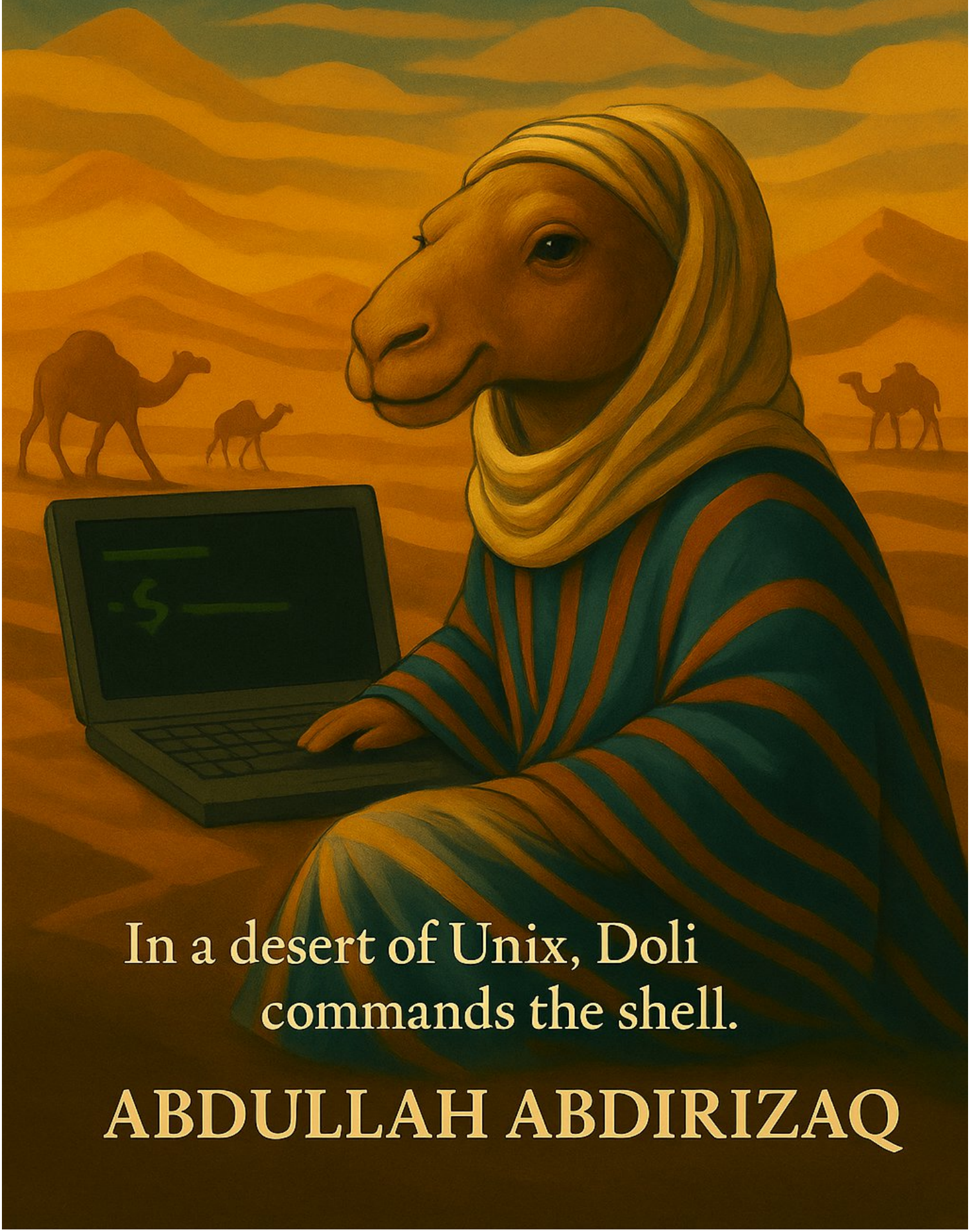
Doli in the Shell



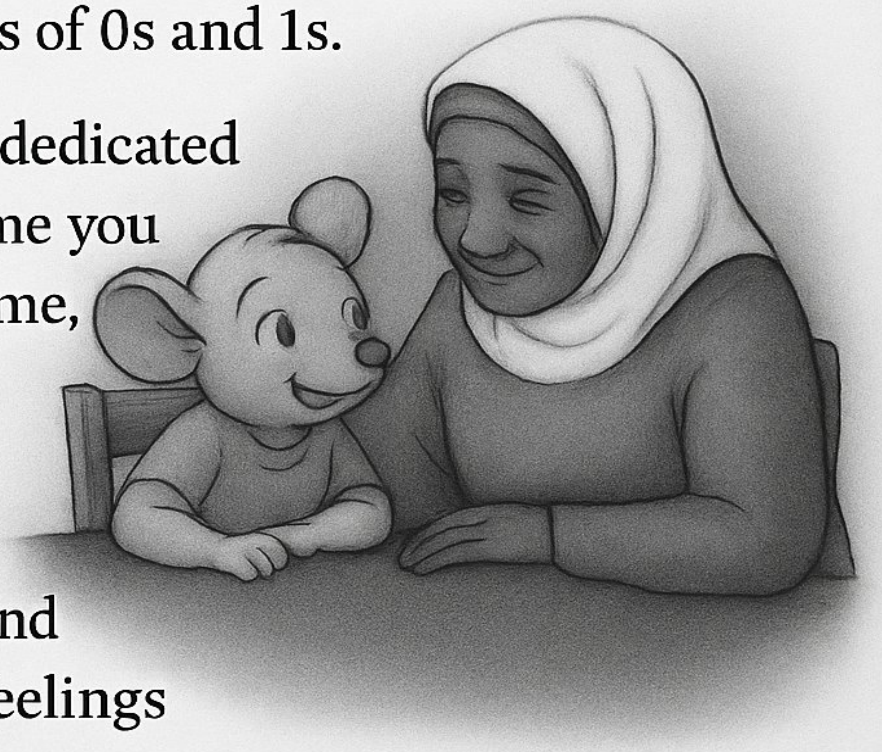
In a desert of Unix, Doli
commands the shell.

ABDULLAH ABDIRIZAQ

DEDICATION

To my Aunt Hodan, who bravely tried to teach me mathematics. While she patiently showed me techniques like using her fingers to multiply six through nine, it wass the 0s and 1s that proved an enduring struggle. Just as operating systems abstract hardware details from the user, so too are the ones and zeroes of binary language hidden beneath digital communication. Imagine everything you see, touch, and hear—all reduced to combinations of 0s and 1s.

This book is dedicated for all the time you fought with me, even though the world of binary still evades me, and despite my feelings



Doli-in-the-shell

Contents

Chapter1:

What is the Shell?

Doli, the little mouse, think the shell is the most important in the computer.

From Doli's perspective, almost everything you interact with on your computer is, in some fundamental way, a 'shell' of sorts!"

"Let Doli explain further," she says, gesturing towards the screen.

The Shell: Your Gateway to the Kernel

"At the very heart of your computer, deep inside, lives the Kernel. It's the true brain, the master controller, the core program that manages everything – the memory, the processor, the hard drive, the network, every single piece of hardware and software. But the Kernel is like a brilliant, ancient deity that only understands its own complex, low-level language. It doesn't have a 'face' or a 'voice' for you to interact with directly."

"So, you need an intermediary. You need a Shell!"

Two Faces of the Same Coin: Text-Based vs. Graphical Shells

"A 'shell' is simply a program that provides you with an interface to the operating system's Kernel. It's your gateway, your control panel, your translator," Doli emphasizes. "And these gateways come in different forms:"

The Command-Line Shell (What we're mastering!):

"This is the one we've been exploring – the text-based interface, like your terminal window," Doli says, tapping the console line. "Here, you type commands directly, and the Shell (like Bash, Zsh, or Fish) interprets those text commands into instructions the Kernel understands. It's direct, powerful, and incredibly precise. It's like speaking directly to the computer in its own specialized language."

Purpose: Direct command, automation, scripting, remote control, fine-grained system management.

The Graphical Shell (Your Desktop, Widgets, Apps!):

"But then, there's the other kind of shell – the one you probably use most often without even thinking about it!" Doli exclaims, pointing to an imaginary desktop. "Your Desktop Environment (like GNOME, KDE Plasma, macOS Finder, or Windows Explorer) – that's a graphical shell!"

"When you click an icon, drag a file, open a window, or interact with a widget, you're not talking directly to the Kernel. Oh no! You're interacting with the graphical shell program. This program then translates your clicks and drags into the underlying commands and instructions that the Kernel can execute."

Purpose: Visual interaction, ease of use, intuitive navigation for everyday tasks, multimedia, web browsing.

Doli's Conclusion: All Paths Lead to the Kernel

"So, you see?" Doli concludes, doing a little happy dance. "Whether it's the blinking text prompt or the colorful icons on your screen, they are both shells! They are both interfaces designed to make the powerful, silent Kernel understandable and usable for you."

"The Command-Line Shell is the deep, precise, and infinitely scriptable way. The Graphical Shell is the visually intuitive and user-friendly way."

"Both are essential, but for the true Data Alchemist, understanding and mastering the Command-Line Shell unlocks a level of power, automation, and control that the graphical world can only hint at!"
(Doli gives a knowing wink, then scurries off, leaving you with the profound realization that you're always interacting with a 'shell' of some kind.)

What Does "Always Interacting with a 'Shell' of Some Kind" Mean?

