Lab4-1-Extra-Broadcast

题目背景

课下我们在 MOS 系统中实现了 `ipc_send`、`ipc_recv` 函数，实现了进程间的通信。

现在你需要仿照 `ipc_send` 函数在 $user/lib/ipc.c$ 中实现 `ipc_broadcast` 函数，使得调用 `ipc_broadcast` 可以使当前进程向其后代进程（也即当前广播进程的子进程、子进程的子进程、子进程的子进程的子进程...以此类推）发起广播消息，等待其后代进程进入接收状态时，开始实际发送。若进程 A 的控制块中的 `env_id` 和进程 B 的控制块中的 `env_parent_id` 相等，则进程 B 为进程 A 的子进程。

需要完成的新增函数如下：

- `ipc_broadcast` 需要在 $user/lib/ipc.c$ 新增：

  `void ipc_broadcast(u_int val, void * srcva, u_int perm);`

  - 参数：

    `val`： 进程广播传递的具体数值, 与 `ipc_send` 函数中的定义相同。

    `srcva`： 进程广播发送页的对应用户虚地址，与 `ipc_send` 函数中的定义相同。

    `perm`：传递的页面的权限位设置，与 `ipc_send` 函数中的定义相同。

  - 行为描述：

    当前进程向后代进程发起广播消息，等待后代进程进入接收状态（即调用 `ipc_recv`）后，开始实际发送，与 `ipc_send` 行为相似。

题目要求

- 在 $user/include/lib.h$ 中声明以下函数：

  `void ipc_broadcast(u_int val, void * srcva, u_int perm);`

- 在 $user/lib/ipc.c$ 中实现 `ipc_broadcast`

  【注意点】 无需考虑进程间的父子关系成环，测试程序不会出现该情况。

  - 你可以实现 `syscall_ipc_try_broadcast` 系统调用，使其行为类似于 `syscall_ipc_try_send`，但尝试发送给当前进程的所有后代进程。 你也可以尝试在用户空间利用 `envs` 实现相关行为。 发送广播消息时，你可以先等待所有后代进程进入接受状态，再统一进行实际传输，也可以依次等待每个后代进程，一旦其处于接受状态，当即对其进行实际传输。

测试数据范围

实际测试中，进程数量不超过10个。