

# Major Telecomm Company

## Training and Education Center

---

### System-X™ — Testing DCE Client/Server Applications Unit 1: System-X™ DCE Functionality

*NOTE: This sample is an integrated instructor guide and participant guide in a single file. The participants see everything except the large blue type such as the type you are now reading; these are the notes and directives to the instructor. The document is printed double-sided.*

*Leader Guide*





## **Table of Contents**

<b>Subject</b>	<b>Page</b>
UNIT OVERVIEW .....	5
UNIT OBJECTIVES .....	5
OVERVIEW OF THE SYSTEM-X™ SYSTEM AND DCE.....	6
THE EMULATED CLIENT.....	10
EMULATED SERVER .....	13
LESSON REVIEW .....	17
EXERCISE 1: REVIEW OF CONCEPTS AND TERMS.....	18



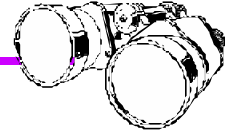
## Overview

(This is an advance-organizer to set up the students for learning.

Go over lightly; do not end up teaching the Unit from this slide.)

### **Unit One Overview**

---



- ◆ System-X / DCE architecture
- ◆ Architecture for
  - Testing the client
  - Testing the server
- ◆ Emulated client
  - Where it comes from
  - Major script operations
- ◆ Emulated server
  - Where it comes from
  - Major script operations
  - Controlling the emulated server

10/20/96

Slide 11

This Unit provides the conceptual framework for using the SYSTEM-X / DCE component. The rest of the course will be practical instruction in using the DCE component.

## Objectives

(This is an advance-organizer to set up the students for learning.

### **Unit One Objectives**

---

By the end of this lesson you will be able to:

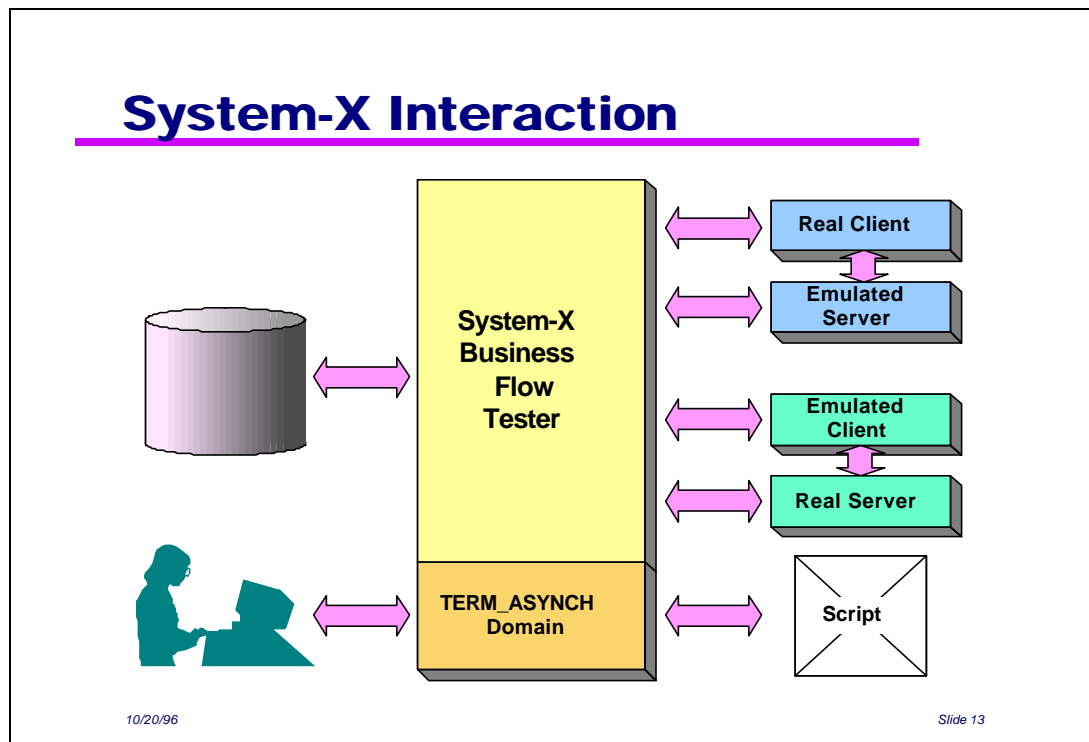
- ◆ List and describe the components of the System-X™ System when set up for DCE testing
- ◆ Describe fundamental DCE architecture
- ◆ Describe how the client and the server are tested
- ◆ Define “emulated client” and “emulated server”
- ◆ Describe how to generate the emulated client and emulated server
- ◆ List and describe major test script operations
- ◆ Interpret emulated server messages

