



IBM Software Group

TPF Users Group Spring 2007

Pseudo-file Systems in z/TPF: PROCFS and SYSFS

z/TPF PUT04 APAR PJ31637

Name : Stephen Record / Michael Drejza
Venue : Database / TPFDF Subcommittee

AIM Enterprise Platform Software

IBM z/Transaction Processing Facility Enterprise Edition 1.1.0

© IBM Corporation 2007

Any references to future plans are for planning purposes only. IBM reserves the right to change those plans at its discretion. Any reliance on such a disclosure is solely at your own risk. IBM makes no commitment to provide additional information in the future.

Trademarks

IBM and WebSphere are trademarks of International Business Machines Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Notes

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

This presentation and the claims outlined in it were reviewed for compliance with US law. Adaptations of these claims for use in other geographies must be reviewed by the local country counsel for compliance with local laws.

Agenda

- Background
- Infrastructure
- Attributes
- Examples
- Motivation
- Implementation
- Summary

What is a pseudo-file system?

- Easier to say what it is not than to specify what it is....
- A Virtual File System (VFS) is an abstraction layer on top of a set of mountable "concrete" or "physical" file systems.
- The VFS allows programs to access the physical file systems in a uniform way (via a conventional file system API, in particular POSIX) without regard to the specific characteristics of their implementations.
- A pseudo-file system is a mountable file-system which has no conventional file store (e.g. a dedicated pool of disk blocks) associated with it.
- The entities in a pseudo-file system (i.e. regular files, directories, links) are instead instantiated by blocks of program code which simulate file system operations in some appropriate manner.
- The prototypical use of pseudo-files is to provide access to operating system kernel variables and data areas through the file system APIs.

