Basic Shell Scripting Questions

1. **What is a shell script?**
    - A shell script is a text file containing a series of commands that the shell can execute. It automates tasks in a Unix/Linux environment.
2. **How do you create a shell script?**
    - You can create a shell script using any text editor (e.g., `nano`, `vim`, `gedit`). Start the file with a shebang (`#!/bin/bash`) to specify the interpreter.
3. **How do you make a shell script executable?**
    - You can make a shell script executable using the `chmod` command: `chmod +x script.sh`.
4. **What is the purpose of the shebang (`#!`) in a shell script?**
    - The shebang indicates the interpreter that should be used to execute the script. For example, `#!/bin/bash` specifies the Bash shell.
5. **How do you pass arguments to a shell script?**
    - You can pass arguments to a shell script when executing it. Inside the script, you can access them using `$1`, `$2`, etc., for the first, second, and subsequent arguments.

Intermediate Shell Scripting Questions

6. **What is the difference between `=` and `==` in shell scripting?**
    - In shell scripting, `=` is used for assignment, while `==` is used for string comparison in conditional statements (e.g., `if` statements).
7. **How do you read user input in a shell script?**
    - You can read user input using the `read` command. For example: `read -p "Enter your name: " name`.
8. **What is a conditional statement in shell scripting?**
    - A conditional statement allows you to execute different commands based on certain conditions. Common conditional statements include `if`, `else`, and `case`.
9. **How do you create a loop in a shell script?**
    - You can create loops using `for`, `while`, or `until`. For example:

```
bash
RunCopy code
1for i in {1..5}; do
2    echo "Number $i"
3done
```

10. **What is the purpose of the `exit` command in a shell script?**

- The `exit` command terminates the script and can return a status code to the calling process. A status of `0` typically indicates success, while any non-zero value indicates an error.

Advanced Shell Scripting Questions

11. **What are functions in shell scripting?**
    - Functions are reusable blocks of code that can be defined and called within a script. They help organize code and avoid repetition.
12. **How do you define a function in a shell script?**
    - You can define a function using the following syntax:

```
bash
RunCopy code
1function_name() {
2    # commands
3}
```

13. **What is the purpose of the `trap` command?**
    - The `trap` command allows you to specify commands to be executed when the script receives certain signals or exits, helping to manage cleanup tasks.
14. **How do you handle errors in shell scripting?**
    - You can handle errors using conditional statements, checking the exit status of commands with `$?`, and using the `trap` command for cleanup.
15. **What is the difference between `>` and `>>` in shell scripting?**
    - The `>` operator redirects output to a file, overwriting it, while `>>` appends output to the end of a file without overwriting.

Shell Scripting Best Practices

16. **What is the purpose of comments in shell scripts?**
    - Comments (preceded by #) are used to document the code, making it easier to understand and maintain.
17. **How do you debug a shell script?**
    - You can debug a shell script using the `-x` option when running the script (e.g., `bash -x script.sh`) to display each command before execution.
18. **What is the `set -e` command?**
    - The `set -e` command causes the script to exit immediately if any command returns a non-zero exit status, helping to catch errors early.
19. **How do you check if a file exists in a shell script?**
    - You can check if a file exists using the `-e` flag in a conditional statement:

```
bash
RunCopy code
1if [ -e filename ]; then
2    echo "File exists."
3fi
```

20. **What is the purpose of the `grep` command in shell scripting?**

    - The `grep` command is used to search for specific patterns in files or output, making it useful for filtering data.

Practical Shell Scripting Questions

21. **How do you create a backup of a file using a shell script?**

    - You can create a backup of a file using the `cp` command. For example:

```
bash
RunCopy code
1cp original_file.txt backup_file.txt
```

22. **How do you find and replace text in a file using a shell script?**

    - You can use the `sed` command for find and replace. For example:

```
bash
RunCopy code
1sed -i 's/old_text/new_text/g' filename.txt
```

23. **How do you list all files in a directory using a shell script?**

    - You can list all files using the `ls` command:

```
bash
RunCopy code
1ls -l
```

24. **How do you check the disk usage of a directory in a shell script?**

    - You can check disk usage using the `du` command:

```
bash
RunCopy code
1du -sh /path/to/directory
```

25. **How do you schedule a shell script to run at a specific time?**

    - You can schedule a shell script using `cron`. You would add an entry to the crontab file using `crontab -e`, specifying the time and the script path. For example:

```
RunCopy code
10 2 * * * /path/to/script.sh
2``` ### Additional Shell Scripting Questions
```

26. **What is the purpose of the `awk` command in shell scripting?**

- The `awk` command is a powerful text processing tool used for pattern scanning and processing, allowing you to manipulate and analyze data in files or input streams.

