Lecture 3

Organizing Code

- Introductions
- Lab 0 — Preview
- Workflows in CSCI 136
- Code Organization
  - `ssh` for secure remote connections
  - `nano` for console text editing
  - `git` for code management

Lab 0 — Preview

Lab 0 — Java/Git Intro

Labs are released online at 5pm on Tuesdays.

They are due the following Tuesday before 5pm.

- Try to look at the lab handout before your lab, and (ideally) start working on it.

You may want to practice visiting TAs and labs.

- TA rooms and hours are on Google Calendar.

The main goal of Lab 0 is to familiarize yourself with the computing environment.

- You will move between the Mac and Unix labs.

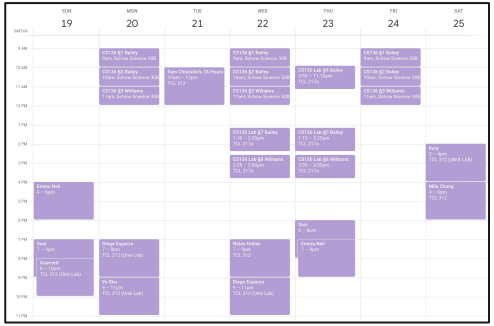
It also is chance for you to test your basic Java programming (e.g., writing *Hello, World!*).

- Try to do this without looking at your notes.

Laboratories

Description | Lectures | Resources

Date	Topic	Handout
September 14	Lab 0: Setting up a Git and Java Environment	Lab Handout



Laboratories link and TA schedule.

Computer Science CS136 (Fall 2021)
Duane Bailey & Aaron Williams
Laboratory 0
Setting up a Git and Java Environment

Objective. To set up a workflow for using Git and Java on CS lab machines.

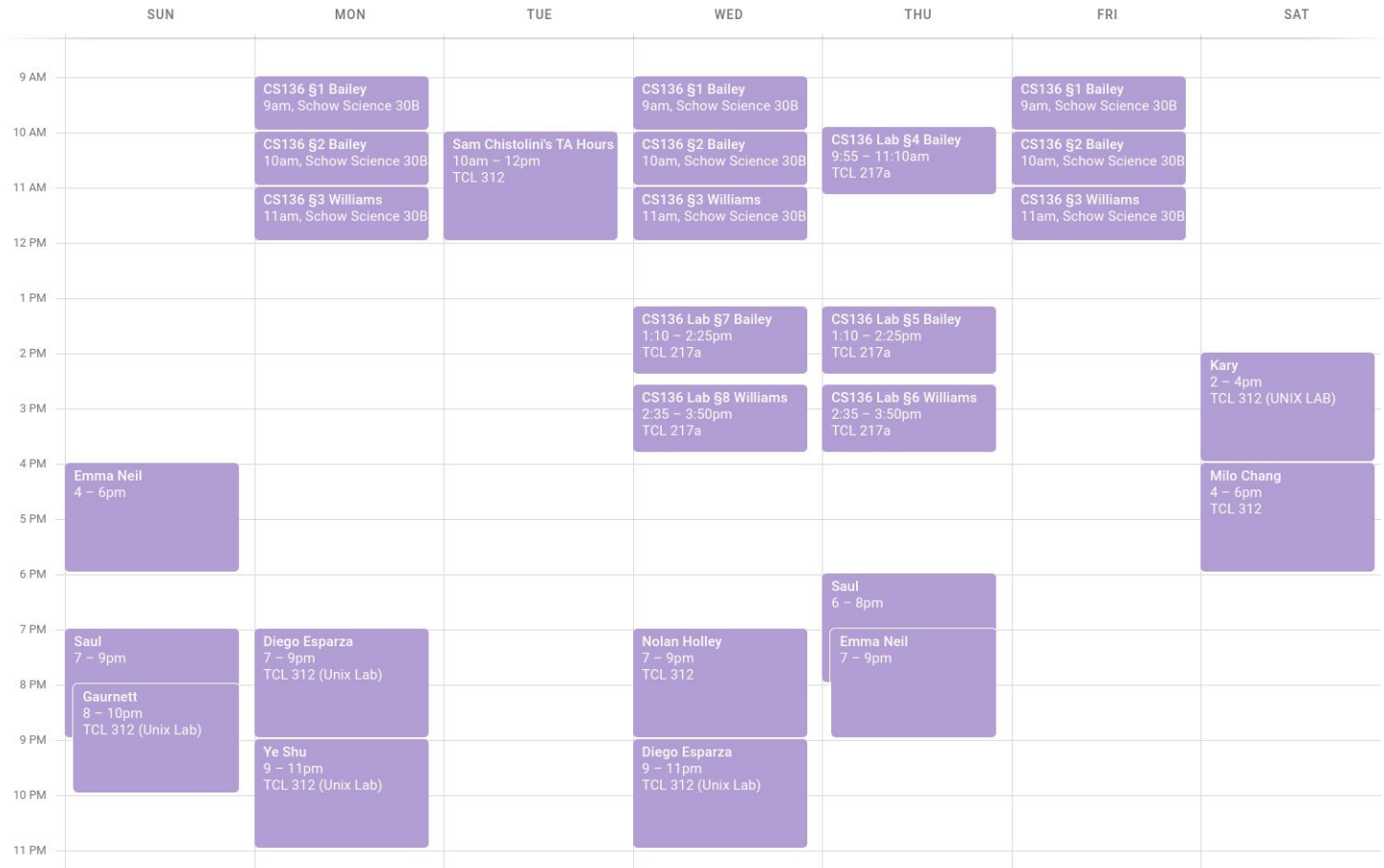
Overview. This semester, we will be making use of the computers in the various Computer Science labs:

"The Mac Labs" in Chemistry 216 and 217a. These labs are populated with Mac computers running macOS, a unix derivative. Your CS credentials will give you access to each Mac. The files are not shared between Macs: the work you perform on one computer is stored locally on that machine.

