showmappings

by

Cheli Jefry
Amarpal Singh

Implementation

```
int
mon_showmappings(int argc, char **argv, struct Trapframe *tf) {
    if (argc != 3) {
        cprintf("Usage: showmappings START END\n");
        cprintf("\n");
        cprintf("Note: START and END must be in hexadecimal\n");
        return 1;
    }
    long startva, endva;
    pte_t *pte;
    int p, w, u, pwt, pcd, a, d, ps, g;

    startva = ROUNDDOWN(strtol(argv[1], NULL, 16), PGSIZE);
    endva = ROUNDDOWN(strtol(argv[2], NULL, 16), PGSIZE);
    for (long i = startva; i >= startva && i <= endva; i+=PGSIZE) {
        void *va = (void *) i;
        cprintf("0x%08x:", va);
        pte = pgdir_walk(kern_pgdir, (void *) va, 0);
        if (pte == NULL) {
            cprintf(" umapped / no page present");
        } else {
            physaddr_t addr = PTE_ADDR(*pte); // address in page table entry
            p = ((*pte & PTE_P) == PTE_P);
            u = ((*pte & PTE_U) == PTE_U);
            pwt = ((*pte & PTE_PWT) == PTE_PWT);
            pcd = ((*pte & PTE_PCD) == PTE_PCD);
            a = ((*pte & PTE_A) == PTE_A);
            d = ((*pte & PTE_D) == PTE_D);
            ps = ((*pte & PTE_PS) == PTE_PS);
            g = ((*pte & PTE_G) == PTE_G);
            cprintf(" 0x%08x", addr);
            cprintf(" PTE_P: %d", p);
            cprintf(" PTE_U: %d", u);
            cprintf(" PTE_PWT: %d", pwt);
            cprintf(" PTE_PCD: %d", pcd);
            cprintf(" PTE_A: %d", a);
            cprintf(" PTE_D: %d", d);
            cprintf(" PTE_PS: %d", ps);
            cprintf(" PTE_G: %d", g);
        }
        cprintf("\n");
    }
    return 0;
}
```

First we convert the address from a string to a hexadecimal long using strtol.

Implementation

```
int
mon_showmappings(int argc, char **argv, struct Trapframe *tf) {
    if (argc != 3) {
        cprintf("Usage: showmappings START END\n");
        cprintf("\n");
        cprintf("Note: START and END must be in hexadecimal\n");
        return 1;
    }
    long startva, endva;
    pte_t *pte;
    int p, w, u, pwt, pcd, a, d, ps, g;

    startva = ROUNDDOWN(strtol(argv[1], NULL, 16), PGSIZE);
    endva = ROUNDDOWN(strtol(argv[2], NULL, 16), PGSIZE);
    for (long i = startva; i >= startva && i <= endva; i+=PGSIZE) {
        void *va = (void *) i;
        cprintf("0x%08x:", va);
        pte = pgdir_walk(kern_pgdir, (void *) va, 0);
        if (pte == NULL) {
            cprintf(" umapped / no page present");
        } else {
            physaddr_t addr = PTE_ADDR(*pte); // address in page table entry
            p = ((*pte & PTE_P) == PTE_P);
            u = ((*pte & PTE_U) == PTE_U);
            pwt = ((*pte & PTE_PWT) == PTE_PWT);
            pcd = ((*pte & PTE_PCD) == PTE_PCD);
            a = ((*pte & PTE_A) == PTE_A);
            d = ((*pte & PTE_D) == PTE_D);
            ps = ((*pte & PTE_PS) == PTE_PS);
            g = ((*pte & PTE_G) == PTE_G);
            cprintf(" 0x%08x", addr);
            cprintf(" PTE_P: %d", p);
            cprintf(" PTE_U: %d", u);
            cprintf(" PTE_PWT: %d", pwt);
            cprintf(" PTE_PCD: %d", pcd);
            cprintf(" PTE_A: %d", a);
            cprintf(" PTE_D: %d", d);
            cprintf(" PTE_PS: %d", ps);
            cprintf(" PTE_G: %d", g);
        }
        cprintf("\n");
    }
    return 0;
}
```

First we convert the address from a string to a hexadecimal long using strtol.

Next we round the address ranges down to align with PGSIZE.

