Advanced Programming in the UNIX Environment

Week 13, Segment 2:
eUIDs, file flags, mount options, securelevels

Department of Computer Science
Stevens Institute of Technology

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

Changing eUIDs

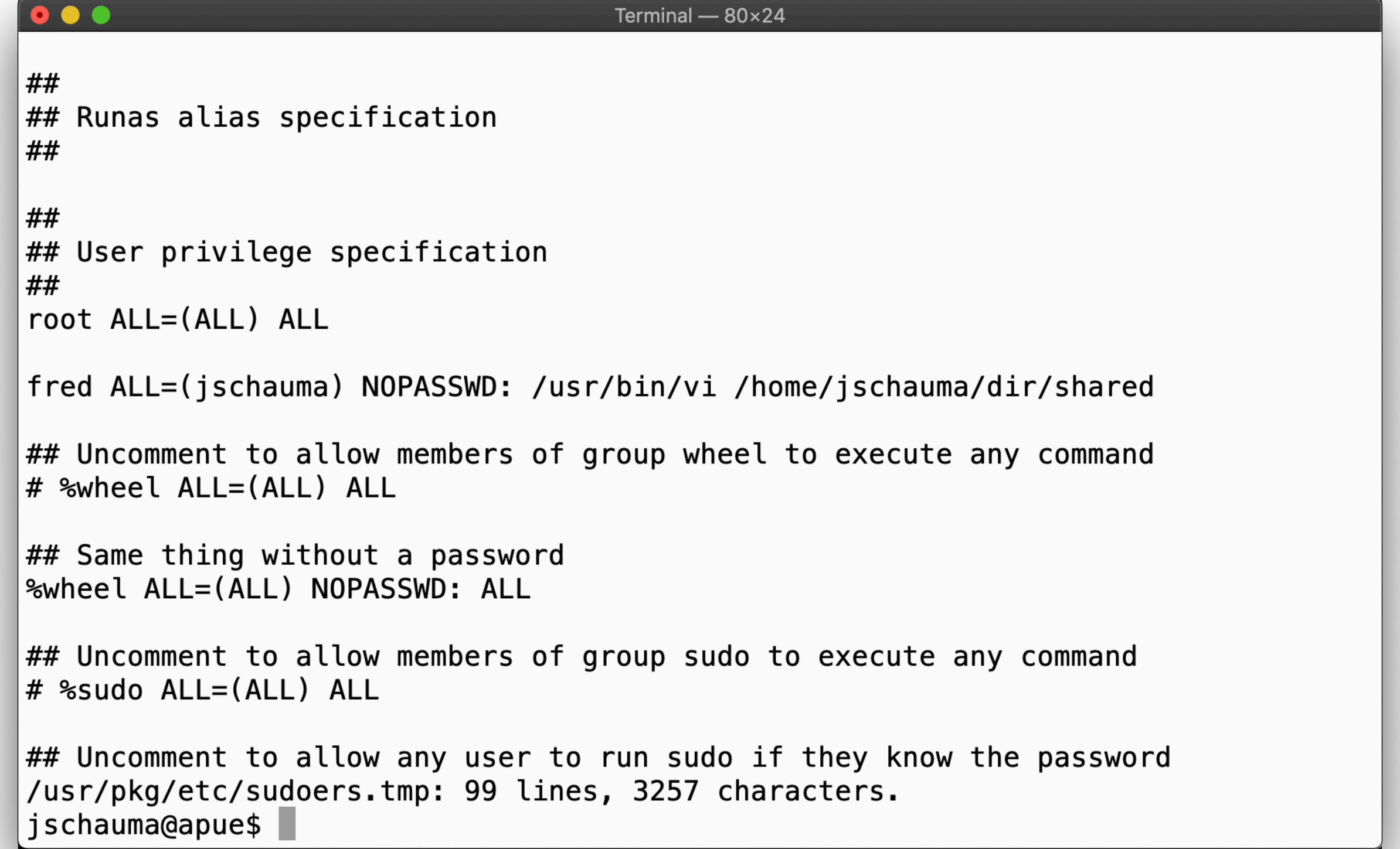
ACLs control access to files and directories by eUID/eGID. Recall from Week 03, Segment 2 that we can change those: `setuid.c`

Common examples:

- necessary access to privileged resources (*e.g.*, binding to a port<1024, use of raw sockets for ICMP, …)
- handling logins (*e.g.*, `login(1)`, `sshd(8)`)
- *raising* and *changing* privileges (*e.g.*, `su(1)`, `sudo(8)`)

