VFS Mount

- All mounted filesystems are included in a list, referenced by *vfsmntlist*
- Each element of the list is a *struct vfsmount*

```
mnt_dev:            device number
mnt_devname:        device name
mnt_dirname:        mount point
mnt_flags:          mount flags
mnt_sb:             pointer to superblock
mnt_dquot:          disk quota mount options
mnt_dnext:          pointer to next element
```

VFS Mount

- *add_vfsmnt(), remove_vfsmnt()*
  - add or remove an element from the list
- *lookup_vfsmnt*()
  - search a specific mounted filesystem and return a pointer to *vfsmnt* structure

VFS Mount

- *sys_mount()*
  - *mount()* service routine
  - parameters:
    - pathname of the device file
      - NULL when not required
    - pathename of the mount point
    - filesystem type
    - mount flags
    - pointer to filesystem dependent data structure
      - NULL when not required

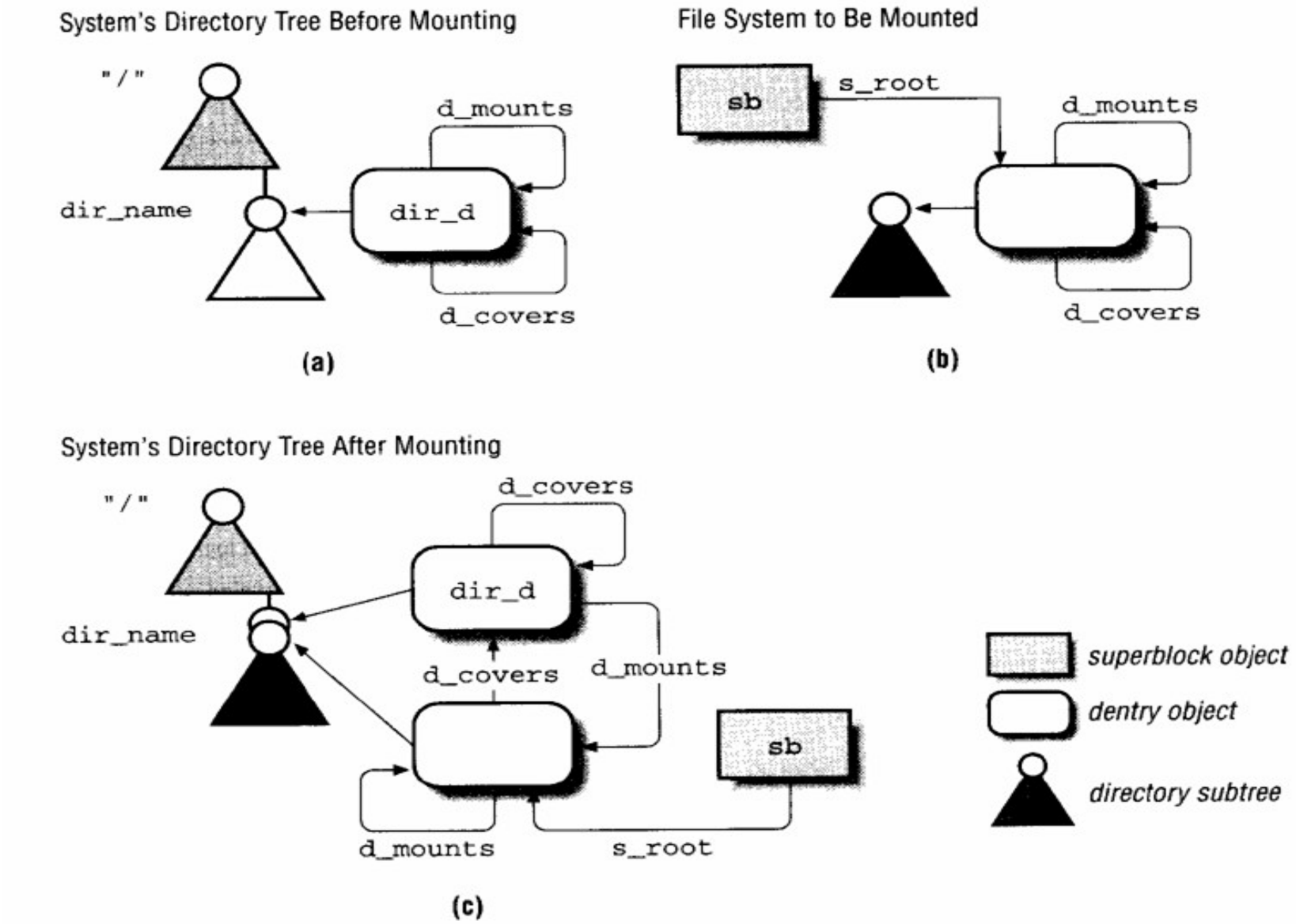
VFS Mount

- *sys_mount():*
  - check user permissions
  - call do_remount() if appropriate and return
  - get filesystem type
  - check parameters
  - call do_mount() to perform the actual operation

VFS Mount

- *do_mount():*
  - get directory entry (lock semaphore) and check it
  - read superblock
  - add filesystem to list of mounted filesystems
  - update directory pointers
  - unlock semaphore

VFS Mount


System's Directory Tree Before Mounting
"/"
d_mounts
dir_name
dir_d
d_covers
(a)
File System to Be Mounted
sb
s_root
d_mounts
d_covers
(b)
System's Directory Tree After Mounting
"/"
d_covers
dir_d
dir_name
d_covers
d_mounts
sb
d_mounts
s_root
(c)
superblock object
dentry object
directory subtree

VFS UnMount

- *sys_unmount()*
  - *umount()* service routine
  - check access permission
  - determine device (+ check)
  - release dentry
  - lock semaphore and call *umount_dev()->do_umount()*
  - unlock semaphore

VFS UnMount

- *do_umount()*
  - shrink dentry cache
  - fsync device
  - check if device can be umounted (not /)
  - check access count & state, release structures
  - update (if needed) and release superblock
  - remove *struct vfsmnt* from list

Lookup Path Name

- Derive inode from corresponding pathname
- Break path into a sequence of filenames
- All components (except the last) must identify directories
- If the first character is / then pathname is absolute
  - use *current->fs->root*
- else
  - *current->fs->pwd*
- For each component of the pathname (left to right) derive the corresponding inode, lookup and continue

Lookup Path Name

- A component might be a symbolic link which expands in an arbitrary pathname
- Circular references may arise
- A filename might be the mountpoint of another filesystem
- Access rights must be checked for each access

Lookup Path Name

- *lookup_dentry()*
  - called by *namei() and lnamei()*
  - recursive
  - Name: file pathname
  - Base: pointer to a *dentry* object
  - flags
    - LOOKUP_FOLLOW, LOOKUP_DIRECTORY, LOOKUP_SLASHOK

Lookup Path Name

- *lookup_dentry()*
  - Determine from the first character of *name* (and *base*) and where the search must start from
  - Gets the inode of the initial directory
  - Iteratively repeat on each part of the filename:

Lookup Path Name

- Check process permissions
- Computes hash value
- Updates name
- Calls *reserved_lookup()*

Lookup Path Name

- *reserved_lookup()*
  - Initalizes the *dentry* local variable
  - Locks the *i_sem* semaphore
  - Re-execute *cached_lookup* and, if needed, calls the *lookup* method filling a new *dentry* object
  - Release the *i_sem* semaphore
  - Invokes *follow_mount()* and *do_follow_link()*
  - If the inode references a directory either starts a new cycle or returns *base* (depending on value of *flags)*
  - Returns *base* or an error code