Source Code Management

Chinmai.A

A866185824016

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.

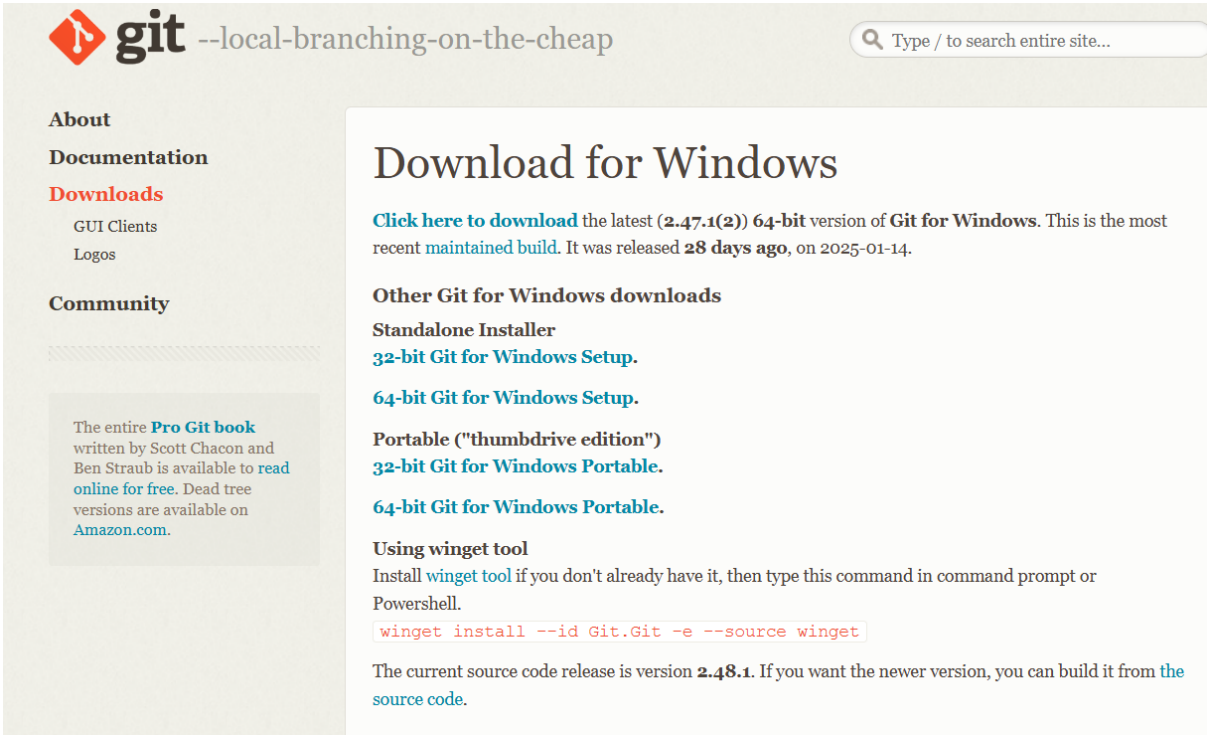
Version

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.

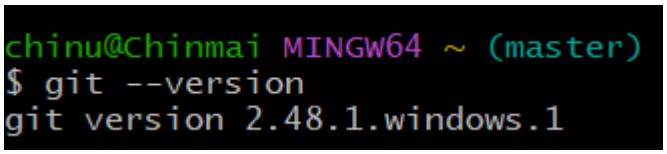
Lab Practical 1

1. Installing Git in Windows

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



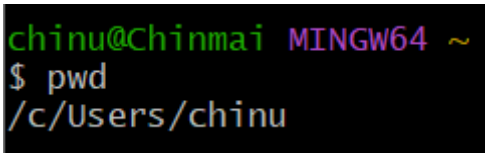
Step 2: Verify Git Installation:



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.

```
chinu@Chinmai MINGW64 ~ (master)
$ ls
 265/
 AppData/
'Application Data'@
 Assignment/
 Contacts/
 Cookies@
 Documents/
 Downloads/
 Favorites/
 Links/
'Local Settings'@
 Music/
'My Documents'@
 NTUSER.DAT
 NTUSER.DAT{739759be-573b-11ef-b696-2c7ba0d2adfb}.TM.blf
 NTUSER.DAT{739759be-573b-11ef-b696-2c7ba0d2adfb}.TMContainer00000000000000000001.regtrans-ms
 NTUSER.DAT{739759be-573b-11ef-b696-2c7ba0d2adfb}.TMContainer00000000000000000002.regtrans-ms
 NetHood@
 OneDrive/
 PrintHood@
 Recent@
'Saved Games'/
 Searches/
 SendTo@
'Start Menu'@
 Templates@
 Videos/
 assignment1/
 assignment2/
 assisment/
 code.cpp
 code_4016/
 ntuser.dat.LOG1
 ntuser.dat.LOG2
 ntuser.ini
 num_4016/
 prg1.c
```

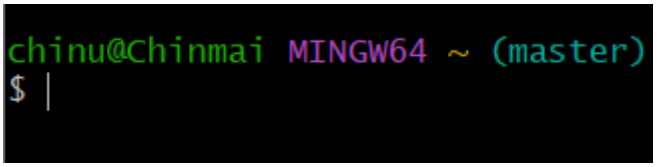
2. Command: date

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

```
chinu@Chinmai MINGW64 ~ (master)
$ date
Wed May 28 11:32:50 IST 2025
```

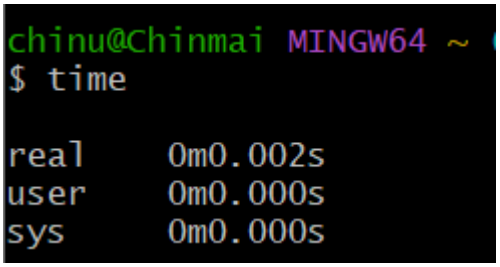
3. Command: clear

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



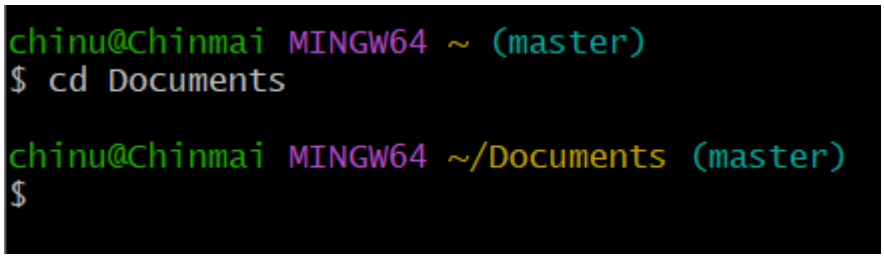
4. Command: time

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



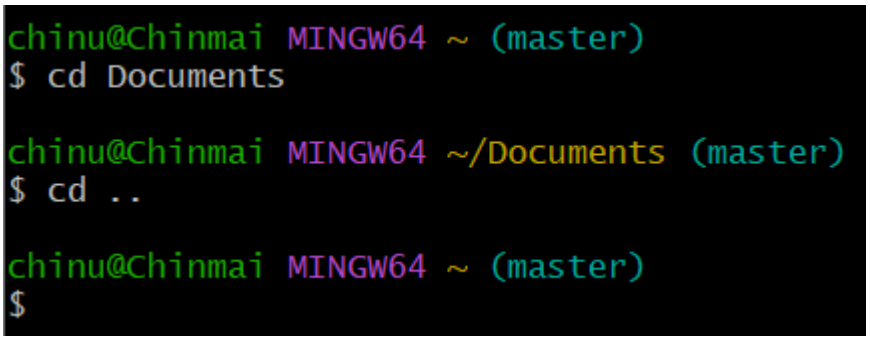
5. Command: cd 'Directory'

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



6. Command: cd ..

Description: Goes back to the previous directory.