Think of a **shell** *as an* **interface** *– a protective outer layer or a way to communicate with something deeper.*

Computer Shells:

Command Line (like a Unix terminal): *This is the most literal "shell." You type commands (text) into it, and the shell translates those commands into instructions the computer's core (the operating system, the hardware) can understand and act upon. It then shows you the results. It's a text-based interpreter.*

Graphical User Interfaces (GUIs): *Even when you're clicking icons, dragging windows, or tapping on your phone screen, you're still using a shell! The buttons, menus, and visual elements are a "shell" that translates your clicks and taps into commands for the operating system. It's a visual interpreter.*

Apps: *Every app you use is a kind of shell. When you tap "Like" on social media, that tap is translated by the app's shell into a complex instruction for servers far away.*

Beyond Computers:

Language: *When you speak, you're using language as a "shell" to convey your thoughts and feelings. The words themselves aren't the thoughts, but they're the interface through which you express them.*

Clothes: *Your clothes are a "shell" that protects your body and conveys information about you (style, professionalism, etc.).*

Social Roles: *When you act as a student, a parent, or an employee, you're interacting through a "shell" of expectations and behaviors associated with that role.*

The core idea is that we rarely interact directly with the raw, underlying reality. Instead, we use **layers of abstraction** *– these "shells" – that make interaction easier, safer, or more manageable. They take our simple input and translate it into something more complex, or they present complex information in a simplified way.*

Doli's wink is a playful acknowledgment of this constant translation and interpretation happening all around us!

The Unix Philosophy: Small Tools, Big Power
The Unix shell isn't just a command line. Its a playground full of tiny Tools.

Each tool does one simple job well. But when you connect them together, you can build something huge and powerful just like building with Lego bricks.

Each Tool Does One Job
Unix doesn't give you one massive application with every button. It gives you many small commands that you can mix and match.

[cat] read files
[grep] find text
[sort] arrange lines
[wc] count words or lines
[cut] slice out columns
[awk] scan & process text
Each one seems simple... but together?
Combine Them to Do Big Things
You can chain them using the `|` (pipe) symbol.
The output of one becomes
the input of another like passing a note from one person to the next.

+-------+ +--------+ +-------+ +-----+
FILE | cat | | grep | | sort | | head | DONE
+-------+ +--------+ +-------+ +-----+

This command means:
Bash cat logs.txt | grep error | sort | head
Read the file, find lines with 'error', sort them,
and show the top few."

Use small tools. Connect them.
Build what you need. Thats the Unix way.

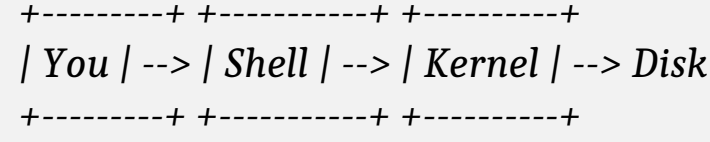
The Shell Hides the Mess
Even though it looks simple,
the shell does a lot of heavy lifting for you.
When you type:

The shell will remember every command you type,
The shell will allow you to search, for files and directories,
If you type a letter the shell will list any commands or program
The more you type the smaller the list, less you type the more
related commands shown to you.

The shell is performing something call auto-complete or read-line.
Different shells have different features, some shells more inter-active then other,
Some more sciencertivice then others, and some are standard but if shell will increase
In size, productivity because the shell will keep learning and the user will keep adding
more capability.

Bash cp file.txt /backup/

Here's what's really happening:



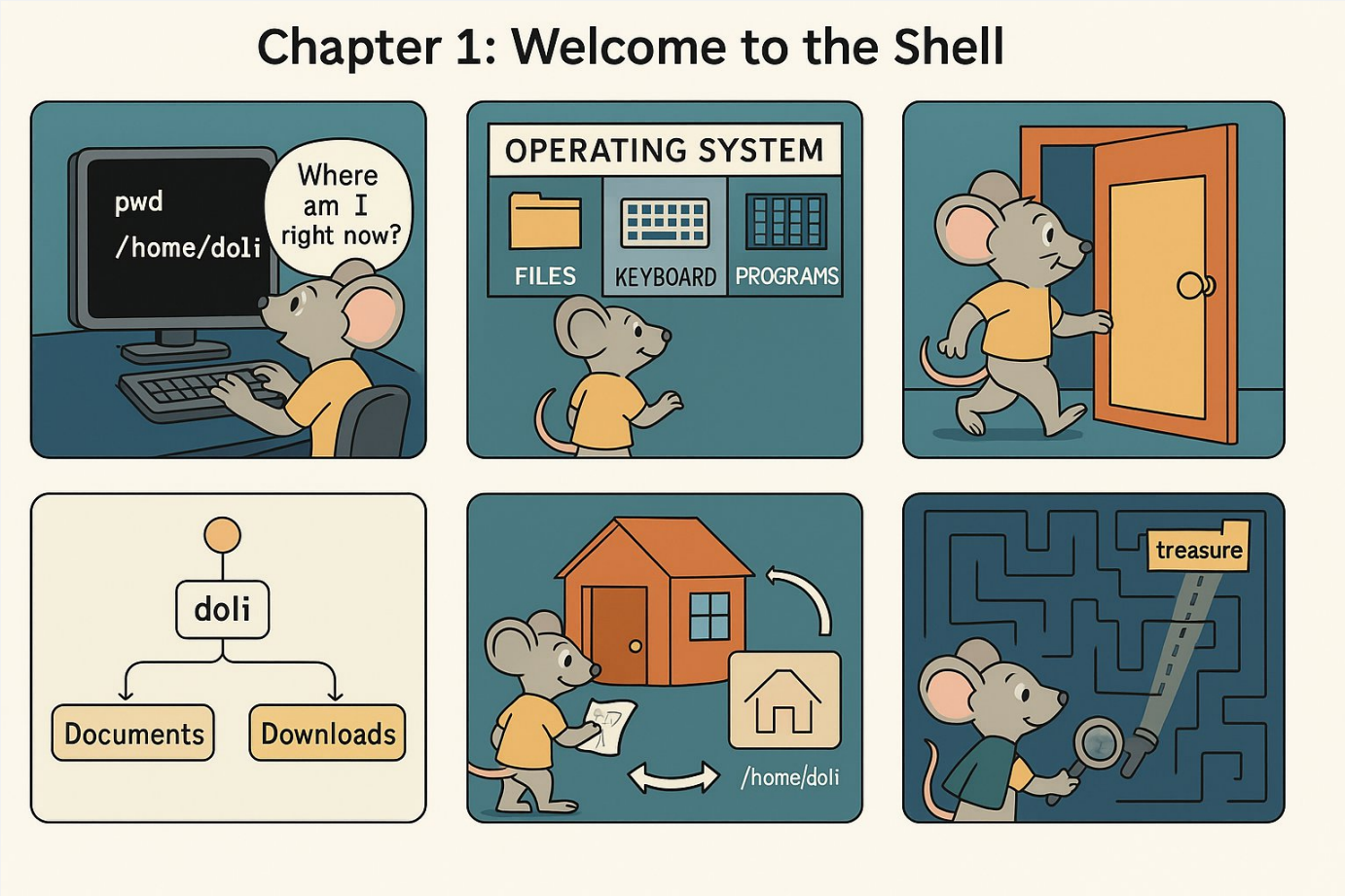
Figures out where "file.txt" is
Checks if "/backup/" exists
Talks to the OS to move the data

You don't need to know about hardware, filesystems, memory the shell abstracts all of that away.
Just type. The shell handles the hard parts.

Why This Helps (Especially for ADHD Minds)
Tiny tools = less to remember Mix and match = freedom to explore Shell hides the wires = fewer distractions The Unix shell lets you focus on what you want to do,not how the computer works insideYou = The Builder The shell is not a program. Its your work bench, your remote control, your magic wand

.

YOU
[cat notes.txt]
[grep important]
[sort]
[head -n 5]



Imagine Doli wants to open a door. Instead of clicking, Doli types:

open door

The shell reads it and tells the OS, "Doli wants this door open."

Comic Panel:

Doli types: open door

Shell says: "Processing..."

OS swings the door open.

It's precise, fast, and fun!

1.4 Why is the Shell Important? "Why not just click stuff?" you ask

Because the shell can do things faster, smarter, and from far away.

Doli's Superpowers with the Shell:

Speed: A few typed words can do what 10 clicks would.

