

Access Control Lists (ack-els)

Andrina Kelly

C.O.R.E. Feature Animation
&
AFP548.com

What is an ACL?

A File Attribute

controls permissions

controls access to
files

very good...

But what does
that mean to me?

The old way...
(kick it old skool)

Unix File Permissions

File owned by one user

File owned by one group

Ability to only control
3 actions

Read

Write

Execute

For each action:

Ability to grant access

Ability to deny access

Total of 9 available
permissions

-rwxrwxrwx



So, unix permissions are:

A Standard Across *nix Systems

Relatively Flexible

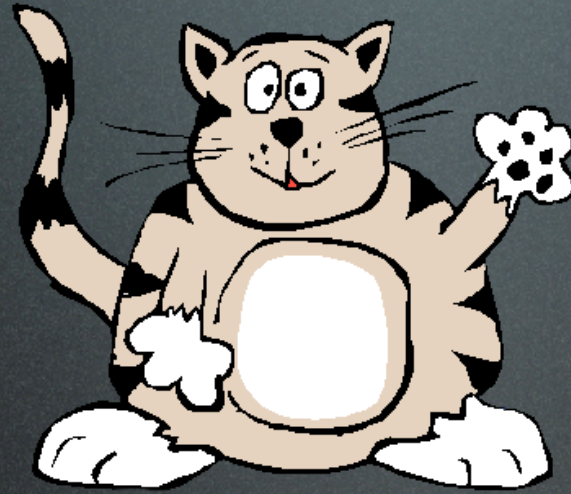
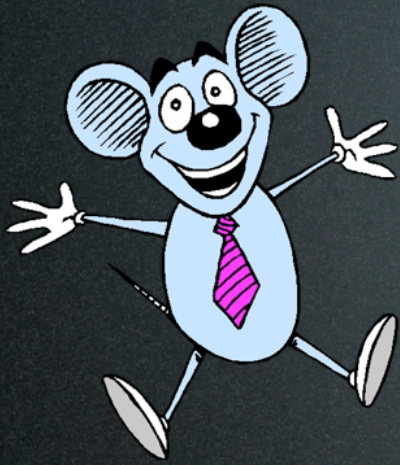
Easy to Understand

However...

What if you want
more?

What if you want....

- The owner to be a group
- More than one group assigned to an entity
- A user to be in more than 16 groups
- More flexibility
- Set your permissions from a Windows based OS



~~fatcat staff drwxr-xr-x~~

~~fatcat domestics drwxrwxr-x~~

Now What!?

Enter Access Control
Lists

A little more detail...

An ACL is an ordered
list of rules that
control file
permissions.

