CS3523: Operating Systems-2

Programming Assignment 6

Paging

Part-1:Implementing demand paging:

The steps followed to implement demand paging:

- Modify exec.c

```
if((sz = allocuvm(pgdir, sz, ph.vaddr + ph.filesz)) == 0)

goto bad;

sz = sz + ph.memsz - ph.filesz;
```

- Modify trap.c. A new case for demand paging.
- For the function used in trap.c, since the mappages is static, a non-static function map_pages is created in vm.c.
- Add the declaration of map_pages in def.s

```
int map_pages(pde_t *pgdir, void *va, uint size, uint pa, int perm);
```

- A user program(mydemandPage.c) to exercise the demand paging is created
- Make modifications in the make file. In UPROGS,

```
_mydemandPage\
```

- Another modification in the make file. In extras, give the name of the source file

```
mydemandPage.c\
```

- Typing mydemandPage to xv6 shell prompt will print the required output

  Output for global array size 3000:

```
$ mydemandPage
global addr from user space: B00
page fault occurred, doing demand paging for address: 0x1000
pgdir entry num: 0, Pgt entry num: 0, Virtual addr: 0x0, Physical addr: 0xdee2000
pgdir entry num: 0, Pgt entry num: 1, Virtual addr: 0x1000, Physical addr: 0xdfbc000
pgdir entry num: 0, Pgt entry num: 2, Virtual addr: 0x5000, Physical addr: 0xdedf000
page fault occurred, doing demand paging for address: 0x2000
pgdir entry num: 0, Pgt entry num: 0, Virtual addr: 0x0, Physical addr: 0xdee2000
pgdir entry num: 0, Pgt entry num: 1, Virtual addr: 0x1000, Physical addr: 0xdfbc000
pgdir entry num: 0, Pgt entry num: 2, Virtual addr: 0x2000, Physical addr: 0xdf76000
pgdir entry num: 0, Pgt entry num: 3, Virtual addr: 0x5000, Physical addr: 0xdedf000
page fault occurred, doing demand paging for address: 0x3000
Printing final page table:
pgdir entry num: 0, Pgt entry num: 0, Virtual addr: 0x0, Physical addr: 0xdee2000
pgdir entry num: 0, Pgt entry num: 1, Virtual addr: 0x1000, Physical addr: 0xdfbc000
pgdir entry num: 0, Pgt entry num: 2, Virtual addr: 0x2000, Physical addr: 0xdf76000
pgdir entry num: 0, Pgt entry num: 3, Virtual addr: 0x3000, Physical addr: 0xdfbf000
pgdir entry num: 0, Pgt entry num: 4, Virtual addr: 0x5000, Physical addr: 0xdedf000
Value: 2
$
```

  Observations:

  - For global array size 1000, number of page faults are 1
  - For global array size 3000, number of page faults are 3
  - For global array size 5000, number of page faults are 5

- For global array size 10000, number of page faults are 10

The number of page faults increases as the global array size increases.

Part-2:Implementing copy-on-write:

The steps followed to implement copy-on-write:

- A new function copyuvm_cow is created which is similar to copyuvm in vm.c and its declaration is added in defs.h. It first maps the kernel to the child, it does not clone the use-mode memory. It maps the parent's physical frame read-only in both the parent and child by clearing the PTE_w flag from the corresponding page table entries of parent and child. The function flushes the TLB because the active page table is changed. copyuvm in fork is replaced with copyuvm_cow in proc.c
- When any process tries to write, the CPU will raise a page fault exception. To deal with this exception a function handle_pgflt() is written in vm.c. It first reads the faulting address from cr2 (read_cr2). It then gets the physical address of the currently mapped physical frame from the page table and clones the frame into a new page. It maps this new page into the current process writable and flushes the TLB.
- Two functions (`read_cr2` and `flush_tlb_all`) into `trapasm.s`

```
.globl read_cr2

  read_cr2:

  movl %cr2, %eax

       ret

.globl flush_tlb_all

  flush_tlb_all:

  movl %cr3, %eax

  movl %eax, %cr3

       Ret
```

- Their declarations are added to defs.h

```
uint read_cr2 (void);

void flush_tlb_all (void);
```

- A user program to check the working of cow (myCOW.c)
- Make modifications in the make file. In UPROGS,

```
_myCOW\
```

- Another modification in the make file. In extras, give the name of the source file

```
myCOW.c\
```

- Typing myCOW to xv6 shell prompt will print the required output.

Reference used for copy-on-write: link