Shivam Goyal, L1M3, Presentation

Agenda:

- DevOps
- CI/CD:
  - CI
  - CD
  - Deployment Strategies
- Linux: Overview
- Shell Script task

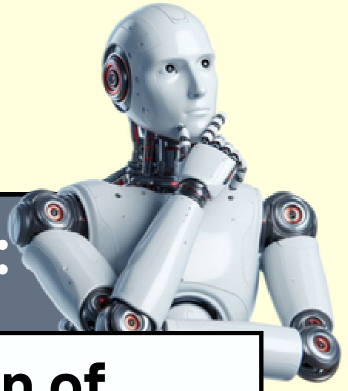
DevOps and it’s benefits:

Definition:

DevOps is a set of practices that combines software development (Dev) and IT operations (Ops) to shorten the development lifecycle and deliver high-quality software continuously.

Promotes Collaboration:

Promotes collaboration and communication between development, operations, and other stakeholders throughout the software development process.



Involves Automation:

Emphasizes automation of repetitive tasks such as testing, integration, and deployment, which helps improve efficiency and reduce human error.

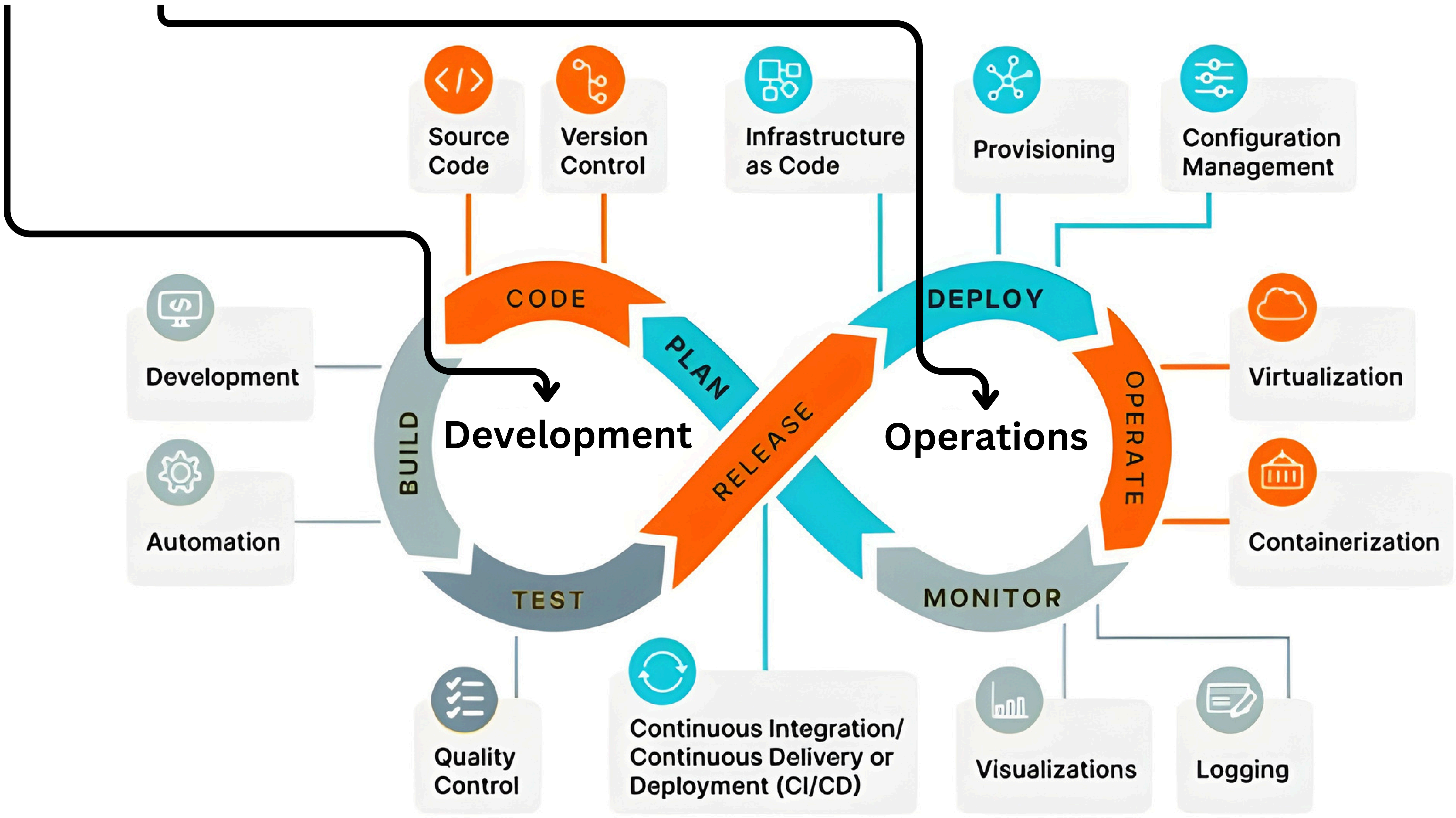
Implements CI/CD:

Involves practices that enable frequent code changes to be automatically tested and deployed, allowing for rapid release cycles.

IaC

Treats infrastructure configuration and management in a similar way to application code, enabling automated and consistent environment setups.

DevOps Lifecycle:



Continues Integration

Definition:

- CI is the practice of merging code changes into a central repository frequently, often multiple times a day. Each integration is then verified by an automated build and automated tests.

Purpose:

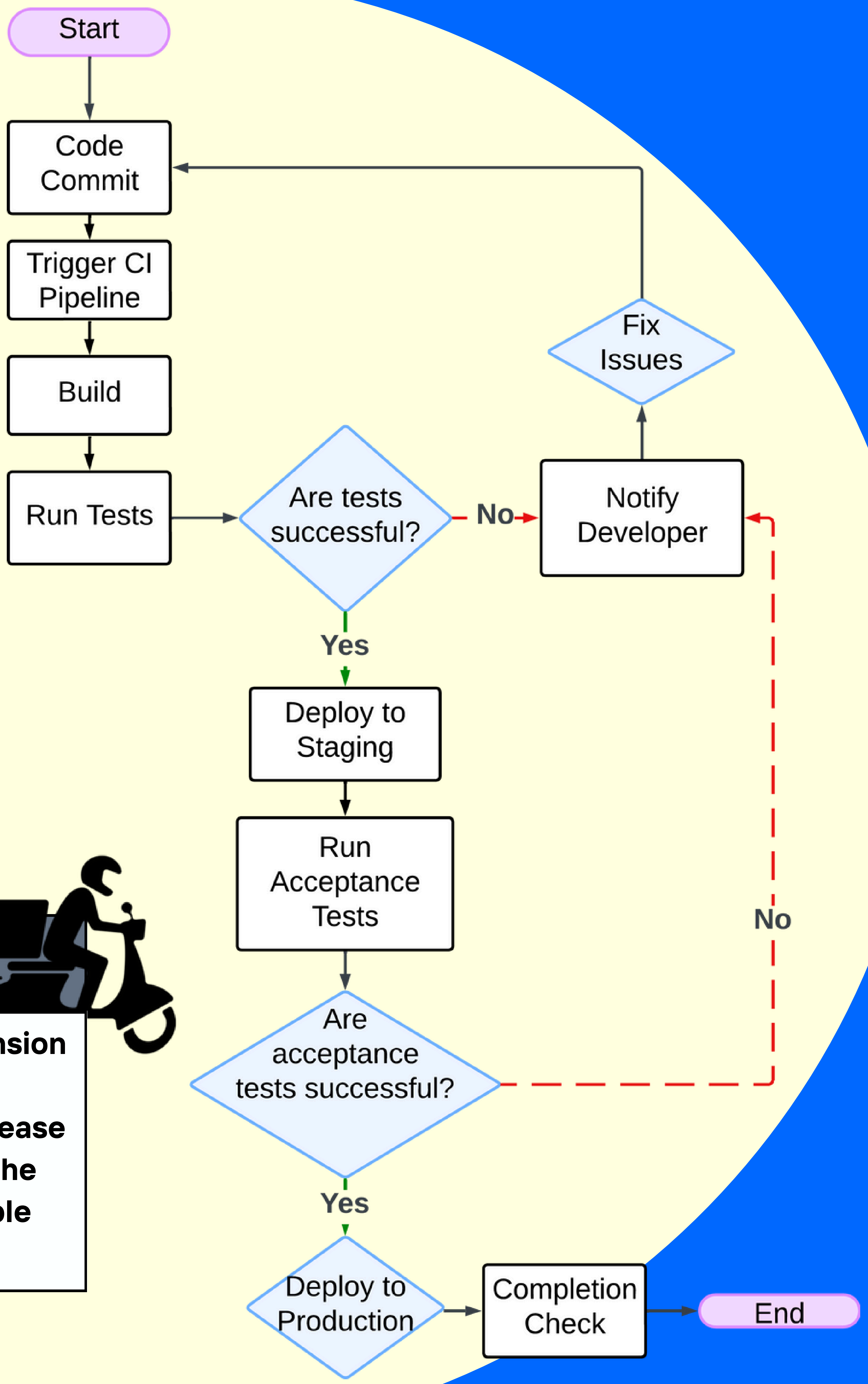
- The goal of CI is to detect and fix integration issues early, improve software quality, and reduce the time it takes to release new software updates.

Key Activities:

- Automated testing: Running unit tests, integration tests, and other checks to ensure code changes do not introduce new bugs.
- Build automation: Compiling code and creating executable files or packages.

Continuous Delivery:

Continuous Delivery is the extension of CI, where code changes are automatically prepared for a release to production. This means that the software is always in a deployable state.