Implementation

```
int
mon_showmappings(int argc, char **argv, struct Trapframe *tf) {
    if (argc != 3) {
        cprintf("Usage: showmappings START END\n");
        cprintf("\n");
        cprintf("Note: START and END must be in hexadecimal\n");
        return 1;
    }
    long startva, endva;
    pte_t *pte;
    int p, w, u, pwt, pcd, a, d, ps, g;

    startva = ROUNDDOWN(strtol(argv[1], NULL, 16), PGSIZE);
    endva = ROUNDDOWN(strtol(argv[2], NULL, 16), PGSIZE);
    for (long i = startva; i >= startva && i <= endva; i+=PGSIZE) {
        void *va = (void *) i;
        cprintf("0x%08x:", va);
        pte = pgdir_walk(kern_pgdir, (void *) va, 0);
        if (pte == NULL) {
            cprintf(" umapped / no page present");
        } else {
            physaddr_t addr = PTE_ADDR(*pte); // address in page table entry
            p = ((*pte & PTE_P) == PTE_P);
            u = ((*pte & PTE_U) == PTE_U);
            pwt = ((*pte & PTE_PWT) == PTE_PWT);
            pcd = ((*pte & PTE_PCD) == PTE_PCD);
            a = ((*pte & PTE_A) == PTE_A);
            d = ((*pte & PTE_D) == PTE_D);
            ps = ((*pte & PTE_PS) == PTE_PS);
            g = ((*pte & PTE_G) == PTE_G);
            cprintf(" 0x%08x", addr);
            cprintf(" PTE_P: %d", p);
            cprintf(" PTE_U: %d", u);
            cprintf(" PTE_PWT: %d", pwt);
            cprintf(" PTE_PCD: %d", pcd);
            cprintf(" PTE_A: %d", a);
            cprintf(" PTE_D: %d", d);
            cprintf(" PTE_PS: %d", ps);
            cprintf(" PTE_G: %d", g);
        }
        cprintf("\n");
    }
    return 0;
}
```

First we convert the address from a string to a hexadecimal long using strtol.

Next we round the address ranges down to align with PGSIZE.

For every address we call pgdir_walk with create flag set to zero to get the pte entry if it exists.

Implementation

```
int
mon_showmappings(int argc, char **argv, struct Trapframe *tf) {
    if (argc != 3) {
        cprintf("Usage: showmappings START END\n");
        cprintf("\n");
        cprintf("Note: START and END must be in hexadecimal\n");
        return 1;
    }
    long startva, endva;
    pte_t *pte;
    int p, w, u, pwt, pcd, a, d, ps, g;

    startva = ROUNDDOWN(strtol(argv[1], NULL, 16), PGSIZE);
    endva = ROUNDDOWN(strtol(argv[2], NULL, 16), PGSIZE);
    for (long i = startva; i >= startva && i <= endva; i+=PGSIZE) {
        void *va = (void *) i;
        cprintf("0x%08x:", va);
        pte = pgdir_walk(kern_pgdir, (void *) va, 0);
        if (pte == NULL) {
            cprintf(" umapped / no page present");
        } else {
            physaddr_t addr = PTE_ADDR(*pte); // address in page table entry
            p = ((*pte & PTE_P) == PTE_P);
            u = ((*pte & PTE_U) == PTE_U);
            pwt = ((*pte & PTE_PWT) == PTE_PWT);
            pcd = ((*pte & PTE_PCD) == PTE_PCD);
            a = ((*pte & PTE_A) == PTE_A);
            d = ((*pte & PTE_D) == PTE_D);
            ps = ((*pte & PTE_PS) == PTE_PS);
            g = ((*pte & PTE_G) == PTE_G);
            cprintf(" 0x%08x", addr);
            cprintf(" PTE_P: %d", p);
            cprintf(" PTE_U: %d", u);
            cprintf(" PTE_PWT: %d", pwt);
            cprintf(" PTE_PCD: %d", pcd);
            cprintf(" PTE_A: %d", a);
            cprintf(" PTE_D: %d", d);
            cprintf(" PTE_PS: %d", ps);
            cprintf(" PTE_G: %d", g);
        }
        cprintf("\n");
    }
    return 0;
}
```

First we convert the address from a string to a hexadecimal long using strtol.

Next we round the address ranges down to align with PGSIZE.

For every address we call pgdir_walk with create flag set to zero to get the pte entry if it exists.

We use the pte to convert to a physical address using PTE_ADDR.

