Source Code Management

Course Code: CSE 2015

Slot: L3-l4

Lab Session 1: Git Fundamentals

Computer

A **computer** is any device capable of performing calculations, whether they are logical or mathematical.

Program/Code

A **program** (or **code**) is a set of instructions, often organized as an algorithm, that directs a computer to perform a specific task.

Need for Managing Source Code

Modern applications, such as Spotify, consist of multiple programs working together on both the frontend and backend to deliver smooth user experience.
Regular updates are essential for:

- **Fixing Bugs:** Quickly resolving errors that may occur.
- **Improving UI/UX:** Enhancing the user interface and overall experience.
- **Optimizing Performance:** Addressing and refining issues for better performance.

For programmers, effective management of source code is crucial because:

- It ensures that all files remain in context throughout the lifecycle of the program.
- It facilitates collaboration, allowing multiple developers to work together on a shared codebase.

Tools for Source Code Management

1. **Git:**
   A version control system that runs locally on your computer. Git helps track changes and manage versions of your project.
2. **GitHub:**
   A global, cloud-based platform that hosts Git repositories, enabling developers to share, collaborate, and contribute to projects from anywhere in the world.

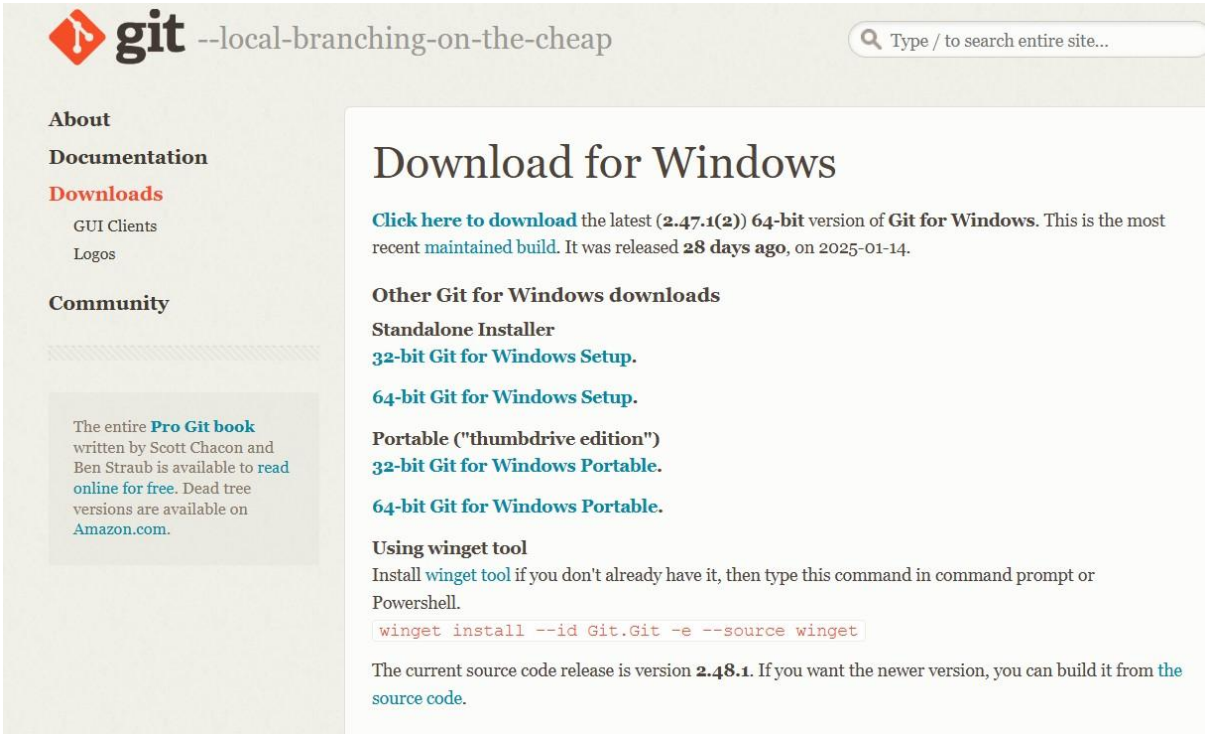
Version

A **version** in version control represents a snapshot of your project at a specific moment in time. This snapshot allows you to review, revert, or compare changes made throughout the development process.