10/20/96

Slide 12

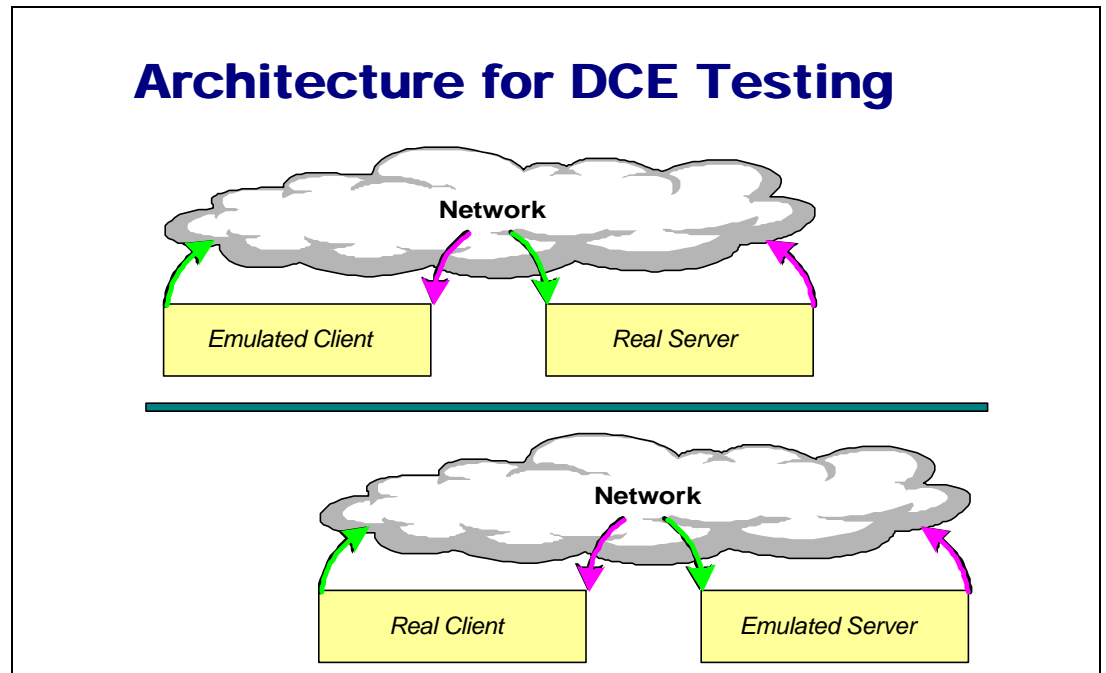


## OVERVIEW OF SYSTEM-X<sup>Ô</sup> AND DCE



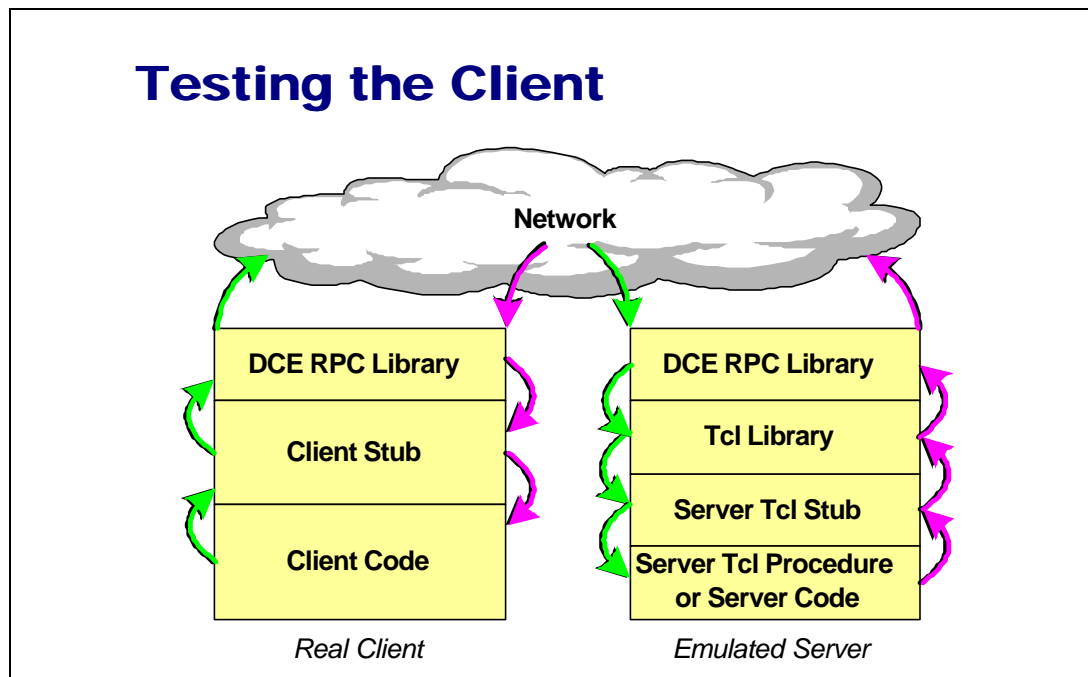
- The System-X™ System Enables you to interact with it from a terminal to
  - Write scripts
  - Control the system itself
  - Establish connections to the server, emulated server, client, and emulated client
  - Run Tcl scripts to test one of these application components
  - Capture the outputs (intended and error messages) from the tested component
  - Compare the outputs to ideal outputs (with field blanking to ignore anticipated differences)
- At the present time The DCE component is a stand-alone executable.
- You use the TermAsynch component for interacting with the stand-alone.
  - Writing scripts
  - Extracting output from the stand-alone

