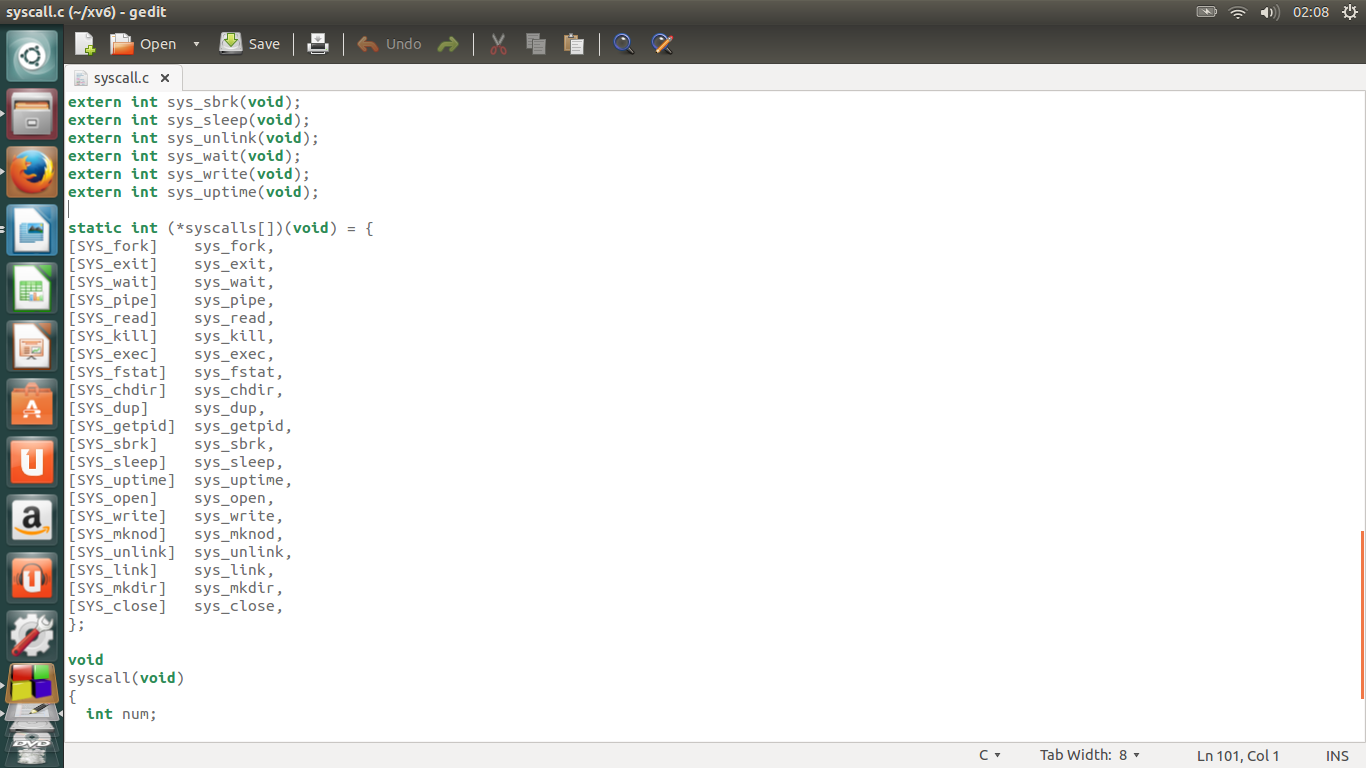
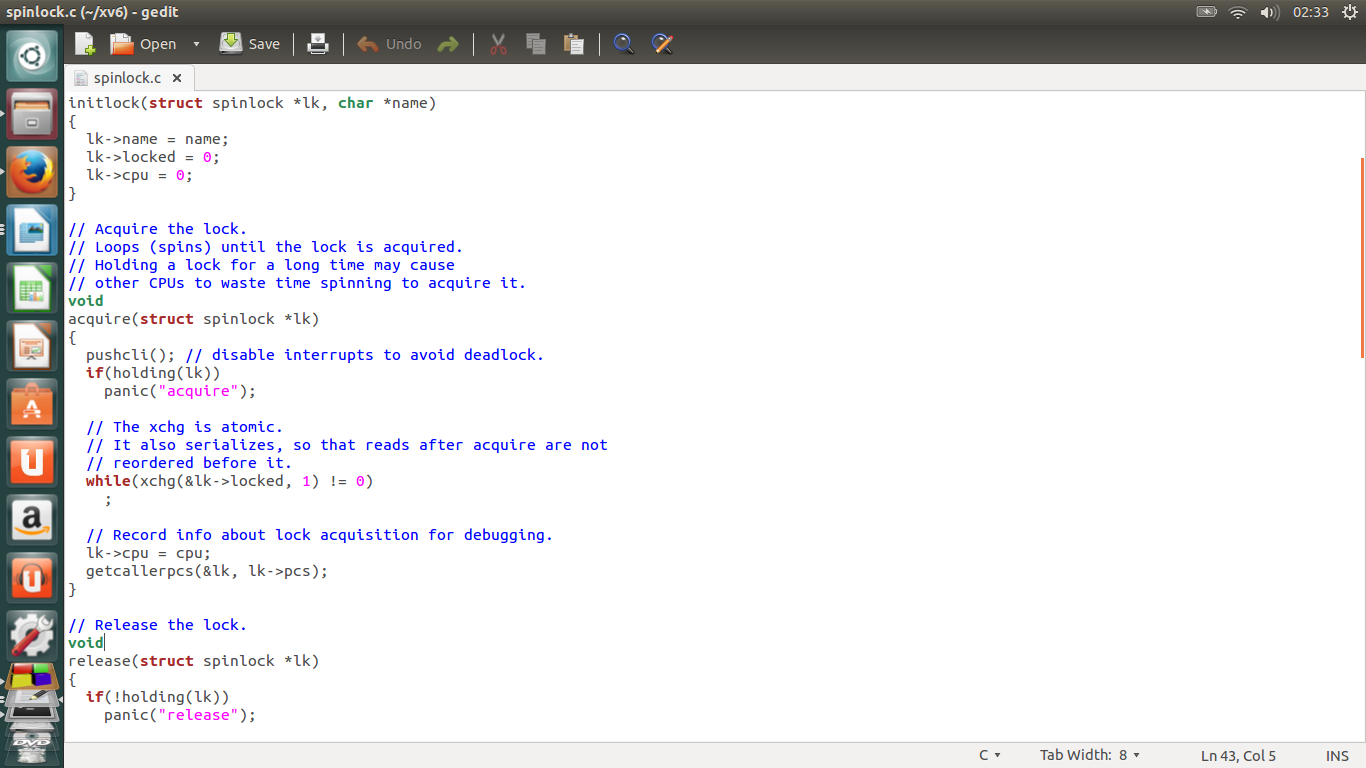
1. ¿Qué herramientas incluye el sistema operativo, o sólo consiste en el kernel?

