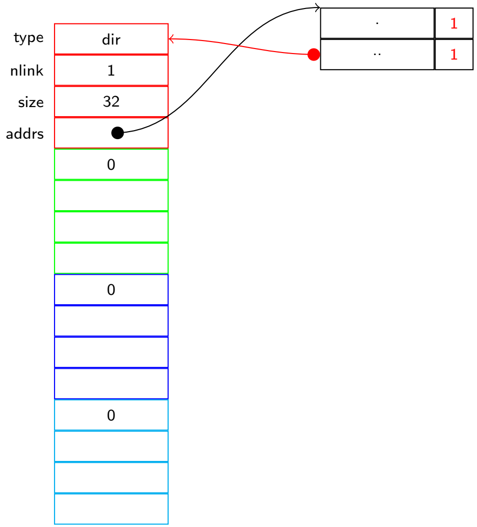
January 20, 2017

dinodes and directories on disk



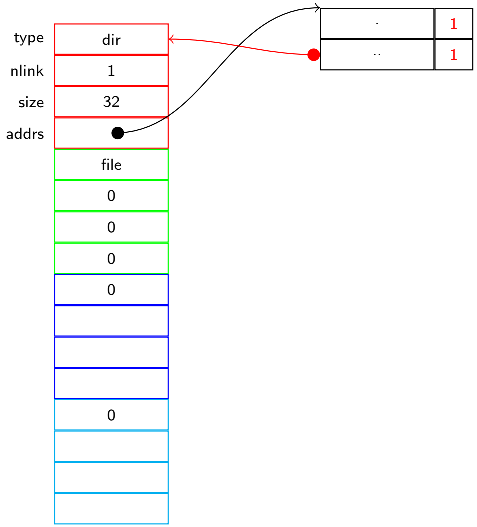
Empty file system

dinodes and directories on disk



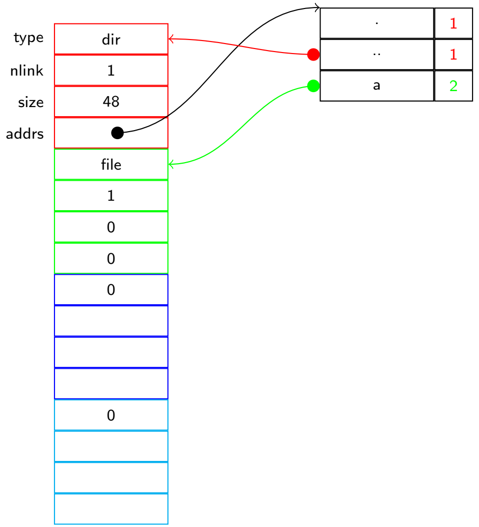
open("/a",O_CREAT)

dinodes and directories on disk



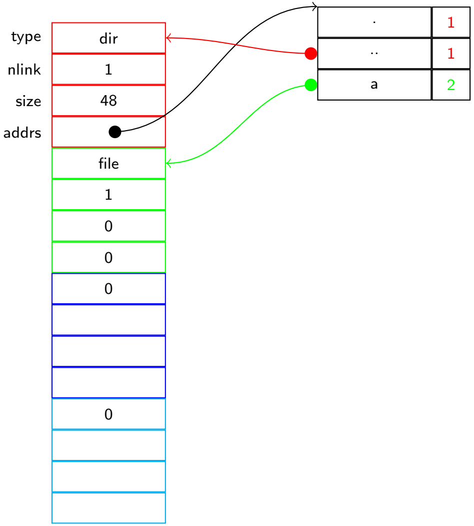
open("/a",O_CREAT)

