
阅码场
YOMOCODE

多进程编程(5)

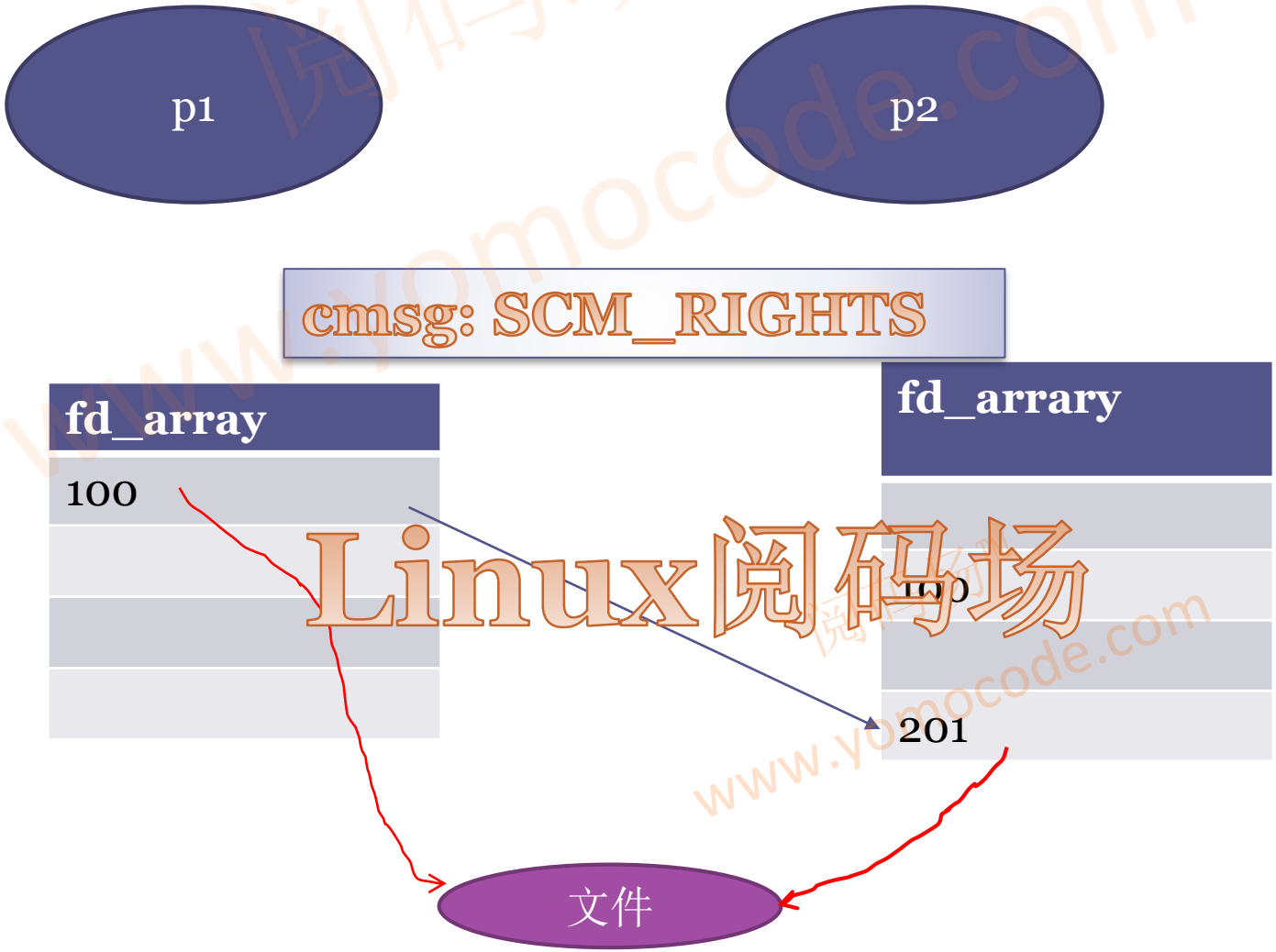
麦当劳喜欢您来，喜欢您再来

扫描关注
Linux阅码场

多进程编程(5)

