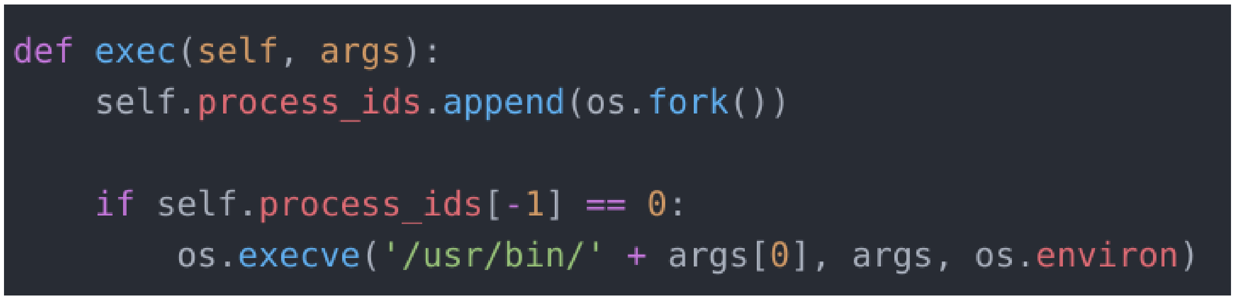
Alex Avila

Pre-Shell Exercise

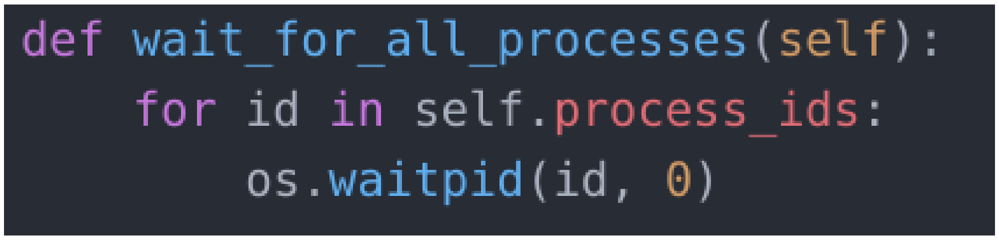
For this assignment we needed to create 4 different processes for 4 different commands that we could run using Linux. These commands needed to be executed in python using the os module and to create a child process for each command using os.fork(). Then to execute the command we could use the execv() command however because we needed to also run a python program then it will be more flexible to use execve() which takes the environment variables of the operating system to run the python program, this method needs the path to the bin folder of the executable commands which on Linux they are store in the /usr/bin/ folder then it needs a list of the arguments for the command.

I tried doing this with less than 20 lines of code, but I was not able to do. Before I needed 4 different if statements that check if the for the child process when the fork method was invoked so I made a method for all this statements that only takes the arguments since it is the only think that is different about the method.



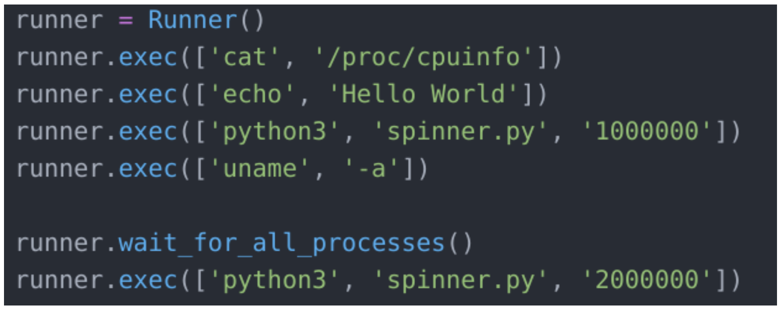
```
def exec(self, args):
    self.process_ids.append(os.fork())

    if self.process_ids[-1] == 0:
        os.execve('/usr/bin/' + args[0], args, os.environ)
```

Then all the ids of the processes are saved into a list to wait for them after the first 4 processes are done, this is accomplished by using a loop that stores all the ids of the previous child processes.

