|  |  |  |
| --- | --- | --- |
| **Number** | **Description** | **Experiments** |
| 1 | **Login Shell:** Starts with a - in its name or with --login.  **Interactive Shell:** Started without a script, or with -i, and connected to a terminal.  **What Bash Reads at Startup :**  1. Interactive Login Shell  Reads /etc/profile first.  Then reads the first readable file from:  ~/.bash\_profile  ~/.bash\_login  ~/.profile  On exit: runs ~/.bash\_logout  Use --noprofile to skip these.  2. Interactive Non-Login Shell  Reads ~/.bashrc (if it exists).  Use --norc to skip this.  Use --rcfile <file> to load a custom config.  3. Non-Interactive Shell (e.g., running scripts)  If BASH\_ENV is set, Bash runs the file it points to.  **When Bash is Invoked as sh**  Acts like the traditional sh shell.  Reads /etc/profile and ~/.profile (for login).  Uses ENV variable (for interactive).  --rcfile has no effect.  Enters POSIX mode after startup files. | A close-up of a document  AI-generated content may be incorrect. |
| 2 | Reserved words of bash shell : **! case do done elif else esac fi for function if in select then until while { } time [[ ]]** |  |
| 3 | A **simple command** is a **single command** with **optional arguments** and **redirections**. sample syntax : **command [arguments] [redirection]** Examples : **ls -l**  **echo "Hello, World"**  **cp file1.txt file2.txt**  **rm myfile.txt** |  |
| 4 | A **pipeline** in Bash is a way to **connect multiple commands,** where the **output of one command becomes the input of the next.**  **Sample syntax:** cmd1 | cmd 2 | cmd 3 ...  **Examples :** cat myfile.txt | grep "error" grep "warning" log.txt | wc -l cat names.txt | sort | uniq |  |
| 5 | command lists in Bash that let us run multiple commands together using special separators. There are 3 types of separators:   1. ; Run commands one after another, regardless of success  *echo "Start"; echo "End"* 2. & Run the command in the background   *long\_running\_script.sh & echo "Launched!"*   1. && Run the next command only if the previous succeeds.   *[ -f myfile.txt ] && echo "File exists"*   1. || Run the second command only if the first command fails (i.e., returns a non-zero exit status).  *[ -f myfile.txt ] || echo "File NOT found"*   *mkdir test && cd test || echo "Failed to enter test"* | A white text on a black background  AI-generated content may be incorrect. |
| 6 | A compound command in Bash is a group of commands executed together using control structures like loops, conditionals, or grouping symbols. These are more complex than simple commands.  Examples : { echo "Hello"; echo "World"; }   if [ -f myfile.txt ]; then  echo "File exists"  else  echo "No file"  fi   for i in 1 2 3; do  echo "Number: $i"  done  while [ $x -lt 5 ]; do  echo $x  x=$((x + 1))  done  greet() {  echo "Hello, $1!"  }  greet Prashantha |  |
| 7 |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **Number** | **Goal** | **Comment** |
| 1✅ | Basic principles | Find out the concepts of login shell, interactive shell, bash initialization files and when the settings for each of them take effect. |
| 2✅ | Reserved words of bash shell | Find out the meanings of at least five of these, for example. Why can't these be used for any other purpose? (You should probably do this last, as this is easier to do when you have more experience) |
| 3✅ | Simple Command | What is this ? Give some example. |
| 4✅ | Pipeline | Tell me what this is. Make an example pipeline and show how it works. (again worth doing a little later). |
| 5✅ | Command list | Make a command list of two commands that shows what the operators && and || do. (do this later). |
| 6 | Compound Command | Give a simple example of this. |
| 7 | Expression | Give a couple of examples of these. |
| 8 | For control structure | Make a shell command containing a loop made with for. |
| 9 | Second variant of for (very similar to basic C++ for) | Do same as in above. |
| 10 | Select control structure | Try this. |
| 11 | Case control structure | Try this. |
| 12 | If control structure | Try this. |
| 13 | While control structure | Try this. |
| 14 | Until control structure | Try this. |
| 15 | Quoting mechanism | What is this ? Test how it works. |
| 16 | Shell variable | How are these made ? Make couple of variables and try them. |
| 17 | Array | How you make this ? How you use it ? Make trials to inspect this. |
| 18 | Shell script | How do you make a short shell script that prints for example “Hello, World!” |
| 19 | Comments | How you write these to shell scripts ? Try it. |
| 20 | Expansion mechanism | For example, what do you think are the most common of these: what does this mean? Give it a try. |
| 21 | Command substitution mechanism | What is this ? Make a trial! |
| 22 | Arithmetic | How to do arithmetic operations in the bash shell ? Try these! |
| 23 | Pathname expansion mechanism | What does this mean ? Try a few cases. |
| 24 | Redirection mechanism | What does this mean ? Give it a try. For example, what does > word 2>&1 mean? What about: >> word 2>&1? |
| 25 | Conditional Expressions | What are these ? Try a few. |
| 26 | Environment variables | What are these ? How are they different from regular shell variables? Try their functionality. |
| 27 | Return value | How you pass return values from a shell script to its caller. Make a small example: returning & utilizing the return value. |
| 28 | Line input | How do you read input to a variable from a terminal in shell script ? Try some examples. |
| 29 | History mechanism | What is this ? How do you take advantage of this? Try examples. |
| 30 | Shell internal commands | Find out for example about five different commands (try to choose commonly used ones) |
| 31 | Shell function | Try this. A few parameters must be passed to the function, and again a return value must be passed from the function to the caller in some way. |
| 32 | Command line arguments to a shell script | Try this mechanism. |
| 33 | Find out the meaning of some common shell ready made variables | For example five of them is enough in here. |

An example of working according to the goal could be the following:

• Objective: to learn how to handle shell variables

• Working: search sources for information on what shell variables are, how they are handled, etc.

• You present a set of shell commands / shell scripts that show the accumulation of the required competence by running them.

• Return the following to the table:

o a brief verbal description of what the objective means; e.g. what are shell variables?

o Your experiment in screenshots, text copies, etc.

o a brief summary of the lesson learned; e.g., where can I take advantage of shell variables?

For goals, try to focus on the main things, which are usually simpler than, for example, the exceptions to the details on the man page, etc. So try not to drown in the “detail porridge” and see the forest from the trees: what is this bigger issue (e.g. handling variables)? The details can be studied in real use situations, for example in industry, but it is much easier when the main things are in place.