Operating Systems, Practice Session 9

Deadlock

T. Tolga Sarı (sarita@itu.edu.tr)
Sultan Çoğay (cogay@itu.edu.tr)
Doğukan Arslan (arslan.dogukan@itu.edu.tr)

April 27, 2022

Today

Operating Systems, PS 9

Deadlock

Examples

Deadlock

A contradiction arises if occurrence of an event is based on a condition whereas the condition itself is dependent on the occurrence of the same event.

- An inexperienced person can not find a job. A person can not gain experience if he/she can not find a job.
- You do not vote for a party that has a low voting percentage. A party that is not voted can not increase its voting percentage.
- For being a good team, qualified players have to be included. Qualified players are transferred to good teams.

If these kinds of contradictions exist related to processes, operating system may face with deadlock situations.

- A system has 200 Kb memory. Process A and Process B hold 80 Kb and 70 Kb memory, respectively. In order to terminate they require 70 Kb and 60 Kb more memory, respectively.

Necessary Conditions

A deadlock situation can arise if all of the following conditions hold simultaneously in a system:

1. **Mutual Exclusion:** At least one resource must be held in a non-shareable mode. Only one process can use the resource at any given instant of time.
2. **Resource Holding:** A process is currently holding at least one resource and requesting additional resources which are being held by other processes.
3. **No Preemption:** A resource can be released only voluntarily by the process holding it, after that process has completed its task.
4. **Circular Wait:** A process must be waiting for a resource which is being held by another process, which in turn is waiting for the first process to release the resource.

A Simple Deadlock Example

```
1  // mutex variable declarations
2  pthread_mutex_t lock_1;
3  pthread_mutex_t lock_2;
```

```
1
2  void* faulty_functionA(void *arg){
3    pthread_mutex_lock(&lock_1); // start of Critical Region 1
4    printf("\nA is in Critical Region 1\n");
5    fflush(stdout); // to print out buffer contents immediately
6    sleep(2); // sleep for 2 seconds
7    pthread_mutex_lock(&lock_2); // start of Critical Region 2
8    printf("\nA is in Critical Region 2\n");
9    fflush(stdout); // to print out buffer contents immediately
10   pthread_mutex_unlock(&lock_2); // end of Critical Region 2
11   pthread_mutex_unlock(&lock_1); // end of Critical Region 1
12 }
```

A Simple Deadlock Example

```
1  void* faulty_functionB(void *arg){
2    sleep(1); // sleep for 1 second
3    pthread_mutex_lock(&lock_2); // start of Critical Region 2
4    printf("\nB is in Critical Region 2\n");
5    fflush(stdout); // to print out buffer contents immediately
6    sleep(2); // sleep for 2 seconds
7    pthread_mutex_lock(&lock_1); // start of Critical Region 1
8    printf("\nB is in Critical Region 1\n");
9    fflush(stdout); // to print out buffer contents immediately
10   pthread_mutex_unlock(&lock_1); // end of Critical Region 1
11   pthread_mutex_unlock(&lock_2); // end of Critical Region 2
12 }
```

A Simple Deadlock Example

```
1  int main(){
2    pthread_t threadA, threadB; // declaring two threads
3    pthread_mutex_init(&lock_1,NULL); // initializing mutex variables
4    pthread_mutex_init(&lock_2,NULL); // initially unlocked
5    if( pthread_create(&threadA,NULL,faulty_functionA,NULL)){ // creating threadA
6      printf("Thread creation error");
7      exit(1);
8    }
9    if( pthread_create(&threadB,NULL,faulty_functionB,NULL)){ // creating threadB
10     printf("Thread creation error");
11     exit(1);
12   }
13   if( pthread_join(threadA,NULL)){ // waiting for threadA to terminate
14     printf("Thread join error");
15     exit(1);
16   }
17   if( pthread_join(threadB,NULL)){ // waiting for threadB to terminate
18     printf("Thread join error");
19     exit(1);
20   }
21   pthread_mutex_destroy(&lock_1); // destroying mutex variables
22   pthread_mutex_destroy(&lock_2);
23   return 0;
24 }
```

A Simple Deadlock Example: Output

```
A is in Critical Region 1

B is in Critical Region 2
```

