Homework: Naming

This assignment requires the files xv6.pdf and xv6_rev0.zip. You may download them from the Assignments page.

Read: namei in fs.c, fd.c, sys_open, sys_close, sys_dup, sys_mknod, sys_unlink, sys_link, sys_mkdir, and sys_chdir.

This homework should be turned in at the beginning of lecture.

Symbolic Links

As you read namei and explore its varied uses throughout xv6, think about what steps would be required to add symbolic links to xv6. A symbolic link is simply a file with a special type (e.g., T_SYMLINK instead of T_FILE or T_DIR) whose contents contain the path being linked to.

Turn in a short writeup of how you would change xv6 to support symlinks. List the functions that would have to be added or changed, with short descriptions of the new functionality or changes.

This completes the homework.

The following is *not required*. If you want to try implementing symbolic links in xv6, here are the files that the course staff had to change to implement them:

```
Makefile: 4 lines added, 2 modified
defs.h: 1 line added
fs.c: 45 lines added, 4 modified
fs.h: 1 line added
syscall.c: 4 lines added
syscall.h: 1 line added
sysfile.c: 9 lines added
user.h: 1 line added
usys.S: 1 line added
```

Also, here is an *ln* program:

```
#include "types.h"
#include "user.h"

int
main(int argc, char *argv[])
{
  int (*ln)(char*, char*);

  ln = link;
  if(argc > 1 && strcmp(argv[1], "-s") == 0){
    ln = symlink;
    argc--;
    argv++;
  }

  if(argc != 3){
    printf(2, "usage: ln [-s] old new (%d)\n", argc);
```

```
    exit();
  }
  if(ln(argv[1], argv[2]) < 0){
    printf(2, "%s failed\n", ln == symlink ? "symlink" : "link");
    exit();
  }
  exit();
}
```