İşletim Sistemleri Uygulama 9

Ölümcül kilitlenme

Bilgisayar Mühendisliği

İstanbul Teknik Üniversitesi
34469 Maslak, İstanbul

April 13, 2011

Bugün

İşletim Sistemleri Uygulama 9

- Ölümcül kilitlenme
- Ölümcül kilitlenmeden korunma yolları

Catch-22

Joseph Heller'in Catch-22 adlı romanı bir çelişki üzerine kuruludur. Romanda, olayın gerçekleşmesi bir şarta bağlıdır ancak o şartın gerçekleşmesi de aynı olaya bağlıdır.

- Deneyimsiz biri iş bulamaz. İş bulamazsa deneyim kazanamaz.
- Oy oranı düşük bir partiye oy verilmez. Oy verilmeyen partinin oy oranı artmaz.
- İyi bir takım olmak için iyi futbolcular gerekir. İyi futbolcular iyi takımlara transfer olurlar.

Benzer çelişkiler süreçleri ilgilendiriyorsa, işletim sistemi ölümcül kilitlenmeyle karşılaşabilir.

Ölümcül kilitlenme şartları

Ölümcül kilitlenmenin dört şartını Coffmann 1971'de listelemiştir:

Karşılıklı dışlama koşulu Kaynaklar paylaşılamaz.

Sahiplenme ve bekleme koşulu Bir süreç elinde bulunan kaynakları bırakmadan yeni kaynaklar istemelidir.

Geri alınamaz kaynak koşulu Bir sürecin elindeki kaynak kendi isteği dışında bıraktırılamaz.

Çevrel bekleme koşulu Kaynaklar için bekleyen süreçler çizgesi en az bir çevrim barındırmalıdır.

Ölümcül kilitlenme örnekleri

Kısıtlı bellek 200K sekizli bellek alanı olan bir sistemde A süreci 80K ve B süreci 70K kullanmaktadır. A ve B süreçleri sırasıyla 60K ve 80K istemektedirler.

Davul ve baget İki çocuk aynı anda davul çalmak istiyorlar. Oyuncak torbasını birlikte karıştırıyorlar. Biri davulu diğeri bageti buluyor. Her ikisi de karşısındakinden elinde olanı vermesini istiyor.

Berber dükknı Bir berber dükknın iki odasından birinde saç kesmek için bir sandalye, diğerinde beklemek için n koltuk var. Berber, sandalyedeki müşterinin saçını kestiken sonra bekleme salonuna bakıyor. Bekleyen müşteri varsa onu sandalyeye oturtup saçını kesiyor. Bekleyen yoksa, geri dönüp sandalyede uyumaya başlıyor. Bir müşteri dükkna gidiğinde berbere bakıyor. Berber uyuyorsa uyandırıyor. Saç kesiyorsa geri dönüp bekleme salonunda bekliyor. Bekleme salonunda yer yoksa, çıkıp gidiyor. Berberi saç keserken gören müşteri henüz salona geçmemişken berber saç kesimini bitirip salona bakarsa, uyumaya gidecektir. Bu durumda berber içeride uyurken müşteri salonda bekler.

Canlı kilitlenme

Ölümcül kilitlenmenin özel bir durumu olarak görülebilir. Ancak iki kişinin geçeceği bir koridorda karşılaşan iki kişi aynı anda yana adım atarlarsa tekrar karşılaşırlar. Yeniden yana adım attıklarında yine karşı karşıya kalırlar.

Ölümcül kilitlenmeden korunma yolları

Ölümcül kilitlenmeden korunmanın dört yolundan söz edilir:

- Devekuşu yaklaşımı
- Ölümcül kilitlenmeyi belirleme ve kotarma
- Ölümcül kilitlenmeyi engelleme
- Ölümcül kilitlenmeden kaçınma

Devekuşu yaklaşımı

- Ekonomik olarak anlamlı olduğu düşünülüyorsa kullanılır.
- Ölümcül kilitlenmenin sık sık yaşanmayacağı öngörülüyorsa, bu tehlike tamamen göz ardı edilir.
- Ölümcül kilitlenmeyi yaşamanın getireceği kayıp ölümcül kilitlenmeyi önlemek için harcanacak kaynaklardan daha az olabilir.

Belirleme ve kotarma

Yaklaşımın ilk yarısı ölümcül kilitlenmenin varlığının belirlenmesidir. Ölümcül kilitlenme belirlenirse kotarmak için üç şey yapılabilir:

- Ölümcül kilitlenmeye yol açan süreçlere bir süre yeni kaynak verilmez.
- Gereken bir kaynağı boşa çıkarmak için süreçlerden birini ya da birkaçını daha önceki bir duruma geçirmek.
- Ölümcül kilitlenme ortadan kalkana kadar kilitlenmeye neden olan süreçlerden birini ya da birkaçını sonlandırmak. (Sürekli aynı süreç sonlandırılabilir.)

