现代操作系统第二次作业

Modern Operating Systems homework

陈涵 2019 级软件工程 1 班 201936380086

chapter 6 deadlocks problems in P465

23. Consider the previous problem again, but now with p processes each needing a maximum of m resources and a total of r resources available. What condition must hold to make the system deadlock free?

Answer:

To make the system deadlock free, while all p processes have occupied m-1 resources, at least one of processes can get one more resource. So, $r \geq p(m-1)+1$.

26. A system has four processes and five allocatable resources. The current allocation and maximum needs are as follows:

	Allocated					Maximum					Available				
Process A	1	0	2	1	1	1	1	2	1	3	0	0	x	1	1
Process B	2	0	1	1	0	2	2	2	1	0					
Process C	1	1	0	1	0	2	1	3	1	0					
Process D	1	1	1	1	0	1	1	2	2	1					

What is the smallest value of x for which this is a safe state?

Answer:

If x=0, the system will deadlock immediately. If x=1, because of lack of the first 2 types resources, Process A, B and C can’t run. Process D will work alone when Process D finished:

Table 1 Process D running

	Allocated					Maximum					Available				
Process D	1	1	2	2	1	1	1	2	2	1	0	0	0	0	0
Process A	1	0	2	1	1	1	1	2	1	3					
Process B	2	0	1	1	0	2	2	2	1	0					
Process C	1	1	0	1	0	2	1	3	1	0					

Table 2 Process D finished

	Allocated					Maximum					Available				
Process D	0	0	0	0	0	1	1	2	2	1	1	1	2	2	1
Process A	1	0	2	1	1	1	1	2	1	3					
Process B	2	0	1	1	0	2	2	2	1	0					
Process C	1	1	0	1	0	2	1	3	1	0					

Then process C will deadlock, because the third type resource is lack. If x=2, Process C can work well after process D finished and process A and process B will have enough resources to serve in the system, so the smallest value for x is 2.

31. One way to prevent deadlocks is to eliminate the hold-and-wait condition. In the text it was proposed that before asking for a new resource, a process must first release whatever resources it already holds (assuming that is possible). However, doing so introduces the danger that it may get the new resource but lose some of the existing ones to competing processes. Propose an improvement to this scheme.
Answer:

In my view, one of the improvements is that while a process asking for a new resource, the process hold the resources it has got and obtain the new one if the new resource is available. Else, the process will release all the resources it has got. In this situation, the system is free from deadlock. But the resources must can be released like printer or CD driver.

