Advanced Programming in the UNIX Environment

Week 04, Segment 2: Links

Department of Computer Science
Stevens Institute of Technology

Jan Schaumann
jschauma@stevens.edu
https://stevens.netmeister.org/631/

link(2)

```
#include <fcntl.h>
#include <unistd.h>

int link(const char *path1, const char *path2);
int linkat(int fd1, const char *path1, int fd2, const char* path2, int flags);
```

Returns: 0 on success, -1 on error

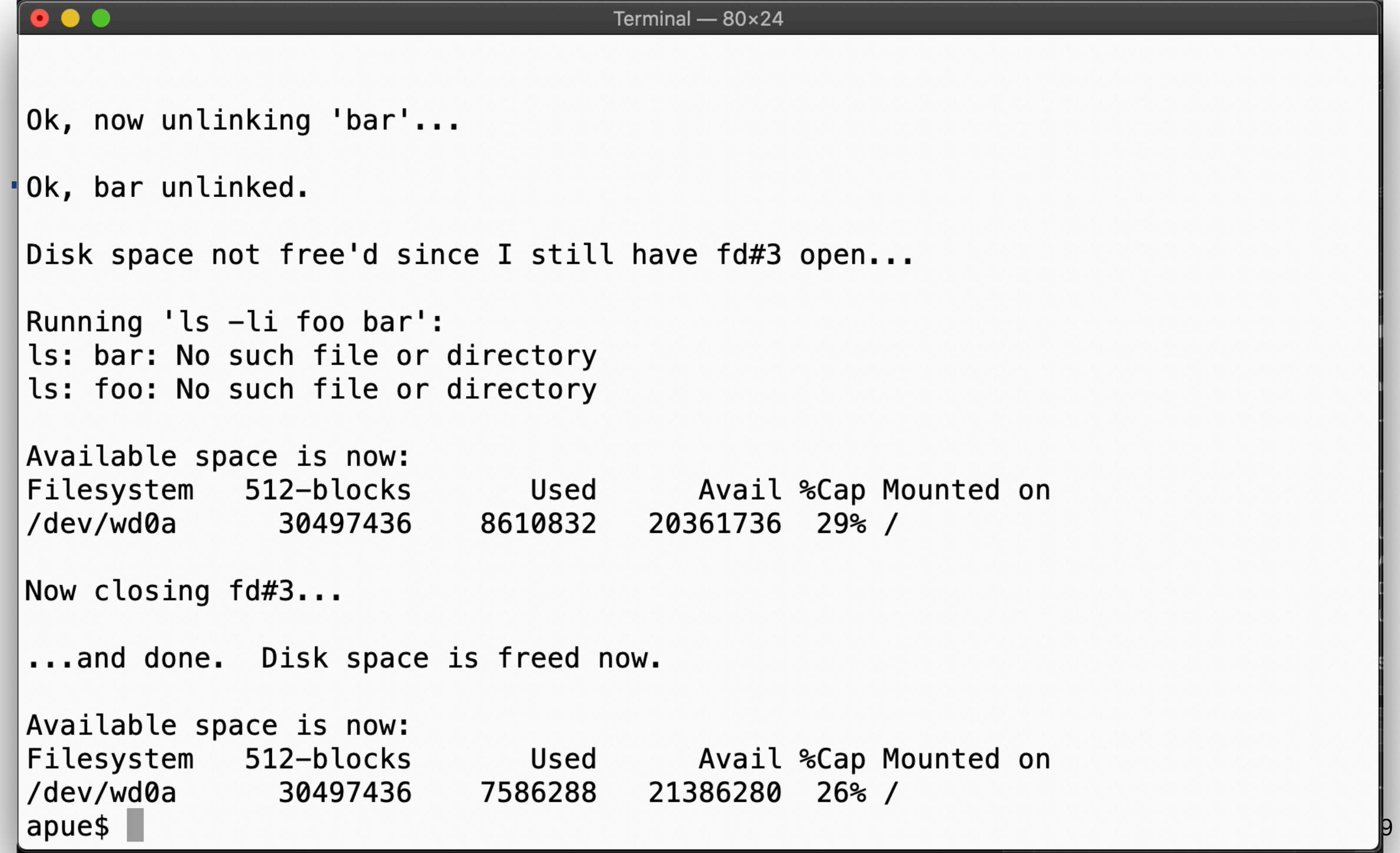
- creates a (hard) link to an existing file, incrementing `st_nlink` in the process
- POSIX allows hard links across filesystems, most implementations don't
- only euid 0 can create links to directories (loops in filesystem are bad)

unlink(2)

