



hp services

education

course description

TACL programming U4199S

course overview

Master the art of writing functions in the Tandem Advanced Command Language (TACL) program in this 5-day course. Through student projects and hands-on labs, you will gain valuable experience with TACL programming. After completing this course, you will be able to write macros and routines, perform file I/O, use structured data, and write server functions.

audience

- System programmers
- System and network managers
- Application designers
- Application programmers
- System analysts
- Data communications programmers and analysts

benefits to you

- Segment files
- Define process
- MACRO and ROUTINE functions
- Variable editing
- Server functions
- Exception handling
- Debugging

pre-requisites

- Concepts and Facilities course
- Knowledge of at least one other programming language
- At least six months of programming experience

to order

You can order this course online at <http://education.hp.com>. At the site, select a country, then choose "registration" or "Book a course" and fill out the online registration form.

why hp education?

- Experienced and best-in-the-field HP instructors
- Comprehensive student materials
- State-of-the-art classroom facilities
- Hands-on practice
- Focus on job-specific skills
- More than 120 locations worldwide
- Customized on-site delivery
- Online instructor-led and self-paced training at <http://itresourcecenter.hp.com>

detailed course outline: TACL programming (U4199S)

module	key topics
overview of TACL features	<ul style="list-style-type: none">• Productivity aids provided by TACL: HISTORY, FC, ?, ! HELP facility• Function key, custom prompts, file name templates, and macro files• TACL features as a programming language
TACL variables	<ul style="list-style-type: none">• Obtaining information about variables using either commands or built-in functions• Using commands or built-in functions to create, initialize, modify, and eliminate variables• Concept of a "frame" and how it relates to managing variables• Variable stacks and their levels: what they are and how to create, reference, and eliminate them• Syntax rules for writing TACL functions• Lab Exercise (20 minutes): Learn and understand how to logon and use TACL function keys
directories and segments	<ul style="list-style-type: none">• Creating a segment file containing a library function• Using the existing segment file by attaching it to a directory• Getting information on the segment file• Syntax rules for writing TACL functions• Lab Exercise (30 minutes): Learn to create and use a segment file
editing variables	<ul style="list-style-type: none">• Performing variable file I/O• Performing global editing of a variable• Performing line editing of a variable• Performing character editing of a variable• Locating the position of a string in a variable• Extracting lines and characters from a variable
writing functions-macros	<ul style="list-style-type: none">• Syntax required to write macro functions• TACL's handling of arguments to macro functions• TACL's expansion of macro functions• Writing macro functions
writing functions - #IF statements	<ul style="list-style-type: none">• Write functions that use the TACL #IF THEN ELSE construct• Lab Exercise (1 hour)• Describe the syntax required to write functions in general and macro type functions in particular• Describe the different forms of the "control" built-in #IF and contrast when to use one form or the other (#IF or #IF NOT)• Write a macro type function that accepts one or more arguments and ensures that the arguments are correct by making use of the "control" built-in #IF
writing functions - #LOOP statements	<ul style="list-style-type: none">• Write functions that use the TACL #LOOP DO UNTIL construct• Write functions that use the TACL #LOOP WHILE DO construct• Lab Exercise (1 hour)• Describe the syntax required to write general functions, with particular focus on macro type functions• Describe the two forms of the "control" built-in #LOOP and determine when to use #LOOP DO UNTIL or #LOOP WHILE DO • Write a macro type function that outputs all of the volume names on the system
writing functions - #CASE statements	<ul style="list-style-type: none">• Writing functions that use the TACL #CASE construct
writing functions - debugging	<ul style="list-style-type: none">• Using the TACL debugging facility provided by TACL to aid in getting functions to work• Lab Exercise (2 hours)• Start and stop the Debugger• Set and clear breakpoints• Display and modify the contents of a variable• Single step through your function and resume execution of your function• Describe the syntax for #IF, #LOOP, and #CASE constructs• Write a function that employs the #CASE built-in
writing functions – file I/O	<ul style="list-style-type: none">• How TACL is able to do device independent I/O• Using #REQUESTER and #WAIT to perform either "waited" or "no-waited" I/O to files and devices
writing functions - routines	<ul style="list-style-type: none">• Writing "Routine" type functions and use #ARGUMENT, #MORE, and #REST• Lab Exercise (3 hours)• Modify and write routine functions• Describe the syntax and usage of #ARGUMENT and #MORE• Describe additional capabilities that routines offer that macros do not

detailed course outline: TACL programming (U4199S)

- Describe the use of the built-ins: #MYSYSTEM, #PROCESSORSTATUS, and #PROCESSORTYPE, #LOOP, and #CASE
- using structures
- Using a STRUCT to access data
- inline processing
- Performing process I/O using the INLINE facility
 - Controlling the display of the process output
 - Logging the process output to a variable debugger
 - Lab Exercise (30 minutes)
 - Describe the syntax required to write INLINE functions in general
 - Use the INLINE facility for interfacing with the PERUSE utility
 - Practice coding techniques using the variable editing built-ins Review the usage of #INPUTV, #LOOP, and #IF
 - Describe the use of #INLINEPREFIX, INLPREFIX, #INLINETO, and INLTO
 - Write a macro-type function that purges jobs from the spooler and prompts the user for permission to purge each job
- writing functions – server files
- How the server file facility provides for communication between a TACL function and a process it has activated
 - Situations in which it is appropriate to use implicit server files
 - Writing functions that use implicit server files
 - Lab Exercise (45 minutes)
 - Describe the syntax and usage of functions that employ implicit servers
 - Describe the usage of the RUN-options:
 - INV <var> DYNAMIC PROMPT <var>
 - OUTV <var>, and STATUS <var>
 - Describe the usage of the following built-ins:
 - #APPEND, #APPENDV
 - #EXTRACT, #EXTRACTV
 - #WAIT
 - #REQUESTER
 - Describe the conditions under which to use implicit servers
 - Write functions that make use of implicit servers
- define process
- Define Process facility
 - Using the Define Process variables to start, stop, and manage processes
 - Specifying where complete information on the Define Process facility can be found
- writing functions – exception handling
- Three types of exceptions that TACL allows a function to handle in its own way
 - Using the built-in functions #ERRORTEXT, #EXCEPTION, #FILTER, #RAISE, #RESET, and #RETURN
 - Structure and the organization of a function that contains "exception handling" code
 - Writing functions that contain their own "exception handling" code
- using DEFINES
- Four types of DEFINE classes
 - Their usage and comparing them to ASSIGNS
 - Using the DEFINE command within TACL to create a DEFINE, delete a DEFINE, and alter a DEFINE

for more information

For more information on HP Education Services, contact any of our worldwide offices or visit our worldwide web site on the internet at <http://education.hp.com>

Technical information in this document is subject to change without notice.

Microsoft®, Windows®, MS Windows®, and Windows NT® are U.S. registered trademarks of Microsoft Corporation. UNIX® is a registered trademark of the Open Group.

©Copyright Hewlett-Packard Company 2000. All Rights Reserved. Reproduction, adaptation, or translation without prior written permission is prohibited except as allowed under the copyright laws.

9/02 U4199S

