

**Operating Systems – Fall 2003**  
**Laboratory Assignment Number 5**  
**Assigned September 17, 2003**  
**Due September 24, 2003**  
**100 Points**

**Name:** \_\_\_\_\_

**Instructions and Lab Objective:**

The goal of this laboratory assignment is to implement concurrent processing in the Linux operating system using fork and shared memory. You may discuss with each other during the lab. However, the submitted assignment must be completely your own work. Submit all your source code files and associated Makefile and data files all combined into a tar.gz archive. Also include a README file giving instructions on how to compile and run your programs.

***Input***

The input is a text file containing one string on each line. Please get the name of the file as input. There are some strings which are palindromes. Palindrome is a word or phrase that reads the same backward as forward.

e.g. MADAM,  
MALAYALAM

Don't nod,

Live not on evil,

O had I nine more hero-men in Idaho!,

Pull up if I pull up.

(Just ignore the special characters , just consider alphanumeric strings)

***Procedure***

The main program will read the strings from the input file into shared memory. The child process will be executed by a fork() and exec() procedure where the command-line argument given to the child should be the index of the string to be processed.

The children will determine whether the string is a palindrome and write the string into an output text file. You can get the name of the output file from the user. You will have to use the code for multiple process ipc problem to protect the critical resources - shared memory.

Make sure you never have more than 4 processes in the system at any time.

***Output***

The preferred output format is:

PID String Yes or No

"Yes" means that the string is a palindrome. "No" means its not a palindrome. If the child starts to execute code to enter the critical section, it must print a message to that effect on

stderr. It will be a good idea to include the time when that happens. Also, indicate the time on stderr when the process actually enters and exits the critical section. Within the critical section, wait for 2 seconds before you write into the file, and then, wait for another 2 seconds before leaving the critical section.

You will be required to create separate parallel processes from your main process. That is, the main process will just spawn the child processes and wait for them to finish. The main process also sets a timer at the start of computation to 100 seconds. If computation has not finished by this time, the main process kills all the spawned processes and then exits. Make sure that you print appropriate message(s).

### *Useful Link for This Project*

[UNIX/LINUX System Calls and Subroutines using C \(http://www.cs.cf.ac.uk/Dave/C/\)](http://www.cs.cf.ac.uk/Dave/C/)

### **Note**

This laboratory assignment has been adapted from <http://www.cs.nmt.edu/~cs325/>.