El Sistema Operativo contiene varios system calls para hacer operaciones basicas como pipe, fork, etc. Esto se puede observar en el archivo syscall.c y ademas al correr xv6 tambien se puede observar algunas funciones al escribir ls en la linea de comandos.

2. ¿Qué mecanismos para protección de secciones críticas implementa?

Utiliza un mecanismo de protección llamado spinlock. Que a diferencia de un lock convencional, el thread se queda en un estado de loop esperando que el lock se encuentre disponible. Una vez adquirido el lock se mantiene hasta que sea explicitamente liberado. Este mecanismo se puede encontrar implementado en el archivo spinlock.c.

3. ¿Con qué técnica (TSL, Peterson's, etc.) realiza dichas protecciones?  
En el archivo spinlock.c se encuentran las siguientes lineas de codigo:  
  
// The xchg is atomic.

// It also serializes, so that reads after acquire are not

// reordered before it.

while(xchg(&lk->locked, 1) != 0)

;

Que nos dice que xchg es un valor atomico. Lo cual es con lo que trabaja la tecnica TSL (Test and Set Lock.

4. ¿Qué información se almacena por proceso en la tabla de procesos?

La informacion almacenada en la tabla de procesos es la siguiente: process id, el estado del proceso, y los apuntadores para la maquina ejecutable.   
  