Continuous Deployment :

Definition:

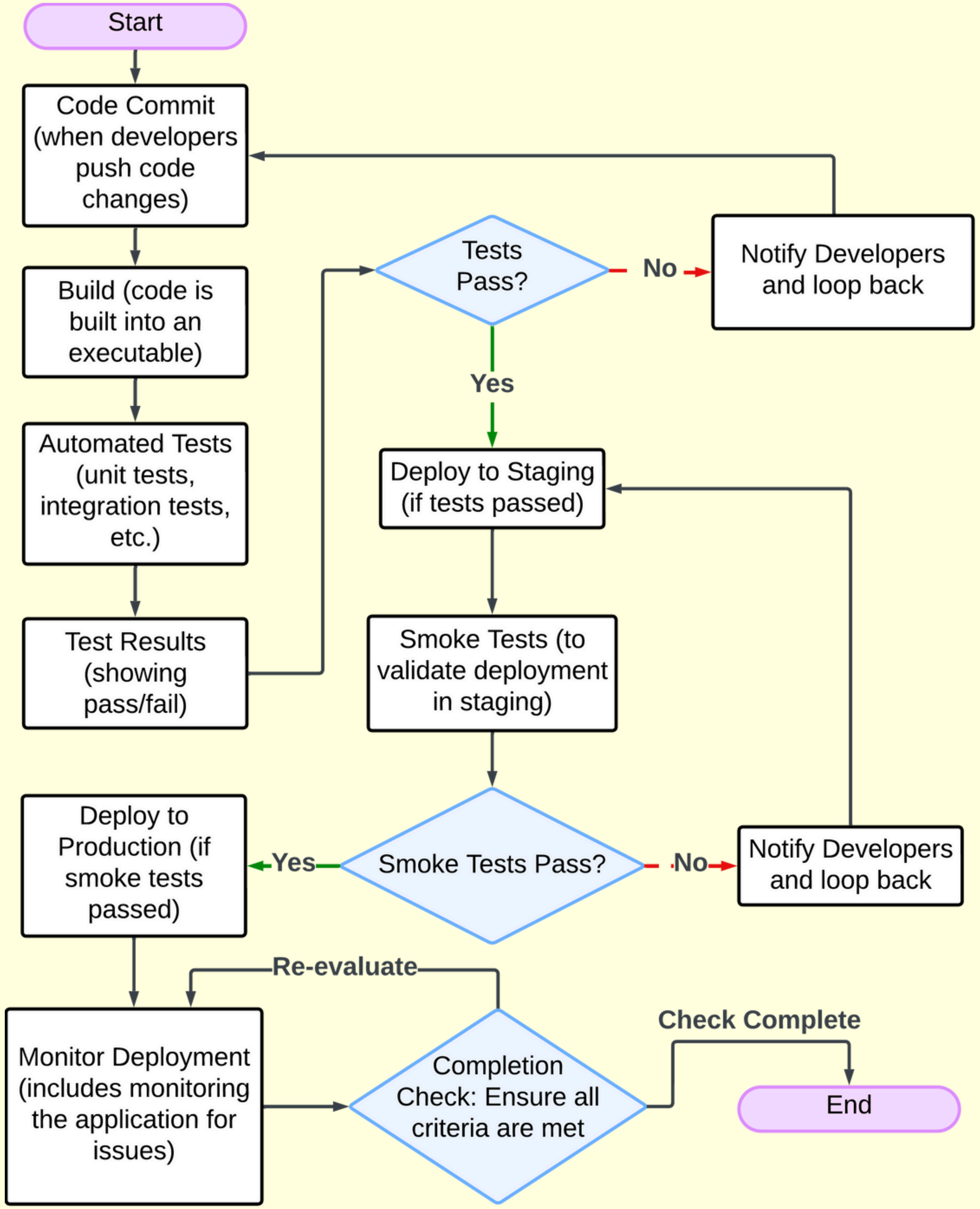
- Continuous Deployment is an extension of Continuous Delivery, where every change that passes all tests is automatically deployed to production without manual intervention.

Purpose:

- The goal is to shorten the release cycle and deliver features to users as soon as they are ready.

Key Activities:

- Automated deployment: Code changes are automatically deployed to a staging environment for further testing.
- Full automation: Every successful change goes straight to production, making the process seamless and efficient..


Start
Code Commit (when developers push code changes)
Build (code is built into an executable)
Automated Tests (unit tests, integration tests, etc.)
Test Results (showing pass/fail)
Tests Pass?
No
Notify Developers and loop back
Yes
Deploy to Staging (if tests passed)
Smoke Tests (to validate deployment in staging)
Smoke Tests Pass?
No
Notify Developers and loop back
Yes
Deploy to Production (if smoke tests passed)
Monitor Deployment (includes monitoring the application for issues)
Re-evaluate
Completion Check: Ensure all criteria are met
Check Complete
End

Infrastructure as code:

Definition:

- Infrastructure as Code (IaC) is a key DevOps practice where infrastructure is provisioned and managed using code and automation, instead of manual configuration. With IaC, you write scripts or configuration files that define and manage servers, networking, databases, and other infrastructure components. This allows you to automate the setup, configuration, and management of infrastructure, ensuring consistency, repeatability, and scalability.

Declarative vs. Imperative:

- Declarative: You define the desired state of your infrastructure, and the tool (e.g., Terraform, CloudFormation) takes care of making sure the infrastructure matches that state.
- Imperative: You specify step-by-step instructions on how to achieve the desired state.

Version Control

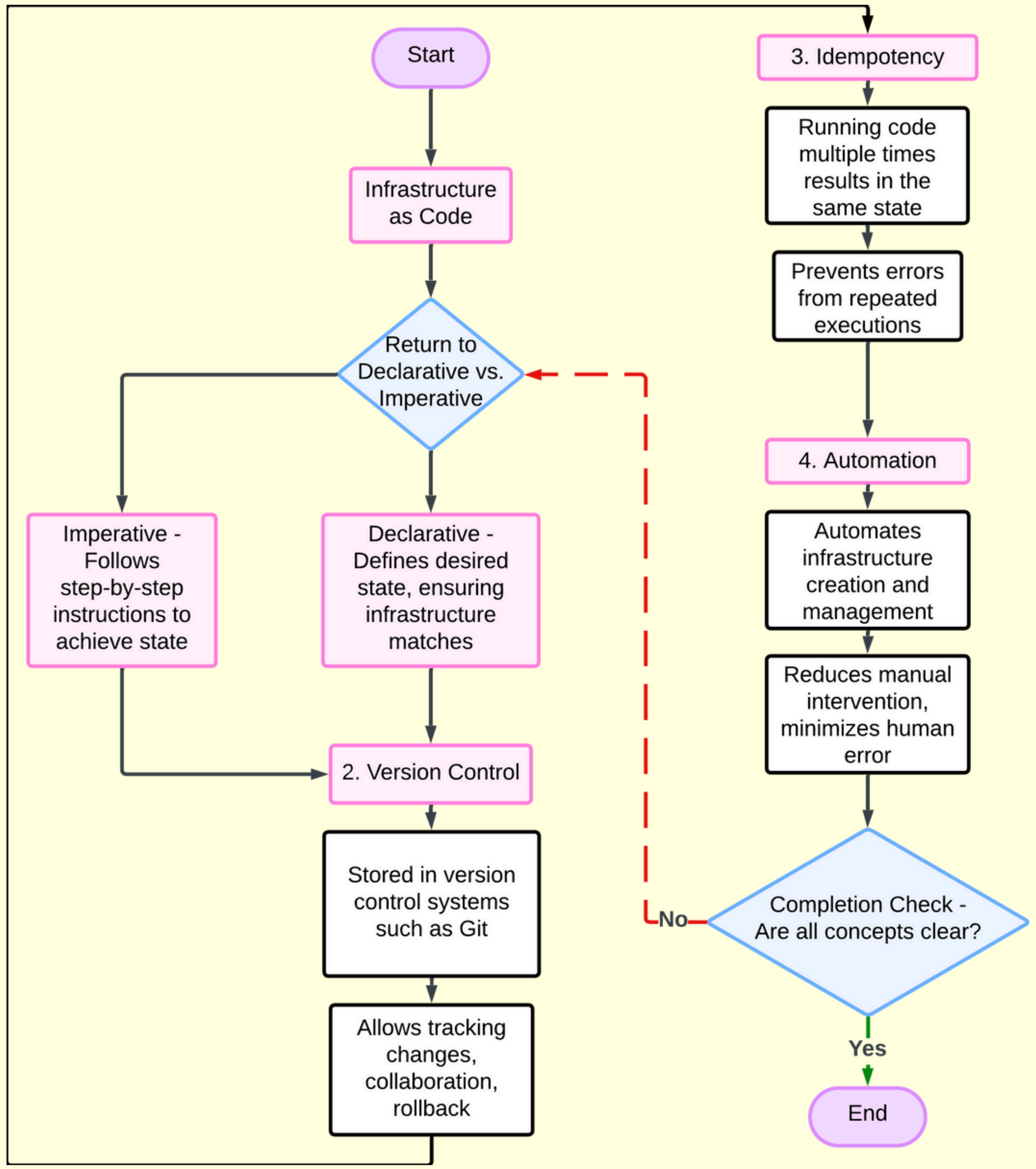
- Since IaC is written in code (often as configuration files), it can be stored in version control systems (e.g., Git). This allows for tracking changes, collaboration, and rollbacks to previous versions.

Idempotency:

- Running the same code multiple times should result in the same infrastructure state, preventing errors from repeated executions.

Automation:

- IaC allows for the automatic creation and management of infrastructure, reducing the need for manual intervention and minimizing human error.


Start
Infrastructure as Code
Return to Declarative vs. Imperative
Imperative - Follows step-by-step instructions to achieve state
Declarative - Defines desired state, ensuring infrastructure matches
2. Version Control
Stored in version control systems such as Git
Allows tracking changes, collaboration, rollback
3. Idempotency
Running code multiple times results in the same state
Prevents errors from repeated executions
4. Automation
Automates infrastructure creation and management
Reduces manual intervention, minimizes human error
Completion Check - Are all concepts clear?
No
Yes
End

Common types of testing in CI/CD:

Unit Testing:

Tests individual components or functions in isolation.
Typically automated and run quickly to catch basic errors early in the development process

Integration Testing:

Tests the interactions between different components or systems.
Focuses on the interfaces and flow of data between modules to ensure they work together as expected.

Functional Testing:

Verifies that the application behaves according to specified requirements.
Often includes end-to-end scenarios that reflect user interactions.

Smoke Testing:

A preliminary test to check the basic functionality of an application.
Ensures that the most critical features work and that the build is stable enough for further testing.

