Student 1 - aavit004
Student 2 - cfeng017
Team #19

Lab3 - Writeup

1. lab3_test.c: Changed all the exit(0) calls to exit();
2. exec.c: change the parameters when calling allocuvm so that the pages start below the KERNBASE
    a. Changed the second parameters to allocuvm() so it located the top of the user part of memory and changed the third parameter to the virtual address of the last page we mapped to.
    b. Changed the second parameter when calling clearpteu() so that there is extra page
    c. Set the stack pointer (sp) to KERNBASE - 1
    d. Set the curproc's stack page to 1

```
...
if(allocuvm(pgdir, PGROUNDDOWN(KERNBASE - 1), (KERNBASE - 1)) == 0)
   goto bad;
 clearpteu(pgdir, (char*)(KERNBASE-1  - 2*PGSIZE));
 sp = KERNBASE -1;
 curproc->stack_pages = 1; ...
```

3. syscall.c:
    a. For all the following functions: changed the checks that check against sz to our new location (KERNBASE -1)
        i. fetchinit()

```
...
 if(addr >= (KERNBASE -1)|| addr+4 > (KERNBASE -1)) …
```

        ii. fetchstr()

```
...
if(addr >= (KERNBASE - 1)) ...
ep = (char*)(KERNBASE -1); …
```

        iii. argptr()

```
if(size < 0 || (uint)i >= (KERNBASE - 1) || (uint)i+size >(KERNBASE - 1))
...
```

4. vm.c
    a. copyuvm: takes into account the new stack and it iterates over the stack pages

```
for(i = PGROUNDUP(KERNBASE - 1 - (myproc()->stack_pages * PGSIZE)); i <
(KERNBASE - 1); i += PGSIZE
)
...//copy the existing for loops contents
```

5. proc.h
    a. Add member, int stackpages, to keep track of what page the current process is on.
6. trap.c
    a. Add a case to catch page faults:

```
case T_PGFLT: ;
   uint location = rcr2(); //We read in the address in the cr2 register

   if(allocuvm(myproc()->pgdir, PGROUNDDOWN(location), location) ==
0)//Page is not under the current bottom of the stack
     exit();
   else{
     cprintf("expanding\n");
     myproc()->stack_pages++; //If not we grow the stack
   }
   break;
```