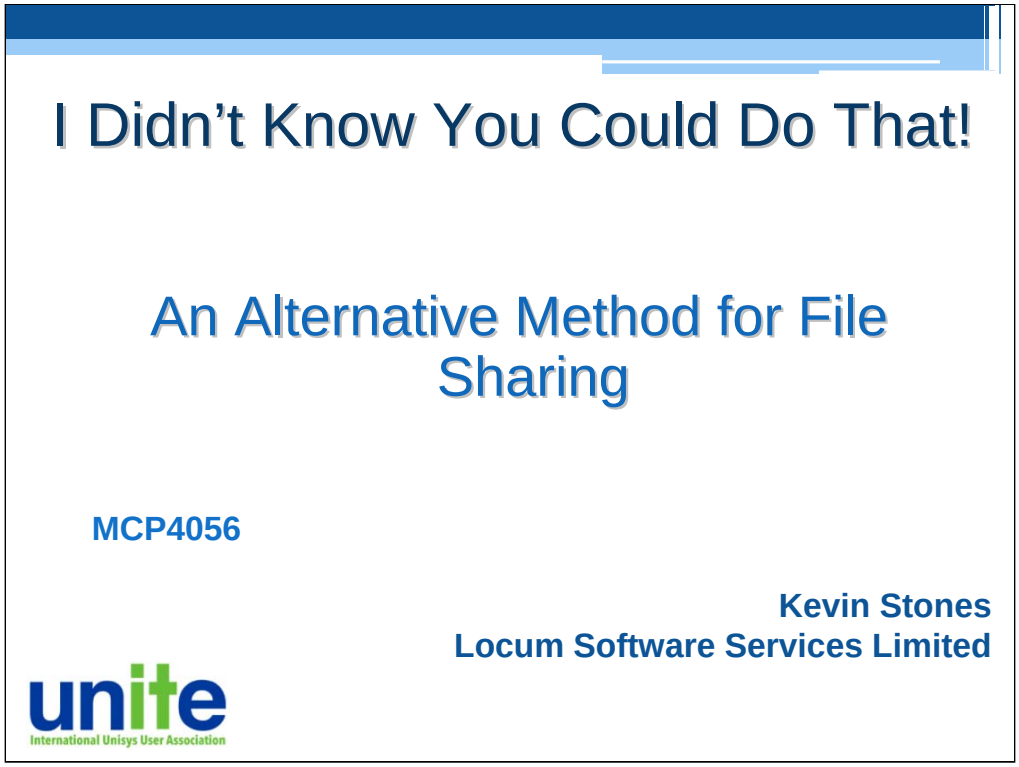
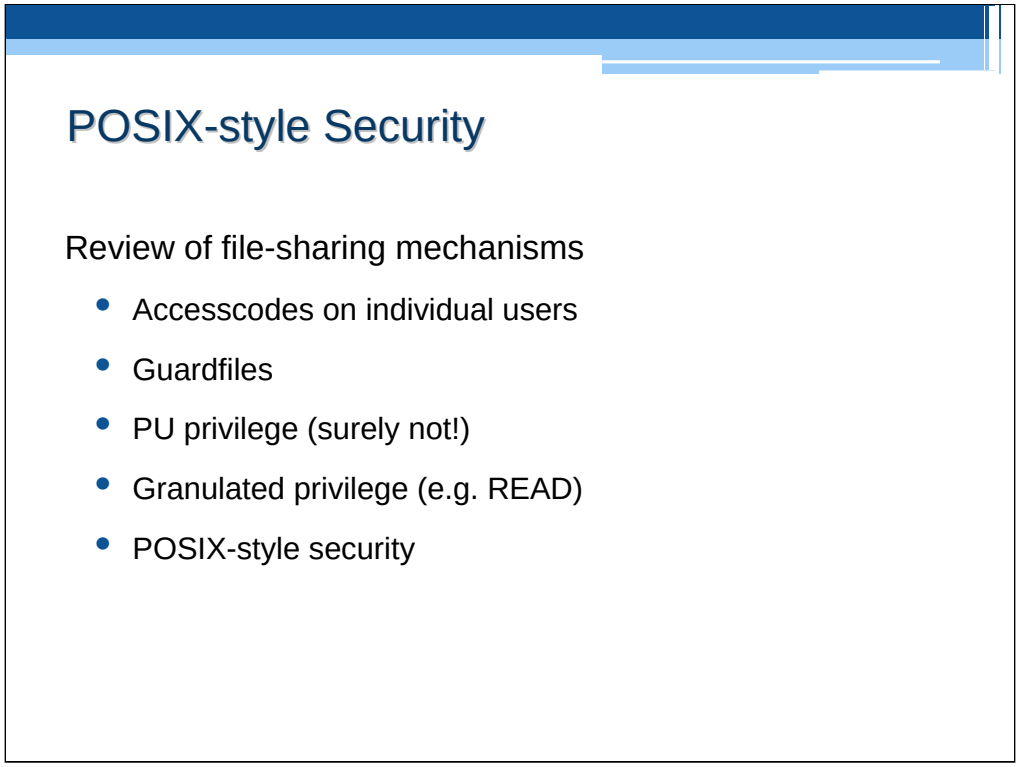