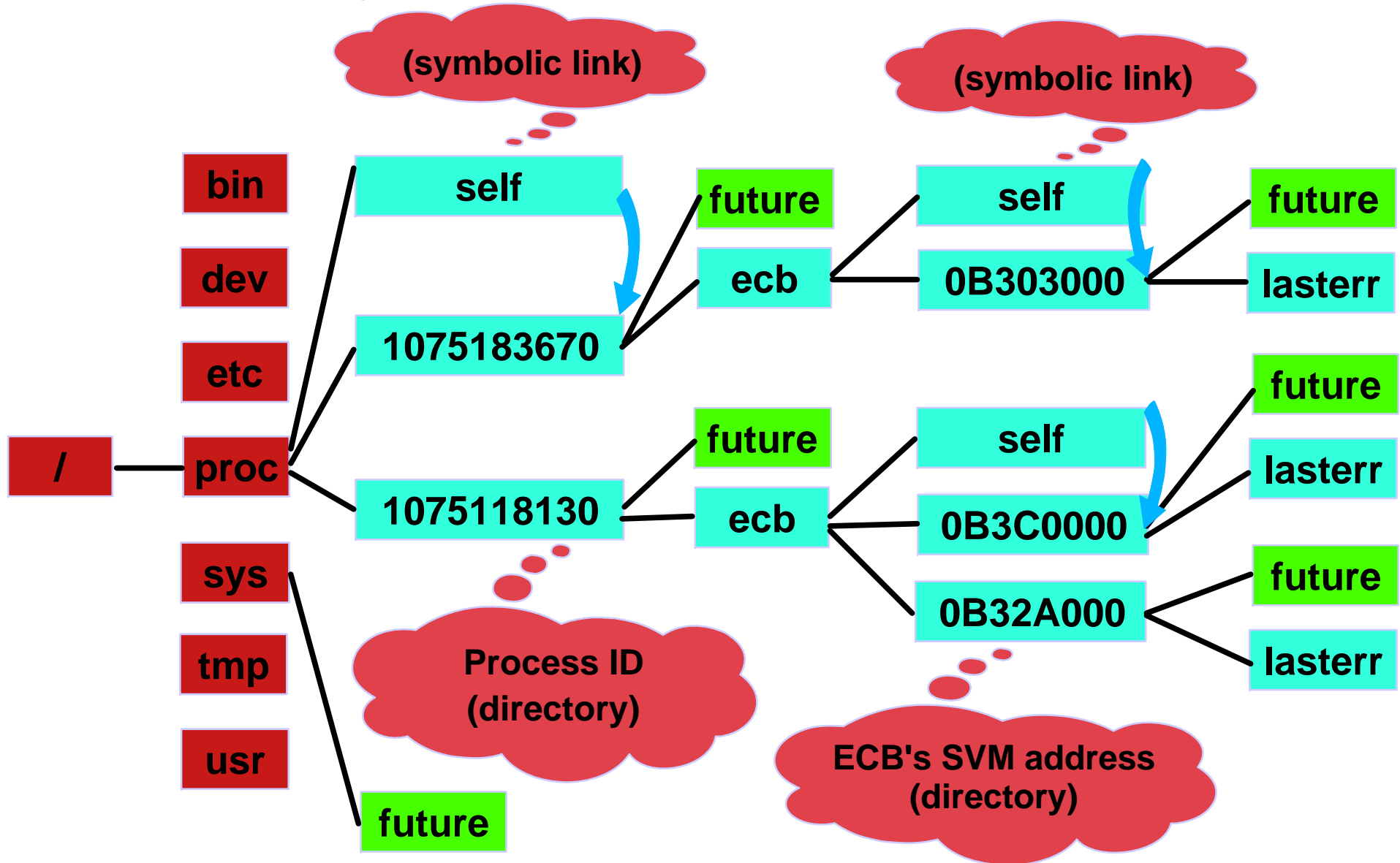
PROCFS - the process pseudo-file system

- The PROCFS is mounted on (and known familiarly as) /proc
- Top level subdirectories represent active processes
 - ▶ the subdirectory name is the numeric process ID
 - ▶ "self" is a symbolic link to the current process
- Under each process, subdirectories and files representing process level constructs may be added
- The infrastructure includes the "ecb" subdirectory which contains subdirectories representing the threads of the process
 - ▶ the subdirectory name is the hexadecimal SVA of its ECB
 - ▶ "self" is a symbolic link to the SVA of the current ECB
- Under each ECB SVA, subdirectories and files representing ECB level constructs may be added
 - ▶ under each SVA, the infrastructure provides the "lasterr" file for PROCFS and SYSFS diagnostic purposes

SYSFS - the system pseudo-file system

- The SYSFS is mounted on (and known familiarly as) /sys
- Non-process-specific primitives (e.g. globals and events) will be represented in the SYSFS as needed
 - ▶ direction taken in Linux 2.6 kernel
- Similar to the PROCFS in concept but different in details
 - ▶ no directory structure representing processes or ECBs
 - ▶ initially only filesystem infrastructure is provided

Initial Directory Structure



Attributes of PROCFS and SYSFS

- Processor and subsystem unique
- No persistent data stored in pseudo-file systems
 - ▶ data is written and read either
 - to/from accessible z/TPF system data areas
 - or
 - via z/TPF APIs
- Only one PROCFS and one SYSFS instance may exist per processor per subsystem
 - ▶ must be mounted on /proc and /sys, respectively
 - ▶ mounted automatically by file system initialization
 - ▶ may be unmounted if necessary
 - for flexibility when debugging file system problems

Example - list all active processes

- Write a 'C' program to list all of the active processes in the system without implementing TPF specific code to access the z/TPF data structures directly. The program might do the following:

```
DIR* dir = NULL;
struct dirent* dirent = NULL;
dir = opendir( "/proc" );
while( ( dirent = readdir( dir ) ) != NULL )
    printf( "%s", dirent->d_name );
closedir( dir );
```

- To obtain similar results from the command line, use:
`zfile ls /proc`
- Or there's the more traditional command line alternative:
`zfile ps`

Example - obtain diagnostic info after a PROCFS error

- Add error checking to the previous example. If the `readdir()` function fails, check for PROCFS specific diagnostic information that would further describe the error.

```
errno = 0;
while( ( dirent = readdir( dir ) ) != NULL )
    printf( "%s", dirent->d_name );
if ( errno != 0 )
{
    char buffer_le[ 512 ];
    int fd = open(
"/proc/self/ecb/self/lasterr", O_RDONLY, 0 );
    read( fd, buffer_le, 512 );
    close( fd );
    printf( buffer_le );
}
```

What are PROCFS and SYSFS good for?

- Means of providing language neutral access to z/TPF or POSIX primitives:
 - ▶ ECB information: e.g. owner information and resource limits
 - ▶ POSIX process information: e.g. environment and memory
 - ▶ z/TPF system information: e.g. globals and events
- Structure within which to provide interfaces to system services and data areas through the file system API
 - ▶ modeled on a long-standing UNIX design, also in Linux
 - ▶ exploit the virtual file system (VFS) architecture
 - ▶ satisfy a requirement for access from languages like Java
 - ▶ provide easy extensibility for further development by both IBM and customers

Implementation details - targets

All files, directories and symbolic links are implemented in the pseudo-file systems as "targets."

- A target consists of:
 - ▶ an entry in a table defining:
 - the name and type of target
 - its initial RWX permissions
 - where it exists within the pseudo-file system's subtree
 - ▶ a program that provides the behavior of the target
 - the program is invoked by the PROCFS/SYSFS when references are made to the target
 - the program simulates the requested file system operation, utilizing z/TPF data structures and z/TPF APIs to access the information represented by the target, and returns a result to the caller

More about targets

- New targets may be added by both IBM and customers
 - ▶ IBM will be announcing additional PROCFS and SYSFS targets at a later time
 - For example eownrc may be implemented under the directory: `/proc/<pid>/ecb/<sva>/owner`
 - ▶ Users can implement a new target by
 - adding a new table entry to the UPRC or USYS user exit and
 - creating a new program that defines the behavior of the target which is referred to by the user exit table entry
- For more information on creating new targets, search for the heading "Creating the target program" in the Information Center at the following URL: <http://www.ibm.com/tpf/pubs/tpfpubs.htm>

Summary

- PROCFS and SYSFS support added by PJ31637
- Provide access to kernel data via the file system API
- Readily extensible by both IBM and customers
- Documented in detail in the Information Center
 - ▶ z/TPF Database User's Guide
 - ▶ z/TPF Process Pseudo-File System and System Pseudo-File System Targets