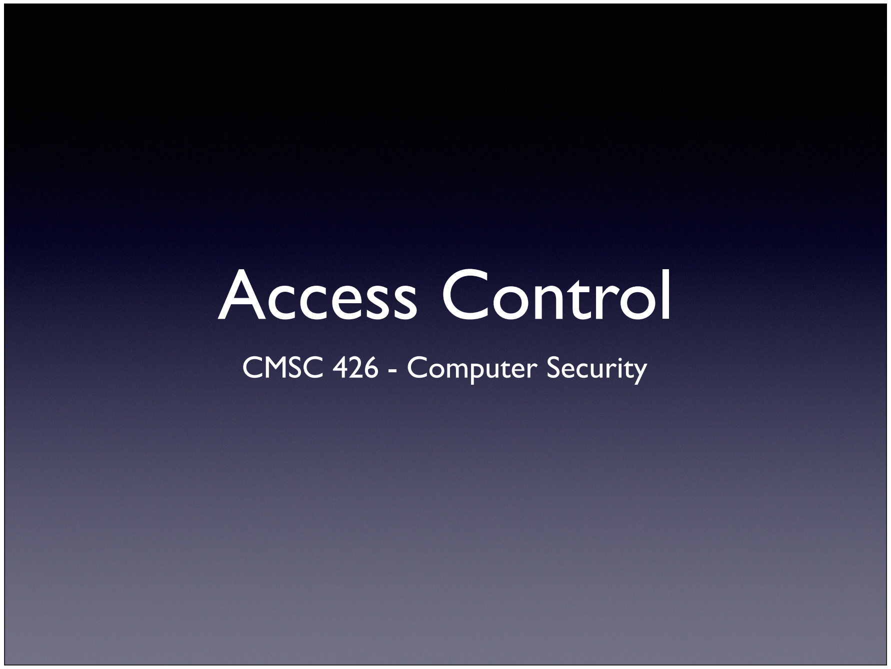
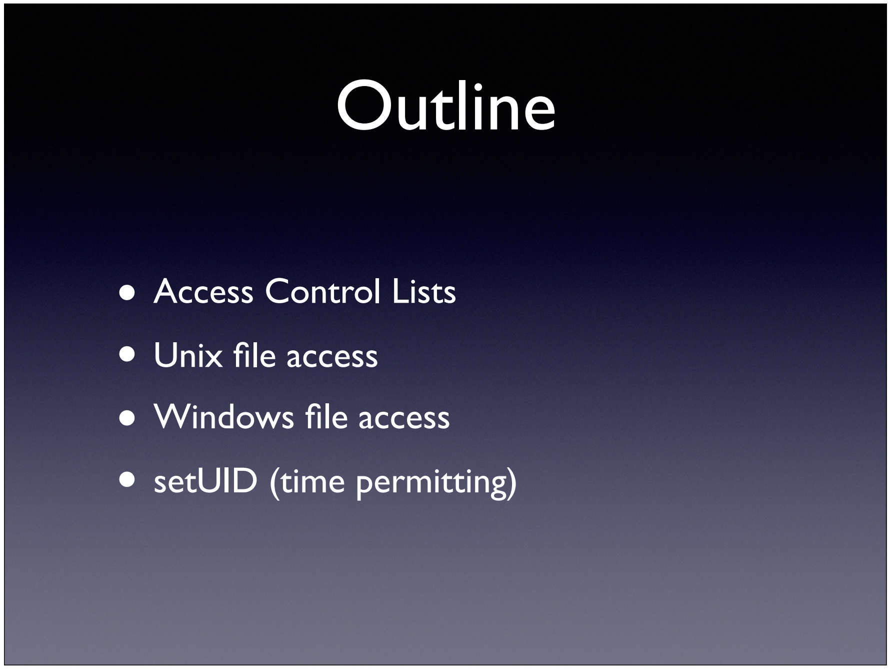