Regression Testing

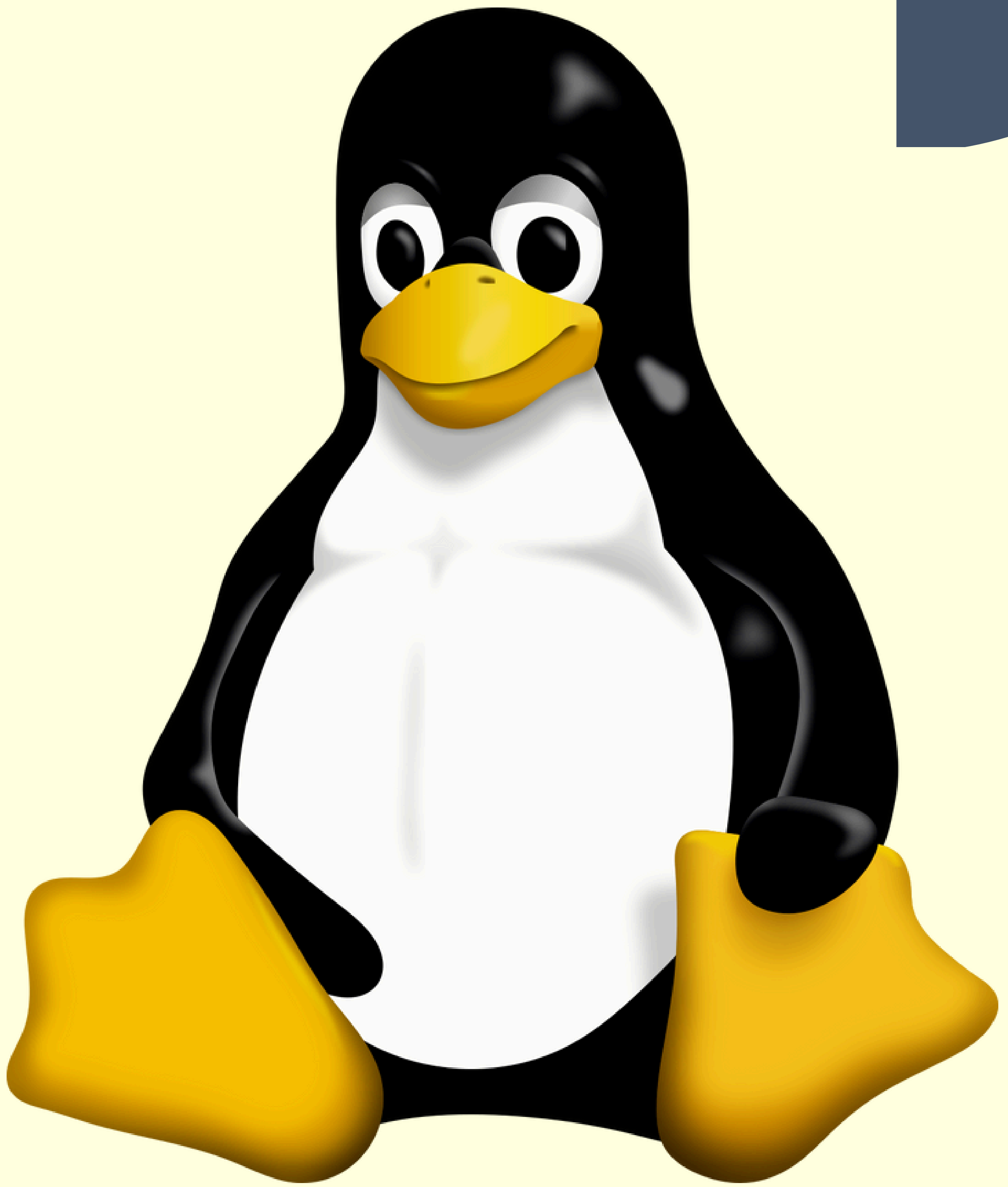
Ensures that recent changes to the codebase haven't adversely affected existing features.
Can be automated to run frequently, especially after each integration.

Deployment Strategies:

Recreate	Ramped	Blue/Green	Canary	A/B testing	Shadow
Version A is terminated then version B is rolled out.	(also known as rolling-update or incremental): Version B is slowly rolled out and replacing version A.	Version B is released alongside version A, then the traffic is switched to version B.	Version B is released to a subset of users, then proceed to a full rollout.	Version B is released to a subset of users under specific condition.	Version B receives real-world traffic alongside version A and doesn’t impact the response
• **Easy to setup.** • **Application state entirely renewed**	• **Easy to set up.** • **Version is slowly released across instances.** • **Convenient for stateful applications that can handle rebalancing of the data.**	• **Instant rollout/rollback.** • **Avoid versioning issue, the entire application state is changed in one go.**	• **Version released for a subset of users.** • **Convenient for error rate and performance monitoring.** • **Fast rollback.**	• **Several versions run in parallel.** • **Full control over the traffic distribution.**	• **Performance testing of the application with production traffic.** • **No impact on the user.** • **No rollout until the stability and performance of the application meet the requirements.**
• **High impact on the user, expect downtime that depends on both shutdown and boot duration of the application.**	• **Rollout/rollback can take time.** • **Supporting multiple APIs is hard.** • **No control over traffic.**	• **Expensive as it requires double the resources.** • **Proper test of the entire platform should be done before releasing to production.** • **Handling stateful applications can be hard**	• **Slow rollout**	• **Requires intelligent load balancer.** • **Hard to troubleshoot errors for a given session, distributed tracing becomes mandatory**	• **Expensive as it requires double the resources.** • **Not a true user test and can be misleading.** • **Complex to setup.** • **Requires mocking service for certain cases.**

Linux:

Linux is a family of open-source Unix-like operating systems based on the Linux kernel, an operating system kernel first released on September 17, 1991, by Linus Torvalds. Linux is typically packaged as a Linux distribution (distro), which includes the kernel and supporting system software and libraries—most of which are provided by third parties—to create a complete operating system, designed as a clone of Unix and released under the copyleft GPL license.



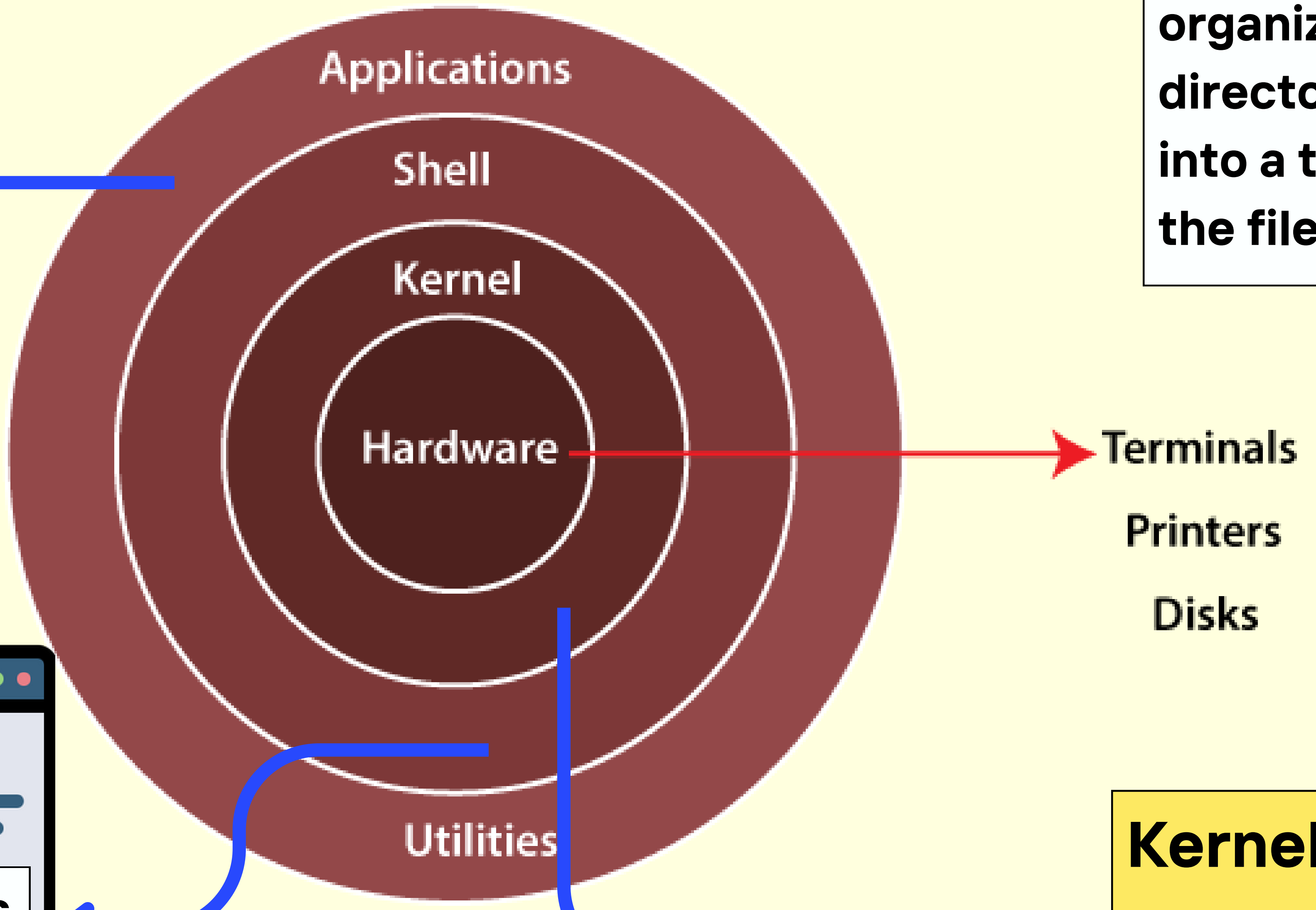
Components of Linux:

Commands and Utilities:

There are various commands and utilities which you can make use of in your day to day activities. ftp, ssh, cp, mv, cat and grep, etc.

Files and Directories:

All the data of Unix is organized into files. All files are then organized into directories. These directories are further organized into a tree-like structure called the filesystem.



Shell:

The shell is the utility that processes your requests. When you type in a command at your terminal, the shell interprets the command and calls the program that you want.

Kernel:

The kernel is the heart of the Linux operating system. It interacts with the hardware and most of the tasks like memory management, task scheduling and file management.

Basic Linux Commands

File and Directory Management Commands:

- ls - List directory contents.
- cd - Change the current directory.
- pwd - Print the current working directory.
- mkdir - Create a new directory.
- rmdir - Remove an empty directory.
- rm - Remove files or directories.
- cp - Copy files and directories.
- mv - Move or rename files and directories.
- touch - Create an empty file or update the timestamp of a file

File Permissions and Ownership Commands

- chmod - Change the permissions of a file or directory.
- chown - Change the owner of a file or directory.
- chgrp - Change the group ownership of a file or directory.
- umask - Set default permission for new files and directories.
- getfacl - Get file access control lists.
- setfacl - Set file access control lists

System Information and Management Commands

- top - Display running processes and system resource usage.
- ps - Report a snapshot of current processes.
- df - Display disk space usage.
- du - Display directory/file space usage.
- free - Display memory usage.
- uname - Display system information.
- who - Show who is logged in.
- uptime - Show how long the system has been running.