Pitfalls when changing eUIDs

- setuid programs
  - require careful raising and lowering privileges *only when needed* (Least Privilege)
  - rely on correct ownership and permissions (*i.e.*, factors outside of the control of the program)
- `su(1)`
  - requires sharing of a password
  - grants all or nothing access
- `sudo(8)`
  - often misconfigured granting too broad access (`ALL:ALL`)
  - additional authentication often dropped (`NOPASSWD`)
  - restrictions often overlook privilege escalation


Terminal — 80×24
##
Runas alias specification
##
##
User privilege specification
##
root ALL=(ALL) ALL
fred ALL=(jschauma) NOPASSWD: /usr/bin/vi /home/jschauma/dir/shared
Uncomment to allow members of group wheel to execute any command
%wheel ALL=(ALL) ALL
Same thing without a password
%wheel ALL=(ALL) NOPASSWD: ALL
Uncomment to allow members of group sudo to execute any command
%sudo ALL=(ALL) ALL
Uncomment to allow any user to run sudo if they know the password
/usr/pkg/etc/sudoers.tmp: 99 lines, 3257 characters.
jschauma@apue$

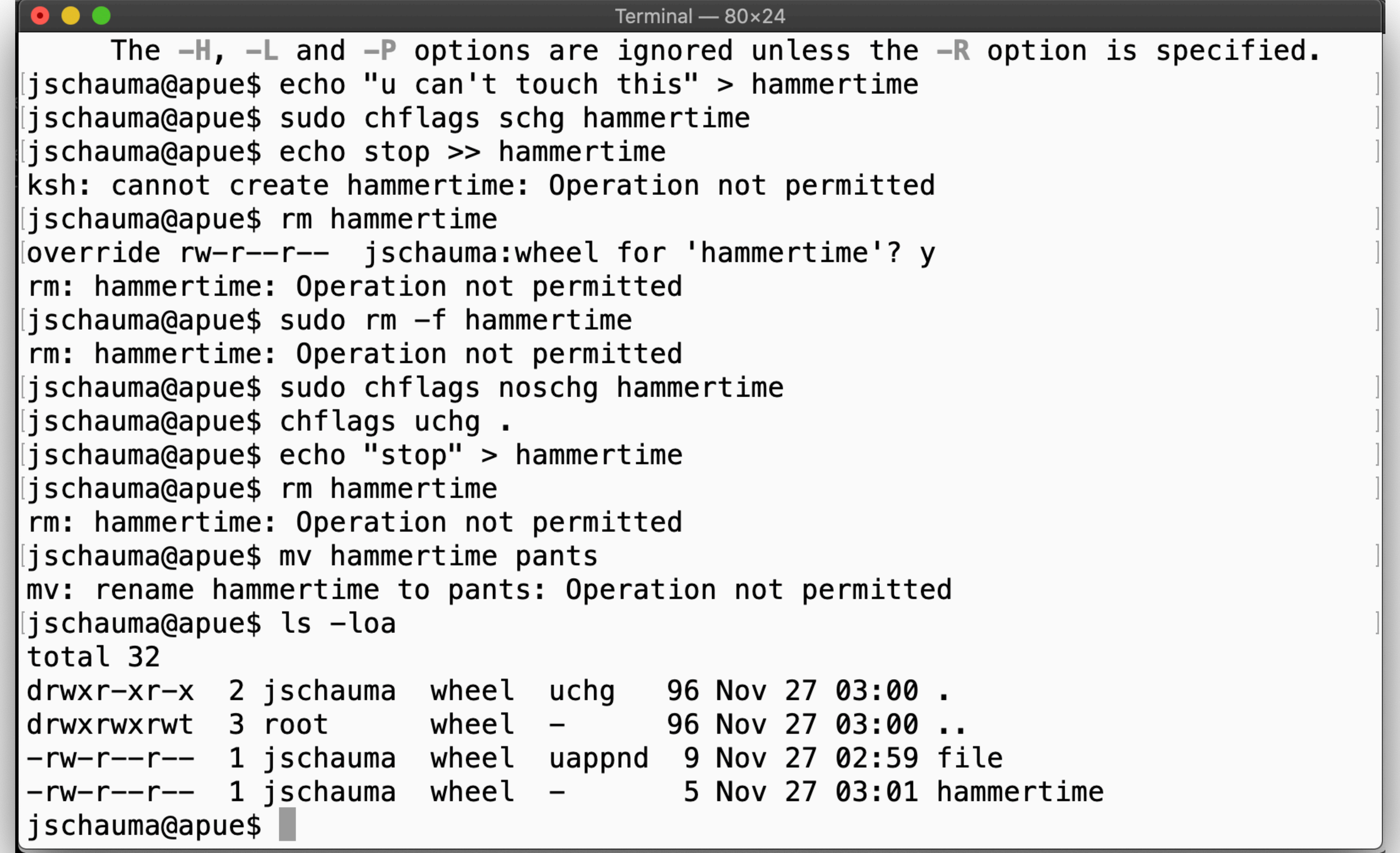
chflags(2)

```
#include <sys/stat.h>
#include <unistd.h>
int chflags(const char *path, u_long flags);
int lchflags(const char *path, u_long flags);
int fchflags(int fd, u_long flags);
```