Access Control

CMSC 426 - Computer Security

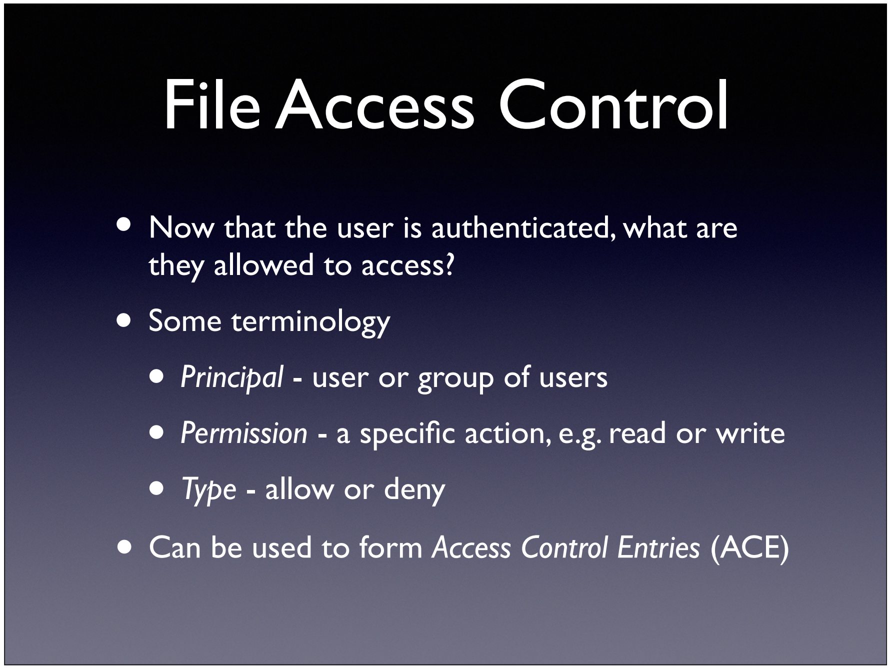
1



Outline

- Access Control Lists
- Unix file access
- Windows file access
- setUID (time permitting)

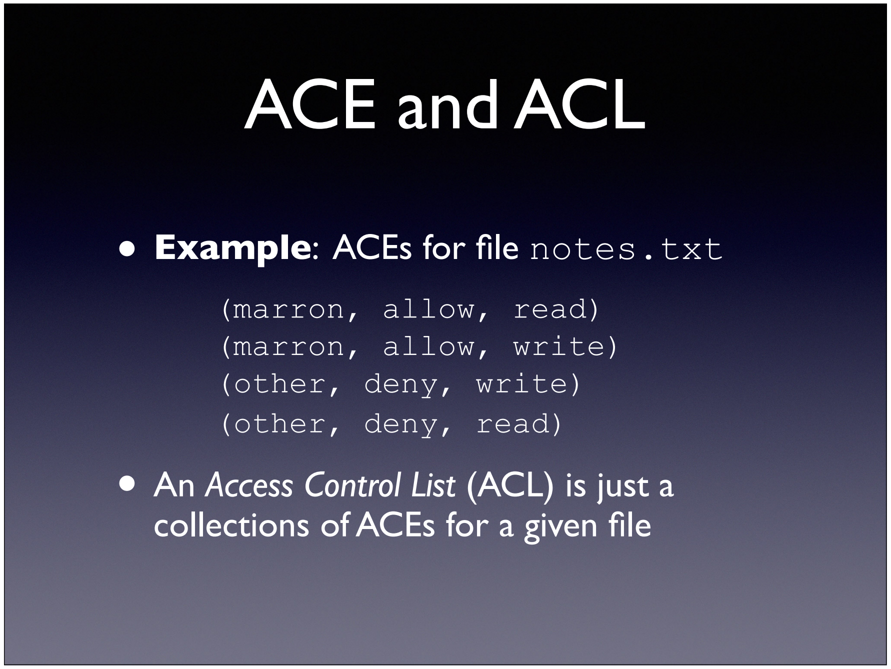
2



File Access Control

- Now that the user is authenticated, what are they allowed to access?
- Some terminology
  - *Principal* - user or group of users
  - *Permission* - a specific action, e.g. read or write
  - *Type* - allow or deny
- Can be used to form *Access Control Entries* (ACE)

