Understanding MPE's File System Limitations

an Allegro Consultants Whitepaper

Let's begin this discussion with a good, working definition of what a file system actually is. Turning to the vast resource know as Wikipedia – we find that **all** file systems share some common characteristics. These include:

- **Space management**: the control of the transfer of data between programs and peripheral devices such as disks. This includes being aware of what storage is in-use and free.
- **Naming rules**: the number and types of characters allowed to name a file, directory, group and account
- **Directory structure**: traditional MPE has a relatively flat, three-level directory structure (file, group and account). Other computers use a multi-level hierarchical file system. Regardless of how the files are ordered – the important point is the files ARE organized.
- **Metadata**: the stored characteristics of each file. This includes things like the size and ownership.
- **Maintaining integrity**: insures that, regardless of the actions by programs accessing the data, the file system structure remains consistent. This includes taking action if a program modifying data terminates abnormally or neglects to inform the file system that is has completed its activities. This may include updating the metadata, the directory entry and handling any data that was buffered but not yet updated on the physical storage media.

Additionally, all file systems –*including MPE's* – have functional limits. They include things like the maximum storable data capacity within that system. These limits are frequently a best-guess effort by the designer.

For this whitepaper, we are going to focus on a particular limit – the number of files in a group or directory or more precisely the Transaction Manager's (XM) ability to process a large number of files.

As was already mentioned, one of the most important jobs for the file system is to maintain integrity. On MPE, the integrity is maintained by the Transaction Manager – which, by design, is limited.

The general rule of thumb is to try to not allow traditional MPE groups to have more than 50,000 files and HFS directories to have more than 20,000 files.

Consider the following scenario: on our MPE system there is a big account dedicated to a specific application. In this account, there is a group – LOGS – that contains transaction log files. Each log file's name begins with an "A" and the remaining seven characters are digits. For *years and years*, the application has been diligently writing out these log files. One day, there's a program abort due to a duplicate, permanent file name. The application wanted to create a file named "A0134569.LOGS" but there was already a file with that name. No problemo. We'll just purge that file (it's ages old, after all) and all will be well.

Uh huh.

A few moments after entering the purge command, the system crashes with a System Abort 2216 ("The procedure XM_ADD_COPYF_RECS_TO_BUFFER determined that either there were too many records to be copied forward or there was insufficient space for them."). Ugh.

Don't confuse this with a lack of free space or the size of the files. It's an entirely different problem – the number of files in one location!

When the file system went to remove "A0134569.LOGS", it had to also shift all the files that came after it "up" one position. It's this massive amount of "thrashing" that overwhelmed the XM – the maintainer of integrity! Rather than risk integrity, the Transaction Manager aborted the system.

After the system comes back up, your pointy-haired manager says "clean up all those old files!" After another SA2216 or two you realized that almost anything you do to the files in this group is going to cause the system to crash. (And it's only Tuesday ☹)

The only way to resolve this problem is to purge the files in *reverse order*. (A "purgegroup" won't work because it is going to attempt to purge the files in alphabetical order.) Storing the files, before you start is critical. You'll also need to identify what *tiny* fileset is going to be restored (e.g., files 30-days old or newer) once all the files have been removed.

The way to keep this from happening again – or to find out if you're already in hot water – is to count of the number of files in a group or directory. One way to do this is to use the following shell command:

```
# ls -l /ACCOUNT/GROUP | wc -l
```

This counts the number of lines/files ("`wc -l`") in the directory listing ("`ls -l`") for a particular GROUP.ACCOUNT.

Naturally, we also suggest that you give Allegro's free "bigdirs" program a try since it's designed to check your entire system rather than a single location.

This count needs to be done on a regular basis (say, once a week) to keep the situation from blindly developing again. If the count returned nears a threshold, steps need to be taken to reduce the number of files.