Networking Commands

- ping - Check the network connection to a host.
- ifconfig / ip - Configure network interfaces (use ip for modern systems).
- netstat - Display network connections, routing tables, and interface statistics.
- curl - Transfer data from or to a server using various protocols.
- wget - Download files from the web.
- ssh - Securely connect to a remote machine.
- scp - Securely copy files between hosts.
- traceroute - Trace the route packets take to a network host.

Custom Linux shell script:

Important!!

How the script is mainly made

- This is a custom shell script written in bash.
- I’ve mainly used “Cases” to implement the task and also each step has a full proof error handling like the command must follow the proper syntax or a error will be thrown.
- Manual of this also written in Groff and added to environmental variable to access the manual globally.

Creating and viewing users

- **user create <username>** : Creates new user with particular credential
- **user list** : Lists all the regular users present of the server
- **user list --sudo-only :** Show users with sudo access

CPU and Memory Commands

- **cpu getinfo** : shows cpu information
- **memory getinfo** : Shows Memory information

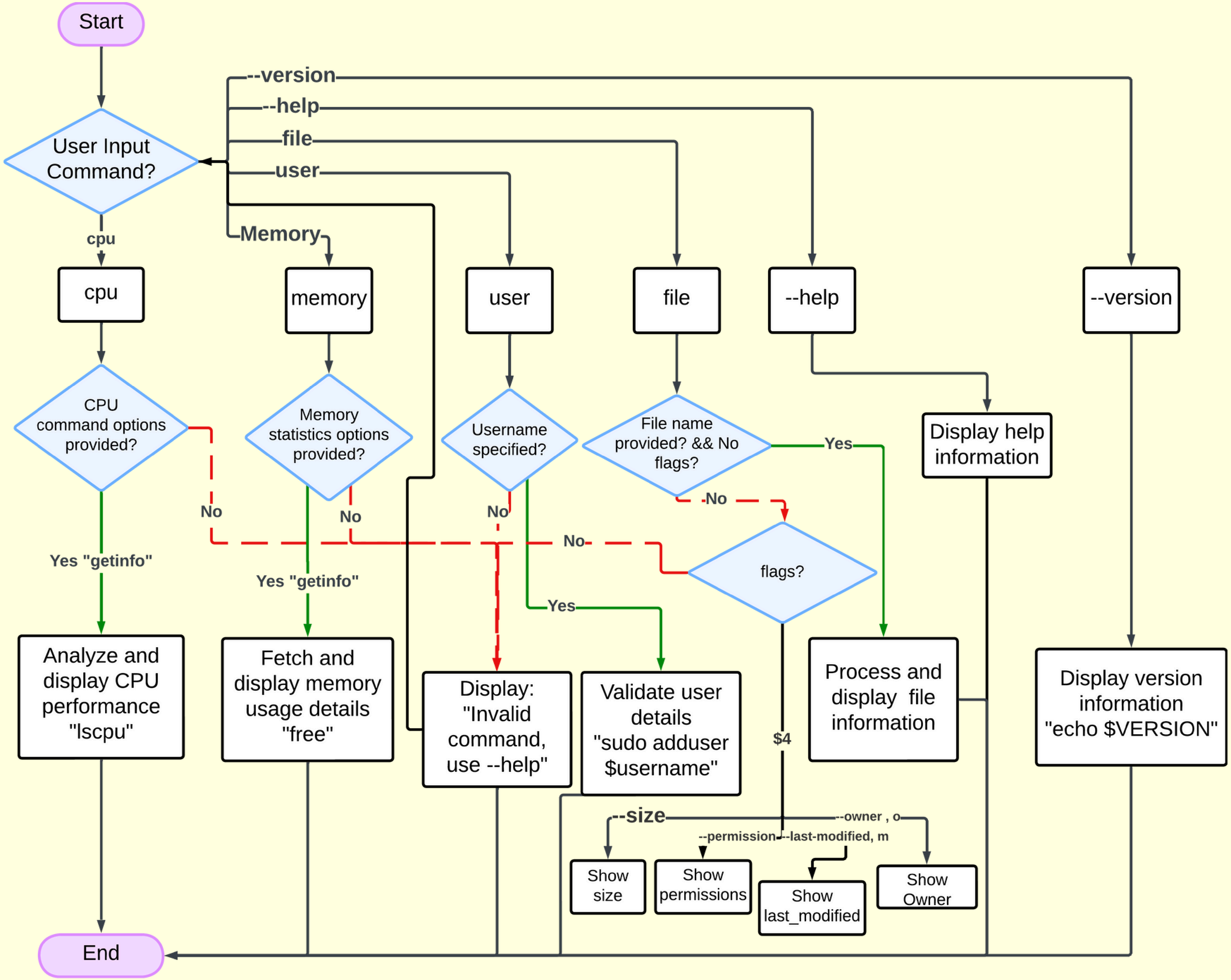
Documentation Features

- **--help** : Shows the help page
- **--version** : Shows the current version of the script

Documentation Features

- **file getinfo <file-name>** : Lists information about the file access, size, owner, last modified
- **--size** : for file size
- **--owner** : to know the owner of the file
- **--last-modified** : to know when the file was last modified
- **--permissions** : to show permissions on a file

Workflow of the internsctl script:


Start
User Input Command?
--version
--help
file
user
Memory
cpu
cpu
memory
user
file
--help
--version
CPU command options provided?
Memory statistics options provided?
Username specified?
File name provided? && No flags?
Yes
No
Display help information
No
No
No
No
No
flags?
Yes "getinfo"
Yes "getinfo"
Yes
Analyze and display CPU performance "lscpu"
Fetch and display memory usage details "free"
Display: "Invalid command, use --help"
Validate user details "sudo adduser $username"
$4
Process and display file information
Display version information "echo $VERSION"
--size
--owner , o
--permission
--last-modified, m
Show size
Show permissions
Show last_modified
Show Owner
End

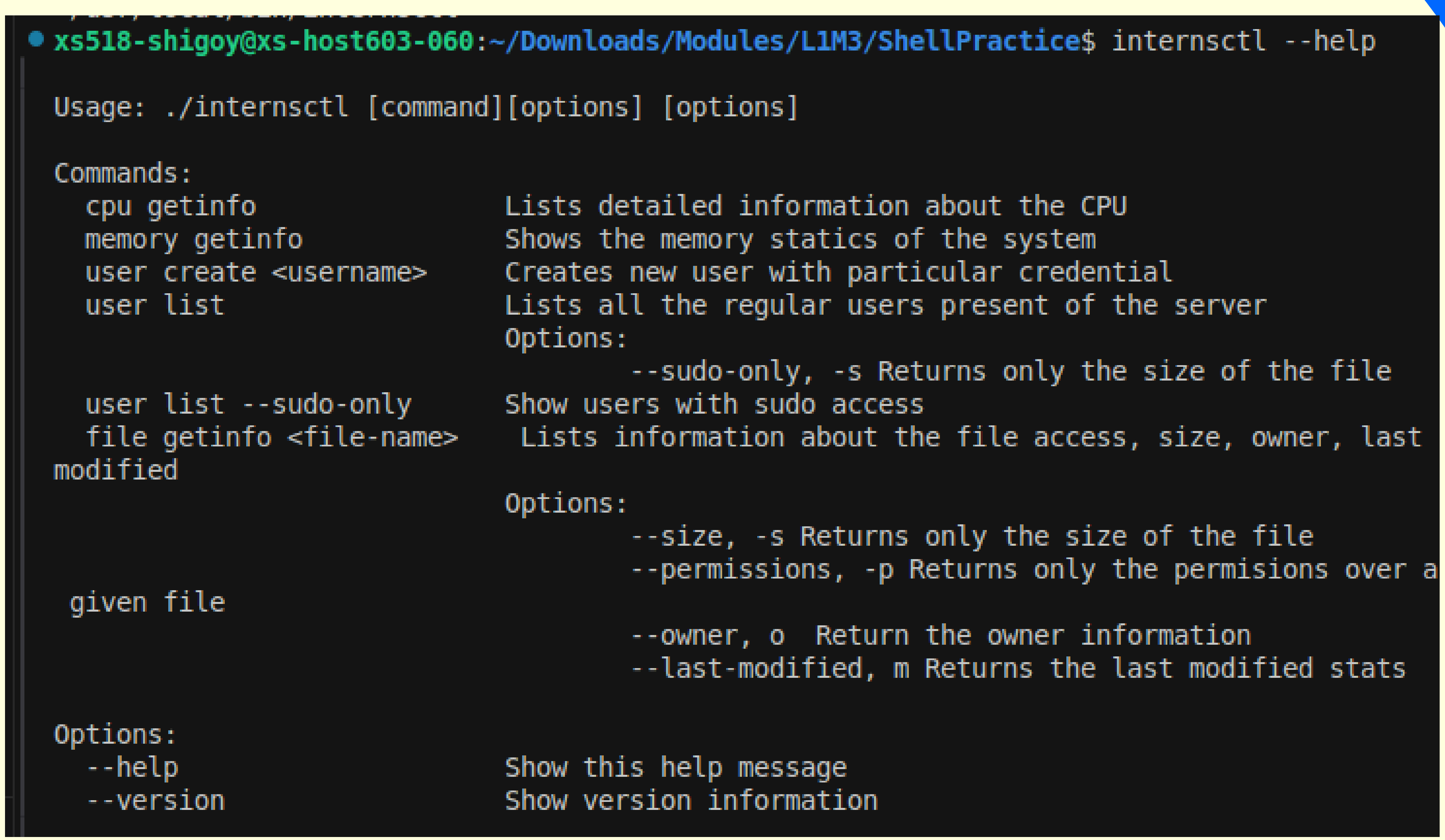
Working of --help:

Basically how this is working is

1) Check the first parameter i.e. “$1” is <--help>

2) if Case --help) == true, invoke the show_help() function which cat echo this text

3) if Case “) show use --help message (also if $1 is any invalid starter)



```
xs518-shigoy@xs-host603-060:~/Downloads/Modules/L1M3/ShellPractice$ internsctl --help

Usage: ./internsctl [command][options] [options]

Commands:
  cpu getinfo                Lists detailed information about the CPU
  memory getinfo             Shows the memory statics of the system
  user create <username>     Creates new user with particular credential
  user list                  Lists all the regular users present of the server
                             Options:
                                     --sudo-only, -s Returns only the size of the file
  user list --sudo-only      Show users with sudo access
  file getinfo <file-name>    Lists information about the file access, size, owner, last
modified
                             Options:
                                     --size, -s Returns only the size of the file
                                     --permissions, -p Returns only the permisions over a
 given file
                                     --owner, o  Return the owner information
                                     --last-modified, m Returns the last modified stats


Options:
  --help                     Show this help message
  --version                  Show version information
```

