Advanced Programming in the UNIX Environment

Week 13, Segment 1:
Restricting Processes - POSIX. 1e ACLs

Department of Computer Science
Stevens Institute of Technology

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

Restricting Processes

The nature of UNIX being a multitasking multiuser OS implies the need for:

- user privileges
- file permissions
- process ownership
- management of all finite resources

That is, we have a constant need to *restrict* processes, to *control* process groups, and to *contain* applications.

https://www.netmeister.org/blog/restricting-processes.html

What we know so far…

- Resource Limitations (Lecture 02 / Lecture 06), *e.g.*, use of `getrlimit(2)`/`sysconf(2)` in `openmax.c`
  - per-process or per-user limits
  - system-wide hard-coded limits
  - system tunable configuration options

- UNIX Access Semantics based on File Ownership (Lecture 03)

Filesystem access

Recall from Week 03 how access semantics are applied, in order:

1. If effective-uid == `st_uid`
    1.1. if appropriate user permission bit is set, grant access;
    1.2. else, deny access
2. If effective-gid == `st_gid`
    2.1.if appropriate group permission bit is set, grant access
    2.2. else, deny access
3. If appropriate other permission bit is set, grant access, else deny access

Filesystem access

Limitations of the traditional Unix access semantics:

- a file can only have one group owner
- group membership quickly becomes convoluted
- different (file- and operating-) systems have different limits on the number of groups a user can be a member of
- any modification of group membership requires the sysadmin to make changes (add/remove members, create new groups, ...)

Access Control Lists

POSIX.1e Access Control Lists (ACLs) provide more fine-grained access control:

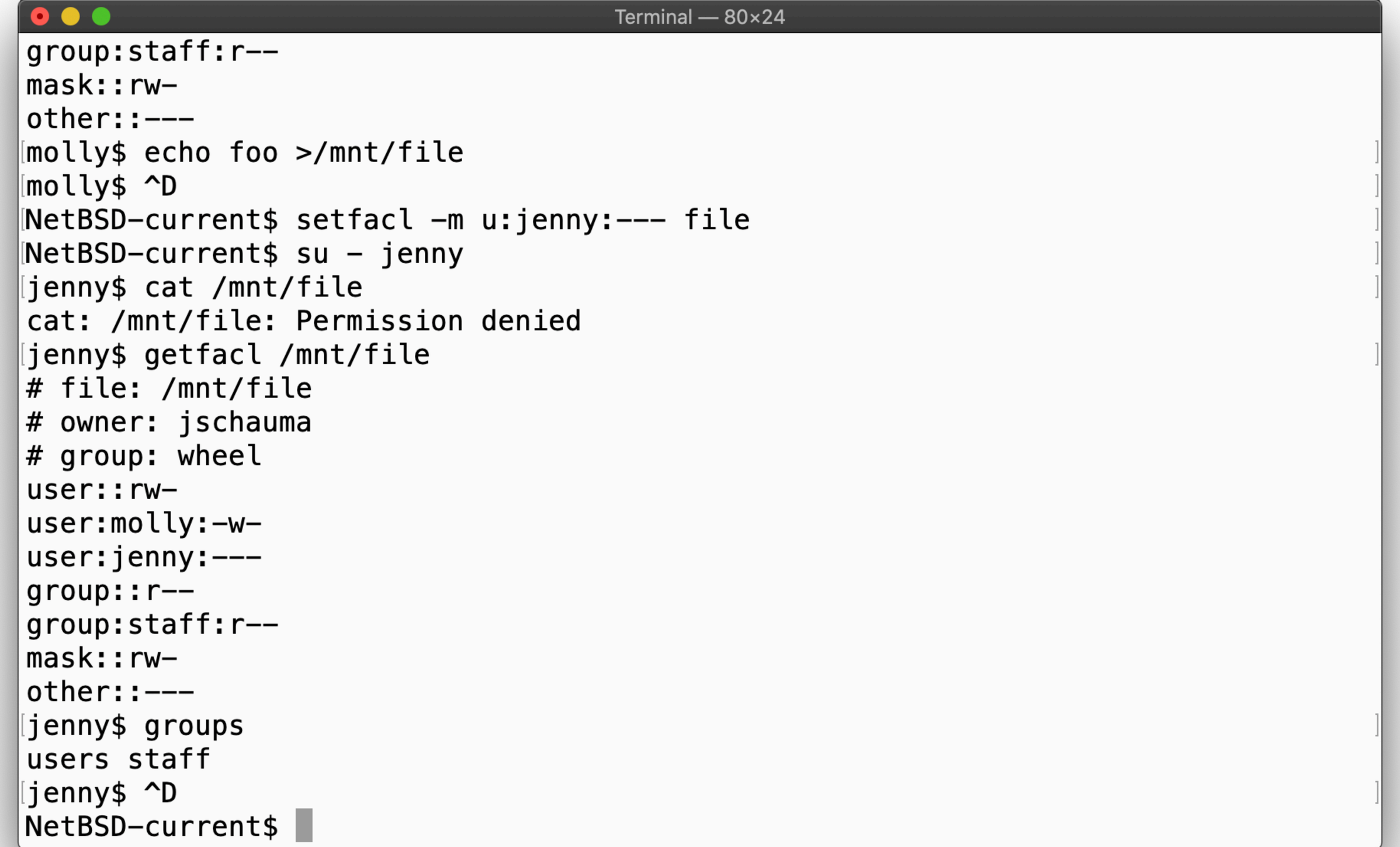
- user can specify individuals or groups with different access
- implemented as 'Extended Attributes' in the filesystem
- `ls(1)` indicates their presence via a '+' at the end of the permissions string
- requires special tools: `getfacl(1)`, `setfacl(1)`

```
GROUP  professor  r--
group  sigsegv    r--
mask              r--
other             ---

linux$ cp -p cat2.c cat3.c
linux$ ls -l *.c
-rw-r-----+ 1 jschauma professor 501 Nov 25 13:57 cat2.c
-rw-r-----+ 1 jschauma professor 501 Nov 25 13:57 cat3.c
-rw-r-----+ 1 jschauma professor 501 Nov 25 13:46 simple-cat.c
linux$ getfacl -t cat3.c
# file: cat3.c
USER   jschauma   rw-
user   eminnix    r--
user   mxiong3    r--
GROUP  professor  r--
group  sigsegv    r--
mask              r--
other             ---

linux$ setfacl -b cat3.c
linux$ ls -l cat3.c
-rw-r----- 1 jschauma professor 501 Nov 25 13:57 cat3.c
linux$
```

```
Terminal — 80×24
 0: group:wheel allow read
macos$ chmod +a "daemon deny read" simple-cat.c
macos$ ls -le simple-cat.c
-rwx------+ 1 jans  staff  806 Oct 29 18:29 simple-cat.c
 0: user:daemon deny read
 1: group:wheel allow read
macos$ man chmod
macos$ chmod -a# 0 simple-cat.c
macos$ ls -le simple-cat.c
-rwx------+ 1 jans  staff  806 Oct 29 18:29 simple-cat.c
 0: group:wheel allow read
macos$ chmod -a # 0 simple-cat.c
usage:  chmod [-fhv] [-R [-H | -L | -P]] [-a | +a | =a  [i][# [ n]]] mode|entry
file ...
        chmod [-fhv] [-R [-H | -L | -P]] [-E | -C | -N | -i | -I] file ...
macos$ ls -ld # this is a comment
drwx------  58 jans  staff  1856 Oct 29 18:29 .
macos$ ls -ld# this is a comment
ls: illegal option -- #
usage: ls [-@ABCFGHLOPRSTUWabcdefghiklmnopqrstuwx1%] [file ...]
macos$ chmod -N simple-cat.c
macos$ ls -l simple-cat.c
-rwx------  1 jans  staff  806 Oct 29 18:29 simple-cat.c
macos$
```


Terminal — 80×24
group:staff:r--
mask::rw-
other::---
molly$ echo foo >/mnt/file
molly$ ^D
NetBSD-current$ setfacl -m u:jenny:--- file
NetBSD-current$ su - jenny
jenny$ cat /mnt/file
cat: /mnt/file: Permission denied
jenny$ getfacl /mnt/file
file: /mnt/file
owner: jschauma
group: wheel
user::rw-
user:molly:-w-
user:jenny:---
group::r--
group:staff:r--
mask::rw-
other::---
jenny$ groups
users staff
jenny$ ^D
NetBSD-current$

POSIX. 1e Access Control Lists

- ACLs are stored as Extended Attributes and thus require support of the file system
- Ordering of ACLs may be relevant — experiment with different users and groups to verify the impact of the order.
- Different OS implement POSIX ACLs differently
  - Linux: `getfacl(1)`/`setfacl(1)`/`acl(5)`
  - macOS: `chmod(1)` to create/manipulate, `ls(1)` to inspect
  - NetBSD: upcoming in NetBSD 10.0; see `https://man.netbsd.org/posixle.3`