

CS341 - Operating Systems

Lab Assignment #1

Spring 2004

The purpose of this assignment is to use UNIX system calls to do some very basic process operations, and to retrieve arguments from the command line using C/C++.

For this assignment, you are to write a program that accepts two command line arguments – a “sleep” time and an iteration count. This program should contain a loop which sleeps for the specified amount of time, then wakes up and prints its name, process ID (PID) and iteration count to `stderr`. The loop should execute the number of times specified in the iteration count. Then the program should print a message and terminate.

For example, if the name of your program is `testprog`, the following command should execute the program for 10 iterations, sleeping 4 seconds between iterations:

```
testprog 4 10
```

The output should look something like:

```
Executing testprog, process id 28435, iteration number 1
Executing testprog, process id 28435, iteration number 2
Executing testprog, process id 28435, iteration number 3
Executing testprog, process id 28435, iteration number 4
Executing testprog, process id 28435, iteration number 5
Executing testprog, process id 28435, iteration number 6
Executing testprog, process id 28435, iteration number 7
Executing testprog, process id 28435, iteration number 8
Executing testprog, process id 28435, iteration number 9
Executing testprog, process id 28435, iteration number 10
testprog is now exiting.
```

You can run this program on any UNIX/LINUX system that you can find; however, I would suggest that you use the machine named `opsys` for your final test run, since it is the machine that will be used to test the operation of your assignment.

In addition to running the program, you are to determine as many statistics about the process as you can find. For example, what is the size (in bytes) of the program? How much is code; how much is data? How much memory does it occupy while executing? There are numerous UNIX utility programs that can collect and report statistics of the program.

Another thing to try is to observe the execution of multiple instances of the program. The `top` program will display a list of the processes that are currently active in the system. Start up several instances of your program 1, and watch the execution using `top`.