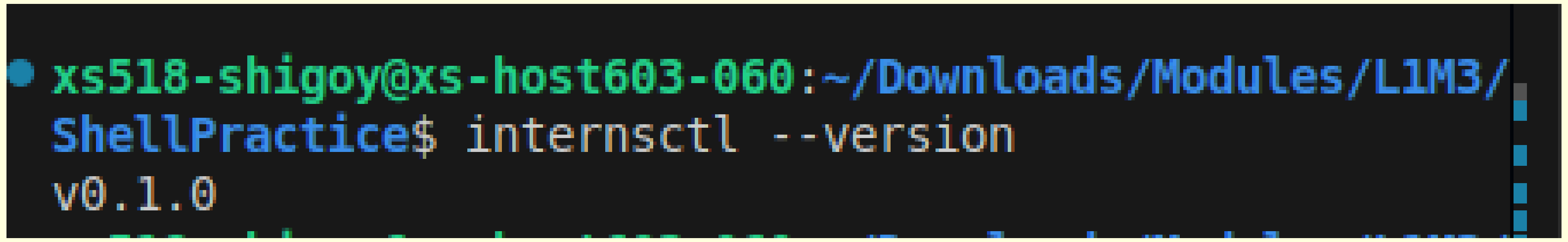
Working of --version:

Basically how this is working is

1) Check the first parameter i.e. “$1” is <--version>

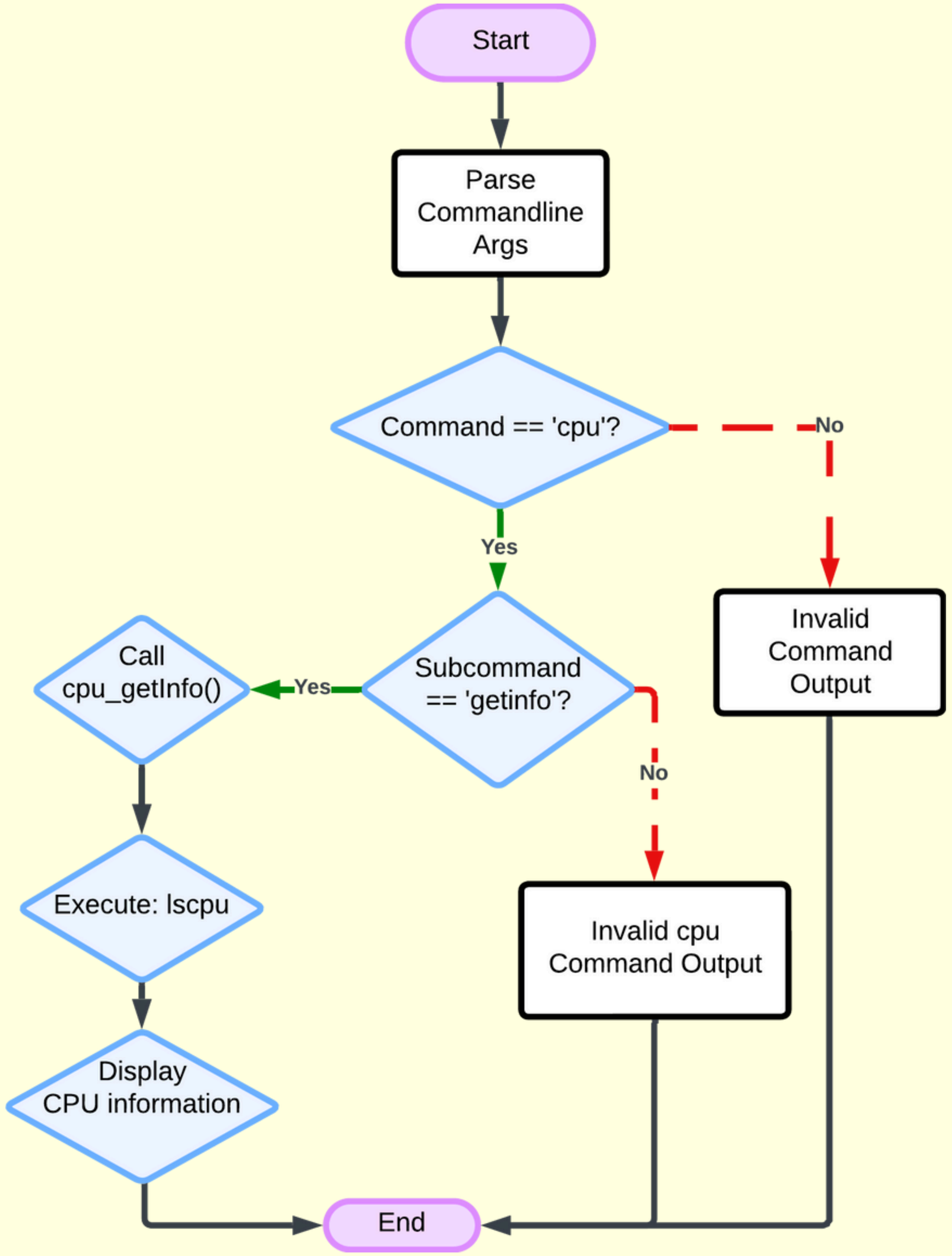
2) if Case --version) == true, echo out the version stored inside the <VERSION> variable

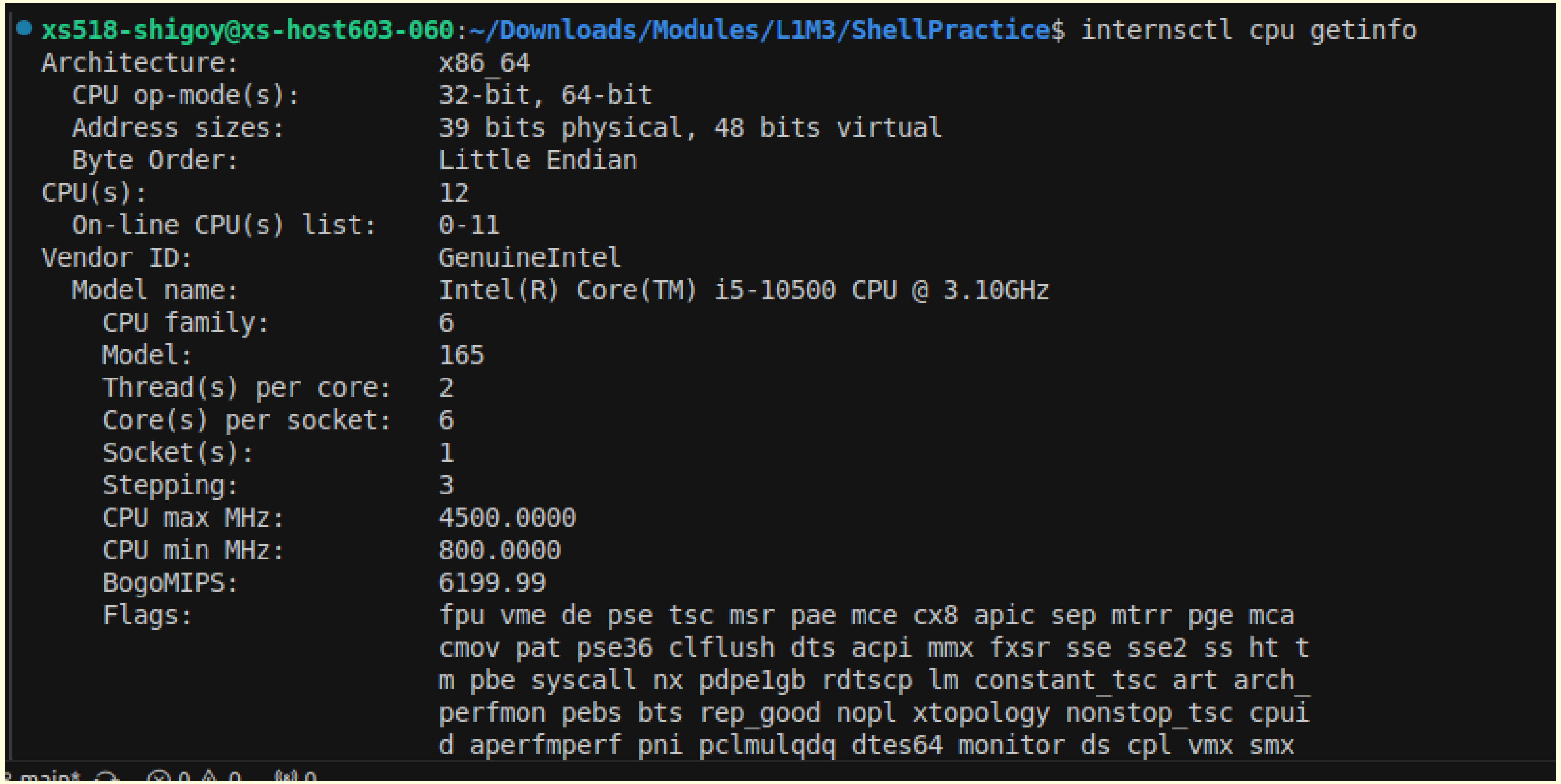
3) if Case “) show use --help message (also if $1 is any invalid starter)



```
xs518-shigoy@xs-host603-060:~/Downloads/Modules/L1M3/
ShellPractice$ internsctl --version
v0.1.0
```