34. Explain the differences between deadlock, livelock, and starvation.
Answer:

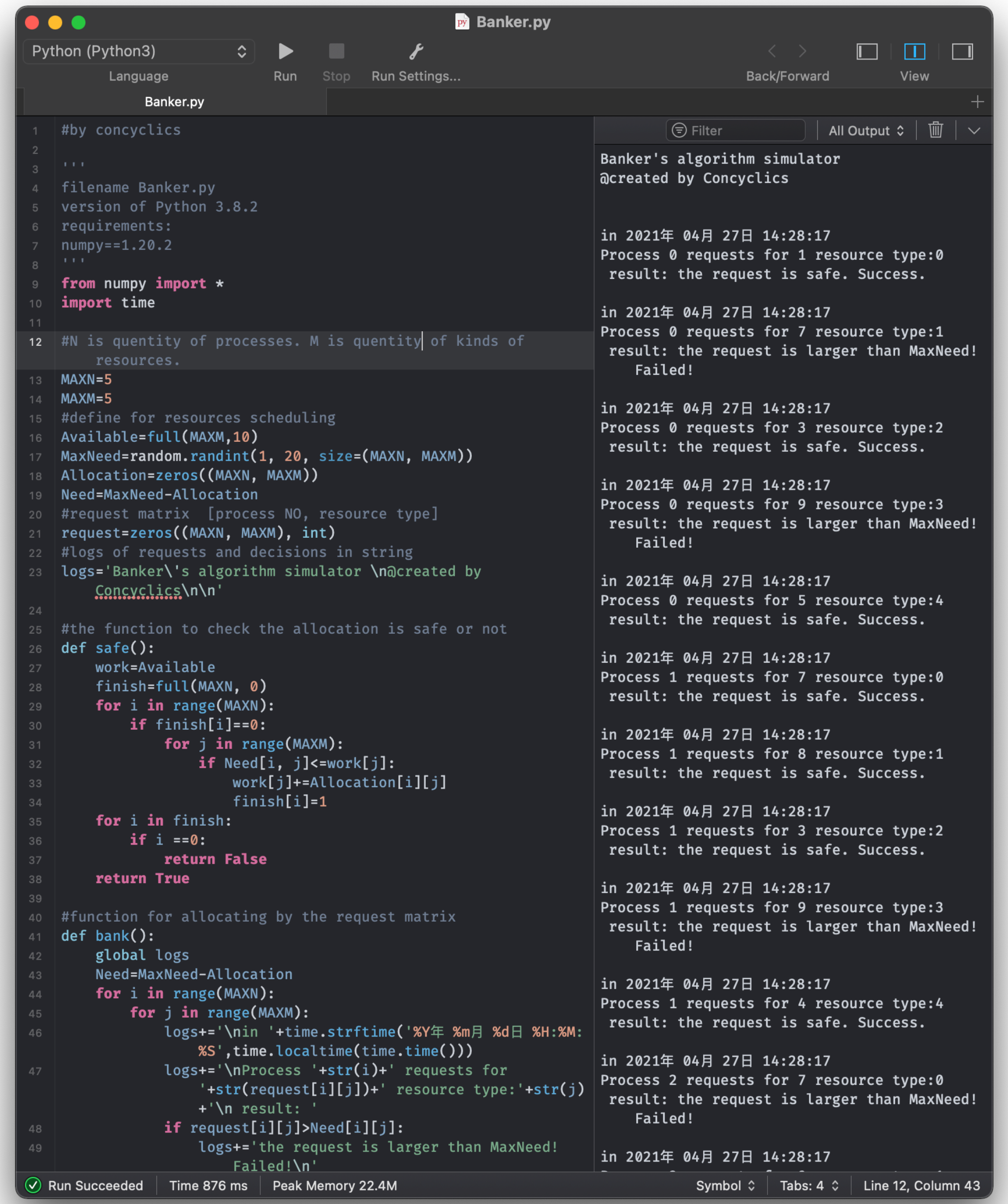
deadlock	Both the processes hold a part of resources and wait for resources held by another one. The processes in a deadlock will be block and can't get out from this situation by themselves.
livelock	A process tries to be polite by giving up to obtain new resource and leaves it to the other process while another process do the same. The processes in a livelock won't be block and will repeatedly check for resources they need and cost CPU times. They can get out from this situation themselves occasionally.
starvation	A process needs one kind of resource while this resource is obtained by other processes with higher priority in system scheduling. The process in starvation will wait for a long time and even can't get enough resources ever.

41. Program a simulation of the banker's algorithm. Your program should cycle through each of the bank clients asking for a request and evaluating whether it is safe or unsafe. Output a log of requests and decisions to a file.
Answer:

Filename: Banker.py
version of Python 3.8.2
requirements:
numpy==1.20.2
codes:

```
#by concyclics

'''
filename Banker.py
version of Python 3.8.2
requirements:
numpy==1.20.2
'''
```

