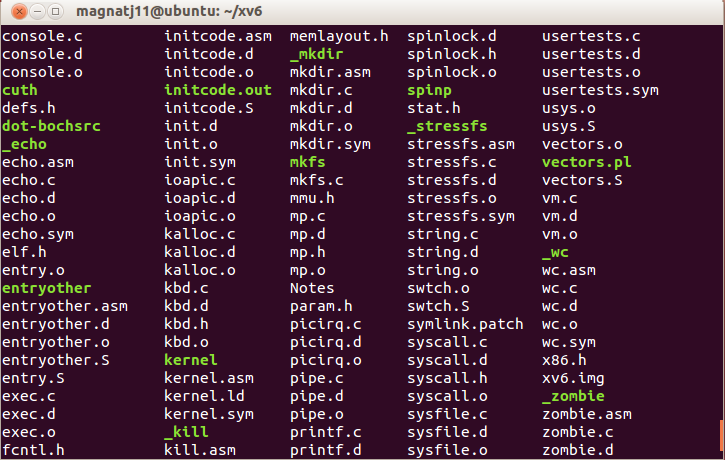
**XV6 QUESTIONS**

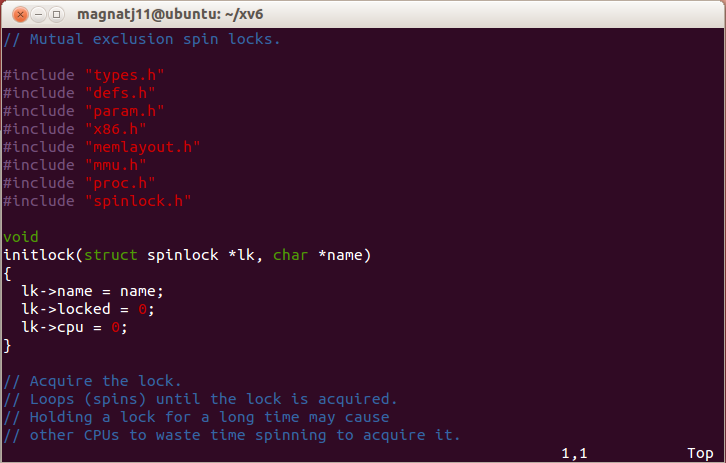
1. What tools does the OS include? Or is it only the kernel?

Some tools that the xv6 has are system calls such as kill, echo, printf, mkdir, etc… It is not only the kernel.



2. What critical section protection mechanisms implement?

Mutual exclusion spin locks, the next image shows it:



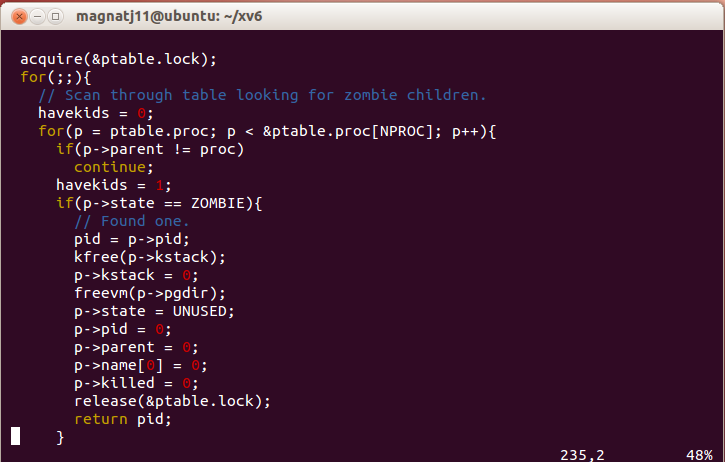
3. With which techniques (TSL, Peterson's, etc) does the operating system do this protection?

Apparently xv6 uses xchg to do the protection as the next image says:



4. What information is stored in the process table?

The next image shows us what information is stored in our process table which is the state, the id, the stack, the parent of that process, the name of the process and if that process has been killed.



5. Which are the process states in this OS?

The process states in this OS are:

* Unused
* Embryo
* Sleeping
* Runnable
* Running
* Zombie



6. Up to how many processes can be run simultaneously by the operating system?

As it is a mutual exclusion algorithm it may run simultaneously 2 processes!



7. Modify fork so that it searches for an available space in the process table starting from the end of it and finishing at the first element.

int

fork(void)

{

int i, pid;

struct proc \*np;

// Allocate process.

if((np = allocproc()) == 0)

return -1;

// Copy process state from p.

if((np->pgdir = copyuvm(proc->pgdir, proc->sz)) == 0){

kfree(np->kstack);

np->kstack = 0;

np->state = UNUSED;

return -1;

}

np->sz = proc->sz;

np->parent = proc;

\*np->tf = \*proc->tf;

// Clear %eax so that fork returns 0 in the child.

np->tf->eax = 0;

for(i = NOFILE; i > 0; i--)

if(proc->ofile[i])

np->ofile[i] = filedup(proc->ofile[i]);

np->cwd = idup(proc->cwd);

pid = np->pid;

np->state = RUNNABLE;

safestrcpy(np->name, proc->name, sizeof(proc->name));

return pid;

}