CS2506 Operating Systems II

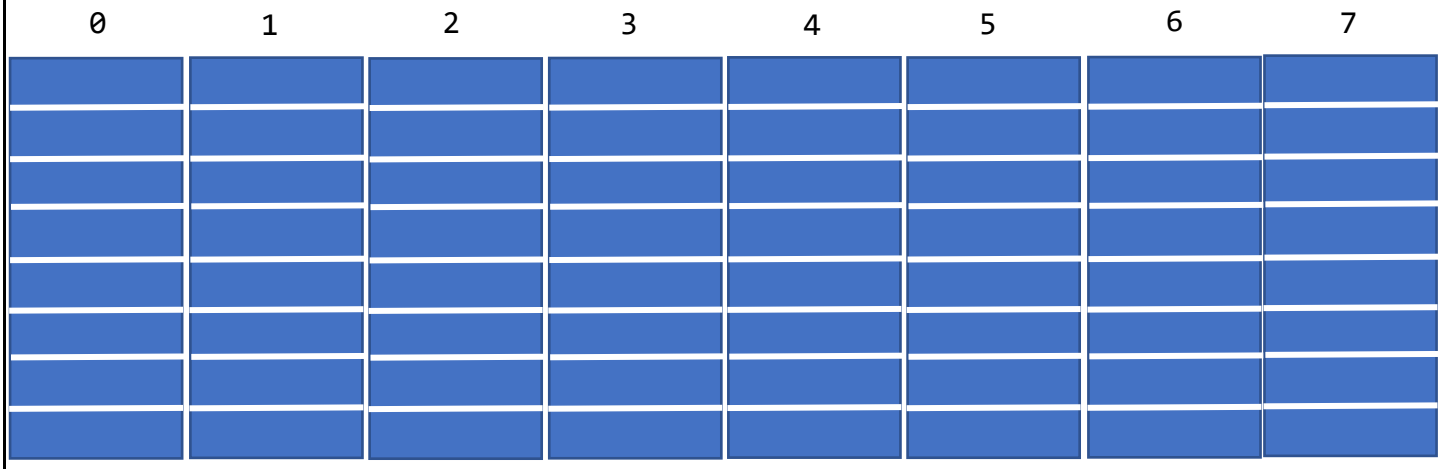
Processes & Scheduling II

LAB2

Colin Kelleher – 117303363

1. Analysis

8 Multilevel Feedback Queues



PL: 4	PL: 6	PL: 8	PL: 10	PL: 12	PL: 14	PL: 16	PL: 18
$Q:2^4$ = 16	$Q:2^6$ = 64	$Q:2^8$ = 256	$Q:2^{10}$ = 1024	$Q:2^{12}$ = 4096	$Q:2^{14}$ = 16384	$Q:2^{16}$ = 65536	$Q:2^{18}$ = 262144

PL: Priority Level
Q = Time quantum

Explain what happens when all user processes terminate and there is no other process ready to execute
When all the user processes terminate and there is no other process ready to execute, the CPU goes into IDLE