4.1 跨进程共享文件描述符fd
4.2 memfd_create与共享内存
4.3 dma-buffer:Multimedia/Graphis数据跨进程共享
4.4 dma-buffer跨进程共享
4.5 dma-buffer跨进程共享

socket发送fd



SCM_RIGHTS/cmsg

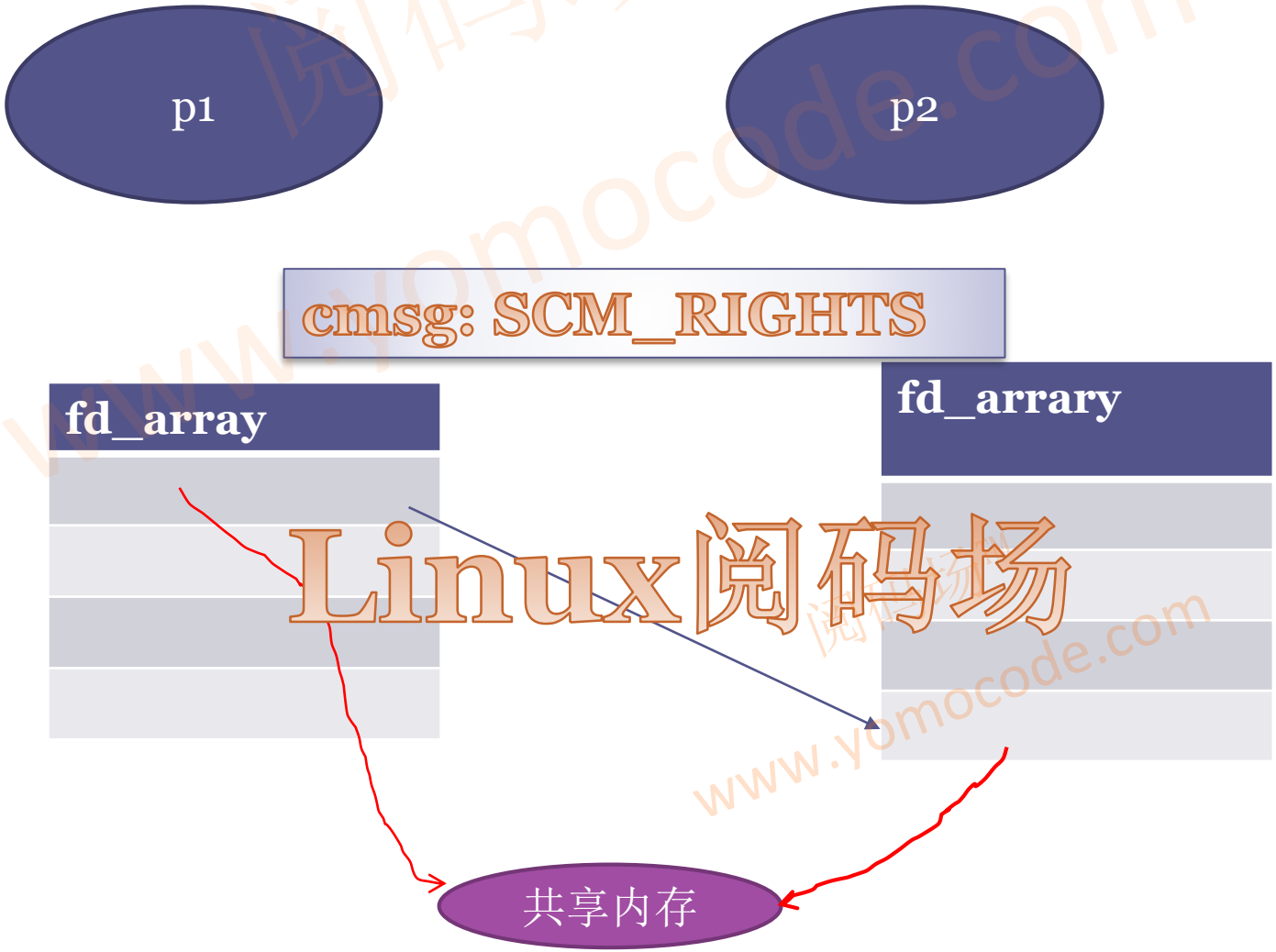
```
static
void send_fd(int socket, int *fds, int n)
{
        struct msghdr msg = {0};
        struct cmsghdr *cmsg;
        char buf[CMSG_SPACE(n * sizeof(int))], data;
        memset(buf, '\0', sizeof(buf));
        struct iovec io = { .iov_base = &data, .iov_len = 1 };

        msg.msg_iov = &io;
        msg.msg_iovlen = 1;
        msg.msg_control = buf;
        msg.msg_controllen = sizeof(buf);

        cmsg = CMSG_FIRSTHDR(&msg);
        cmsg->cmsg_level = SOL_SOCKET;
        cmsg->cmsg_type = SCM_RIGHTS;
        cmsg->cmsg_len = CMSG_LEN(n * sizeof(int));

        memcpy ((int *) CMSG_DATA(cmsg), fds, n * sizeof (int));

        if (sendmsg (socket, &msg, 0) < 0)
                handle_error ("Failed to send message");
}
```