(Kevin gives apologies for absence)

A common concern of MCP customers is how to allow multiple users to share access to files. An example might be a report produced by a production run that would typically reside under the production usercode.

How can we safely and simply allow access to such files without compromising security?



Firstly, let's have a look at the possibilities. We can always copy files to individual user directories, but that's not an ideal solution.

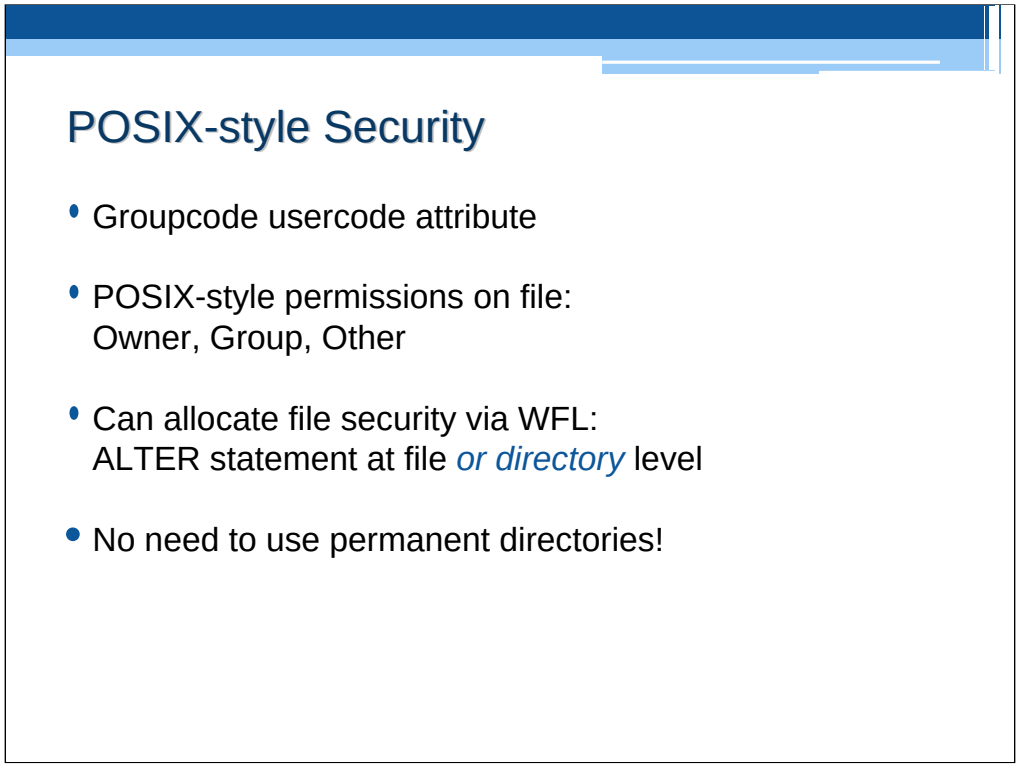
Many customers deploy accesscodes for file sharing, where a common usercode is shared by a group of individuals. This is a very general solution, not tailored for specific files.

