Operating System Concepts
COP4610.02
Mini Project 3

Noah Baldwin
Cody Carroll
Paul Teleweck

Work Breakdown:

Names	Code	Report	Documentation	Presentation
Noah Baldwin	33.33%	33.33%	33.33%	33.3%
Cody Carroll	33.33%	33.33%	33.33%	33.3%
Paul Teleweck	33.33%	33.33%	33.33%	33.3%

Abstract:

In this project, you'll be changing xv6 to support a feature virtually every modern OS does: causing an exception to occur when your program dereferences a null pointer. The goals of the project are as follows:

- To familiarize you with the xv6 virtual memory system.
- To add a few new Virtual Memory (VM) features to xv6 that are common in modern OSs.

Changes made:

vm.c

```
if((d = setupkvm()) == 0)
  return 0;
for(i = PGSIZE; i < sz; 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);
  if((mem = kalloc()) == 0)
    goto bad;
  memmove(mem, (char*)pa, PGSIZE);
  if(mappages(d, (void*)i, PGSIZE, PADDR(mem), PTE_W|PTE_U) < 0)
    goto bad;
}
return d;
```

The for loop was initially i=0 and we changed this loop to equal PGSIZE. This forces program to start on the next available page.

exec.c

```
// Load program into memory.
sz = PGSIZE;
for(i=0, off=elf.phoff; i<elf.phnum; i++, off+=sizeof(ph)){
  if(readi(ip, (char*)&ph, off, sizeof(ph)) != sizeof(ph))
    goto bad;
  if(ph.type != ELF_PROG_LOAD)
    continue;
  if(ph.memsz < ph.filesz)
    goto bad;
  if((sz = allocuvm(pgdir, sz, ph.va + ph.memsz)) == 0)
    goto bad;
  if(loaduvm(pgdir, (char*)ph.va, ip, ph.offset, ph.filesz) < 0)
    goto bad;
}
iunlockput(ip);
ip = 0;
```

Changed sz = 0 to sz = PGSIZE. Setting size to allocuvm, sz corresponds to the current location. Ph.va + ph.memsz corresponds to the expected location.

syscall.c

```c
// Fetch the int at addr from process p.
int
fetchint(struct proc *p, uint addr, int *ip)
{
  if(addr >= p->sz || addr+4 > p->sz || addr == 0)
    return -1;
  *ip = *(int*)(addr);
  return 0;
}
```

Within syscall.c we added piping to set the address == 0 initially.

makefile.mk

```
# user programs
USER_PROGS := \
        cat\
        echo\
        forktest\
        grep\
        init\
        kill\
        ln\
        ls\
        mkdir\
        rm\
        sh\
        stressfs\
        tester\
        usertests\
        wc\
        hello\
        zombie
```

Added user program “hello”

```
# location in memory where the program will be loaded
USER_LDFLAGS += --section-start=.text=0x1000
```

Change text section start from 0 to 4096 (1000 in hex)
Setting the address to 0x1000 makes the program start on a different page (not 0).

hello.c

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

int main(void) {
    int* hello;
    hello = NULL;
    printf(1, "This will not work %p\n", *hello);

    exit();
}
```

Created a user program that adds a pointer for the system call.

Output

```
QEMU
.              1 1 512
..             1 1 512
wc             2 2 9776
README         2 3 1793
usertests      2 4 34828
cat            2 5 9448
rm             2 6 9056
kill           2 7 9008
forktest       2 8 5784
hello          2 9 8860
init           2 10 9292
ls             2 11 10552
grep           2 12 10568
ln             2 13 8980
tester         2 14 8924
echo           2 15 8992
mkdir          2 16 9072
sh             2 17 16180
stressfs       2 18 9216
zombie         2 19 8780
console        3 20 0
$ hello
pid 4 hello: trap 14 err 4 on cpu 1 eip 0x101b addr 0x0--kill proc
$ _
```

Causes an error because we are attempting to access a location that does not exist (De-referencing a NULL pointer).