# SECURITY (COMP0141):
# ACCESS CONTROL

# ACCESS CONTROL

`uname,passwd`

“give me Bob’s file”

still need to ensure **access control**

“open Bob’s unit”

define

How to ~~design~~ a secure system?

one that meets a specific security policy

How to define a security policy?

use threat model and build policy to address it

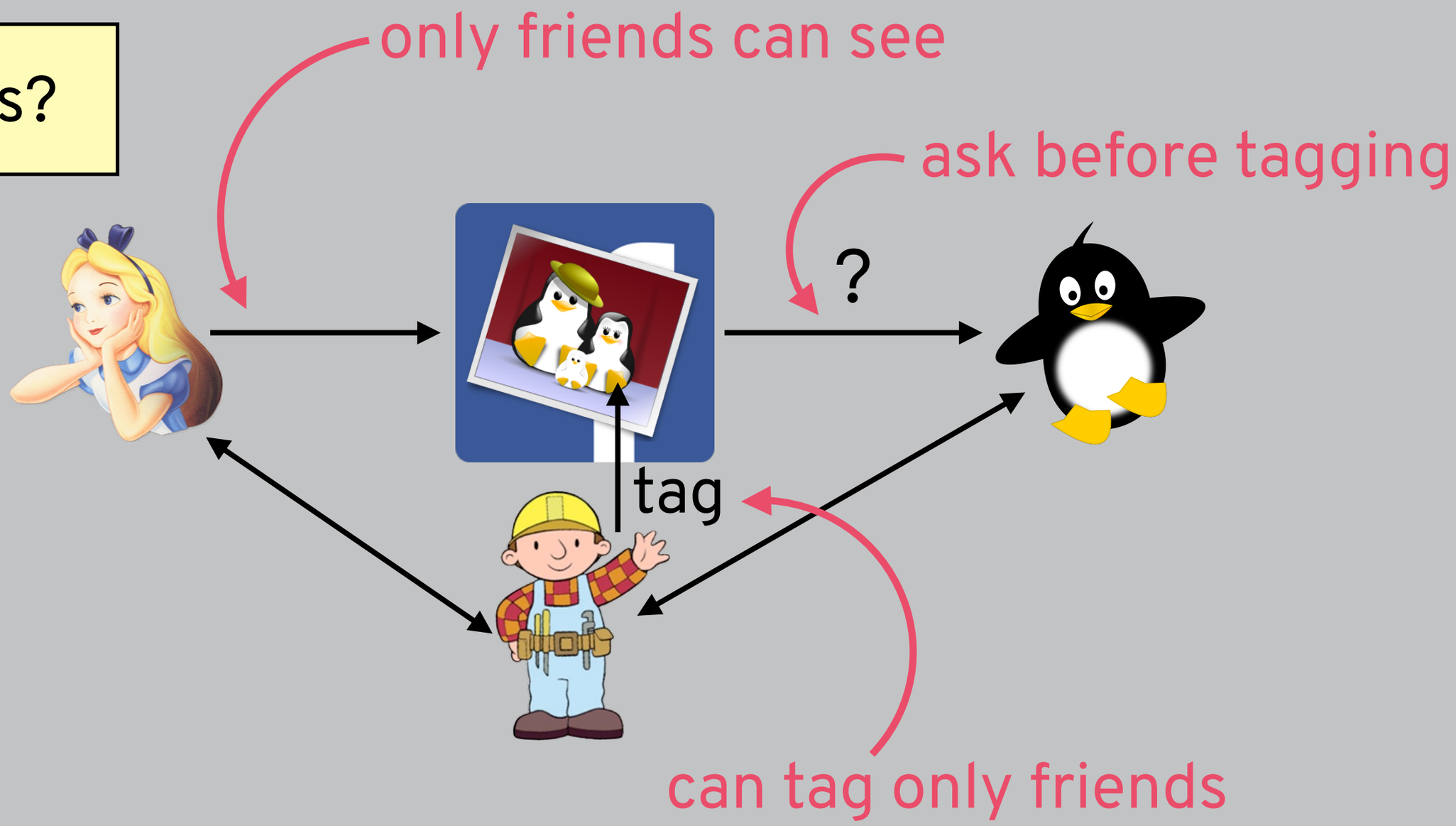
# ACCESS CONTROL

**Access control** is the ability of one entity to permit or deny the use of a particular resource to another

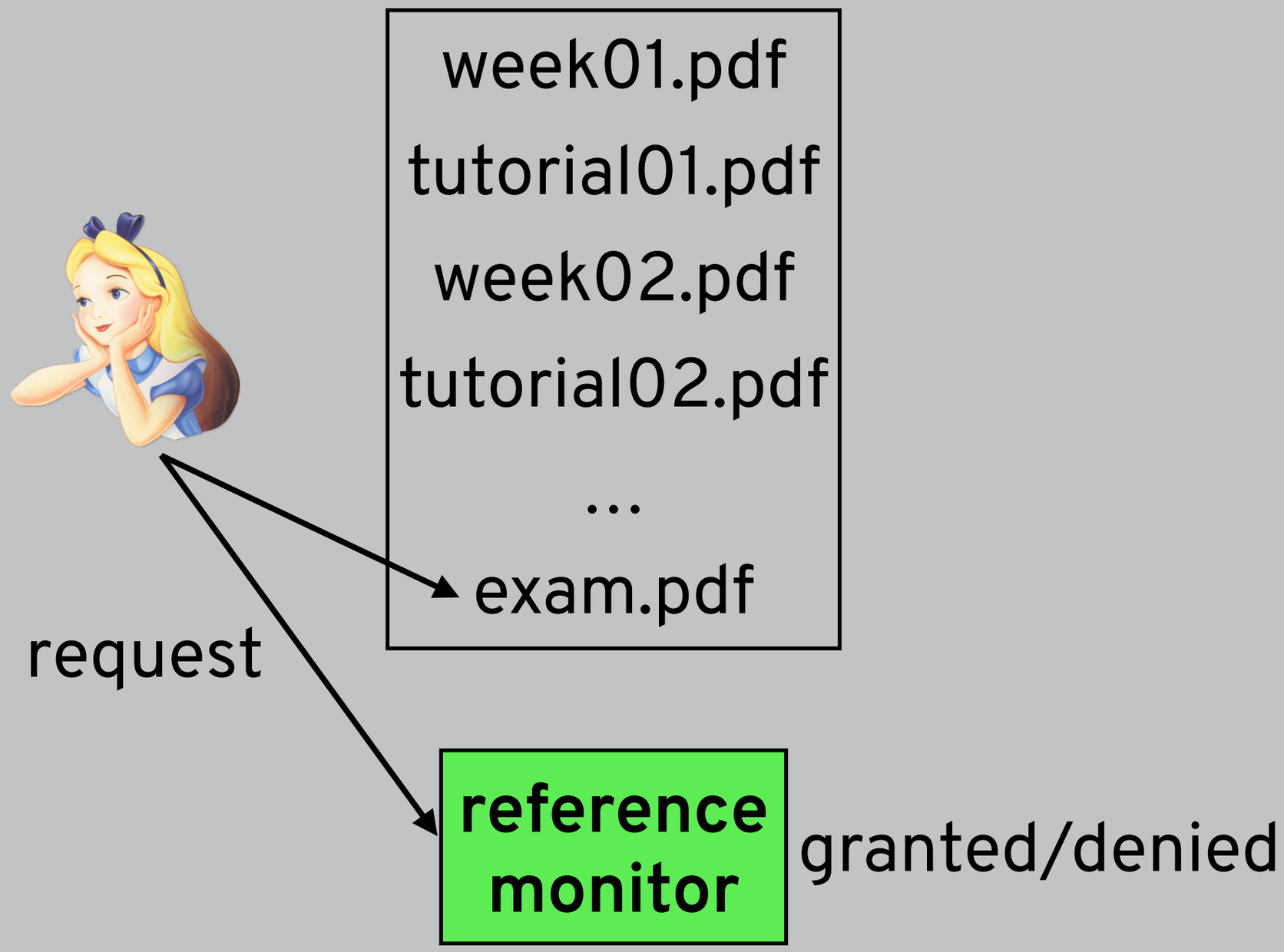
Informal: “We don’t want people wandering in off the street”
Formal: “Only UCL staff and students can enter that area”

Authentication is already a (coarse) form of access control

# EXAMPLE: SOCIAL NETWORKS

# ACCESS RIGHTS



# TYPES OF FILE ACCESSES

**subjects** (s)

**objects** (o)

**access rights** (r/p)

| | non-ALT | ALT |
|---|---|---|
| non-OBS | execute | append |
| OBS | read | write |

**Subjects** are the users of the system

**Objects** are the different files

**Access rights:** execute, read, write, append (some combination of ALTeration and OBServation)

# ACCESS CONTROL MATRIX

S: Alice, Bob

O: cw01,cw02, cw03

R: read, write

| | cw01 | cw02 | cw03 |
|---|---|---|---|
| Alice | read<br>write | | read |
| Bob | | read<br>write | read<br>write |

can Alice read cw01?

can Bob write cw01?

