# ADVANCED KORN SHELL PROGRAMMING

**General:**

This course provides experienced users of the UNIX® System V Korn Shell with further insights into its effective and efficient use. The student is prepared to design, code, test debug, and execute structured Shell procedures making full use of the basic, as well as the more advanced features of the Shell.

**Objectives:**

Upon successful completion of this course, the student will be able to:

- Describe the Shell interpreter and how it makes use of the UNIX system calls to execute a procedure.
- Describe the order of evaluation of a command line and make use of the evaluation sequence in writing effective Shell procedures.
- Use the standard Shell predefined variables, as well as user defined positional and keyword parameters.
- Use all of the input/output redirection capabilities of the Shell including redirection involving file descriptors 0 to 9.
- Describe and use the Shell invocation options and all of the built-in Shell statements.
- Write structured, readable, and maintainable Shell procedures that operate in a *UNIX like* manner and use the standard UNIX facilities.
- Write reliable Shell procedures that properly handle execution time external events and debug these procedures when necessary.
- Write more efficient Shell procedures.

**Audience:**

Technical Users, Applications Programmers, and Systems Programmers requiring an in-depth understanding of the Shell.

**Prerequisites:**

Shell programming and UNIX tools courses with a working knowledge of UNIX including at least three (3) months experience in using both the basic command and programming language features of the Shell.

**Duration:**

Three (3) days including classroom lecture and lab sessions.

**PanTech**
**908-281-6759**

*ADVANCED SHELL PROGRAMMING*
*COURSE OUTLINE*

I. **SHELL REVIEW**

- A. Variable Review
- B. Control Structure Review
- C. `select` Statement

II. **SHELL FUNCTIONALITY**

- A. Definitions
  1. Shell Interpreter
  2. Process
- B. Process Related System Calls
  1. Fork
  2. Exec
  3. Wait
- C. Shell Program vs. Compiled Program Process Invocation
- D. Foreground vs. Background Processes
- E. Shell Environment Variables

III. **COMMAND LINE EVALUATION**

- A. Order of Evaluation of Command Line
- B. Command Grouping
  1. Parenthesis `()`
  2. Braces `{}`
- C. Pipes
  1. Exit Status of Pipe
  2. Piping to/from Looping Constructs
  3. Piping to/from Conditional Statements
- D. `eval` Statement

IV. **PARAMETERS AND VARIABLES**

- A. Predefined Variables
- B. `${*}` vs. `${@}`
- C. Positional Parameters
- D. `set` Statement
- E. Keyword Parameters
- F. Special Characters
- G. Quoting
  1. Backslash
  2. Single Quotes
  3. Double Quotes
- H. `getopt` and `getopts` Commands
- I. Arrays

V. **FILE DESCRIPTORS AND REDIRECTION**

- A. Definitions
  1. File Descriptor
  2. File Descriptor Table
  3. Redirection
- B. Multiple Redirections to Same File
- C. File Creation through Redirection
- D. Here Documents
- E. Redirection Using File Descriptors other than 0, 1, and 2
- F. Using `exec` Statement to Open Files

VI. **EXECUTION FLAGS AND BUILT-IN STATEMENTS**

- A. Shell Innovation Flags
  1. Restricted Shell (`-r`)
  2. Interactive Shell (`-i`)
  3. Non-execution Shell (`-n`)
  4. Immediate EXIT Shell (`-e`)
  5. Unset Variable EXIT Shell (`-u`)
- B. Built-in Statements

VII. **READABILITY AND MAINTAINABILITY**

- A. Comments
  1. `#` vs. `:`
  2. Commenting Standards
- B. Formatting Conventions
  1. Control Structures
  2. Here Documents
- C. Variable Naming and Formatting Conventions
- D. Error Message Conventions
- E. EXIT Code Standards
- F. Standard Files and Directories
- G. Manual Page Creation

**PanTech**
**908-281-6759**

VIII. **RELIABILITY**

A. Special Substitutions
   1. Expression Substitution (`:-`)
   2. Variable Substitution (`:=`)
   3. Undefined Variable EXIT (`:?`)
   4. Variable Reassignment (`:+`)
   5. Omitting the Colon (`:`)
B. Shell Debugging Aids
C. Causes for Shell Program Termination
D. Interrupt Processing
   1. `trap` Statement
   2. Catching Signals
   3. Ignoring Signals
   4. Nested Traps
   5. Sending Signals

IX. **EFFICIENCY**

A. PATH Organization
B. Full vs. Relative Pathnames
C. Use of Built-in Statements
D. Benchmarking Programs
E. Optimizing Pipes
F. Efficient Use of Control Structures
G. Recursion
H. Shell Functions
   1. Syntax
   2. Autoloading
   3. Recursion
I. Shell vs. 'C'