2. Design

Processes_and_Scheduling2.py – PSEUDOCODE

```
Class Process:
     def __init__(self, quanta, IO, state, ID, priorityLevel)
          quanta – allocation of CPU
          state – state of process
          id – process ID
          IO – if I/O is required or not
          PriorityLevel = level of priority given to process

     def getQuanta
          return quanta
     def setQuanta
          set new quanta

     def getstate
```

```
            return state
      def setstate
            set new state

      def getID
            return ID
      def setID
            set new ID

      def getIOState
            return IOState
      def setIOstate
            set new IOstate

      def getpriority
            return priority
      def setpriority
            set new priority
      def Printprocess
            formatted string of process id, quanta, state,
priority, IO

Class CPU:
      def __init__ (self):
            currentprocess
            queue
            queues(8) =[ [], [] , [] , [] , [] , [] , [] , [] ]
            blockedqueue = []

      def getcurrentprocess
      def set currentporcess

      def getqueue
      def set queue

      def print blocked queue
            print id of process in blocked queue

      def printQueues
            for each queue in the list of queues
                  for each item in the queue
                        print ID

      def addProcess(process):
            **calculate which queue process depending on
priority
            base
            index = processes priority – base / 2
            append process to queues[index]
```

```
      def addReadyProcess(process)   :
            addProcess
            call process and add it appropriately


      def addBlockedProcess
            appended to blocked queue
            blockedqueue.append(process)


      def AddSuspendedProcess:
            update the state to be suspended

      def IORequ:
            process = blocked queue index 0…
            change process state from ‘blocked’ to ‘ready’
            change IO from True to False
            if the priority of the process isn’t 4
                  take two away from the priority
            call ‘addProcess’
                  run addProcess method which will add the
process appropriately

      def Check
            #check to see if all processes finished
            Finish = True
            For each queue in the list of queues
                  If the queue is not empty
                        FinishQ = False
            Return FinishQ

      def runProcess
            initiate index
            While index less than qty of queues
                  Queue = index (value of index) within queues
                  While queue is empty
                        If the index is one less than the qty
                              Return index to 0
                        Otherwise
                              Increment index by one

                  Set current process to top of active queue
                  Print – Process ID x is currently running with
priority x and quanta x

                  If the current process has a quanta of zero
                        Print ‘Process ID x completed
                        Remove process from the queue
```

```
                Else if IO is required
                        Print 'Process ID x sent to blocked queue
                        Decrement the quanta
                        Add to blocked queue
                        Remove from top of queue
                        Require input

                Else
                        Decrement the quanta of the current pro
                        Return the current process to the back of
the ready queue
                        Remove from top of queue

                        If the blocked queue is not empty
                                Update the state of the first
process in the blocked queue to be ready
                                Process at top of blocked queue is
sent to ready queue
                                Print 'Process ID x returned to
ready state
                                Remove from blocked queue

          Call checkPro function to see if all processes are
finished running
          If true, print message and return


     def Schedule
          call runProcess - used to run processes when


def Test()

     ***testing code goes here***
```