3



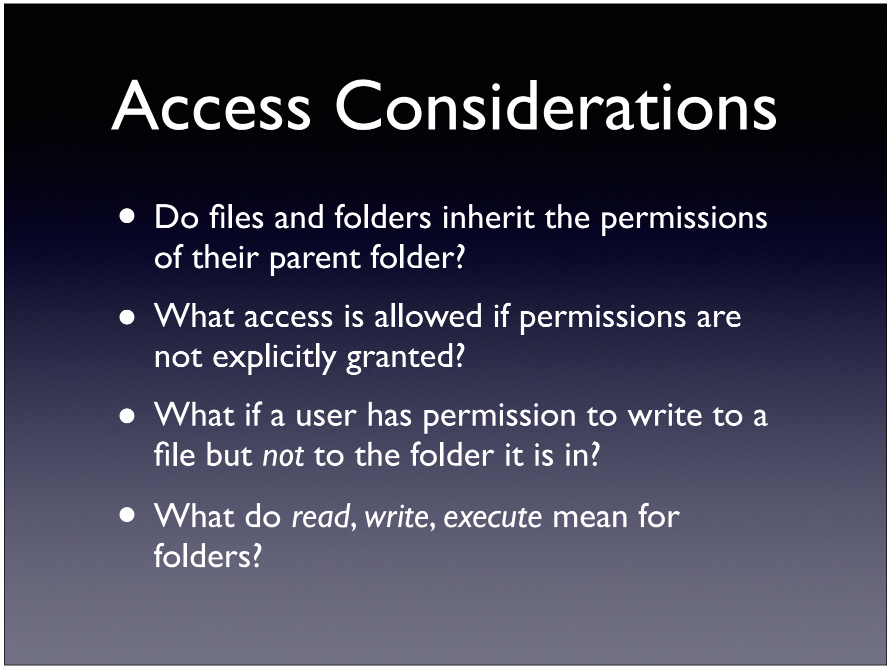
ACE and ACL

- **Example**: ACEs for file `notes.txt`

```
(marron, allow, read)
(marron, allow, write)
(other, deny, write)
(other, deny, read)
```

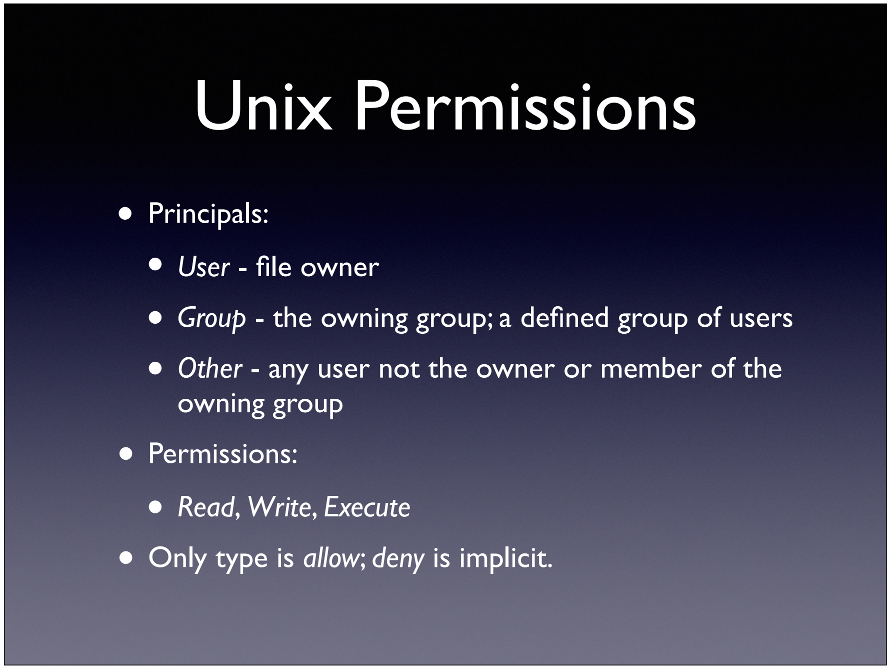
- An *Access Control List* (ACL) is just a collections of ACEs for a given file

