CS2506 Operating Systems II

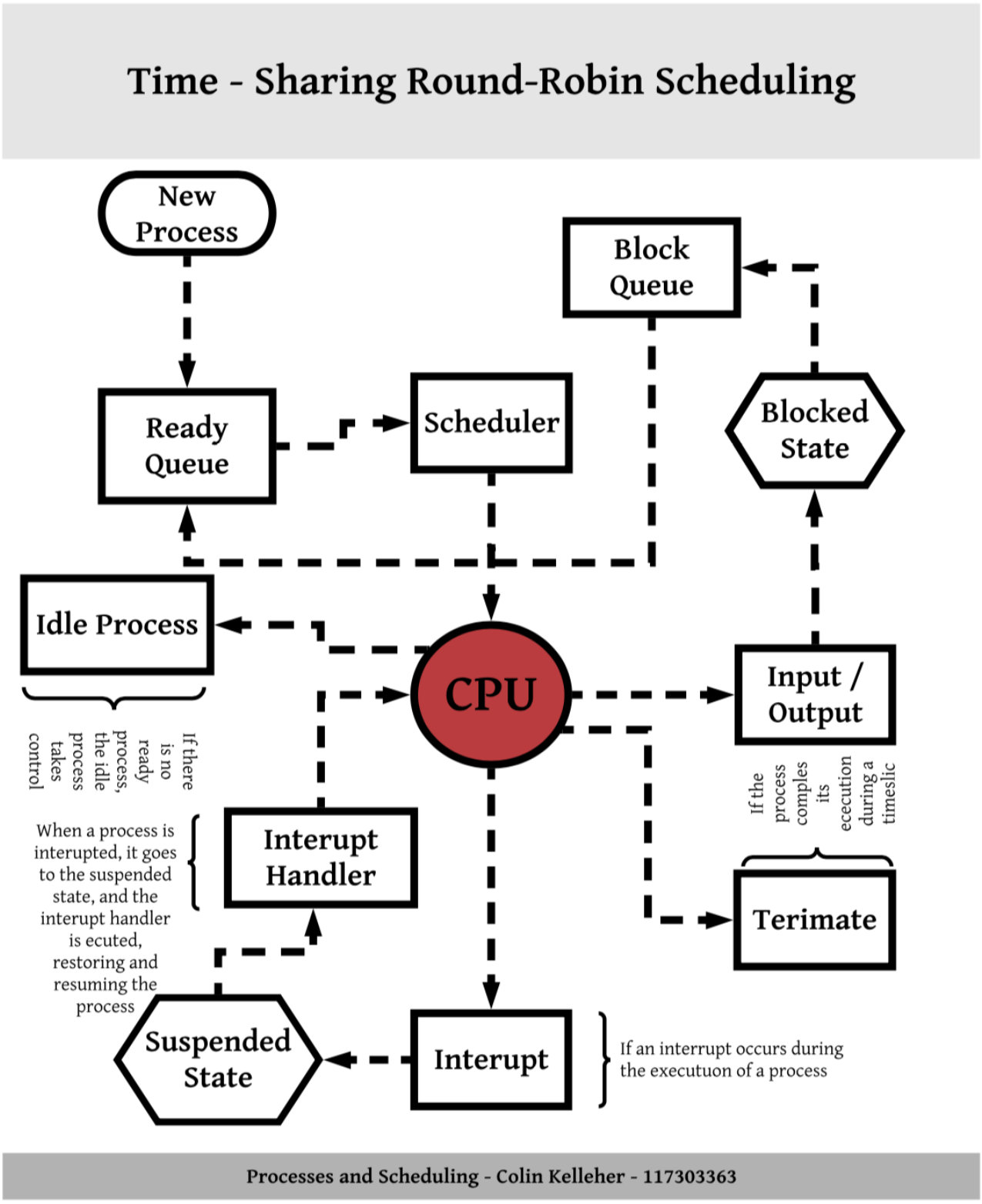
Processes & Scheduling

LAB1

Colin Kelleher – 117303363

1. Analysis

The Process Lifecycle



What is the difference between the blocked and suspended transitions of a process?

A process is in a blocked transition state when it cannot carry on without an event occurring or a change of state. E.g.: A process many be blocked if it is trying to call a device that is not online. Processes many also block when they require a user to input a value or make a selection.