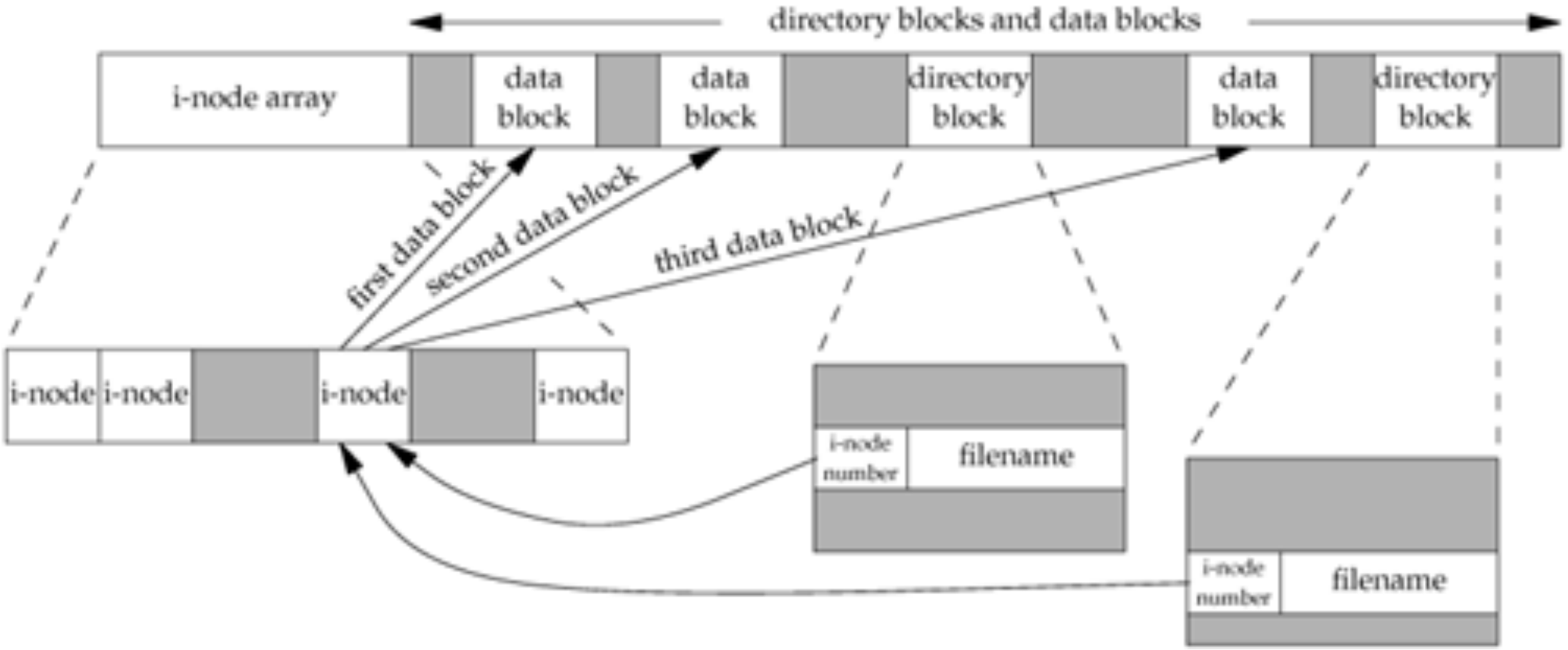
```
#include <fcntl.h>
#include <unistd.h>

int unlink(const char *path);
int unlinkat(int fd, const char *path, int flags);
```