dinodes and directories on disk



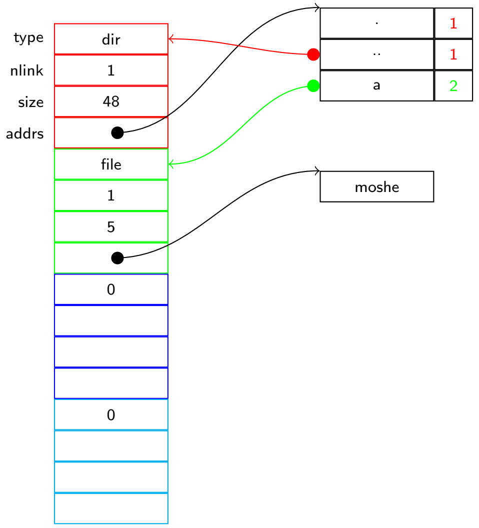
open("/a",O_CREAT)

dinodes and directories on disk



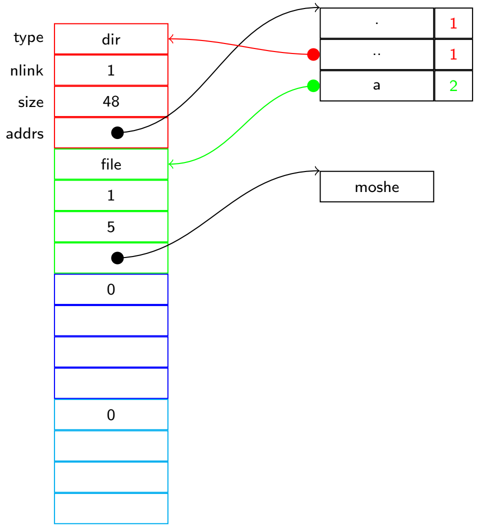
write(,"moshe",5)

dinodes and directories on disk



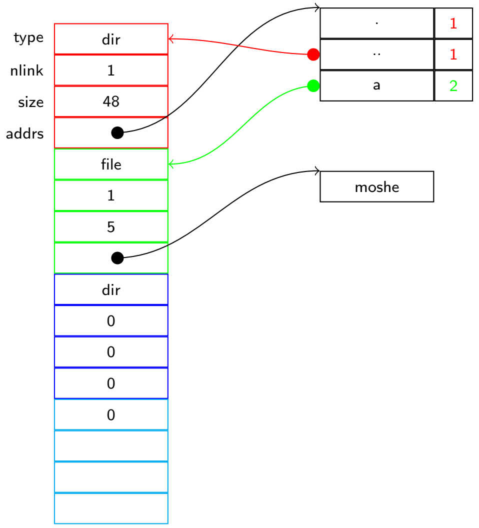
write(,"moshe",5)

dinodes and directories on disk



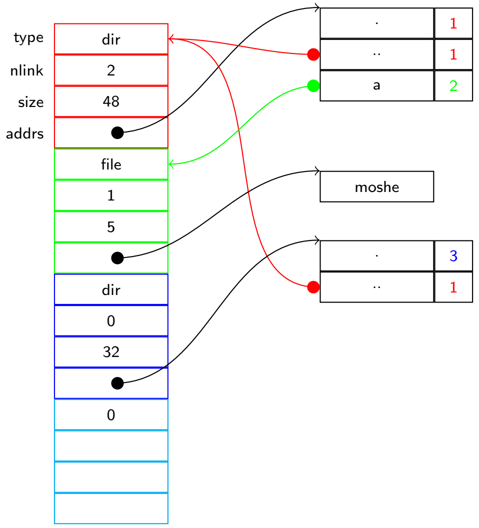
mkdir("/b")

dinodes and directories on disk



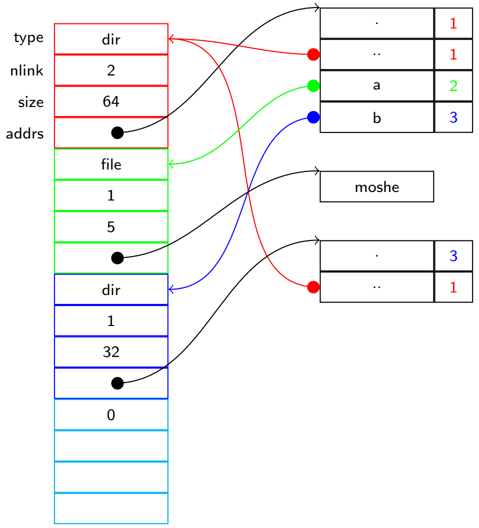
mkdir("/b")