Existe una llamada con el nombre allocproc cuyo trabajo es alocar un espacio en la tabla de proceso y iniciar los estados de proceso necesarios para que ejecute su thread. Menciono esto pues xv6 tiene un archivo llamado proc.c donde se encuentra el codigo que declara la tabla de procesos.  
  
A continuacion un una funcion en el archivo proc.c donde se muestra lo que contiene la tabla de procesos:  
  
for(;;){

// Scan through table looking for zombie children.

havekids = 0;

for(p = ptable.proc; p < &ptable.proc[NPROC]; p++){

if(p->parent != proc)

continue;

havekids = 1;

if(p->state == ZOMBIE){

// Found one.

pid = p->pid;

kfree(p->kstack);

p->kstack = 0;

freevm(p->pgdir);

p->state = UNUSED;

p->pid = 0;

p->parent = 0;

p->name[0] = 0;

p->killed = 0;

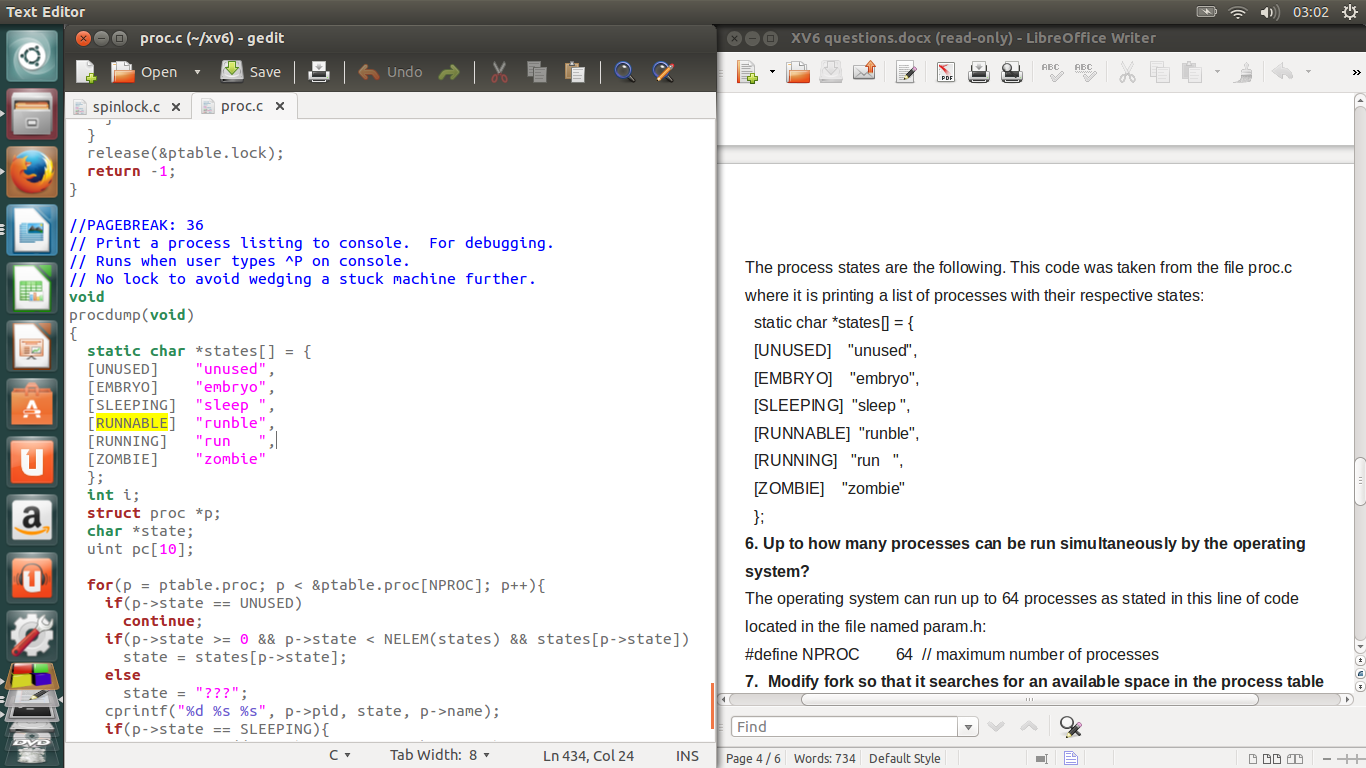
release(&ptable.lock);

return pid;