Each rule specifies 3
things

1. A User or Group

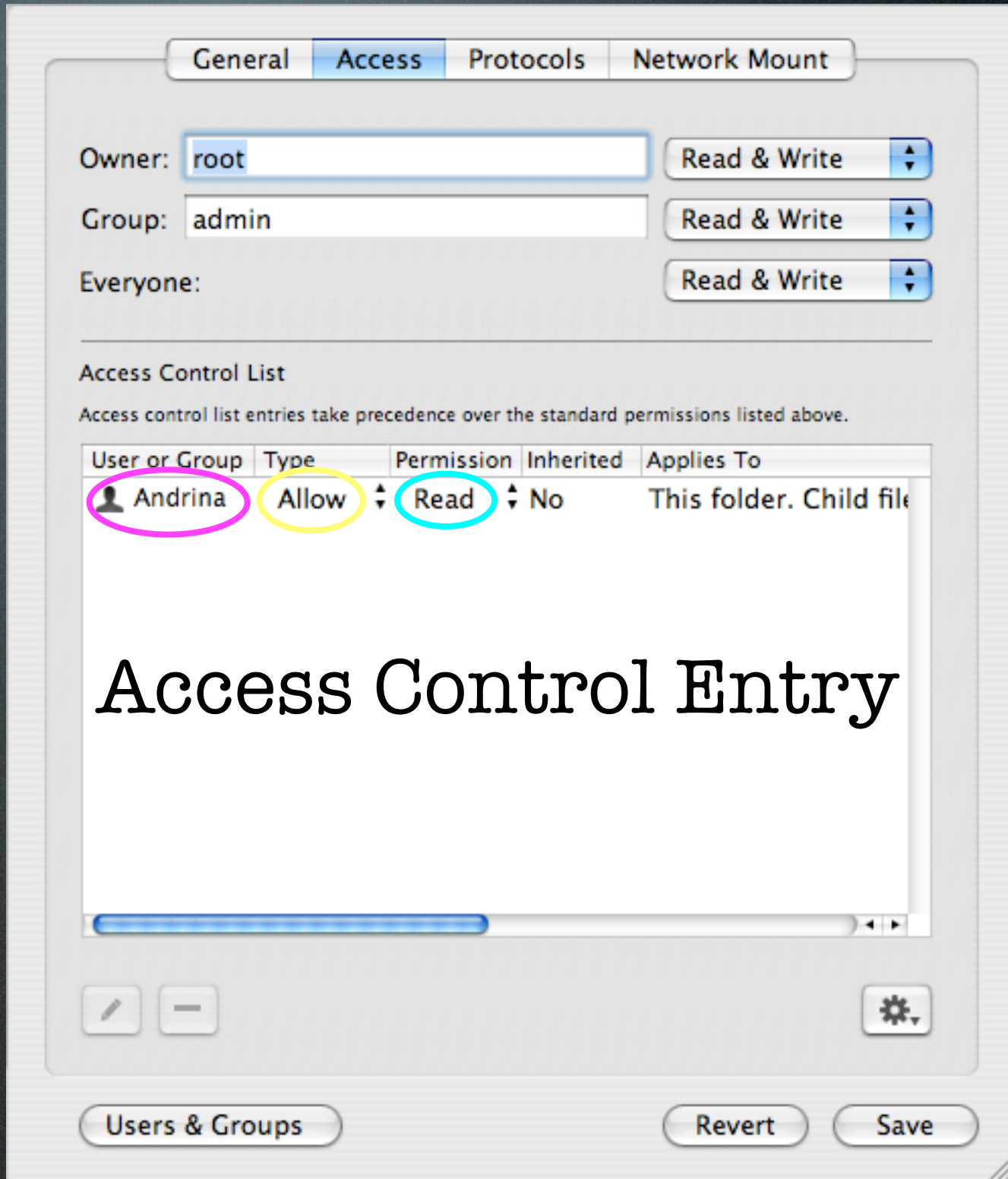
2. An Action

3. If that Action is
Allowed or Denied

User

Action

Allowed or Denied



User/Group: Andrina Kelly

Permission Type: Allow

Inherited: No

Permission

- Administration
 - Change Permissions
 - Change Owner
- Read
 - Read Attributes
 - Read Extended Attributes
 - List Folder Contents (Read Data)
 - Traverse Folder (Execute File)
 - Read Permissions
- Write
 - Write Attributes
 - Write Extended Attributes
 - Create Files (Write Data)
 - Create Folder (Append Data)
 - Delete
 - Delete Subfolders and Files
- Inheritance
 - Apply to this folder
 - Apply to child folders
 - Apply to child files
 - Apply to all descendants

Cancel

OK

User

Action

Allowed or Denied

What about that
inheritance column?

Inheritance allows
you to determine how
an ACL is passed from
parent to
descendants

Propagated at 2
distinct times

1. When a file or folder is created, the kernel determines what permissions are inherited from the parent

2. After you set an explicit ACE for a folder, Workgroup Manager propagates to the descendants (or the propagate permissions action)

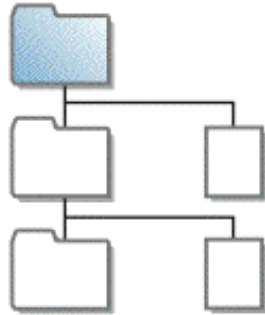
Propagate Permissions Action

Select the information you want to propagate to child objects.

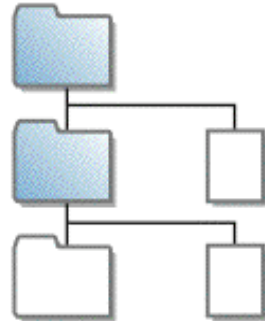
<input type="checkbox"/> Owner name	<input type="checkbox"/> Owner permissions
<input type="checkbox"/> Group name	<input type="checkbox"/> Group permissions
	<input type="checkbox"/> Everyone permissions