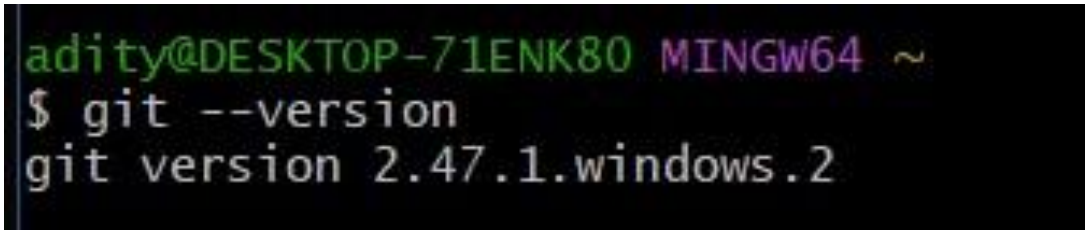
Lab Practical 1

1. Installing Git in Windows

Step 1: Visit section 1.5 of pro git document and navigate to Windows section



Step 2: Verify Git Installation:



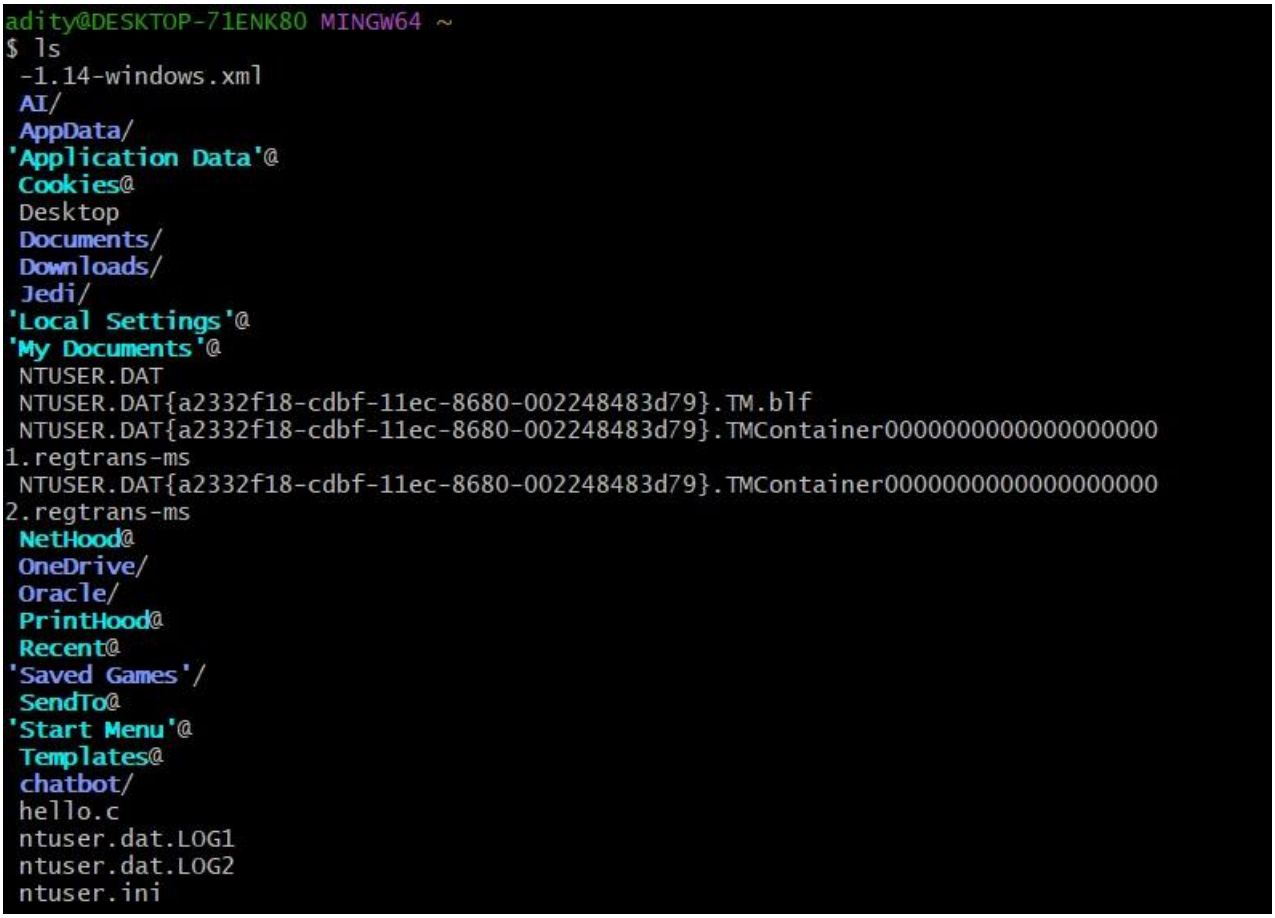
2. Basic CLI Commands

1) Command: pwd

Description: Prints the directory the user is working in.

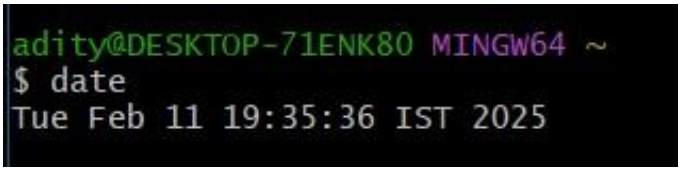
2) Command: ls

Description: Lists all files and directories in the current directory.



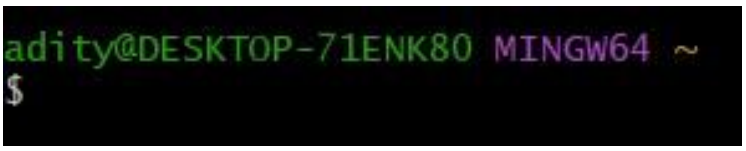
3. Command: date

Description: shows the current date and time in a standard format



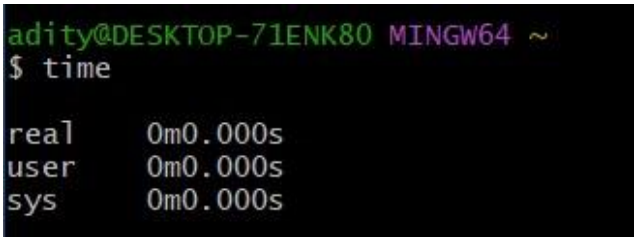
4. Command: clear

Description: The `clear` command in the CLI is used to clear all the current text and output displayed in the terminal window.



5. Command: time

Description: The `time` command in the CLI is used to measure the execution time of a command or program.



6. Command: cd ‘Directory’

Description: Changes the current working directory to the desired directory.

```
adity@DESKTOP-71ENK80 MINGW64 ~
$ cd Documents

adity@DESKTOP-71ENK80 MINGW64 ~/Documents
$
```

7. Command: cd ..

Description: Goes back to the previous directory.

```
adity@DESKTOP-71ENK80 MINGW64 ~
$ cd Documents

adity@DESKTOP-71ENK80 MINGW64 ~/Documents
$ cd ..

adity@DESKTOP-71ENK80 MINGW64 ~
$
```

8. Command: mkdir

Description: To create a new directory.

```
adity@DESKTOP-71ENK80 MINGW64 ~/Documents
$ mkdir Trial

adity@DESKTOP-71ENK80 MINGW64 ~/Documents
$ ls
'Pro Tools'/   Trial/
```

