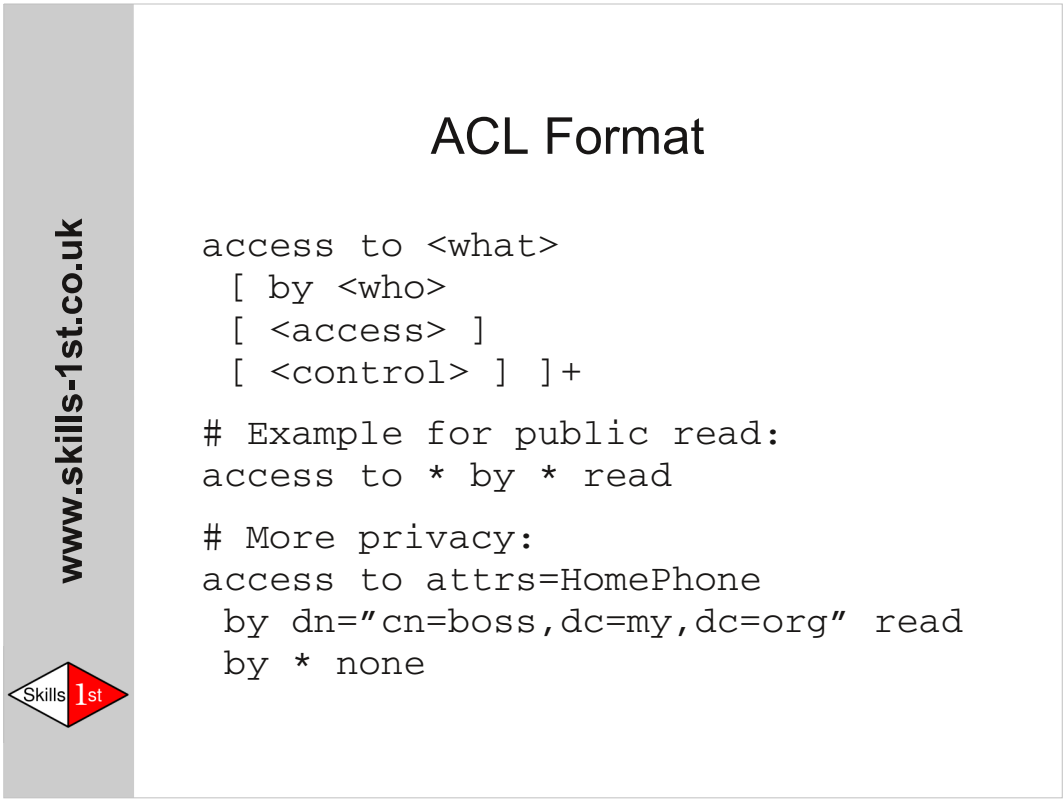


Each *access* statement is nominally on a single line. It is often convenient to split it over several lines by indenting the continuation lines. Be very careful about the placement of comments if you do this.

Specifying <what>

- Entry specifiers
  - DNs: exact, onelevel, subtree, children, regex
  - LDAP filter to match entries
- Attribute specifiers
  - Attribute list
  - Object class
  - Specific values of an attribute

Examples of <what>

dn.exact=”uid=af27,dc=people,dc=example,dc=org"
dn.children=”dc=people,dc=example,dc=org"
dn.regex=”uid=([^,]+),ou=([^,]+),dc=example,dc=org"
filter=”(objectclass=inetorgperson)”

attrs=userPassword
attrs=homePhone,mobilePhone
attrs=@mySpecialObjectClass
attrs=telephoneNumber val.regex=”^+44”

Specifying <who>

- Special keywords
  - *
  - anonymous
  - users
  - self
- dn (exact, one, sub, children, regex...)
- group
- Many more...

Examples of 'by <who>'

by *
by self
by dn.exact=”cn=Helpdesk,dc=example,dc=org”
by dn.subtree=”ou=MyDept,dc=example,dc=org”
by group=”cn=Helpdesk,ou=groups,dc=example,dc=org”
by dn.exact,expand=”uid=admin,ou=$2,dc=example,dc=org”
by ssf=56 [Require at least 56 bit encryption]

Specifying <access>

- By level:
  - `none disclose auth compare search read {write|add|delete} manage`
- By privilege:
  - `{=|+|-}`
  - `{0|d|x|c|s|r|{w|a|z}|m}+`

Each *level* includes the permissions granted by the preceding level. Each *privilege* can be controlled independently. Levels are implemented as collections of privileges.

