Chapter 18: Special Directions & Files

3 special permissions:

SUID (octal 4000) chmod u+s filename
SGID (octal 2000) chmod g+s filename
Sticky bit (octal 1000) chmod +t dirname

Chapter 18 : Special Directions & Files

The SUID bit is often set on an executable binary file:

chmod u+s /bin/myscript (**or** chmod *4NNN*)

The SUID bit can also be set using the four digit octal number, *4NNN*, where *NNN* are the standard permissions (rwx).

A file that has its SUID bit set will be run as though the user owner were executing the file.

Ex: /usr/bin/passwd is run as though it were user root executing it

Chapter 18 : Special Directions & Files

The SGID bit is often set on a collaborative directory:

```
mkdir   ~/project
chgrp  music   ~/project
chmod  g+s  ~/project  (or chmod 2NNN ~/project)
```

The SGID bit can also be set using the four digit octal number, *2NNN*, where *NNN* are the standard permissions (rwx).

Any **newly created** files in ~/project will now be owned by the group owner, music. Existing files will maintain their existing group owner.

Chapter 18 : Special Directions & Files

If the SGID bit is set, it appears as either an s or an S where the group owner's x permission is usually found:

```
[steven@localhost ~]$ ls -ld project/
drwxrwx---. 2 steven music 4096 Feb 26 21:15 project/
[steven@localhost ~]$ chmod g+s project/
[steven@localhost ~]$ ls -ld project/
drwxrws---. 2 steven music 4096 Feb 26 21:15 project/
[steven@localhost ~]$
```

A lowercase s means that there is a lowercase x underneath. That is, the group owner has the x permission.

Chapter 18 : Special Directions & Files

If the SGID bit is set, it appears as either an s or an S where the group owner's x permission is usually found:

```
[steven@localhost ~]$ ls -ld project/
drwxrw----. 2 steven music 4096 Feb 26 21:15 project/
[steven@localhost ~]$ chmod g+s project/
[steven@localhost ~]$ ls -ld project/
drwxrwS---. 2 steven music 4096 Feb 26 21:15 project/
[steven@localhost ~]$
```

An uppercase S means that there is NO lowercase x underneath. That is, the group owner has NO x permission.

Chapter 18 : Special Directions & Files

The sticky bit is used to prevent users from deleting each other's files in a directory:

chmod o+t ~/project (**or** chmod *1NNN* ~/project)

The sticky bit can also be set using the four digit octal number, *1NNN*, where *NNN* are the standard permissions (rwx).

Any newly created files in ~/project can only be deleted by their user owner, root, or the user owner of the project directory.

Chapter 18 : Special Directions & Files

If the sticky bit is set, it appears as either a t or a T where the other access class' x permission is usually found:

```
[steven@localhost ~]$ ls -ld project/
drwxrws--x. 2 steven music 4096 Feb 26 21:15 project/
[steven@localhost ~]$ chmod o+t project/
[steven@localhost ~]$ ls -ld project/
drwxrws--t. 2 steven music 4096 Feb 26 21:15 project/
[steven@localhost ~]$
```

A lowercase t means that there is a lowercase x underneath. That is, the other access class has the x permission.

Chapter 18 : Special Directions & Files

If the sticky bit is set, it appears as either a t or a T where the other access class' x permission is usually found:

```
[steven@localhost ~]$ ls -ld project/
drwxrws---. 2 steven music 4096 Feb 26 21:15 project/
[steven@localhost ~]$ chmod o+t project/
[steven@localhost ~]$ ls -ld project/
drwxrws--T. 2 steven music 4096 Feb 26 21:15 project/
[steven@localhost ~]$
```

An uppercase T means that there is NO lowercase x underneath. That is, the other access class has NO x permission.

Chapter 18 : Special Directions & Files

Using octal, you can set both the SGID and sticky bits at once. SGID has a value of 2, sticky bit a value of 1. 2+1=3.