A process is in a suspended transition state when none of the processes in main memory are in the "ready" state, and one of these processes is swapped into the Suspended queue, when the OS has performed this swapping process, it can run a newly created process or it can bring in a process from the suspended queue, it is preferred to bring in a previously suspended process

Events	Associated action(s)	Comments
Processes created	Inserted into the ready queue on their arrival	When a process is created and has a "READY" state – it is inserted into the ready queue on their arrival time – ordered by time
I/ O Completion	Triggers the switch of the process from the blocked state to the ready state	When an input / output process is finished, the process is moved from the blocked queue to the ready queue
Interrupt	Current process is suspended	When an interrupt occurs, the current process is moved into the suspended queue
I/O Request	Placed in the I/O Queue	When a process issues an I/O request, this process is then placed in the I/O Queue
Ready State	Scheduled on a core / CPU in a round-robin manner	When a process is in a ready queue, it is waiting to be scheduled on a core / CPU, this is done depending on the order in which they arrive – (First-come, First-served)

Blocked State	Waits for an event or change in state to occur before continuing	A process many be blocked if it is trying to call a device that is not online. Processes many also block when they require a user to input a value or make a selection.
Scheduler	Determines how to move processes between queues	The scheduler determines how to move processes within the system

2. Design

Processes_and_Scheduling.py - PSEUDOCODE

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

      def getState
            return state
      def setState
            set new state

      def getid
            return id
      def setid
            set new id

      def getEvent
            return event
      def setevent
            set new event

Class Scheduler:
      def __init__(self)
            current process
            ready queue = []
            blocked queue = []

      def get current process
            return current process
      def set current process
            set new current process
```

```
      def runProcess
            set the current process running in CPU
            decrement the quanta
            check if cycle is finished
                  if so remove from readyqueue
            if not finished check if I/O is needed
                  if it is, add it to blocked queue
                  remove from ready queue
                  require I/O (user input)
                  add it back to ready queue
            if the blocked queue is not empty
                  set the state of the process at top of queue to
ready and move it
      if both ready and blocked queues are empty – all the processes
have finished

      def addProcess (process):
            If the process is in the 'ready state'
                  Add to ready process (call 'addReadyProcess')
            Otherwise if the process is not in the ready state
                  Add to the blocked process (call
'addBlockedProcess')

      def addReadyProcess:
            append to ready queue and print message if successful

      def addBlockedProcess:
            append to blocked queue and print message if successful

      def addSuspendedProcess:
            Update the state of the process to be "Suspended"
            Append this to the queue

      def printBlockedQueue:
            loop to print each process in the queue (list)
            format using ID

      def printReadyQueue:
            loop to print each process in the queue (list)
            format using ID

      def Schedule: - used to run process
            As long as the readyQueue is not empty
                  Run Processes (runProcess)
      def test:
            Add processes and test running the processes
                  Use random numbers for Interrupt
                  Experiment with different parameters

If __name__ == 'main':
      test(); if within main program, run test block!
```

3. Programming

Processes_and_Scheduling.py

```
'''
Processes_and_Scheduling.py

Colin Kelleher - 117303363

CS2506 Operating Systems 2 - Lab1 - Processes and Scheduling
'''
from random import randint #import randint - used for random number for interrupt

class Process: #defining the 'Process; class
    def __init__(self, quantum, IO, state, id): #giving it the parameters
        self._quantum = quantum #the quantum quantity - allocation
        self._IO = IO # if I/O is required or not in the process
        self._state = state #ready or blocked - state of the process
        self._id = id #the process ID

    #getters and setters for state
    def setState(self,newstate): #set the new state, passing in the new state to be updated
        self._state = newstate #state is now equal to the new state
    def getState (self):
        return self._state

    #getters and setters for IO
    def getIO(self):
        return self._IO
    def setIO(self,newIO):
        self._IO = newIO

    #getters and setters for ID
    def getID(self):
        return self.id
    def setID (self,newID):
        self.id = newID

class Scheduler(object):
    def __init__(self):
```

```
        self._current = None #initialising the current process
to none
        self._readyQueue = [] #initialising the ready queue
        self._blockedQueue = [] #nitialising the blocked queue

