

lockinresource.py (Page 1 of 2)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # lockinresource.py
5: # Author: Bob Dondero
6: #-----
7:
8: import threading
9:
10: #-----
11:
12: class BankAcct:
13:
14:     def __init__(self):
15:         self._balance = 0
16:         self._lock = threading.RLock()
17:
18:     def get_balance(self):
19:         self._lock.acquire()
20:         try:
21:             return self._balance
22:         finally:
23:             self._lock.release()
24:
25:     def deposit(self, amount):
26:         self._lock.acquire()
27:         temp = self._balance
28:         temp += amount
29:         print(temp)
30:         self._balance = temp
31:         self._lock.release()
32:
33:     def withdraw(self, amount):
34:         self._lock.acquire()
35:         temp = self._balance
36:         temp -= amount
37:         print(temp)
38:         self._balance = temp
39:         self._lock.release()
40:
41: #-----
42:
43: class DepositThread (threading.Thread):
44:
45:     def __init__(self, bank_acct):
46:         threading.Thread.__init__(self)
47:         self._bank_acct = bank_acct
48:
49:     def run(self):
50:         for _ in range(10):
51:             self._bank_acct.deposit(1)
52:
53: #-----
54:
55: class WithdrawThread (threading.Thread):
56:
57:     def __init__(self, bank_acct):
58:         threading.Thread.__init__(self)
59:         self._bank_acct = bank_acct
60:
61:     def run(self):
62:         for _ in range(5):
63:             self._bank_acct.withdraw(2)
64:
65: #-----

```

lockinresource.py (Page 2 of 2)

```

66:
67: def main():
68:
69:     bank_acct = BankAcct()
70:
71:     deposit_thread = DepositThread(bank_acct)
72:     withdraw_thread = WithdrawThread(bank_acct)
73:
74:     deposit_thread.start()
75:     withdraw_thread.start()
76:
77:     deposit_thread.join()
78:     withdraw_thread.join()
79:
80:     print('Final balance:', bank_acct.get_balance())
81:
82: #-----
83:
84: if __name__ == '__main__':
85:     main()

```

lockinresourcew.py (Page 1 of 2)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # lockinresourcew.py
5: # Author: Bob Dondero
6: #-----
7:
8: import threading
9:
10: #-----
11:
12: class BankAcct:
13:
14:     def __init__(self):
15:         self._balance = 0
16:         self._lock = threading.RLock()
17:
18:     def get_balance(self):
19:         with self._lock:
20:             return self._balance
21:
22:     def deposit(self, amount):
23:         with self._lock:
24:             temp = self._balance
25:             temp += amount
26:             print(temp)
27:             self._balance = temp
28:
29:     def withdraw(self, amount):
30:         with self._lock:
31:             temp = self._balance
32:             temp -= amount
33:             print(temp)
34:             self._balance = temp
35:
36: #-----
37:
38: class DepositThread (threading.Thread):
39:
40:     def __init__(self, bank_acct):
41:         threading.Thread.__init__(self)
42:         self._bank_acct = bank_acct
43:
44:     def run(self):
45:         for _ in range(10):
46:             self._bank_acct.deposit(1)
47:
48: #-----
49:
50: class WithdrawThread (threading.Thread):
51:
52:     def __init__(self, bank_acct):
53:         threading.Thread.__init__(self)
54:         self._bank_acct = bank_acct
55:
56:     def run(self):
57:         for _ in range(5):
58:             self._bank_acct.withdraw(2)
59:
60: #-----
61:
62: def main():
63:
64:     bank_acct = BankAcct()
65:

```

lockinresourcew.py (Page 2 of 2)

```

66:     deposit_thread = DepositThread(bank_acct)
67:     withdraw_thread = WithdrawThread(bank_acct)
68:
69:     deposit_thread.start()
70:     withdraw_thread.start()
71:
72:     deposit_thread.join()
73:     withdraw_thread.join()
74:
75:     print('Final balance:', bank_acct.get_balance())
76:
77: #-----
78:
79: if __name__ == '__main__':
80:     main()

```

conditions.py (Page 1 of 2)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # conditions.py
5: # Author: Bob Dondero
6: #-----
7:
8: import threading
9:
10: #-----
11:
12: class BankAcct:
13:
14:     def __init__(self):
15:         self._balance = 0
16:         self._lock = threading.RLock()
17:         self._condition = threading.Condition(self._lock)
18:
19:     def get_balance(self):
20:         self._lock.acquire()
21:         try:
22:             return self._balance
23:         finally:
24:             self._lock.release()
25:
26:     def deposit(self, amount):
27:         self._condition.acquire()
28:         self._balance += amount
29:         print(self._balance)
30:         self._condition.notify_all()
31:         self._condition.release()
32:
33:     def withdraw(self, amount):
34:         self._condition.acquire()
35:         while self._balance < amount:
36:             self._condition.wait()
37:         self._balance -= amount
38:         print(self._balance)
39:         self._condition.release()
40:
41: #-----
42:
43: class DepositThread (threading.Thread):
44:
45:     def __init__(self, bank_acct):
46:         threading.Thread.__init__(self)
47:         self._bank_acct = bank_acct
48:
49:     def run(self):
50:         for _ in range(10):
51:             self._bank_acct.deposit(1)
52:
53: #-----
54:
55: class WithdrawThread (threading.Thread):
56:
57:     def __init__(self, bank_acct):
58:         threading.Thread.__init__(self)
59:         self._bank_acct = bank_acct
60:
61:     def run(self):
62:         for _ in range(5):
63:             self._bank_acct.withdraw(2)
64:
65: #-----

```

conditions.py (Page 2 of 2)