Returns: 0 on success, -1 on error

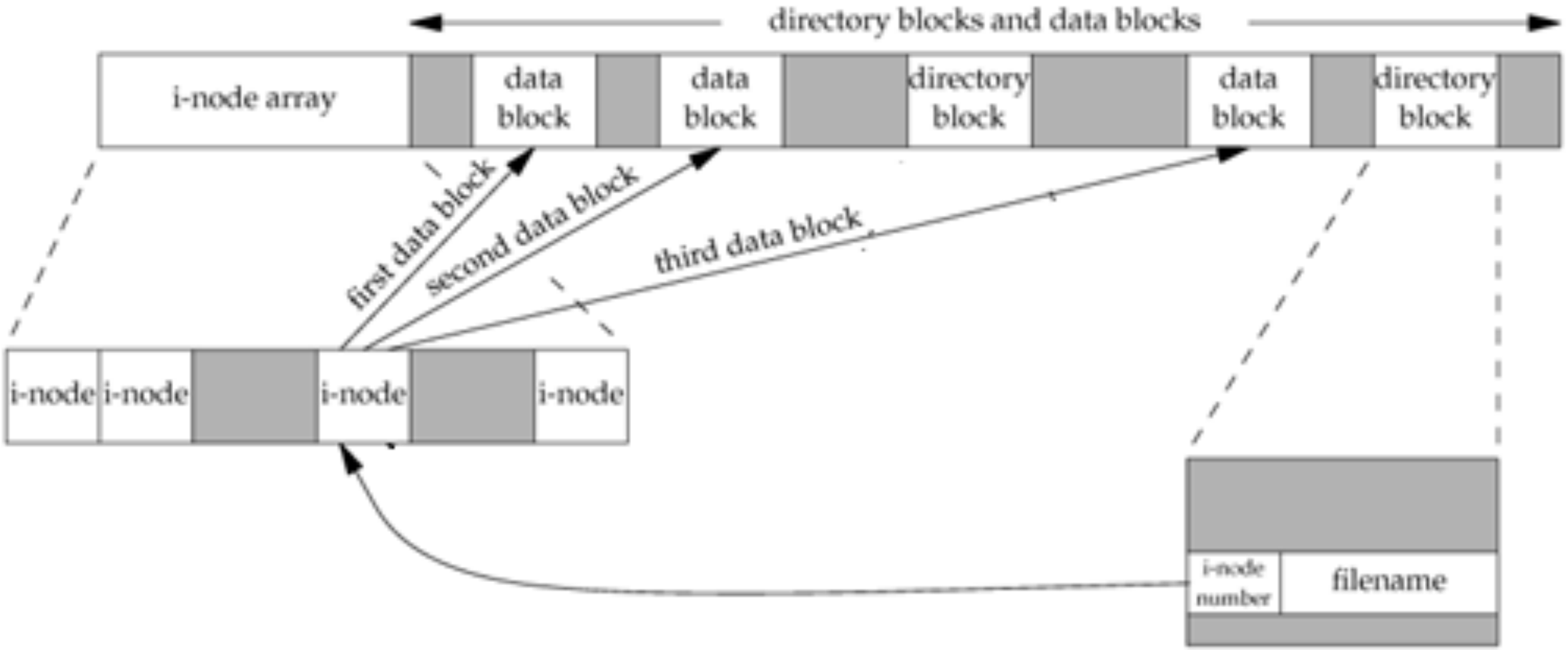
- removes the given directory entry, decrementing `st_nlink` in the process
- if `st_nlink` == 0, free data blocks associated with file (...unless processes have the file open)


Terminal — 80×24
Ok, now unlinking 'bar'...
Ok, bar unlinked.
Disk space not free'd since I still have fd#3 open...
Running 'ls -li foo bar':
ls: bar: No such file or directory
ls: foo: No such file or directory
Available space is now:
Filesystem 512-blocks Used Avail %Cap Mounted on
/dev/wd0a 30497436 8610832 20361736 29% /
Now closing fd#3...
...and done. Disk space is freed now.
Available space is now:
Filesystem 512-blocks Used Avail %Cap Mounted on
/dev/wd0a 30497436 7586288 21386280 26% /
apue$