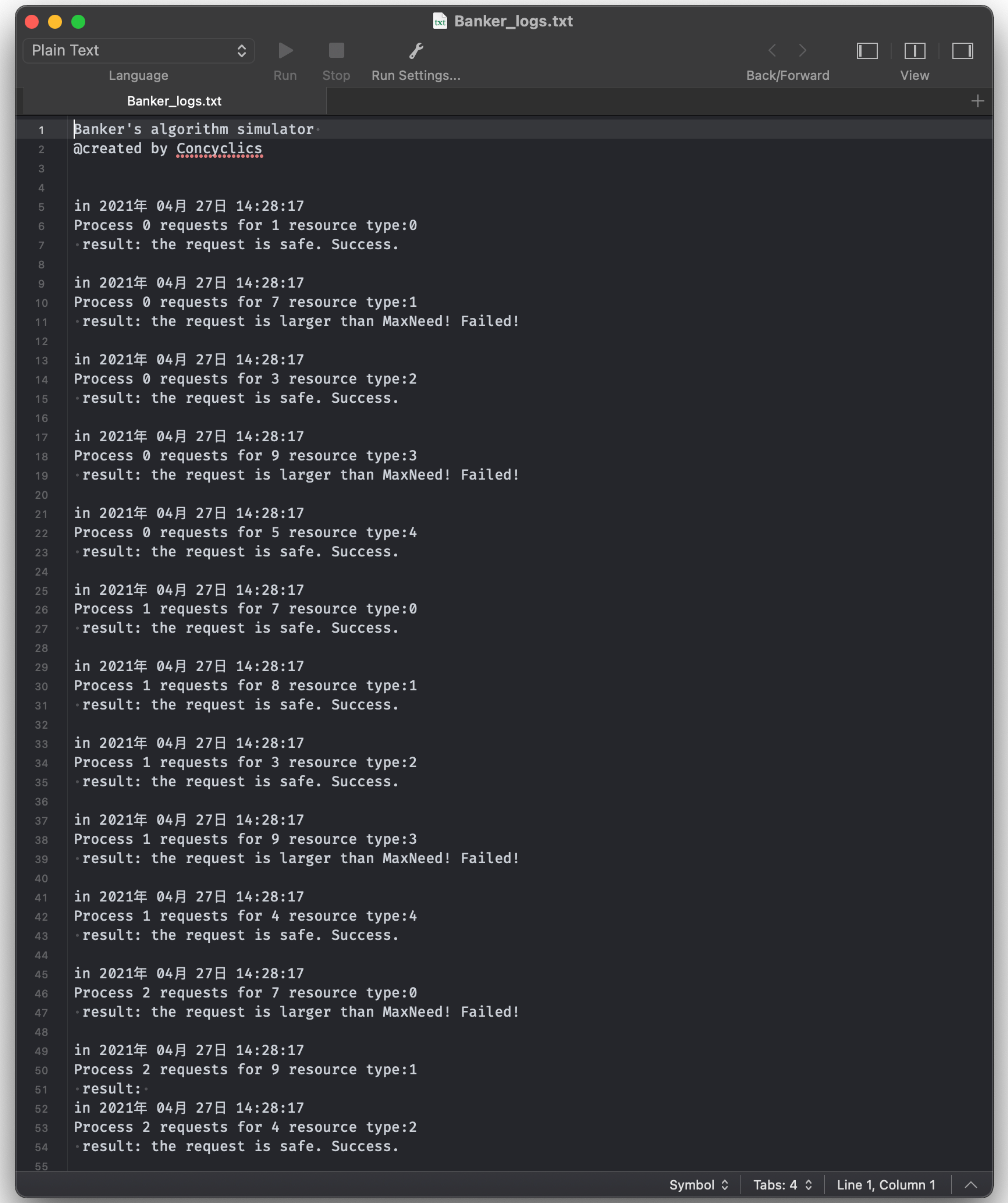
```
from numpy import *
import time

#N is quantity of processes. M is quantity of kinds of resources.
MAXN=5
MAXM=5
#define for resources scheduling
Available=full(MAXM,10)
MaxNeed=random.randint(1, 20, size=(MAXN, MAXM))
Allocation=zeros((MAXN, MAXM))
Need=MaxNeed-Allocation
#request matrix    [process NO, resource type]
request=zeros((MAXN, MAXM), int)
#logs of requests and decisions in string
logs='Banker\'s algorithm simulator \n@created by Concyclics\n\n'

#the function to check the allocation is safe or not
def safe():
                work=Available
                finish=full(MAXN, 0)
                for i in range(MAXN):
                          if finish[i]==0:
                                      for j in range(MAXM):
                                                  if Need[i, j]<=work[j]:
                                                          work[j]+=Allocation[i][j]
                                                          finish[i]=1
                for i in finish:
                          if i ==0:
                                      return False
                return True

#function for allocating by the request matrix
def bank():
                global logs
                Need=MaxNeed-Allocation
                for i in range(MAXN):
                          for j in range(MAXM):

                                      logs+='\nin '+time.strftime('%Y年  %m月  %d

                                 日  %H:%M:%S',time.localtime(time.time()))

                                      logs+='\nProcess '+str(i)+' requests for
                                 '+str(request[i][j])+' resource type:'+str(j)+'\n result: '
                                      if request[i][j]>Need[i][j]:
                                        logs+='the request is larger than MaxNeed!
                                Failed!\n'
                                elif request[i][j]<=Available[j]:
                                        Available[j]-=request[i][j]
                                        Allocation[i][j]+=request[i][j]
                                        Need[i][j]-=request[i][j]
                                        if safe()==False:
                                                Available[j]+=request[i][j]
                                                Allocation[i][j]-=request[i][j]
                                                Need[i][j]+=request[i][j]
                                                logs+='the request will make the
                                        system danger! Wait!\n'
                                        else:
                                                logs+='the request is safe. Success.\n'


if __name__=='__main__':
        request+=random.randint(1, 10, size=(MAXN,MAXM))
        bank()
        print(logs)
        #write logs into file Banker_logs.txt
        with open('Banker_logs.txt','w',encoding='utf-8') as fileLogs:
                fileLogs.write(logs)
```



```python
#by concyclics

'''
filename Banker.py
version of Python 3.8.2
requirements:
numpy==1.20.2
'''
from numpy import *
import time

#N is quentity of processes. M is quentity of kinds of resources.
MAXN=5
MAXM=5
#define for resources scheduling
Available=full(MAXM,10)
MaxNeed=random.randint(1, 20, size=(MAXN, MAXM))
Allocation=zeros((MAXN, MAXM))
Need=MaxNeed-Allocation
#request matrix  [process NO, resource type]
request=zeros((MAXN, MAXM), int)
#logs of requests and decisions in string
logs='Banker\'s algorithm simulator \n@created by Concyclics\n\n'

#the function to check the allocation is safe or not
def safe():
    work=Available
    finish=full(MAXN, 0)
    for i in range(MAXN):
        if finish[i]==0:
            for j in range(MAXM):
                if Need[i, j]<=work[j]:
                    work[j]+=Allocation[i][j]
                    finish[i]=1
    for i in finish:
        if i ==0:
            return False
    return True

#function for allocating by the request matrix
def bank():
    global logs
    Need=MaxNeed-Allocation
    for i in range(MAXN):
        for j in range(MAXM):
            logs+='\nin '+time.strftime('%Y年 %m月 %d日 %H:%M:%S',time.localtime(time.time()))
            logs+='\nProcess '+str(i)+' requests for '+str(request[i][j])+' resource type:'+str(j)+'\n result: '
            if request[i][j]>Need[i][j]:
                logs+='the request is larger than MaxNeed! Failed!\n'
```

```
Banker's algorithm simulator
@created by Concyclics


in 2021年 04月 27日 14:28:17
Process 0 requests for 1 resource type:0
 result: the request is safe. Success.

in 2021年 04月 27日 14:28:17
Process 0 requests for 7 resource type:1
 result: the request is larger than MaxNeed!
    Failed!

in 2021年 04月 27日 14:28:17
Process 0 requests for 3 resource type:2
 result: the request is safe. Success.

in 2021年 04月 27日 14:28:17
Process 0 requests for 9 resource type:3
 result: the request is larger than MaxNeed!
    Failed!

in 2021年 04月 27日 14:28:17
Process 0 requests for 5 resource type:4
 result: the request is safe. Success.

in 2021年 04月 27日 14:28:17
Process 1 requests for 7 resource type:0
 result: the request is safe. Success.

in 2021年 04月 27日 14:28:17
Process 1 requests for 8 resource type:1
 result: the request is safe. Success.

in 2021年 04月 27日 14:28:17
Process 1 requests for 3 resource type:2
 result: the request is safe. Success.

in 2021年 04月 27日 14:28:17
Process 1 requests for 9 resource type:3
 result: the request is larger than MaxNeed!
    Failed!

in 2021年 04月 27日 14:28:17
Process 1 requests for 4 resource type:4
 result: the request is safe. Success.

in 2021年 04月 27日 14:28:17
Process 2 requests for 7 resource type:0
 result: the request is larger than MaxNeed!
    Failed!

in 2021年 04月 27日 14:28:17
```