## OVERVIEW OF THE SYSTEM-X<sup>®</sup> SYSTEM AND DCE



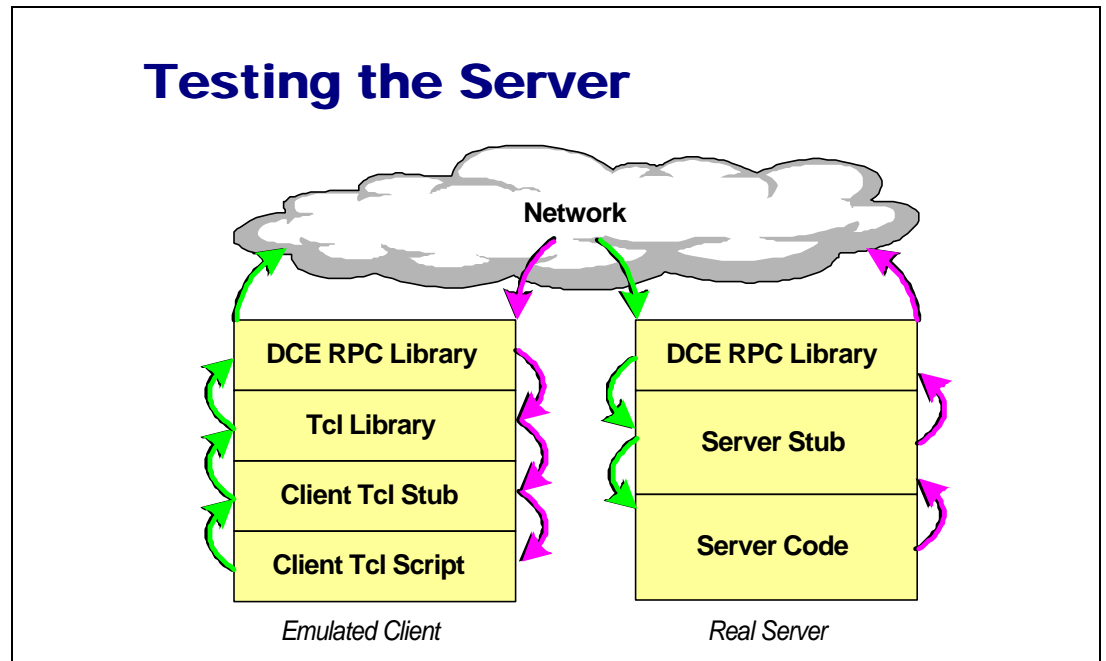
- The Architecture for DCE Testing with SYSTEM-X consists of:
  - Server (real or emulated)
  - Client (emulated or real)
  - Connected by a network.
- If you want to test a server:
  - You use an emulated client and the real server.
- If you want to test a client:
  - You use the real client and an emulated server.

