Programming Assignment 3 Documentation File

Alex Lerma & Sahand Zeinali

Group 7

CS 450

Professor Leung

04/08/2017

Project Goal

Create a new system call which displays the current memory usage of the process calling it, in terms of number of pages allocated to the process. Further display the number of pages that are accessible and writable by user program. Make sure you write several test programs which tries to allocate varying amounts of memory (until no more memory can be allocated) and record how allocation affects page table for the process. Your modifications must not prevent xv6 from functioning normally.

To achieve this goal we created these files memcount.c and memcounttest.c . And we added code into these files sysproc.c and kalloc.c.

Code Added into memcount.c:

```
#include "types.h"
#include "user.h"
#define INT_MAX 2147483647

struct test {
        int a;
        int arr[50];
};
```

```
void test(int testsize) {
        int size = sizeof(struct test) * testsize;
        printf(1,"Allocated a pointer to test struct of size  %d\n", size);
        struct test *x = (struct test *)malloc(size);
        x->a = INT_MAX;
}

int main(int argc, char **argv) {
        int testsize = atoi(argv[1]);
        if(testsize == 0) {
                exit();
        }
        printf(1, "Test size: %d\n", testsize);
        printf(1, "Initial Memory Status:\n");
        memcount();
        test(testsize);
        printf(1, "Final Memory Status:\n");
        memcount();
        exit();
}
```

Code Added into memcounttest.c:

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

static int run_all() {
        char *testsizes[] = {"1", "10", "100", "1000", "10000","50000", "100000", "500000"
,"1000000", "10000000"};
        int n_sizes = 10;
        int i;
```

```
        for(i = 0; i < n_sizes; i++) {
                if (fork() == 0) {
                        exec("./memcount", &(testsizes[i]));
                }
                wait();
        }
        return 0;
}

int main(int argc, char **argv) {
        run_all();
        exit();
}
```

Code Added into sysproc.c:

```
int sys_memcount(void) {
  uint pages = PGROUNDUP(proc->sz) / PGSIZE;
  uint present = 0, writable = 0, user = 0;
  uint pdef, ptef;
  pde_t *pde, *pte;

  int i, j;
  for (i = 0; i < NPDENTRIES; i++) {
    pde = &proc->pgdir[i];
    pdef = PTE_FLAGS(*pde);

    //check if page directory entry is valid
    if(is_valid_pde(&pdef)) {
      for(j = 0; j < NPTENTRIES; j++) {
```

```
      pte = &pde[j];
      ptef = PTE_FLAGS(*pte);
      check_flags(&present, &writable, &user, &ptef);
    }
   }
  }

  cprintf("\n");
  cprintf("*********************************************\n");
  cprintf("*************** Memory Count ***************\n");
  cprintf("*********************************************\n");
  cprintf(" - Pages used by current process: %d\n", pages);
  cprintf(" - Pages free: %d\n", kfreepagecount());
  cprintf(" - Pages present: %d\n", present);
  cprintf(" - Pages writable: %d\n", writable);
  cprintf(" - Pages user accessible: %d\n", user);
  cprintf("*********************************************\n");
  cprintf("*********************************************\n");
  cprintf("*********************************************\n");
  cprintf("\n");

  return 0;
}
```

Code Added into kalloc.c:

```
int kfreepagecount(void) {
  struct run *r;
  uint freepages = 0;

  if(kmem.use_lock)
    acquire(&kmem.lock);
```

```
 r = kmem.freelist;
 while(r) {
   freepages++;
   r = r->next;
 }

 if(kmem.use_lock)
   release(&kmem.lock);

 return freepages;
}
```

And other files were tweaked so that the system call was added to xv6 successfully. Thesefiles are syscall.h, user.h, defs.h

Code Added into syscall.h:

```
#define SYS_memcount 23
#define NUM_CALLS  23
```

Code Added into user.h:

```
int memcount(void);
```

Code Added into defs.h:

```
int                 kfreepagecount(void);
```