figure1 code running



```
Banker's algorithm simulator
@created by Concyclics


in 2021年 04月 27日 14:28:17
Process 0 requests for 1 resource type:0
 result: the request is safe. Success.

in 2021年 04月 27日 14:28:17
Process 0 requests for 7 resource type:1
 result: the request is larger than MaxNeed! Failed!

in 2021年 04月 27日 14:28:17
Process 0 requests for 3 resource type:2
 result: the request is safe. Success.

in 2021年 04月 27日 14:28:17
Process 0 requests for 9 resource type:3
 result: the request is larger than MaxNeed! Failed!

in 2021年 04月 27日 14:28:17
Process 0 requests for 5 resource type:4
 result: the request is safe. Success.

in 2021年 04月 27日 14:28:17
Process 1 requests for 7 resource type:0
 result: the request is safe. Success.

in 2021年 04月 27日 14:28:17
Process 1 requests for 8 resource type:1
 result: the request is safe. Success.

in 2021年 04月 27日 14:28:17
Process 1 requests for 3 resource type:2
 result: the request is safe. Success.

in 2021年 04月 27日 14:28:17
Process 1 requests for 9 resource type:3
 result: the request is larger than MaxNeed! Failed!

in 2021年 04月 27日 14:28:17
Process 1 requests for 4 resource type:4
 result: the request is safe. Success.

in 2021年 04月 27日 14:28:17
Process 2 requests for 7 resource type:0
 result: the request is larger than MaxNeed! Failed!

in 2021年 04月 27日 14:28:17
Process 2 requests for 9 resource type:1
 result: 
in 2021年 04月 27日 14:28:17
Process 2 requests for 4 resource type:2
 result: the request is safe. Success.
```

figure2 output in Banker_logs.txt