A simple DAC example: Unix

Unix provides a mechanism based on simplified ACLs suitable for a restricted class of DAC policies.

- Controls access per object (file, directory, ...) using permission scheme *owner/group/other*.
- Objects are assigned to a single user (owner) and a single group (normally the group of the directory containing it). They can be changed by using the `chown` and `chgrp` commands, respectively.
- Each user can be assigned to multiple groups (cf. `useradd`).
- Permission bits assigned to objects by their owners or by the administrator (root), (cf. `chmod`).

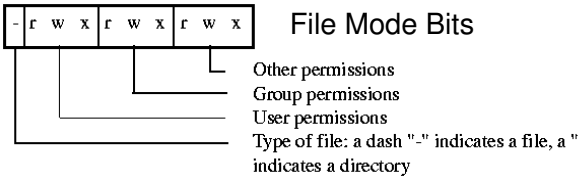
```
> ls -la
drwx------ 8 armando users 12288 May 26 22:34 .
drwx------ 9 armando users  4096 May 26 19:07 ..
-rw-r--r-- 1 armando users  6523 May 27 00:35 access.tex
drwxr-xr-x 2 armando users  2048 May 26 22:27 fig/
> groups
users admin libvirtd
```

A simple DAC example: Unix

Unix provides a mechanism based on simplified ACLs suitable for a restricted class of DAC policies.

- Controls access per object (file, directory, ...) using permission scheme *owner/group/other*.
- Objects are assigned to a single user (owner) and a single group (normally the group of the directory containing it). They can be changed by using the `chown` and `chgrp` commands, respectively.
- Each user can be assigned to multiple groups (cf. `useradd`).
- Permission bits assigned to objects by their owners or by the administrator (root), (cf. `chmod`).

```
> ls -la
drwx------ 8 armando
drwx------ 9 armando
-rw-r--r-- 1 armando
drwxr-xr-x 2 armando
> groups
users admin libvirtd
```



Structure of File Mode Bits

The file mode bits have two parts:

- file permission bits, which control ordinary access to the file
- special mode bits, which affect only only executable files (programs) and, on most systems, directories

File Permission Bits

File permission bits control ordinary access to the file:

1. permission to read the file. For directories, this means permission to list the contents of the directory.
2. permission to write to (change) the file. For directories, this means permission to create and remove files in the directory.
3. permission to execute the file (run it as a program). For directories, this means permission to access files in the directory.

Special Mode Bits

Special mode bits affect only only executable files (programs) and directories:

1. Set the process's effective user ID to that of the file upon execution (called the "set-user-ID bit", or "setuid bit").
2. Set the process's effective group ID to that of the file upon execution (called the "set-group-ID bit", or "setgid bit").
3. When a user other than the owner executes the file, the process will run with user and/or group permissions set upon it by its owner. For example, if the file is owned by user root and group wheel, it will run as root:wheel no matter who executes the file.
4. Prevent unprivileged users from removing or renaming a file in a directory unless they own the file or the directory; this is called the "restricted deletion flag" for the directory, and is commonly found on world-writable directories like `/tmp`.

A simple DAC example: Unix

- Not all policies can be directly mapped onto this mechanism.
  How would we express that a patient can access his medical records at a hospital?
  Who owns the records?
- Supports limited delegation of rights using setuid [or setgid].
  - Executor takes on owner's user [group] identity during execution.
  - Example: normal users "upgraded" to root to change their passwords in the password file.
  - Open to abuse and the cause of many security holes.

A simple DAC example: Unix

- Not all policies can be directly mapped onto this mechanism.
  How would we express that a patient can access his medical records at a hospital? Who owns the records?
- Supports limited delegation of rights using setuid [or setgid].
  - Executor takes on owner's user [group] identity during execution.
  - Example: normal users "upgraded" to root to change their passwords in the password file.
  - Open to abuse and the cause of many security holes.