Returns: 0 on success, -1 on error

Your eUID controls access to resources. But we can restrict certain access further via *e.g.*, “file flags”:

`UF_APPEND`	The file may only be appended to. (owner or super-user)
`UF_IMMUTABLE`	The file may not be changed. (owner or super-user)
`SF_APPEND`	The file may only be appended to. (super-user only)
`SF_IMMUTABLE`	The file may not be changed. (super-user only)


Terminal — 80×24
The -H, -L and -P options are ignored unless the -R option is specified.
jschauma@apue$ echo "u can't touch this" > hammertime
jschauma@apue$ sudo chflags schg hammertime
jschauma@apue$ echo stop >> hammertime
ksh: cannot create hammertime: Operation not permitted
jschauma@apue$ rm hammertime
override rw-r--r-- jschauma:wheel for 'hammertime'? y
rm: hammertime: Operation not permitted
jschauma@apue$ sudo rm -f hammertime
rm: hammertime: Operation not permitted
jschauma@apue$ sudo chflags noschg hammertime
jschauma@apue$ chflags uchg .
jschauma@apue$ echo "stop" > hammertime
jschauma@apue$ rm hammertime
rm: hammertime: Operation not permitted
jschauma@apue$ mv hammertime pants
mv: rename hammertime to pants: Operation not permitted
jschauma@apue$ ls -loa
total 32
drwxr-xr-x 2 jschauma wheel uchg   96 Nov 27 03:00 .
drwxrwxrwt 3 root   wheel -   96 Nov 27 03:00 ..
-rw-r--r-- 1 jschauma wheel uappnd 9 Nov 27 02:59 file
-rw-r--r-- 1 jschauma wheel -   5 Nov 27 03:01 hammertime
jschauma@apue$