## OVERVIEW OF THE SYSTEM-XÔ SYSTEM AND DCE



- Shown here is a real client and an emulated server. The job of the emulated server is to receive an RPC and send the appropriate output back to the client.
  - The output is “faked” by a Tcl script.
- Here’s how it works.
  - The client code makes a remote procedure call (RPC).
  - The client stub catches the RPC and calls the RPC library .
  - The client’s DCE RPC library sends the call over the network to the server.
  - The server DCE RPC library catches the RPC.
  - The server Tcl library translates the server procedure call into Tcl.
  - The server Tcl stub catches the Tcl procedure call.
  - And executes a Tcl script that returns “faked”  
output

## OVERVIEW OF THE SYSTEM-XÔ SYSTEM AND DCE



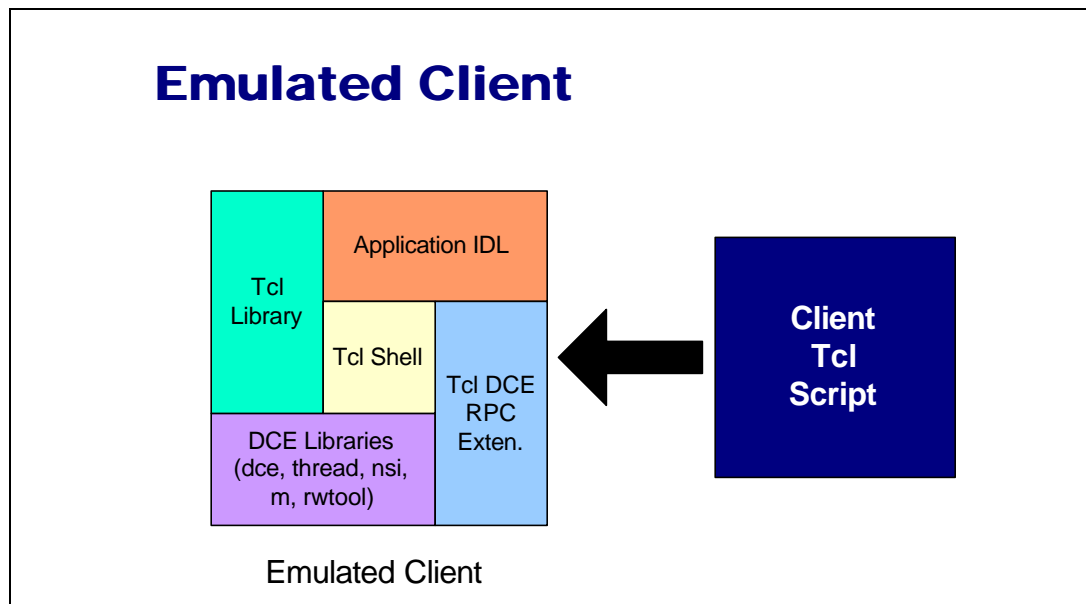
- Shown here is an emulated client and a real server. This is the setup to test the server.
- The job of the emulated client is to execute a Tcl script that generates an RPC and then catches and manipulates the RPC return. Here's how it works.
  - The client script makes a remote procedure call (RPC).
  - The client Tcl stub catches the RPC and calls the Tcl library .
  - The client Tcl library turns the Tcl code into real code and fetches the data value from C-space, and passes them to the DCE RPC library.
  - The client's DCE RPC library sends the call over the network to the server.
  - The server DCE RPC library catches the RPC.
  - The server stub catches the procedure call.
  - And executes some code that returns some output. The output goes back through the reverse





## THE EMULATED CLIENT

- The emulated client acts as a Tcl interpreter.
- And emulates the RPC part of the client software.

