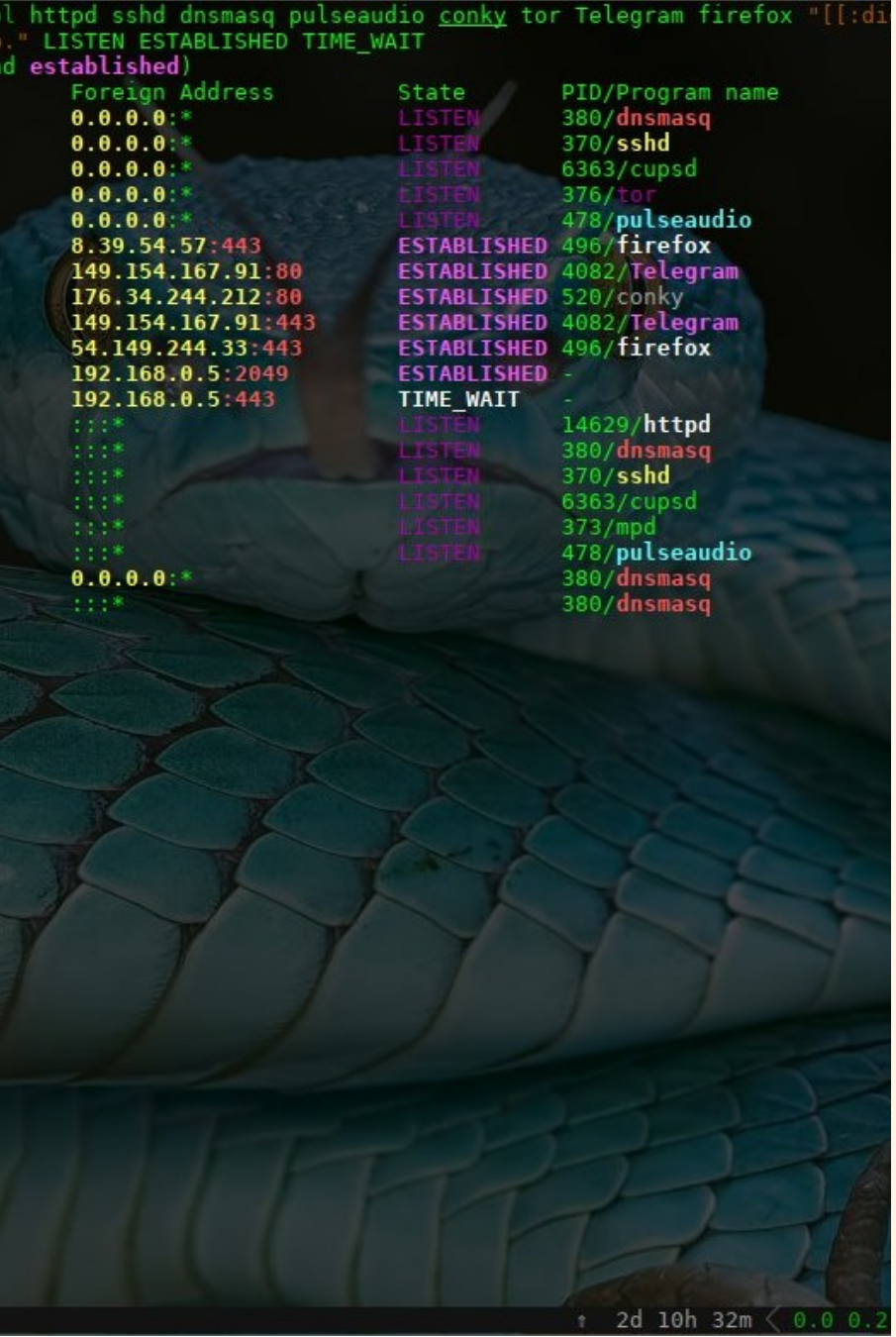


Shell Scripting in Linux

Welcome to the world of shell scripting in Linux! In this presentation, we will explore the power and possibilities of automating tasks using the Linux command line.

by Vinitha Anumulapuri

Made with Gamma

The Basics of Shell Scripting

What is a Shell Script?

