File Descriptor Limitation

Issue #114: There is a limit to the number of files that a process may have open at any given time. This limit is determined with this command:

```
% limit descriptors
descriptors 1024
```

Some systems will allow you to change your limit by specifying a new limit.

```
% limit descriptors 2048
% limit descriptors
descriptors 2048
```

Other systems will require a privileged user to do this. On the large system at LANL, Cielo, the System Admins set the limit for a select user group on ci-mesh to a higher number so that serial inspection applications that need to open all files in a set could do so without error.

To close this issue, we still need to implement a cache mechanism so that the PLFS library can live within any limit.

Correct plfsrc man Page

Issue #121: Some of the descriptions of the plfsrc parameters in the man page were incorrect. For example, "statfs" did not explain that it must follow a "mount_point" directive and only modifies the "mount_point" directive that it follows. Parameter explanations were clarified and corrected.

Plfs Daemon Should Fail When plfs_check_config Fails

Issue #122: In the past the "plfs" program (FUSE-based daemon) would mount a PLFS "mount_point" defined in the PLFSRC file, even if "plfs_check_config" noted an error. For example, if one or more (including all) backends were wrong or not online, "plfs_check_config" would return an error, but "plfs" would mount it and erroneous behavior would ensue.

We discussed this in the PLFS meeting on 11/8/12. Due to the fact that we do not want "plfs" to stat every backend from every node that mounts all the PLFS file systems at mount time, because it is not scalable, we'll allow "plfs" to mount a file system, even if a backend is incorrect or not online at the time.

We will investigate whether or not a run-time error can be returned that will advise a user to ask an Admin to run plfs_check_config when something goes wrong because of a missing backend or the like.

Call "readdir" and all PLFS API Calls without FUSE Mount

Issue #123: We need to be able to call "readdir" and all PLFS library API calls without requiring a FUSE mount to be present. The PLFS library was intended to be used by applications and not require an operating system file system mount to be active. Rather, the PLFS library can use the PLFSRC file to determine the validity and backend storage of a PLFS file system.

Data Sieving & O_RDWR in ad_plfs

Issue #124: We never want data sieving turned on for PLFS. Data sieving requires O_RDWR, but O_RDWR badly affects PLFS performance. The ad_plfs suite of files was fixed for Cray's MPI and OpenMPI such that data sieving is always off and that it does not allow O_RDWR to be used to open a file.

Renaming Files Failed

Issues #79, #128 and #130: When using "mv" to rename a file on a PLFS file system it would fail with a "mv: cannot move" error. It was an error in the new abstraction layer, IOstore, that allows different specific storage implementations to work underneath PLFS. This error was fixed and "mv" now works correctly.

"plfs_check_config" Doesn't Error When it Should

Issues #131, #132, and #133: The plfs_check_config utility used to return an error when a backend was offline or incorrectly named in the PLFSRC file. Now, it does not. You can put a backend in that does not exist, for example, and it will return success. This error would only occur for the second or later backend. It would correctly detect if the first backend was missing. Additionally, the "-mkdir" parameter would not create missing backend directory trees. This has all been fixed so that plfs_check_config errors when it should and the "-mkdir" parameter creates missing backend directory trees.

"ad_plfs" Patch Documentation & Fixes

Issues #134, #135, and #136: MPI patching needed better instructions and to correctly handle weak symbols used by Cray's MPI.

README Documentation Clarifications

Issues #137, #138, #139, #140, and #145: Updates to the README documentation to make things clearer to developers and users.

Renaming a Symbolic Link is Broken

Issue #142: If you created a symbolic link to a file, then renamed the link, the new link did not point to the file and the metadata was broken. This has been fixed.

Can’t Write to mlog Prevents PLFS from Writing

Issue #144: When the file system to which mlog wrote its file filled we would see an error ENOSPACE, and then PLFS would not write. This turned out to be that mlog calls would set “errno” to ENOSPACE, even though mlog did not use that error, thus overwriting previous values of “errno” set by PLFS. We made a change to mlog so that the previous value of “errno” is preserved and PLFS reacts correctly, irrespective of whether or not mlog succeeded logging messages.

Using MPI/IO on a “shared_file” FS Allowed Write/Read to Directory

Issues #146 and #147: Using MPI/IO in container mode (shared_file or N-1), it was possible to open a directory as a file and write to it and read from it. This has been fixed so that if one opens a directory for write or read, an error is generated.

POSIXstore Timing Enhancements

Issue #149: With the introduction of the IOstore abstraction, some of the timing information was lost when doing POSIX I/O. Some of it was restored. We’re still thinking about how to add this back using a storage layer abstractions with specific storage implementations under the abstraction.