A Simple Deadlock Example - Modified

```
1  void* functionA(void *arg){
2    pthread_mutex_lock(&lock_1); // start of Critical Region 1
3    printf("\nA is in Critical Region 1\n");
4    fflush(stdout); // to print out buffer contents immediately
5    sleep(5); // sleep for 5 seconds
6    while(pthread_mutex_trylock(&lock_2)){ // try to acquire lock_2
7      pthread_mutex_unlock(&lock_1); // release lock_1
8      sleep(1); // sleep for 1 second
9      printf("\nA is WAITING\n");
10     fflush(stdout); // to print out buffer contents immediately
11     pthread_mutex_lock(&lock_1); // reacquire lock_1
12   }
13   // start of Critical Region 2
14   printf("\nA is in Critical Region 2\n");
15   fflush(stdout); // to print out buffer contents immediately
16   pthread_mutex_unlock(&lock_2); // end of Critical Region 2
17   pthread_mutex_unlock(&lock_1); // end of Critical Region 1
18 }
```

A Simple Deadlock Example - Modified

```
1  void* functionB(void *arg){
2    sleep(1); // sleep for 1 second
3    pthread_mutex_lock(&lock_2); // start of Critical Region 2
4    printf("\nB is in Critical Region 2\n");
5    fflush(stdout); // to print out buffer contents immediately
6    sleep(4); // sleep for 4 seconds
7    while(pthread_mutex_trylock(&lock_1)){ // try to acquire lock_1
8      pthread_mutex_unlock(&lock_2); // release lock_2
9      sleep(1); // sleep for 1 second
10     printf("\nB is WAITING\n");
11     fflush(stdout); // to print out buffer contents immediately
12     pthread_mutex_lock(&lock_2); // reacquire lock_2
13   }
14   // start of Critical Region 1
15   printf("\nB is in Critical Region 1\n");
16   fflush(stdout); // to print out buffer contents immediately
17   pthread_mutex_unlock(&lock_1); // end of Critical Region 1
18   pthread_mutex_unlock(&lock_2); // end of Critical Region 2
19 }
```

A Simple Deadlock Example - Modified: Output

Output 1:

```
A is in Critical Region 1

B is in Critical Region 2

B is in Critical Region 1

A is WAITING

A is in Critical Region 2
```

Output 2:

```
A is in Critical Region 1

B is in Critical Region 2

A is in Critical Region 2

B is WAITING

B is in Critical Region 1
```

A More Realistic Example (Race Condition)

```
1  class Pair{ // Pair class declaration (C++)
2      int a;
3      int b;
4      pthread_mutex_t plock; // mutex variable
5    public:
6      Pair(int,int); // constructors
7      Pair(void){};
8      ~Pair(); // destructor
9      // overloaded operators for comparison
10     bool operator<(Pair &);
11     bool operator>(Pair &);
12     bool operator==(Pair &);
13     // methods for setting attributes
14     void setA(int);
15     void setB(int);
16     void setAB(int,int);
17     void print(string); // print method
18     // methods for mutex operations
19     void lock();
20     void unlock();
21 };
```

A More Realistic Example (Race Condition)

```
1  // constructor
2  Pair::Pair(int a_in,int b_in){
3    a=a_in;
4    b=b_in;
5  }
```

```
1  // set methods
2  void Pair::setA(int a_in){ a=a_in;}
3
4  void Pair::setB(int b_in){ b=b_in; }
5
6  void Pair::setAB(int a_in,int b_in){
7    a=a_in;
8    b=b_in;
9  }
10
11 // print method
12 void Pair::print(string name){
13   cout << endl << name << ": (" << a <<","<<b<<")"<<endl;
14 }
```

A More Realistic Example (Race Condition)

```
1  // overloaded operators
2  bool Pair::operator<(Pair &other){
3    if(a<other.a)
4      return true;
5    if(a==other.a && b<other.b)
6      return true;
7    return false;
8  }
9  bool Pair::operator>(Pair &other){
10   if(a>other.a)
11     return true;
12   if(a==other.a && b>other.b)
13     return true;
14   return false;
15 }
16 bool Pair::operator==(Pair &other){
17   if(a==other.a && b==other.b)
18     return true;
19   return false;
20 }
```