```

66:
67: def main():
68:     bank_acct = BankAcct()
69:
70:     deposit_thread = DepositThread(bank_acct)
71:     withdraw_thread = WithdrawThread(bank_acct)
72:
73:     deposit_thread.start()
74:     withdraw_thread.start()
75:
76:     deposit_thread.join()
77:     withdraw_thread.join()
78:
79:     print('Final balance:', bank_acct.get_balance())
80:
81: #-----
82:
83: if __name__ == '__main__':
84:     main()

```

conditionsw.py (Page 1 of 2)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # conditionsw.py
5: # Author: Bob Dondero
6: #-----
7:
8: import threading
9:
10: #-----
11:
12: class BankAcct:
13:
14:     def __init__(self):
15:         self._balance = 0
16:         self._lock = threading.RLock()
17:         self._condition = threading.Condition(self._lock)
18:
19:     def get_balance(self):
20:         with self._lock:
21:             return self._balance
22:
23:     def deposit(self, amount):
24:         with self._condition:
25:             self._balance += amount
26:             print(self._balance)
27:             self._condition.notify_all()
28:
29:     def withdraw(self, amount):
30:         with self._condition:
31:             while self._balance < amount:
32:                 self._condition.wait()
33:             self._balance -= amount
34:             print(self._balance)
35:
36: #-----
37:
38: class DepositThread (threading.Thread):
39:
40:     def __init__(self, bank_acct):
41:         threading.Thread.__init__(self)
42:         self._bank_acct = bank_acct
43:
44:     def run(self):
45:         for _ in range(10):
46:             self._bank_acct.deposit(1)
47:
48: #-----
49:
50: class WithdrawThread (threading.Thread):
51:
52:     def __init__(self, bank_acct):
53:         threading.Thread.__init__(self)
54:         self._bank_acct = bank_acct
55:
56:     def run(self):
57:         for _ in range(5):
58:             self._bank_acct.withdraw(2)
59:
60: #-----
61:
62: def main():
63:     bank_acct = BankAcct()
64:
65:     deposit_thread = DepositThread(bank_acct)

```

conditionsw.py (Page 2 of 2)

```

66:     withdraw_thread = WithdrawThread(bank_acct)
67:
68:     deposit_thread.start()
69:     withdraw_thread.start()
70:
71:     deposit_thread.join()
72:     withdraw_thread.join()
73:
74:     print('Final balance:', bank_acct.get_balance())
75:
76: #-----
77:
78: if __name__ == '__main__':
79:     main()

```

prodconprocesses.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # prodconprocesses.py
5: # Author: Bob Dondero
6: #-----
7:
8: import multiprocessing
9:
10: #-----
11:
12: def produce(producer_conn):
13:     for i in range(100):
14:         producer_conn.send(i)
15:         print('Produced:', i)
16:
17: #-----
18:
19: def consume(consumer_conn):
20:     for _ in range(100):
21:         num = consumer_conn.recv()
22:         print('Consumed:', num)
23:
24: #-----
25:
26: def main():
27:
28:     producer_conn, consumer_conn = multiprocessing.Pipe()
29:
30:     producer_process = multiprocessing.Process(
31:         target=produce, args=[producer_conn])
32:     consumer_process = multiprocessing.Process(
33:         target=consume, args=[consumer_conn])
34:
35:     producer_process.start()
36:     consumer_process.start()
37:
38:     producer_process.join()
39:     consumer_process.join()
40:
41:     print('Finished')
42:
43: #-----
44:
45: if __name__ == '__main__':
46:     main()

```

prodconthreads.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # prodconthreads.py
5: # Author: Bob Dondero
6: #-----
7:
8: import threading
9: import queue
10:
11: QUEUE_SIZE = 8 # Arbitrary
12:
13: #-----
14:
15: class ProducerThread (threading.Thread):
16:
17:     def __init__(self, intqueue):
18:         threading.Thread.__init__(self)
19:         self._intqueue = intqueue
20:
21:     def run(self):
22:         for i in range(100):
23:             self._intqueue.put(i)
24:             print('Produced:', i)
25:
26: #-----
27:
28: class ConsumerThread (threading.Thread):
29:
30:     def __init__(self, intqueue):
31:         threading.Thread.__init__(self)
32:         self._intqueue = intqueue
33:
34:     def run(self):
35:         for _ in range(100):
36:             num = self._intqueue.get()
37:             print('Consumed:', num)
38:
39: #-----
40:
41: def main():
42:
43:     intqueue = queue.Queue(QUEUE_SIZE)
44:
45:     producer_thread = ProducerThread(intqueue)
46:     consumer_thread = ConsumerThread(intqueue)
47:
48:     producer_thread.start()
49:     consumer_thread.start()
50:
51:     producer_thread.join()
52:     consumer_thread.join()
53:
54:     print('Finished')
55:
56: #-----
57:
58: if __name__ == '__main__':
59:     main()