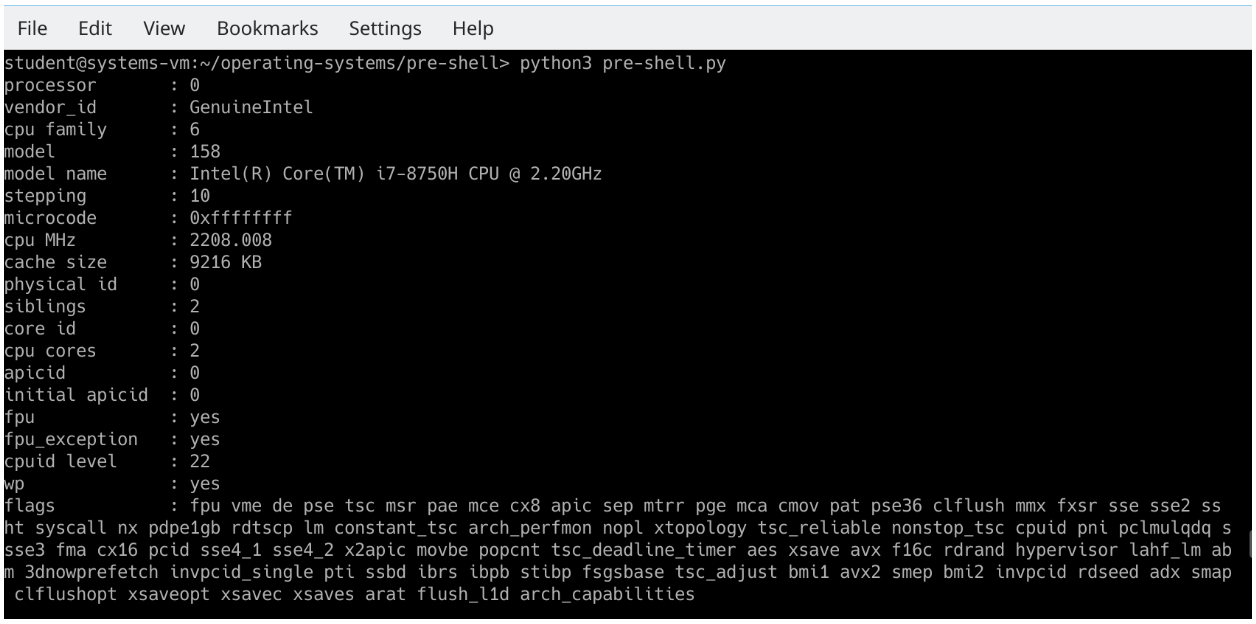


```
def wait_for_all_processes(self):
    for id in self.process_ids:
        os.waitpid(id, 0)
```