0 (zero)		Used to remove all privileges, as '=0'
d		Disclose
x		Authentication
c		Compare
s		Search
r		Read
w		Write
	a	Add
	z	Delete
m		Manage

ACL Controls

- Control the flow of execution
  - Stop – execution stops immediately
  - Continue – consider the following <*who*> clauses in this *access* statement
  - Break – skip to the next *access* statement
- e.g. Give global update power in an early rule:

```
access to *
 by dn="cn=mgr,dc=example,dc=org" write
 by * break
```

www.skills-1st.co.uk

Typical ACL

- Allow users to change passwords
- Allow public read

```
access to attrs="userPassword"
        by self =w
        by * auth
access to * by * read
```

We do not want passwords to be readable so we handle them first. “by self =w” uses the privilege form to give the entry owner write access without also allowing them to read the password. “by * auth” allows other users (including anon) to authenticate by supplying a password. Access to all other attributes is controlled by the second statement, which just grants global read permission.

www.skills-1st.co.uk

Helpdesk access

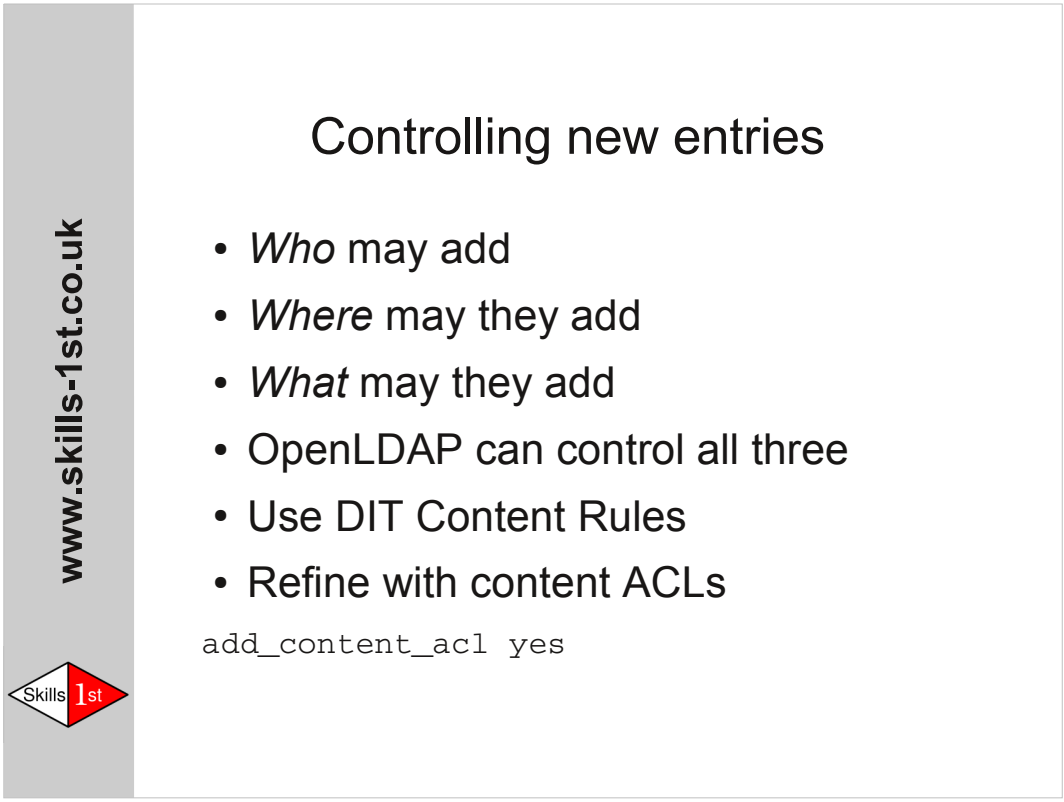
```
access to attrs="userPassword"
 by self =w
 by group="cn=helpdesk,ou=groups,..." =w
 by * auth

access to * by * read
```