memfd_create

创建一个匿名fd，指向一片内存

memfd_create() creates an anonymous file and returns a file descriptor that refers to it. The file behaves like a regular file, and so can be modified, truncated, memory-mapped, and so on. However, unlike a regular file, it lives in RAM and has a volatile backing storage.

memfd与共享内存



memfd与sealing

add

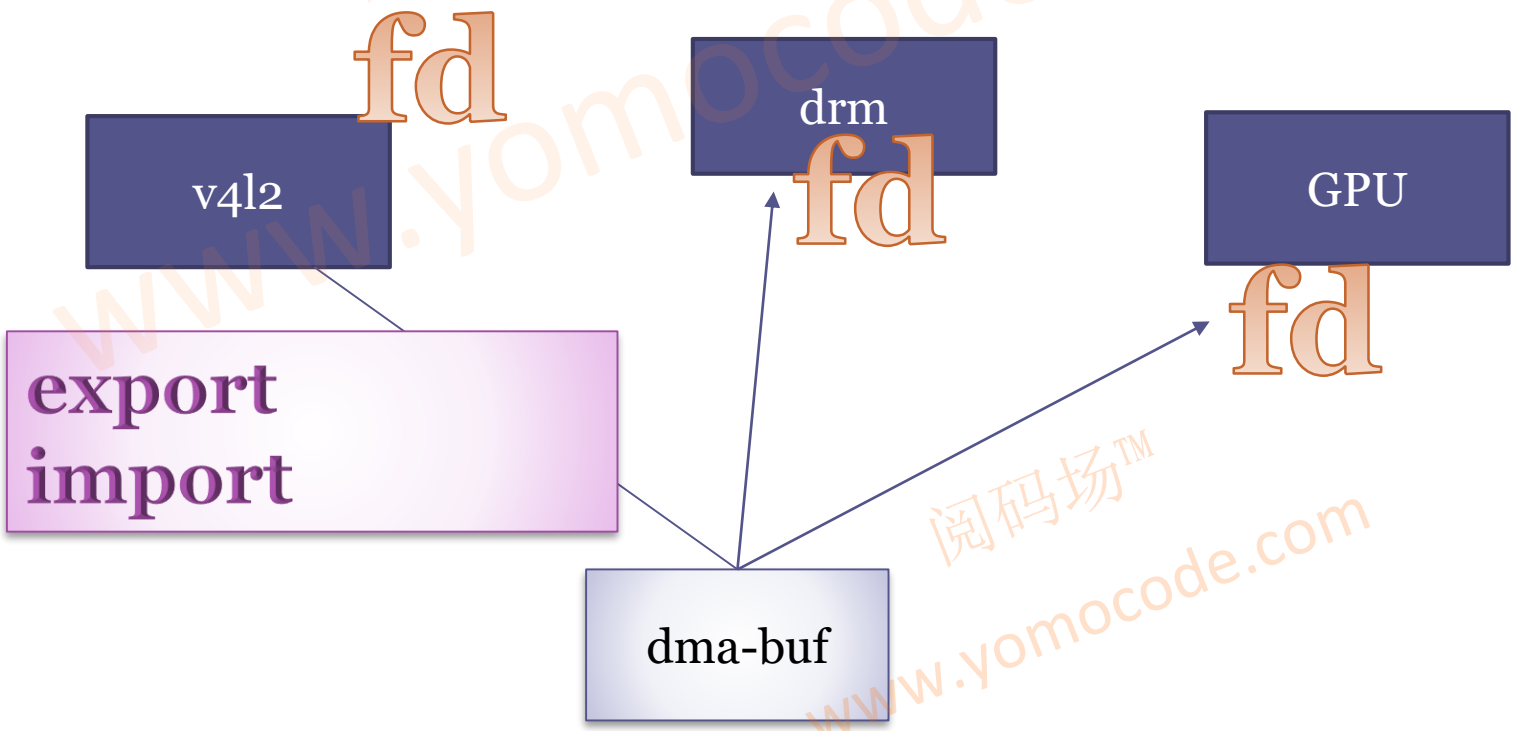
```
fd = memfd_create(name, MFD_ALLOW_SEALING);

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

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

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");
}
```