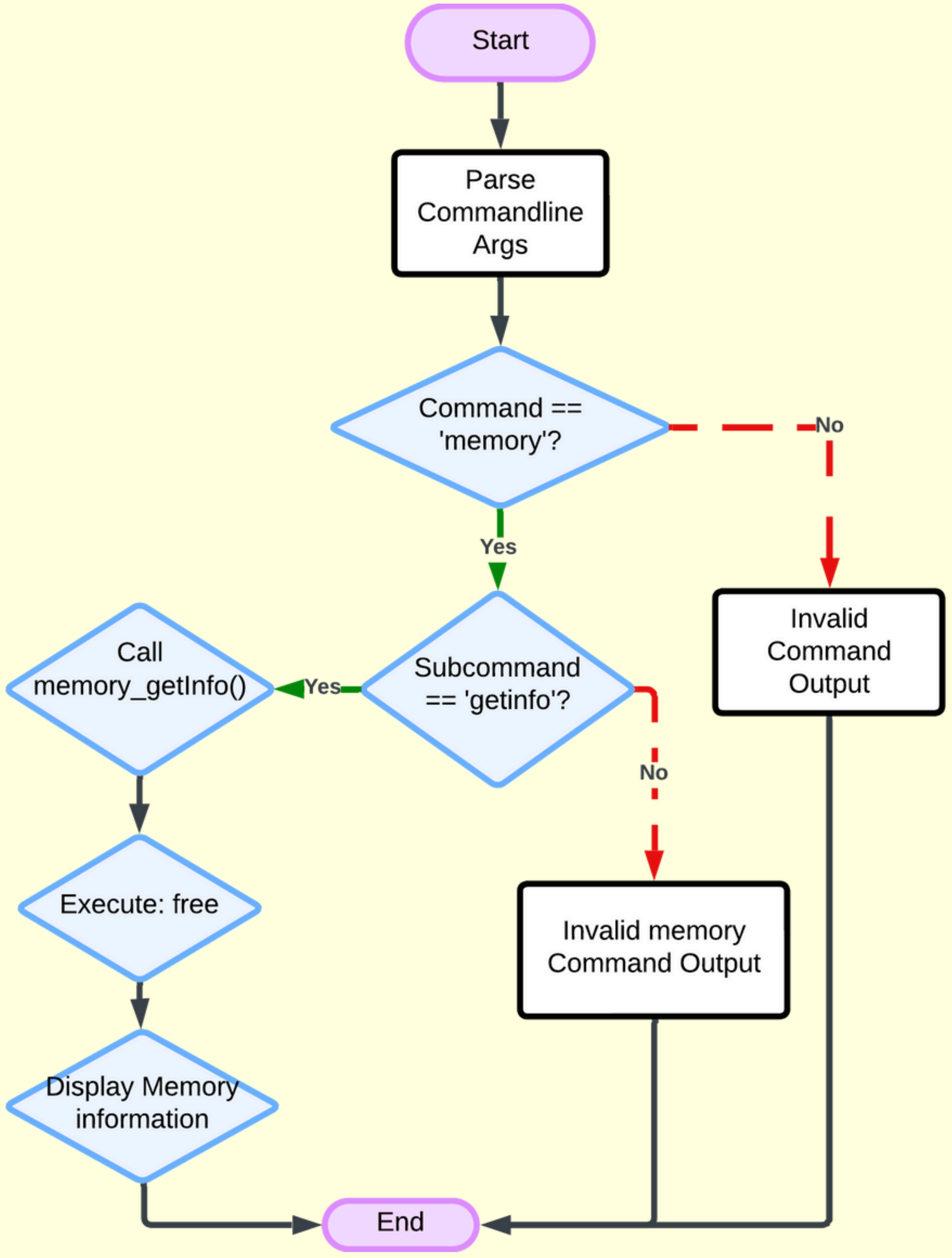
Working of “cpu getinfo”:





```
xs518-shigoy@xs-host603-060:~/Downloads/Modules/L1M3/ShellPractice$ internsctl cpu getinfo
Architecture:            x86_64
  CPU op-mode(s):        32-bit, 64-bit
  Address sizes:         39 bits physical, 48 bits virtual
  Byte Order:            Little Endian
CPU(s):                  12
  On-line CPU(s) list:   0-11
Vendor ID:               GenuineIntel
  Model name:            Intel(R) Core(TM) i5-10500 CPU @ 3.10GHz
    CPU family:          6
    Model:               165
    Thread(s) per core:  2
    Core(s) per socket:  6
    Socket(s):           1
    Stepping:            3
    CPU max MHz:         4500.0000
    CPU min MHz:         800.0000
    BogoMIPS:            6199.99
    Flags:               fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca
                         cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht t
                         m pbe syscall nx pdpe1gb rdtscp lm constant_tsc art arch_
                         perfmon pebs bts rep_good nopl xtopology nonstop_tsc cpui
                         d aperfmperf pni pclmulqdq dtes64 monitor ds cpl vmx smx
```

Working of “memory getinfo”:



Start

Parse Commandline Args

Command == 'memory'?

No

Invalid Command Output

Yes

Subcommand == 'getinfo'?

Yes

Call memory_getInfo()

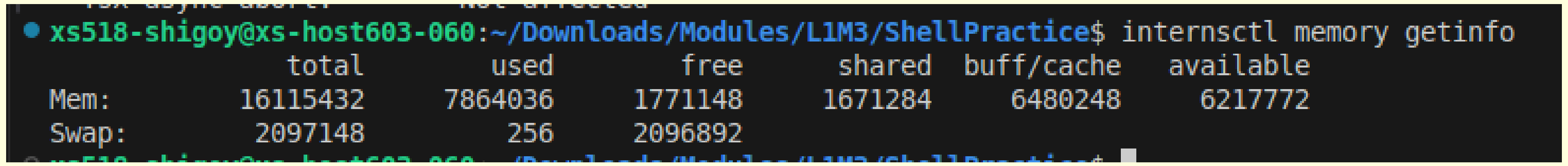
Execute: free

Display Memory information

No

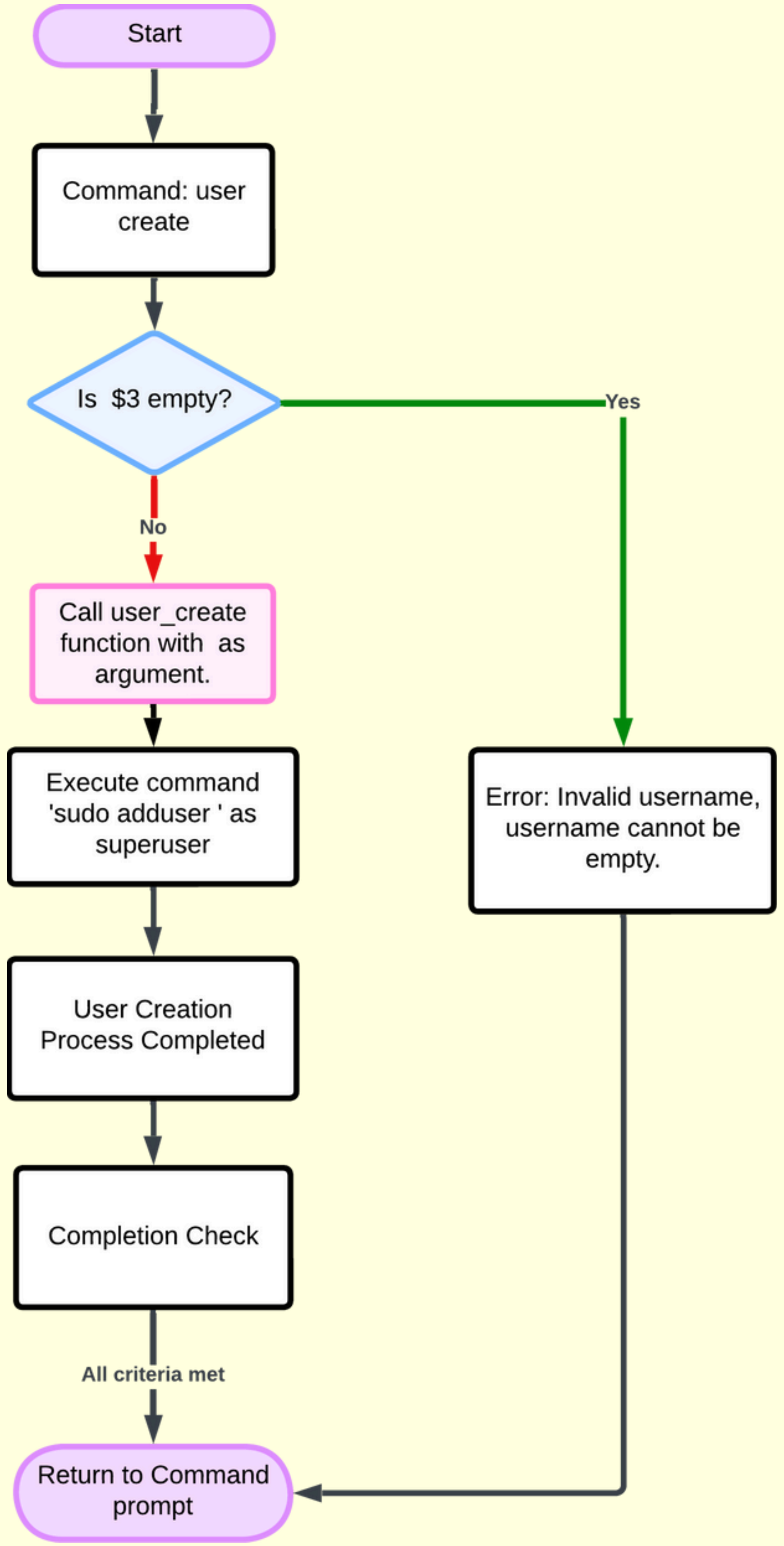
Invalid memory Command Output

End



```
  Tsx async abort:          Not affected
● xs518-shigoy@xs-host603-060:~/Downloads/Modules/L1M3/ShellPractice$ internsctl memory getinfo
               total        used        free      shared  buff/cache   available
 Mem:       16115432     7864036     1771148     1671284     6480248     6217772
 Swap:       2097148         256     2096892
○ xs518-shigoy@xs-host603-060:~/Downloads/Modules/L1M3/ShellPractice$ 
```

Working of “user create <username>”:



Start

Command: user create

Is $3 empty?

Yes

No

Call user_create function with as argument.

Error: Invalid username, username cannot be empty.