```

Conditions.java (Page 1 of 2)

```

1: //-----
2: // Conditions.java
3: // Author: Bob Dondero
4: //-----
5:
6: class Bank
7: {
8:     private int balance = 0;
9:
10:    public synchronized int getBalance()
11:    {
12:        return balance;
13:    }
14:
15:    public synchronized void deposit(int amount)
16:    {
17:        balance += amount;
18:        System.out.println(balance);
19:        notifyAll();
20:    }
21:
22:    public synchronized void withdraw(int amount)
23:    {
24:        try
25:        {
26:            while (balance < amount)
27:                wait();
28:            balance -= amount;
29:            System.out.println(balance);
30:        }
31:        catch (InterruptedException e)
32:        {
33:            Thread.currentThread().interrupt();
34:        }
35:    }
36: }
37:
38: //-----
39:
40: class DepositThread extends Thread
41: {
42:     private Bank bank;
43:
44:     public DepositThread(Bank bank)
45:     {
46:         this.bank = bank;
47:     }
48:
49:     public void run()
50:     {
51:         for (int i = 0; i < 10; i++)
52:             bank.deposit(1);
53:     }
54: }
55:
56: //-----
57:
58: class WithdrawThread extends Thread
59: {
60:     private Bank bank;
61:
62:     public WithdrawThread(Bank bank)
63:     {
64:         this.bank = bank;
65:     }

```

Conditions.java (Page 2 of 2)

```

66:
67:     public void run()
68:     {
69:         for (int i = 0; i < 5; i++)
70:             bank.withdraw(2);
71:     }
72: }
73:
74: //-----
75:
76: public class Conditions
77: {
78:     public static void main(String[] args)
79:     {
80:         Bank bank = new Bank();
81:
82:         Thread depositThread = new DepositThread(bank);
83:         Thread withdrawThread = new WithdrawThread(bank);
84:
85:         depositThread.start();
86:         withdrawThread.start();
87:
88:         try
89:         {
90:             depositThread.join();
91:             depositThread.join();
92:         }
93:         catch (InterruptedException e)
94:         {
95:             Thread.currentThread().interrupt();
96:         }
97:
98:         System.out.println("Final balance: " + bank.getBalance());
99:     }
100: }

```

conditions.c (Page 1 of 2)

```

1: /*-----*/
2: /* conditions.c */
3: /* Author: Bob Dondero */
4: /* To build use the -pthread option */
5: /*-----*/
6:
7: #include <pthread.h>
8: #include <stdio.h>
9: #include <stdlib.h>
10:
11: /*-----*/
12:
13: /* Think of this as a BankAcct object. */
14:
15: static int bankAcctBalance = 0;
16: static pthread_mutex_t bankAcctMutex;
17: static pthread_cond_t bankAcctCondition;
18:
19: static void deposit(int amount)
20: {
21:     pthread_mutex_lock(&bankAcctMutex);
22:     bankAcctBalance += amount;
23:     printf("%d\n", bankAcctBalance);
24:     pthread_cond_signal(&bankAcctCondition);
25:     pthread_mutex_unlock(&bankAcctMutex);
26: }
27:
28: static void withdraw(int amount)
29: {
30:     pthread_mutex_lock(&bankAcctMutex);
31:     while (bankAcctBalance < amount)
32:         pthread_cond_wait(&bankAcctCondition, &bankAcctMutex);
33:     bankAcctBalance -= amount;
34:     printf("%d\n", bankAcctBalance);
35:     pthread_mutex_unlock(&bankAcctMutex);
36: }
37:
38: /*-----*/
39:
40: static void *makeDeposits()
41: {
42:     int i;
43:     for (i = 0; i < 10; i++)
44:         deposit(1);
45:     return NULL;
46: }
47:
48: /*-----*/
49:
50: static void *makeWithdrawals()
51: {
52:     int i;
53:     for (i = 0; i < 5; i++)
54:         withdraw(2);
55:     return NULL;
56: }
57:
58: /*-----*/
59:
60: int main(void)
61: {
62:     pthread_t depositThread;
63:     pthread_t withdrawThread;
64:
65:     pthread_mutex_init(&bankAcctMutex, NULL);

```

conditions.c (Page 2 of 2)

```

66:     pthread_cond_init(&bankAcctCondition, NULL);
67:
68:     pthread_create(&depositThread, NULL, makeDeposits, NULL);
69:     pthread_create(&withdrawThread, NULL, makeWithdrawals, NULL);
70:
71:     pthread_join(depositThread, NULL);
72:     pthread_join(withdrawThread, NULL);
73:
74:     pthread_mutex_destroy(&bankAcctMutex);
75:     pthread_cond_destroy(&bankAcctCondition);
76:
77:     printf("Final bankAcctBalance: %d\n", bankAcctBalance);
78:     return 0;
79: }

```