inspect

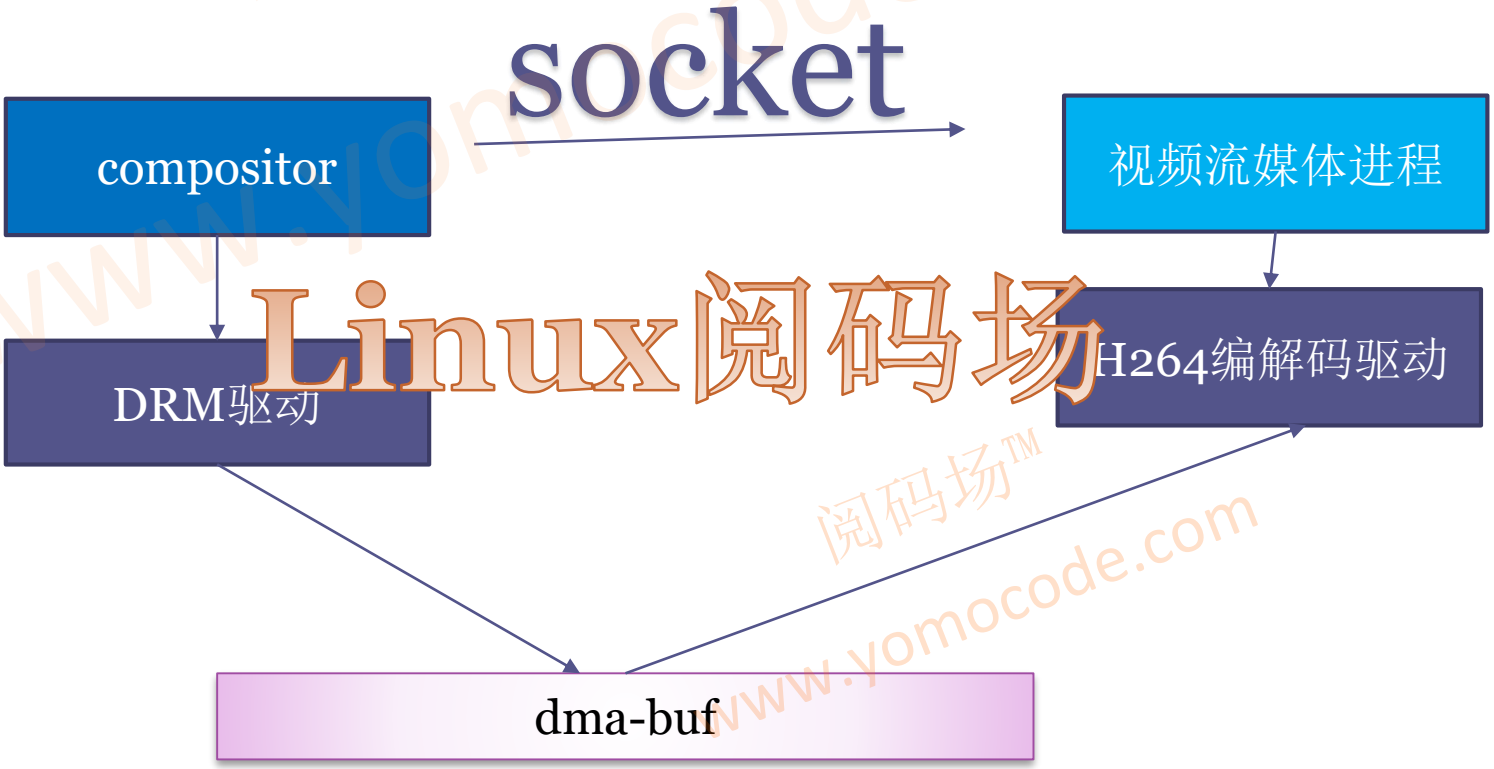
```
seals = fcntl(fd, F_GET_SEALS);
if (seals == -1)
    errExit("fcntl");

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");
```

