ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

FS 2017 Prof. R. Wattenhofer

Computer Engineering II

Exercise Sheet Chapter 9

Quiz

1 Quiz (True or False)

a) Unnamed pipes cannot be shared among processes and are used for process-internal communication only.

b) A pipe can only have a single writing process and a single reading process.

c) The operation `counter++` is atomic.

d) Race conditions are difficult to debug because they appear seemingly at random.

e) Peterson's solution does not work on modern hardware because the memory accesses may be reordered to increase performance.

2 Atomic Tree

Assume we have a datastructure consisting of a binary tree we'd like to access concurrently. The tree is stored in shared memory and any time any number of processes may be reading or writing to it. Reading processes do not modify the tree, while writing processes may insert a new node or remove a node in the tree.

a) Give an example of an execution in which two writing processes may result conflict. Hint: is there a way one process may overwrite the changes of the other?

b) Describe how a mutex can be used to ensure that operations do not corrupt the tree.

c) It might be desirable to group a number of operations of a writing process, so that either all of them succeed and are atomically applied or they are all discarded. This is a transactional model of operation: a transaction either atomically succeeds or it fails and may be attempted again. Does your single mutex idea still work in this scenario? Do reading processes also need to acquire the lock to avoid reading incomplete transactions?

Acquiring and releasing the mutex for every operation is expensive and limits the concurrency. A friend proposes the following scheme: at the beginning of a transaction a process copies the root of the tree to a new location, then performs the changes copying nodes that have been modified, along with their ancestors. This results in a partial copy of the tree on which the transaction is performed. Meanwhile the reading processes may concurrently access the original copy of the tree, always starting from the root. Once the transaction is ready to be committed the process performs a CAS-operation on the root of the tree, i.e., swapping the old tree with the new tree. If the CAS-operation fails the transaction is considered to have failed and will be restarted, if it succeeds the transaction is committed and the new tree is the active copy.

d) Does this correctly implement mutual exclusion?

e) What are the advantages over locking the entire tree? What are the disadvantages and how would you address them?