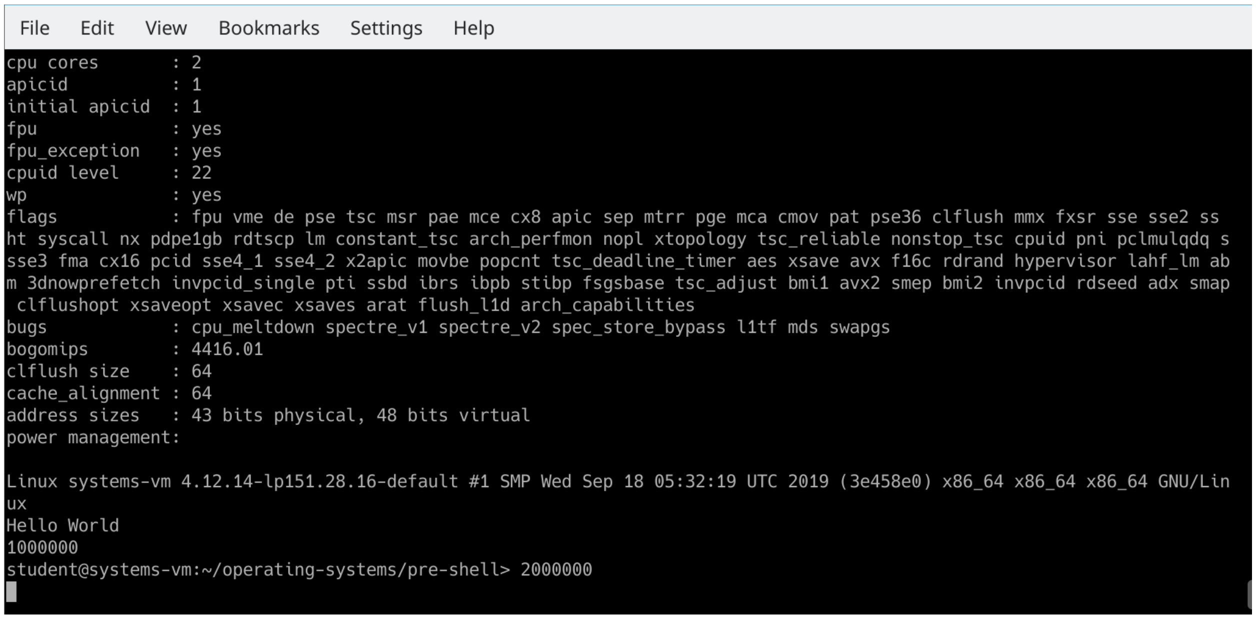
The order in which the processes are execute does not matter in some cases hello world is echo before uname -a or vice versa the only process that is always the last is the spinner in its 1,000,000 loop cycle since it appears to be the most consuming process. After this one is done we see the directory console telling us that the program finished but the process that was created with the 2,000,000 loop will be still running and then print 2,000,000 after a fraction of a second.



```
runner = Runner()
runner.exec(['cat', '/proc/cpuinfo'])
runner.exec(['echo', 'Hello World'])
runner.exec(['python3', 'spinner.py', '1000000'])
runner.exec(['uname', '-a'])

runner.wait_for_all_processes()
runner.exec(['python3', 'spinner.py', '2000000'])
```



```
student@systems-vm:~/operating-systems/pre-shell> python3 pre-shell.py
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 158
model name      : Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz
stepping        : 10
microcode       : 0xffffffff
cpu MHz         : 2208.008
cache size      : 9216 KB
physical id     : 0
siblings        : 2
core id         : 0
cpu cores       : 2
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 22
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ss
ht syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon nopl xtopology tsc_reliable nonstop_tsc cpuid pni pclmulqdq s
sse3 fma cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand hypervisor lahf_lm ab
m 3dnowprefetch invpcid_single pti ssbd ibrs ibpb stibp fsgsbase tsc_adjust bmi1 avx2 smep bmi2 invpcid rdseed adx smap
 clflushopt xsaveopt xsavec xsaves arat flush_l1d arch_capabilities
```



```
cpu cores       : 2
apicid          : 1
initial apicid  : 1
fpu             : yes
fpu_exception   : yes
cpuid level     : 22
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ss
ht syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon nopl xtopology tsc_reliable nonstop_tsc cpuid pni pclmulqdq s
sse3 fma cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand hypervisor lahf_lm ab
m 3dnowprefetch invpcid_single pti ssbd ibrs ibpb stibp fsgsbase tsc_adjust bmi1 avx2 smep bmi2 invpcid rdseed adx smap
 clflushopt xsaveopt xsavec xsaves arat flush_l1d arch_capabilities
bugs            : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass l1tf mds swapgs
bogomips        : 4416.01
clflush size    : 64
cache_alignment : 64
address sizes   : 43 bits physical, 48 bits virtual
power management:

Linux systems-vm 4.12.14-lp151.28.16-default #1 SMP Wed Sep 18 05:32:19 UTC 2019 (3e458e0) x86_64 x86_64 x86_64 GNU/Lin
ux
Hello World
1000000
student@systems-vm:~/operating-systems/pre-shell> 2000000
```