Into to Technology

John

Fachschaft General & Computational Linguistics
University of Tübingen

WS 2024/25
Pre-course

Outline

Intro to Commandline

When you turn your PC's brightness up so you can mine without having to place torches

What is a computer

- Your computer a big calculator
- It has a list of instructions and it executes it one by one

How to interact with the calculator

- Usually you interact with a gui
- What did they do before guis?
- They used the command line

```
 Sep 15:53 .
0. Sep 15:53 ..
9. Sep 2015 bin -> usr/bin
21. Sep 09:31 boot
19. Sep 15:50 dev
21. Sep 09:32 etc
30. Sep 15:52 home
30. Sep 2015 lib -> usr/lib
23. Sep 2015 lib64 -> usr/lib
1. Jul 10:01 lost+found
30. Aug 22:45 mnt
21. Sep 2015 opt
21. Sep 15:52 private -> /home/encryp
12. Sep 08:15 proc
21. Aug 15:37 root
30. Sep 15:50 run
21. Sep 2015 sbin -> usr/bin
21. Sep 15:51 srv
12. Sep 15:45 sys
12. Aug 15:39 tmp
30. Jul 10:25 usr
                var
```

What can you do with the Command Line?

- Short answer: Everything
- Simple examlpes: set a timer, do arithmetic, play a text based game
- Most importantly: run programs

The file system

- How does the computer know what to do?
- Files
- You should be familiar, when uploading photos
- A file is not just a photo, but any piece of data
- Example: Excel data, Word file, your Minecraft World
- Others: Settings, Programs, Folders, Mouse and Keyboard

Directories

- Basically file manager but no mouse.
- Files are the exact same, just a different way to access them.
- Example: `/Desktop/homework/assignment1.java`
- Instead of clicking through folders you have to type in which folder you want to switch to.
- The names of folders separated by slashes is the path.

Other things you can do

- Many gui actions have command line equivalents
- Opening a file it in notepad.
- Editing the file in Notepad
- Moving a file to the trash can
- Making a new file in Notepad

But do I really need this?

- In our program, mostly for Text Technology, running python, and doing ssh
- As a beginner, you can mostly get by by clicking the green button in VS Code
- Sometimes green button no work, so its good to know how to do it from the command line
- Make sure you do not develop command line phobia.
- No demo because too scary

Outline

What is Git?

git
I prefer the real version control
SUBVERSION
I said the *real* version control
Final 2
Final Final
FINAL FOR REAL THIS TIME
OK This is definitely final
Perfection

What is Git (for real)?

- *Git is a distributed version control system that tracks changes in any set of computer files, usually used for coordinating work among programmers who blah blah blah*
    - Basically google docs for code.
    - A command line program (but GUIs exist).
    - Most importantly, it's how you download code off of github and upload

github

- git hub is the website, git is the actual program you use
- As an analogy, git is like Word, and Github is the Google Drive.
- Many classes require github
- Most open source softwre is hosted on github
- follow me on github btw

But why can't I just use google drive/email/whats app/print out all my code on paper.

Why use Git?

- Keeps a history of all changes.
- Easy collaboration through branching and merging capabilities. (meaning two people can work on different parts of the code base at the same time)
- Distributed, meaning everyone that uses git has a copy of the source code, so you don't need internet to work on it.
- An industry standard that everyone expects you to know
- Can potentially show you did not cheat

Basic Git Commands

- `git clone` Download your file, you only do this once.
- `git add` Selects which changes you want to upload
- `git commit` Save your changes
- `git push` Uploads you changes to github

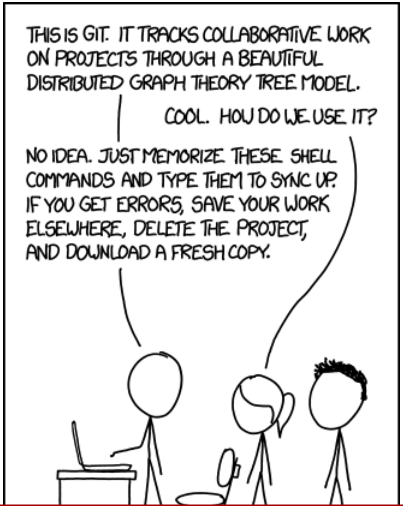
Workflow

- Do some work
- Save in your editor
- Git add your changes, (sometimes you don't want to add all your changes)
- Git commit, and give a message
- Git pull to see if anyone else made any changes. If there are changes you need to merge them.
- git push to see it on github.
- Repeat

But that’s too hard

- Download a program to help you
- github desktop, gitkraken,
- vscode also has git integration
- Alternatively, edit directly on github (not recommended)

Basically


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

Outline

But we didn’t start writing code yet

- Debugging is an important skill
- More time is spent debugging than writing code
- Some debugging techniques are general that don’t require use of a specfic tool

What is a bug?

- Basically, you are doing something wrong
- It's a lot easier writing wrong code than correct code.

Types of Errors

- Compile Errors
- Runtime Errors
- Logic Errors

Compile Errors

- Basically, you made a grammar mistake
- Computers are stubborn, they know what the error is but make you fix them
- Examples: Missing semicolons, undeclared variables, mismatched brackets, typos
- Usually there will be a red squiggly line
- Relatively easy to fix, usually you copied something wrong.

Runtime Errors

- Occur while the program is running.
- If compile error is a grammar mistake, a runtime error is a semantic mistake.
- Analogy, if you ask for the 10th person in line, but there are only 5 people in line.
- Examples: Dividing by zero, trying to access an out-of-bounds array index, the famous null pointer exception
- Usually your program crashes.
- There is usually a line number to see where it failed

Logic Errors

- The code runs, but doesnt do what you want it to do
- Examples: Incorrect formulas, missing steps in a process.
- Often the hardest to dear has made a mistake in their mental model.
- Analogy, you are baking, but the white powder you thought was sugar was salt.

Debugging Techniques

- Adding Print Statements
- Using Paper and Pencil
- Duck Debugging
- Using a debugger

Adding Print Statements

- Adding lines of code to display variables or messages.
- Allows you to see which code is being run, which code is not.
- Relatively simple, you learn hello world on the first day and that's all you need.

Using Paper and Pencil

- Write down variables and their values as you trace through the code.
- Allows for manual simulation of how the code runs.
- Sometimes writing stuff down just makes everything make sense

Duck Debugging

- Explain your code or problem out loud, as if to a rubber duck or inanimate object.
- Sounds really stupid, but sometimes works
- if you do this in public, you may get bullied

Debugger

- Will not go over, since it is technical
- Basically lets you go through your progarm line by line and obsveve variables
- Very powerful, learning how to effectively use is worth the effort

Other useful tips

- Google is your best friend
- Test individual subcomponents
- Sometimes the most simple bugs are the most hard to find.
- Go to sleep