CS 333: Operating Systems Lab
Autumn 2017

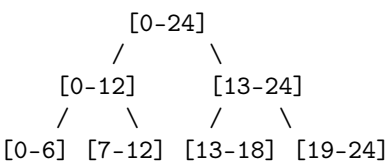
Lab Quiz # 1, 20 marks

1. **divide-and-conquer with fork**
   Write a C program `parmax.c` that creates a tree of processes in order to recursively compute the maximum in an array of integers. The process at the root of the tree reads a file which specifies the value n and the list of n numbers to be stored in an array (process can read via STDIN and use the re-direction operator. check readme for syntax). The initially unsorted array is printed by the root process.

   Any process in the tree handles a chunk of the array A. The chunk is delimited by two indices L and R, the left and right indices. For the root process, $L = 0$ and $R = n - 1$. Any process P in the tree (including the root) first counts the number of integers in the chunk. If count of elements is less than 10, the process P itself computes the maximum element in its chunk, prints it, and exits. If the chunk size of array assigned to process P is more than 10, then P creates two child processes PL and PR which handle the chunks [L, M] and [M + 1, R] in A respectively, where M = (L + R) / 2. Assume the start index of the array is 0. P waits until the two child processes PL and PR exit. It then computes the maximum of the two maximum values computed by PL and PR, prints this maximum, and exits. Every non-root process returns to its parent (via the exit status) the maximum value for its chunk. During the printing of the maximum value computed by a process P, the PID and the parent PID of P are also printed.

   Note:

   - Input n is followed by n numbers in range 0 to 255.
     Output of each process should print the following:
     `<pid>, <ppid>, left-index, right-index, max-value-in-range`
     The maximum value is printed by the first process at the end.
   - Left and right indices use 0-based indexing and split is done as follows:
     left child: [left, (int)(left+right)/2]
     right child: [(int)(left+right)/2+1 , right]

   - System calls and C library calls of interest: `fork`, `getpid`, `getppid`, `wait`, `exit`
   - A sample input and output files are available as part of the labquiz files.

2. **plumbing for mosh**
   Any simple Linux command followed by the pipe operator `|` and a second simple command (e.g., `cat abc.txt | grep hi`) should cause the output of the first command to be redirected as the input of the second. The output of the second command should be printed. You may assume that the two commands are simple Linux commands without any complications like chained pipes and re-directions. The command should execute in the foreground. An incorrect command format must print an error and prompt for the next command. On typing `exit` at the mosh prompt, your shell program should exit.
   This question does **not** ask you to implement the full shell functionality, you are expected to implement the pipe functionality connecting two commands only. The tokenizer to parse the command line is provided.

   System calls and C library calls of interest: `fork`, `exec`, `pipe`, `dup`

3. **Deadline: 11th August 5.05 p.m.**
   Expected time for completion of lab: 3 hours.
   Submission via moodle.