

EECS 338 Assignment #2

Spring 2002

G. Ozsoyoglu

Prepared by Steven Huwig

Due February 12, 2002

This assignment involves writing two separate programs and using lock files and signals as a form of interprocess communication. You should look closely at Program 4.2 (p. 97) and Program 4.4 (p. 109) in your Unix textbook, as almost everything covered in this assignment is demonstrated in these examples.

Program 1 should be invoked as a background process with a single argument; e.g.
`mozart % ./program1 filename &`. It should perform the following actions:

- Set up a signal handler to catch SIGUSR1.
- Create a file named *filename* and write the current date and time, hostname, and process ID to this file.
- Lock this file from reads using `fcntl()` and loop infinitely.
- When SIGUSR1 is received, the process should print this fact, and write the current date, time, hostname, and process ID to the file. It should then unlock the file and exit.

Program 2 should be invoked with two arguments and run in the foreground; e.g.
`mozart % ./program2 filename tries`. It should perform the following actions:

- Attempt to get a read lock on the file (again using `fcntl()`) in use by Program 1. It should attempt this for *tries* number of times. When it fails (as it must), it should print the time and both its own process ID and the process ID of the program locking *filename*. It should sleep() one second between each try.
- After the number of times exceeds *tries*, it should send SIGUSR1 to Program 1, and try to get a lock on the file again. Failures should be noted as above.
- When it succeeds, it should remove the lock and use a variant of `exec()` to send the file to the `cat` program for output to the screen.

You will need to use several library functions and system calls to complete this assignment. Among others, you will probably need to use `fprintf()`, `open()`, `fdopen()`, `signal()`, `fcntl()`, `kill()`, `close()`, `fclose()`, and the `flock` structure. Your information should be clearly labeled in both the file and the terminal output. Use the `script` facility as in Assignment 1, and demonstrate your code for several variations of *filename* and *tries*. This is not a complicated assignment, but it does require you to understand how Unix and the C Standard Library interact.

As before, you will submit a hardcopy printout of both your source code and your program output, and make your binary available on your web page on the due date. On the sixth day after the assignment is due, make your source available on the web page.