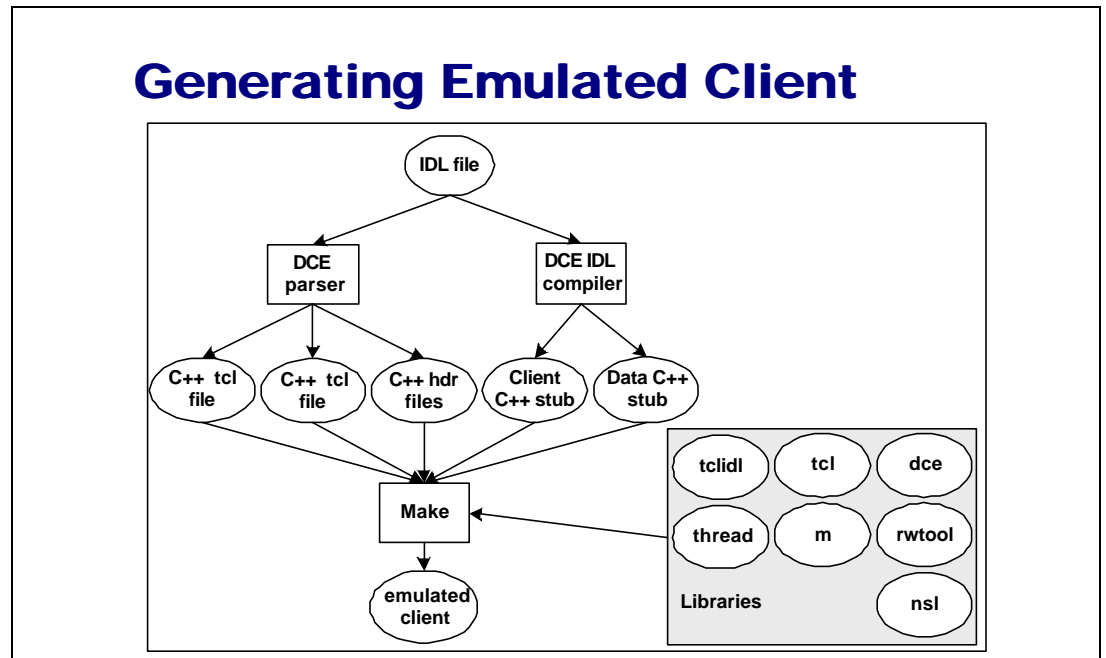


- It consists of:
  - Tcl shell
  - Tcl library
  - Application IDL (a.k.a. interface)
    - This defines the RPCs for the application
    - The IDL also has a corresponding Auxiliary Control File (ACF file)
  - Tcl extensions for DCE RPC
  - Other libraries
    - dce = main dce library
    - thread = library to provide thread services
    - nsi = Name Server Independent database
    - m = math library
    - rwtol = RogueWave library.
- The emulated client reads a script or standard input.
- For each RPC defined in the application IDL, the client contains
  - a Tcl command of the same name.
    - The interpreter invokes the RPC by calling this Tcl command with the appropriate arguments.
- For each data type defined in the IDL, the client contains
  - a Tcl command for creating a new instance of that type
    - the “constructor”
  - The constructor returns a handle,



## THE EMULATED CLIENT

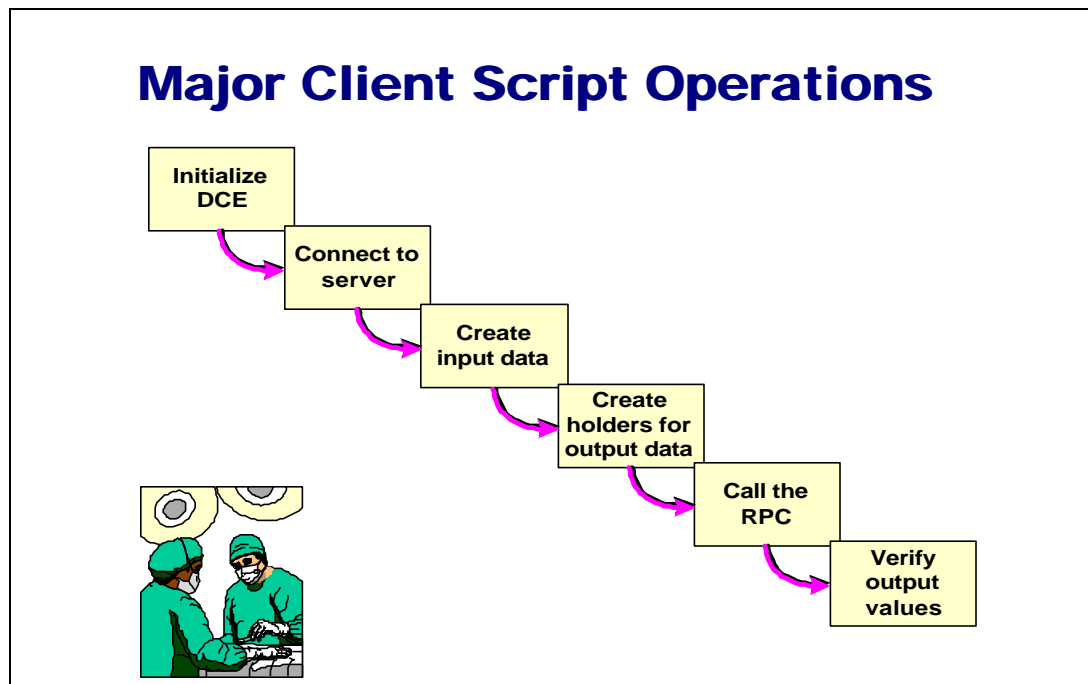
Note: there is also an auxiliary control file (.acf file) of the same name that contains code defining the System-X System™ interface



- The IDL compiler parses the IDL file and creates C stub files
- These files are compiled and linked with support libraries
  - To create the emulated client.
- The name of the generated files
  - By default, is derived from the name of the interface defined in the IDL file.
  - But may be specified with the “-n name” option to parser.
- Generating the emulated client is done by
  - Tester
  - Test automation administrator.
- After the “make”
  - You can edit the Tcl file
    - To add functionality
    - To break it into separate Tcl files
      - >E.g., one for each IDL function.