The permissions shown here are typically found with a collaborative directory, where many users are working together on a project and need to access each other's files:

```
[steven@localhost ~]$ chmod 3770 project/
[steven@localhost ~]$ ls -ld project/
drwxrws--T. 2 steven music 4096 Feb 26 21:15 project/
[steven@localhost ~]$
```

Chapter 16

A user name is associated with:

- a unique 32 bit userid (uid)
- every process
- every file.

Chapter 16

3 kinds of users:

1. Superusers
   - userid 0
   - conventional username root
   - can map other usernames to uid 0.
2. System users
   - userid 1 to 999
   - run processes and manipulate files for RPMs
   - use /sbin/nologin as login shell.
3. Human (non-root) users
   - userid 1000 and higher recommended.

Chapter 16

Fields in /etc/passwd :

1 2 3 4 5 6 7

elvis:x:502:502::/home/elvis:/bin/bash

1: username
2: password placeholder (no longer stores passwords)
3: user id (uid)
4: primary group id (gid)
5: GECOS (usually name, contact info, comment)
6: home directory
7: login shell.

Chapter 16

Parameters in /etc/default/useradd:

GROUP primary group id (gid)
HOME
INACTIVE (/etc/shadow)
EXPIRE (/etc/shadow)
SHELL
SKEL
CREATE_MAIL

Chapter 16

Parameters in /etc/login.defs:

MAIL_DIR
PASS_MAX_DAYS
PASS_MIN_DAYS
PASS_MIN_LEN
PASS_WARN_AGE
UID_MIN
UID_MAX
GID_MIN
GID_MAX
UMASK

Chapter 16

Fields in /etc/group :

1 2 3 4

dwarfs:x:206:sleepy,grumpy,doc

1: group name
2: group password (rarely used)
3: group id (GID)
4: group members.

Chapter 16

/etc/group

```
steven:x:501:
elvis:x:502:
madonna:x:503:
geeks:x:504:steven,elvis
music:x:505:elvis,steven
```

- user private groups show nothing in field 4
- secondary groups show members in field 4.

Chapter 16

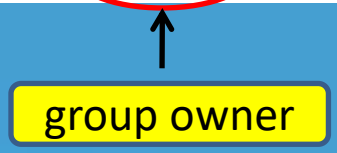
Every user is a member of:

- 1 primary group (by default, their user private group)
- 0 or more secondary groups.

Chapter 16

Every file must have a group owner:

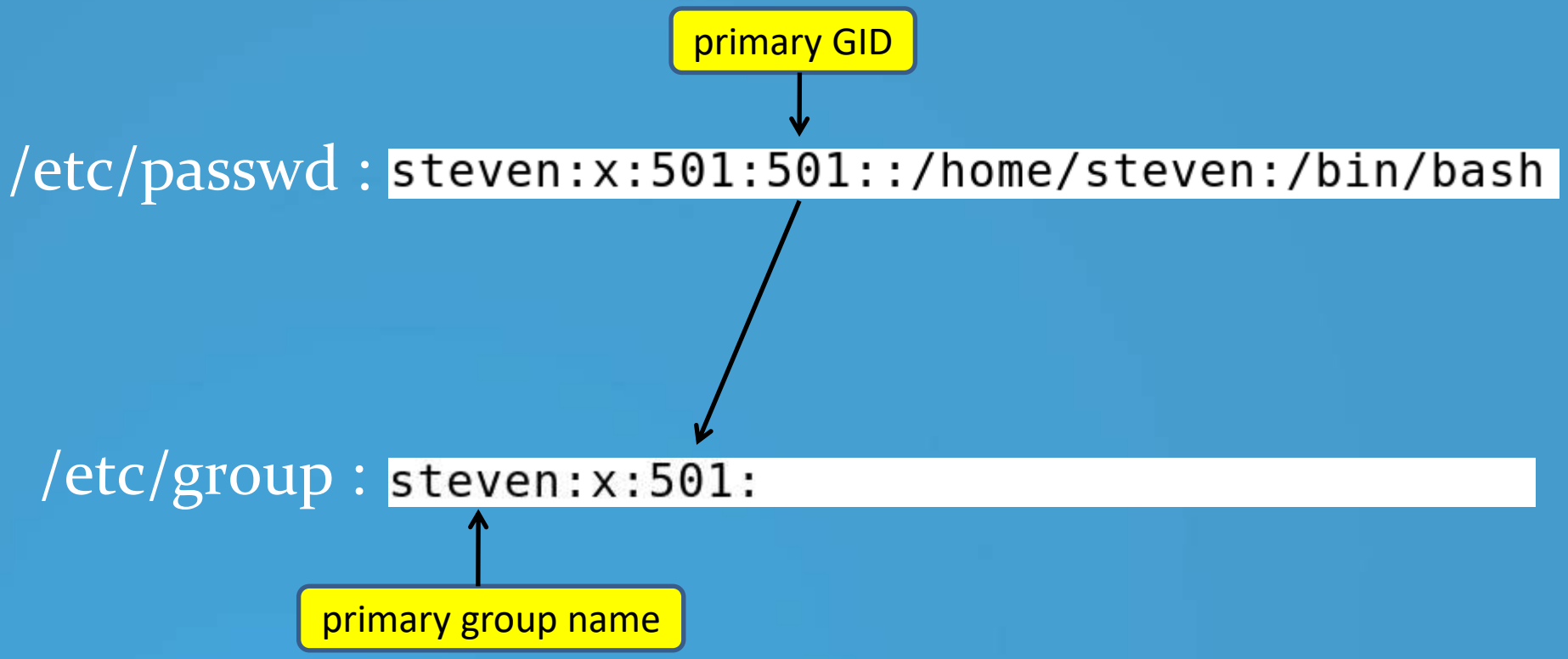
```
[steven@localhost ~]$ echo text > resume.txt
[steven@localhost ~]$ ls -l resume.txt
-rw-rw-r-- 1 steven steven 5 Nov 30 11:47 resume.txt
```



