Some software development tools

Professor Hossein Saiedian

EECS 348: Software Engineering

Fall 2024

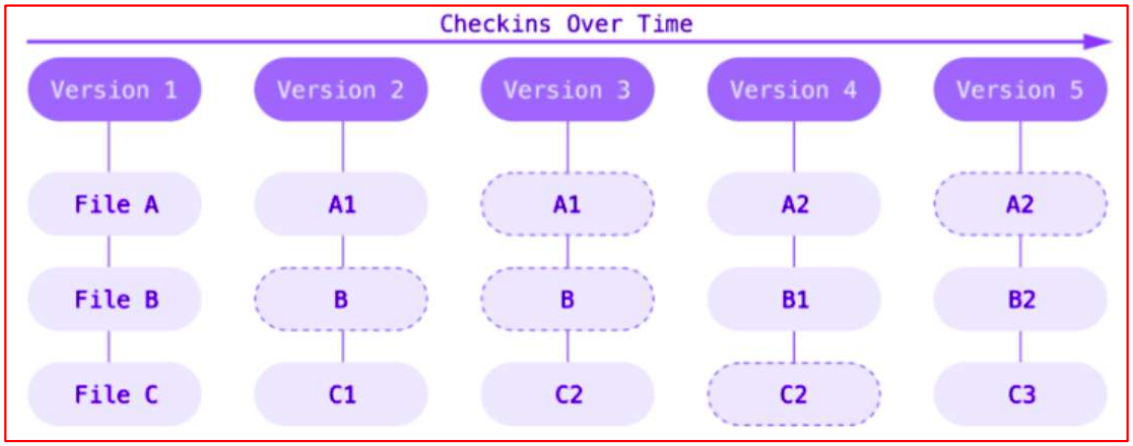
THE UNIVERSITY OF KANSAS

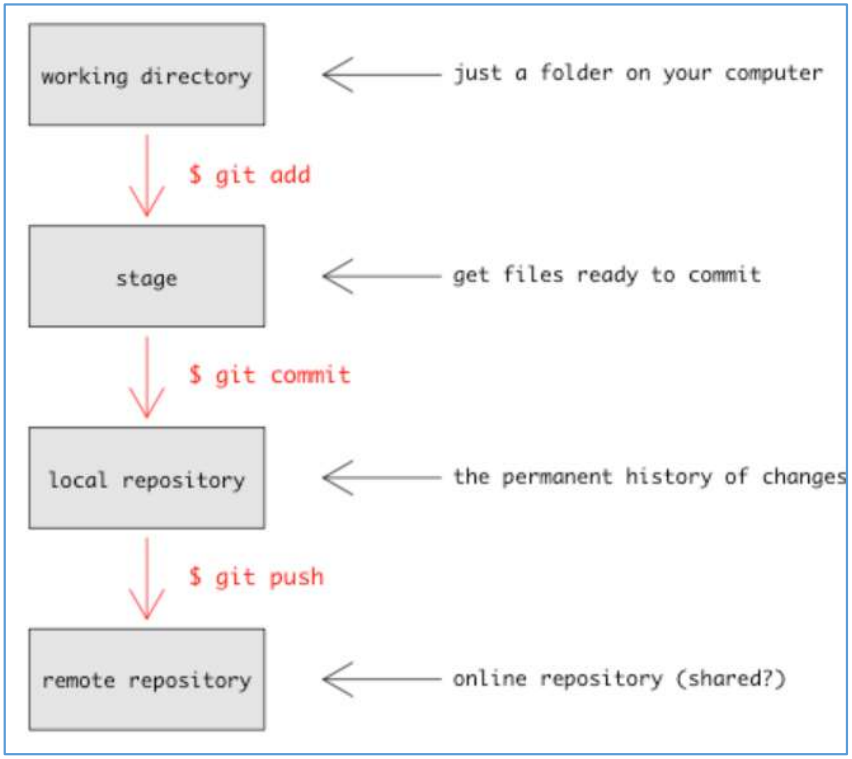
A brief intro to Git and GitHub

- Software tool: **git**
- Git is a version control system (VCS)
- Allows to maintain multiple versions of a code base
  - Keeps a history of previous changes (*track changes*)
  - How? *Snapshots*
  - Let's you see the changes you make to your code and easily revert them
  - Sometimes across multiple developers
    * Collaborate with other developers
    * Push and pull code from repositories such as GitHub