9. Command: rmdir

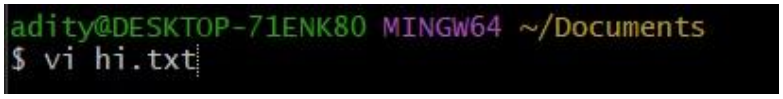
Description: To delete a directory

```
adity@DESKTOP-71ENK80 MINGW64 ~/Documents
$ mkdir Trial

adity@DESKTOP-71ENK80 MINGW64 ~/Documents
$ ls
'Pro Tools'/   Trial/

adity@DESKTOP-71ENK80 MINGW64 ~/Documents
$ rmdir Trial

adity@DESKTOP-71ENK80 MINGW64 ~/Documents
$ ls
'Pro Tools'/
```

3. Vim Text Editor

1) Command: vi hi.txt

Description: Opens (or creates) the file `hi.txt` in the Vim text editor.



2) Command: `i` (Insert Mode)

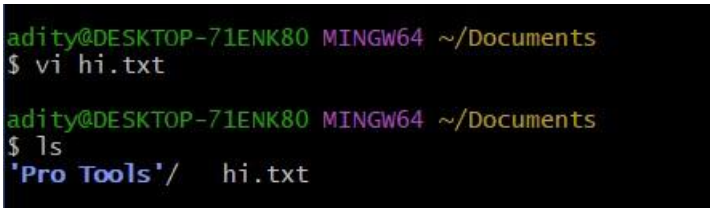
Description: Enters insert mode in Vim to allow text input.

3) Command: esc

Description: Used to exit insert mode

4) Command: `:wq`

Description: Saves the changes and exits the Vim editor.



4. Git Commands

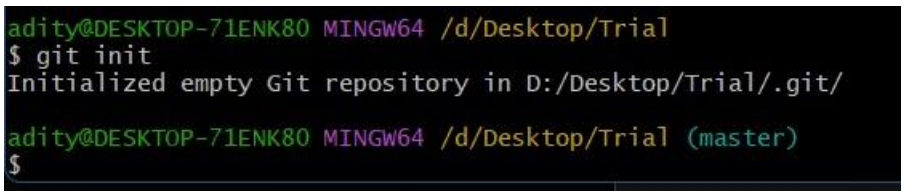
1. Command: git - - version

Description: The `git --version` command is used to check the installed version of Git on your system.

bash

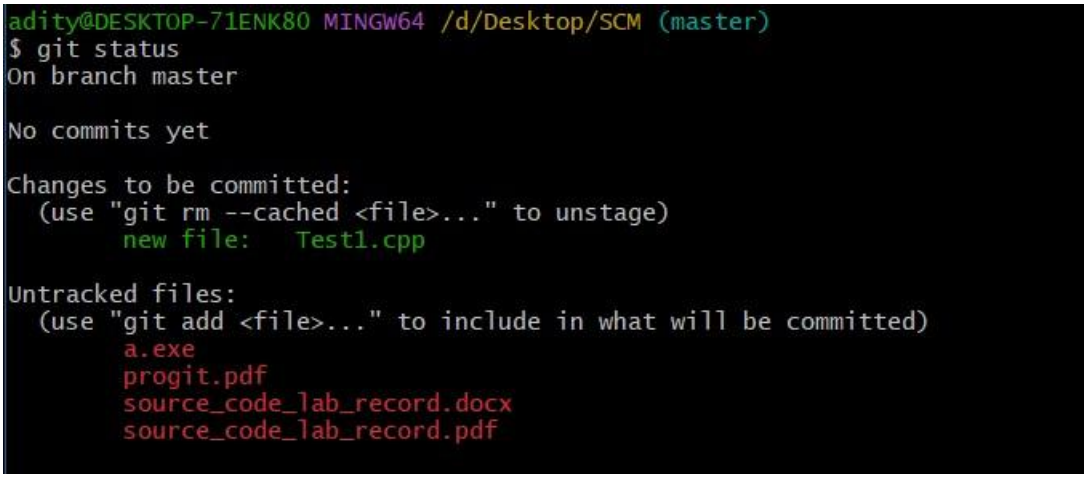
2. Command: git init

Description: Initializes a new Git repository in the current directory.



3. Command: git status

Description: Displays the current status of the working directory and staging area.