7. Command: mkdir

Description: To create a new directory.

```
chinu@Chinmai MINGW64 /c/Users/Public
$ mkdir source

chinu@Chinmai MINGW64 /c/Users/Public
$ ls
AccountPictures/  Documents/  Libraries/  Pictures/  desktop.ini
Desktop/          Downloads/  Music/      Videos/    source/
```

9. Command: rmdir

Description: To delete a directory

```
chinu@Chinmai MINGW64 /c/Users/Public
$ ls
AccountPictures/  Documents/  Libraries/  Pictures/  desktop.ini
Desktop/          Downloads/  Music/      Videos/    source/

chinu@Chinmai MINGW64 /c/Users/Public
$ rmdir source

chinu@Chinmai MINGW64 /c/Users/Public
$ ls
AccountPictures/  Documents/  Libraries/  Pictures/  desktop.ini
Desktop/          Downloads/  Music/      Videos/
```

Vim Text Editor

Command: vi program.cpp

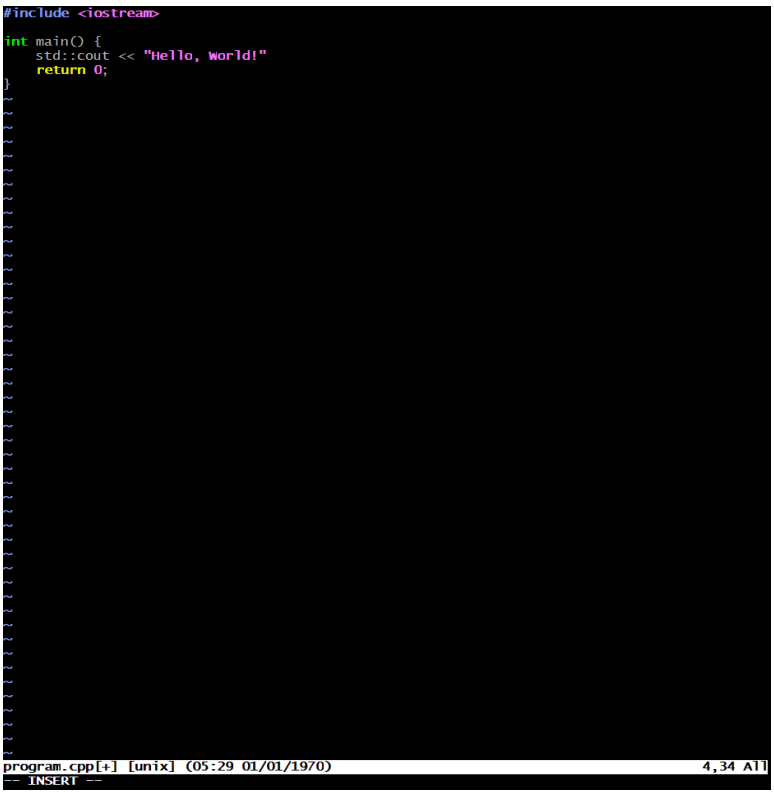
Description: Opens (or creates) the file `program.cpp` in the Vim text editor.

```
chinu@Chinmai MINGW64 /c/Users/Public
$ vi program.cpp
```

1) Command: `i` (Insert Mode)