4



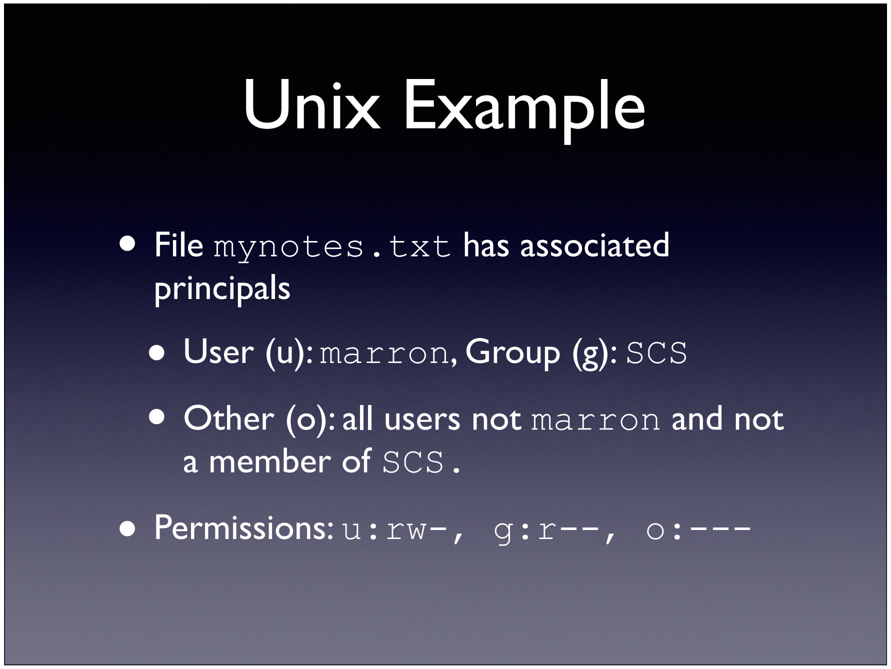
Access Considerations

- Do files and folders inherit the permissions of their parent folder?
- What access is allowed if permissions are not explicitly granted?
- What if a user has permission to write to a file but *not* to the folder it is in?
- What do *read*, *write*, *execute* mean for folders?



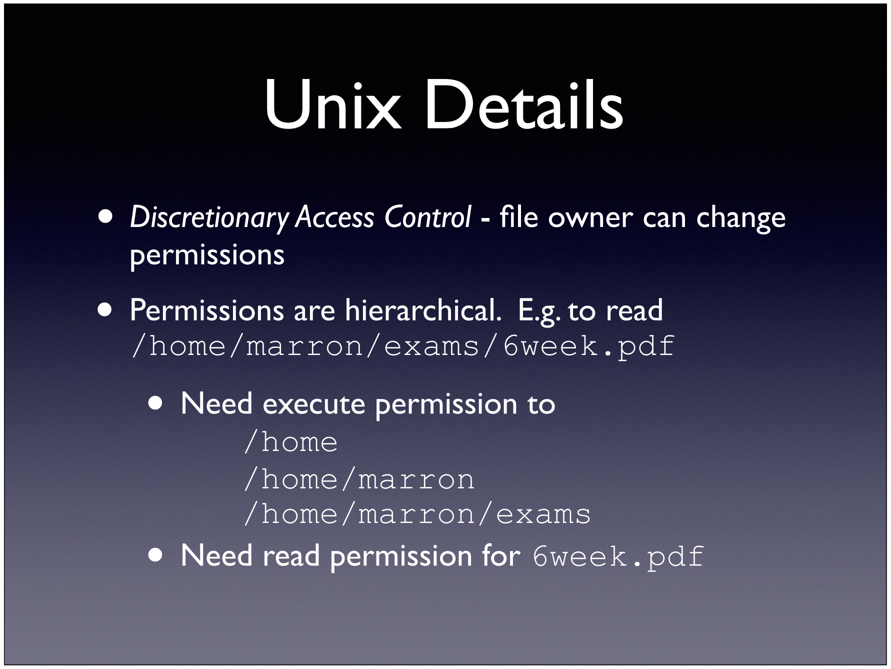
Unix Permissions

- Principals:
  - *User* - file owner
  - *Group* - the owning group; a defined group of users
  - *Other* - any user not the owner or member of the owning group
- Permissions:
  - *Read*, *Write*, *Execute*
- Only type is *allow*; *deny* is implicit.



