tmsyscall Documentation

Release 0.0.0

João Pinto

May 16, 2018

Contents

tmsyscall is a python library wrapper for some Linux kernel system calls.

It is based in the code from Morgan-Stanley's threadmill project.

This library can help you:

- List mounted filesystems
- Mount/unmount filesystems
- Unshare linux namespaces (man unshare)
- Move the root filesystem of the current process (man privot_root)

CHAPTER 1

Sections

1.1 Filesystem Mount API

1.2 Namespace Unshare API

1.3 PivotRoot API

1.4 Example Code

```
#!/bin/python
#
# You can test it using:
#   wget http://dl-cdn.alpinelinux.org/alpine/v3.7/releases/x86_64/alpine-minirootfs-
↪3.7.0-x86_64.tar.gz
#   sudo bash -c "mkdir /tmp/rootfs; tar xvf alpine-minirootfs-3.7.0-x86_64.tar.gz -C␣
↪/tmp/rootfs"
#   sudo python example.py /tmp/rootfs bash

from __future__ import print_function
import os
import sys
from tmsyscall.unshare import unshare, CLONE_NEWNS, CLONE_NEWUTS, CLONE_NEWIPC, CLONE_
↪NEWPID, CLONE_NEWNET
from tmsyscall.mount import mount, unmount, MS_BIND, MS_PRIVATE, MS_REC, MNT_DETACH
from tmsyscall.mount import mount_procfs
from tmsyscall.pivot_root import pivot_root
from os.path import exists, join

def setup_process_isolation():
    # Detach from parent's mount, hostname, ipc and net  namespaces
    unshare(CLONE_NEWNS| CLONE_NEWUTS | CLONE_NEWIPC| CLONE_NEWNET)
```

(continues on next page)

(continued from previous page)

```
    # Set mount propagation to private recursively. Hopefully equivalent to
    #    mount --make-rprivate /
    # This is needed to prevent mounts in this container leaking to the parent.
    mount('none', '/', None, MS_REC|MS_PRIVATE, "")

    root_fs = sys.argv[1]

    # This bind mount call is needed to satisfy a requirement of the `pivotroot`␣
↪system call
    #   "new_root and put_old must not be on the same file system as the current root"
    # It is achieved by mounting "new_root" as a bind mount to "new root"
    mount(root_fs, root_fs, "", MS_BIND|MS_REC, "")

    old_root = join(root_fs, ".old_root")
    if not exists(old_root):
        os.makedirs(old_root, 0o700)
    pivot_root(root_fs, old_root)

    # We don't want the host root to be available to the container
    unmount("/.old_root", MNT_DETACH)
    os.rmdir("/.old_root")

    os.chdir("/")

    # Mount /proc for apps that need it
    if not exists("proc"):
        os.makedirs("proc", 0o700)
    mount_procfs('.')


def child():
    setup_process_isolation()
    os.execvp(sys.argv[2], sys.argv[2:])


def parent(child_pid):
    pid, status = os.waitpid(child_pid, 0)
    print("wait returned, pid = %d, status = %d" % (pid, status))


def main():
    # Detach from pid namespace so that our child get's a clean /proc with the new␣
↪namespace
    unshare(CLONE_NEWPID)
    pid = os.fork()
    if pid == 0:
        child()
    else:
        parent(pid)

main()
```