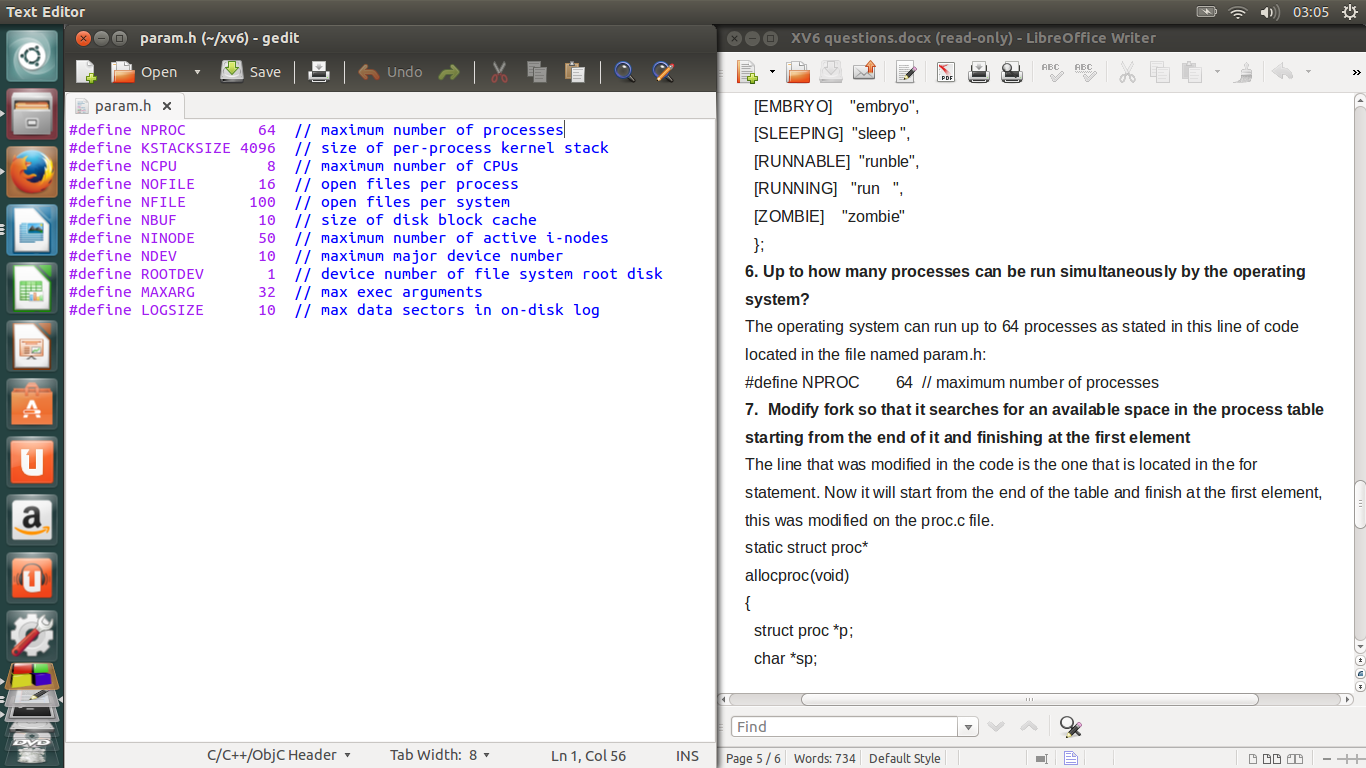
}

5. ¿Cuáles son los estados de los procesos en este sistema operativo?

Unused, embryo, sleep, runnable, running, zombie,, todos estos estados se pueden observar en el archivo proc.c



6. ¿Hasta cuántos procesos puede ejecutar el sistema operativo en un momento dado?  
64. Esta explícitamente declarado en el archivo param.h como se muestra a continuacion.



7. Modificar fork para que en vez de que busque un espacio disponible en la tabla de procesos de principio a fin lo haga de fin al principio.

Archivo Modificado: Proc.c, una linea cambiada. Especificamente fue la siguiente linea fue la que se terminó escribiendo: for(p = &ptable.proc[NPROC-1]; p >= ptable.proc; p--)