A user's primary group:

- is the group owner of newly created files
- is associated with each user.

Chapter 16


primary GID
/etc/passwd : steven:x:501:501::/home/steven:/bin/bash
/etc/group : steven:x:501:
primary group name

Chapter 16

User info commands:

- `id`: prints user info, including group membershipts
- `groups`: outputs group memberships
- `whoami`: current user's name

Who's logged in?

- users
- w
- who

chapter 16

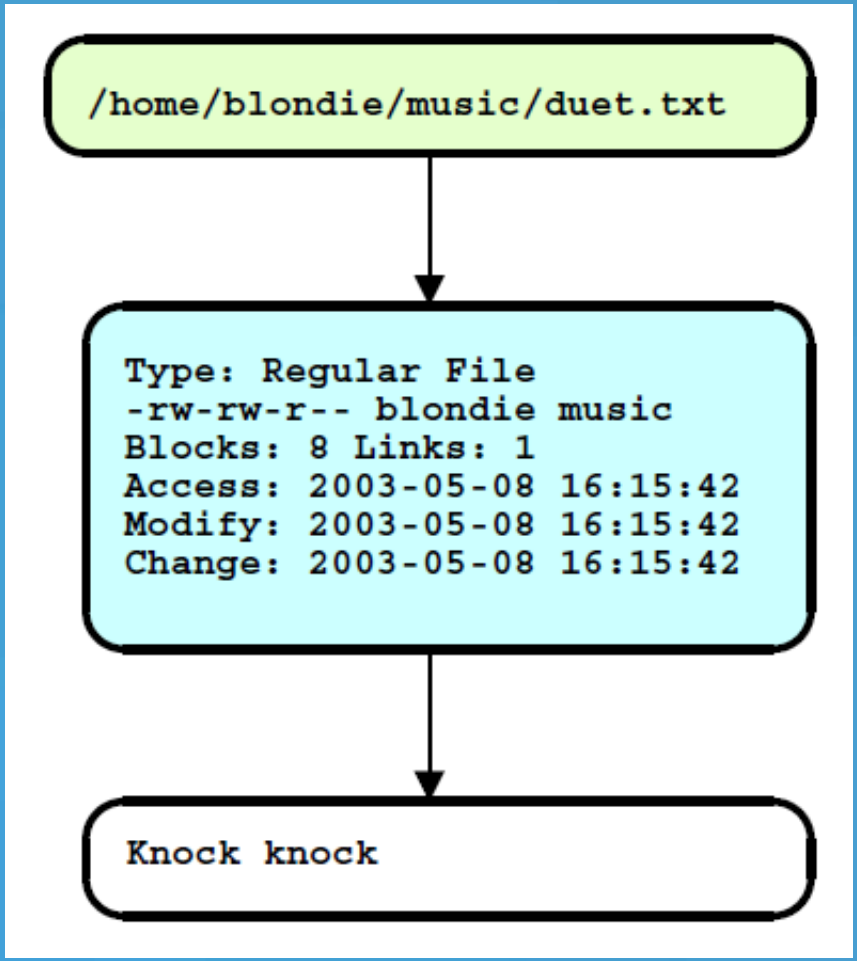
#chage userName

Options:

- -d (YYYY-MM-DD; date password last changed)
- -m (min. # of days between changes)
- -M (max. # of days password is valid)
- -W (# of days before expiration to warn user)
- -I (# of days after expiration before locking account)
- -E (YYYY-MM-DD; account expiration date).

Links

Blondie's regular file:



Links

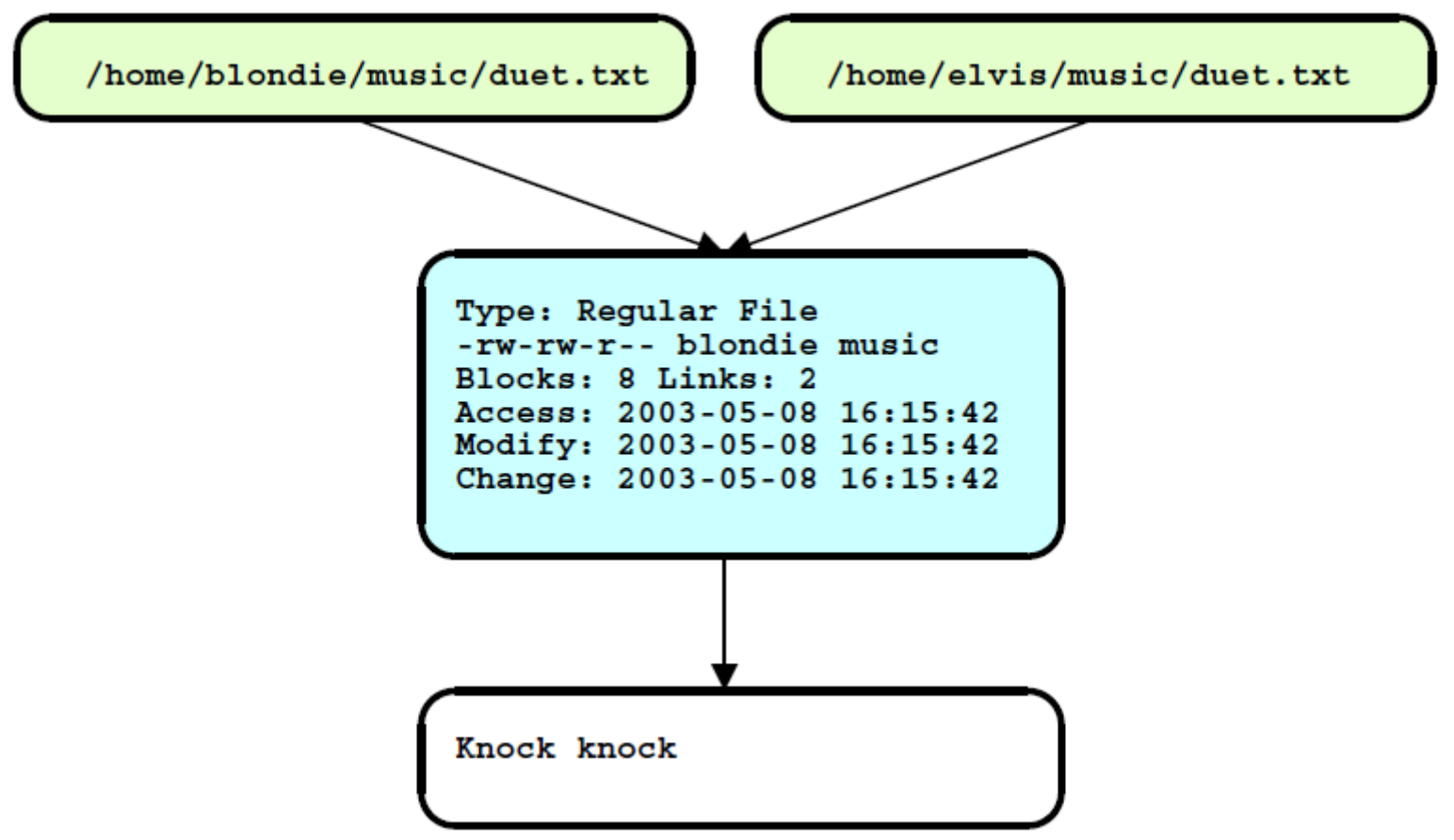
To create a hard link:

- create 2 collaboration directories (ex: ~/music)
- chgrp the 2 directories (ex: chgrp music music/)
- create the file to be shared (ex: ~/music/duet.txt)
- chgrp the shared file
  (ex: chgrp music music/duet.txt)

Ex: ln ~blondie/music/duet.txt ~elvis/music/duet.txt

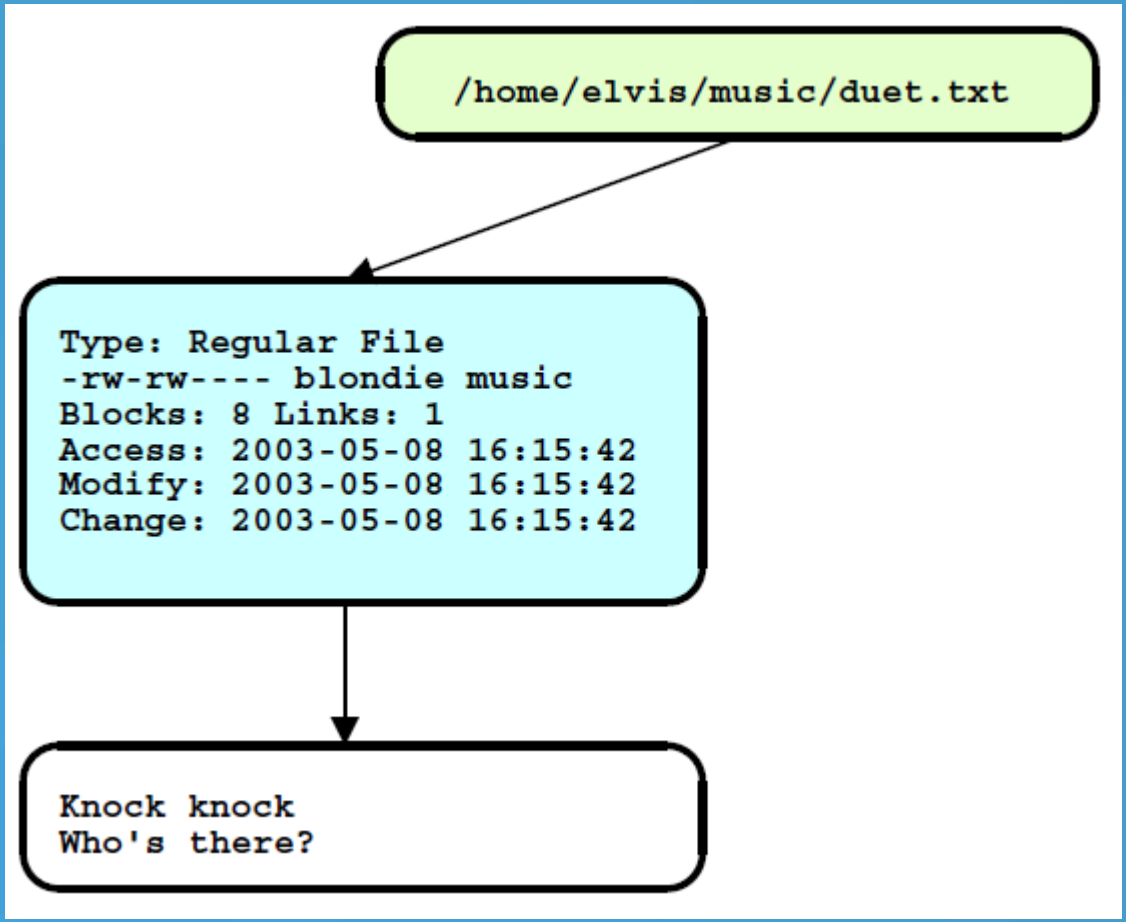
Links

After creating a hard link to Blondie's file:



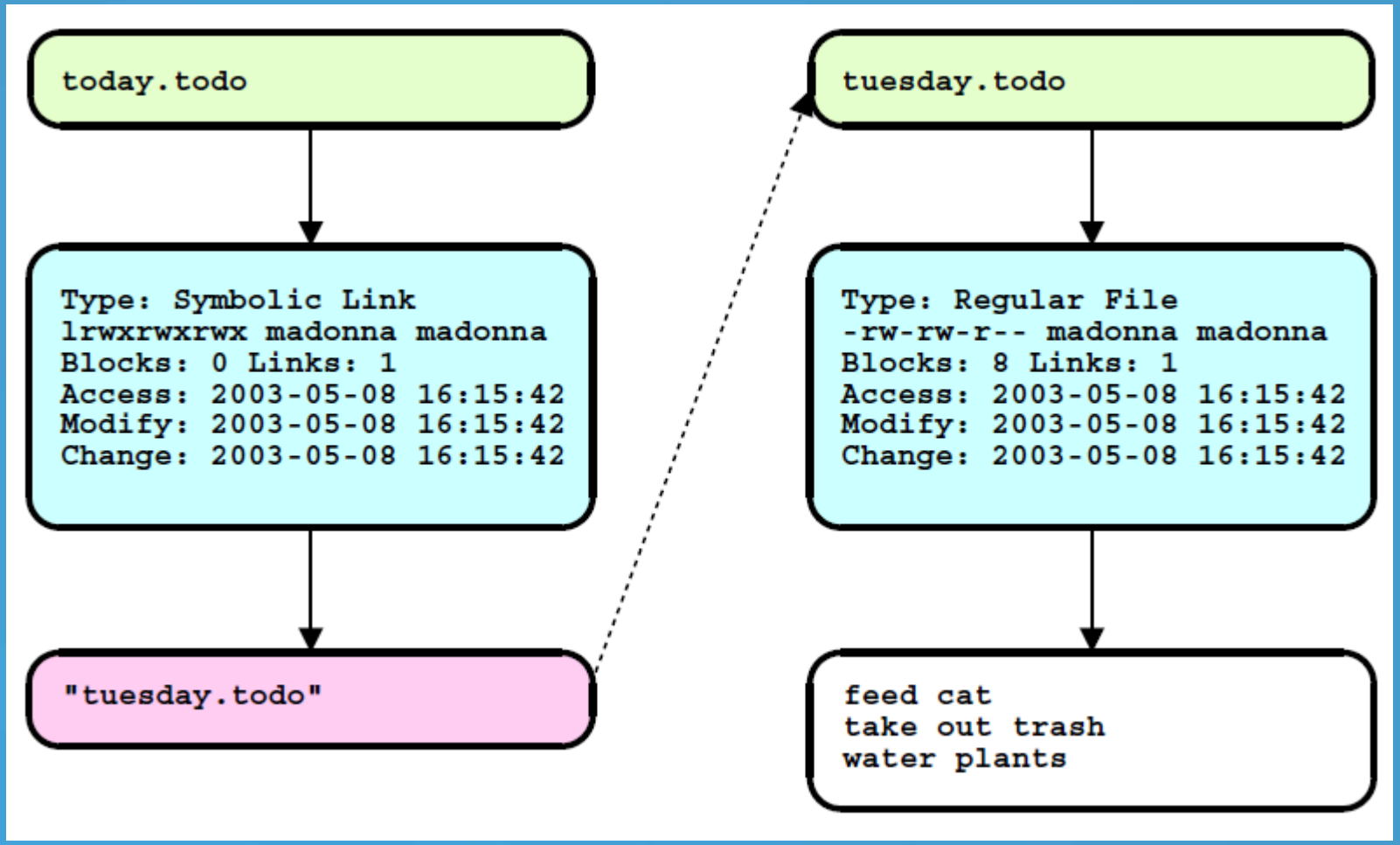
Links

Hard link after removing