

My Running Images

My Code Running – Log 1

```
Booting from Hard Disk..xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start
pagefault++
Page fault by process "initcode" (pid: 1) at 0x0
virt_to_phys: translated "initcode"(1)'s VA 0x0 to PA 0xdf7c000 (pgdir)
Allocated page table at 0x8dfbd000
virt_to_phys: translated "initcode"(1)'s VA 0x0 to PA 0xdf7c000 (shadow_pgdir)
pagefault--
pagefault++
Page fault by process "init" (pid: 1) at 0x0
virt_to_phys: translated "init"(1)'s VA 0x0 to PA 0xdf38000 (pgdir)
Allocated page table at 0x8df34000
virt_to_phys: translated "init"(1)'s VA 0x0 to PA 0xdf38000 (shadow_pgdir)
pagefault--
pagefault++
Page fault by process "init" (pid: 1) at 0x2fe4
virt_to_phys: translated "init"(1)'s VA 0x2fe4 to PA 0xdf35fe4 (pgdir)
Allocated page table at 0x8df34000
virt_to_phys: translated "init"(1)'s VA 0x2fe4 to PA 0xdf35fe4 (shadow_pgdir)
pagefault--
init: starting sh
pagefault++
Page fault by process "init" (pid: 2) at 0x352
virt_to_phys: translated "init"(2)'s VA 0x352 to PA 0xdef3352 (pgdir)
Allocated page table at 0x8deef000
virt_to_phys: translated "init"(2)'s VA 0x352 to PA 0xdef3352 (shadow_pgdir)
pagefault--
pagefault++
Page fault by process "init" (pid: 2) at 0x2fbc
virt_to_phys: translated "init"(2)'s VA 0x2fbc to PA 0xdef0fbc (pgdir)
Allocated page table at 0x8deef000
virt_to_phys: translated "init"(2)'s VA 0x2fbc to PA 0xdef0fbc (shadow_pgdir)
pagefault--
pagefault++
Page fault by process "sh" (pid: 2) at 0x0
virt_to_phys: translated "sh"(2)'s VA 0x0 to PA 0xde6c000 (pgdir)
Allocated page table at 0x8de67000
virt_to_phys: translated "sh"(2)'s VA 0x0 to PA 0xde6c000 (shadow_pgdir)
pagefault--
pagefault++
Page fault by process "sh" (pid: 2) at 0x3fe8
virt_to_phys: translated "sh"(2)'s VA 0x3fe8 to PA 0xde68fe8 (pgdir)
Allocated page table at 0x8de67000
virt_to_phys: translated "sh"(2)'s VA 0x3fe8 to PA 0xde68fe8 (shadow_pgdir)
pagefault--
pagefault++
Page fault by process "sh" (pid: 2) at 0x12f9
virt_to_phys: translated "sh"(2)'s VA 0x12f9 to PA 0xde6a2f9 (pgdir)
Allocated page table at 0x8de67000
```

My Code Running - Usertests

```
mem ok
pipe1 ok
preempt: kill... wait... preempt ok
exitwait ok
rmdot test
rmdot ok
fourteen test
fourteen ok
bigfile test
bigfile test ok
subdir test
subdir ok
linktest
linktest ok
unlinkread test
unlinkread ok
dir vs file
dir vs file OK
empty file name
empty file name OK
fork test
fork test OK
bigdir test
bigdir ok
uio test
pid 551 usertests: trap 13 err 0 on cpu 0 eip 0x3607 addr 0xcf9c--kill proc
uio test done
exec test
ALL TESTS PASSED
$
```

Extra Points

About Kernel Memory Leak Tests

```
Booting from Hard Disk..xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ usertests
usertests starting
arg test passed
createdelete test
createdelete ok
linkunlink test
linkunlink ok
concreate test
concreate ok
fourfiles test
fourfiles ok
sharedfd test
sharedfd ok
bigarg test
bigarg test ok
bigwrite test
bigwrite ok
bigarg test
bigarg test ok
bss test
bss test ok
sbrk test
```

This is Vanilla Code (with no modification without erase something)
's Usertests

But It is not executed and stop here.

I don't know why but, Vanilla code don't have any pagefault

I Think this should be runned.

And also my program acts just like Vanilla code.

We should free shadowpgdir in proc.c

```
for(;;){
    // Scan through table looking for exited children.
    havekids = 0;
    for(p = ptable.proc; p < &ptable.proc[NPROC]; p++){
        if(p->parent != curproc)
            continue;
        havekids = 1;
        if(p->state == ZOMBIE){
            // Found one.
            pid = p->pid;
            kfree(p->kstack);
            p->kstack = 0;
            freevm(p->pgdir);
            p->pid = 0;
            p->parent = 0;
            p->name[0] = 0;
            p->killed = 0;
            p->state = UNUSED;
            release(&ptable.lock);
            return pid;
        }
    }
}
```

when we setupkvm the shadow_pgdir
we should free when it is in zombie
state
but it is not freed in proc.c – wait
function