# ACCESS RIGHTS

cw01.pdf
tutorial01.pdf
cw02.pdf
tutorial02.pdf
...
exam.pdf

| | cw01 | cw02 | cw03 |
|---|---|---|---|
| Alice | read<br>write | | read |
| Bob | | read<br>write | read<br>write |

1 request

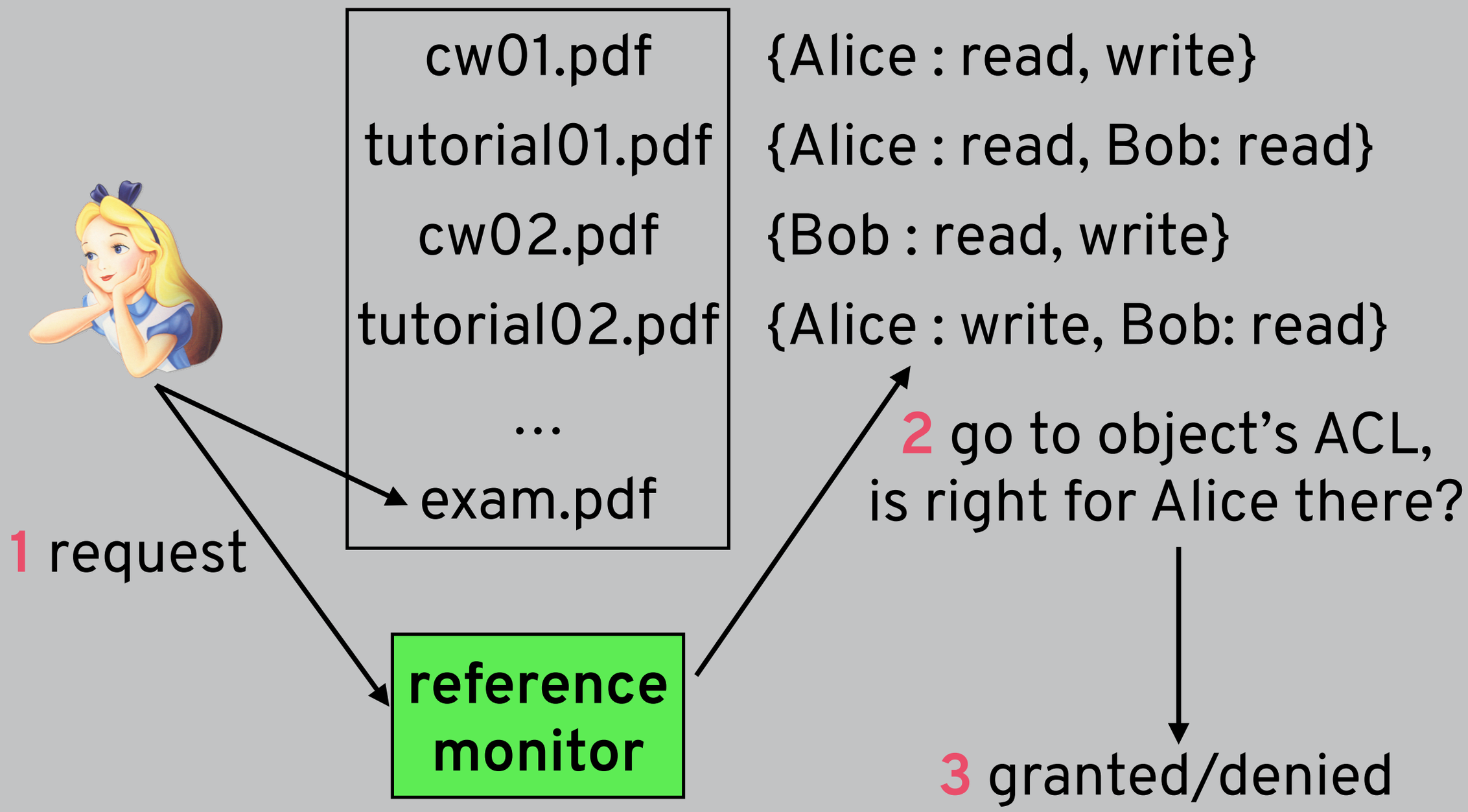
2 right in (Alice,object)?