3. Programming

Processes_and_Scheduling.py

```
'''
Processes_and_Scheduling.py

Colin Kelleher - 117303363

CS2506 Operating Systems 2 – Lab2 - Processes and Scheduling - II
'''
from random import randint #Import module to generate random numbers
class Process: #defines a class, to allow the creation of a process
    def __init__(self, quanta, IO, state, id, priority):
        self._quanta = quanta #allocation of CPU
        self._IO = IO  # If I/O operation required or not
        self._state = state #state of process (ready or blocked)
        self._id = id #id of process
        self._priority = priority #level of priority given to process
    #getter and setter for quanta
    def getquanta(self):
        return self._quanta #return the value currently in quanta
    def setquanta(self, quantaN):
        self._quanta = quantaN #update quanta with a new value

    # getter and setter for ID
    def getIO(self):
        return self._IO #return the value currently in IO
    def setIO(self, ION):
        self._IO = ION #update IOwith a new value

    # getter and setter for state
    def getState(self):
        return self._state#return the value currently in state
    def setState(self, stateN):
        self._state = stateN #update statewith a new value

    # getter and setter for ID
    def getID(self):
        return self._id#return the value currently in ID
    def setID(self, idN):
        self._id = idN #update ID with a new value
```

```
    # getter and setter for priority
    def getPriority(self):
        return self._priority #return the value currently in
Priority
    def setPriority(self, priorityN):
        self._priority = priorityN #update priority with a new
value

    #return a formatted string ot the process - Process ID x
has a Quanta of x, is in the ready/blocked state and has a
priority of x
    def PrintProcess(self):
        print('Process ID:',self._id, 'has a Quanta of:',
self._quanta, 'is in the', self._state, 'state' ' and has a
priority of:', self._priority)


class CPU: #defining CPU class
    def __init__(self):
        self._curprocess = None #current process running
        self._queue = None #queue
        self._queues = [[], [], [], [], [], [], [], []]  #
multilevel feedback queues (8)
        self._blockedqueue = [] #blocked queue

    # getter and setter for ID
    def getCurProcess(self):
        return self._curprocess #return the value currently in
curprocess
    def setCurProcess(self, newCurProcess):
        self._curprocess = newCurProcess #update curprocess
with a new value

    # getter and setter for ID
    def getQueue(self):
        return self._queue #return the value currently in
queue
    def setQueue(self, newQ):
        self._queue = newQ #update queue with a new value

    def printBlockedQueue(self): #print the contents of the
blocked queue
        print('Blocked Queue:') #print title
        for item in self._blockedqueue:
            print('Process id:%i' %item._id) #print the
process id of processes in blocked queue
```

```
    def printQueues(self): #print the contents of the
multilevel queues
        print('Queues') #print the title
        for queue in self._queues: #for each queue in the
multilevel queues
            for item in queue: #for each item in each queue
                print('Process id:%i' %item._id) #print the
process id of the processes in the queue

    def addProcess(self, process): #adding a process
        base = 4 #using base 4 to calculate which queue is
added according to priority
        index = (process._priority - base) // 2 #priority of
process divided by two
        self._queues[index].append(process) #using index,
append the process to the index of the queues
        print('Process %i added!' % process._id) #print
message

    def addBlockedProcess(self, process): #add a blocked
process
        self._blockedqueue.append(process)
        print('Process %i added to Blocked Queue' %
process._id)

    def IORequired(self, process): #i/o REQUIRED
        process = self._blockedqueue[0] #Add process to index
0 of blocked queueue
        process._state = 'ready' #change state to ready
        process._IO = False #change IO to false
        if process._priority is not 4: #íf the priority is not
fpur
            process._priority -= 2 #take/update priority to be
two
        self.addProcess(process) #send process to addProcess
method

    def addReadyProcess(self, process): #method for adding a
ready rpocess
        self.IORequired(process) #call IO required method

    def checkPro(self): #check if there is more processes to
run or if processing is complete
        finishQ = True #boolean to keep track
        for queue in self._queues: #for each queue in the list
of queues
            if queue is not []: #if the queue is not empty
                finishQ = False #update status to false
        return finishQ #return status
```

```
    def runProcess(self): #used to run the process within CPU
        index = 0 #inititate index
        while index < 8: #while the index is less than 8
            self._queue = self._queues[index] #update queue
with index of queues
            while self._queue == []: # check if current active
queue has a process
                if index == 7: #if the index is 7
                    index = 0 #update index
                else: #otherwise
                    index += 1 #íncrement by one
                self._queue = self._queues[index] #queues =
queues index

            self._curprocess = self._queue[0] # set current
process to top of active queue
            print("Process ID:%i is currently running, with
priority: %i and  quanta: %i" %
                  (self._curprocess._id,
self._curprocess._priority, self._curprocess._quanta))

            if self._curprocess._quanta == 0: #if the current
process has a quanta of zero
                print("Process ID:%i completed" %
self._curprocess._id) #print message
                self._queue.pop(0)  #remove from queue

            elif self._curprocess._IO is not False: #check if
IO is required from IO state in process
                print("Process ID:%i sent to Blocked queue." %
self._curprocess._id)
                self._curprocess._quanta -= 1 #decrement the
current processes quanta
                self.addBlockedProcess(self._curprocess)  #add
to blockedQueue
                self._queue.pop(0)  #remove from top of queue
                x = input('Process ID: %i required input:'
%self._curprocess._id) #input required
            else:
                self._curprocess._quanta -= 1 #decrement the
quanta of the current process
                self.addProcess(self._curprocess) #return the
current process to the back of the ready queue
                self._queue.pop(0)#remove from top of queue

                if self._blockedqueue is not []: #if the
blocked queue is not empty
```

```
                    self._blockedqueue[0]._state = "ready"
#update the state of the first process in the blocked queue to
be 'ready'

self.addReadyProcess(self._blockedqueue[0]) #process at top of
blocked queue is sent to ready queue
                    print("Process ID:%i returned to ready
state" % self._blockedqueue[0]._id)
                    self._blockedqueue.pop(0) #remove from
blocked queue

            finish = self.checkPro() #calls checkPro function
to see if finished
            if finish: #if the above is true, print the
following message
                print("All processes are now complete and the
CPU is now idle.")
                return


def Schedule(CPU):
    CPU.runProcess()  # CPU runs processes


def test(): #test blocked
    cpu = CPU()
    p1 = Process(6, False, 'blocked', 1, 6) #process1
    cpu.addProcess(p1) #adding process1
    p2 = Process(0, True, 'ready', 2, 6) #process2
    cpu.addProcess(p2) #adding process 2
    p3 = Process(1, False, 'blocked', 3, 9) #process3
    cpu.addProcess(p3) #adding process3
    p4 = Process(8, True, 'ready', 4, 5) #process4
    cpu.addProcess(p4) #adding process4
    p4 = Process(3, True, 'ready', 5, 7) #process5
    cpu.addProcess(p4) #adding process5
    while len(cpu._queues) is not 0: #while there is something
in the queue
        num = randint(1, 1000) #generate a random number
        if num < 200: #if the number is less than 200
            print("Interrupt") #print
            print("Process Suspended") #print
            x = cpu.getCurProcess() #suspend process
            cpu.addSuspendedProcess(x)
        Schedule(cpu) #run above processes

if __name__ == "__main__":
    test() #only run the test block, if running the main
program
```

