Alexandros Karakasidis

Assignment 3

Exercise 9:

Such a server would operate much more like the UNIX inetd deamon server, except for being a multithread server, instead of using fork and processes:

- The server's main thread will continuously listen to specific ports, which are application specific.
- Each time a socket connection is initiated from a client the server creates a new thread, assigning the requested operations to this thread.
- When the operations are completed, this thread dies.

Exercise 12:

Some design issues for an object adapter to support persistent objects are the following:

- Object adapters should be generic, thus objects must be handled more or less in the same manner, no matter being persistent or transient.
- A mechanism has to be implemented, in order to maintain state data of the persistent objects. The aim of such a mechanism, would be not only to save the state of the objects, after the server termination, but to keep incremental state saves in order to achieve fault tolerance.
- One other important issue has to do with the way the adapter handles multiple requests. In specific, a policy has to be decided, when these object access shared resources simultaneously.
- A security policy has to be specified, in order to define what acces rights each client might have

Exercise 17:

Strong Mobility in UNIX could be implemented like this:

- A parent wishing to fork, will create a child.
- This child can start its execution at the same machine with the parent.
- Either when the system gets overloaded, or in an arbitrary moment, the child should probably continue execution elsewhere. Then the execution segment is transferred to the target machine, and execution continuous.

A major problem with strong mobility is the manipulation of the resource segment. Some resources may not be accessible by the machine, to which the child has been sent, either because are not transferable (e.g. large files) or because there is no permission for access (e.g. printers). In such a case, when a resource is needed and is transferable (i.e. small files) it can be transferred to the child's machine and execution may continue. When needed resources cannot be transferred, then the process will have to migrate back to the originating machine. Thus, problems will occur with strong mobility when the migrating process will have to access not transferable resources, since the overhead of communication.