    #getter and setter for the current process
    def getCurProcess(self): #call getCurProcess
        return self._current #returning the current process
    def setCurProcess(self,newCurProcess): #set the new
process, passing in the new process to be updated
        self._current = newCurProcess #current process is now
equal to the new process

    def runProcess(self): #This is where the processes are run
        self._current = self._readyQueue[0] #assign the first
item in the ready queue to the current process
        print("Process ID:%i is currently running" %
self._current._id) #print the name of the process
        self._current._quantum -= 1 #reduce the quanta number
by one
        if self._current._quantum == 0: #if cycle has finished
(quantum = 0
            self._readyQueue.pop(0)  # remove from ready queue
        elif self._current._IO == True: #if I/O is required
(=True_
            print("Process ID:%i sent to blocked queue." %
self._current._id) #print the process id send to blocked queue
            self.addBlockedProcess(self._current)  # add to
blockedQueue
            self._readyQueue.pop(0)  # remove from readyQueue
            print("Input required for Process %s:"
%self._current._id) #print input required for process
            if input(":"): #take a user input
                self._current._IO ==False #update the IO to
alse
                Process.setState(self._current, "ready") #wet
the state to ready
                self.addReadyProcess(self._current) #add to
ready queue
                print("Process added to Ready Queue")
                self._readyQueue.pop(0)  # remove from top of
readyQueue
        else: #otherwise if no I/O Required
            # return process to end of readyQueue when quanta
finished
            self.addReadyProcess(self._current) #add process
            self._readyQueue.pop(0)  # remove fromreadyQueue
            if self._blockedQueue is not []: #if the blocked
queue is not empty
```

```
                self._blockedQueue[0]._state = "ready" #update
the state of the process at front of blocked queue to ready
                self.addReadyProcess(self._blockedQueue[0])
#add process from front of blocked queue to ready queue
                print("Process ID:%i returned to ready queue"
% self._blockedQueue[0]._id) #print message with ID
                self._blockedQueue.pop(0) #remove from blocked
quque
        if self._readyQueue == [] and self._blockedQueue ==
[]: #if both queues are empty
            print("All processes comleted.")#processes are
finished running
            return #return message

    def addProcess(self, process):#method for determining
whether blocked or ready process
        if process._state == "ready": #if state of process is
ready
            self.addReadyProcess(process) #call addReadyProces
        elif process._state == "blocked": #if state of process
is blocked
            self.addBlockedProcess(process) #call
addBlockedProcess

    def addReadyProcess(self, process):
        self._readyQueue.append(process) #append the ready
process to the ready queue
        print("Process successfully added")

    def addBlockedProcess(self, process):
        self._blockedQueue.append(process) #append the blocked
process to the blocked queue
        print("Process added to Blocked Queue")

    def printBlockedQueue(self): #for each blocked process in
the blocked queue
        for blockedprocess in self._blockedQueue:
            print("Blocked Process ID: %s" %
self._current._id, blockedprocess) #print the blocked process
by ID

    def printReadyQueue(self):
        for readyprocess in self._readyQueue: #for each
process in the ready queue
            print("Ready Process ID: %s\n" %
self._current._id, readyprocess) #print the process by ID,
formatted as seen
```

```
    def addSuspendedProcess(self,process):
        Process.setState(self._current, "suspended") #update
the state of the current process to be suspended
        self._readyQueue.append(process) #then append it to
the ready queue for the simulation
        print("Process Suspended") #print "Process Suspended"


def Schedule(SCH): #scheduling function to run each process
    while SCH._readyQueue is not []: #as long as the ready
queue is not empty
        SCH.runProcess() #run the processes

def test(): #TEST BLOCK
    sch = Scheduler()
    #processName = Process(quantim, IO,state, ID)
    process1 = Process(5, False, "ready",1)
    sch.addProcess(process1)
    process2 = Process(2, False, "ready",3)
    sch.addProcess(process2)
    process3 = Process(10, False, "ready",2)
    sch.addProcess(process3)
    process4 = Process(2, True, "blocked", 4)
    sch.addProcess(process4)
    process5 = Process(12, False, "blocked",5)
    sch.addBlockedProcess(process5)
    while len(sch._readyQueue) is not 0: #while the ready
queue is not emplty
        num = randint(1, 1000) #select a random number
        if num < 200: #if number is less than 300
            print("Interrupt")
            print("Process Suspended")
            x = sch.getCurProcess() #get currentprocess
            sch.addSuspendedProcess(x) #suspend process
        Schedule(sch)

if __name__ == '__main__': #if we are within the main program
    test() #run the test block
```

4. Testing