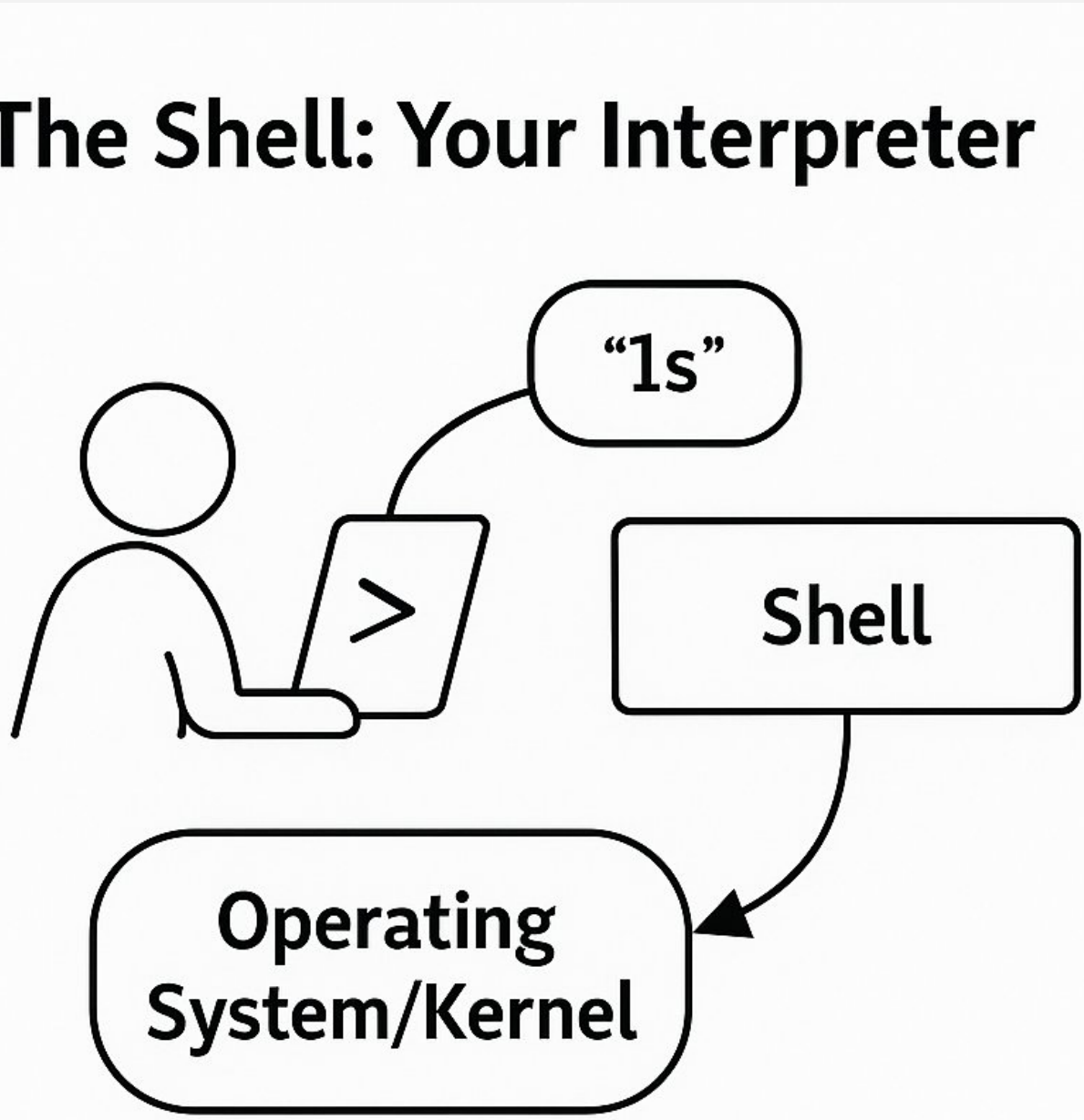
Flexibility: Mix and match commands to automate chores.

Remote Control: Use the shell to command other computers, even on the moon!

Scripting: Write a spellbook (script) to repeat jobs anytime.

Doli's Tip: Shell = Superpowers + Simplicity.

What made the shell or still make the shell and will make the shell relevant is its ability to combine many Different tools to make something unique.
Cat > doli.txt | sort | printed.
There is no mouse which mean no graphic, so the shell Provide other ways of been productive and it has from Writing text, to viewing images, or even playing games The shell provides, the shell also know where you been why type it again, more importantly if you type a command once the auto completes for you.
You keep hearing automation or scripting all those not the some thing , is nothing more then commands you don't type any more



1.5.1 pwd – Present Working Directory

pwd

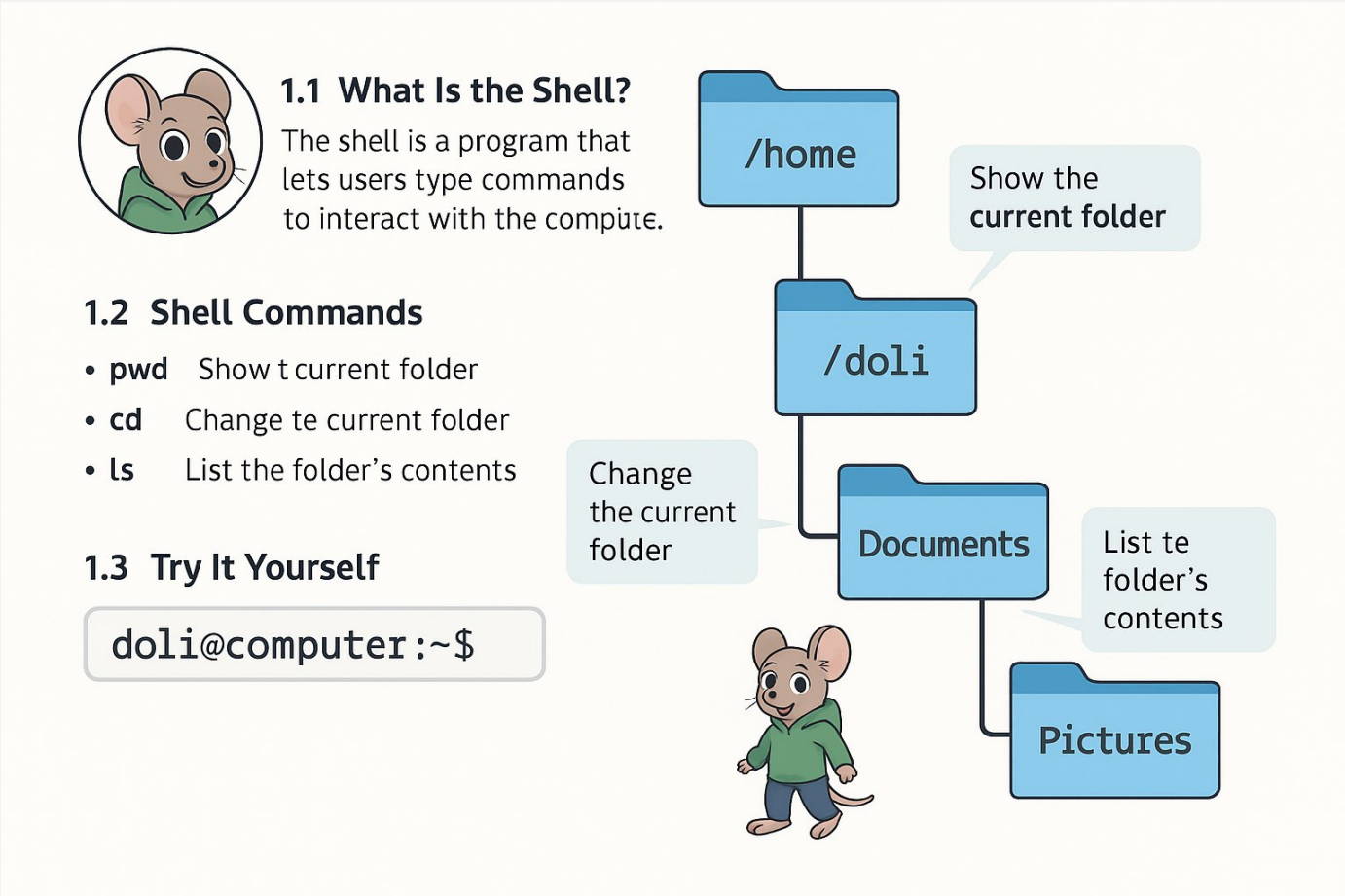
Shows Doli's current location.

Example Output:

/home/doli

Doli's Note: "I'm at my home. Good to know where I start!"

Visual: A tree diagram with Doli marked at /home/doli.

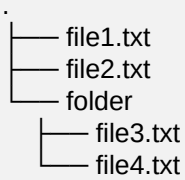


1.5.2 ls – List What's Here

ls

Shows the items in the current location.

Example Output:



Documents Downloads Music Pictures

Visual: Doli looking at signs labelled "Documents", "Downloads", etc.

1.5.3 cd – Change Directory

cd Documents

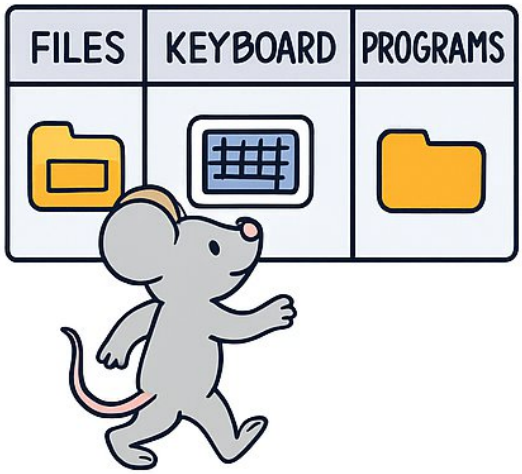
Moves Doli into the "Documents" room.
cd ..

Doli goes one step back—to the previous room.

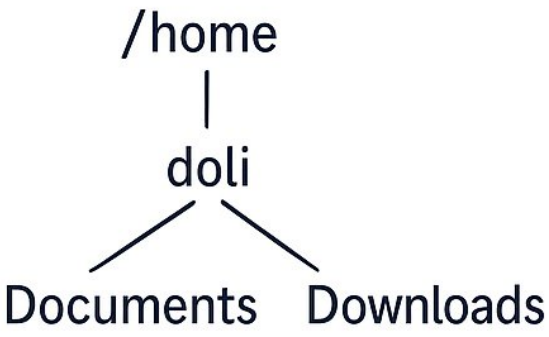
Visual: An animation frame showing Doli walking through doors labelled with

folder names.

