readers/writers problem

Thursday, 29. September 2022 15:53

a)
```
void reader()
{
    down(&mutex);
    readcount = readcount + 1;
    if (readcount == 1) down(&writer);
    up(&mutex);
    read_shared_object(&data);
    down(&mutex);
    readcount = readcount - 1;
    up(&mutex);
    if (readcount == 0) up(&writer);
}
```

deadlock

```
void writer()
{
    down(&writer);
    write_shared_object(&data);
    up(&writer);
}
```

deadlock

problem

when a reader comes out of the second critical section (mutex) it checks a state of a shared variable, but since up(&mutex) and the if statement are not atomic together another reader might change the state of readcount and increase it to 1. So up(writer) will not execute and writer will not be unblocked, additionally the reader that changed readcount to 1 will get blocked on down(&writer) and will deadlock in the critical section.

b)
```
void reader()
{
    down(&mutex);
    readcount = readcount + 1;
    if (readcount == 1) down(&writer);
    up(&mutex);
    read_shared_object(&data);
    down(&mutex);
    readcount = readcount - 1;
    if (readcount == 0) {
        up(&mutex);
      * up(&writer);
    } else {
        up(&mutex);
    }
}
```

```
void writer()
{
    down(&writer);
    write_shared_object(&data);
    up(&writer);
}
```

the problem might occur at *, since this might not create fatal problems but it might lead to starvation of the writer. The problem occurs because again, up(&mutex) and up(&writer) are not atomic together, and things happen between them.

c)
```
void reader()
{
    down(&mutex);
    readcount = readcount + 1;
    if (readcount == 1) down(&writer);
    up(&mutex);
    read_shared_object(&data);
    down(&mutex);
    readcount = readcount - 1;
    if (readcount == 0) up(&writer);
    up(&mutex);
}
```

deadlock

```
void writer()
{
    down(&writer);
    down(&mutex);
    write_shared_object(&data);
    up(&mutex);
    up(&writer);
}
```

deadlock

the reader is correct but the problem appears when writer and reader execute at the same time. writer will down(&writer) and a reader will down(&mutex) at the same time, so a writer will get blocked before (mutex) and reader will get blocked on (writer), so a deadlock will occur.