



TACL Programming U4199S

Master the art of writing functions in the Tandem Advanced Command Language (TACL) program in this 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.

HPE course number	U4199S
Course length	5 days
Delivery mode	ILT, VILT
View schedule, local pricing, and register	View now
View related courses	View now

Why HPE Education Services?

- IDC MarketScape leader 5 years running for IT education and training*
- Recognized by IDC for leading with global coverage, unmatched technical expertise, and targeted education consulting services*
- Key partnerships with industry leaders OpenStack®, VMware®, Linux®, Microsoft®, ITIL, PMI, CSA, and SUSE
- Complete continuum of training delivery options—self-paced eLearning, custom education consulting, traditional classroom, video on-demand instruction, live virtual instructor-led with hands-on lab, dedicated onsite training
- Simplified purchase option with HPE Training Credits

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

* Realize Technology Value with Training, IDC Infographic 2037, Sponsored by HPE, October 2017

Detailed course outline

Module 1: 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
Module 2: 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
Module 3: 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
Module 4: 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
Module 5: 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
Module 6: 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
Module 7: 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
Module 8: Writing functions: #CASE statements	<ul style="list-style-type: none"> • Writing functions that use the TACL #CASE construct

Module 9: Writing functions—debugging

- 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

Module 10: Writing functions—file I/O

- 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

Module 11: Writing functions—routines

- 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
- Describe the use of the built-ins: #MYSYSTEM, #PROCESSORSTATUS, and #PROCESSORTYPE, #LOOP, and #CASE

Module 12: Using structures

- Using a STRUCT to access data

Module 13: 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 and review the usage of #INPUTV, #LOOP, and #IF
- Describe the use of #INLINEPREFIX, INLPREFIX, #INLINETO, and INL TO
- Write a macro-type function that purges jobs from the spooler and prompts the user for permission to purge each job

Module 14: 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

Course data sheet

Module 15: 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

Module 16: 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

Module 17: 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
-

Learn more at
hpe.com/ww/learnnonstop

Follow us:



© Copyright 2019 Hewlett Packard Enterprise Development LP. The information contained herein is subject to change without notice. The only warranties for Hewlett Packard Enterprise products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Hewlett Packard Enterprise shall not be liable for technical or editorial errors or omissions contained herein.

Microsoft is either a registered trademark or trademark of Microsoft Corporation in the United States and/or other countries. The OpenStack Word Mark is either a registered trademark/service mark or trademark/service mark of the OpenStack Foundation, in the United States and other countries and is used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation or the OpenStack community. Pivotal and Cloud Foundry are trademarks and/or registered trademarks of Pivotal Software, Inc. in the United States and/or other countries. Linux is the registered trademark of Linus Torvalds in the U.S. and other countries. VMware is a registered trademark or trademark of VMware, Inc. in the United States and/or other jurisdictions.

U4199S J.01, September 2019