A shell script is a file containing a series of commands that the shell can execute.

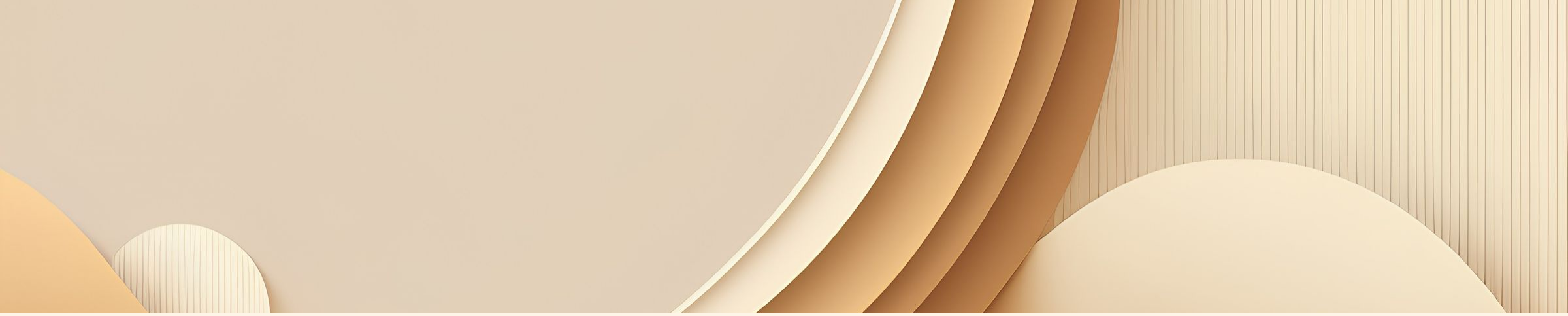
Why Use Shell Scripts?

Shell scripts can greatly simplify repetitive tasks, enhance productivity, and automate complex processes.

Getting Started

All you need is a text editor and a terminal to start writing shell scripts.

Made with Gamma



Variables and Data Types

1 Creating Variables

You can assign a value to a variable using the assignment operator (=).

2 Data Types

Shell scripting supports string, numeric, and boolean data types.

3 Using Variables

Variables allow you to store and manipulate data within your scripts.

Made with Gamma

Conditional Statements

1

If Statements

Use if statements to perform different actions based on specified conditions.

2

Else Statements

Else statements provide alternative actions when the conditions of the if statement are not met.

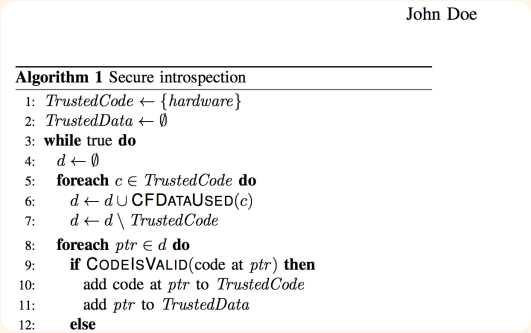
3

Case Statements

Case statements offer a structured way to handle multiple possible conditions.

Made with Gamma

Loops and Iteration



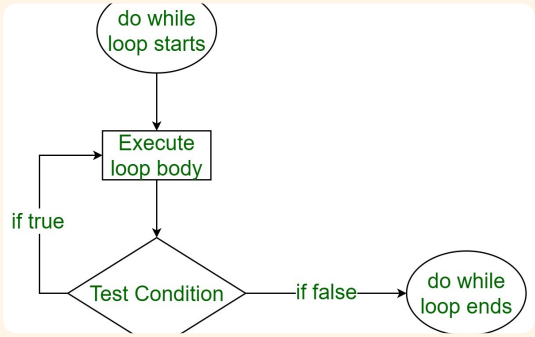
John Doe

Algorithm 1 Secure introspection

```
TrustedCode ← {hardware}
TrustedData ← ∅
while true do
  d ← ∅
  foreach c ∈ TrustedCode do
    d ← d ∪ CFDataUsed(c)
    d ← d \ TrustedCode
  foreach ptr ∈ d do
    if CodeIsValid(code at ptr) then
      add code at ptr to TrustedCode
      add ptr to TrustedData
    else
```

For Loops

A for loop allows you to repeatedly execute a block of code for a specified number of times.



