LUA shell script status report

Implementation wise we are basically done. You can output math results now, which means we have tested basic math handling, strings and concatenation, and printing to the screen. The kernel modification is done and working great. We are able to open, close, and read files for the interpreter. We have not tested opening files in the script yet, and the proper fprintf functionality is not done yet either. This means no file writing until it can write to any file descriptor and not just forwarding to printf. In terms of math, LUA 1.1 only uses doubles. Our code just treats the output as an int. So internally it's all done with doubles and the result will cut off the fractional part.

Below is the finished code for the kernel modification. I left out all the original code, since we submit the full version later. This is only our modification and nothing else.

```
char lua_path[] = "lua";
char lua_header[] = "--LUA 1.1";
char lua_test[sizeof(lua_header)];
char* script_path = (char*)0; //set to NULL
int is_lua_script = 0;
//lua header check
if (readi(ip, lua_test, 0, sizeof(lua_header - 1)) == sizeof(lua_header - 1)) //skip reading
a null char ;) this tests to see if the file has at least this many bytes
{
    if (strncmp(lua_header, lua_test, sizeof(lua_header - 1)) == 0) //actually see if it
matches signiture exactly
    {
          cprintf("lua script detected\n");
          is_lua_script = 1;
          script_path = path; //save the old path which is the script
          path = lua_path; //swap lua programs path where the script path was
          cprintf("script path: %s \n", script_path);
          iunlockput(ip); //unlock script file
          end_op(); //end op on script file

          begin_op(); //repeat steps above but now we are using lua's executable instead :D
          ip = namei(path);
          ilock(ip); //don't need to assign pgdir to 0 because it was done before here
    }
}
if (is_lua_script == 0)  {
//original argument pushing code
}
else  {
      //push lua path name and then the script name
      argc = 0; //push lua path
      sp = (sp - (strlen(lua_path) + 1)) & ~3;
      if (copyout(pgdir, sp, lua_path, strlen(lua_path) + 1) < 0)
          goto bad;
      ustack[3 + argc] = sp;
      //cprintf("argv[argc] %s \n", argv[argc]);

      argc++; //push script name
      sp = (sp - (strlen(script_path) + 1)) & ~3;
      if (copyout(pgdir, sp, script_path, strlen(script_path) + 1) < 0)
          goto bad;
      ustack[3 + argc] = sp;
      //cprintf("argv[argc] %s \n", argv[argc]);
      argc++;
  }
```