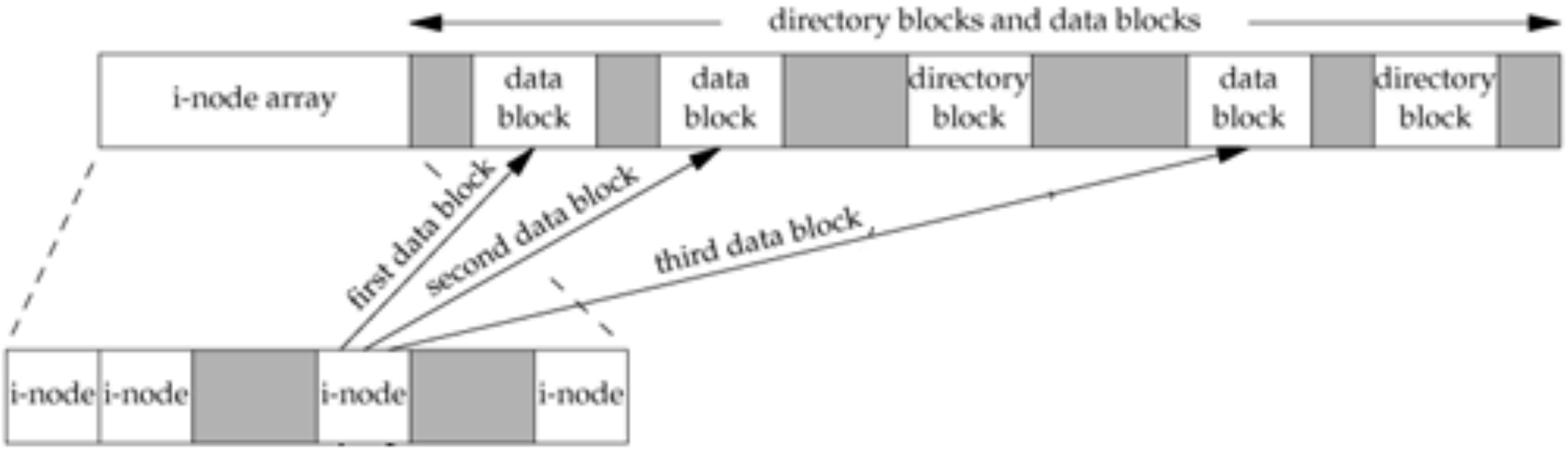
unlink(2)


directory blocks and data blocks
i-node array
data block
data block
directory block
data block
directory block
first data block
second data block
third data block
i-node
i-node
i-node
i-node
i-node number
filename
i-node number
filename

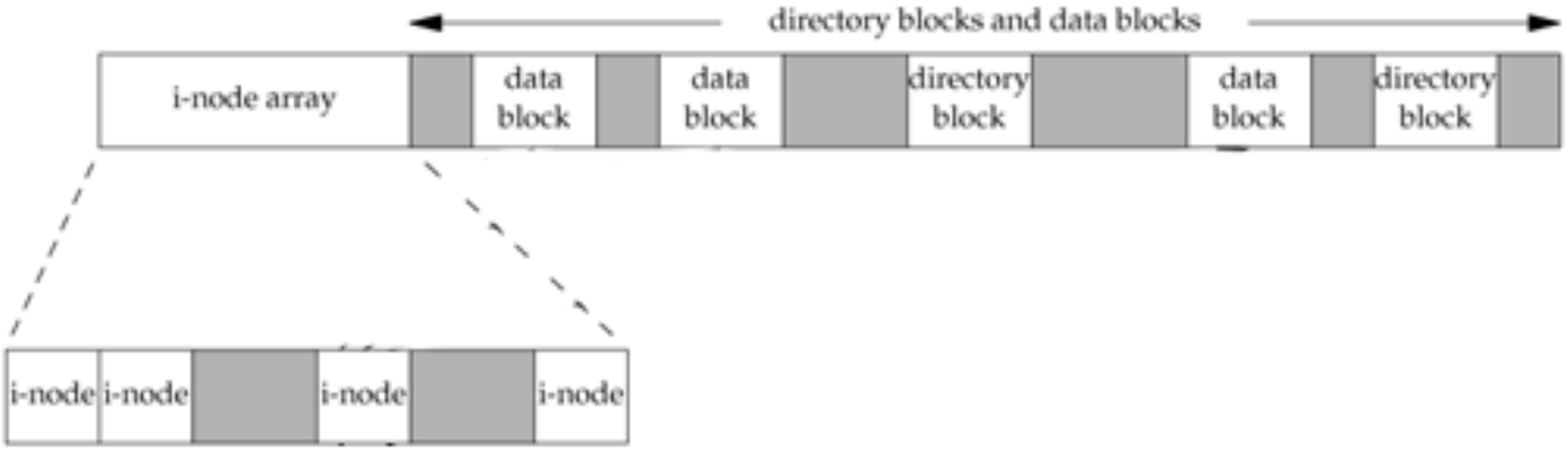
unlink(2)