dinodes and directories on disk



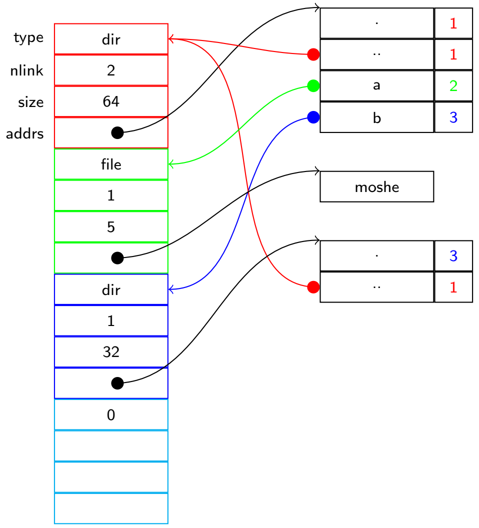
mkdir("/b")

dinodes and directories on disk



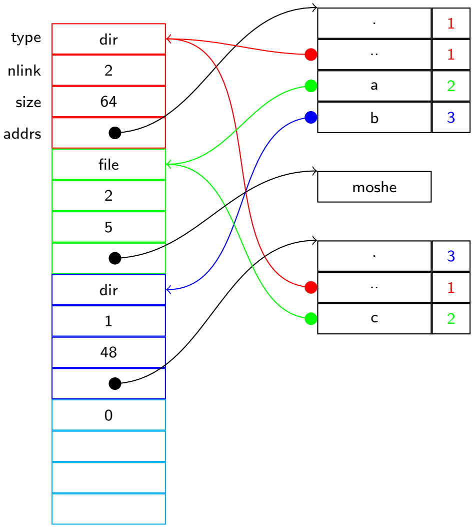
mkdir("/b")

dinodes and directories on disk



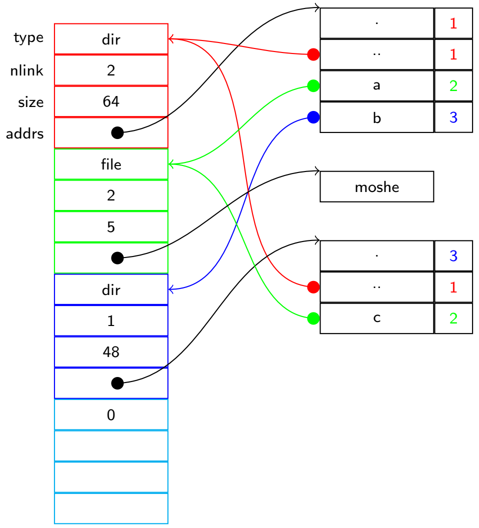
link("/a","/b/c")

dinodes and directories on disk



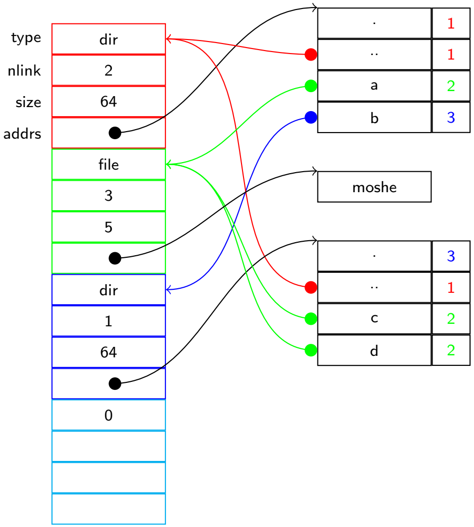
link("/a","/b/c")

