4/17/70

Specific actions for directories (IP list specifications) (etc.)

I) object creation (different operation for each kind of object, differentiated by a fixed parameter?)

A) non scratch

open directory	(with object creation bit on)
Text name	(use block data xfer?)
data to define object	

B) scratch

open directory	(with object creation bit on)
Text name	(use block data xfer?)
access key number	[for full access entry to object] (data)
data to define object	

II) object access (produces closed version of the object)

open directory	
Text name	(use block data xfer?)
access key	[on object]
C-list index	for return of capability [or use capability return feature?] [probably!]

The directory is scanned by some algorithm for a match with the text name, If no match is found, The action fails, either by error or F-return. I Think F-return is best.

If a match is found, Then The access is authenticated and an option bit field constructed as follows:

If The access key is not null, The access list for the entry is scanned for a matching accesskey number. If not found, then the action fails. (error? F-return?). If found, an option bit field is formed by and-ing The option bits associated with The access key number in The access list with The option bit field of the presented access key.

If The access key is null, and The implicit access bit is on in The capability for the directory then:

if The scratch bit is not on in This entry, Then an option bit field with all options on is formed

If the scratch bit is on, then an option bit field with only the destruction bit on is formed.

If This entry was a soft link entry, the above actions are re-entered with The directory pointed to by the link. The text name to be used is That in The link. The access key to be used has the same number as That in the link, and the option bit field just constructed. [of course an actual access key does not have to be constructed] [the director has, of course, to be opened] [Then closed when done]

Having reached either an ownership entry or a Hard link entry, a capability, representing the closed object is constructed as follows:

The option bit field is that last constructed during the above part of the algorithm,

The type field is that associated with this kind of object. (specified in the ~~of entry~~ object part of the entry.) The data word is the full object part of the entry.

III entry modification

A) adding an access key to the access list

open directory	
text name	(use block data xfer?)
access key	(an object)
access key number	(data)

First an entry is found and authenticated as in object access. [except that softlinks are not allowed] Then, if the access key number is not already in the list, it is placed in the list. finally the option bit field constructed is associated with this access key number in the access list.

B) deleting an access key from an access list

open directory	
text name	(use block data xfer?)
access key	(an object)
access key number	(data)

an entry is found as in A). If the option bit field constructed does not have proper bit on, error. Else the access key number specified is removed from the access list.

Ia) rename an entry

open directory

text name

access key

new text name

as before, the constructed option bit field must have proper bit on.
also, soft links are not followed. (i.e. as in A)

IV) destroy an entry

open directory

text name

access key

find and authenticate an access as in III) A)
If entry destruction bit is on in constructed option bit field, then proceed,
else error.

If Hard link entry, or soft link entry, just remove from the directory
If ownership entry, Then object destruction bit must also be on,
and destroy object and remove entry from Directory.

V) List of kinds of objects that can appear in directories

A description of the contents of the object designation field (and hence the data' field of the ecs capability representing the closed object) will appear in later documents. (As well as specifications for construction as well as implications of destruction)

directories files subprocess descriptors	implemented as lowlevel disk files
access keys	
global ecs object ecs goodie	I have forgotten the new names