27. **How do you concatenate strings in a shell script?**

- You can concatenate strings by simply placing them next to each other. For example:

```
bash
RunCopy code
1str1="Hello"
2str2="World"
3result="$str1 $str2"
```

28. **What is the difference between `$(...)` and backticks `` `...` `` in shell scripting?**

- Both are used for command substitution, but `$(...)` is preferred for readability and nesting commands, while backticks can be harder to read and nest.

29. **How do you check the exit status of the last command in a shell script?**

- You can check the exit status using the special variable `$?`, which holds the exit code of the last executed command.

30. **What is the purpose of the `find` command in shell scripting?**

- The `find` command is used to search for files and directories in a directory hierarchy based on various criteria, such as name, type, size, and modification time.

31. **How do you create a temporary file in a shell script?**

- You can create a temporary file using the `mktemp` command, which generates a unique temporary file name.

32. **What is the purpose of the `sort` command in shell scripting?**

- The `sort` command is used to sort lines of text files or input streams in a specified order, such as ascending or descending.

33. **How do you redirect both stdout and stderr to a file in a shell script?**

- You can redirect both stdout and stderr using `&>` or by using `>` for stdout and `2>` for stderr:

```
bash
RunCopy code
1command > output.txt 2>&1
```

34. **What is the purpose of the `cut` command in shell scripting?**

- The `cut` command is used to extract sections from each line of input, allowing you to specify delimiters and fields.

35. **How do you check if a variable is empty in a shell script?**

- You can check if a variable is empty using a conditional statement:

```
bash
RunCopy code
1if [ -z "$variable" ]; then
```

```
2    echo "Variable is empty."
3fi
```

36. **What is the purpose of the `basename` command?**

- The `basename` command is used to strip the directory and suffix from filenames, returning just the file name.

37. **How do you create a directory in a shell script?**

- You can create a directory using the `mkdir` command:

```
bash
RunCopy code
1mkdir new_directory
```

38. **What is the purpose of the `chmod` command in shell scripting?**

- The `chmod` command is used to change the file mode (permissions) of a file or directory, controlling who can read, write, or execute it.

39. **How do you check the current working directory in a shell script?**

- You can check the current working directory using the `pwd` command.

40. **What is the purpose of the `echo` command in shell scripting?**

- The `echo` command is used to display text or variables to the standard output, making it useful for printing messages or debugging.

41. **How do you use `sed` to delete lines from a file?**

- You can use `sed` to delete lines by specifying the line number or pattern. For example:

```
bash
RunCopy code
1sed -i '2d' filename.txt  # Deletes the second line
```

42. **What is the purpose of the `tee` command in shell scripting?**

- The `tee` command reads from standard input and writes to standard output and files simultaneously, allowing you to view and save output at the same time.

43. **How do you check the number of lines in a file using a shell script?**

- You can check the number of lines in a file using the `wc` command:

```
bash
RunCopy code
1wc -l filename.txt
```

44. **What is the purpose of the `ps` command in shell scripting?**

- The `ps` command displays information about currently running processes, allowing you to monitor system activity.

45. **How do you terminate a running process in a shell script?**

- You can terminate a running process using the `kill` command followed by the process ID (PID):

```
bash
RunCopy code
1kill PID
```

21. **What is the purpose of the `env` command?**
    - The `env` command is used to display the current environment variables and their values, or to run a command in a modified environment.
22. **How do you check the system uptime in a shell script?**
    - You can check the system uptime using the `uptime` command, which shows how long the system has been running.
23. **What is the purpose of the `history` command in shell scripting?**
    - The `history` command displays a list of previously executed commands, allowing users to recall and reuse them.
24. **How do you create a symbolic link in a shell script?**
    - You can create a symbolic link using the `ln -s` command:

```
bash
RunCopy code
1ln -s target_file link_name
```

25. **What is the purpose of the `alias` command in shell scripting?**
    - The `alias` command is used to create shortcuts for longer commands, making it easier to execute frequently used commands with a simple name