4. Testing

```
'''
Processes_and_Scheduling.py

Colin Kelleher - 117303363

CS2506 Operating Systems 2 – Lab2 - Processes and Scheduling
II
'''
```

```
def test():
    cpu = CPU()
    p1 = Process(3, False, 'ready', 1, 6)
    cpu.addProcess(p1)
    Process.PrintProcess(p1)

if __name__ == "__main__":
    test()
```

Result of running 'PrintProcess' on process1

```
main
/Library/Frameworks/Python.framework/Versions/3.6/bin/python3.6 "/Volumes/GoogleD
  SCIENCE/SECOND YEAR/Semester Two/CS2506 – Operating Systems 2/Lab02/main.py"
Process ID: 1 has a Quanta of: 3 is in the ready state and has a priority of: 6

Process finished with exit code 0
```

Result of running 'PrintProcess' on Process1 after it has been added

```
def test():
    cpu = CPU()
    p1 = Process(3, False, 'ready', 1, 7)
    cpu.addProcess(p1)
    p2 = Process(1, True, 'ready', 2, 6)
    cpu.addProcess(p2)
    p3 = Process(6, False, 'ready', 3, 9)
    cpu.addProcess(p3)
    p4 = Process(2, True, 'ready', 4, 10)
    cpu.addProcess(p4)
    Process.PrintProcess(p1)
    Process.PrintProcess(p2)
    Process.PrintProcess(p3)
    Process.PrintProcess(p4)

if __name__ == "__main__":
    test()
```

```
main
/Library/Frameworks/Python.framework/Versions/3.6/bin/python3.6 "/Volumes/GoogleDrive
  SCIENCE/SECOND YEAR/Semester Two/CS2506 - Operating Systems 2/Lab02/main.py"
Process ID: 1 has a Quanta of: 3 is in the ready state and has a priority of: 7
Process ID: 2 has a Quanta of: 1 is in the ready state and has a priority of: 6
Process ID: 3 has a Quanta of: 6 is in the ready state and has a priority of: 9
Process ID: 4 has a Quanta of: 2 is in the ready state and has a priority of: 10

Process finished with exit code 0
```

Result of adding 4 processes, and calling 'PrintProcess' on these four processes – it returns their ID, quanta, state & priority

```
def test():
    cpu = CPU()
    p1 = Process(3, False, 'ready', 1, 6)
    cpu.addBlockedProcess(p1)
    p2 = Process(1, True, 'ready', 2, 6)
    cpu.addProcess(p2)
    p3 = Process(1, False, 'ready', 3, 9)
    cpu.addProcess(p3)
    p4 = Process(8, True, 'ready', 4, 10)
    cpu.addProcess(p4)
    cpu.printBlockedQueue()
    cpu.printQueues()

if __name__ == "__main__":
    test()
```

Printing Queues

```
main
/Library/Frameworks/Python.framework/Versions/
  SCIENCE/SECOND YEAR/Semester Two/CS2506 - Op
Blocked Queue:
Process id:1
Queues
Process id:2
Process id:3
Process id:4

Process finished with exit code 0
```

Calling 'PrintblockedQueue' and 'PrintQueue' which prints the contents of the queues

```
def test():
    cpu = CPU()
    p1 = Process(3, False, 'ready', 1, 6)
    cpu.addProcess(p1)
    p2 = Process(1, True, 'ready', 2, 6)
    cpu.addProcess(p2)
    p3 = Process(1, False, 'ready', 3, 9)
    cpu.addProcess(p3)
    p4 = Process(8, True, 'ready', 4, 10)
    cpu.addProcess(p4)

if __name__ == "__main__":
    test()
```

```
main
/Library/Frameworks/Python.framework/Ve
  SCIENCE/SECOND YEAR/Semester Two/CS250
Process 1 added!
Process 2 added!
Process 3 added!
Process 4 added!

Process finished with exit code 0
```

