1/22/70

· objects in directories

files

directories

access keys

subprocess descriptors ← probably represent as files

ecs goodies

soft links

note on ecs goodie

basic ops ~~to present~~

1. present an ecs goodie and ask for an ecs object in return [one gets the option bits on the goodie anded with the bits with the object?]

2. present an ecs goodie and an ecs object and ask that they be identified [requires a special option bit on the goodie?]

representation of files, directories, accesskeys, ecs goodies
[note: subprocess descriptors later

	directory	ecs
file	disk address + unique name	special type capability data word holds disk address + uname
directory	"	" (different type)
accesskey	a number	special type capability data word holds the # [~~a~~ for uniformity, in preference to using ~~[illegible]~~ the class code mechanism]
ecs goodie	a number + unique name (a slot address of an ecs ~~directory~~ table of the ecs objects then unique name would be checked]	# same as file

directory structure for an entry

name, object, accesskey list

<entry> ::= <name> <object> <accesskeylist>

<accesskeylist> ::= ~~<keynumber> <optionbits>~~ <access> | <accesskeylist><access>

<access> ::= <keynumber><optionbits>

note: associated with each directory will be a special accesskey
That access key will appear on all accesskey lists in the directory
with full option bits.

so a directory is being "held"

see →

each of these actions will cause at least a disk read & maybe a read-write if directory modified

used if multiple actions desired on a directory?

Hold a directory

directory

release a directory

directory

directory operations [by user process]

fetch an entry

params

α [directory ← representation in ecs form
name
access key ← access keys?] multiple?

c-list index for result

result

ecs representation of the object at indicated place in c-list

This should be user code interface ∴ if we blow the command process others can be written

Log on
gives
user process
2 directories
temp
perm
& an accesskey

If more than
one logon on
same "name"
Then different
temp parts
but same perm
|
|
|

Create an entry [i.e. create a disk system object]

params

directory

name

[description of object]

destroy an entry

params

α [directory
name
accesskey

[note: destroys the object]

add access keys to an entry

params

α [directory
name
access key
access key number (datum)
option bit mask } [a list?]

rename an entry (?)

params

α [directory
name
accesskey

delete access keys from an entry

same as add access keys

subprocess descriptor

class code ?

class code of father ?

mapentries — need individual mapentries

- file
- fileaddress
- cm address
- CNT
- RO

compiled mapsize

subprocess fl

entry pnt

c-list — need c-list size entries

calling operations defined by subprocess descriptor

3 ways in which objects at arms might be known

1. Known to constructor of the subprocess descriptor
   e.g. The file containing the code for the subprocess
   The class code of the subprocess

2. Known to constructor of the subprocess
   e.g. (have none) (but clear this is needed)

3. Local to the sub process it~~self~~
   e.g. scratch rw file (local clist)

hmm what directory do we put this in
or is there files not in directories
or can't have user name access because it would contain secret data
at very least, don't want user writing in it.

4. ccs objects
   e.g. standard c-lists (for indirect ref)

#'ed objects could be controlled by access names

3. by # for certain ccs actions