directory blocks and data blocks
i-node array
data block
data block
directory block
data block
directory block
first data block
second data block
third data block
i-node
i-node
i-node
i-node
i-node number
filename

unlink(2)


directory blocks and data blocks
i-node array
data block
data block
directory block
data block
directory block
first data block
second data block
third data block
i-node
i-node
i-node
i-node

unlink(2)


directory blocks and data blocks
i-node array
data block
data block
directory block
data block
directory block
i-node
i-node
i-node
i-node

rename(2)

```
#include <stdio.h>
#include <unistd.h>

int rename(const char *from, const char *to);
int renameat(int fromfd, const char *from, int tofd, const char* to, int flags);
```

Returns: 0 on success, -1 on error

If from refers to a file:

- if to exists and it is not a directory, it's removed and from is renamed to
- if to exists and it is a directory, an error results
- must have w+x perms for the directories containing from/to

rename(2)

```
#include <stdio.h>
#include <unistd.h>

int rename(const char *from, const char *to);
int renameat(int fromfd, const char *from, int tofd, const char* to, int flags);
```

Returns: 0 on success, -1 on error

If from refers to a directory:

- if to exists and is an empty directory (contains only . and ..), it is removed; from is renamed to
- if to exists and is a file, an error results
- must have w+x perms for the directories containing from/to
- if from is a prefix of to an error results

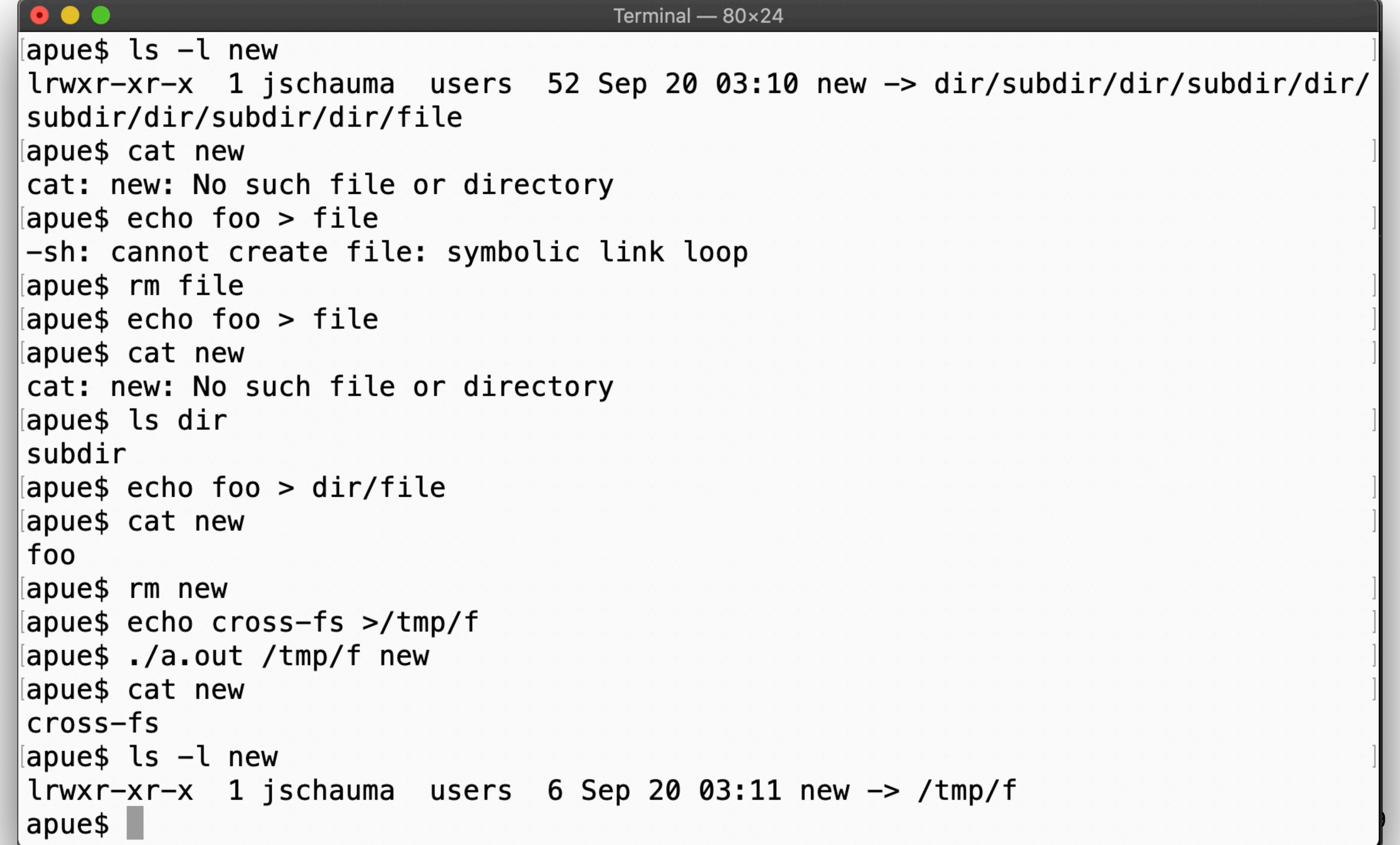
```
apue$ ./a.out dir dir2
apue$ ls -l dir2
total 0
-rw-r--r--  1 jschauma  users  0 Sep 20 02:51 file
apue$ mkdir dir
apue$ ./a.out dir dir2
Unable to rename 'dir' to 'dir2': Directory not empty
apue$ rm dir2/file
apue$ ./a.out dir dir2
apue$ touch file
apue$ ./a.out dir2 file
Unable to rename 'dir2' to 'file': Not a directory
apue$ mkdir -p dir/subdir/subsubdir
apue$ ./a.out dir/subdir/subsubdir dir/subdir
Unable to rename 'dir/subdir/subsubdir' to 'dir/subdir': Directory not empty
apue$ ./a.out dir dir/subdir2
Unable to rename 'dir' to 'dir/subdir2': Invalid argument
apue$ ./a.out dir /tmp/dir
Unable to rename 'dir' to '/tmp/dir': Cross-device link
apue$ ./a.out /dir
Usage: ./a.out from to
apue$ ./a.out dir /dir
Unable to rename 'dir' to '/dir': Permission denied
apue$
```