If the IDL changes

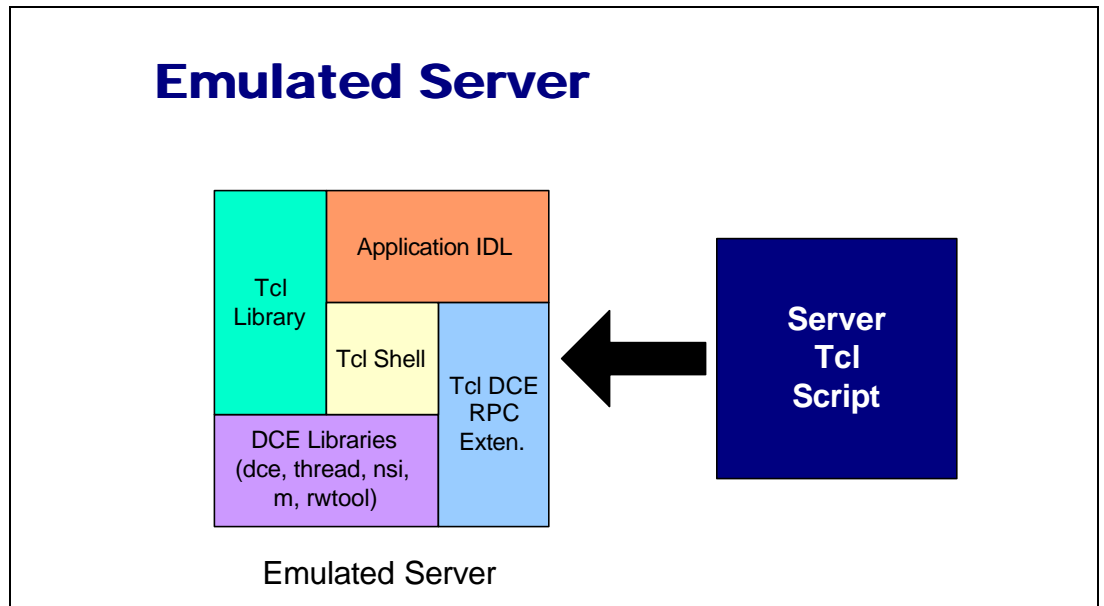
## THE EMULATED CLIENT



- The slide shows the steps that the emulated client script must perform. The emulated client stub that is created by the “make” takes care of
  - initializing DCE
  - binding to the server
  - and calling the RPC.
- It is up to the tester to
  - create the data
  - and put it into the placeholders in the stub,
  - and later to format and perhaps save, compare, or otherwise analyze the output received from the server and from DCE.
- More on these when we get into script writing