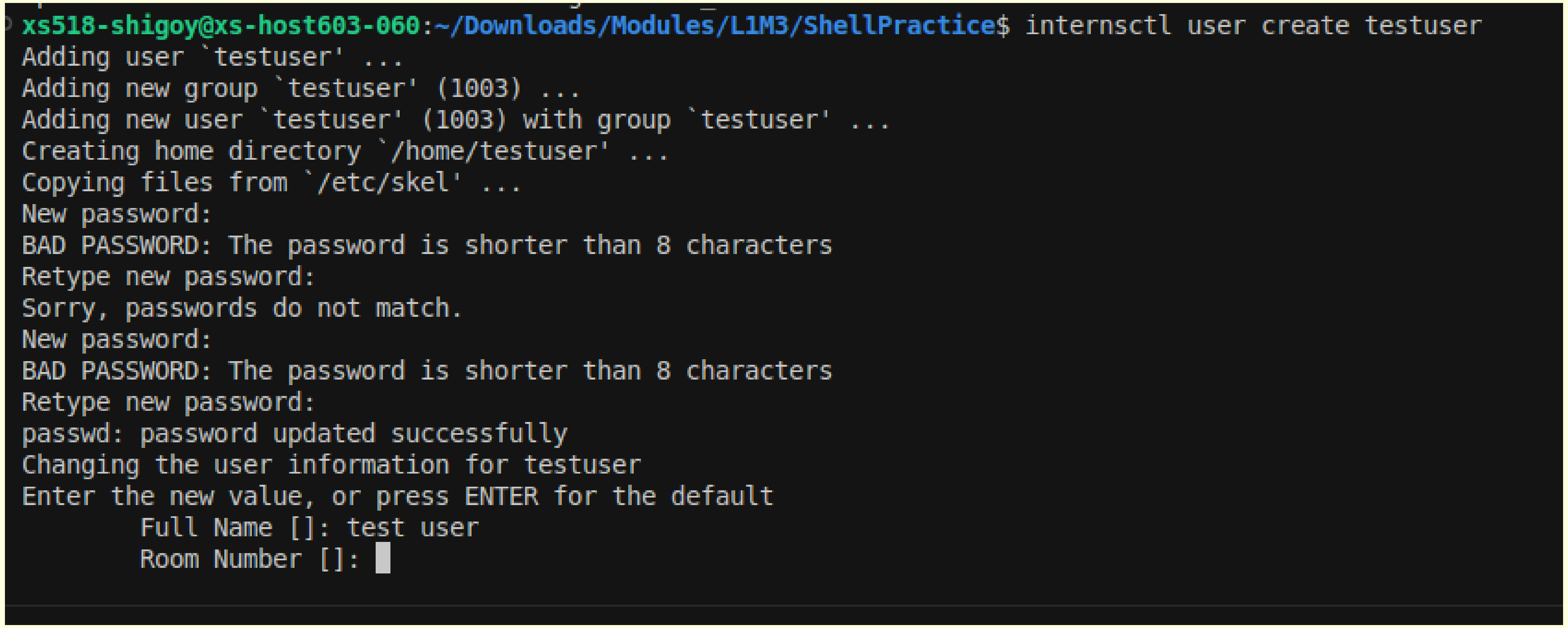
Execute command 'sudo adduser ' as superuser

User Creation Process Completed

Completion Check

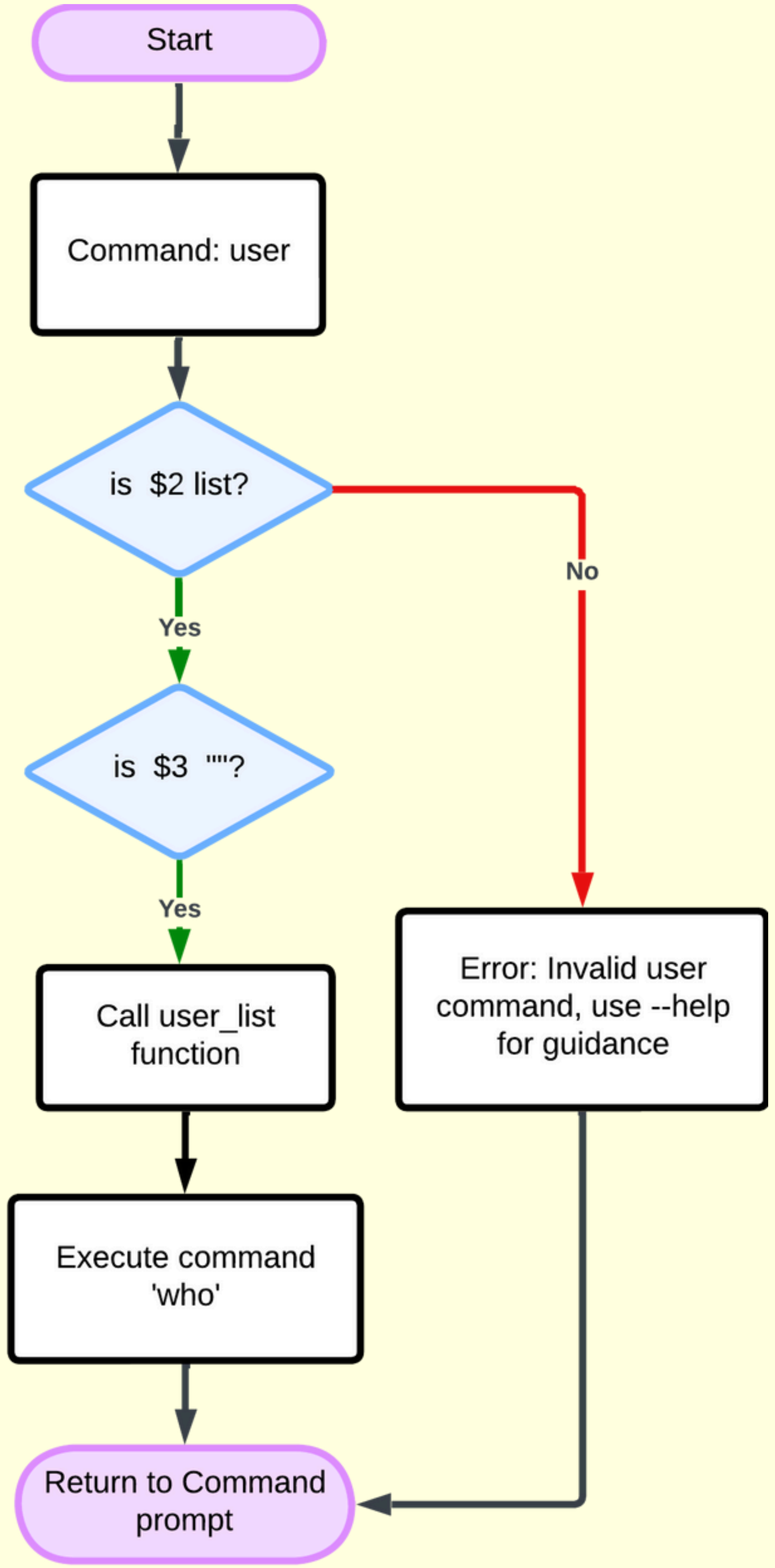
All criteria met

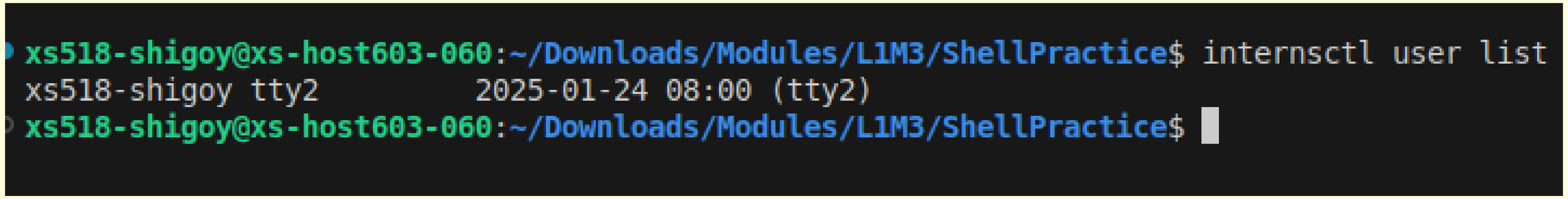
Return to Command prompt



```
xs518-shigoy@xs-host603-060:~/Downloads/Modules/L1M3/ShellPractice$ internsctl user create testuser
Adding user `testuser' ...
Adding new group `testuser' (1003) ...
Adding new user `testuser' (1003) with group `testuser' ...
Creating home directory `/home/testuser' ...
Copying files from `/etc/skel' ...
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
Sorry, passwords do not match.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: password updated successfully
Changing the user information for testuser
Enter the new value, or press ENTER for the default
        Full Name []: test user
        Room Number []: 
```

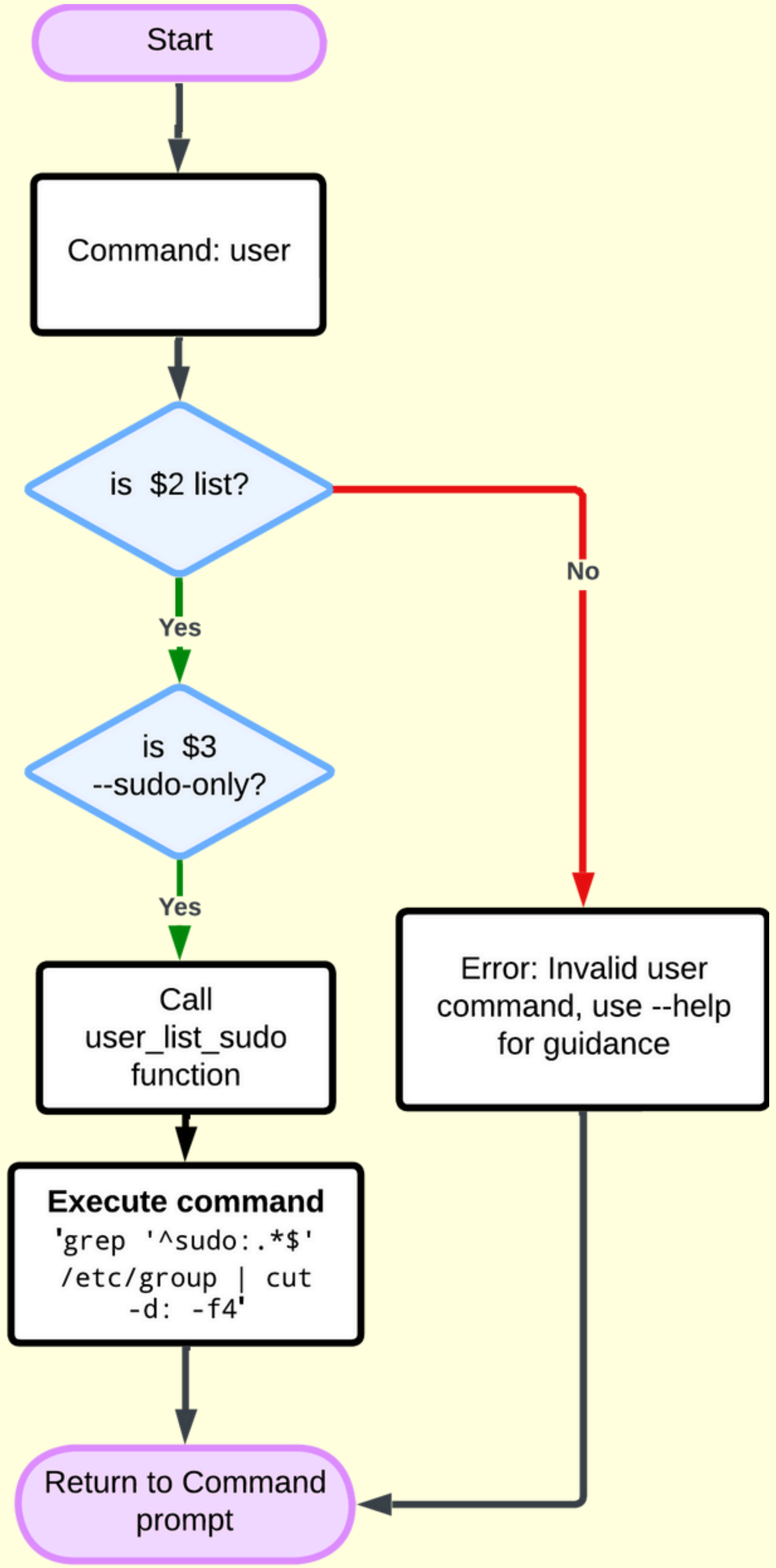
Working of “user list”:





```
xs518-shigoy@xs-host603-060:~/Downloads/Modules/L1M3/ShellPractice$ internsctl user list
xs518-shigoy tty2         2025-01-24 08:00 (tty2)
xs518-shigoy@xs-host603-060:~/Downloads/Modules/L1M3/ShellPractice$
```

Working of “user list --sudo-only”:



Start

Command: user

is $2 list?