Access Control List

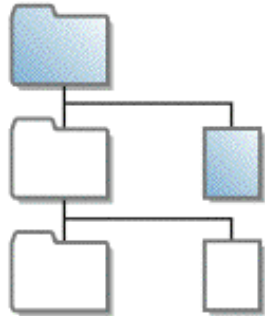
- ▼ Inheritance
 - Apply to this folder
 - Apply to child folders
 - Apply to child files
 - Apply to all descendants



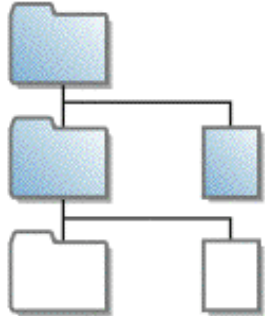
- ▼ Inheritance
 - Apply to this folder
 - Apply to child folders
 - Apply to child files
 - Apply to all descendants



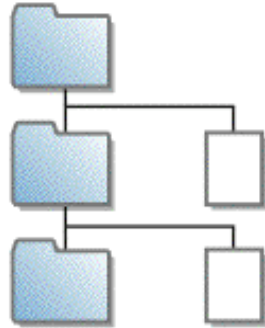
- ▼ Inheritance
 - Apply to this folder
 - Apply to child folders
 - Apply to child files
 - Apply to all descendants



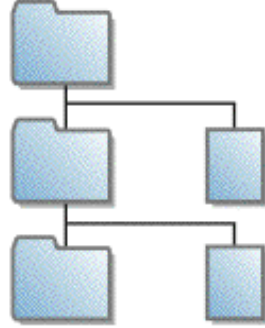
- ▼ Inheritance
 - Apply to this folder
 - Apply to child folders
 - Apply to child files
 - Apply to all descendants



- ▼ Inheritance
 - Apply to this folder
 - Apply to child folders
 - Apply to child files
 - Apply to all descendants

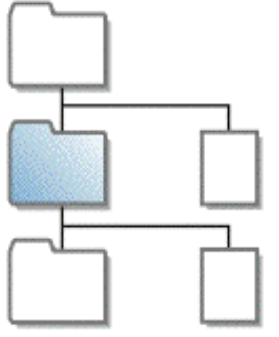


- ▼ Inheritance
 - Apply to this folder
 - Apply to child folders
 - Apply to child files
 - Apply to all descendants



▼ Inheritance

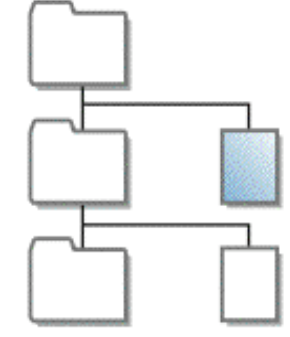
- Apply to this folder
- Apply to child folders
- Apply to child files
- Apply to all descendants



A hierarchical folder tree diagram with a root folder at the top. The root folder has two child folders. The left child folder is highlighted in blue. This left child folder has two child files.

▼ Inheritance

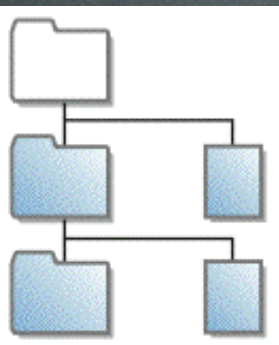
- Apply to this folder
- Apply to child folders
- Apply to child files
- Apply to all descendants



A hierarchical folder tree diagram with a root folder at the top. The root folder has two child folders. The right child folder is highlighted in blue. This right child folder has two child files.

▼ Inheritance

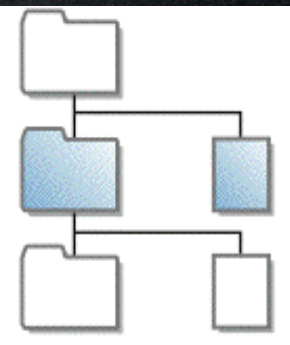
- Apply to this folder
- Apply to child folders
- Apply to child files
- Apply to all descendants



A hierarchical folder tree diagram with a root folder at the top. The root folder has two child folders. Both child folders are highlighted in blue. The left child folder has two child files, and the right child folder has two child files.

▼ Inheritance

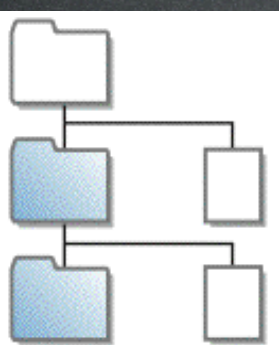
- Apply to this folder
- Apply to child folders
- Apply to child files
- Apply to all descendants



A hierarchical folder tree diagram with a root folder at the top. The root folder has two child folders. Both child folders are highlighted in blue. The left child folder has two child files, and the right child folder has two child files.