```
'''
Processes_and_Scheduling.py

Colin Kelleher - 117303363

CS2506 Operating Systems 2 - Lab1 - Processes and Scheduling
'''
```

```
def test():
    sch = Scheduler()
    process1 = Process(5, False, "Blocked", 1)
    sch.addBlockedProcess(process1)
test()
```

```
Main
/Library/Frameworks/Python.framework/Versions/3
  "/Volumes/GoogleDrive/My Drive/COMPUTER SCIEN
  Operating Systems 2/Lab01/Main.py"
Process added to Blocked Queue

Process finished with exit code 0
```

Adding process to Blocked Queue

```
def test():
    sch = Scheduler()
    sch.printReadyQueue()
    sch.printBlockedQueue()
    Schedule(sch)
test()
```

```
Main
/Library/Frameworks/Python.framework/Vers
  "/Volumes/GoogleDrive/My Drive/COMPUTER
  Operating Systems 2/Lab01/Main.py"
Ready Process ID:1
Ready Process ID:3
Ready Process ID:2
Blocked Process ID:4

Process finished with exit code 0
```

Printing the Ready Queue and the Blocked Queue

```
def test():
    sch = Scheduler()
    process1 = Process(5, False, "ready", 1)
    sch.addProcess(process1)
test()
```

```
Main ×
/Library/Frameworks/Python.framework/Versions/
  "/Volumes/GoogleDrive/My Drive/COMPUTER SCIE
  Operating Systems 2/Lab01/Main.py"
Process successfully added

Process finished with exit code 0
```

Adding a process

```
def test():
    sch = Scheduler()
    process1 = Process(5, False, "ready",1)
    sch.addProcess(process1)
    process2 = Process(2, False, "ready",3)
    sch.addProcess(process2)
    process3 = Process(10, False, "ready",2)
    sch.addProcess(process3)
    process4 = Process(2, True, "blocked", 4)
    sch.addProcess(process4)
    while len(sch._readyQueue) is not 0:
        num = randint(1, 1000)
        if num < 100:
            print("Interrupt")
            x = sch.getCurProcess()
            sch.addSuspendedProcess(x)
        Schedule(sch)
test()
```

```
Main ×
/Library/Frameworks/Python.framework/Versions/3.6/b
  "/Volumes/GoogleDrive/My Drive/COMPUTER SCIENCE/S
  Operating Systems 2/Lab01/Main.py"
Process successfully added
Process successfully added
Process successfully added
Process added to Blocked Queue
Interrupt
Process Suspended
```

When an interrupt occurs, the process is suspended, to be dealt with by the interrupt handler – I used random numbers to trigger an interrupt

```
def test():
    sch = Scheduler()
    process1 = Process(5, False, "ready",1)
    sch.addProcess(process1)
    process2 = Process(2, False, "ready",3)
    sch.addProcess(process2)
    process3 = Process(10, False, "ready",2)
    sch.addProcess(process3)
    process4 = Process(2, True, "blocked", 4)
    sch.addProcess(process4)
    process5 = Process(12, False, "blocked",5)
    sch.addBlockedProcess(process5)
    process6 = Process(14, False, "ready", 6)
    sch.addProcess(process6)
    while len(sch._readyQueue) is not 0:
        num = randint(1, 200)
        if num < 200:
            print("Interrupt")
            x = sch.getCurProcess()
            sch.addSuspendedProcess(x)
        Schedule(sch)
```