A More Realistic Example (Race Condition)

```
1  int main(){
2    pthread_t mythreadA; // declaring mythreadA
3    Pair* x=new Pair(1,2);
4    Pair* y=new Pair(2,3);
5    // creating a list of two Pairs (x and y)
6    Pair* pList[]={x,y};
7    // creating mythreadA
8    if( pthread_create(&mythreadA,NULL,thread_function,(void*) pList)){
9      printf("error creating thread");
10     abort();
11   }
12   sleep(1); // to have a race
13   // set attribute a of x to 5 and print x
14   x->setA(5);
15   pList[0]->print("x");
16   // wait for mythreadA to terminate
17   if( pthread_join(mythreadA,NULL)){
18     printf("error joining thread");
19     abort();
20   }
21   delete x;
22   delete y;
23   return 0;
24 }
```

A More Realistic Example (Race Condition)

```
1  // thread handling function
2  void* thread_function(void *arg){
3    Pair** pList=(Pair**) arg;
4    // print x and y
5    pList[0]->print("x");
6    pList[1]->print("y");
7    sleep(1); // to have a race
8    // compare x and y and print the result
9    if((*pList[0])>(*pList[1]))
10     cout<<endl<<"x>y"<<endl;
11   if((*pList[0])<(*pList[1]))
12     cout<<endl<<"x<y"<<endl;
13   if((*pList[0])==(*pList[1]))
14     cout<<endl<<"x=y"<<endl;
15   return NULL;
16 }
```

A More Realistic Example (Race Condition): Output

```
x: (1,2)

y: (2,3)

x<y

x: (5,2)
```

```
x: (1,2)

y: (2,3)

x: (5,2)

x>y
```

A More Realistic Example (Deadlock)

```
1  // constructor
2  Pair::Pair(int a_in,int b_in){
3    a=a_in;
4    b=b_in;
5    pthread_mutex_init(&plock,NULL);
6  }
7
8  // destructor
9  Pair::~Pair(){
10   pthread_mutex_destroy(&plock);
11 }
```

```
1  // set methods (using mutex)
2  void Pair::setA(int a_in){
3    lock();
4    a=a_in;
5    unlock();
6  }
```

setB and setAB are modified similarly to include mutex.

A More Realistic Example (Deadlock)

```
1 // mutex lock method
2 void Pair::lock(){
3   pthread_mutex_lock(&plock);
4 }
5
6 // mutex unlock method
7 void Pair::unlock(){
8   pthread_mutex_unlock(&plock);
9 }
```

A More Realistic Example (Deadlock)

```
1  bool Pair::operator<( Pair &other){
2    // acquire own lock
3    lock();
4    sleep(1); // to ensure deadlock
5    // acquire other's lock
6    other.lock();
7    if(a<other.a){
8      // release locks
9      unlock();
10     other.unlock();
11     return true;
12   }
13   if(a==other.a && b<other.b){
14     // release locks
15     unlock();
16     other.unlock();
17     return true;
18   }
19   // release locks
20   unlock();
21   other.unlock();
22   return false;
23 }
```

`operator>` and `operator==` are modified similarly to include mutex.

A More Realistic Example (Deadlock)

```
1  int main(){
2    pthread_t mythreadA,mythreadB; // declaring two threads
3    Pair* x=new Pair(1,2);
4    Pair* y=new Pair(2,3);
5    // creating two lists of Pairs (x,y) and (y,x)
6    Pair* pList[]={x,y};
7    Pair* qList[]={y,x};
8    // creating two threads
9    if( pthread_create(&mythreadA,NULL,thread_function,(void*) pList)){
10     printf("error creating thread");
11     abort();
12   }
13   if( pthread_create(&mythreadB,NULL,thread_function,(void*) qList)){
14     printf("error creating thread");
15     abort();
16   }
17   if( pthread_join(mythreadA,NULL)){ // waiting for threadA to terminate
18     printf("error joining thread");
19     abort();
20   }
21   if( pthread_join(mythreadB,NULL)){ // waiting for threadB to terminate
22     printf("error joining thread");
23     abort();
24   }
25   delete x; delete y;
26   return 0;
27 }
```

A More Realistic Example (Deadlock): Output

```
x: (2,3)
y: (1,2)
x: (1,2)
y: (2,3)
```