While Loops

While loops execute a block of code as long as a specified condition is true.

Do-While Loops

Do-while loops ensure that the code block is executed at least once, even if the condition is initially false.

Made with Gamma

Input and Output

Standard Input

Standard input allows you to accept data from the user or other sources.

Standard Output

Standard output allows you to display the results of your scripts.

Redirection

You can redirect input and output to and from files using redirection operators.

Made with Gamma

Functions and Libraries

1

User-Defined Functions

Create reusable code blocks by defining your own functions.

2

Built-in Functions

Linux provides a rich set of built-in functions that can be used to perform common tasks.

3

Libraries and Modules

Extend the functionality of your scripts by using libraries and modules.

Made with Gamma

Error Handling

1 Exit Codes

Learn how to interpret exit codes to determine the success or failure of a script.

2 Error Messages

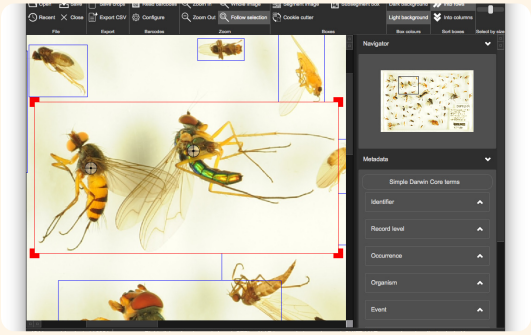
Create informative and user-friendly error messages to assist in troubleshooting.

3 Error Handling Techniques

Explore different error handling techniques to ensure robustness in your scripts.

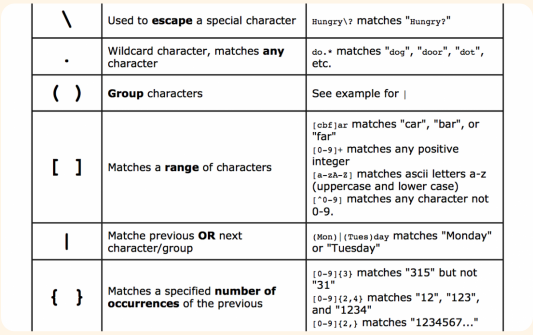
Made with Gamma

Advanced Techniques



File Manipulation

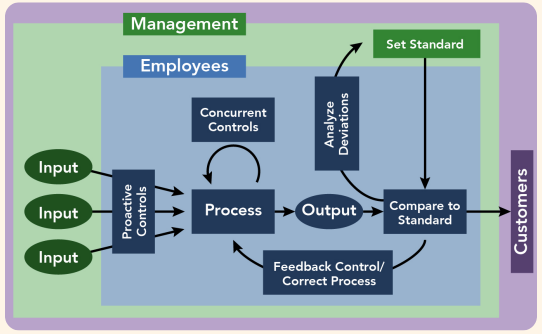
Learn how to read, write, and modify files using shell scripting.



\	Used to **escape** a special character	`Hungry\?` matches "Hungry?"
.	Wildcard character, matches **any** character	`do.` matches "dog", "doo", "dot", etc.
( )	**Group** characters	See example for |
[ ]	Matches a **range** of characters	`[cbf]ar` matches "car", "bar", or "far" `[0-9]+` matches any positive integer `[a-zA-Z]` matches ascii letters a-z (uppercase and lower case) `[^0-9]` matches any character not 0-9.
|	Matche previous **OR** next character/group	`(Mon)|(Tues)day` matches "Monday" or "Tuesday"
{ }	Matches a specified **number of occurrences** of the previous	`[0-9]{3}` matches "315" but not "31" `[0-9]{2,4}` matches "12", "123", and "1234" `[0-9]{2,}` matches "1234567..."

Regular Expressions

Utilize powerful pattern matching techniques to search for and manipulate text.



Process Management

Gain control over system processes using shell scripts.

Made with Gamma