**reference monitor**

3 granted/denied

# ACCESS CONTROL LIST

**How is this actually implemented?**



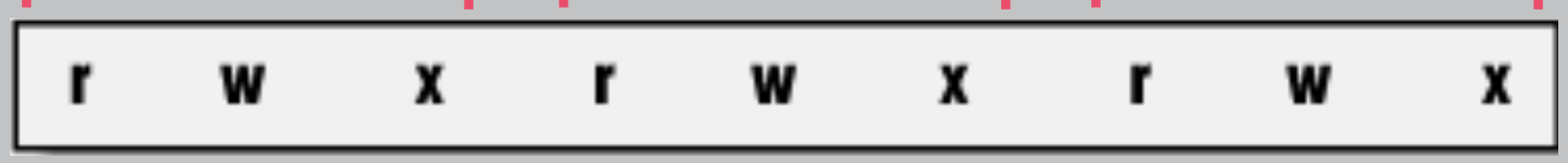
# UNIX PERMISSIONS

cw01.pdf
tutorial01.pdf
cw02.pdf
tutorial02.pdf
...
exam.pdf

can file owner read (**r**), write (**w**), execute (**x**)?

can **group member** r, w, x?

can **anyone** r, w, x?

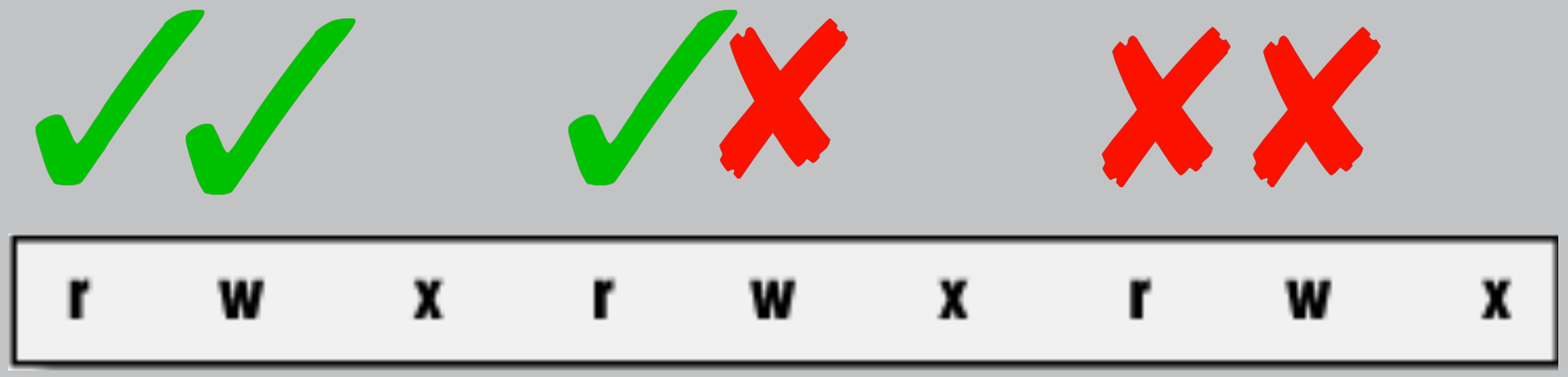


# UNIX PERMISSIONS

owner
module

cw01.pdf
tutorial01.pdf
cw02.pdf
tutorial02.pdf
...
exam.pdf

module

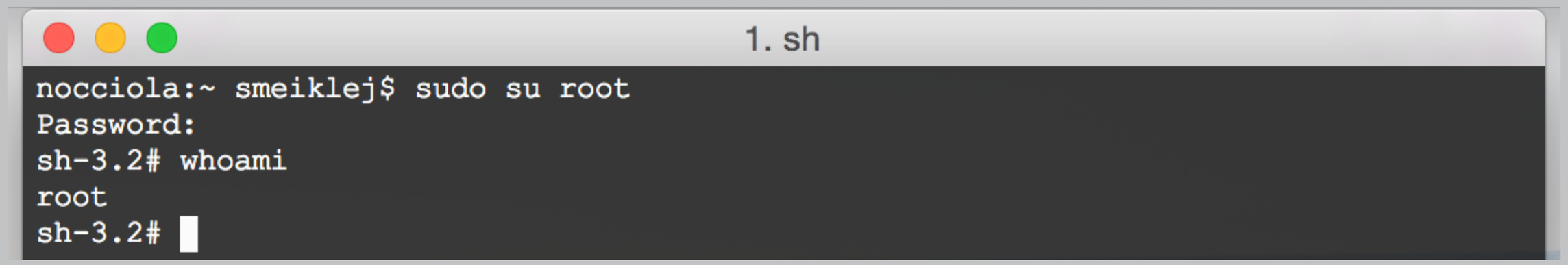


# UNIX PERMISSIONS: DEMO

1. bash

```
nocciola:~ smeiklej$ ls -l
total 0
drwx------   3 smeiklej  staff   102  4 Feb 12:27 Applications
drwx------+ 11 smeiklej  staff   374 16 Feb 15:01 Desktop
drwx------+  3 smeiklej  staff   102 24 Feb 09:32 Documents
drwx------+  4 smeiklej  staff   136 17 Feb 14:51 Downloads
drwx------@ 12 smeiklej  staff   408 15 Feb 11:59 Dropbox
drwx------@ 48 smeiklej  staff  1632 19 Nov 09:33 Library
drwx------+  3 smeiklej  staff   102 18 Sep 11:17 Movies
drwx------+  3 smeiklej  staff   102 18 Sep 11:17 Music