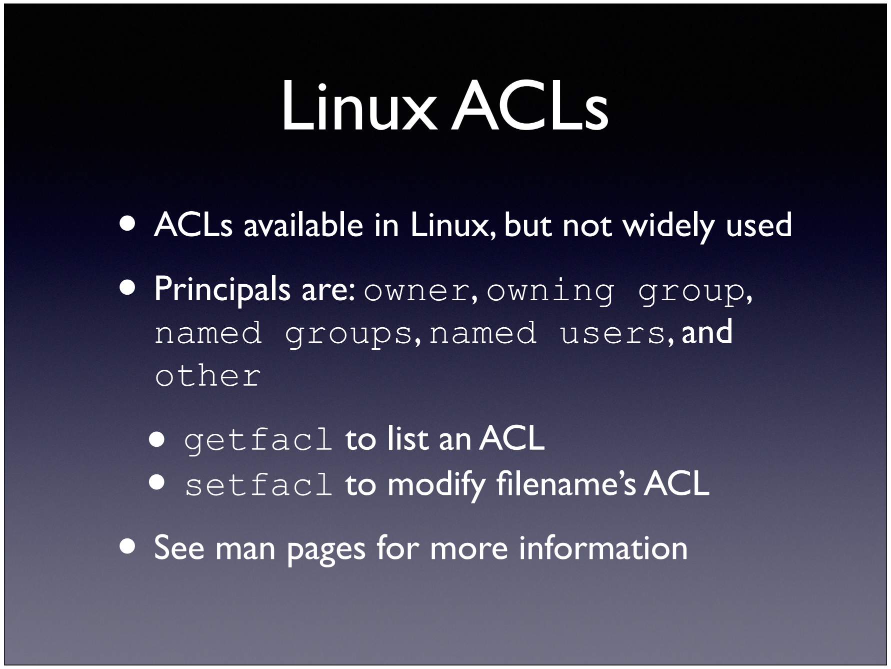
Unix Example

- File `mynotes.txt` has associated principals
  - User (u): `marron`, Group (g): `SCS`
  - Other (o): all users not `marron` and not a member of `SCS`.
- Permissions: `u:rw-, g:r--, o:---`



Unix Details

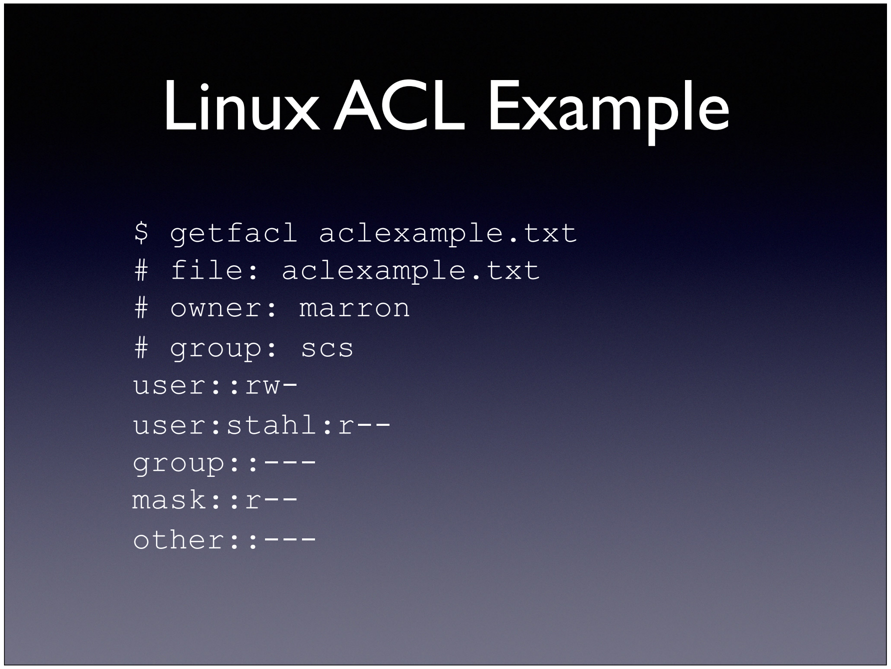
- *Discretionary Access Control* - file owner can change permissions
- Permissions are hierarchical. E.g. to read `/home/marron/exams/6week.pdf`
  - Need execute permission to
    `/home`
    `/home/marron`
    `/home/marron/exams`
  - Need read permission for `6week.pdf`



Linux ACLs

- ACLs available in Linux, but not widely used
- Principals are: `owner`, `owning group`, `named groups`, `named users`, and `other`
  - `getfacl` to list an ACL
  - `setfacl` to modify filename's ACL
- See man pages for more information