dinodes and directories on disk



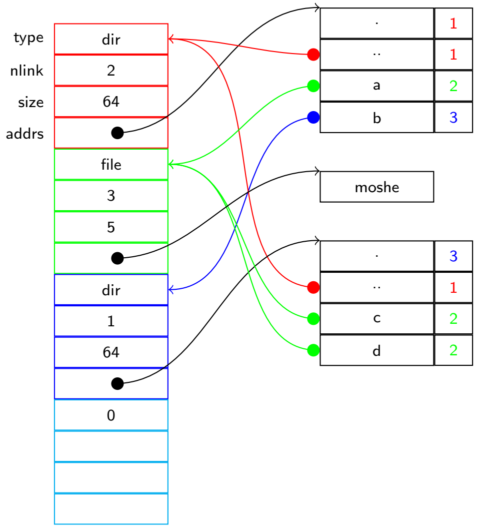
link("/b/c","/b/d")

dinodes and directories on disk



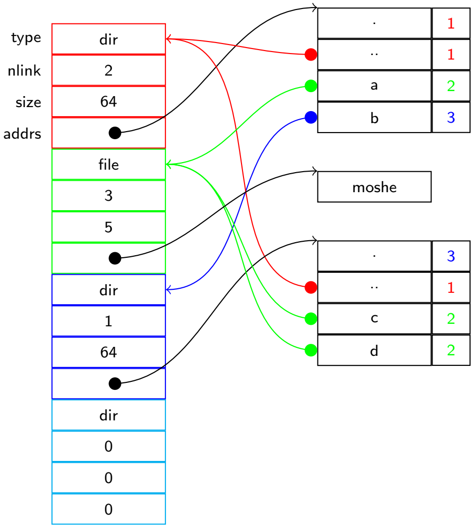
link("/b/c","/b/d")

dinodes and directories on disk



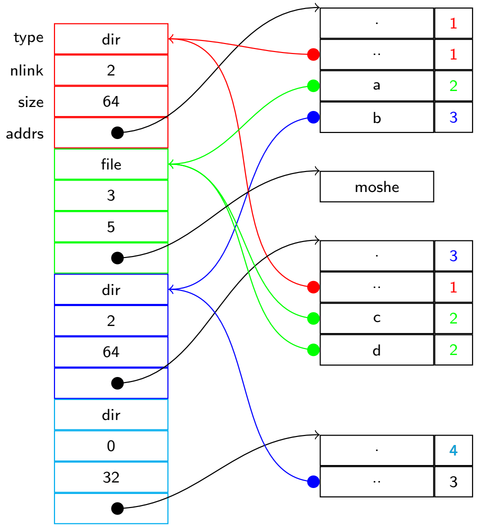
mkdir("/b/e")

dinodes and directories on disk



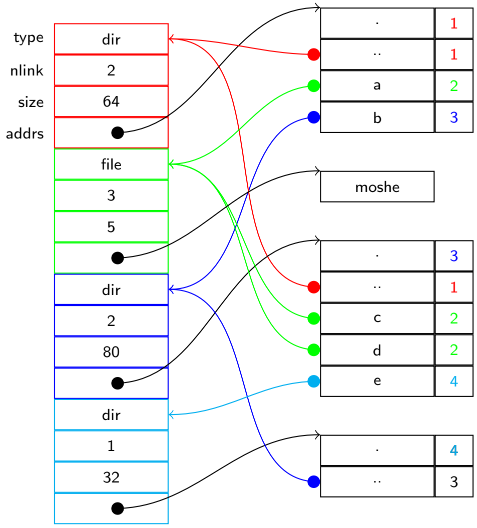
mkdir("/b/e")

dinodes and directories on disk



mkdir("/b/e")

dinodes and directories on disk



mkdir("/b/e")

System calls using the name layer

- sys_open.
- sys_mkdir.
- sys_link.
- sys_unlink.

open(char *path, int mode)

- omode flags:

```
#define O_RDONLY 0x000
#define O_WRONLY 0x001
#define O_RDWR 0x002
#define O_CREATE 0x200
```

- If `O_CREATE` set then we need to create file.
- If `O_CREATE` is clear we need to open existing file.
- Folders can be opened only for readonly access.