drwx------+  3 smeiklej  staff   102 18 Sep 11:17 Pictures
drwxr-xr-x+  5 smeiklej  staff   170 18 Sep 11:17 Public
drwx------  11 smeiklej  staff   374 16 Feb 11:53 research
drwx------   3 smeiklej  staff   102  8 Dec 11:27 teaching
nocciola:~ smeiklej$
```

# ROOT USER



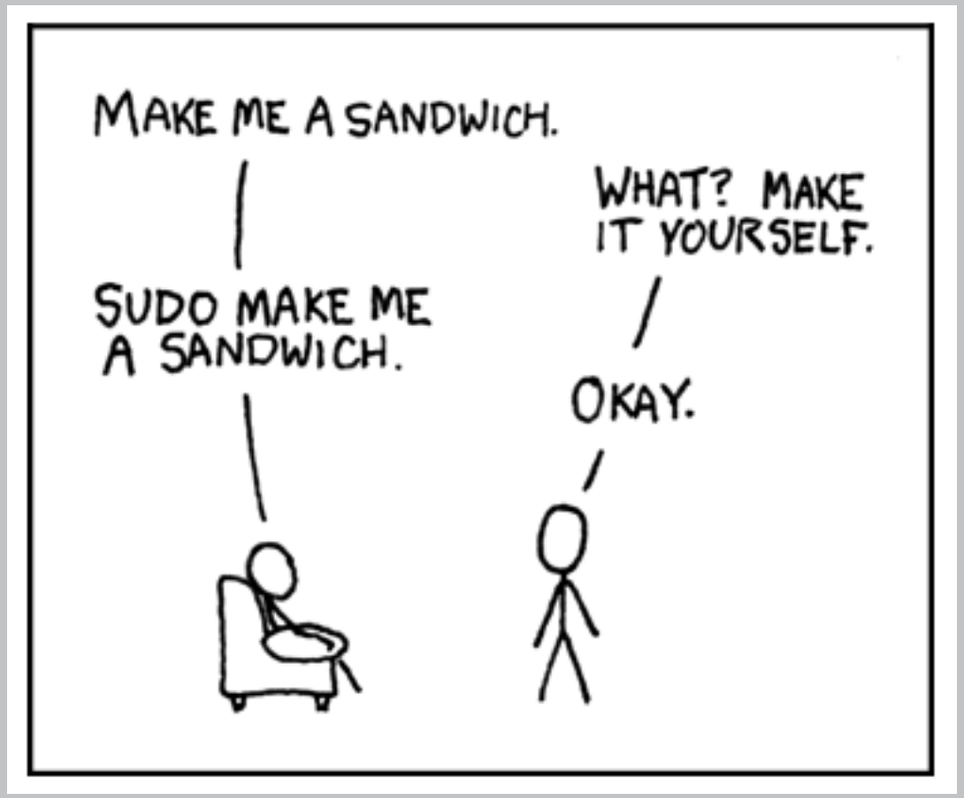

default owner of all system files

protects users from themselves!

especially important in multi-user systems

but what if I want to execute certain tasks?

# SUDO




allows one user to temporarily run things with privileges of another (often `root`)

accountability: `sudo` usage is logged

# PERMISSIONS FOR DIRECTORIES

cw01.pdf
tutorial01.pdf
cw02.pdf
tutorial02.pdf
...
exam.pdf

read = list contents
execute = traverse
write = **modify** files

can file owner read (**r**), write (**w**), execute (**x**)?

can **group member** r, w, x?

can **anyone** r, w, x?

| r | w | x | r | w | x | r | w | x |
|---|---|---|---|---|---|---|---|---|

# STICKY BIT

Can do this even if you don't have write permissions on the individual files!