1. Doli at Terminal Using *pwd*



2. OS Control Center

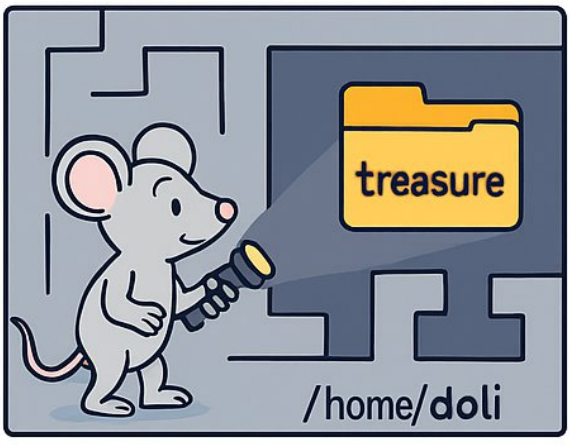


3. Folder Tree Diagram

4. Doli Walking to Documents

/home/doli

5. Doli Walking Back Through Folders



5. Doli Finding *treasure* Folder in Maze

1.6 Doli's First Adventure

Let’s follow Doli's first few steps:

Where am I?
pwd

Terminal: /home/doli
What’s around me?

ls

Terminal: Documents Downloads Music
Let’s go to Documents

cd Documents

Doli tiptoes in, tail twitching.
Check location again

pwd

Terminal: /home/doli/Documents
Go back to hallway

cd ..

Activity: Open a terminal and do what Doli did! Write your journey down like a map.

1.7 Practice with Doli

Try it with Doli!

Open your terminal and try:

pwd ls cd .. pwd

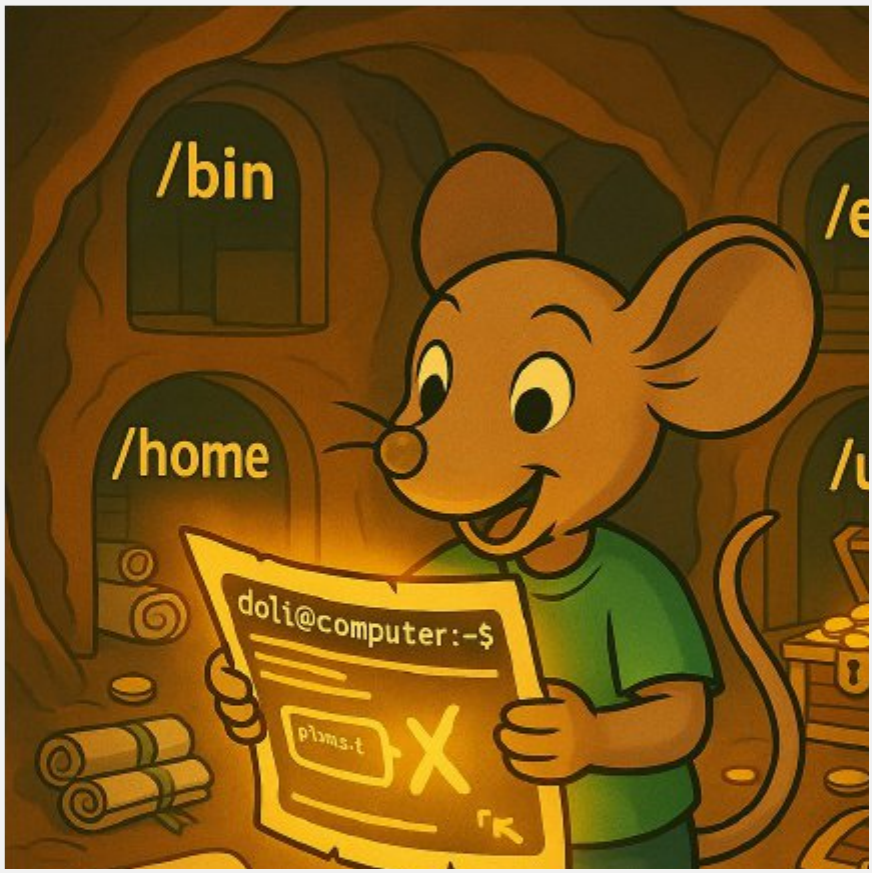
Can you draw a map of where you went?

We said the shell provides everything you need, if not
Already installed you can install,
So knowing where you been is easy and typing

- History
Will the command entered
1. w
2. who
3. whoami
4. uname
5. ls
6. ls -la
7. cd /Downloads/doli

in the shell if you type commands you don't have re-type
the shell remembers and give similar option you may
Want. Don't be afraid of shell, its here to help, the only
to know if something works is to try it.

Doli~in~the~shell



Doli's Challenge: There's a folder called "treasure" hidden somewhere in /home/doli. Can you find it?

End of Chapter 1

In the next chapter, Doli discovers how to make new files, create directories, and even delete things—so get ready to dig deeper into the tunnels of the shell!

Chapter 2:

The shell afterall is nothing more then a program, and a Program is a file in storage or a file which is loaded and Become a program.
We have seen how to deal with files, what to use i.e

Ls
cd
Cat
Mkdir
Cut
Rm

Every program has setting and the shells setting and behave Can be set or found in bashrc. The bashrc file controls how The shell looks, acts, alias basically its a ever growing file Because you keep add and remove always customerising To you liking .

Next we start making or creating files directories,
You will see some of the same commands, most of time you Will be seeing the same commands over and over that way Scripting or automation come, if you see yourself doing The something over and over make sense to script it.

2.1 Doli Builds a New File

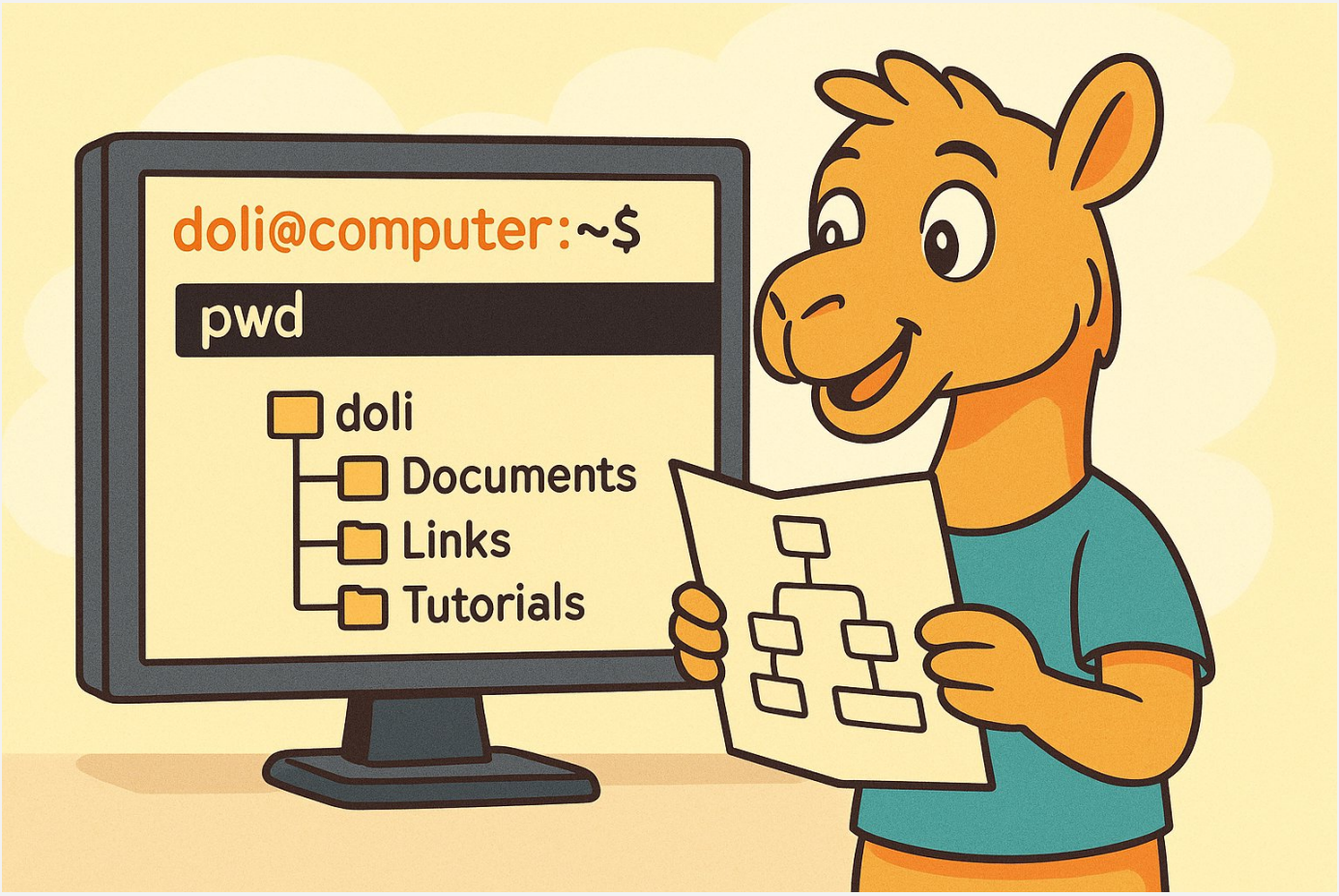
Just like an artist needs a blank canvas, Doli sometimes needs to create new files. These might be notes, scripts, or snack lists.

To create a new, empty file, Doli uses the touch command:

touch snacklist.txt

This tells the computer: "Hey, make a file called nacklist.txt."

Doli's Example:
touch treasure.txt
Creates an empty file named treasure.txt in the current folder.
Visual: Doli gently placing a piece of paper into a drawer labelled Documents.



One rainy afternoon in the terminal, Doli needed a clean slate somewhere to record her next discovery With a quick flick of her fingers, she typed: touch evidence.txt And just like that,

the file appeared. The `touch` command is Doli's quiet assistant: it creates empty with no fuss or questions asked.

Whether she's preparing todo.txt, logfile.log, or mission.md, touch canvas in seconds.

No editors needed. No content required. Just a place holder, ready for action.