Her adımda bir belirleme algoritması çalıştırılması gerekir. bilinen algoritmalar $O(n^2)$ karmaşıklıktadır.

Engelleme

- Bilinen en ünlü yöntem Dijkstra'nın banker algoritmasıdır.
- Bu algoritmada güvenli durum ve tehlikeli durum tanımlanır.
- Güvenli durum, tüm süreçlerin istedikleri tüm kaynakları alabilecekleri en az bir çalışma sıralamasının olduğu durumdur.
- Tehlikeli durum, tüm süreçlerin istedikleri tüm kaynakları alabilecekleri hiç bir çalışma sıralamasının olmadığı durumdur.
- Bir süreç kaynak istediğinde, isteği karşılandıktan sonra tüm süreçlerin sonlanmasına olanak veren bir çalışma sıralaması varsa (oluşacak durum güvenliyse) kaynaklar sürece atanır.
- Algoritmanın çalışması için:
  - Süreçler azmi kaynak isteklerini önceden bildirmeli.
  - Süreç ve kaynak sayısı sabit olmalı.
  - Süreçlerin yürütülme sıraları önemsiz olmalı.
  - Süreçler kaynakları bırakmadan sonlanmamalı.
- Banker algoritması kaynakların etkin kullanımını engeller.
- Her kaynak isteğinde $O(N^2)$ karmaşıklığındaki algoritma çalıştırılmalıdır.

Kaçınma

Dört nedenin herhangi birinden kaçınmak ölümcül kilitlenmeyi önler.

Karşılıklı dışlama Bu etkiden kaçınmak olanaklı değil.

Sahiplenme ve bekleme Bir süreç çalışmaya başlamadan evvel tüm kaynakları tek seferde alır. Verimsizdir ve açlık tehlikesi vardır.

Geri alınamaz kaynak Bir sürecin kaynakları isteği dışında elinden alınabilir. Sürecin durumu yitirilebilir ve açlık tehlikesi vardır.

Çevrel bekleme Tüm kaynaklar için global bir öncelik sırası belirlenebilir ve kaynaklar bu sırada istenebilir. Tüm kaynakların adlandırılması ve sıralanması gerekecektir.

Basit bir ölümcül kilitlenme örneği

```
1   void* faulty_functionA(void *arg){

            pthread_mutex_lock(&lock_1);
            printf("\nA kritik bolge 1'de\n");
            fflush(stdout);
6
            sleep(2);
            pthread_mutex_lock(&lock_2);

            printf("\nA kritik bolge 2'de\n");
11          fflush(stdout);

            pthread_mutex_unlock(&lock_2);
            pthread_mutex_unlock(&lock_1);
    }
```

Basit bir ölümcül kilitlenme örneği

```
void* faulty_functionB(void *arg){
        sleep(1);
        pthread_mutex_lock(&lock_2);
        printf("\nB kritik bolge 2'de\n");
        fflush(stdout);

        sleep(2);
        pthread_mutex_lock(&lock_1);

        printf("\nB kritik bolge 1'de\n");
        fflush(stdout);

        pthread_mutex_unlock(&lock_1);
        pthread_mutex_unlock(&lock_2);

}
```

Basit bir ölümcül kilitlenme örneği

```
int main(){

        pthread_t threadA,threadB;

        pthread_mutex_init(&lock_1,NULL);
        pthread_mutex_init(&lock_2,NULL);

        if( pthread_create(&threadA,NULL,faulty_functionA,NULL)){
                printf("iplik yaratilirken hata");
                exit(1);
        }

        if( pthread_create(&threadB,NULL,faulty_functionB,NULL)){
                printf("iplik yaratilirken hata");
                exit(1);
        }

        if( pthread_join(threadA,NULL)){
                printf("iplik birlesirken hata");
                exit(1);
        }
        if( pthread_join(threadB,NULL)){
                printf("iplik birlesirken hata");
                exit(1);
        }

        pthread_mutex_destroy(&lock_1);
        pthread_mutex_destroy(&lock_2);

        return 0;
}
```

Basit bir ölümcül kilitlenme örneği

```
void* functionA(void *arg){

        pthread_mutex_lock(&lock_1);
        printf("\nA kritik bolge 1'de\n");
        fflush(stdout);

        sleep(5);
        while(pthread_mutex_trylock(&lock_2)){
                pthread_mutex_unlock(&lock_1);
                sleep(1);
                printf("\nA bekliyor\n");
                fflush(stdout);
                pthread_mutex_lock(&lock_1);
        }

        printf("\nA kritik bolge 2'de\n");
        fflush(stdout);

        pthread_mutex_unlock(&lock_2);
        pthread_mutex_unlock(&lock_1);
}
```