Snapshots, not differences

- Git: Stores snapshots





- Available on Linux cycle servers (*command line interface*)
- Available for installation on Windows and Mac machines

What is GitHub

- Another tool: GitHub.com
- GitHub.com is a website server that hosts git repositories
- Hosting repositories facilitates the sharing of codebases among teams by providing a GUI to easily clone repos to a local machine
- When you push your code repositories on GitHub, you will be creating your own developer's portfolio
- Lots of resources online and on YouTube

Lab 1: git and GitHub

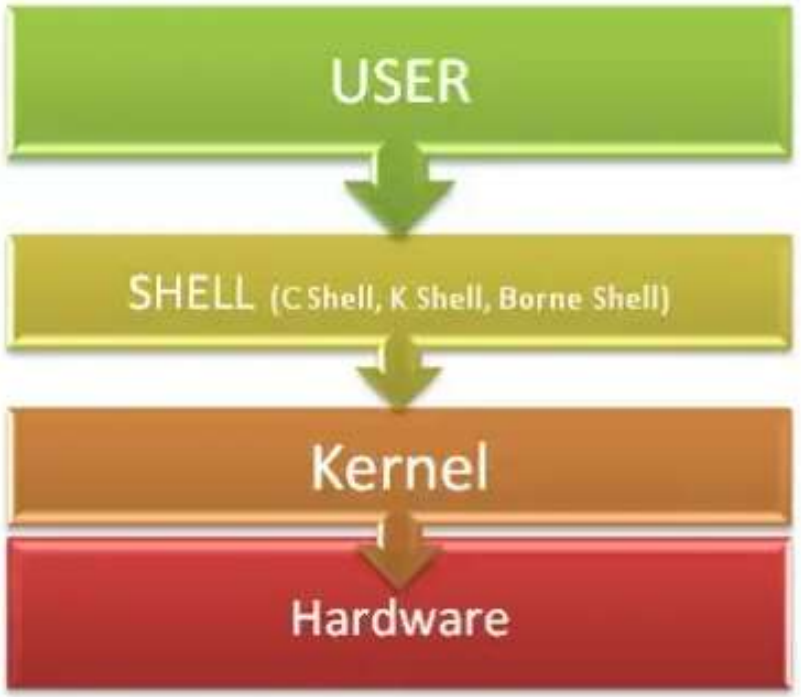
- Objective
  - Learn how to create a repo (repository) via **git**
  - Learn how to push/pull a repo to/from **GitHub**
- Important concepts
  - Repositories (repo)
  - Snapshots and commit
  - Pushing/pulling

The shell (operating system)

- Another tool: the shell
- Operating systems provide a "command line" interface which allows the user to enter commands
  - These commands are translated by the shell into something the kernel can comprehend and execute
- Shell is not part of the operating system kernel
- It is a command line interpreter (CLI)
- In Unix, a user can pick their shell
  - Popular Unix (Linux) shell: **sh, bash, ksh, csh, tcsh**, …
  - Default shell on EECS Linux machines: **bash**

A simplified view of a shell

- A shell is the primary interface between a user sitting at the terminal and the operating system



The heart of the OS

https://medium.com/@clturner23

Default shell

- Linux default shell: most Linux systems default to the **bash** shell
  - I prefer **csh** (some similarity with the C language)
- Once you learn more about the shell options, you may want to change to another one
  - Command to change shell: **chsh**

Useful shell commands

- Directory: **mkdir, rmdir, cd, ls, …**
- Files: **cp, cat, mv, rm, sort, wc, …**
- Search: **grep, find, …**
- Editor: **vi, vim, emacs, nano** (for the beginners)
- File/directory permission: **chmod, chown, …**
- Software development: **make, tar, git, vim, …**
- Many others