▼ Inheritance

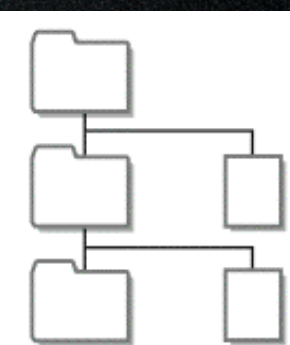
- Apply to this folder
- Apply to child folders
- Apply to child files
- Apply to all descendants



A hierarchical folder tree diagram with a root folder at the top. The root folder has two child folders. Both child folders are highlighted in blue. The left child folder has two child files, and the right child folder has two child files.

▼ Inheritance

- Apply to this folder
- Apply to child folders
- Apply to child files
- Apply to all descendants



A hierarchical folder tree diagram with a root folder at the top. The root folder has two child folders. The left child folder has two child files, and the right child folder has two child files. No folders are highlighted.

Actions

17 distinct attributes

Permission

- ▼ Administration
 - Change Permissions
 - Change Owner
- ▼ Read
 - Read Attributes
 - Read Extended Attributes
 - List Folder Contents (Read Data)
 - Traverse Folder (Execute File)
 - Read Permissions
- ▼ Write
 - Write Attributes
 - Write Extended Attributes
 - Create Files (Write Data)
 - Create Folder (Append Data)
 - Delete
 - Delete Subfolders and Files
- ▼ Inheritance
 - Apply to this folder
 - Apply to child folders
 - Apply to child files
 - Apply to all descendants

That's 98,304
combinations of
actions...

98,304!

How do I manage
98,304 actions!?

Manage Actions at the Group Level

- Assign individual actions only as an exception
- You can add and delete users from groups without having to change actions on all folders/files
- For example, allow all “animals” read and write on then deny “Max Mouse” write

Gradually Add Permissions

- If only using Allow, the permissions are additive
- i.e. allow “animals” read only on full share point, then allow “animals” write in a sub-folder

Use the Effective Permissions Inspector

- Quickly shows the user's permissions in a certain folder
- use it after changing ACLs

Use “Deny” only when you have to

- When a deny ACE is encountered, it overrides the Allow ACEs
- i.e. use allow “animals” read as opposed to allow all “animals” read and write & deny “animals” write

Don't propagate permissions unless you have to

- Inheritance is very powerful, planning is important (make your will now)
- Propagation is forcing inheritance, and does not have an undo (consider birth control...)

KISS

(Keep it Stupid Simple)

- Even though you have 98,304 (!!)
actions, it might make more sense to
use the standard UNIX permissions
- A simple logical folder structure is a
great start

Sounds simple
enough...

Where to start?

Plan!

Plan some more...

Plan again!

Think about what you
want to accomplish?

Draw out a basic
folder structure on
paper

Decide what groups
are going to have
access to which
directories

Assign users to
groups

Double-check your
plan

Create a test
environment

Test it!

Plan some more...

Demo 1

Anyone can create,
Few can delete

(demo summary)

Demo 2

It works in Education
too

(demo summary)

I ♥ Terminal

Enable ACLs via the
command line?

```
fsaclctl -p / -e
```


fsaclctl?

File System ACL ConTroL

fsaclctl -p / -e

ls -le


```
cactus:/tmp andrina$ ls -le
drwxrwxrwt + 8 andrina  admin  272 Nov 30 17:55 folder
0: user:admin allow delete
```


drwxrwxrwt +

drwxrwxrwt +


```
chmod +a "fatcat allow delete" folder
```



```
cactus:/tmp andrina$ ls -le
drwxrwxrwt + 8 andrina admin 272 Nov 30 17:55 folder
l: user:fatcat allow delete
O: user:admin allow delete
```


In summary

ACLs can be complex

plan, test, plan some
more, test some
more...

Understand actions

Understand
inheritance

Resources

- [Tiger Server Manuals](#)
- [AFP548.com](#)
- [developer.apple.com](#)
- [discussions.info.apple.com](#)
- [lists.apple.com/mailman/listinfo/macos-x-server](#)

Q & A
(thanks!)