```
Main
Process ID:1 is currently running
Process ID:4 returned to ready queue
Process ID:3 is currently running
Process ID:5 returned to ready queue
Process ID:2 is currently running
Process ID:6 is currently running
Process ID:1 is currently running
Process ID:4 is currently running
Process ID:4 sent to blocked queue.
Input required for Process 4:
:2
Process added to Ready Queue
```

When a process is blocked it is sent to the blocked Queue – Waits for I/O – I Simulate this using a user input as shown – this is then returned to the ready queue

Small simulation showing a blocked process

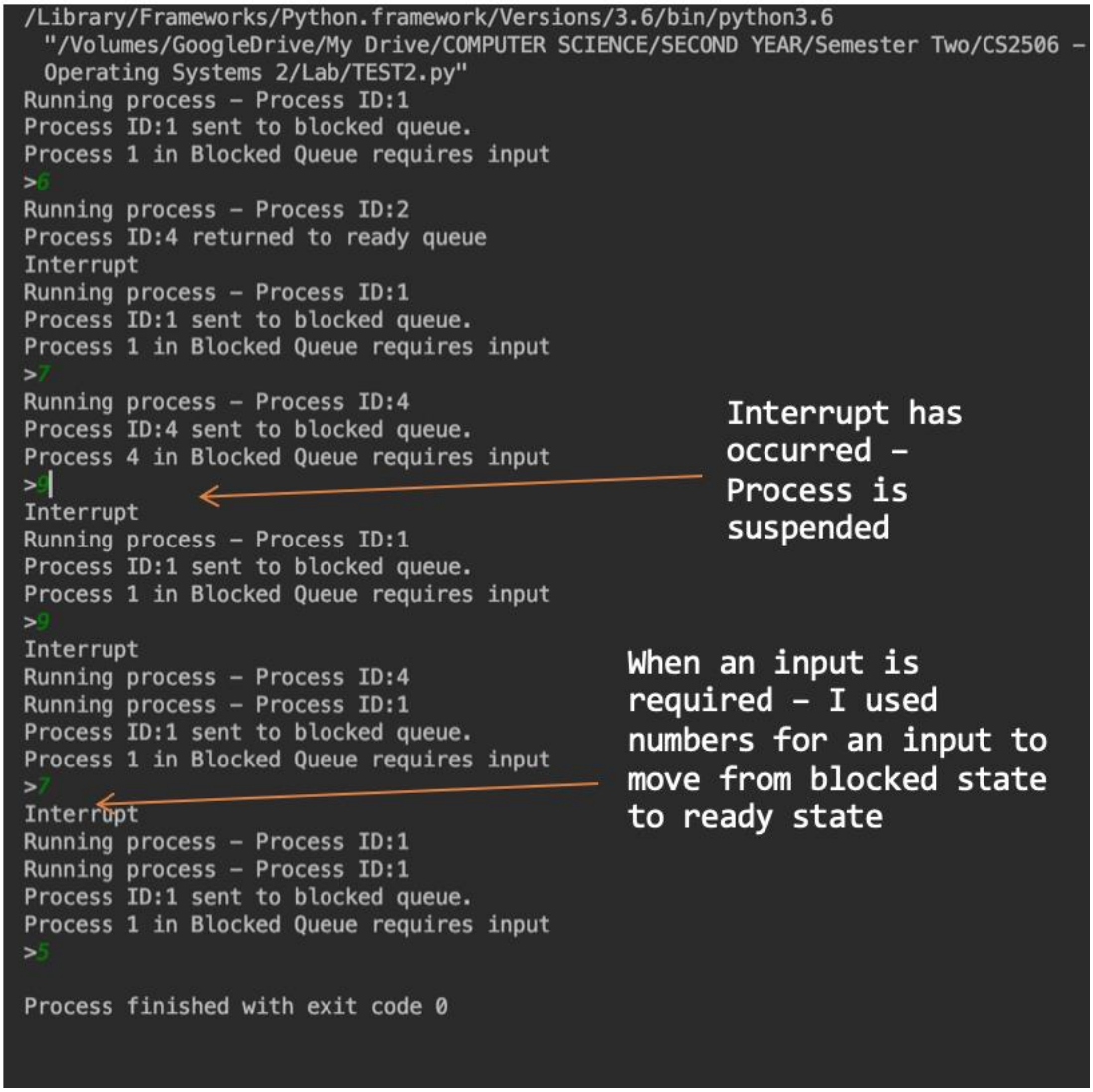
```
def test():
    sch = Scheduler()
    process1 = Process(5, False, "ready",1)
    sch.addProcess(process1)
    process2 = Process(2, False, "ready",3)
    sch.addProcess(process2)
    process3 = Process(10, False, "ready",2)
    sch.addProcess(process3)
    process4 = Process(2, True, "blocked", 4)
    sch.addProcess(process4)
    process5 = Process(12,False, "blocked",5)
    sch.addBlockedProcess(process5)
    while len(sch._readyQueue) is not 0:
        num = randint(1, 1000)
        if num < 100:
            print("Interrupt")
            x = sch.getCurProcess()
            sch.addSuspendedProcess(x)
        Schedule(sch)