sys_open logic

- If creating is attempted then:
  - Delegate to the create function in the inode layer.
  - (Thus, either creating a file, or opening an existing file).
  - Note create an ilock'ed inode pointer.
- If opening is attempted then:
  - Delegate to the namei function in the inode layer.
  - (Thus opening an existing file).
  - If the file is directory make sure readonly access is in effect.
- Build a FD_INODE file structure pointing to the found/created inode.
- Hide the file pointer in the ofile vector.

sys_open, invoking lower layer

```
6401 int sys_open(void) {
      char *path;
      int omode;
      struct inode *ip;
      if (argstr(0, &path) < 0 || argint(1, &omode) < 0) retur
       begin_op();
      if (omode & O_CREATE) {
        if ((ip = create(path, T_FILE, 0, 0)) == 0) {
           end_op(); return -1;
        }
       } else {
        if ((ip = namei(path)) == 0) { end_op(); return -1;}
        ilock(ip);
        if (ip->type == T_DIR && omode != O_RDONLY) {
         iunlockput(ip);
         end_op();
         return -1;
        }
     }
```

sys_open, housekeeping

```
if ((f=filealloc()) == 0 || (fd=fdalloc(f)) < 0) {
 if (f)
  fileclose(f);
 iunlockput(ip);
 end_op()
 return -1;
}
iunlock(ip);
end_op();

f->type = FD_INODE;
f->ip = ip;
f->off = 0;
f->readable = !(omode & O_WRONLY);
f->writable = (omode & O_WRONLY)||(omode & O_RDWR);
return fd;
}
```

sys_mkdir

```
int sys_mkdir(void) {
 char *path;
 struct inode *ip;

 begin_op();
 if (argstr(0, &path) < 0 ||
     (ip = create(path, T_DIR, 0, 0)) == 0) {
  end_op();
  return    1 ;
 }
 iunlockput(ip);
 end_op();
 return 0;
}
```

sys_link (1)

```
6202  sys_link(void) {
       char name[DIRSIZ], *new, *old;
       struct inode *dp, *ip;

       if (argstr(0, &old) < 0 || argstr(1, &new) < 0)
        return -1;
       begin_op()
       if ((ip = namei(old)) == 0) {
        end_op();
        return -1;
       }
       ilock(ip);
       if (ip->type == T_DIR) {
        iunlockput(ip);
        end_op();
        return -1;
       }
```

sys_link (2)

```
ip->nlink++;
iupdate(ip);
iunlock(ip);

if((dp = nameiparent(new, name)) == 0)
 goto bad;
ilock(dp);
if(dp->dev != ip->dev || dirlink(dp, name, ip->inur
 iunlockput(dp);
 goto bad;
}
iunlockput(dp);
iput(ip);

end_op();
return 0;
```

sys_link (3)

```
6241 bad:
   ilock(ip);
   ip->nlink    ;
   iupdate(ip);
   iunlockput(ip);
   end_op();
   return -1;
 }
```

sys_unlink

```
int sys_unlink(void) {
 struct inode *ip, *dp;
 struct dirent de;
 char name[DIRSIZ], *path;
 uint off;

 if(argstr(0, &path) < 0)
  return -1;
 begin_op();
 if((dp = nameiparent(path, name)) == 0) {
  end_op();
  return -1;
 }
 ilock(dp);
 if(namecmp(name, ".")==0 || namecmp(name, "..")==0)
  goto bad;
```

sys_unlink (2)

```
if((ip = dirlookup(dp, name, &off)) == 0)
 goto bad;
ilock(ip);

if(ip->nlink < 1)
 panic("unlink:␣nlink␣<␣1");
if(ip->type == T_DIR && !isdirempty(ip)){
 iunlockput(ip);
 goto bad;
}

memset(&de, 0, sizeof(de));
if(writei(dp, (char*)&de, off, sizeof(de)) != sizeo
 panic("unlink:␣writei");
```

sys_unlink (3)

```
if(ip->type == T_DIR) {
  dp->nlink--;
  iupdate(dp);
}
iunlockput(dp);

ip->nlink--;
iupdate(ip);
iunlockput(ip);

end_op();

return 0;
```

sys_unlink (4)

```
bad:
  iunlockput(dp);
  end_op();
  return    1 ;
}
```