
Subject: [RFC PATCH 5/5] syscall: sys_fbind() introduced

Posted by [Stanislav Kinsbursky](#) on Wed, 15 Aug 2012 16:22:25 GMT

[View Forum Message](#) <> [Reply to Message](#)

This syscall allows to bind socket to specified file descriptor.
Descriptor can be gained by simple open with O_PATH flag.
Socket node can be created by sys_mknod().

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>

```
arch/x86/syscalls/syscall_32.tbl | 1 +
arch/x86/syscalls/syscall_64.tbl | 1 +
include/linux/syscalls.h         | 1 +
kernel/sys_ni.c                  | 3 +++
net/socket.c                     | 25 +++++
5 files changed, 31 insertions(+), 0 deletions(-)
```

```
diff --git a/arch/x86/syscalls/syscall_32.tbl b/arch/x86/syscalls/syscall_32.tbl
```

```
index 7a35a6e..9594b82 100644
```

```
--- a/arch/x86/syscalls/syscall_32.tbl
```

```
+++ b/arch/x86/syscalls/syscall_32.tbl
```

```
@@ -356,3 +356,4 @@
```

```
347 i386 process_vm_readv sys_process_vm_readv compat_sys_process_vm_readv
```

```
348 i386 process_vm_writev sys_process_vm_writev compat_sys_process_vm_writev
```

```
349 i386 kcmp sys_kcmp
```

```
+350 i386 fbind sys_fbind
```

```
diff --git a/arch/x86/syscalls/syscall_64.tbl b/arch/x86/syscalls/syscall_64.tbl
```

```
index 51171ae..f964df8 100644
```

```
--- a/arch/x86/syscalls/syscall_64.tbl
```

```
+++ b/arch/x86/syscalls/syscall_64.tbl
```

```
@@ -319,6 +319,7 @@
```

```
310 64 process_vm_readv sys_process_vm_readv
```

```
311 64 process_vm_writev sys_process_vm_writev
```

```
312 64 kcmp sys_kcmp
```

```
+313 common fbind sys_fbind
```

```
#
```

```
# x32-specific system call numbers start at 512 to avoid cache impact
```

```
diff --git a/include/linux/syscalls.h b/include/linux/syscalls.h
```

```
index 19439c7..9e78fa4 100644
```

```
--- a/include/linux/syscalls.h
```

```
+++ b/include/linux/syscalls.h
```

```
@@ -602,6 +602,7 @@ asmlinkage long sys_setsockopt(int fd, int level, int optname,
```

```
asmlinkage long sys_getsockopt(int fd, int level, int optname,
```

```
char __user *optval, int __user *optlen);
```

```
asmlinkage long sys_bind(int, struct sockaddr __user *, int);
```

```
+asmlinkage long sys_fbind(int, int);
```

```
asmlinkage long sys_connect(int, struct sockaddr __user *, int);
```

```

asmlinkage long sys_accept(int, struct sockaddr __user *, int __user *);
asmlinkage long sys_accept4(int, struct sockaddr __user *, int __user *, int);
diff --git a/kernel/sys_ni.c b/kernel/sys_ni.c
index dbff751..30c393a 100644
--- a/kernel/sys_ni.c
+++ b/kernel/sys_ni.c
@@ -206,3 +206,6 @@ cond_syscall(compat_sys_open_by_handle_at);

/* compare kernel pointers */
cond_syscall(sys_kcmp);
+
+cond_syscall(sys_fbind);
+
diff --git a/net/socket.c b/net/socket.c
index 6e0ccc0..67d9795 100644
--- a/net/socket.c
+++ b/net/socket.c
@@ -1432,6 +1432,31 @@ out:
    return err;
}

+SYSCALL_DEFINE2(fbind, int, fd, int, sk_fd)
+{
+ struct socket *sock;
+ int err, fput_sk, fput_fd;
+ struct file *file;
+
+ sock = sockfd_lookup_light(sk_fd, &err, &fput_sk);
+ if (!sock)
+    return err;
+
+ err = -EBADF;
+ file = fget_raw_light(fd, &fput_fd);
+ if (!file)
+    goto out_put_sk;
+
+ err = -EINVAL;
+ if (sock->ops->fbind)
+    err = sock->ops->fbind(file, sock);
+
+ fput_light(file, fput_fd);
+out_put_sk:
+ fput_light(sock->file, fput_sk);
+ return err;
+}
+
+/*
+ * Bind a name to a socket. Nothing much to do here since it's

```

* the protocol's responsibility to handle the local address.