- Give access to *groups,* not individuals

The Helpdesk group used here is assumed to be of class *groupOfNames* where the members are defined by DN values of the *member* attribute. *GroupOfNames* is a difficult class to use as the set of members may not be empty, so many sites define their own group class. To use such a class in ACLs it must be explicitly referenced:

by group/exampleGroup/member="cn=Helpdesk,dc=...."




```
ditcontentrule ( 2.16.840.1.113730.3.2.2
        NAME 'dcrPerson'
        DESC 'Limit aux classes in
              inetOrgPerson entries'
        )
```

The OID belongs to the inetOrgPerson objectClass. In this form the rule simply bans all auxiliary objectClasses. It can optionally list acceptable classes, add to the MUST and MAY attribute lists, and ban other attributes that would otherwise be permitted.
Note that a DIT Content Rule applies to the whole server: it cannot currently be restricted to a specific part of the DIT.

www.skills-1st.co.uk

Ex-directory rules

- Use an attribute value to trigger rules: `exampleVisibility=users`
- Users can control own visibility

```
access to filter="(exampleVisibility=internet)"
        attrs="entry"
        by * read
access to filter="(exampleVisibility=users)"
        attrs="entry"
        by users read
        by * none
access to attrs="entry"
        by self read
        by * none
```

The value of the exampleVisibility attribute controls who can see the entry.
The third clause controls access if the value does not match the first two rules – this includes the case of the attribute not being set.

www.skills-1st.co.uk

Attribute visibility

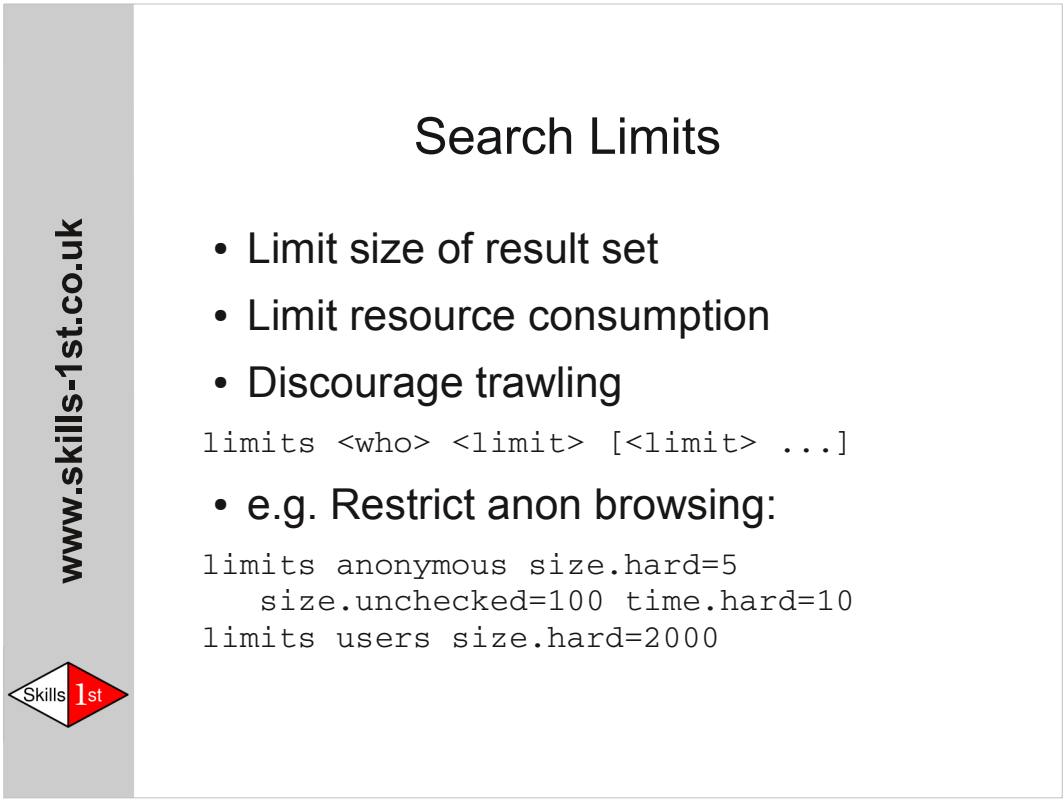
- Use an objectClass to define set of attributes

```
objectclass ( 1.2.826.0.1.3458854.666.3.1
        NAME 'attrsetAnonVisible'
        DESC 'Attributes visible to anonymous users'
        AUXILIARY
        MAY ( objectclass $ cn $ sn $ mail )
        )
access to filter="(objectclass=person)"
        attrs="@attrsetAnonVisible"
        by * +rsc break
access to filter="(objectclass=person)"
        attrs="userPassword"
        by anonymous +x break
        by * break
access to * by * stop
```

