memfd_create − (anonymous file)

#include <sys/memfd.h>

int memfd_create(const char ****name*, unsigned int** *flags***);**

memfd_create() (anonymous file) (truncate) RAM (anonymous memory) **memfd_create**() (**MAP_ANONYMOUS mmap**(2))

0 **ftruncate**(2) (**write**(2))

name */proc/self/fd/* *memfd:*

flags **memfd_create**()

MFD_CLOEXEC
close−on−exec (**FD_CLOEXEC**) **open**(2) **O_CLOEXEC**

MFD_ALLOW_SEALING
sealing **fcntl**(2) **F_ADD_SEALS F_GET_SEALS** seal seal **F_SEAL_SEAL** seal

flags 0

memfd_create() (**O_RDWR**) **O_LARGEFILE**

fork(2) **execve**(2) **memfd_create**() **fork**(2) close−on−exec **execve**(2)

memfd_create() −1 *errno*

EFAULT
name

EINVAL
flags *name*

EMFILE

ENFILE

ENOMEM

memfd_create() Linux 3.17 GNU C

memfd_create() Linux

memfd_create() *tmpfs* **memfd_create**() **fcntl**(2) file−sealing API

memfd_create() file sealing (**MFD_ALLOW_SEALING** file−sealing) *tmp* **open**(2) **O_TMP-FILE**

file sealing

file sealing () **SIGBUS** (**SIGBUS**)

sealing

sealing

1. **memfd_create**() *tmpfs* **memfd_create**()
2. **ftruncate**(2) **mmap**(2)
3. **fcntl**(2) **F_ADD_SEALS** seal (seal **F_SEAL_WRITE**)
4. *tmpfs*
   * **memfd_create**() UNIX (**unix**(7) **cmsg**(3)) **mmap**(2)
   * **fork**(2) (ID file sealing)
   * */proc/<pd>/fd/<fd>* *<pid>* (**memfd_create**()) PID *<fd>* **memfd_create**() **mmap**(2)
5. **fcntl**(2) **F_GET_SEALS** seal (**F_SEAL_SEAL** seal) seal

memfd_create() file sealing API 2

t_memfd_create.c **memfd_create**() *tmpfs* seal 3 2 2 3 seal

2 *t_get_seals.c* **memfd_create**() seal

tmpfs seal

```
$ ./t_memfd_create my_memfd_file 4096 sw &
[1] 11775
PID: 11775; fd: 3; /proc/11775/fd/3
```

t_memfd_create **memfd_create**() */proc/PID/fd* **memfd_create**() */proc/PID/fd* *t_get_seals* seal

```
$ readlink /proc/11775/fd/3
/memfd:my_memfd_file (deleted)
$ ./t_get_seals /proc/11775/fd/3
Existing seals: WRITE SHRINK
```

: t_memfd_create.c

```
#include <sys/memfd.h>
#include <fcntl.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <stdio.h>

#define errExit(msg)    do { perror(msg); exit(EXIT_FAILURE); \
                        } while (0)

int
main(int argc, char *argv[])
{
    int fd;
    unsigned int seals;
    char *addr;
    char *name, *seals_arg;
    ssize_t len;

    if (argc < 3) {
        fprintf(stderr, "%s name size [seals]\n", argv[0]);
        fprintf(stderr, "\t'seals' can contain any of the "
                "following characters:\n");
        fprintf(stderr, "\t\tg − F_SEAL_GROW\n");
        fprintf(stderr, "\t\ts − F_SEAL_SHRINK\n");
        fprintf(stderr, "\t\tw − F_SEAL_WRITE\n");
```

```
            fprintf(stderr, "\t\tS − F_SEAL_SEAL\n");
            exit(EXIT_FAILURE);
        }

        name = argv[1];
        len = atoi(argv[2]);
        seals_arg = argv[3];

        /* Create an anonymous file in tmpfs; allow seals to be
           placed on the file */

        fd = memfd_create(name, MFD_ALLOW_SEALING);
        if (fd == −1)
            errExit("memfd_create");

        /* Size the file as specified on the command line */

        if (ftruncate(fd, len) == −1)
            errExit("truncate");

        printf("PID: %ld; fd: %d; /proc/%ld/fd/%d\n",
                (long) getpid(), fd, (long) getpid(), fd);

        /* Code to map the file and populate the mapping with data
           omitted */

        /* If a 'seals' command−line argument was supplied, set some
           seals on the file */

        if (seals_arg != NULL) {
            seals = 0;

            if (strchr(seals_arg, 'g') != NULL)
                seals |= F_SEAL_GROW;
            if (strchr(seals_arg, 's') != NULL)
                seals |= F_SEAL_SHRINK;
            if (strchr(seals_arg, 'w') != NULL)
                seals |= F_SEAL_WRITE;
            if (strchr(seals_arg, 'S') != NULL)
                seals |= F_SEAL_SEAL;

            if (fcntl(fd, F_ADD_SEALS, seals) == −1)
                errExit("fcntl");
        }

        /* Keep running, so that the file created by memfd_create()
           continues to exist */

        pause();

        exit(EXIT_SUCCESS);
    }
```

: t_get_seals.c

```
    #include <sys/memfd.h>
    #include <fcntl.h>
    #include <unistd.h>
    #include <stdlib.h>
```

```
#include <string.h>
#include <stdio.h>

#define errExit(msg)    do { perror(msg); exit(EXIT_FAILURE); \
                        } while (0)

int
main(int argc, char *argv[])
{
   int fd;
   unsigned int seals;

   if (argc != 2) {
      fprintf(stderr, "%s /proc/PID/fd/FD\n", argv[0]);
      exit(EXIT_FAILURE);
   }

   fd = open(argv[1], O_RDWR);
   if (fd == −1)
      errExit("open");

   seals = fcntl(fd, F_GET_SEALS);
   if (seals == −1)
      errExit("fcntl");

   printf("Existing seals:");
   if (seals & F_SEAL_SEAL)
      printf(" SEAL");
   if (seals & F_SEAL_GROW)
      printf(" GROW");
   if (seals & F_SEAL_WRITE)
      printf(" WRITE");
   if (seals & F_SEAL_SHRINK)
      printf(" SHRINK");
   printf("\n");

   /* Code to map the file and access the contents of the
      resulting mapping omitted */

   exit(EXIT_SUCCESS);
}
```

fcntl(2), **ftruncate**(2), **mmap**(2), **shmget**(2), **shm_open**(3)