Guardfiles provide a more specific access authorisation, but can be awkward to maintain. I have seen situations where guardfiles are not attached to ***any*** file, or where the guardfile attached to a data file no longer exists!

Handing out PU privilege will work, but is not easy to hide from the auditor!!!

Granulated privileges such as the READ attribute on the usercode is only useful if you want to grant access to ***all*** files on the system.

This leads us to POSIX-style security which I believe to be more refind solutions in most cases.



POSIX-style security is based on the concept of Owner, Group and Other access. Each category can be given read, write, or execute access, in any combination.

The Owner of a file is typically, but not necessarily, the usercode of the creator. We can establish a group of users by attaching the GROUPCODE user attribute in Userdatafile to individual usercodes. ‘Other’ is defined as any usercode not defined by Owner or Group.

We can allocate this type of file security through the Work Flow statement ALTER, which can be used at either file or directory level.

And the coolest aspect of this mechanism is that POSIX-style security does ***not*** require Permanent Directories to be established. This is a common misconception! It also makes things very much easier. This is the ‘I didn’t know you could do that’ moment!

POSIX-style Security

Case study (example)

I want to make all files in the directory (TSC)BATCH/= available to my operations staff for read-only purposes

The operations people have usercodes of OPS1, OPS2, OPS3

So, let’s have a look at the process of implementing this mechanism.

(Maybe read out the text of the slide).

POSIX-style Security

The current security on the batch files is:

```
lfil      batch/1       :sec
#RUNNING 35855
#?
ON WORK
(TSC) : DIRECTORY
. BATCH : DIRECTORY
. . 1 : JOBSYMBOL SECURITY=OWNER TSC:RWX, GROUP <none>:NO,
OTHER:NO
          PRIVATE (I/O)
```

MCP attempts to interpret ‘traditional’-style security in terms of POSIX. You can see here that this file can only be accessed by the usercode under which it resides (or by a privileged user, of course).

The POSIX interpretation shows access is allowed only to the owner.

POSIX-style Security

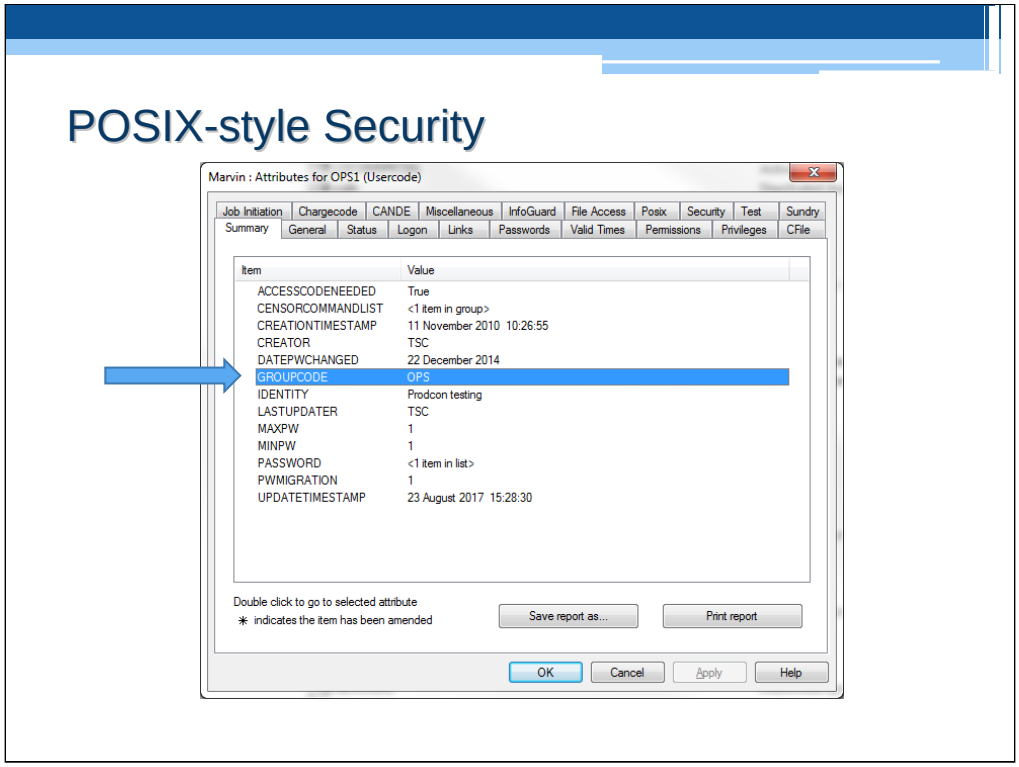
```
wfl alter batch/= (group=ops, grouprwx=r)
#RUNNING 35864
#35864 PK501 64 FILES ALTERED IN BATCH/= ON WORK
#

lfil batch/1:sec
#RUNNING 35865
#?
ON WORK
(TSC) : DIRECTORY
. BATCH : DIRECTORY
. . 1 : JOBSYMBOL SECURITY=OWNER TSC:RWX, GROUP OPS:R,
        OTHER:NO PUBLIC (IN)
#
```