The **sticky bit** (T) for a directory changes write privileges, can rename or delete files only if you are the owner (or `root`)

read = list contents
execute = traverse
write = **modify** files
= create, rename, or delete

```
nocciola:~ smeiklej$ chmod 1700 research
nocciola:~ smeiklej$ ls -l
total 0
drwx------+ 10 smeiklej  staff   320 12 Mar 16:29 Desktop
drwx------+  4 smeiklej  staff   128  6 Mar  2015 Documents
drwx------+  4 smeiklej  staff   128 13 Mar 09:30 Downloads
drwx------@ 18 smeiklej  staff   576 11 Mar 21:18 Dropbox
drwx------@ 74 smeiklej  staff  2368  5 Nov 11:59 Library
drwx------+  3 smeiklej  staff    96 18 Sep  2014 Movies
drwx------+  6 smeiklej  staff   192 20 Nov  2017 Music
drwx------+  6 smeiklej  staff   192 21 Jun  2017 Pictures
drwxr-xr-x+  5 smeiklej  staff   160 18 Sep  2014 Public
drwx-----T  21 smeiklej  staff   672 17 Dec 10:17 research
drwx------   8 smeiklej  staff   256 20 Dec 13:58 teaching
drwx------   6 smeiklej  staff   192  5 Jan 21:11 writing
```

# UNIX PERMISSIONS: POP QUIZ!

| permissions | owner | group | filename |
|---|---|---|---|
| rwx-----x | bob | eng | week01.pdf |
| rwxrwxrwx | bob | eng | cw01.pdf |
| rwx--x--x | alice | alice | week02.pdf |
| rw-r----- | alice | cs | cw02.pdf |
| rw-r--r-- | bob | cs | week03.pdf |
| rw--wxr-- | root | cs | exam.pdf |

alice,cs

which files can Alice write?