Basit bir ölümcül kilitlenme örneği

```
void* functionB(void *arg){
        sleep(1);
        pthread_mutex_lock(&lock_2);
        printf("\nB kritik bolge 2'de\n");
        fflush(stdout);

        sleep(4);
        while(pthread_mutex_trylock(&lock_1)){
                pthread_mutex_unlock(&lock_2);
                sleep(1);
                printf("\nB bekliyor\n");
                fflush(stdout);
                pthread_mutex_lock(&lock_2);
        }

        printf("\nB kritik bolge 1'de\n");
        fflush(stdout);

        pthread_mutex_unlock(&lock_1);
        pthread_mutex_unlock(&lock_2);

}
```

Daha gerçekçi bir örnek

```
#include <string.h>

using namespace std;

class Pair{

        private:

                int a;
                int b;
                pthread_mutex_t plock;

        public:
                bool operator<(Pair &);
                bool operator>(Pair &);
                bool operator==(Pair &);
                void setA(int);
                void setB(int);
                void setAB(int,int);
                Pair(int,int);
                Pair(void);
                ~Pair();
                void print(string);
                void lock();
                void unlock();
};
```

Daha gerçekçi bir örnek

```
bool Pair::operator<( Pair &other){

        if(this->a<other.a)
            return true;

        if(this->a==other.a && this->b<other.b)
                return true;

        return false;
}

bool Pair::operator>( Pair &other){

        if(this->a>other.a)
                return true;

        if(this->a==other.a && this->b>other.b)
                return true;

        return false;
}

bool Pair::operator==( Pair &other){

        if(this->a==other.a && this->b==other.b)
                return true;

        return false;
}
```

Daha gerçekçi bir örnek

```
1   Pair::Pair(int a,int b){
        this->a=a;
        this->b=b;
    }

6   Pair::Pair(){
        this->a=(rand()%10);
        this->b=(rand()%10);
    }

11  void Pair::print(string name){
        cout << endl << name << ": (" << this->a <<","<<this->b<<")"<<endl;
    }

    void Pair::setA(int a){
16      this->a=a;
    }

    void Pair::setB(int b){
        this->b=b;
21  }

    void Pair::setAB(int a,int b){
        this->a=a;
        this->b=b;
26  }
```

Daha gerçekçi bir örnek

```
int main(){

        pthread_t mythreadA,mythreadB;

        Pair* x=new Pair(1,2);
        Pair* y=new Pair(2,3);

        Pair* pList[]={x,y};

        if( pthread_create(&mythreadA,NULL,thread_function,(void*) pList)){
                printf("error creating thread");
                abort();
        }

        sleep(1);

        x->setA(5);
        pList[0]->print("x");

        if( pthread_join(mythreadA,NULL)){
                printf("error joining thread");
                abort();
        }

        return 0;
}
```

Daha gerçekçi bir örnek

```
void* thread_function(void *arg){

        Pair** pList=(Pair**) arg;

        pList[0]->print("x");
        pList[1]->print("y");

        sleep(2);

        if((*pList[0])>(*pList[1]))
                cout<<endl<<"x>y"<<endl;

        if((*pList[0])<(*pList[1]))
                cout<<endl<<"x<y"<<endl;

        if((*pList[0])==(*pList[1]))
                cout<<endl<<"x=y"<<endl;


        return NULL;
}
```

Daha gerçekçi bir örnek

```
bool Pair::operator<( Pair &other){

        this->lock();
4       other.lock();

        if(this->a<other.a){
                this->unlock();
                other.unlock();
9               return true;
        }

        if(this->a==other.a && this->b<other.b){
                this->unlock();
14              other.unlock();
                return true;
        }


19      this->unlock();
        other.unlock();

        return false;
}
```

Daha gerçekçi bir örnek

```
1   Pair::Pair(int a,int b){
        this->a=a;
        this->b=b;
        pthread_mutex_init(&plock,NULL);
    }
6
    Pair::~Pair(){
        pthread_mutex_destroy(&plock);
    }

11  void Pair::setA(int a){
        this->lock();
        this->a=a;
        this->unlock();
    }
16
    void Pair::lock(){
        pthread_mutex_lock(&plock);
    }

21  void Pair::unlock(){
        pthread_mutex_unlock(&plock);
    }
```

Daha gerçekçi bir örnek

```
int main(){

        pthread_t mythreadA,mythreadB;

        Pair* x=new Pair(1,2);
        Pair* y=new Pair(2,3);

        Pair* pList[]={x,y};
        Pair* qList[]={y,x};

        if( pthread_create(&mythreadA,NULL,thread_function,(void*) pList)){
                printf("error creating thread");
                abort();
        }

        if( pthread_create(&mythreadB,NULL,thread_function,(void*) qList)){
                printf("error creating thread");
                abort();
        }

        sleep(1);

        if( pthread_join(mythreadA,NULL)){
                printf("error joining thread");
                abort();
        }

        if( pthread_join(mythreadB,NULL)){
                printf("error joining thread");
                abort();
        }

        return 0;
}
```