4. Command: git add Test.c

Description: Add Test.c to the staging area in preparation for a commit.

```
adity@DESKTOP-71ENK80 MINGW64 /d/Desktop/SCM (master)
$ git add Test.c
warning: in the working copy of 'Test.c', LF will be replaced by CRLF the next
ime Git touches it
```

5. Command: git commit -m “add file one”

Description: Commits the stage changes with the message “add file one”.

```
adity@DESKTOP-71ENK80 MINGW64 /d/Desktop/SCM (master)
$ git commit -m "add file one"
[master 9e27bc1] add file one
 1 file changed, 1 insertion(+), 1 deletion(-)
```

6. Command: git log

Description: Display the commit history of the repository.

```
adity@DESKTOP-71ENK80 MINGW64 /d/Desktop/SCM (master)
$ git log
commit 9e27bc1e7968583b9f990dddd92f7bd6fa3b3591 (HEAD -> master)
Author: Aditya12705 <aditya.banerjee12705@gmail.com>
Date:   Wed Feb 19 10:17:57 2025 +0530

    add file one
```

7. Command: git clone

Description: To obtain a copy of an existing Git repository.

```
adity@DESKTOP-71ENK80 MINGW64 /d/Desktop/SCM (master)
$ git clone https://github.com/AsmSafone/MusicPlayer
Cloning into 'MusicPlayer'...
remote: Enumerating objects: 266, done.
remote: Counting objects: 100% (116/116), done.
remote: Compressing objects: 100% (33/33), done.
remote: Total 266 (delta 97), reused 83 (delta 83), pack-reused 150 (fro
m 2)
Receiving objects: 100% (266/266), 1.84 MiB | 2.24 MiB/s, done.
Resolving deltas: 100% (145/145), done.
```

8. Command: git log --oneline

Description: For generating shorter commit ID.

```
adity@DESKTOP-71ENK80 MINGW64 /d/Desktop/SCM (master)
$ git log --oneline
9e27bc1 (HEAD -> master) add file one
751add6 Committing Test.c and Test2.c
0bdc113 committing Test.c
ef0886b committing Test.c
3e40141 committing Test.c
```

9. Command: git diff

Description: To compare two files.

```
adity@DESKTOP-71ENK80 MINGW64 /D/Desktop/SCM/SCM (master)
$ git diff b46a555 521a0d3
diff --git a/Trial.c b/Trial.c
index 7a64e83..e6ef922 100644
--- a/Trial.c
+++ b/Trial.c
@@ -8,5 +8,4 @@ int main()
        printf("Addition: %d", a+b);
        printf("Subtraction: %d", a-b);
        printf("Multiplication: %d", a*b);
-       printf("Division: %d", a/b);
 }
```

10. Command: git remote add “Variable”

Description: To connect with the Users GitHub account.

```
adity@DESKTOP-71ENK80 MINGW64 /D/Desktop/SCM/SCM (master)
$ git remote add SCM https://github.com/Aditya12705/SCM.git
```

11. Command: git remote

Description: To check the status of the repositories connected with the Users account.

```
adity@DESKTOP-71ENK80 MINGW64 /D/Desktop/SCM/SCM (master)
$ git remote
SCM
```

12. Command: git push -u “Variable” master

Description: To push all the files to the Users account.

```
adity@DESKTOP-71ENK80 MINGW64 /D/Desktop/SCM/SCM (master)
$ git push -u SCM master
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 12 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (12/12), 1.01 KiB | 343.00 KiB/s, done.
Total 12 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/Aditya12705/SCM.git
 * [new branch]      master -> master
branch 'master' set up to track 'SCM/master'.
```

13. Command: git merge “File_Name” -m “comment”

Description: To merge a branch with main branch.

```
adity@DESKTOP-71ENK80 MINGW64 /c/Users/aditya/Desktop/SCM-Project (main)
$ git merge aditya-css -m "Merging aditya-css branch"
Merge made by the 'ort' strategy.
 Blockchain.css | 494 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 1 file changed, 494 insertions(+)
 create mode 100644 Blockchain.css
```