
Distributed Systems

Winter Term 2024/25

Roland Wismüller
Universität Siegen
roland.wismueller@uni-siegen.de
Tel.: 0271/740-4050, Büro: H-B 8404

Stand: October 10, 2024

Distributed Systems

Winter Term 2024/25

4 Name Services



Content

- Basics
- Example: JNDI

Literature

- Tanenbaum, van Steen: Ch. 4.1
- Farley, Crawford, Flanagan: Ch. 7
- <http://docs.oracle.com/javase/tutorial/jndi/overview>

4.1 Basics



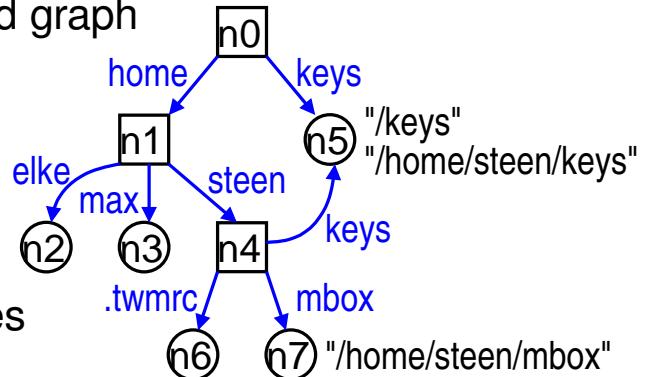
Names, Addresses and IDs

- **Name**: character or bit sequence that refers to a unit
 - ↳ unit: e.g. computer, printer, file, user, website, ...
- **Address**: name of the entry point of a unit
 - ↳ entry point allows access to the unit
 - ↳ several entry points per unit are possible
 - ↳ entry point may change over time
- A **position-independent name** identifies a unit independently from its entry point
- **ID**: name with the following properties:
 - ↳ ID refers to at most one unit, unit has at most one ID
 - ↳ ID always refers to the same unit (not reused)

4.1 Basics ...

Namespaces

- represented by directed, labelled graph
 - ↳ leaf node: named unit, with information / status if required
 - ↳ inner node: directory node
 - ↳ edges are labeled with names
- Units are named by paths in the graph:
 - Start node: < Label-1, Label-2, ... >*
 - ↳ absolute path: starting from root (of namespace)
 - ↳ relative path: starting from any node
- Example: names in the UNIX file system



4.1 Basics ...

Aliasing and Linking

- **Alias:** alternative name for the same unit
- Possibilities for the realization of aliases:
 - ↳ allow several absolute pathnames for one unit
 - ↳ e.g. *hard link* in Unix
 - ↳ a (special) leaf node stores pathname of the unit
 - ↳ e.g. *symbolic link* in Unix
- Transparent linking of different namespaces:
 - ↳ a (special) directory node stores the ID of a directory node in another namespace
 - ↳ e.g. *mounted* file system in Unix



Name Resolution

- Finding the node (or information) that corresponds to a name
 - ↳ start at the start node
 - ↳ look up first label in directory table
⇒ ID of the next node
 - ↳ etc., until the path is completely processed
- **Conclusion mechanism:** determination of the start node
 - ↳ usually implicit
- **Global names:** resolution independent of specific context
- **Local names:** resolution is context-dependent
 - ↳ e.g. pathname relative to working directory in Unix