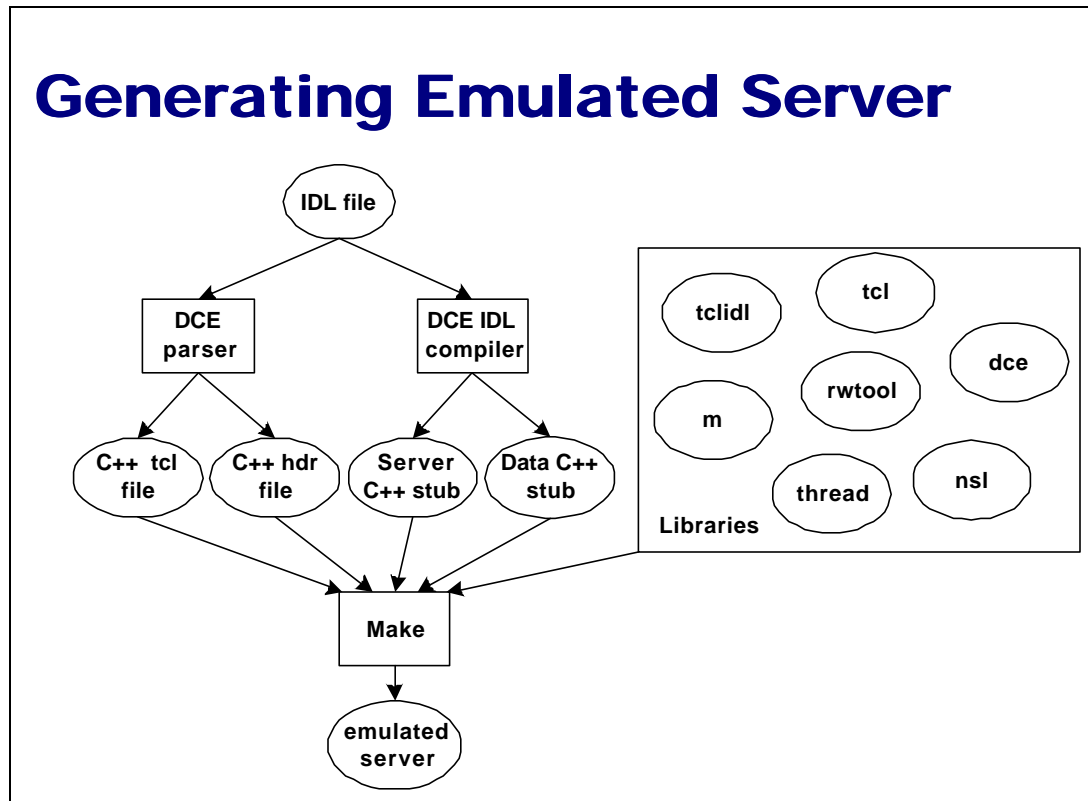


## EMULATED SERVER



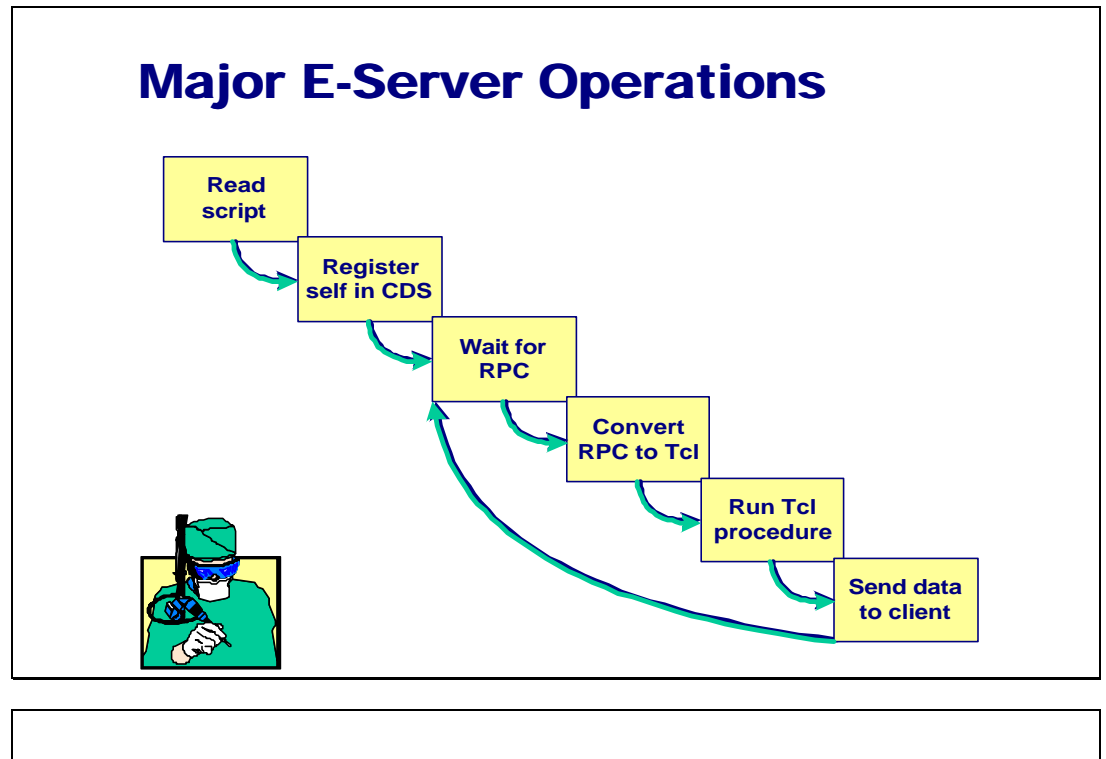
- The emulated server is a DCE server application.
- The components are similar to those of the emulated client.
- The emulated server reads a Tcl script .
  - That provides Tcl code to generate output for each procedure in the IDL.
  - The script name is given on the command line .
  - You can cause it to re-load the script .
    - Send it SIG\_USR1.
- The only function of the emulated server is to return responses to the client.
- Results can include
  - DCE error messages (not from the emulated server)
  - Server error messages
  - Output from the Tcl procedure:  
Text for diagnostic purposes