Implementation

```
int
mon_showmappings(int argc, char **argv, struct Trapframe *tf) {
    if (argc != 3) {
        cprintf("Usage: showmappings START END\n");
        cprintf("\n");
        cprintf("Note: START and END must be in hexadecimal\n");
        return 1;
    }
    long startva, endva;
    pte_t *pte;
    int p, w, u, pwt, pcd, a, d, ps, g;

    startva = ROUNDDOWN(strtol(argv[1], NULL, 16), PGSIZE);
    endva = ROUNDDOWN(strtol(argv[2], NULL, 16), PGSIZE);
    for (long i = startva; i >= startva && i <= endva; i+=PGSIZE) {
        void *va = (void *) i;
        cprintf("0x%08x:", va);
        pte = pgdir_walk(kern_pgdir, (void *) va, 0);
        if (pte == NULL) {
            cprintf(" umapped / no page present");
        } else {
            physaddr_t addr = PTE_ADDR(*pte); // address in page table entry
            p = ((*pte & PTE_P) == PTE_P);
            u = ((*pte & PTE_U) == PTE_U);
            pwt = ((*pte & PTE_PWT) == PTE_PWT);
            pcd = ((*pte & PTE_PCD) == PTE_PCD);
            a = ((*pte & PTE_A) == PTE_A);
            d = ((*pte & PTE_D) == PTE_D);
            ps = ((*pte & PTE_PS) == PTE_PS);
            g = ((*pte & PTE_G) == PTE_G);
            cprintf(" 0x%08x", addr);
            cprintf(" PTE_P: %d", p);
            cprintf(" PTE_U: %d", u);
            cprintf(" PTE_PWT: %d", pwt);
            cprintf(" PTE_PCD: %d", pcd);
            cprintf(" PTE_A: %d", a);
            cprintf(" PTE_D: %d", d);
            cprintf(" PTE_PS: %d", ps);
            cprintf(" PTE_G: %d", g);
        }
        cprintf("\n");
    }
    return 0;
}
```

First we convert the address from a string to a hexadecimal long using strtol.

Next we round the address ranges down to align with PGSIZE.

For every address we call pgdir_walk with create flag set to zero to get the pte entry if it exists.

We use the pte to convert to a physical address using PTE_ADDR.

We then mask off the relevants bits to determine the permissions and cprintf our results.

Implementation

```
int
mon_showmappings(int argc, char **argv, struct Trapframe *tf) {
    if (argc != 3) {
        cprintf("Usage: showmappings START END\n");
        cprintf("\n");
        cprintf("Note: START and END must be in hexadecimal\n");
        return 1;
    }
    long startva, endva;
    pte_t *pte;
    int p, w, u, pwt, pcd, a, d, ps, g;

    startva = ROUNDDOWN(strtol(argv[1], NULL, 16), PGSIZE);
    endva = ROUNDDOWN(strtol(argv[2], NULL, 16), PGSIZE);
    for (long i = startva; i >= startva && i <= endva; i+=PGSIZE) {
        void *va = (void *) i;
        cprintf("0x%08x:", va);
        pte = pgdir_walk(kern_pgdir, (void *) va, 0);
        if (pte == NULL) {
            cprintf(" umapped / no page present");
        } else {
            physaddr_t addr = PTE_ADDR(*pte); // address in page table entry
            p = ((*pte & PTE_P) == PTE_P);
            u = ((*pte & PTE_U) == PTE_U);
            pwt = ((*pte & PTE_PWT) == PTE_PWT);
            pcd = ((*pte & PTE_PCD) == PTE_PCD);
            a = ((*pte & PTE_A) == PTE_A);
            d = ((*pte & PTE_D) == PTE_D);
            ps = ((*pte & PTE_PS) == PTE_PS);
            g = ((*pte & PTE_G) == PTE_G);
            cprintf(" 0x%08x", addr);
            cprintf(" PTE_P: %d", p);
            cprintf(" PTE_U: %d", u);
            cprintf(" PTE_PWT: %d", pwt);
            cprintf(" PTE_PCD: %d", pcd);
            cprintf(" PTE_A: %d", a);
            cprintf(" PTE_D: %d", d);
            cprintf(" PTE_PS: %d", ps);
            cprintf(" PTE_G: %d", g);
        }
        cprintf("\n");
    }
    return 0;
}
```

First we convert the address from a string to a hexadecimal long using strtol.

Next we round the address ranges down to align with PGSIZE.

For every address we call pgdir_walk with create flag set to zero to get the pte entry if it exists.

We use the pte to convert to a physical address using PTE_ADDR.

We then mask off the relevants bits to determine the permissions and cprintf our results.

Debugging:
We had to ensure that our current va (i) is greater than the startva to prevent an infinite loop.