**Revision Notes: Introduction to Shell Scripting**

**Agenda**

* Introduction to Shell Scripting
* Understanding Shell and Bash Scripts
* Wildcards in Shell Scripting
* Control Structures: if-conditions
* Shell Script Variables
* File Checks in Shell Scripting
* Functions in Bash Script

**1. Introduction to Shell Scripting**

Shell scripting is a powerful way to automate tasks in Unix-based systems. It involves writing a series of commands in a script file to be executed by the shell. This can assist in managing repetitive tasks in system administration and DevOps【6:0†transcript.txt】 .

**Bash Script and Shell Script:**

* A shell script is a collection of commands.
* Bash (Bourne Again SHell) is the most widely used shell scripting language.

**2. Fundamentals of a Shell Script**

**A. Writing a Bash Script**

* Begins with a "shebang" (#!) to specify the interpreter.
* Consists of a series of commands that are executed sequentially【6:9†transcript.txt】.

**B. Running a Bash Script**

* The script must have execute permissions. Use chmod +x scriptname.sh to add this permission.
* To run a script, use ./scriptname.sh【6:11†transcript.txt】.

**3. Wildcards in Shell Scripting**

Wildcards are symbols that enable pattern matching. They simplify file searching and manipulation【6:0†transcript.txt】.

* \*: Matches any number of characters. Example: \*.log finds all .log files.
* ?: Matches exactly one character. Example: ?.txt finds a.txt, b.txt, etc.
* []: Matches any one of the enclosed characters. Example: [abc]\*.log matches a.log, b.log, c.log【6:7†transcript.txt】 .

**4. Control Structures**

**A. If Conditions**

Control structures allow scripts to make decisions and execute different code paths based on conditions.

* Basic Syntax:
* if [ condition ]; then
* # commands
* fi
* Use elif for else-if conditions and else for an alternative path【6:3†transcript.txt】.
* String comparisons use == or =. Numeric comparisons use -eq (equal), -ne (not equal), -gt (greater than), -lt (less than) .

**5. Shell Script Variables**

There are two types of variables in shell scripting:

* **Local Variables:** Defined inside scripts and are accessible only within the script.
* **Environment Variables:** Have a global scope and are accessible by any script running in the shell【6:5†transcript.txt】【6:9†transcript.txt】.

**How to Define a Variable**

variablename=value

Variables are treated as strings by default. For numeric operations, ensure appropriate evaluation【6:5†transcript.txt】.

**6. File Checks**

Perform file and directory checks using conditional expressions:

* -e filename: Returns true if the file exists.
* -f filename: Checks if it's a regular file.
* -d filename: Validates if it's a directory.
* -s filename: Checks if the file size is greater than zero【6:10†transcript.txt】 .

**7. Functions in Bash Script**

Functions in bash script facilitate code reusability by grouping commands.

**Defining a Function**

function\_name() {

# commands

}

Call functions by name without parentheses. Use positional parameters ($1, $2, ...) to pass arguments【6:4†transcript.txt】.

**Conclusion**

Shell scripting empowers users to automate complex tasks through scripts that encapsulate multiple commands and logic. As you advance, experiment with different components of shell scripting to grasp various efficiencies and automations it offers【6:14†transcript.txt】.