Simple result of adding processes!

```
def test():
    cpu = CPU()
    p1 = Process(6, False, 'ready', 1, 6)
    cpu.addBlockedProcess(p1)

if __name__ == "__main__":
    test()
```

```
main
/Library/Frameworks/Python.framework/Ve
  SCIENCE/SECOND YEAR/Semester Two/CS250
Process 1 added to Blocked Queue

Process finished with exit code 0
```

Adding a blocked process

```
def test():
    cpu = CPU()
    p1 = Process(6, False, 'ready', 1, 6)
    cpu.addProcess(p1)
    p2 = Process(0, True, 'ready', 2, 6)
    cpu.addProcess(p2)
    p3 = Process(1, False, 'blocked', 3, 9)
    cpu.addProcess(p3)
    p4 = Process(8, True, 'ready', 4, 10)
    cpu.addProcess(p4)
    while len(cpu._queues) is not 0:
        num = randint(1, 1000)
        if num < 200:
            print("Interrupt")
            print("Process Suspended")
            x = cpu.getCurProcess()
            cpu.addSuspendedProcess(x)
        Schedule(cpu)

if __name__ == "__main__":
    test()
```

```
main
/Library/Frameworks/Python.framework/Versions/3.6/bin/python3.6 "/Volumes/Go
  SCIENCE/SECOND YEAR/Semester Two/CS2506 - Operating Systems 2/Lab02/main.py
Process ID:1 is currently running, with  priority: 6 and  quanta: 6
Process ID:3 is currently running, with  priority: 9 and  quanta: 1
Process ID:4 is currently running, with  priority: 10 and  quanta: 8
Process ID:4 sent to blocked queue.
Process 4 added to Blocked Queue
Process ID: 4 required input:2
Process ID:2 is currently running, with  priority: 6 and  quanta: 0
Process ID:2 completed
Interrupt
Process Suspended
Process ID:3 is currently running, with  priority: 9 and  quanta: 0
Process ID:3 completed
```

The Occurrence of an Interrupt causes the current process to be suspended – I used random numbers to trigger an interrupt

```
/Library/Frameworks/Python.framework/Versions/3.6/bin/python3.6 "/Vol
  SCIENCE/SECOND YEAR/Semester Two/CS2506 - Operating Systems 2/Lab02/
Process ID:1 is currently running, with  priority: 6 and  quanta: 6
Process ID:3 is currently running, with  priority: 9 and  quanta: 1
Process ID:4 is currently running, with  priority: 10 and  quanta: 8
Process ID:4 sent to blocked queue.
Process 4 added to Blocked Queue
Process ID: 4 required input:2
Process ID:2 is currently running, with  priority: 6 and  quanta: 0
Process ID:2 completed
Interrupt
Process 2 Suspended
Process ID:3 is currently running, with  priority: 9 and  quanta: 0
Process ID:3 completed
```

Slight modification on process being suspended – now includes process ID

```
def test():
    cpu = CPU()
    p1 = Process(6, False, 'ready', 1, 6)
    cpu.addProcess(p1)
    p2 = Process(0, True, 'ready', 2, 6)
    cpu.addProcess(p2)
    p3 = Process(1, False, 'blocked', 3, 9)
    cpu.addProcess(p3)
    p4 = Process(8, True, 'ready', 4, 10)
    cpu.addProcess(p4)
    while len(cpu._queues) is not 0:
        num = randint(1, 1000)
        if num < 200:
            print("Interrupt")
            print("Process Suspended")
            x = cpu.getCurProcess()
            cpu.addSuspendedProcess(x)
        Schedule(cpu)

if __name__ == "__main__":
    test()
```

main

```
/Library/Frameworks/Python.framework/Versions/3.6/bin/python3.6 "/Volu
  SCIENCE/SECOND YEAR/Semester Two/CS2506 - Operating Systems 2/Lab02/
Process ID:1 is currently running, with  priority: 6 and  quanta: 6
Process ID:3 is currently running, with  priority: 9 and  quanta: 1
Process ID:4 is currently running, with  priority: 10 and  quanta: 8
Process ID:4 sent to blocked queue.
Process 4 added to Blocked Queue
Process ID: 4 required input:2
Process ID:2 is currently running, with  priority: 6 and  quanta: 0
Process ID:2 completed
Process ID:3 is currently running, with  priority: 9 and  quanta: 0
Process ID:3 completed
Process ID:1 is currently running, with  priority: 6 and  quanta: 5
Process ID:4 returned to ready state
Process ID:4 is currently running, with  priority: 8 and  quanta: 7
```