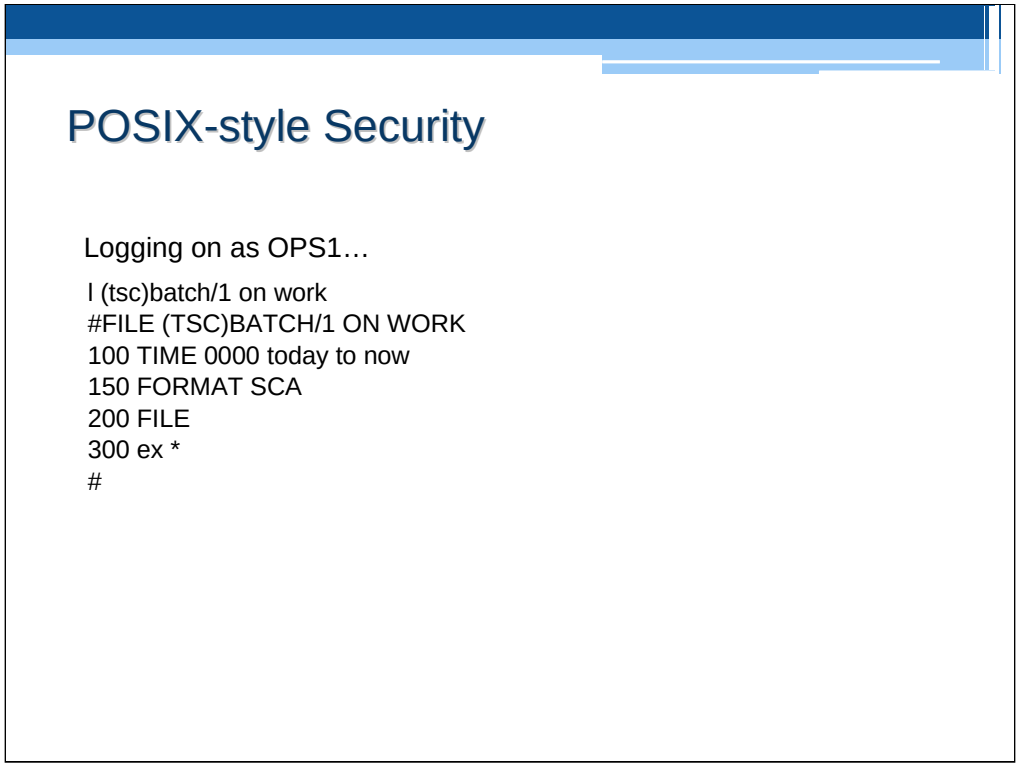
Now, I've changed security via the ALTER statement on the BATCH directory to give the group OPS read-only access.