Software engineering uses

- Writing shell scripts
- Create a text file
- Include in the first line: `#!/bin/bash`
- Write scripts that do different tasks
  - A shell script: a text file that contains a sequence of commands
  - Command sequences in which a user has a need to use repeatedly in order to save time
  - Shell scripts contain ASCII text and are written using a text editor
  - Automating the code compiling process
  - Executing routine backups
  - Shell commands can include assignment statements, if statements, loop statements, and so forth
  - Some special symbols also have their own meanings: #, %, $, |, [ ], …

A simple shell script

- The following bash script with assignment, if, and for statements

```
#!/bin/bash

# Declare a variable
number=10

# If-then-else statement
if [ $number -gt 5 ]; then
  echo "The number is greater than 5."
else
  echo "The number is 5 or less."
fi

# Assignment within a loop
for i in 1 2 3 4 5; do
    let "number = number + 2"  # Increment number by 2 in each iteration
    echo "The current number is: $number"
done

# Loop termination
echo "Loop finished."
```

Summary: a shell

- It is a command line interpreter (CLI)
- Unix/Linux affectionate love the CLI
  - It is very powerful and provides a lot of control
  - It is simple (there is an initial learning curve)
  - Nevertheless, many GUI interfaces too (most Linux sys admins and power users do not use GUI)
- You most likely will learn a lot more in an OS course
- For now, you need to learn the purpose and the very basics of shell scripting

Another SE tool: make

- Linux **make** command is a build automation tool in software development
  - It streamlines the process of building and compiling programs and other files from source code
  - It analyzes dependencies and executes necessary commands to create the final output, saving time and effort
  - Input: a file called **makefile**
- Example: Imagine you have a project with three C files: **main.c, calc.c**, and **report.c**

Another SE tool: make

- Without **make**
  - Change something in **calc.c**: You manually compile **calc.c** and **main.c** separately to update everything that relies on **calc.c**
  - Change something in **main.c**: You manually recompile all C files
  - It's tedious and error-prone, especially as the project grows
- With **make**
  - You write a **Makefile** explaining which files depend on others (e.g., **main.c** depends on **report.c** and **calc.c**).
  - Run **make**; **make** automatically recompiles only the necessary files based on dependencies, saving time and avoiding errors.

Software development IDEs

- Another tool: an IDE
- Integrated Develop Environments (IDEs) are software packages that provide comprehensive support for coding, testing, and debugging
- The components of an IDE
  - Editor
  - Build support (link, compile)
  - Execute
  - Debug

Nice features of an IDE

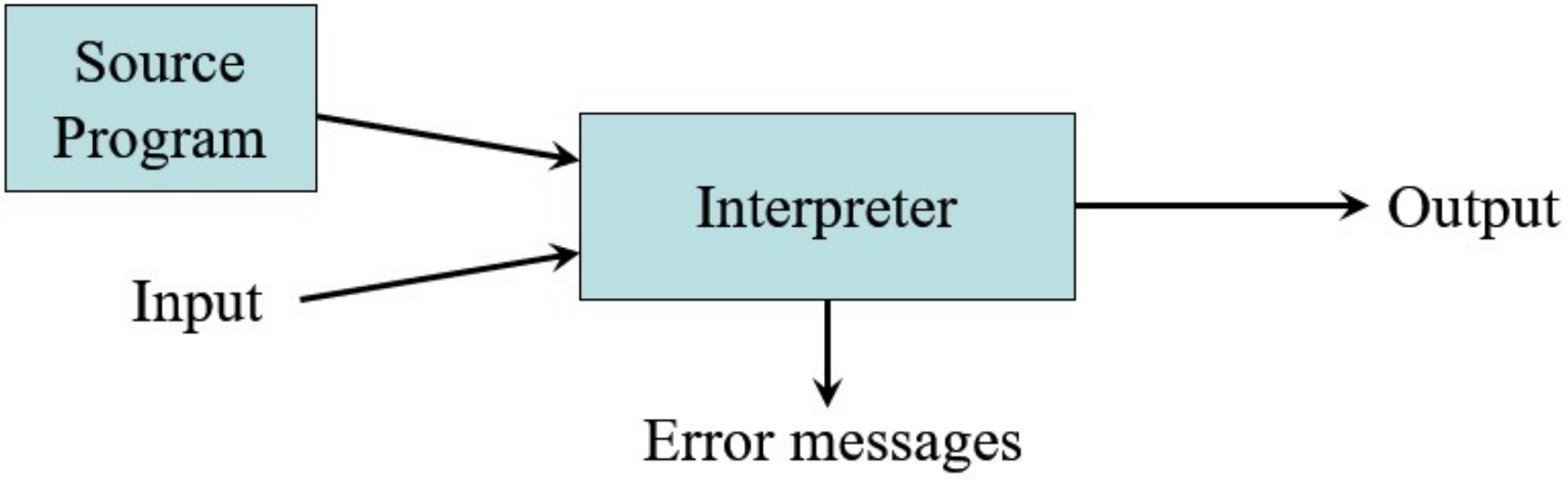
- Syntax highlighting and aid in editing (e.g., matching brackets)
- Packaging options (e.g., **tar** and **zip** archives)
- Posting to an online repository (e.g., GitHub)
- Configurable build support (e.g., multiple programming languages)
- Smart feedback
- Coding templates
- Documentation support/lookup

