Winston Shine

Operating Systems

Lab 1

4/2/2024

I worked mostly solo on this lab, but I did discuss with Dominic about what was going on in threads.c and why the input was correct with a smaller integer input

```
    1  ls
    2  git clone https://github.com/remzi-arpacidusseau/ostep-code.git
    3  ll
    4  git status
    5  cd ostep-code/
    6  git status
    7  git config -l
    8  ls
    9  mv intro/ ..
   10  cd ..
   11  ll
   12  cd intro/
   13  ll
   14  cd ..
   15  mkdir labs
   16  touch os-lab1.md
   17  mv os-lab1.md labs/
   18  cd intro/
   19  make
   20  ll
   21  ./cpu A
   22  ./mem 1
   23  ./threads 10000
   24  ./threads 100000
   25  ./threads 100
   26  ./io
   27  ll
   28  vim
   29  vim io.c
   30  ll /tmp/
   31  vim /tmp/file
   32  ./mem 1
   33  history > lab2-history.txt
```

cpu.c

- takes exactly one command line argument as a string
- loops forever doing:
  - prints given string to stdout
  - calls Spin(), which seems to look repeatedly until 1 second has passed

mem.c

- takes exactly one cmd arg as int
- not sure what atoi does
- endlessly loops

  - waits 1 second
  - increments given int
  - prints new value to stdout along with processid

threads.c

- takes an int as cmd arg (ill call this n)
- creates two threads passing them the following function:
  - loops n times incrementing a counter (global variable shared by both threads)
- prints final counter to stdout

the expected output should be `2*n` in reality with higher input the number is smaller because the program is not using mutex locks when changing the value of counter.

I only observed this with input ~100,000 This output appears correct with small inputs only because thread 1 finishes it's task before thread 2 can even start

io

- opens a file '/tmp/file'
- writes 'hello world\n' to a buffer
- writes the contents of that buffer to the opened file
- not sure what fsync does, but the last statement closes the file