symlink(2)

```
#include <stdio.h>
#include <unistd.h>

int symlink(const char *name1, const char *name2);
int symlinkat(const char *name1, int fd, const char *name2);
```

Returns: 0 on success, -1 on error

- a symbolic link is a special file that contains as its data the pathname of another file
- symlinks can point to any other type of files, including directories
- recall syscalls dereferencing symlinks versus those operating on the link


Terminal — 80×24
apue$ ls -l new
lrwxr-xr-x 1 jschauma users 52 Sep 20 03:10 new -> dir/subdir/dir/subdir/dir/subdir/dir/subdir/dir/file
apue$ cat new
cat: new: No such file or directory
apue$ echo foo > file
-sh: cannot create file: symbolic link loop
apue$ rm file
apue$ echo foo > file
apue$ cat new
cat: new: No such file or directory
apue$ ls dir
subdir
apue$ echo foo > dir/file
apue$ cat new
foo
apue$ rm new
apue$ echo cross-fs >/tmp/f
apue$ ./a.out /tmp/f new
apue$ cat new
cross-fs
apue$ ls -l new
lrwxr-xr-x 1 jschauma users 6 Sep 20 03:11 new -> /tmp/f
apue$

readlink(2)

```
#include <unistd.h>

ssize_t readlink(const char *path, char *buf, size_t bufsiz);
ssize_t readlinkat(int fd, const char *path, char *buf, size_t bufsiz);
```

Returns: number of bytes placed into buf on success, -1 on error

- determine the target of a symbolic link
- buf is **not** NULL terminated

Links

You now can implement `ln(1)`, `mv(1)`, and `rm(1)`.

The link count (`st_nlink`) keeps track of how many names for a file exist; if this count is 0 and no process has a file handle open for this file, the data blocks may be released.

Renaming a file on the same filesystem is trivial, but renaming across filesystems and between files and directories requires a little bit more work.

Symbolic links can link to any file regardless of type, existence, or filesystem / device location.

Coming up: even more details about creating, filling, and removing directories.