

**EECS 338 Assignment #1: Basic Unix Concurrent Process System Calls**  
*Due: Thursday, February 1<sup>st</sup>, 2007*

Spring 2007, G. Ozsoyoglu

In this assignment you will use UNIX process control functions by creating concurrently executing processes, and have them execute simple Unix system calls. Your main program, called the (parent) process P, will print its pid (*getpid()* call), hostname (*gethostname()* call), username (*getuid()* call), the time of the day (*ctime()* or *time()* calls), its working directory (*getcwd()* call) and some text indicating that it is the parent process. Then P will create an environment variable HIPPO, and initialize it to 11 (*putenv()* call). P will then fork out two child processes, processes C1 and C2. Each of the two child processes will then print its pid and its parent pid (*getppid()* call) together with a meaningful text explanation. After this, the three processes will take turns to count backwards and decrement HIPPO, and to print as follows:

```
P: 10 little hippopotamus (HIPPO value is now xx)
C1: 9 little hippopotamus (HIPPO value is now xx)
C2: 8 little hippopotamus (HIPPO value is now xx)
P: 7 little hippopotamus (...)
C1: 6 little hippopotamus
C2: 5 little hippopotamus
P: 4 little hippopotamus
C1: 3 little hippopotamus
C2: 2 little hippopotamus
P: 1 little hippopotamus
```

The three processes achieve this decrementing of HIPPO and printing its value by repetitively sleeping (*sleep()* call), awakening, then printing and flushing the printed line (*fflush()* call). Then, the parent process P waits until both C1 and C2 terminate, increments and prints the final value of HIPPO, and exits. The process C1 changes its directory (*chdir()* call), changes its address space (*execXX()* call) to execute the “ls” command, and exits. The process C2 prints the current working directory (*getcwd()* call), gets the variable HIPPO (*getenv()* call), increments it, prints it, and exits.

Run your program in the *script* environment by using the Unix command *script* to record the entire session. Your output will be captured to a file called *typescript* (open it with *pico* and save it to remove the line-feed characters). Print meaningful output, not just the values.

On the due date (Feb. 1<sup>st</sup>), (a) submit a hard copy printout of your code and output in the class, (b) make your binary and output (but NOT your source code) available on your web page on vorlon (i.e., <http://vorlon.case.edu/xyz133> assuming your case e-mail id is xyz133), and (c) email your binary, output, and source code to the grader, Yixuan Chen ([yixuan.chen@case.edu](mailto:yixuan.chen@case.edu)) with "338 HW1" in the subject.

On the third CWRU day after the assignment is due (February 6th), make your source code also available on your 338 course web page on vorlon.

Remember to error-check all system calls: check return values for success, and use *perror* when possible on failure.