## EMULATED SERVER



- The same programs and libraries go into making the emulated server.
  - As for the emulated client.
- Again, generating the emulated server is done by
  - Tester
  - Test automation administrator.
- After the “make”
  - You can edit the Tcl file
    - To add code to the procedures to emulate server processing of RPCs.
- Of course, if the IDL changes,
  - You have to do the whole process over again

## EMULATED SERVER



- The functions of the emulated server are:
  - Register itself in the Cell Directory Service (CDS)
    - Server entry
    - Server group entry
  - Wait for RPCs.
  - Interpret an RPC to call a Tcl procedure.
  - Run the Tcl procedure.
    - You have to put Tcl code here to generate the desired output.
  - Return the results to the client via the network.
  - Return to waiting for another RPC.
- More on this in the I Unit on script writing

## EMULATED SERVER

### E-Server Messages

Intent	Signal to Use
Shut down	SIG_HUP, SIG_TERM, SIG_INT
Abort	SIG_KILL
Reload script	SIG_USR1




- Kinds of messages to send emulated server:
  - Shut down
    - SIG\_HUP | SIG\_TERM | SIG\_INT
  - Abort
    - SIG\_KILL
  - Reload server proc file
    - SIG\_USR1
    - If you make a change to the Tcl script for the server and want to reload the server without shutting it down
  - Lock/unlock server
    - For servers that span multiple test scripts.
    - Exclusive use is not supported.
- Send messages to server with standard UNIX “kill” command

```
kill [-signal] process_id
```



- SYSTEM-X / DCE architecture
  - Emulated client or server
  - Real server or client

## Unit One Summary



- ◆ System-X / DCE architecture
- ◆ Architecture for
  - Testing the client
  - Testing the server
- ◆ Emulated client
  - Where it comes from
  - Major script operations
- ◆ Emulated server
  - Where it comes from
  - Major script operations
  - Controlling the emulated server

10/20/96 Slide 24



- Emulated client
  - What it is:
    - Tcl interpreter with extensions for DCE.
  - What it contains:
    - Tcl shell, stubs, DCE libraries, Tcl libraries.
  - Where it comes from:
    - You generate it using *xmyDceParser* and *make*.
  - What it does:
    - Interprets Tcl scripts that send RPCs and receive results.
- Emulated server
  - What it is:
    - DCE server application that interprets Tcl script.
  - What it contains:
    - Same as emulated client.
  - Where it comes from:
    - You generate it using *parser* and *make*.
  - What it does:



## Exercise

### EXERCISE 1: REVIEW OF CONCEPTS AND TERMS

Provide a definition for each term below.

- Some of  
These terms  
are from this  
lesson, some  
are from the  
prerequisites

Term	Definition
ACL file	Auxiliary Control File for the IDL
bind	Add listening port information to the server's name, group name, and IP address.
CDS	Cell Directory Service: holds registration information for servers in the cell.
client	Application that can send RPCs to a server as if they were internal function calls.
DCE	Distributed Computing Environment: a computer architecture in which part of the functionality is located in a central server application and the other portions (typically controlled by users or remote equipment) are located in client applications.
DCE library	Code that handles sending and receiving RPCs over the network to the bound server and calling client.

(Allow about  
15 minutes,  
then go over  
answers)

(IG version  
has answers  
as shown at  
the right)

**EXERCISE 1: REVIEW OF CONCEPTS AND TERMS**

emulated client	A System-X/DCE program that takes input and produces output similarly to the client application but actually “fakes” it with Tcl procedures.
emulated server	Same as for emulated client but for the server.
IDL	Interface Definition Language: a structure that defines the interface between the client and the server. All variables and processes are identified in the IDL.
kill	The UNIX command that can send a message (signal) to the server to control its
network	A series of electrical connections between clients and servers that enables one to communicate with the other. The details of request, acknowledgment, and transmission are not usually important to users of the network because they are handled by the
port	The address of the input buffer at which outside communications are received.
RPC	Remote procedure call: a function call to a program (server) that is not in the application but seems to be to the
script	A set of Tcl instructions to control the emulated server or emulated client.

**EXERCISE 1: REVIEW OF CONCEPTS AND TERMS**

server	A computer process that takes RPCs and from remote callers and returns the requested output (as strings only).
server listening	The state of the server when it is listening to the port for RPCs.
SIG_HUP	A message (signal) that can be sent to a running application that causes it to
Tcl	The programming language used in System-