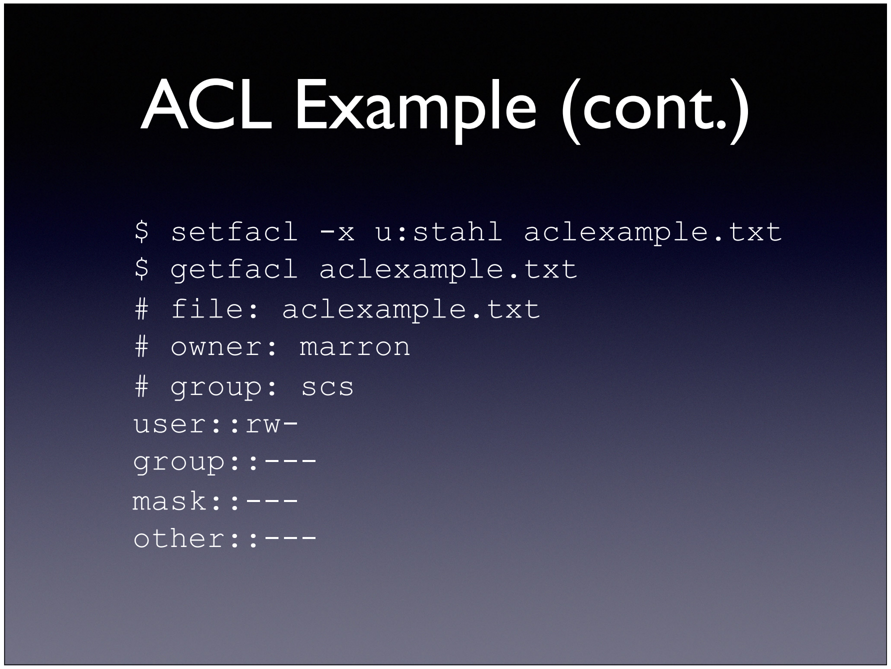
9



Linux ACL Example

```
$ getfacl aclexample.txt
# file: aclexample.txt
# owner: marron
# group: scs
user::rw-
user:stahl:r--
group::---
mask::r--
other::---
```

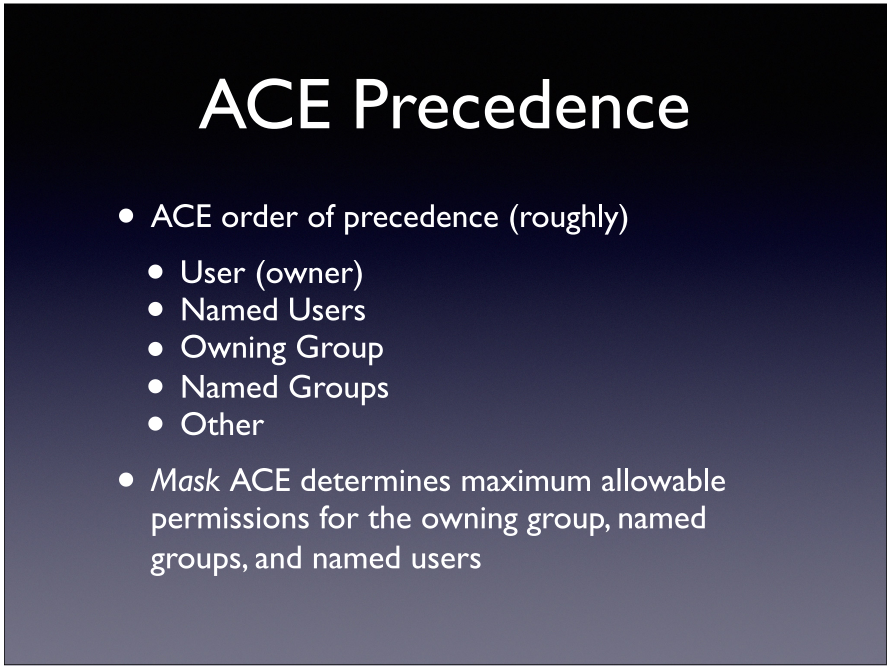
10



ACL Example (cont.)

```
$ setfacl -x u:stahl aclexample.txt
$ getfacl aclexample.txt
# file: aclexample.txt
# owner: marron
# group: scs
user::rw-
group::---
mask::---
other::---
```