seals

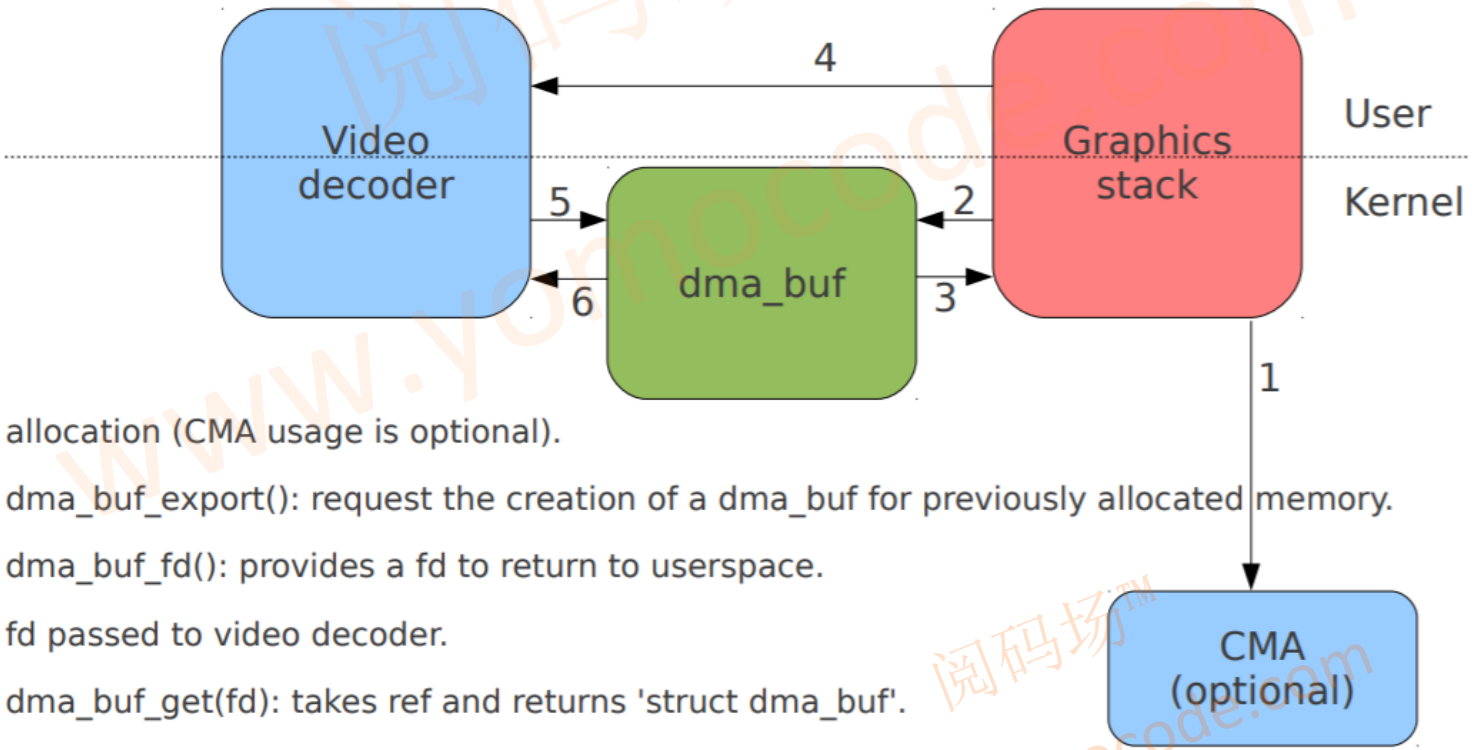
dma-buf跨设备



dma-buf跨进程



dma_buf usage flow



1) allocation (CMA usage is optional).

2) dma_buf_export(): request the creation of a dma_buf for previously allocated memory.

3) dma_buf_fd(): provides a fd to return to userspace.

4) fd passed to video decoder.

5) dma_buf_get(fd): takes ref and returns 'struct dma_buf'.

6) dma_buf_attach() + dma_buf_map_attachment(): to get info for dma

   a) dev->dma_parms should be expanded to tell if receiving device needs contiguous memory or any other special requirements

   b) allocation of backing pages could be deferred by exporting driver until it is known if importing driver requires contiguous memory.. to make things a bit easier on systems without IOMMU

Linaro

谢谢!

阅码场出品