INDIAN INSTITUTE OF TECHNOLOGY PALAKKAD

COMPUTER SCIENCE AND ENGINEERING
Indian Institute of Technology Palakkad
CS3110: Operating Systems Lab
Lab 6

09-Sep-2024

1. Write a program to move integers between two processes A and B over a pair of pipes, one for each direction. The specifics are as follows:

   - A sends an integer to B.
   - Upon reception, B should print `<pid> received` x, where `<pid>` is its process ID.
   - B should send the integer $x + 10$ to A and exit.
   - Upon reception, A should print `<pid> received` y, where `<pid>` is its process ID, and exit.

   Hint: Use the system calls `pipe`, `fork`, `read`, `write`, and `getpid`.

2. Write a concurrent prime sieve program. The first process feeds the numbers 1 through n to the 2^{nd} process. The i^{th} process reads from $(i-1)^{\text{th}}$ process over a pipe. Let x_i be the first value received by the i^{th} process. It will print this value and write to the $(i+1)^{\text{th}}$ process numbers greater than x_i that are not a multiple of i.

3. Implement a user-level `sleep` program for 64-bit RISC-V xv6, along the lines of the UNIX sleep command. Your `sleep` should pause for a user-specified number of ticks. For hints visit `https://pdos.csail.mit.edu/6.828/2024/labs/util.html`.

4. Write an `uptime` program that prints the uptime in terms of ticks using the uptime system call.

5. Write a simple version of the UNIX `xargs` program for xv6: its arguments describe a command to run, it reads lines from the standard input, and it runs the command for each line, appending the line to the command's arguments. For hints visit `https://pdos.csail.mit.edu/6.828/2024/labs/util.html`.

6. Implement a user-level program `halt` for xv6. When invoked, this program should immediately halt the running kernel.

7. Implement a user-level program `syscount` for xv6. When invoked, this program should print the number of system calls that has occurred.

End of document