# DESIGN PRINCIPLES

**Least privilege**

Separation of responsibilities

Complete mediation

Fail-safe default

Defence in depth

Open design

Psychological acceptability

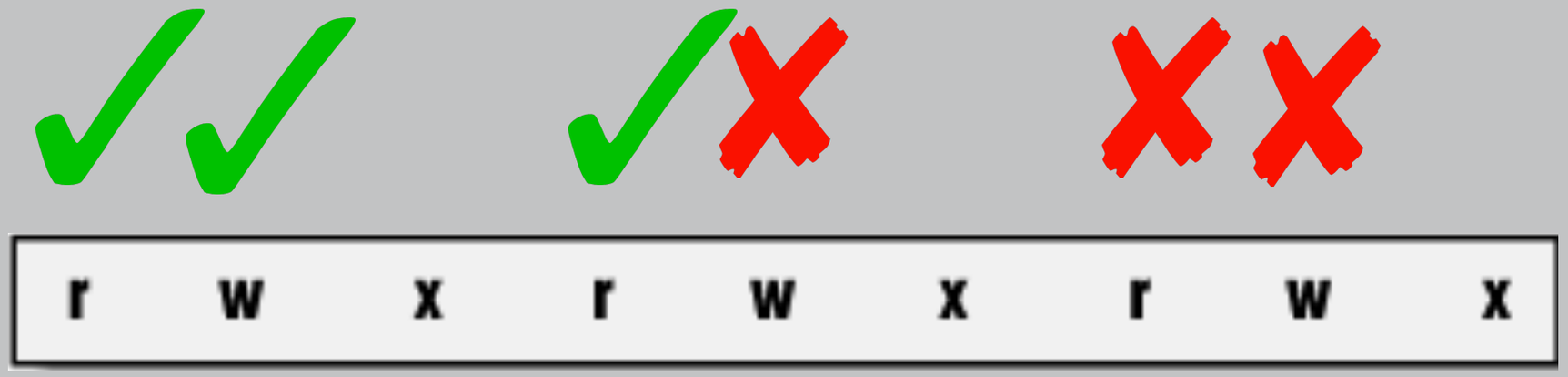
Economy of mechanisms

# LEAST PRIVILEGE

owner
module

cw01.pdf
tutorial01.pdf
cw02.pdf
tutorial02.pdf
...
exam.pdf

module



# DESIGN PRINCIPLES

Least privilege

Separation of responsibilities

**Complete mediation**

Fail-safe default

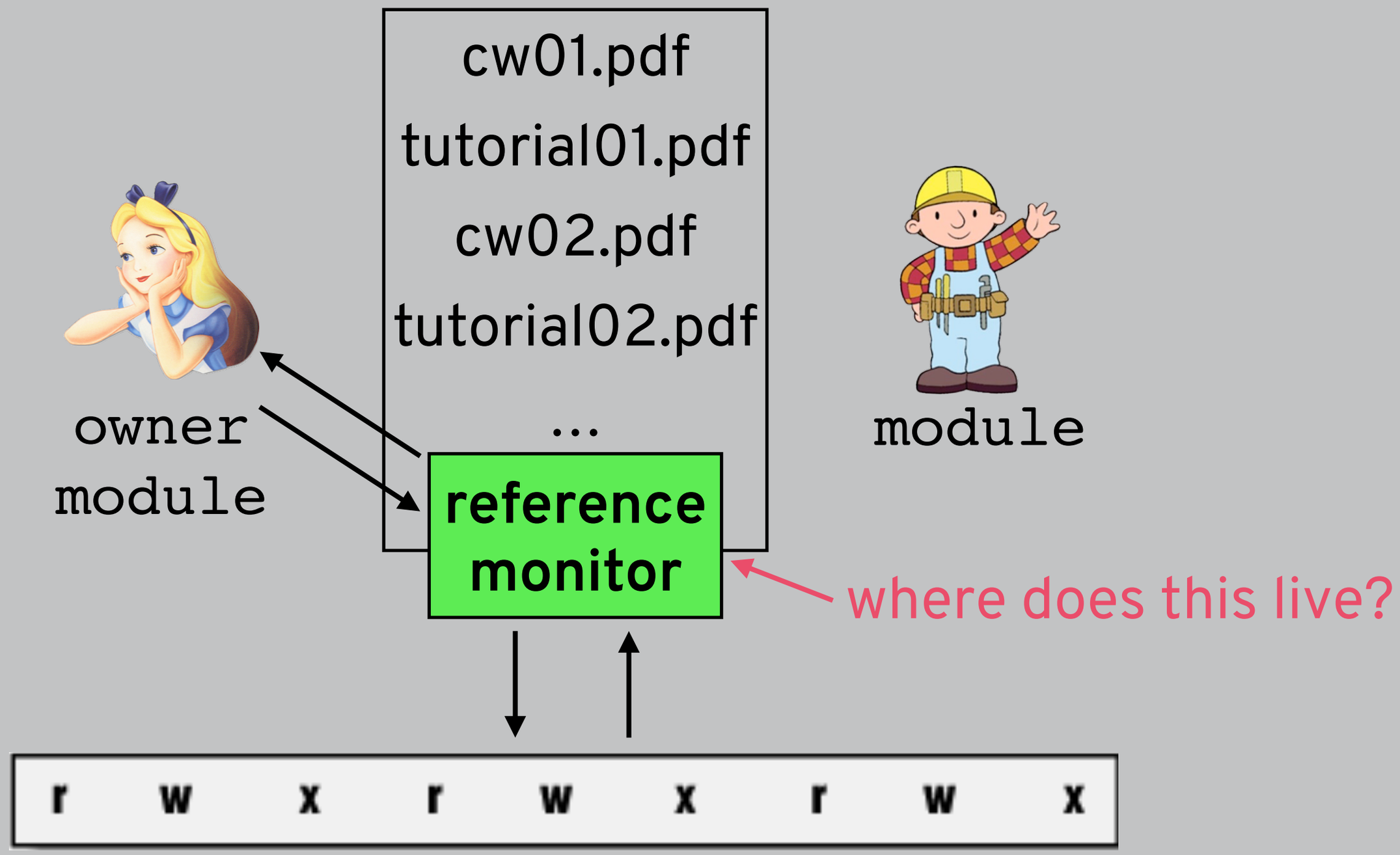
Defence in depth

Open design

Psychological acceptability

Economy of mechanisms

# COMPLETE MEDIATION



# TRUSTED COMPUTING BASE (TCB)

**Trusted computing base (TCB)** refers to every component of the system upon which the security policy relies (could be hardware, software, etc.)

In other words, if something goes wrong then the security policy may be violated

This needs to be kept small!