Yes

No

is $3 --sudo-only?

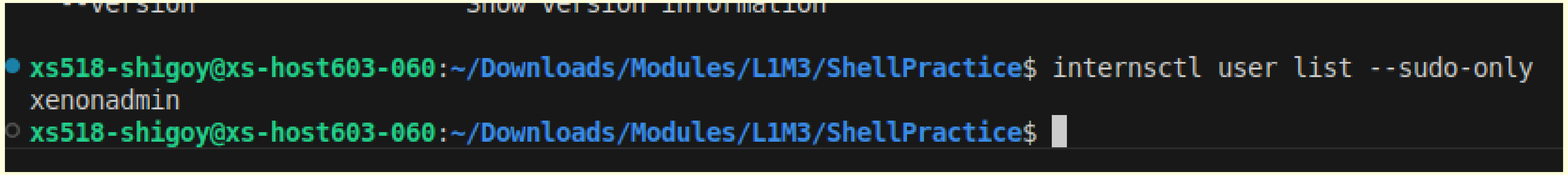
Yes

Call user_list_sudo function

Error: Invalid user command, use --help for guidance

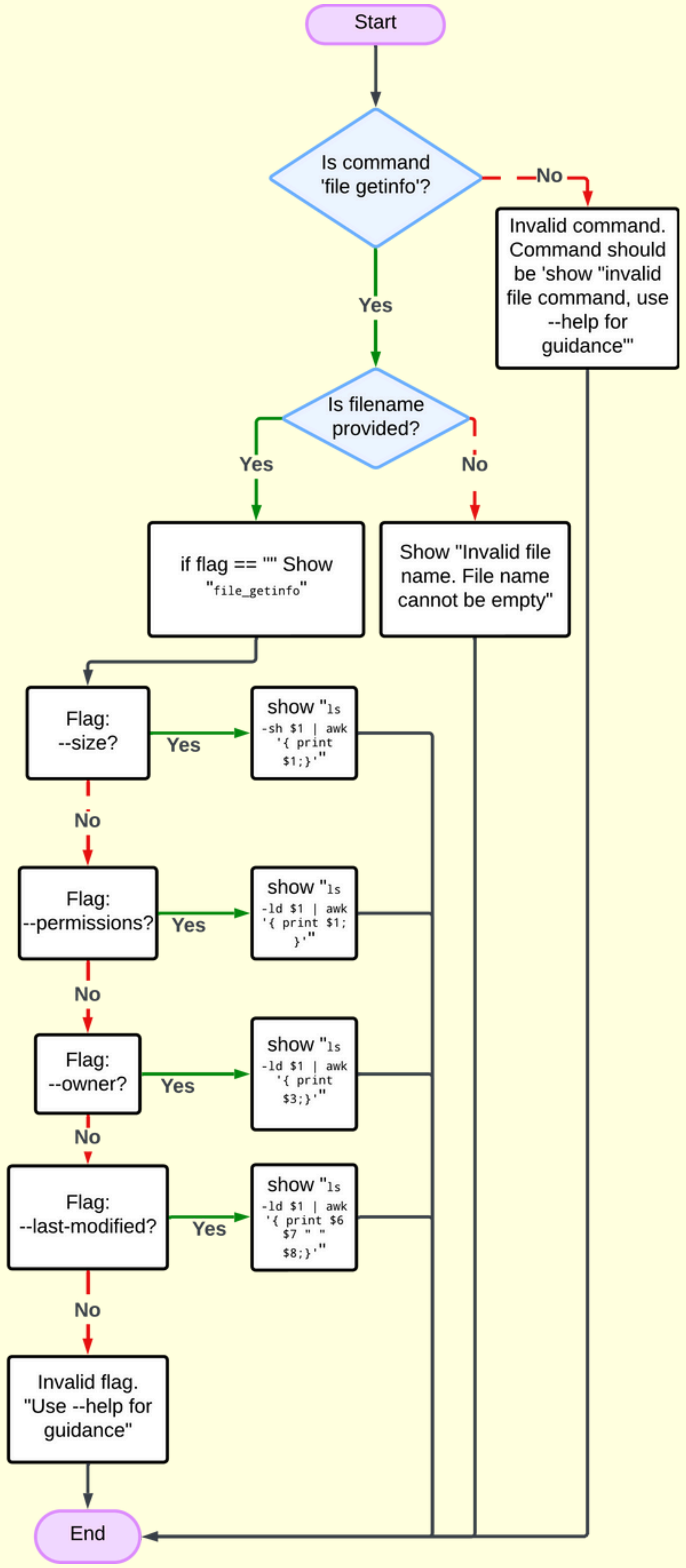
Execute command 'grep '^sudo:.*$' /etc/group | cut -d: -f4'

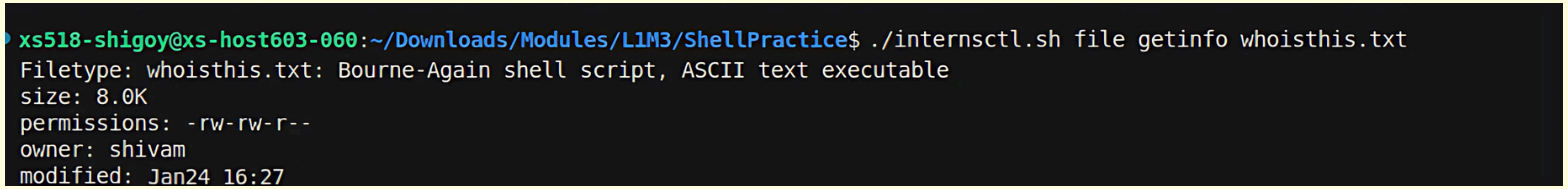
Return to Command prompt



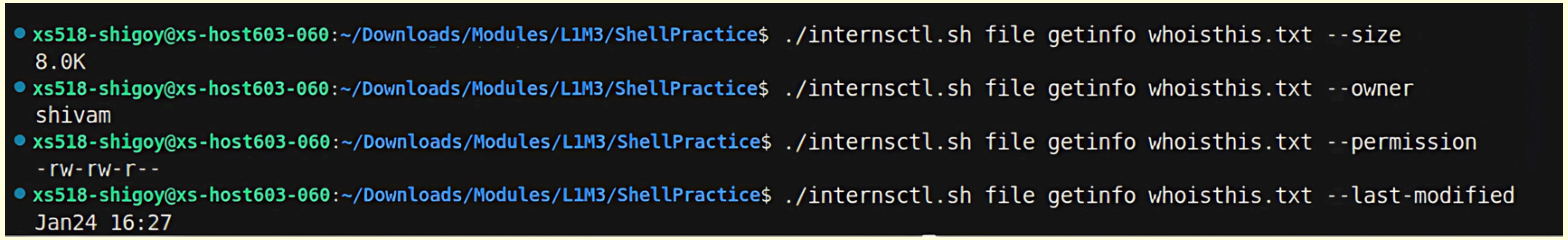
```
  --version                Show version information

xs518-shigoy@xs-host603-060:~/Downloads/Modules/L1M3/ShellPractice$ internsctl user list --sudo-only
xenonadmin
xs518-shigoy@xs-host603-060:~/Downloads/Modules/L1M3/ShellPractice$ 
```

Working of “user list --sudo-only”:


Start
Is command 'file getinfo'?
No
Invalid command. Command should be 'show "invalid file command, use --help for guidance"'
Yes
Is filename provided?
Yes
No
if flag == "" Show "file_getinfo"
Show "Invalid file name. File name cannot be empty"
Flag: --size?
Yes
show "ls -sh $1 | awk '{ print $1;}'"
No
Flag: --permissions?
Yes
show "ls -ld $1 | awk '{ print $1; }'"
No
Flag: --owner?
Yes
show "ls -ld $1 | awk '{ print $3;}'"
No
Flag: --last-modified?
Yes
show "ls -ld $1 | awk '{ print $6 $7 " " $8;}'"
No
Invalid flag. "Use --help for guidance"
End



```
xs518-shigoy@xs-host603-060:~/Downloads/Modules/L1M3/ShellPractice$ ./internsctl.sh file getinfo whoisthis.txt
Filetype: whoisthis.txt: Bourne-Again shell script, ASCII text executable
size: 8.0K
permissions: -rw-rw-r--
owner: shivam
modified: Jan24 16:27
```



```
● xs518-shigoy@xs-host603-060:~/Downloads/Modules/L1M3/ShellPractice$ ./internsctl.sh file getinfo whoisthis.txt --size
  8.0K
● xs518-shigoy@xs-host603-060:~/Downloads/Modules/L1M3/ShellPractice$ ./internsctl.sh file getinfo whoisthis.txt --owner
  shivam
● xs518-shigoy@xs-host603-060:~/Downloads/Modules/L1M3/ShellPractice$ ./internsctl.sh file getinfo whoisthis.txt --permission
  -rw-rw-r--
● xs518-shigoy@xs-host603-060:~/Downloads/Modules/L1M3/ShellPractice$ ./internsctl.sh file getinfo whoisthis.txt --last-modified
  Jan24 16:27
```

Thank you!!