Blondie's file:



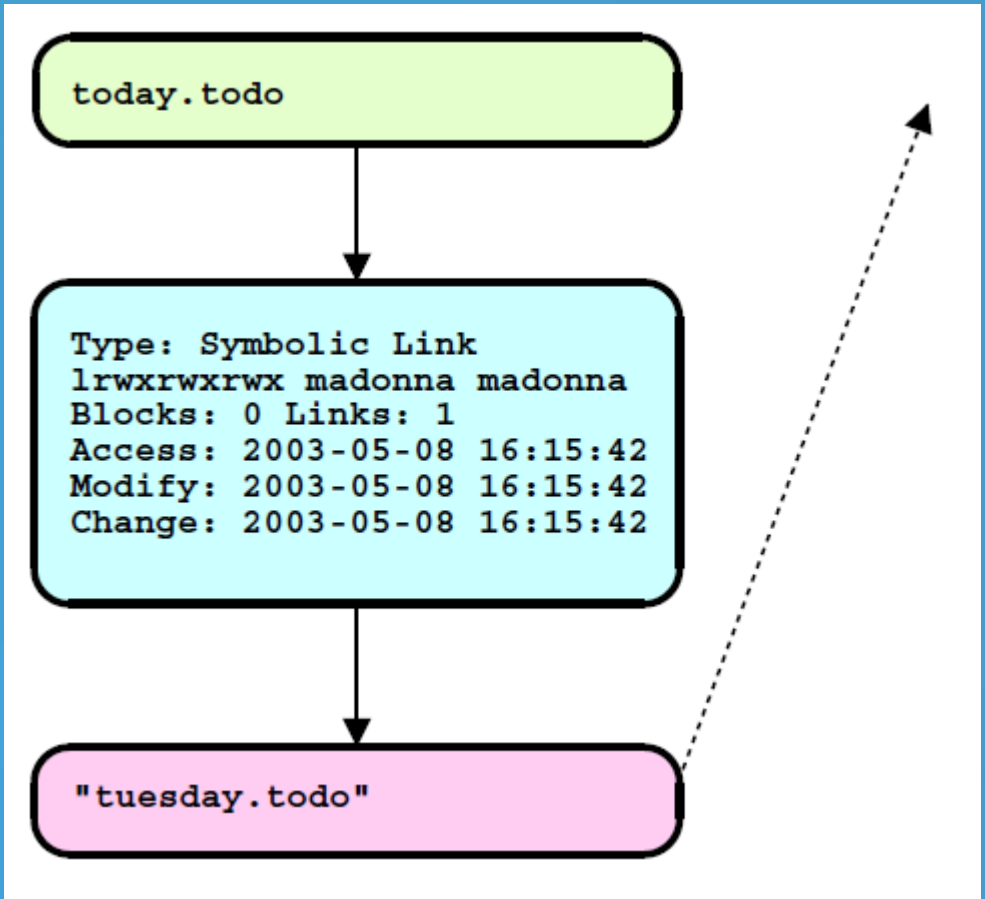
Links

Soft links:



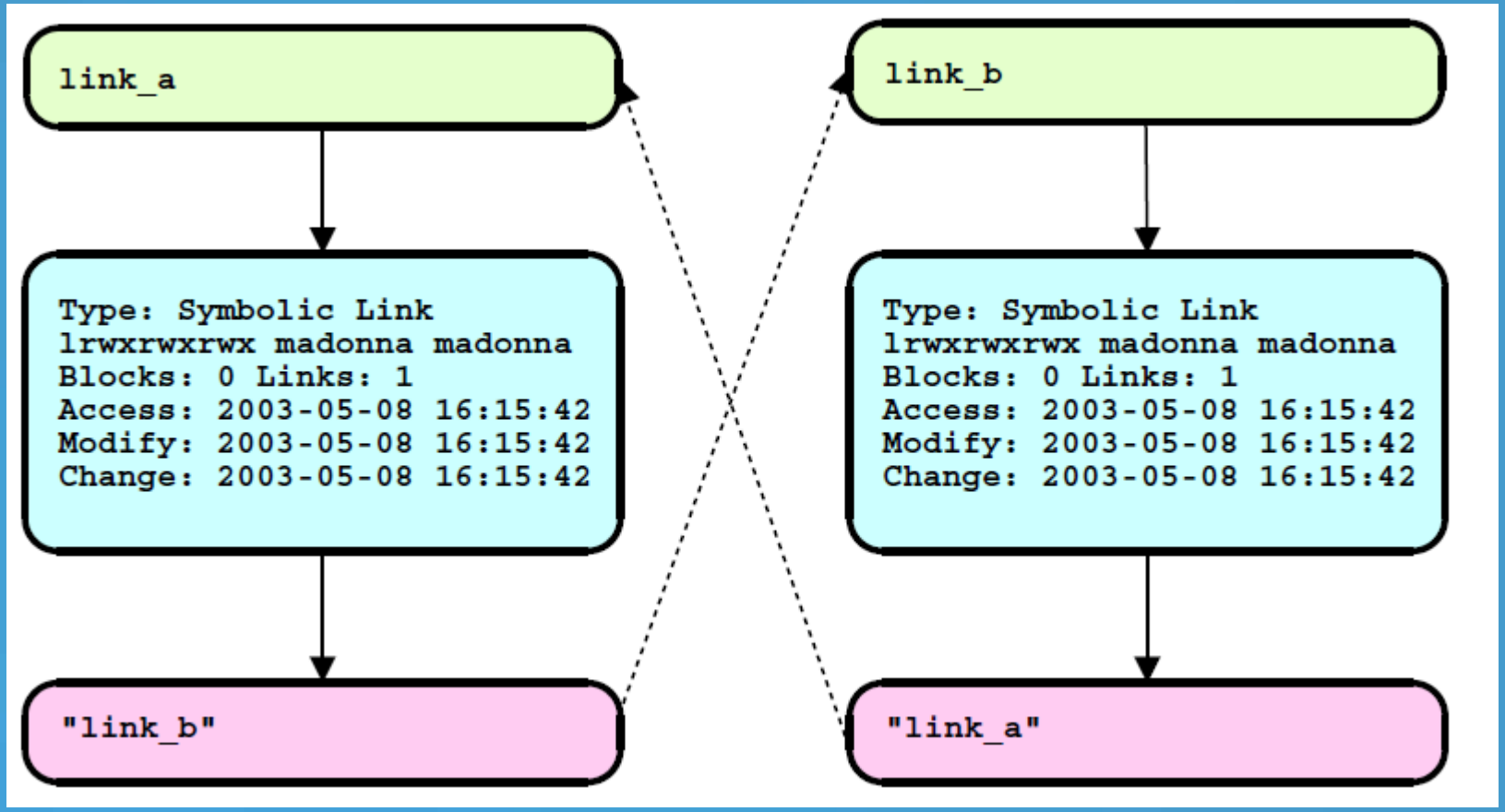
Links

Dangling link:



Links

Recursive link:



Links

You must use a **soft** link to refer to:

- a directory
- a file in a different filesystem (partition).

Ex: ln -s /tmp/work
(creates ./work -> /tmp/work)
Ex: ln -s /tmp/work ~steven
(creates ~steven/work -> /tmp/work)

Links

To remove a **soft** link named mylink:

```
rm  mylink
```

Note that a symbolic link is indicated by an `l`, but a hard link is indicated by a –.