# ECS 150 Programming Project #3

Due date: 11:59 PM, Tuesday, June 6th, 2007

This project will test your understanding of general I/O and file system system calls. No kernel programming is required for this assignment, but you will be expected to read the system call manpages to learn the details for the system calls you use. Do not use functions from `stdio.h` (fprintf, fgets, fopen, etc) for any part of this assignment.

### Part 1: File split program

Your first task is to write a C program (called "splitter") which splits a single file into several smaller files. The usage is:

```
./splitter filename N
```

where `N` is a number. Your program should split the file specified by `filename` into several files name `filename.0`, `filename.1`, etc. Each new file should be exactly `N` bytes long, except for the last one which may be less.

Check all of your system call return values for errors. If an error is detected; print out an error message with `perror()`.

### Part 2: File permission tester

Your next task is to write a C program (called "accesstest") which checks to see if a specified user has a specific type of access to a file. The usage is:

```
./accesstest user filename access
```

- `user` is the unix username of the user to test
- `filename` is the file to test permissions on
- `access` is "r," "w," or "x", signifying read, write, and execute respectively

If the user specified has the desired permission level, print "OK." If not, print "Permission denied." If the specified user does not exist, print "No such user." If the specified file does not exist, print "No such file." You may ignore the special case where `user` is "root" (since the unix root user can override all file permissions).

You will need to use the `stat` and `getpwnam()` system/library calls to implement this program. Make sure you read and understand their manpages.

### Part 3: Directory Lister

Your final task is to write a reimplementation of the unix "ls" command called "myls." Your program will optionally take a single command line parameter (a directory name) and will display the contents of that directory in standard "ls -l" format. If no directory is specified on the command line, use the current directory. If the name

specified on the command line is invalid or not a directory, return an appropriate error message.

Example:

```
./myls /home/roper/foo

dwx------- 2 roper users 4096 May 25 12:00 SomeDir
-rw------- 1 roper users 312 Apr 3 3:00 gradebook
-rwxr-xr-x 1 roper users 1083 Jan 4 2:51 myprog
```

The output should show the following:

- permission bits + directory status as first character; for the Unix gurus in the class, you don't have to worry about setuid/setgid/sticky bits unless you want to
- number of hard links to this file
- username of file owner
- group ownership of file
- size (in bytes)
- last modification time stamp
- filename

This program will require the use of several system and library calls: opendir(), readdir(), stat(), getpwuid(), getgrgid(), and ctime() calls to implement this program. Make sure you read and understand their manpages.

## Submitting Your Assignment

Implement each of your programs in a C file with the executable's name + ".c" (e.g., splitter should be implemented in splitter.c). Provide a single Makefile which will compile each of your programs and a README that specifies which of your programs do/do not work. Email your files as attachments to `roper@cs.ucdavis.edu`