xv6 Memory Introspection System Calls Tutorial

1. Introduction

- Goal: Learn how to inspect process memory from user space.
- Focus: Implement **numvp(), numpp(), getptsize()** system calls.

2. Background

- **Virtual pages (VP):** Pages allocated in the process's virtual address space.
- **Physical pages (PP):** Actual memory frames mapped to the process.
- **Page tables:** Structures that map virtual addresses → physical addresses.
- **xv6 structures:**
- `struct proc` (PCB)
- `pgdir` → page directory pointer
- `walkpgdir()` → helper to walk the page table

3. System Call Design

3.1 numvp()

- Returns the number of virtual pages in the **user space**.
- Formula:

```
num_virtual_pages = ceil(p->sz / PGSIZE) + 1 // stack guard
```

3.2 numpp()

- Returns the number of **physical pages** allocated.
- Walk `pgdir` using `walkpgdir()`. Count **present** pages (`PTE_P` flag).

3.3 getptsize()

- Returns the number of **pages used by the page table**.
- Count **outer page directory** + **inner page table pages**.

4. Implementation in xv6

4.1 syscall numbers (`syscall.h`)

```
#define SYS_numvp 22
#define SYS_numpp 23
#define SYS_getptsize 24
```

4.2 Declare in `user.h`

```
int numvp(void);
int numpp(void);
int getptsize(void);
```

4.3 Implement in `sysproc.c`

```
int sys_numvp(void) {
    struct proc *p = myproc();
    return (p->sz + PGSIZE - 1)/PGSIZE + 1;
}

int sys_numpp(void) {
    struct proc *p = myproc();
    pte_t *pte;
    int count = 0;
    for(uint a = 0; a < p->sz; a += PGSIZE){
        pte = walkpgdir(p->pgdir, (void*)a, 0);
        if(pte && (*pte & PTE_P)) count++;
    }
    return count;
}

int sys_getptsize(void) {
    struct proc *p = myproc();
    int count = 1; // outer page directory
    for(int i=0;i<NPDENTRIES;i++)
        if(p->pgdir[i] & PTE_P) count++;
    return count;
}
```

4.4 Add to syscall dispatch (`syscall.c`)

```
[SYS_numvp] = sys_numvp,
[SYS_numpp] = sys_numpp,
[SYS_getptsize] = sys_getptsize,
```

5. User Program Test

Create `memtest.c`:

```c
#include "types.h"
#include "stat.h"
#include "user.h"

int main(void) {
    printf(1, "Virtual pages: %d\n", numvp());
    printf(1, "Physical pages: %d\n", numpp());
    printf(1, "Page table pages: %d\n", getptsize());
    exit();
}
```

- Add `_memtest` to `UPROGS` in `Makefile` (last line without backslash).

6. Compile and Run

```
make clean
make qemu
```

Inside xv6 shell:

```
$ memtest
```

Sample output:

```
Virtual pages: 10
Physical pages: 10
Page table pages: 2
```

7. Notes

- Initially, virtual pages = physical pages.
- Stack guard counts as 1 extra virtual page.
- Page table pages include outer + inner tables.