Aaron Sigal
Lab3 Report
CS153

Overview:
- Rework the virtual memory system
- Enable stack expansion via dynamic page allocation

The Guts (of Proc)

```
unsigned int pages;          // Lab 3: Number of pages allocated for the stack
```

proc.h: New member variable of Struct proc

```
curproc->pages = 1; // Lab 3
```

exec.c: exec(): the aforementioned proc member variable is initialized

```
// Lab 3
// Allocate two pages at the next page boundary.
// Make the first inaccessible.  Use the second as the user stack.
sz = PGROUNDUP(sz);
if((sp = allocuvm(pgdir, STACK - PGSIZE, STACK)) == 0)
  goto bad;
```

exec.c: exec(): the stack pointer correctly points to the start (top) of the new stack location

```
#define STACK 0x7fffffff        // Lab 3: Stack start address
```

memlayout.h: STACK constant defines the memory location of the new stack

```
// Lab 3
struct proc *curproc = myproc(); // Get current proc

unsigned int pages = curproc->pages;      // Grab num pages

for( i = PGROUNDUP(STACK- PGSIZE); i > STACK - pages * PGSIZE; i-=PGSIZE) {
  if((pte = walkpgdir(pgdir, (void *) i, 0)) == 0)
    panic("copyuvm: pte should exist");
  if(!(*pte & PTE_P))
    panic("copyuvm: page not present");
  pa = PTE_ADDR(*pte);
  flags = PTE_FLAGS(*pte);
  if((mem = kalloc()) == 0)
    goto bad;
  memmove(mem, (char*)P2V(pa), PGSIZE);
  if(mappages(d, (void*)i, PGSIZE, V2P(mem), flags) < 0)
    goto bad;
}
// end of Lab3
```

vm.c: copyuvm(): Iterate over the stack pages

```
case T_PGFLT: // Lab 3
  if (rcr2() <= PGROUNDUP(STACK - PGSIZE*myproc()->pages) &&
      rcr2() >= PGROUNDUP(STACK - PGSIZE*(myproc()->pages + 1))) {


    allocuvm(myproc()->pgdir,
             PGROUNDUP(STACK - PGSIZE*(myproc()->pages + 1)),
             PGROUNDUP(STACK - PGSIZE*(myproc()->pages)));

    myproc()->pages++;
    cprintf("Allocating a new page...\n");
  }

  break;
```

trap.c: trap(): handle page fault. Check if a requested address is in the next un-allocated page. If it is, allocate that page and increment the page counter.

Syscall.c:

Removed all instances of range checking for stack addresses. This is because they were designed to check the stack in its unmodified form (where it couldn't be resized and was located somewhere else). This introduces some additional hazards, but makes running the system possible without major syscall reworks.