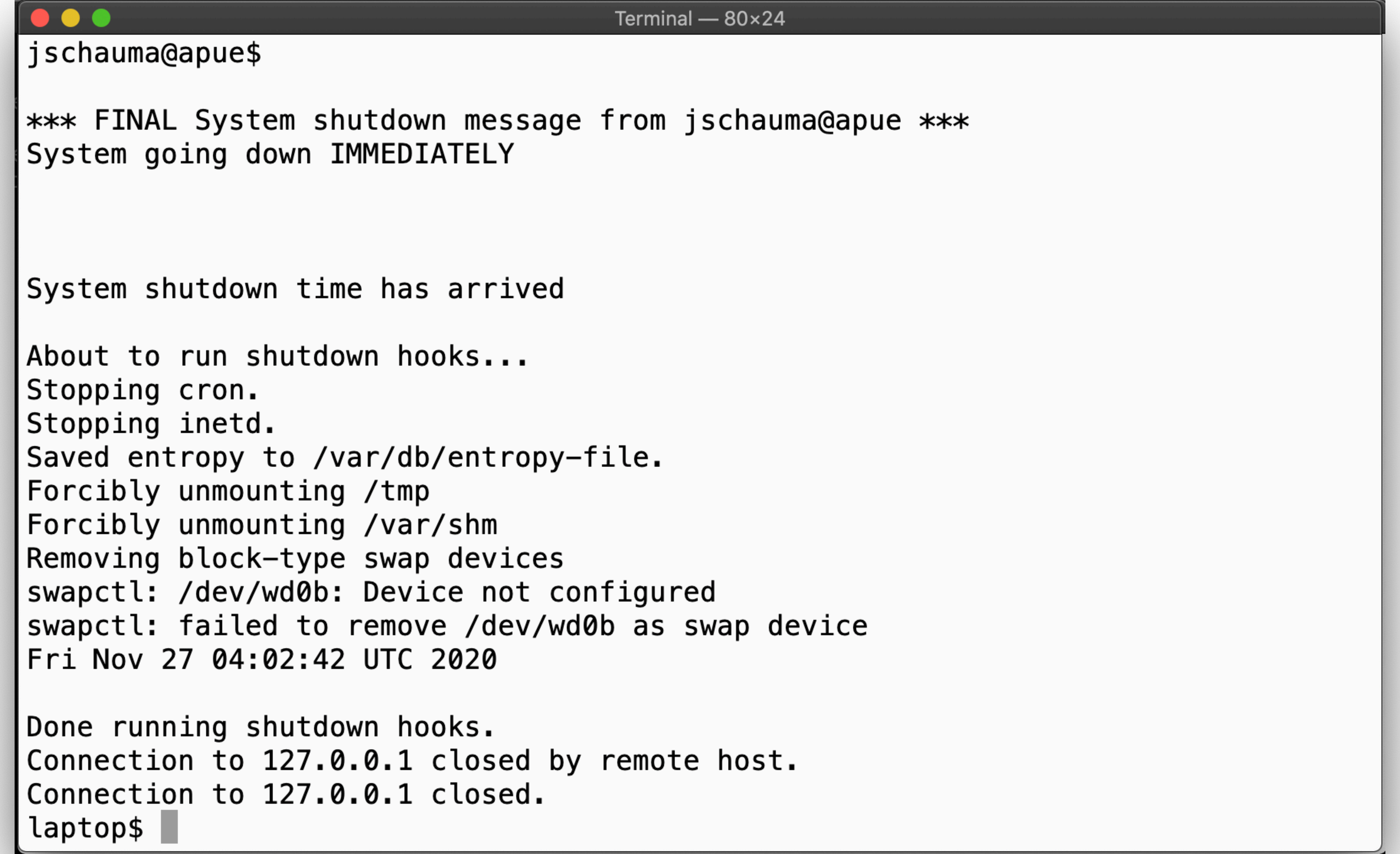
```
Terminal — 80×24
                        file data are invalidated and all inode data are re-
                        read for all active vnodes.

jschauma@apue$ sudo mount -u -o noexec,rdonly /mnt
jschauma@apue$ mount | grep /mnt
/dev/wd1a on /mnt type ffs (noexec, read-only, local)
jschauma@apue$ rm -f a.out
rm: a.out: Read-only file system
jschauma@apue$ sudo rm -f a.out
rm: a.out: Read-only file system
jschauma@apue$ touch newfile
touch: newfile: Read-only file system
jschauma@apue$ ls -la
total 23
drwxr-xr-x   2 jschauma  wheel   512 Nov 27 03:43 .
drwxr-xr-x  21 root      wheel   512 Aug 10 22:54 ..
-rwsr-xr-x   1 nobody    wheel  8392 Nov 27 03:43 a.out
-rwx------   1 jschauma  wheel   193 Nov 27 03:25 setuid.c
jschauma@apue$ sudo mount -u -o rw /mnt
jschauma@apue$ rm a.out
override rwsr-xr-x  nobody:wheel for 'a.out'? y
jschauma@apue$ mount | grep /mnt
/dev/wd1a on /mnt type ffs (local)
jschauma@apue$
```

securelevels

To prevent even eUID 0 from *e.g.*, changing the mount flags, you can employ *securelevels*:

- superuser can raise the securelevel, only `init(8)` can lower it
- in other words, lowering requires a reboot
- four securelevels are defined
  - `-1` “Permanently insecure mode”
  - `0` “Insecure mode”
  - `1` “Secure mode”
  - `2` “Highly secure mode”
- see `secmodel_securelevel(9)`


Terminal — 80×24
jschauma@apue$
*** FINAL System shutdown message from jschauma@apue ***
System going down IMMEDIATELY
System shutdown time has arrived
About to run shutdown hooks...
Stopping cron.
Stopping inetd.
Saved entropy to /var/db/entropy-file.
Forcibly unmounting /tmp
Forcibly unmounting /var/shm
Removing block-type swap devices
swapctl: /dev/wd0b: Device not configured
swapctl: failed to remove /dev/wd0b as swap device
Fri Nov 27 04:02:42 UTC 2020
Done running shutdown hooks.
Connection to 127.0.0.1 closed by remote host.
Connection to 127.0.0.1 closed.
laptop$

Summary

- su(1) and sudo(8) can be used to grant others the ability to run commands as another user, but it can be difficult to restrict access
- “file flags” may restrict certain use; see chflags(1)/chflags(2) on BSD, chattr(1) on Linux
- mount options like noexec, nosuid, rdonly can restrict and protect filesystems per mount point
- to prevent even root from undoing these protections, use securelevels (reboots are noisy.)