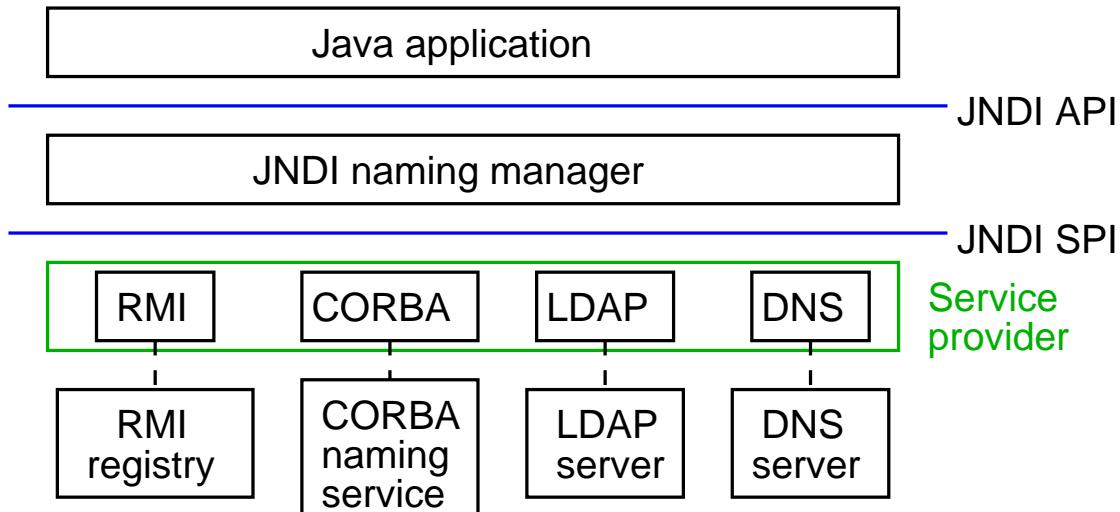
Implementation of Naming Services

- Typical operations:
 - ↳ bind(name, address, attributes)
 - ↳ lookup(name, attributes) ⇒ address, attributes
 - ↳ unbind(name, address)
- In distributed systems:
 - ↳ namespace is stored distributed (usually hierarchically)
 - ↳ for high availability: additionally replicated storage
- Name resolution can be iterative or recursive
 - ↳ iterative: Server responds with address of next server
 - ↳ recursive: server requests even at next server
- Example: *Domain Name Service* (**RN_I, 9.3**)

4.2 Example: JNDI



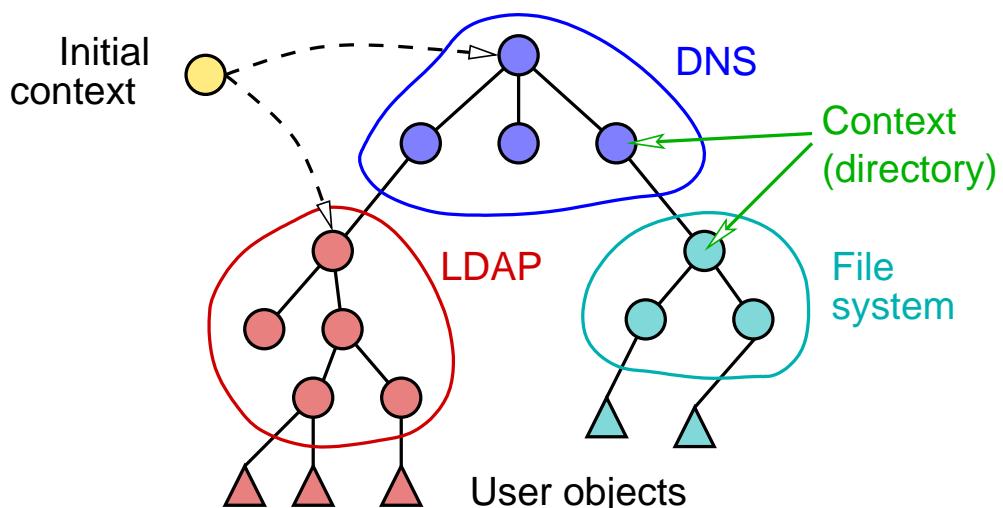
- JNDI: *Java Naming and Directory Interface*
- API for access to different name and directory services
 - directory service also stores attributes of objects



4.2 Example: JNDI ...



- JNDI supports compound namespaces
 - managed by various name or directory services



- Directories are called “contexts”
 - objects are bound to names within a context



The Interface `javax.naming.Context` for Naming Contexts

→ Important methods:

- ↳ `bind()`, `rebind()` : bind objects to names
 - ↳ `bind()` throws exception if name already exists
- ↳ `unbind()` : remove names
- ↳ `rename()` : rename
- ↳ `lookup()` : resolve name to object
- ↳ `listBindings()` : list of all bindings
- ↳ `createSubcontext()` : create sub-context
- ↳ `destroySubcontext()` : delete sub-context



The Interface `javax.naming.Context` for Naming Contexts ...

→ Implementation class `InitialContext`

- ↳ for initial context (depending on the concrete name service)
 - ↳ `Context iC = new InitialContext(properties);`
- ↳ configuration via `Properties` object (`Hashtable`), among others:
 - ↳ "java.naming.factory.initial"
 - ↳ factory for `InitialContext`
 - ↳ "java.naming.provider.url"
 - ↳ contact information for service provider
 - ↳ "java.naming.security.principal" and "java.naming.security.credentials"
 - ↳ user name and password for authentication

4.2 Example: JNDI ...



Example: Accessing the RMI Registry

```
import javax.naming.*;  
...  
  
Properties props = new Properties();  
props.put("java.naming.factory.initial",  
         "com.sun.jndi.rmi.registry.RegistryContextFactory");  
props.put("java.naming.provider.url",  
         "rmi://localhost:1099");  
Context ctx = new InitialContext(props);  
  
obj = (Hello)ctx.lookup("Hello-Server");  
  
message = obj.sayHello();
```

4.2 Example: JNDI ...



Example: Accessing a Local File System

```
import javax.naming.*;  
...  
  
Properties props = new Properties();  
props.put("java.naming.factory.initial",  
         "com.sun.jndi.fscontext.RefFSContextFactory");  
Context ctx = new InitialContext(props);  
  
for (int i=0; i<args.length-1; i++)  
    ctx = (Context)ctx.lookup(args[i]);  
NamingEnumeration<Binding> list  
    = ctx.listBindings(args[args.length-1]);  
while (list.hasMore()) {  
    Binding b = list.next();  
    System.out.println(b.getName()+" : "+b.getClassName());  
}
```