But touch is more than just a file maker — it's also a time stamp adjuster. If the file already exists touch doesn't overwrite it or change its contents.

Instead, it updates the last modified time — like that says, "Hey, I've been here recently."
This makes it perfect for scripting or for marking certain files as freshly accessed.

Doli uses this trick when she wants to sort files by activity or keep old files from being flagged as outdated. touch old_script.shDoli smiles. No one will know it was last changed three years ago. Looks like I just edited it!"
In detective work,
even timestamps can be part of the strategy.

2.2 Making a New Directory

Sometimes Doli needs to group files into a folder. For that, she uses the mkdir command (short for "make directory").

```
mkdir recipes
```

This creates a new folder called recipes.

Doli's Tip: Think of this like building a new room in your digital house.

2.3 Moving Into the New Directory

Just like in Chapter 1, Doli can use cd to enter her new folder:

cd recipes

Now any new file she creates will go into recipes.

Comic: Doli steps through a doorway labelled recipes/, carrying a rolling suitcase of text files.

2.4 Writing Into a File

Doli now wants to write a list of her favorite snacks into a file. The simplest way to do this is with the echo command and the redirection arrow >:

echo "Cheese" > snacklist.txt

This creates the file (if it doesn't exist) and writes "Cheese" inside.

Add more snacks:

echo "Berries" » snacklist.txt

The double arrows » mean "add to the file instead of replacing it."

.

2.5 Checking What's Inside

To peek inside a file, Doli uses the cat command (short for "concatenate", but think of it as "catalogue" for now):

cat snacklist.txt

This shows the full content of the file.

Example Output:

Cheese Berries

Doli's Dictionary: cat = View what's in a file like a scroll opening up.

2.6 Moving and Renaming Files

Let's say Doli wants to move her snack list to another folder, or maybe rename it. The mv command can do both!

Rename:

mv snacklist.txt yumlist.txt

Move to another folder:

mv yumlist.txt ../

This moves the file to the folder one level up.

Visual: Doli carrying a box labelled yumlist.txt upstairs with a label change halfway.

2.7 Copying Files

If Doli wants to duplicate a file, she uses cp:

cp yumlist.txt backup.txt

Now she has two files with the same content.

Comic Frame: A clone-bot handing Doli an identical twin of the file she just made.

2.8 Deleting Files and Folders (Carefully!)

Doli is brave, but deleting things makes her nervous. The rm command deletes files.

rm backup.txt

]To delete a folder and everything inside it:

rm -r old_folder

WARNING: Deleted files don't go to a recycle bin. They vanish!

2.9 Doli's File-Quest

Let’s follow Doli on another adventure:

Make a new folder:

mkdir adventure cd adventure

Create a log file:

touch journal.txt

Add an entry:

echo “Found cheese stash under /usr/bin.” > journal.txt

View the log:

cat journal.txt

Move log to safety:

mv journal.txt ../

Go back and copy it:

cd .. cp journal.txt backup_journal.txt

Activity: Recreate Doli's quest in your terminal. Replace “cheese stash” with your own hidden treasure.

2.10 Practice with Doli

Try these tasks:

Create a folder named experiment

Inside it, create a file notes.txt

Write a line into it using echo

Copy it and rename the copy backup.txt

Delete the original file

Doli’s Challenge: Find your home folder. Create a folder called hideout and put a file named secret.txt in it. Can you do it with just the terminal?

End of Chapter 2

Next time, Doli learns how to search for files, connect commands together, and pipe output from one to another. The shell is just beginning to reveal its true powers!

Chapter 2.2:

Doli the Detective – Finding, Linking, and Piping

1. cat file.txt

What it does:

Reads and prints the contents of the file.

“Hey computer, show me everything inside file.txt.”

This is where the data starts.

2. grep keyword

What it does:

Filters the lines from the file and only keeps the ones containing the word keyword.

“Keep only the lines with the word I’m searching for.”

This narrows down the content to the important stuff.

3. sort

What it does:

Arranges the filtered lines alphabetically or numerically.

“Put the matching lines in order — like organizing notes by title.”

4. head -n 10

What it does:

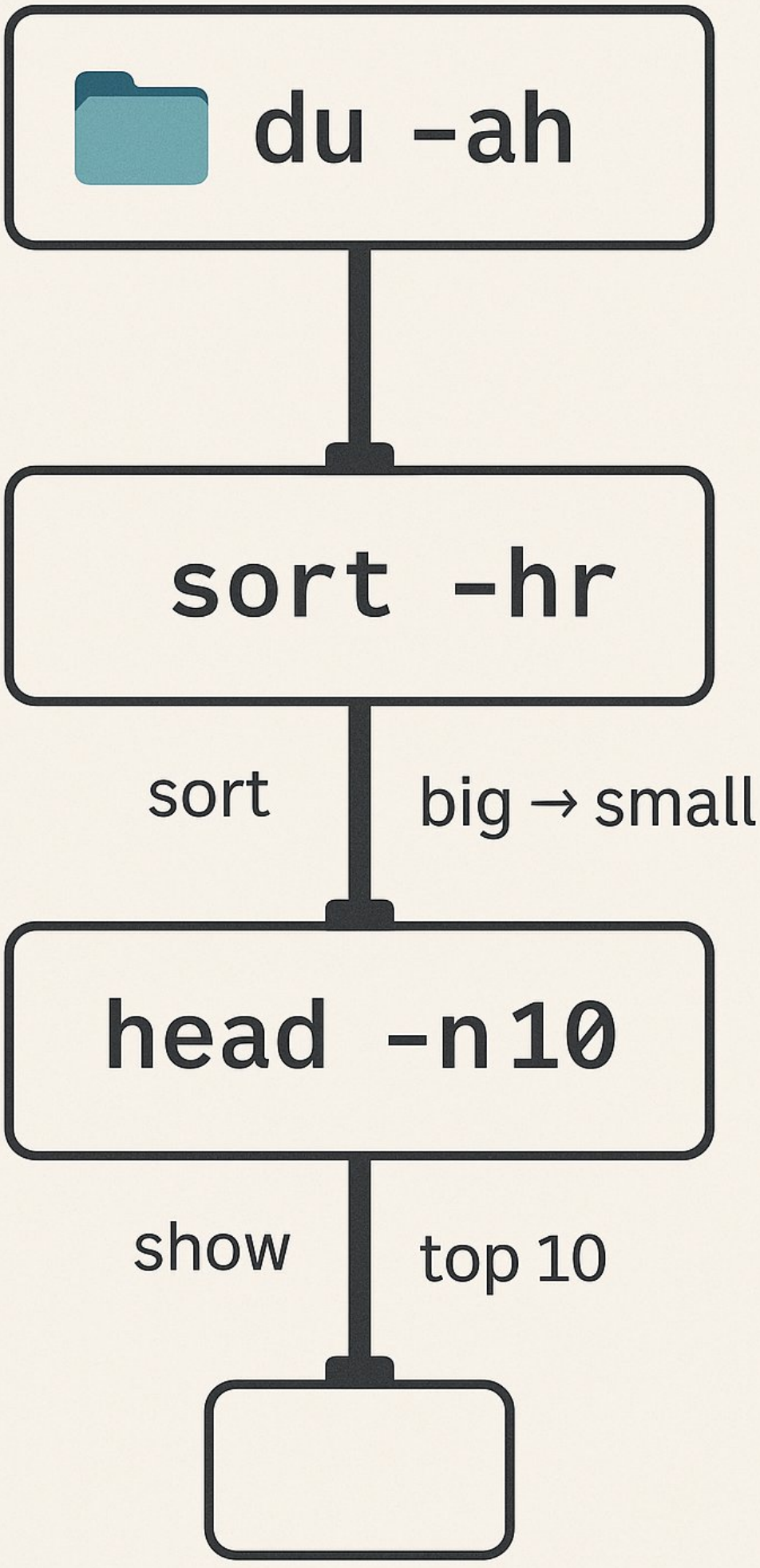
Takes the top 10 lines from the sorted list.

“Just show me the first 10 results — that’s all I need.”

The Big Picture

What’s happening here is a flow of data — passed from one tool to the next without saving anything to a file. It's fast, clean, and extremely flexible.

The pipe (|) is like a conveyor belt:


du -ah
sort -hr
sort
big → small
head -n 10
show
top 10

3.1 Doli Searches for Clues

Sometimes, Doli forgets where she left something. To search for files by name, she uses the find command:

find . -name “cheese.txt”

Explanation:

“start searching from here”

-name tells it to match the name

“cheese.txt” is what we’re looking for

Output Example:

./snacks/cheese.txt

find . -name ".txt"*
Find all files ending with .txt in the current directory and all subfolders.
Absolute Match (case-sensitive)
find /home/doli -name "README"
Find files exactly named README (case-sensitive) in Doli’s home directory.

