```
””” centralserver.py

A time−shared computer consists of a single
central processing unit (CPU) and a number of
terminals. The operator of each terminal ‘thinks’
for a time (exponential, mean 100.0 sec) and then
submits a task to the computer with a service time
(exponential, mean 1.0 sec). The operator then
remains idle until the task completes service and
returns to him or her. The arriving tasks form a
single FCFS queue in front of the CPU.

Upon leaving the CPU a task is either finished
(probability 0.20) and returns to its operator
to begin another ‘think’ time, or requires data
from a disk drive (probability 0.8). If a task
requires access to the disk, it joins a FCFS queue
before service (service time at the disk,
exponential, mean 1.39 sec). When finished with
the disk, a task returns to the CPU queue again
for another compute time (exp, mean 1.$ sec).

the objective is to measure the throughput of
the CPU (tasks per second)
”””
from SimPy.Simulation import *
## from SimPy.SimulationTrace import *
import random as ran

## Model components ------------------------

class Task(Process):
    ””” A computer  task  requires at least
    one use of the CPU and possibly accesses to a
    disk drive.”””
    completed = 0
    rate = 0.0
    def execute(self,maxCompletions):
        while Task.completed < maxCompletions:
            self.debug(” starts thinking”)
            thinktime = ran.expovariate(1.0/MeanThinkTime)
            yield hold,self,thinktime
            self.debug(” request cpu”)
            yield request,self,cpu
            self.debug(” got cpu”)
            CPUtime=ran.expovariate(1.0/MeanCPUTime)
            yield hold,self,CPUtime
            yield release,self,cpu
            self.debug(” finish cpu”)
            while ran.random() < pDisk:
                self.debug(” request disk”)
                yield request,self,disk
                self.debug(” got disk”)
                disktime=ran.expovariate(1.0/MeanDiskTime)
                yield hold,self,disktime
                self.debug(” finish disk”)
                yield release,self,disk
                self.debug(” request cpu”)
                yield request,self,cpu
                self.debug(" got cpu")
                CPUtime=ran.expovariate(1.0/MeanCPUTime)
                yield hold,self,CPUtime
                yield release,self,cpu
            Task.completed += 1
        self.debug(" completed %d tasks"%(Task.completed,))
        Task.rate = Task.completed/float(now())

    def debug(self,message):
        FMT="%9.3f %s %s"
        if DEBUG:
            print FMT%(now(),self.name,message)


## Model ------------------------------
def main():
    initialize()
    for i in range(Nterminals):
        t = Task(name="task"+`i`)
        activate(t,t.execute(MaxCompletions))
    simulate(until = MaxrunTime)
    return (now(),Task.rate)

## Experiment data -------------------------

cpu  = Resource(name='cpu')
disk = Resource(name='disk')
Nterminals = 3        ## Number of terminals = Tasks
pDisk    = 0.8        ## prob. of going to disk
MeanThinkTime = 10.0  ## seconds
MeanCPUTime = 1.0     ## seconds
MeanDiskTime = 1.39   ## seconds

ran.seed(111113333)
MaxrunTime = 20000.0
MaxCompletions = 100
DEBUG = False


## Experiment

result=main()

## Analysis/output -------------------------

print 'centralserver'
```

```
centralserver
```

```
print '%7.4f: CPU rate = %7.4f tasks per second'%result
```

```
842.7865: CPU rate =  0.1210 tasks per second
```