Examples of popular IDEs

- Microsoft Visual Studio Code
- Eclipse
- AWS Cloud9
- Android Studio
- PyCharm
- Spyder
- More …

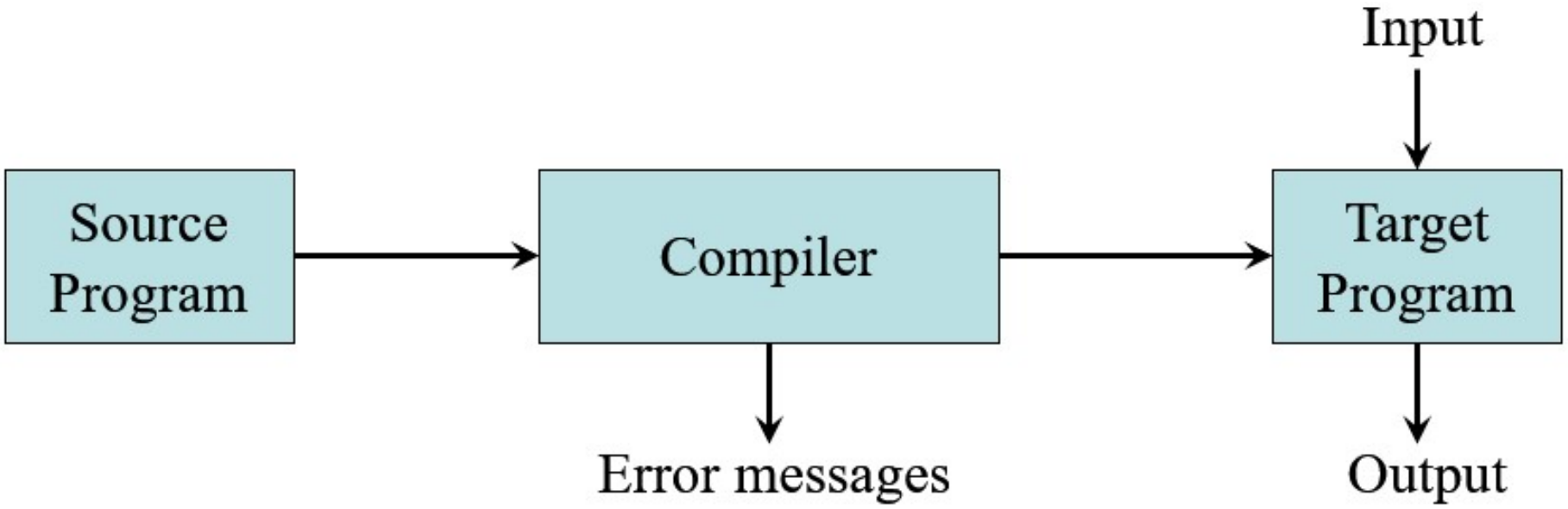
Compiled programs vs interpreters

- Interpretation
  - Performing the operations described by the source program
  - An extremely simplistic view