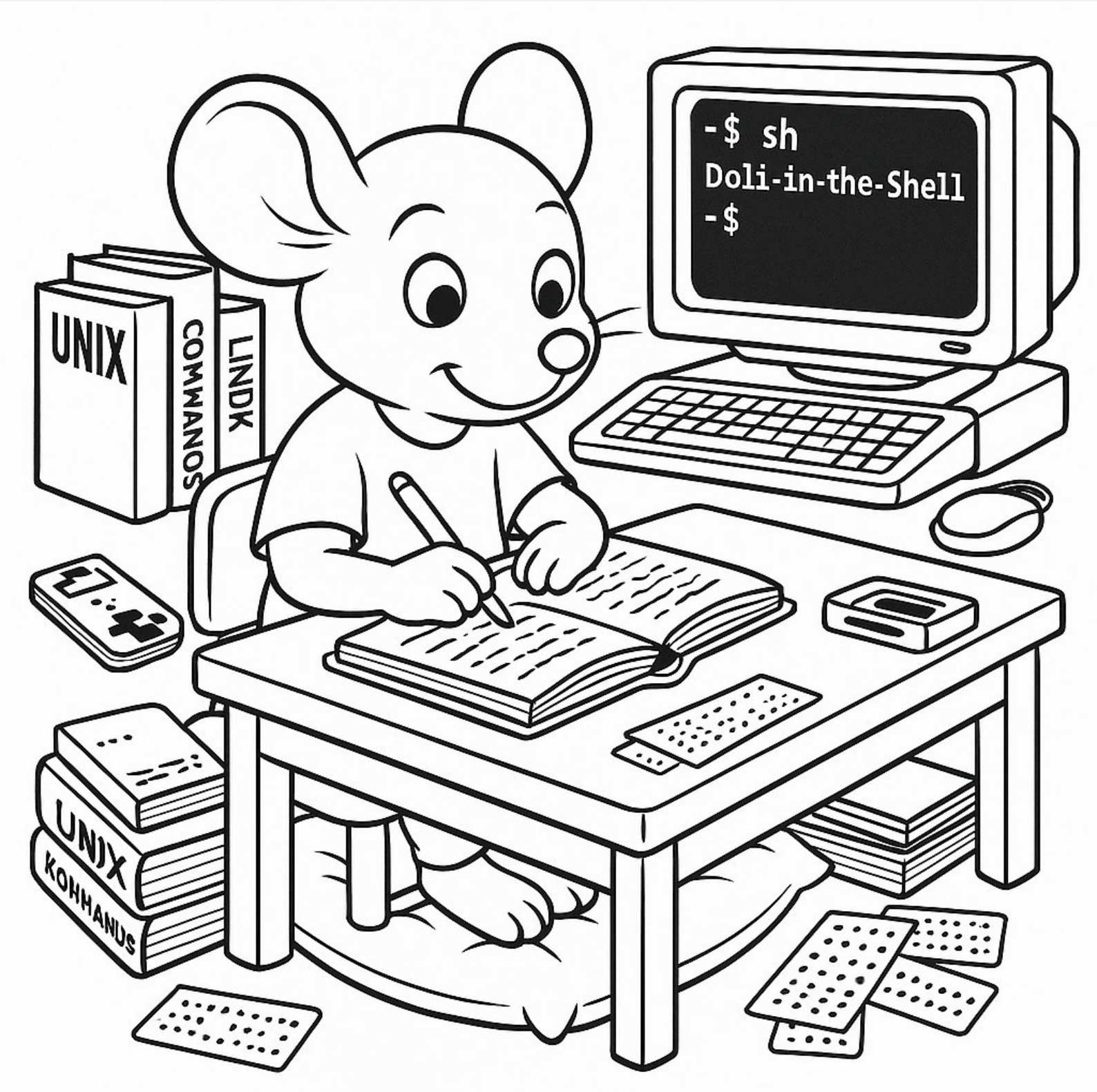
Case-Insensitive Search: -iname
find . -iname ".jpg" Match .jpg, .JPG, .Jpg, etc. Search by Type*

Option	*Meaning*	*Example*
-type f	*Regular file*	*find . -type f -name "*.log"*
-type d	*Directory*	*find . -type d -name "backup"*
-type l	*Symbolic link*	*find . -type l*

Time-Based Searches Option	*Description*	*Example*
-mtime -1	*Modified in the last 24 hours*	*find . -mtime -1*
-mtime +7	*Modified more than 7 days ago*	*find . -mtime +7*
-atime -1	*Accessed in the last 24 hours*	*find . -atime -1*

Size-Based Searches Option	*Description*	*Example*
-size +5M	*Larger than 5 MB*	*find . -size +5M*
-size -10k	*Smaller than 10 KB*	*find . -size -10k*
-size 0	*Empty files*	*find . -size 0*

User-Based Searches find /home -user doli Find all files owned by user doli.
Combine with -exec find . -name ".log" -exec grep "ERROR" {} \;*
For each .log file, run grep to search for the word "ERROR".
Use with xargs for Speed find . -name ".bak" | xargs rm*
Delete all .bak files found.


-$ sh
Doli-in-the-Shell
-$
UNIX
COMMANDS
LINDK
UNIX
KOMMANDS

Doli's Tip: grep is like a super-sifter for words.

Bonus:

grep -r "cheddar" .

Searches recursively through all files from the current directory!

Visual: Doli sifting through a pile of notes with a magnifying glass, pulling out highlighted lines.

3.3 Linking Files (Hard and Soft)

Doli wants to make shortcuts to important files. That's where links come in.

Hard Link:

ln cheese.txt backup.txt

Creates a second name for the same file. Both are equal.

Symbolic Link (Symlink):

ln -s cheese.txt shortcut.txt

Creates a pointer file that "points to" the original.

Comic: Doli tying a red string between two file cabinets.

3.4 Piping: Sending Output from One Command to Another

Doli discovers a magical character: the pipe |

It lets her connect commands like building blocks.

Example:

ls | grep "doc"

This lists items in the folder, then sends them through grep to show only ones with "doc".

Visual: Doli connecting pipes between machines labeled ls and grep with glowing data flowing through.

3.5 Redirection Refresher

Doli reviews her redirection tools:

: Create or overwrite

: Append

: Take input from a file

Example:

echo “Doli is smart” > quote.txt cat < quote.txt

3.6 Practice Quest: Doli’s Secret Recipe

Doli wants to find her special cheese recipe hidden somewhere in /home/doli/recipes.

Step-by-Step:

Search for a file with “cheese” in the name:

find . -name “cheese”

Look inside each one for “secret”:

grep “secret” ./found_file.txt

Create a symlink to it in /home/doli/important/:

ln -s ./found_file.txt ../important/secret_cheese.txt

3.7 Doli’s Challenge

Use what you’ve learned to:

Search for a file in your home directory that contains the word “config”

Make a symlink to it in a new folder called shortcuts

Use cat and | with grep to print only the lines that contain numbers

End of Chapter 3

Next time, Doli learns how to customize her terminal, set aliases, and create her own shell scripts. The tools are building up—soon Doli will be unstoppable!

Chapter 3:

Doli the Detective – Finding, Linking, and Piping

3.1 Doli Searches for Clues with find

Doli once misplaced her map inside her digital burrow. She needed a way to search every corner for it. Enter: the find command.

find . -name "map.txt"

Explanation:

. — Start searching from "right here."

-name — Match files with this name.

"map.txt" — That's what Doli is hunting.

Output Example:

./hidden/notes/map.txt

Try It Yourself:

Create a file:

touch secret_clue.txt

Now search for it:

find . -name "secret_clue.txt"

3.2 Doli Greps Through Text

Doli finds a stack of logs and documents. But which one has the treasure code "cheddar"? She uses grep, her favorite highlighter:

grep "cheddar" snacklist.txt

Bonus: Searching Everywhere

grep -r "cheddar" .

-r means: “Look in every file inside every folder.”

Example Output:

./logs/snacklist.txt:cheddar crackers

Activity:

Create a file with this:

echo “cheddar crackers” > snacklist.txt

Search with grep:

grep “cheddar” snacklist.txt

3.3 Doli Makes Links – Hard and Soft

Sometimes Doli wants one file, but in many places. Like a shortcut tunnel.

Hard Link:

ln snacklist.txt clone.txt

Makes an exact twin. Both share the same data.

Symbolic (Soft) Link:

ln -s snacklist.txt pointer.txt

Creates a shortcut to the original.

Activity:

Create a file:

echo “cheddar crackers” > snacklist.txt

Make a symlink:

ln -s snacklist.txt snacks_link.txt

Open it:

cat snacks_link.txt

Visual: Doli stretching a yarn thread from one file box to another with a label “shortcut!”

3.4 Piping: Connecting Tools with |

Doli finds a magical tube. When she attaches one end to a machine and the other to another, data flows through!

Example:

ls | grep "txt"

ls: list files

send results into…

grep "txt": show only .txt files

Bonus Examples:

cat log.txt | grep "error"

ps aux | grep firefox

Activity:

Make a folder:

mkdir lab && cd lab

Add files:

touch a.txt b.jpg c.txt d.png

Try:

ls | grep ".txt"

3.5 Review of Redirection

Doli remembers her old tools:

overwrites

appends

< reads input

Example:

echo "I love cheese" > message.txt cat < message.txt

Try This:

echo “apple” > list.txt echo “banana” » list.txt cat list.txt

Visual: Doli posting notes into a mailbox with > painted on the side.

113.6 Doli’s Secret Recipe Hunt

Doli hears a rumor: her old “cheese pie” recipe is lost in /home/doli/recipes.

Time to investigate!

Step 1: Search

find /home/doli/recipes -name “cheese”

Step 2: Peek Inside

grep “pie” ./cheddar_tales.txt

Step 3: Save It Safely

ln -s ./cheddar_tales.txt ~/important/cheese_pie.txt

Draw It: Map the path Doli took from her terminal to the recipe.

3.7 Doli’s Detective Practice

Your mission:

Make a folder called case and a file evidence.txt inside.

mkdir case && echo “config A123” > case/evidence.txt

Use grep to find “A123”.

grep “A123” case/evidence.txt

Create a symlink named evidence_link.txt.

ln -s case/evidence.txt evidence_link.txt

Pipe it to a line counter:

cat evidence_link.txt | wc -l

3.8 Doli’s Challenge

Challenge your brain:

Use find to locate a file with “log” in the name.

Use grep to pull lines that contain numbers.

Use | and wc -l to count how many matches you find.

