NAME

psx_syscall3, psx_syscall6, psx_set_sensitivity − POSIX semantics for system calls

SYNOPSIS

```
#include <sys/psx_syscall.h>

long int psx_syscall3(long int syscall_nr,
                      long int arg1, long int arg2, long int arg3);
long int psx_syscall6(long int syscall_nr,
                      long int arg1, long int arg2, long int arg3,
                      long int arg4, long int arg5, long int arg6);
int psx_set_sensitivity(psx_sensitivity_t sensitivity);
void psx_load_syscalls(long int (**syscall_fn)(long int,
                                 long int, long int, long int),
                       long int (**syscall6_fn)(long int,
                                 long int, long int, long int,
                                 long int, long int, long int));
```

Any code that uses one of the above functions can be linked as follows:

ld ... −lpsx −lpthread

gcc ... −lpsx −lpthread

Note, special flags are needed to get **-lcap** to operated with process wide capabilities when linked with **-lpsx**. Namely, use the **pkg-config** option file: **gcc ... −lcap $(pkg-config −−libs −−cflags libpsx)** More details are available in the **cap_get_proc**(3) man page.

DESCRIPTION

The **libpsx** library attempts to fill a gap left by the **pthreads**(7) implementation on Linux. To be compliant POSIX threads, via the **nptl**(7) **setxid** mechanism, glibc maintains consistent UID and GID credentials amongst all of the threads associated with the current process. However, other credential state is not supported by this abstraction. To support these extended kernel managed security attributes, **libpsx** provides a more generic pair of wrapping system call functions: **psx_syscall3**() and **psx_syscall6**(). Like the **setxid** mechanism, the coordination of thread state is mediated by a realtime signal. Whereas the **nptl:setxid** mechanism uses signo=33 (which is hidden by glibc below a redefined **SIGRTMIN**), **libpsx** inserts itself in the **SIGSYS** handler stack. It goes to great length to be the first such handler but acts as a pass-through for other **SIGSYS** uses.

An inefficient macrology trick supports the **psx_syscall**() pseudo function which takes 1 to 7 arguments, depending on the needs of the caller. The macrology (which ultimately invokes **__psx_syscall**()) pads out the call to actually use **psx_syscall3**() or **psx_syscall6**() with zeros filling the missing arguments. While using this in source code will make it appear clean, the actual code footprint is larger. You are encouraged to use the more explicit **psx_syscall3**() and **psx_syscall6**() functions as needed.

psx_set_sensitivity() changes the behavior of the mirrored system calls: **PSX_IGNORE** ensures that differences are ignored (the default behavior); **PSX_WARNING** prints a stderr notification about how the results differ; and **PSX_ERROR** prints the error details and generates a **SIGSYS** signal.

psx_load_syscalls() can be used to set caller defined function pointers for invoking 3 and 6 argument syscalls. This function can be used to configure a library, or program to change behavior when linked against **libpsx**. Indeed,**libcap** uses this function from **libpsx** to o verride its thread scoped default system call based API. When linked with **libpsx**, **libcap** can operate on all the threads of a multithreaded program to operate with POSIX semantics.

RETURN VALUE

The return value for system call functions is generally the value returned by the kernel, or −1 in the case of an error. In such cases **errno**(3) is set to the detailed error value. The **psx_syscall3**() and **psx_syscall6**()

functions attempt a single threaded system call and return immediately in the case of an error. Should this call succeed, then the same system calls are executed from a signal handler on each of the other threads of the process.

CONFORMING TO

The needs of **libcap**(3) for POSIX semantics of capability manipulation. You can read more about why this is needed here:

https://sites.google.com/site/fullycapable/who-ordered-libpsx

Versions of **libpsx** prior to 2.72 only supported pthreads. Since libpsx-2.72 the library works with all Linux thread implementations as it operates at the lowest level of thread abstraction, LWPs.

REPORTING BUGS

The **libpsx** library is distributed from https://sites.google.com/site/fullycapable/ where the release notes may already cover recent issues. Please report newly discovered bugs via:

https://bugzilla.kernel.org/buglist.cgi?component=libcap&list_id=1090757

SEE ALSO

libcap(3), **cap_get_proc**(3), **pthreads**(7) and **nptl**(7).