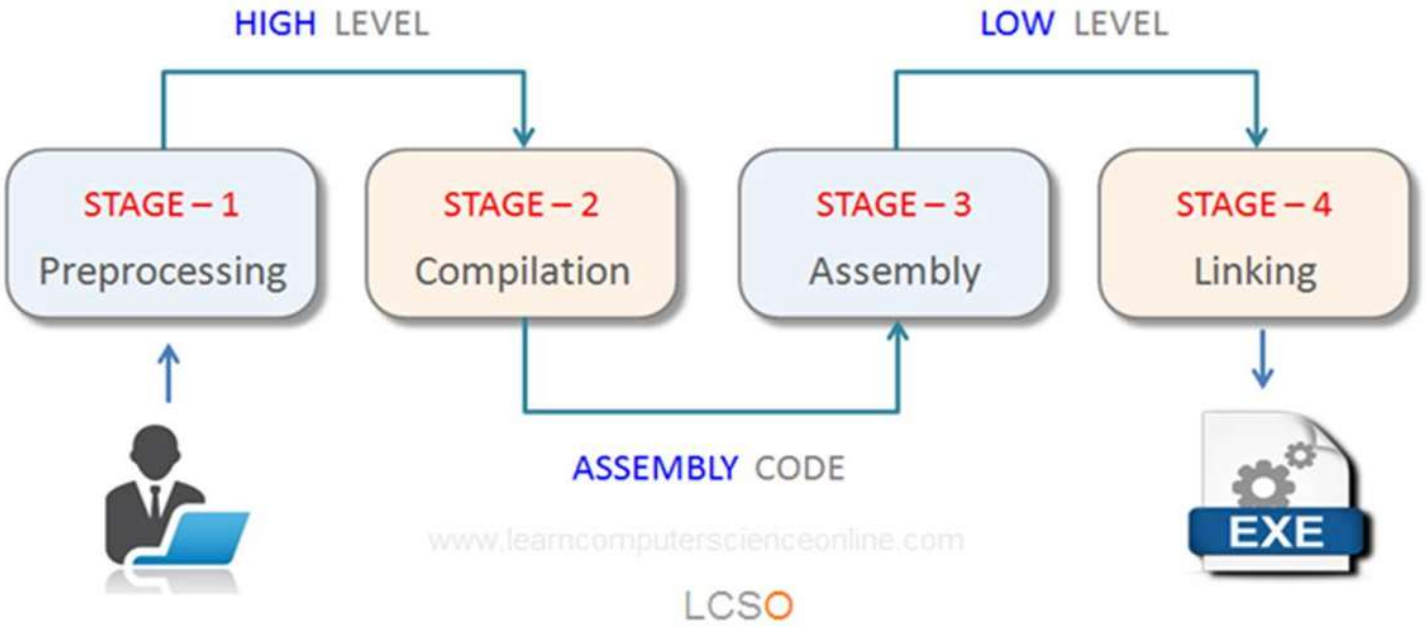
Compiled programs vs interpreters

- Compilation
  - Translation of a program written in a source language into a semantically equivalent program written in a target language
  - An extremely simplistic view



Compiled programs vs interpreters

- Compilation
  - Translation of a program written in a source language into a semantically equivalent program written in a target language
  - Another extremely simplistic view



How to compile a program

- Depends on the OS environment, platform, tools
- A very simplified approach on a Linux environment

```
$ gcc -o my_program my_program.c
```

- **my_program** is now an executable program
- An excellent compilation manager on Linux: **make**
- Version management tools (e.g., **git**) and services (GitHub) are essential

Other tools

- UML diagramming: Visual Paradigm
- Web programming: HTML, CSS, JS, PHP
- Database programming: SQL
- Container programming: Docker
- Editing and searching: Regular expressions
- Debugging

Summary

- **Efficiency and productivity**: Automate repetitive tasks, reducing manual effort and saving time
- **Consistency and accuracy**: Ensure uniformity in coding standards and reduce human errors
- **Collaboration and version control**: Facilitate teamwork and track changes effectively
- **Debugging and testing**: Simplify the process of identifying and fixing bugs
- **Innovation and creativity**: Free up cognitive resources, allowing engineers to focus on creative problem-solving