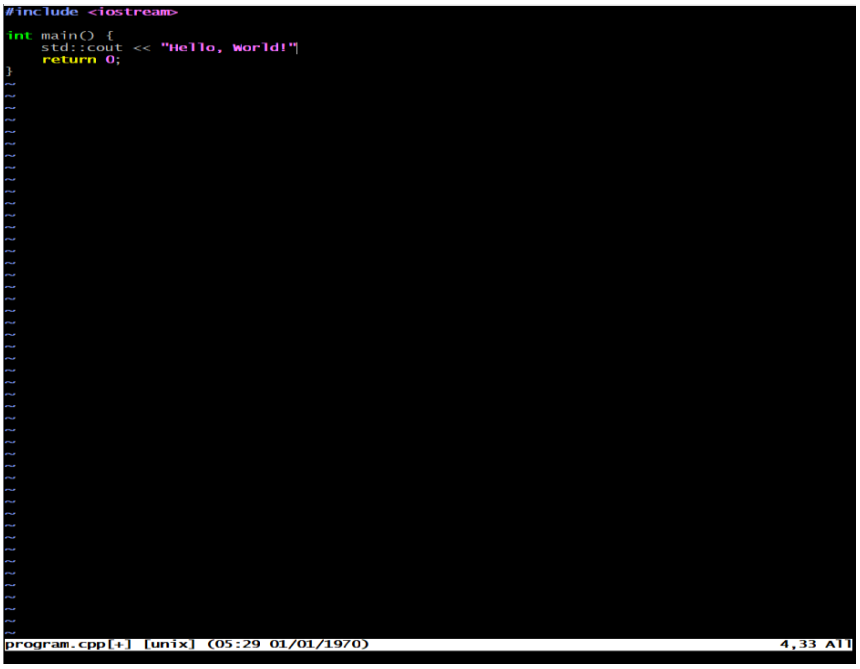
Description: Enters insert mode in Vim to allow text input.


program.cpp [unix] (05:29 01/01/1970) 0,1 All
-- INSERT --


#include <iostream>
int main() {
std::cout << "Hello, World!"
return 0;
}
program.cpp[+] [unix] (05:29 01/01/1970) 4,34 All
-- INSERT --

2) Command: esc

Description: Used to exit insert mode



3) Command: `:wq`

Description: Saves the changes and exits the Vim editor.

```
chinu@Chinmai MINGW64 /c/Users/Public
$ vi program.cpp

chinu@Chinmai MINGW64 /c/Users/Public
$ ls
AccountPictures/  Documents/  Libraries/  Pictures/  desktop.ini
Desktop/          Downloads/  Music/      Videos/    program.cpp
```

4. Git Commands

1. Command: git - - version

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

```
chinu@Chinmai MINGW64 ~/Public
$ git --version
git version 2.48.1.windows.1
```

2. Command: git init

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

```
chinu@Chinmai MINGW64 ~/Public (master)
$ git init
Reinitialized existing Git repository in C:/Users/chinu/Public/.git/

chinu@Chinmai MINGW64 ~/Public (master)
$
```

3. Command: git status

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

```
chinu@Chinmai MINGW64 ~/Public (master)
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

4. Command: git add program.cpp

Description: Add program.cpp to the staging area in preparation for a commit.

```
chinu@Chinmai MINGW64 ~/Public (master)
$ git add program.cpp
```

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

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

```
chinu@Chinmai MINGW64 ~/Public (master)
$ git commit -m "Source project"
[master (root-commit) 1186b46] Source project
 1 file changed, 1 insertion(+)
 create mode 100644 program.cpp
```

6. Command: git log

Description: Display the commit history of the repository.

```
chinu@Chinmai MINGW64 ~/Public (master)
$ git log
commit 1186b46a961b16ad54f46b031150ca75f53cb64f (HEAD -> master)
Author: Chinmai A <chinmai.a@s.amity.edu>
Date:   Wed May 28 12:09:52 2025 +0530

    Source project
```

7. Command: git clone

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

```
chinu@Chinmai MINGW64 ~/Public (master)
$ git clone https://github.com/shreyas656/source-team-project
Cloning into 'source-team-project'...
remote: Enumerating objects: 34, done.
remote: Counting objects: 100% (34/34), done.
remote: Compressing objects: 100% (30/30), done.
remote: Total 34 (delta 9), reused 24 (delta 3), pack-reused 0 (from 0)
Receiving objects: 100% (34/34), 13.00 KiB | 700.00 KiB/s, done.
Resolving deltas: 100% (9/9), done.
```

8. Command: git log --oneline

Description: For generating shorter commit ID.

```
chinu@Chinmai MINGW64 ~/Public (master)
$ git log --oneline
1186b46 (HEAD -> master) Source project
```