LAB 4 Challenge!

sfork()

姓名：程浩 学号：515080910012

- question Challenge! Implement a shared-memory fork() called sfork(). This version should have the parent and child share all their memory pages (so writes in one environment appear in the other) except for pages in the stack area, which should be treated in the usual copy-on-write manner. Modify user/forktree.c to use sfork() instead of regular fork(). Also, once you have finished implementing IPC in part C, use your sfork() to run user/pingpongs. You will have to find a new way to provide the functionality of the global thisenv pointer.
- code

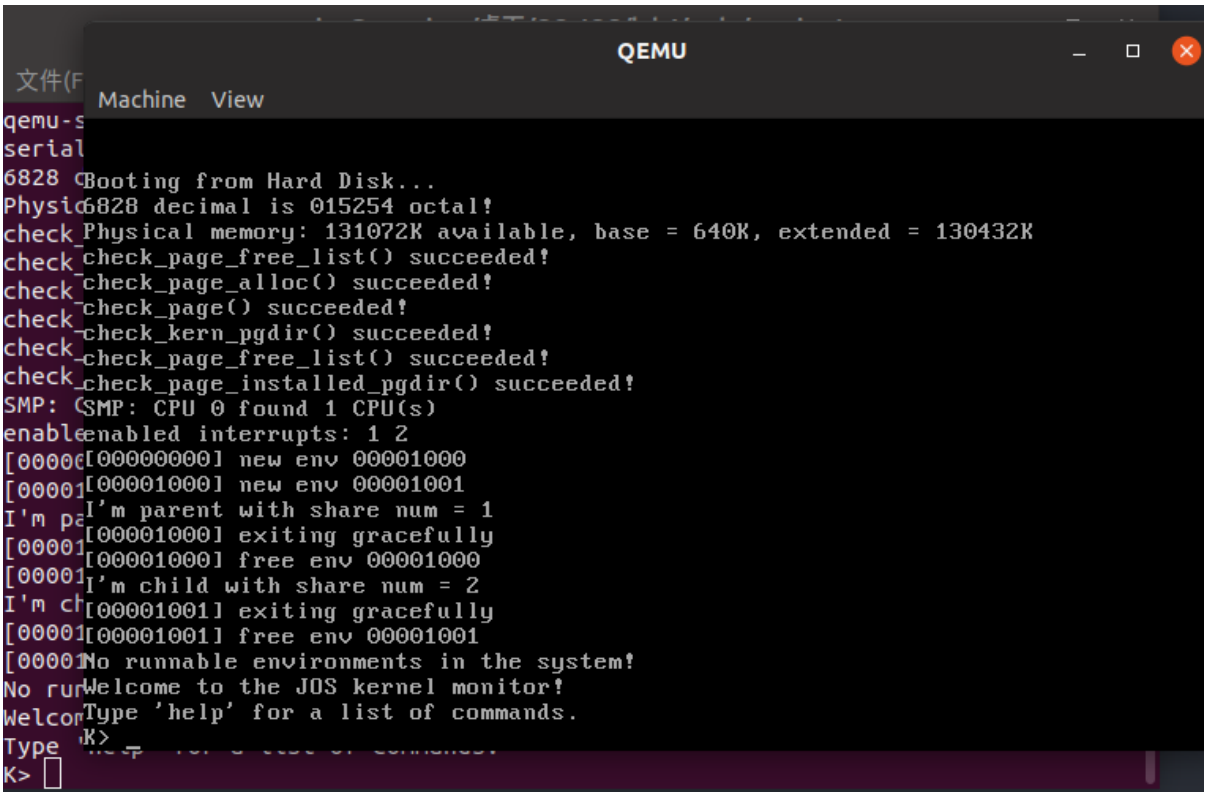
```
lib/fork.c::sfork()

// Challenge!
int
sfork(void)
{
        int r;
        set_pgfault_handler(pgfault);

        envid_t envid = sys_exofork();
        if (envid < 0)
                panic("sys_exofork: %e", envid);

        if (envid == 0)
        {
                thisenv = &envs[ENVX(sys_getenvid())];
                return 0;
        }

        bool stackarea = true;
        for (uint32_t addr = USTACKTOP - PGSIZE; addr >= UTEXT; addr -= PGSIZE)
        {
                if ((uvpd[PDX(addr)] & PTE_P) && (uvpt[PGNUM(addr)] & PTE_P))
                        sduppage(envid, PGNUM(addr), stackarea);
                else
                        stackarea = false;
        }

        if ((r = sys_page_alloc(envid, (void *)(UXSTACKTOP - PGSIZE), PTE_U |
PTE_W | PTE_P)) < 0)
                panic("sfork: sys_page_alloc: %e \n", r);

        extern void _pgfault_upcall(void);
        if ((r = sys_env_set_pgfault_upcall(envid, _pgfault_upcall)) < 0)
```

```
                panic("sfork: sys_env_set_pgfault_upcall: %e \n", r);

        if ((r = sys_env_set_status(envid, ENV_RUNNABLE)) < 0)
                panic("sfork: sys_env_set_status : %e \n", r);

        return envid;

        // panic("sfork not implemented");
        // return -E_INVAL;
}


lib/fork.c::sduppage()
static int
sduppage(envid_t envid, unsigned pn, int cow_enabled)
{
        int r;
        void *addr = (void *)(pn * PGSIZE);
        int perm = PGOFF(uvpt[pn]) & PTE_SYSCALL;

        if (cow_enabled && (perm & PTE_W))
        {
                perm |= PTE_COW;
                perm &= ~PTE_W;

                if ((r = sys_page_map(0, addr, envid, addr, perm)) < 0)
                        panic("sduppage: sys_page_map fail!!! %e\n", r);

                if ((r = sys_page_map(0, addr, 0, addr, perm)) < 0)
                        panic("sduppage: sys_page_map fail!!! %e\n", r);
        }
        else
        {
                if ((r = sys_page_map(0, addr, envid, addr, perm)) < 0)
                        panic("sduppage: sys_page_map fail!!! %e\n", r);
        }

        return 0;
}


user/sfork.c
#include <inc/lib.h>

// parent
int share = 1;

void umain(int argc, char **argv)
{
    int ch = sfork();

    if (ch != 0)
    {
        cprintf ("I'm parent with share num = %d\n", share);
```

```
        // child
        share = 2;
    }
    else
    {
        sys_yield();
        cprintf ("I'm child with share num = %d\n", share);
    }
}
```

基于duppage实现了shared版本的sduppage，目的是使得一个write同时作用在child和parents上，但是在stack area仍然使用COW机制。

- images
  - user/sfork.c



- user/pingpongs.c