Answers for csc256-mp3
Junhan Duan

test1.c

output:

- root@OpenWrt:/mnt# ./test1
- [40.359328] pde: not present or ps is 1, stop
- [40.361656] pml4e physical addr: 00000000358cd5b8
- [40.364052] pdpte physical addr: 000000002fe2f875
- [40.366163] pde physical addr: 00000000719ecfa7
- [40.367400] pte physical addr: 000000000f7cbcdb
- [40.368613] final physical addr: 000000000f7cbcdb
- [40.368613]
- [40.370369] Writing MMU error: pde not present
- error writeMMU()
- pml4e: 0x00000000039cb067
- pdpte: 0x00000000039f3067
- pde: 0x0000000000000000
- pte: 0x0000000000000000

If the virtual address is 0, then it is readable but not writable because the pde is not present.
It is because that Because virtual address 0 is reserved for NULL.

test2.c

output:

- root@OpenWrt:/mnt# ./test2 file_8~1.txt
- [91.103430] pdpte: not present or ps is 1, stop
- [91.105829] pml4e physical addr: 00000000320cc627
- [91.107818] pdpte physical addr: 00000000311d7f3c
- [91.109077] pde physical addr: 000000000f7cbcdb
- [91.110311] pte physical addr: 000000000f7cbcdb
- [91.111528] final physical addr: 000000000f7cbcdb
- [91.111528]
- vaddr: 0x72696d7471666667
- pml4e: 0x0000000001cb2067
- pdpte: 0x0000000000000000
- pde: 0x0000000000000000
- pte: 0x0000000000000000
-
- after access
- [91.115928] pml4e physical addr: 00000000320cc627
- [91.117199] pdpte physical addr: 00000000311d7f3c
- [91.118474] pde physical addr: 000000009b47b625
- [91.119706] pte physical addr: 0000000059be096f
- [91.120957] final physical addr: 000000004ccc1404

- [91.120957]
- vaddr: 0x72696d7471666667
- pml4e: 0x0000000001cb2067
- pdpte: 0x0000000001cb6067
- pde: 0x000000000398d067
- pte: 0x8000000003ea8025

The first time we mmap the file and read it, it is not present, however if we tried to access it and then read it, the entries are set. It is because the kernel is using pure demand paging. The page will be generated after it is truly accessed. I think it is because of the fact that the kernel needs to catch the page fault only when read from the disk.

test3.c

output:

- root@OpenWrt:/mnt# ./test3 file_8~1.txt file_8~2.txt
- [121.942438] pml4e physical addr: 0000000053d3e319
- [121.945036] pdpte physical addr: 00000000fa0ff726
- [121.947475] pde physical addr: 000000002857ce69
- [121.948720] pte physical addr: 00000000a2bd4222
- [121.949991] final physical addr: 000000004ccc1404
- [121.949991]
- vaddr1: 0x72696d7471666667
- pml4e1: 0x00000000039d1067
- pdpte1: 0x00000000039b2067
- pde1: 0x00000000039bc067
- pte1: 0x8000000003ea8025
- [121.953768] pml4e physical addr: 0000000053d3e319
- [121.955022] pdpte physical addr: 00000000fa0ff726
- [121.956286] pde physical addr: 000000002857ce69
- [121.957521] pte physical addr: 000000007d52ddcf
- [121.958823] final physical addr: 000000000e80f394
- [121.958823]
- vaddr2: 0x676678756e6a6b71
- pml4e2: 0x00000000039d1067
- pdpte2: 0x00000000039b2067
- pde2: 0x00000000039bc067
- pte2: 0x8000000003eac025
- pte of file 2 has been changed to pte of file 1.
- vaddr1 is now changed to 0x00000000000003e8
- vaddr2 is now 0x00000000000003e8
- vaddr2 is restored to 0x676678756e6a6b71

The two system calls work properly under the test3. The vaddr2 is modified by the entries of vaddr1.

test4.c

output:

- root@OpenWrt:/mnt# ./test4
- [152.283674] pml4e physical addr: 000000000e65290e
- [152.286278] pdpte physical addr: 000000000e7409cf
- [152.288744] pde physical addr: 00000000452ad529
- [152.290046] pte physical addr: 000000005eb9c95a
- [152.291372] final physical addr: 00000000e8694d26
- [152.291372]
- [152.293189] pml4e physical addr: 0000000032198237
- [152.294462] pdpte physical addr: 000000002bfaf183
- [152.295722] pde physical addr: 000000001bc62947
- [152.296986] pte physical addr: 000000002e73eb65
- [152.298266] final physical addr: 00000000e8694d26
- [152.298266]
- Parent:
- pml4e: 0x0000000000090067
- pdpte: 0x0000000000091067
- pde: 0x0000000000092067
- pte: 0x800000000029b865
-
- root@OpenWrt:/mnt# Child:
- pml4e: 0x0000000000097067
- pdpte: 0x0000000000080067
- pde: 0x0000000000081067
- pte: 0x800000000029b845

As shown above, the physical addresses are the same and the page permissions are different since the page entries are different. I think it is because it is the same buffer so that they are in the same physical address, but the virtual address are different because the fork actually create a new process and allocate a new space of memory.