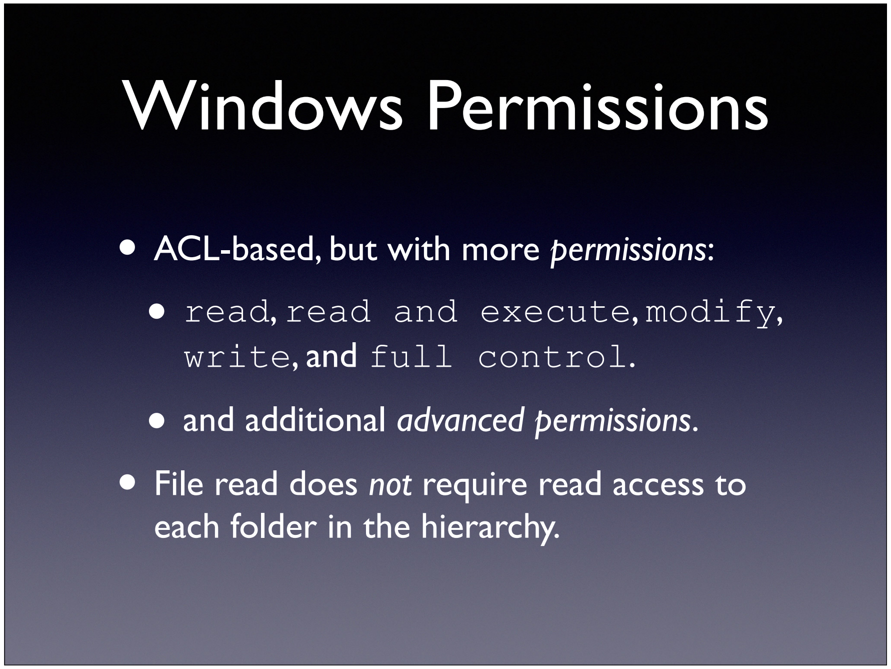
11



ACE Precedence

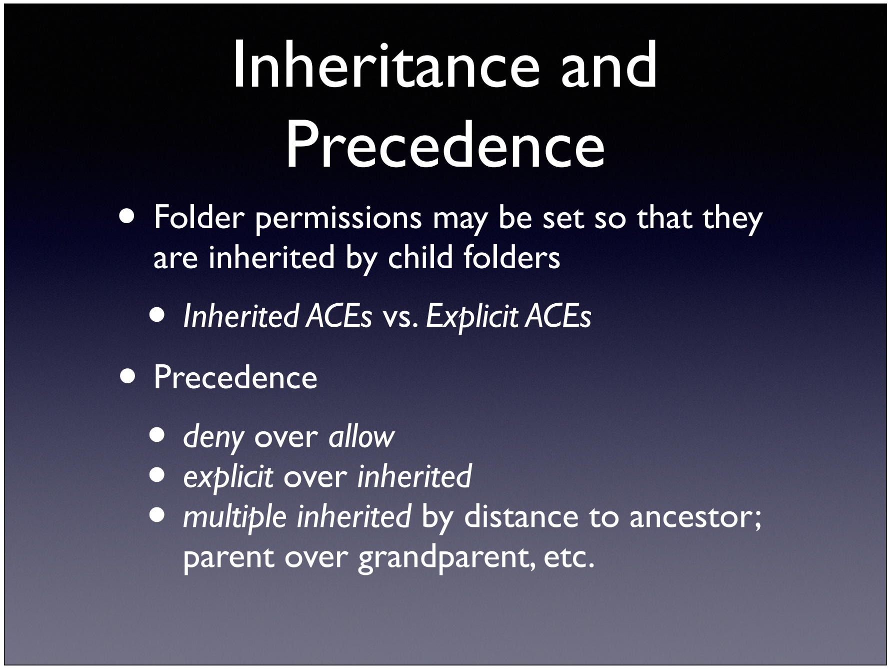
- ACE order of precedence (roughly)
  - User (owner)
  - Named Users
  - Owning Group
  - Named Groups
  - Other
- *Mask* ACE determines maximum allowable permissions for the owning group, named groups, and named users

12



Windows Permissions

- ACL-based, but with more *permissions*:
  - `read`, `read and execute`, `modify`, `write`, and `full control`.
  - and additional *advanced permissions*.
- File read does *not* require read access to each folder in the hierarchy.



Inheritance and Precedence

- Folder permissions may be set so that they are inherited by child folders
  - *Inherited ACEs* vs. *Explicit ACEs*
- Precedence
  - *deny* over *allow*
  - *explicit* over *inherited*
  - *multiple inherited* by distance to ancestor; parent over grandparent, etc.