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

```
Process ID:1 is currently running
Process ID:4 returned to ready queue
Process ID:3 is currently running
Process ID:5 returned to ready queue
Process ID:2 is currently running
Process ID:1 is currently running
Process ID:4 is currently running
Process ID:4 sent to blocked queue.
Input required for Process 4:
:2
Process added to Ready Queue
Process ID:5 is currently running
Process ID:4 returned to ready queue
Process ID:2 is currently running
Process ID:1 is currently running
Process ID:4 is currently running
Process ID:5 is currently running
Process ID:4 is currently running
Process ID:4 sent to blocked queue.
Input required for Process 4:
:3
Process added to Ready Queue
Process ID:1 is currently running
Process ID:4 returned to ready queue
Process ID:5 is currently running
Process ID:4 is currently running
Process ID:4 sent to blocked queue.
Input required for Process 4:
:4
Process added to Ready Queue
Process ID:4 is currently running
Process ID:4 sent to blocked queue.
Input required for Process 4:
:5
Process added to Ready Queue
Process ID:4 is currently running
Process ID:4 sent to blocked queue.
Input required for Process 4:
:6
Process added to Ready Queue
Process ID:4 is currently running
Process ID:4 sent to blocked queue.
Input required for Process 4:
:7
Process added to Ready Queue

Process finished with exit code 0
```

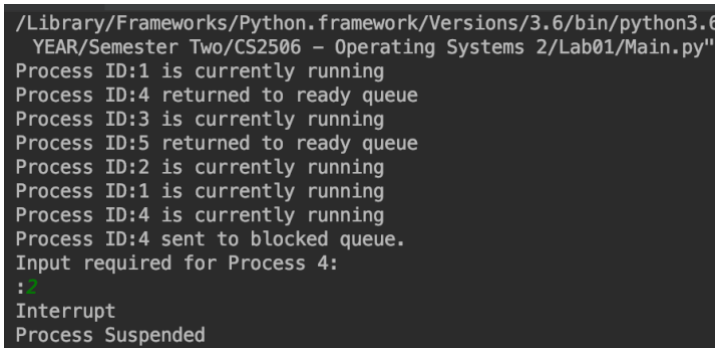
Simulation running the inputs as show

```
/Library/Frameworks/Python.framework/Versions/3.6/bin/python3.6
  "/Volumes/GoogleDrive/My Drive/COMPUTER SCIENCE/SECOND YEAR/Semester Two/CS2506 -
  Operating Systems 2/Lab01/Main.py"
Running process - Process ID:1
Process ID:4 returned to ready queue
Running process - Process ID:3
Process ID:5 returned to ready queue
Running process - Process ID:2
Running process - Process ID:6
Running process - Process ID:1
Running process - Process ID:4
Process ID:4 sent to blocked queue.
Input required for Process 4:
:22
Process added to Ready Queue
Running process - Process ID:5
Process ID:4 returned to ready queue
Running process - Process ID:2
Running process - Process ID:6
Running process - Process ID:1
Running process - Process ID:4
Running process - Process ID:5
Running process - Process ID:4
Process ID:4 sent to blocked queue.
Input required for Process 4:
:2
Process added to Ready Queue
Running process - Process ID:6
Running process - Process ID:1
Process ID:4 returned to ready queue
Running process - Process ID:5
Running process - Process ID:4
Process ID:4 sent to blocked queue.
Input required for Process 4:
:1
Process added to Ready Queue
Running process - Process ID:4
Process ID:4 sent to blocked queue.
Input required for Process 4:
:2
Process added to Ready Queue
Running process - Process ID:4
Process ID:4 sent to blocked queue.
Input required for Process 4:
:34
Process added to Ready Queue
Running process - Process ID:4
Process ID:4 sent to blocked queue.
Input required for Process 4:
:4
Process added to Ready Queue

Process finished with exit code 0
```

Simulation with Six Processes



Simulation with Multiple Interrupts



Process interrupted and suspended