2013 - 2014 Thema 2.1 ITSD

Practicum Week 3 Exercises

Exercises

Chapter 7 Dead locks

Exercise 1.1 There are three major methods for handling deadlock. (See Chapter 7.3.1). But actually there are four. Describe these four methods for handling deadlock.

Exercise 1.2 Consider the following snapshot of a system.

	Allocation				Max				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
P0	0	0	1	2	0	0	1	2	1	5	2	0
P1	1	0	0	0	1	7	5	0				
P2	1	3	5	4	2	3	5	6				
P3	0	6	3	2	0	6	5	2				
P4	0	0	1	4	0	6	5	6				

Answer the following questions using the banker's algorithm:
a. What is the content of the matrix Need?
b. Is the system in a safe state?
c. If a request from process Pi arrives for (0, 4, 2, 0), can the request be granted immediately?

Exercise 1.3 Java’s locking mechanism (the synchronized statement) is considered reentrant. That is, if a thread acquires the lock for an object (by invoking a synchronized method or block), i t can enter other synchronized methods or blocks for the same object. Explain how deadlock would be possible i f Java’s locking mechanism were not reentrant.

Programming Exercise

Exercise 2.1 Write a multithreaded program that implements the banker's algorithm discussed in Section 7.5.3. Create n threads that request and release resources from the bank. The banker will grant the request only if it leaves the system in a safe state. You may write this program using Bibliographical Notes 271 either Pthreads or Win32 threads. It is important that access to shared data is safe from concurrent access. Such data can be safely accessed using mutex locks, which are available in both the Pthreads and Win32 API. Coverage of mutex locks in both of these libraries is described in "'producer-consumer problem" project in Chapter 6.

See Banker-source.zip