Lab1

Kai Wen Tsai (ktsai017)

Partner: Khoa Vo (kvo018)

a. Void exits(int)
   a. Find out all the exit() related functions, copy in to the new exits()
      i. Defs.c
         1. Register exits() function
      ii. Trap.c
         1. Create syscall for exit
      iii. Syscall.c
         1. Create sys_exits to map the syscall function
      iv. Syscall.h
         1. Define the index for sys_exits
      v. Sysproc.c
         1. Assigned first argument to status pointer
      vi. User.h
         1. Assigned exits()
      vii. Usys.S
         1. Map sys_exits()
      viii. Proc.h
         1. Create variable status
      ix. Proc.c
         1. Create a exits(int status) function
            a. Set current process status

```
curproc->status = status;
```

   b. Int wait(int* status)
      i. Defines wait just like exits()
      ii. Sysproc.c
         1. Assigned first argument to status

```
argptr(0, (void*)&status, sizeof(status));
```

            a.
   c. Int waitpid(int pid, int *status, int options)
      i. Defines waitpid just like wait
      ii. Sysproc.c
         1. Assigned pointer of pid, status, and options of waitpid

a.

```
sys_waitpid(void)
){
  int pid;
  int *status;
  int options;
  argint(0, &pid);
  argptr(1, (void*)&status, sizeof(status));
  argint(2, &options);

  //argptr(0, (void*)&status, sizeof(status));
  return waitpid(pid, status, options);
}
```

iii. Proc.c

1. If the current pid is not equal to the pid we pass in, don't return pid

2.

```
for(;;){
  // Scan through table looking for exited children.
  havekids = 0;
  for(p = ptable.proc; p < &ptable.proc[NPROC]; p++){ //wait for the process
    if(p->pid != pid)
      continue;
    havekids = 1;

        if(p->state == ZOMBIE){
          // Found one.
          pid = p->pid;
          kfree(p->kstack);
          p->kstack = 0;
          freevm(p->pgdir);
          p->pid = 0;
          p->parent = 0;
          p->name[0] = 0;
          p->killed = 0;
          p->state = UNUSED;
          release(&ptable.lock);
          *status = p->status;
          return pid;
        }
```