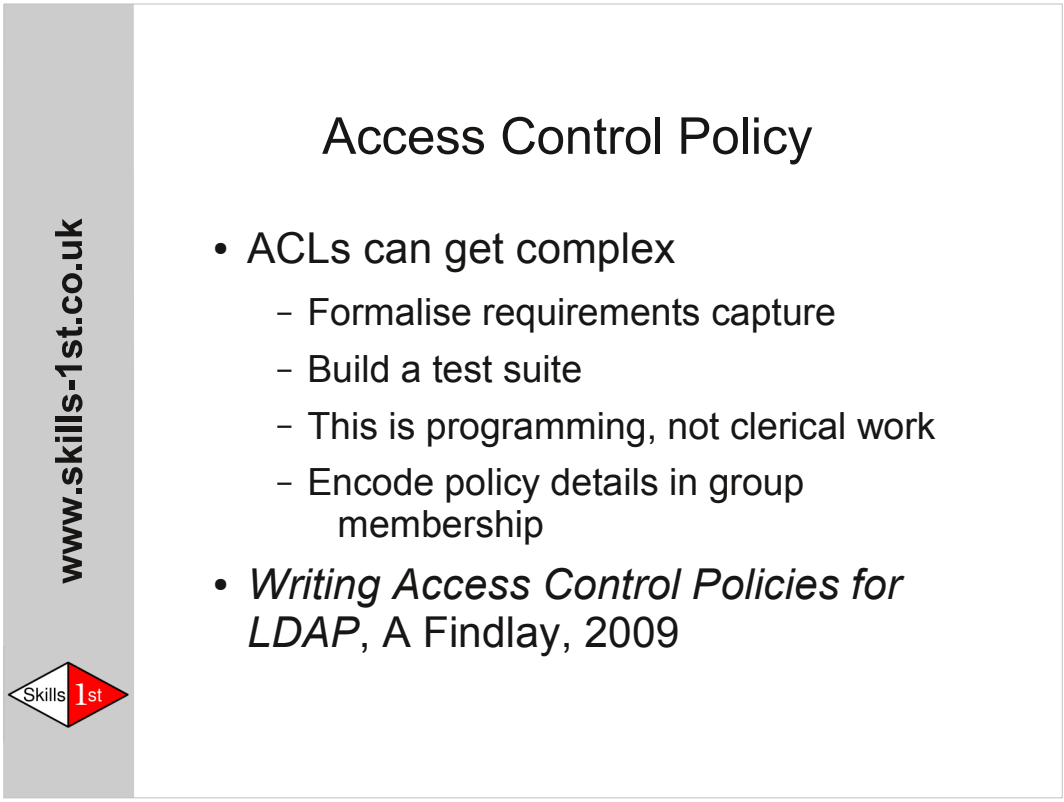
Here we use an objectclass to define a set of attributes.
We can then refer to it in access control rules. The alternative is to list the attributes explicitly in each rule.

The ACL here works by accumulating permissions using the '+priv' syntax. The final line causes evaluation to stop so the accumulated privileges will then be used.

These rules could follow the 'entry' rules from the previous slide, as they act on named attributes rather than on whole entries.



This form of the *limits* command is per-database.
Hard and soft limits can be set on both time in seconds and number of results to be returned. The 'unchecked' limit requires that the search should produce less than 100 results from the operation of attribute indexes alone: this reduces the resources that can be consumed by inefficient (non-indexed) searches.



Draw the DIT, with example entries of each type.
Ask questions: *What access does user A have to entry P?*
Each answer becomes a test in the test suite.
Build the test suite *first*.
Try to design the ACLs so that routine changes in access privilege can be done by modifying membership of groups.
Avoid routine modification of ACLs.

www.skills-1st.co.uk

Exercise 10

- Apply simple ACLs
- Apply search limits