Bonus: Try linking this file to a new folder called shortcuts

End of Chapter 3

Doli has mastered detective skills: searching, linking, and connecting. In the next chapter, Doli will shape her shell, craft aliases, and write her very first shell script. Her powers are growing—line by line!

Chapter 4:

Doli Customizes Her Command Cave – Aliases, Environment, and Script

4.1 Doli Gets Tired of Typing

Doli often types long commands. One day she mutters, “There must be an easier way.” Her friend, the turtle admin, whispers: “Use aliases.”

Aliases: Command Nicknames

alias gs=‘git status’

This tells the shell: When I type gs, pretend I said git status.

View Your Aliases:

alias

Remove an Alias:

unalias gs

Permanent Aliases: Add them to your ~/.bashrc or ~/.zshrc file:

alias update='sudo apt update && sudo apt upgrade'

Visual: Doli posting sticky notes on her wall with shortened command labels.

4.2 Environment Variables: Doli's Room Settings

Every shell session has its own "room" with temperature, lighting, and rules.

These are called environment variables.

View All Environment Variables

env

View One Variable

echo $HOME

Set a Variable Temporarily

MYNAME=Doli export MYNAME

Use It

echo $MYNAME

Set It Permanently

Add it to ~/.bashrc:

export EDITOR=nano

Doli adjusting knobs labeled $PATH, $EDITOR, and $SHELL on a

control panel.

4.3 Doli Makes Her First Script

Writing commands over and over is tiring. So Doli writes her first shell script.

Step 1: Create the File

nano greet.sh

Step 2: Add This Inside:

#!/bin/bash echo "Hello, traveler! This is Doli."

Step 3: Save and Exit (CTRL+O, Enter, CTRL+X)

Step 4: Make It Executable

chmod +x greet.sh

Step 5: Run It

./greet.sh

Doli writing code on parchment and feeding it into a box labeled "script machine."

4.4 Script Variables and Arguments

Doli adds flexibility to her scripts.

Script with a Variable:

#!/bin/bash NAME=$1 echo "Hello, $NAME!"

Save as hello.sh, then run:

./hello.sh CheeseLover

Output:

Hello, CheeseLover!

Doli holding a scroll with $1, $2, $@ written on it, like spell runes.

4.5 A Useful Script: Daily Cleanup

Doli writes a cleanup script for her Downloads folder.

#!/bin/bash echo "Cleaning Downloads..." rm ~/Downloads/.tmp rm ~/Downloads/.bak

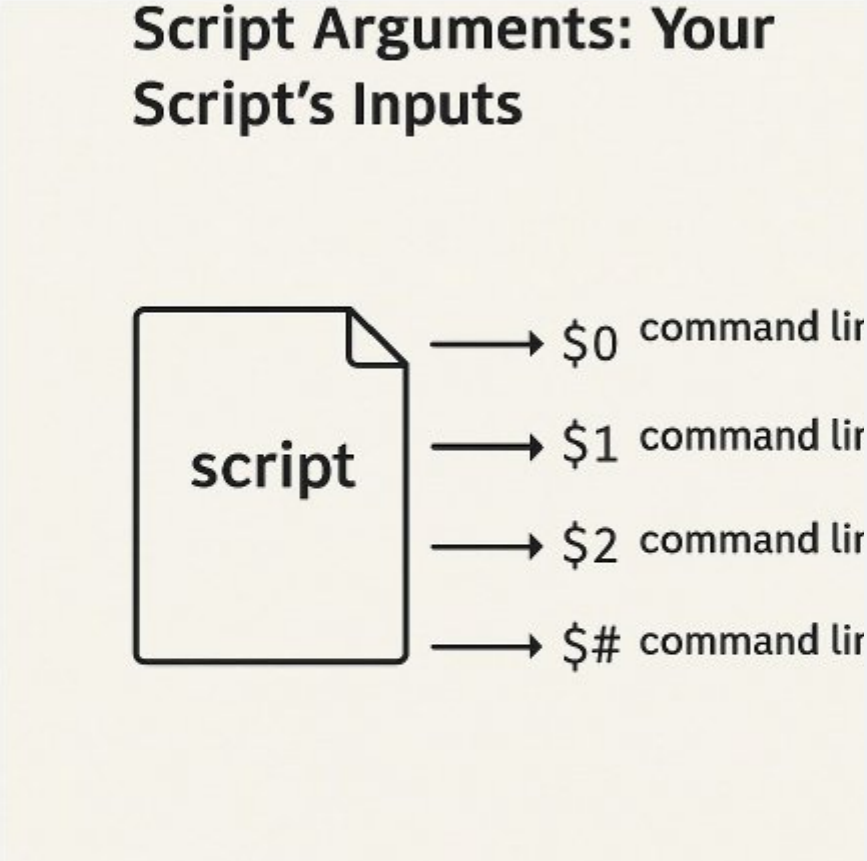
Place in ~/scripts/cleanup.sh and run daily with:

~/scripts/cleanup.sh

Add to .bashrc:

alias cleanup="~/scripts/cleanup.sh"

14Activity: Write a script to backup your Documents folder to a folder called Backups.



4.6 Doli's Shell Startup

When Doli opens her terminal, it reads a secret file first. This file can:

Show a custom greeting

Set up aliases

Define environment variables

File: ~/.bashrc (or ~/.zshrc)

Add at the bottom:

echo "Welcome, Doli! Ready to hack the filesystem."

Then reload it:

source ~/.bashrc

A magic spellbook titled .bashrc glowing as the terminal boots.

4.7 Practice Zone: Doli's Terminal Makeover

Try these tasks:

Set an alias for ls -lah called ll

Set an environment variable FAV_SNACK="cheese"

Write a script that:

Prints your name

Shows today's date

Lists all .txt files in your home

Save it as myinfo.sh and run it!

4.8 Doli's Challenge: Automator Apprentice

Your mission:

Create a script that creates a folder called daily with today's date as its name.

Inside it, create a file named log.txt with the current time.

Bonus:

Add the script to your ~/.bashrc so it runs each time you open the terminal.

Hint:

TODAY=(+

/ /TODAY/log.txt / /TODAY echo "Opened at ( )" >

End of Chapter 4

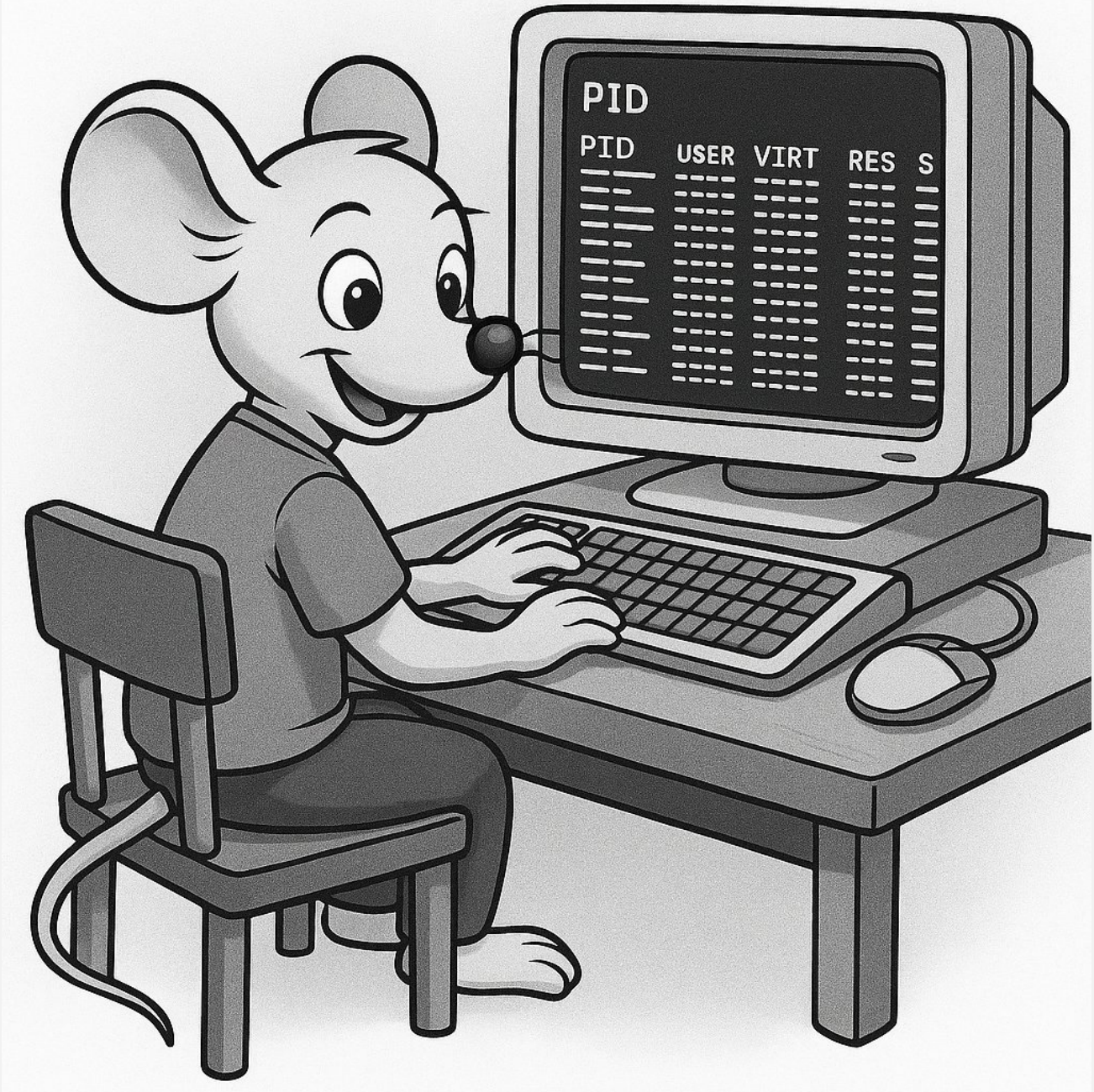
Now Doli has her own customized command cave! With aliases, environment controls, and scripts, she's saving time and gaining power. In the next chapter, Doli will venture into processes, background jobs, and system monitoring. Her journey continues—automated and effectiness