The LFIL response confirms the new access rights. We can see that the owner is still TSC with full access rights, but group OPS are allowed to read the file. Note that MCP attempts to interpret this in terms of traditional MCP file security (hence the 'PUBLIC (IN)).

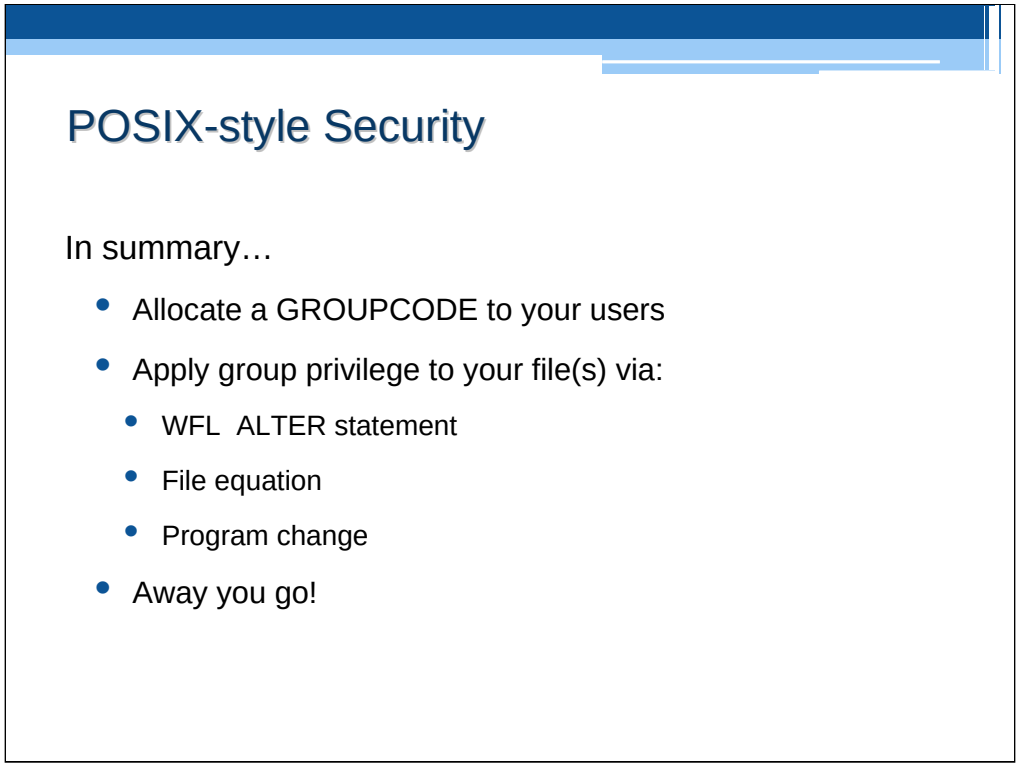
POSIX-style Security



All that remains is to establish the group OPS in Userdatafile. This slide shows the groupcode attribute being set to OPS for usercode OPS1. Repeat for OPS2 and OPS3.



Now I logged on as OPS1, and you can see here that I am allowed to list the contents of file (TSC)BATCH/1.



Here is a summary of the 2 steps required, which can be performed in either order.

Note that group privilege can also be assigned to the files via Work Flow file equation, or programmatically. The suggested method is probably the easiest; the ALTER statement would need to be embedded in the appropriate Work Flow job so that the change occurs after the file has been created.

POSIX-style Security

Of course, can get much more complicated

Other attributes to consider:

- GROUP (file),
- ALTERNATEGROUPS (file),
- SUPPLEMENTARYGROUPS (task)

References:

- I/O Subsystem Programming Guide
- Work Flow Language (WFL) Programming Reference Manual
- File Attributes Programming Reference Manual

The case study shows a simple scenario. For more complicated situations, help is at hand via these reference documents.