INDIAN INSTITUTE OF TECHNOLOGY MADRAS

Operating Systems : Lab 3

SATHVIK JOEL K

CS19B025

September 15, 2021

CONTENTS

1 Printing Page Table Entries

The following steps are followed

- **vmprint()** is written in *kernel/vm.c*, it calls a helper function **vmprinthelper()** the function is very similar to **freewalk()** except that here we just retain the recrusive part and ignore the freeing of pages
- The code for **vmprint** is as follows

```
// >> kernel/vm.c

printf("page table: %p", pagetable);
vmprinthelper(pagetable, 0);
```

- The code for **vmprinthelper** is as follows

```
// >> kernel/vm.c

void
vmprinthelper(pagetable_t pagetable, int level)
{
  // there are 2^9 = 512 PTEs in a page table.
  for(int i = 0; i < 512; i++){
    pte_t pte = pagetable[i];
    if( pte & PTE_V ){
        // this PTE points to a lower-level page table.
        uint64 child = PTE2PA(pte);
        for(int i = 0 ; i <= level ; i++)
        {
                printf("..");
                if( i != level) printf(" ");
        }
        printf("%d: pte  %p pa %p\n",i, pte, child);
        if((pte & (PTE_R|PTE_W|PTE_X)) == 0){
                vmprinthelper((pagetable_t)child, level+1);
        }
    }
  }
}
```

- Every page directory or page table has 512 entries. The above code iterates over all the pages and whenever an entry is valid it prints and if it points to a lower level page table it recrusively calls the same function.

- The function prototype is also added in **defs.h**
- in **exec.c** the following code is addded so that **vmprint** is called from the first user process

```
// >> kernel/exec.c

if( p->pid ==  1 )
{
vmprint(p->pagetable);
}
```

2 ELIMINATE ALLOCATION FROM SBRK()

- sys_sbrk() is modified not to call **grow_proc()**, it just increments the size of the process by n

```
// >kernel/sysproc.c

sys_sbrk(void)
{
  int addr;
  int n;

  if(argint(0, &n) < 0)
    return -1;
  addr = myproc()->sz;

  // The following line increments the size
  myproc()->sz += n;

  //if(growproc(n) < 0)
  //  return -1;

  return addr;
}
```

3 LAZY ALLOCATION

- in **trap.c** the **usertrap()** function is improved to handle pagefaults.
- Values 13 and 14 in Supervisor Cause register (scause) implies Load Page fault or store page fault respectively.

- Code similar to **uvmalloc()** is used here to allocate a page and map it whenever a page fault occurs
- In **vm.c** file the function **uvmumap()** is modified to not panic

```
if( r_scause() == 13 || r_scause() == 15)
{
    int va_int = PGROUNDDOWN(r_stval());
    char* mem = kalloc();
    if( mem == 0 )
    {
        printf("page not allocated: error");
        p->killed = 1 ;
    }
    else
    {
        memset(mem , 0 , PGSIZE );
        if (mappages(p->pagetable, va_int, PGSIZE, (uint64)mem,
                                PTE_W|PTE_X|PTE_R|PTE_U) != 0 )
        {
           p->killed = 1;
        }

    }
}
```