This is an example of economy of mechanisms (could just think of entire system as TCB but this is very unrealistic)

# DESIGN PRINCIPLES

Least privilege

Separation of responsibilities

Complete mediation

**Fail-safe default**

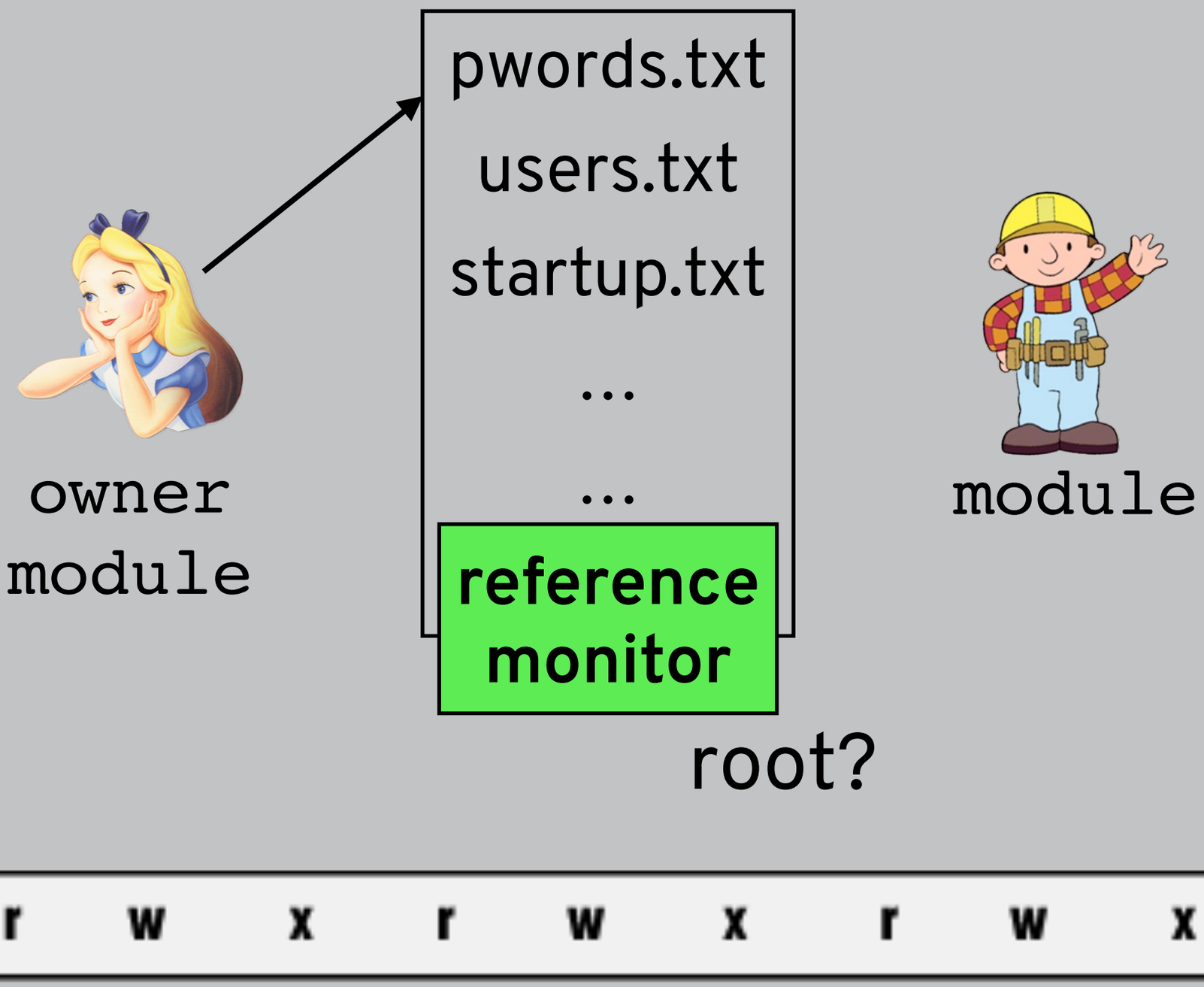
Defence in depth

Open design

Psychological acceptability

Economy of mechanisms

# FAIL-SAFE DEFAULT



# DESIGN PRINCIPLES

Least privilege

Separation of responsibilities

Complete mediation

Fail-safe default

Defence in depth

**Open design**

**Psychological acceptability**

**Economy of mechanisms**

# NEXT TIME

**subjects** (s)

**objects** (o)

**access rights** (r/p)

| | non-ALT | ALT |
|---|---|---|
| non-OBS | execute | append |
| OBS | read | write |

**Subjects** are the users and processes of the system

**Objects** are the different files

**Access rights:** execute, read, write, append (some combination of ALTeration and OBServation)