Process ID 4 is being sent to the blocked queue
Process ID 4 is then added to the blocked queue
Process then requires I/O – I used an input from the user to simulate this – anything can be entered by the user to continue

```
Process ID:1 is currently running, with  priority: 6 and  quanta: 6
Process ID:3 is currently running, with  priority: 9 and  quanta: 1
Process ID:4 is currently running, with  priority: 10 and  quanta: 8
```

Process ID
Priority
Quanta

```
Process ID:4 returned to ready state
Process ID:4 is currently running, with  priority: 8 and  quanta: 7
Process ID:1 is currently running, with  priority: 6 and  quanta: 4
Process ID:4 is currently running, with  priority: 8 and  quanta: 6
Process ID:1 is currently running, with  priority: 6 and  quanta: 3
Process ID:4 is currently running, with  priority: 8 and  quanta: 5
Process ID:1 is currently running, with  priority: 6 and  quanta: 2
Process ID:4 is currently running, with  priority: 8 and  quanta: 4
Process ID:1 is currently running, with  priority: 6 and  quanta: 1
Process ID:4 is currently running, with  priority: 8 and  quanta: 3
Process ID:1 is currently running, with  priority: 6 and  quanta: 0
Process ID:1 completed
```

In the Above screenshot, we can see the quanta decreasing For Process ID: 1 we can see the quanta go 4,3,2,1,0, COMPLETE

```
def test():
    cpu = CPU()
    p1 = Process(6, False, 'blocked', 1, 6)
    cpu.addProcess(p1)
    p2 = Process(0, True, 'ready', 2, 6)
    cpu.addProcess(p2)
    p3 = Process(1, False, 'blocked', 3, 9)
    cpu.addProcess(p3)
    p4 = Process(8, True, 'ready', 4, 5)
    cpu.addProcess(p4)
    p4 = Process(3, True, 'ready', 5, 7)
    cpu.addProcess(p4)
    while len(cpu. queues) is not 0:
```

```
main
/Library/Frameworks/Python.framework/Versions/3.6/bin/python3.6
  "/Volumes/GoogleDrive/My Drive/COMPUTER SCIENCE/SECOND YEAR/Semester
  Operating Systems 2/Lab02/main.py"
Process ID:4 is currently running, with  priority: 5 and  quanta: 8
Process ID:4 sent to Blocked queue.
Process 4 added to Blocked Queue
Process ID: 4 required input:2
Process ID:1 is currently running, with  priority: 6 and  quanta: 6
Process ID:4 returned to ready state
Process ID:3 is currently running, with  priority: 9 and  quanta: 1
Process ID:4 is currently running, with  priority: 3 and  quanta: 7
Process ID:2 is currently running, with  priority: 6 and  quanta: 0
Process ID:2 completed
Process ID:3 is currently running, with  priority: 9 and  quanta: 0
Process ID:3 completed
Process ID:4 is currently running, with  priority: 3 and  quanta: 6
Process ID:5 is currently running, with  priority: 7 and  quanta: 3
Process ID:5 sent to Blocked queue.
Process 5 added to Blocked Queue
Process ID: 5 required input:4
Process ID:4 is currently running, with  priority: 3 and  quanta: 5
Process ID:5 returned to ready state
Process ID:5 is currently running, with  priority: 5 and  quanta: 2
Process ID:1 is currently running, with  priority: 6 and  quanta: 5
Process ID:4 is currently running, with  priority: 3 and  quanta: 4
Process ID:5 is currently running, with  priority: 5 and  quanta: 1
Process ID:1 is currently running, with  priority: 6 and  quanta: 4
Process ID:4 is currently running, with  priority: 3 and  quanta: 3
Process ID:5 is currently running, with  priority: 5 and  quanta: 0
Process ID:5 completed
```

Simulation with 5 processes