Chapter 5:

Doli Tackles Time and Tasks – Processes, Jobs, and Monitoring

ps	*Shows a snapshot of current processes*
top / htop	*Live view of running processes and resource usage*
jobs	*Lists background jobs in the current shell*
&	*Runs a command in the background*
fg / bg	*Brings a job to foreground / background*
kill	*Sends a signal to stop (or manage) a process*
killall	*Kills all processes with a given name*
nice / renice	*Adjusts process priority*
watch	*Repeats a command every few seconds (monitoring)*
time	*Measures how long a command takes*

top



5.1 What Are Processes?

Every program Doli runs becomes a process. A process is like a living creature

in her system—it has:

A name (command)

A unique ID (PID)

A parent (who started it)

A job to do (running a script, program, or command)

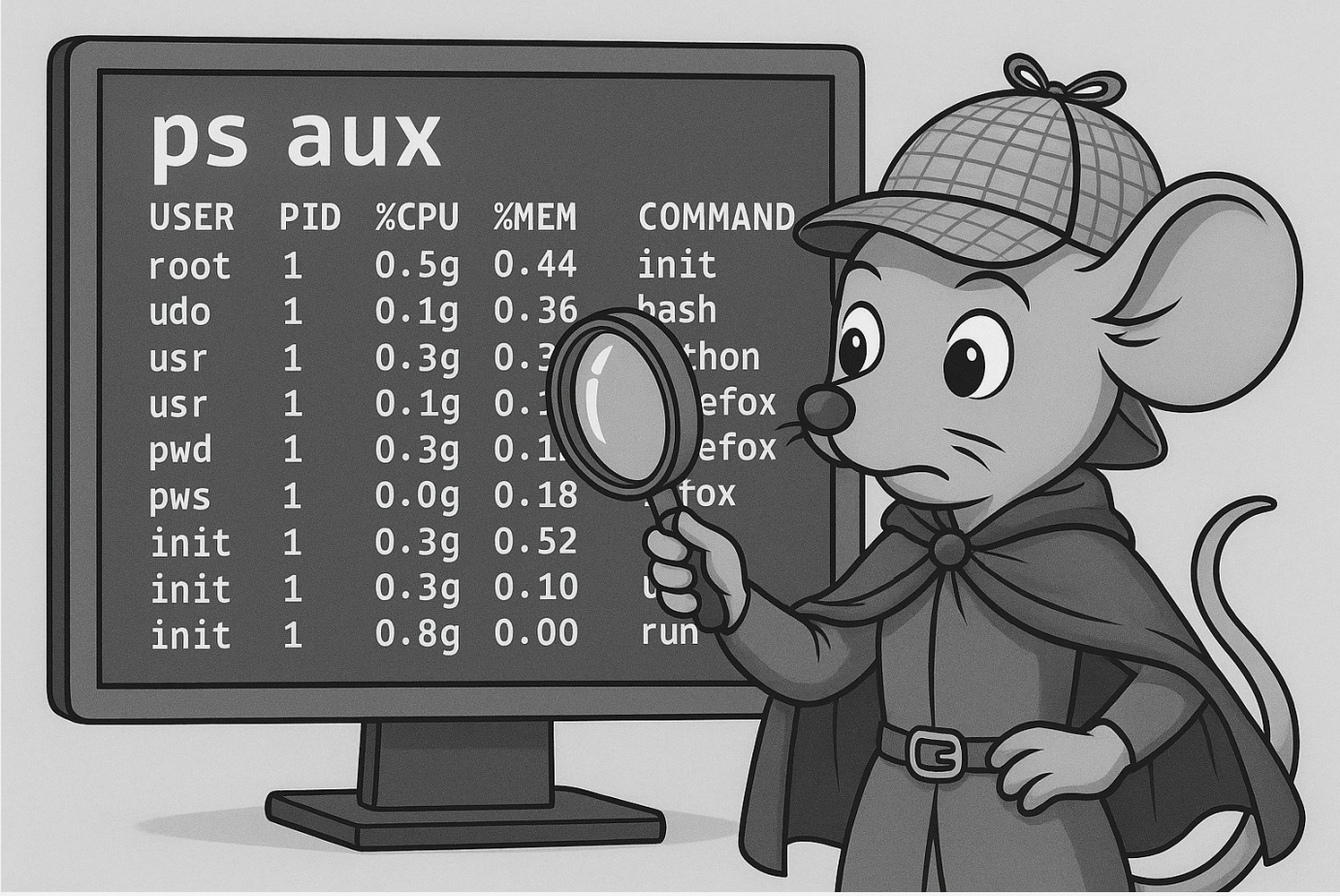
Example:

echo “Hello from Doli!”

This command becomes a temporary process that runs and exits.

To list running processes:

ps aux



each wearing name tags like “bash”, “nano”, “firefox”.

5.2 Foreground vs Background Tasks

When Doli runs a command, it usually runs in the foreground—she waits until it finishes.

But what if she wants to do other things while it runs?

She can send it to the background by adding an &:

sleep 10 &

This sleeps for 10 seconds, but Doli can keep typing.

To see background jobs:

jobs

To bring a job back to the foreground:

fg %1

task.”

5.3 Stopping, Suspending, and Killing Tasks

Doli sometimes needs to pause or stop misbehaving tasks.

Suspend (pause) a running task:

Press Ctrl+Z

Resume it in background:

bg

Bring back to foreground:

fg

Kill (terminate) a process:

kill

Find PID with:

ps aux | grep your_command

Example:

kill 12345

Visual: Doli waving a red flag and clicking a big red STOP button.

5.4 Watching Your System with top and htop

Doli wants to see the health of her system—what's taking up memory? CPU?

Run:

top

It shows:

CPU usage

Memory usage

Process list (live)

Press q to quit.

If installed:

htop

It's a colorful, user-friendly version!

17Visual: Doli at a spaceship-style dashboard, monitoring CPU and memory

charts in real time.

5.5 Scheduling Tasks with cron

Doli wants her scripts to run automatically at certain times.

Step 1: Edit your crontab:

crontab -e

Step 2: Add a job:

*0 8 * * * /home/doli/scripts/backup.sh*

This runs the backup script every day at 8:00 AM.

Cron Time Format:

||||

| | | +—– day of week (0 - 7)

| | +———- month (1 - 12)

| +——— day of month (1 - 31)

+———– hour (0 - 23)

+————— minute (0 - 59)

Visual: Doli setting a giant alarm clock next to a script scroll.

5.6 Doli's Practice Pad: Process Gym

Try these exercises:

Run a command in the background:

sleep 30 &

Check it with:

jobs

Suspend and resume a task:

nano

Press Ctrl+Z, then type:

bg

Monitor the system:

top

5.7 Doli's Challenge: System Ninja

Your mission:

18Write a script that logs your system's memory usage every hour.

Use cron to schedule it.

Sample script log_mem.sh:

#!/bin/bash echo "$(date): $(free -h | grep Mem)" » ~/memlog.txt

Add to cron:

*0 * * * * /home/doli/log_mem.sh*

Bonus: Use grep and tail to analyze it later!

End of Chapter 5

Doli is now a multitasking master! She can juggle jobs, tame wild processes, and automate like a pro. In the next chapter, she'll explore the filesystem deeply—permissions, ownerships, and making things secure. The command line has never been more alive!

Final Recap: What Doli Has Learned So Far

Let's pause and help Doli—and you—remember everything we've explored:

Chapter 1 – Doli discovered what the shell is and how to move around with pwd, ls, and cd. She learned that the terminal is her voice in the digital world.

Chapter 2 – Doli learned how to create and manage files and folders using touch, mkdir, rm, and mv. She became a builder and an organizer.

Chapter 3 – Doli searched her world with find, scanned documents with grep, and linked files with ln. She learned to pipe commands and combine tools like a real problem solver.

Chapter 4 – Doli customized her shell environment with aliases and variables. She began scripting and building her own command tools.

Chapter 5 – Doli mastered multitasking by learning about processes, foreground/background jobs, and automation with cron. Now she can see her system's heartbeat and make things happen on schedule.

Visual: A giant wall mural showing Doli's journey from explorer to script-writing system controller.

Conclusion: Doli's Shell Adventure – From Fear to Freedom

When Doli first stepped into the shell, it looked scary. There were no buttons, no icons, just a blinking cursor. But step by step, she learned to speak its language.

Now she moves with confidence, building files, navigating the filesystem, running scripts, managing tasks, and customizing her command world.

This journey isn't just about Linux commands. It's about learning to think clearly, solve problems, and stay curious.

19If you're someone who's ever thought:

"I'm not technical." "This is too complicated." "The terminal is not for me."

Then Doli is here to say: Yes, it is for you.

The shell is like a magical storybook. The more pages you turn, the more control and creativity you unlock.

Even if you have ADHD, dyslexia, or you're just easily distracted, remember:

You don't have to learn everything at once.

You can explore in tiny steps.

And most importantly: It's okay to ask questions and make mistakes.

Final Visual: Doli looking up at a vast digital sky, filled with stars shaped like commands, scripts, folders, and processes. She smiles, because now she knows how to reach them.

This isn't the end.

This is just your beginning.

20

Index

Index