"The Unix Lab" in Chemistry 312. This lab is populated with Ubuntu, another unix derivative. Your CS credentials will give you access to each workstation. Files created on these machines are all shared: the work you perform on any of these computers can be accessed from any other. Because they are central to the department, each machine is named after a breed of cattle. Here are the current names of these machines:

amerifax bagual barzona brava charolais devon galloway guernsey kuri lidianiata panda rathi reina sharabi sind siri sykia tundaca zebu

Lab 0 handout has 16 steps across 6 pages.



Note: This weekly schedule is subject to change. Check Google Calendar for any updates.

Workflows in CSCI 136

Supported Workflows

①

Use a computer in the Mac lab (TCL 217A). Labs start here.

- Login with your Unix credentials (Lida Doret or Mary Bailey)
- Each Mac computer has its own local environment.
  - You will need to configure `git` on each one you use.
  - A file in your home folder only appears on one computer.
- Text editors with windows are available.
- Room is shared with CSCI 134.

②

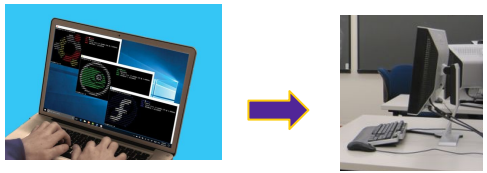
Use a computer in the Unix lab (TCL 312).

- Login with your Unix credentials (Lida Doret or Mary Bailey)
- Unix computers share their local environment.
  - You configure `git` once.
  - Files in your home folder are accessible on all.
- Only terminal-based editors are available.
- Room is <u>not</u> shared with CSCI 134.

③

Use a computer in the Mac lab and `ssh` to a Unix machine.

④

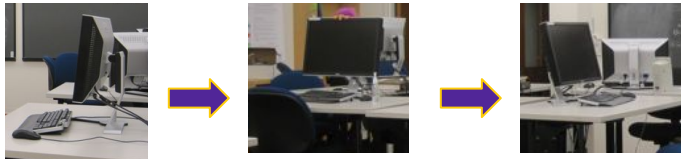


Using your own computer to `ssh` to a Unix machine.

- Terminal on Mac; WSL2 on Windows; Linux on Chrome OS.

Weird / Unsupported Workflows

⑤



Use a computer in the Unix lab to `ssh` to another Unix machine!

- You could do this multiple times.
- Please be reasonable!

Notes:

- The Mac machines cannot be logged into in this way.
-

⑥

Using your own computer (without `ssh` to a Unix machine).

- Use any text editor that you like.
- Not supported for many reasons:
  - We can't control which version of `java` you have.
  - Your computer could break down.
  - Security issues with certificates.

Note: You could use this approach for writing you code, but the teaching team won't answer any questions that you have if you are running it on your own computer, and your code will be tested and graded using our computing environment.

Code Organization

Code Organization

Java code allows for code to be organized in several ways.

- Instructions are organized into *methods* (a.k.a., functions). Examples from Lecture 1.
- *Classes* have methods and *attributes* (a.k.a., fields) and are stored in files. Examples from Lecture 2.
- *Packages* have classes and are stored in folders. Lecture 4 introduces Duane's `structure` package.

More broadly, code organization can refer to a number of higher-level concepts.

- *Version control*. The history of a file can help us find regressions (i.e., where bugs originated).
- *Collaboration*. Large projects involve many developers working on the same code base.
- *Branches*. New features need to be isolated during development.

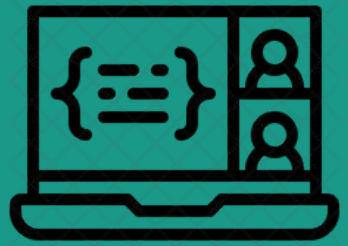
The industry's standard tool for the above points is `git`. We'll only use a fraction of its features.

Code organization can also refer to a number of concerns.

- Working on other computers using `ssh`.
- Configuring tools on a given computer.
  - Setting up `git`.
  - Configuring a text editor.

ssh

Live Coding: `ssh`



Let's login to one of the machines in the Unix lab.

- `ssh user@machine.cs.williams.edu`
  - `user` is your Unix credentials
  - `machine` is the (cow-based) name of one of the machines in the Unix lab
    - see Lab 0 handout for a list of cows (excluding those in the "Knuth lab" at the back of the Unix lab)
  - e.g., `ssh aaron@lohani.cs.williams.edu`
  - `exit` terminates the connection
  - the connection may terminate for other reasons (e.g., network goes down or the laptop lid is closed, etc.)
    - make sure to save your work periodically
- Take a look at the local manual page (`man`) and tldr page online.

Note: Summaries of Live Coding demonstrations will be added to the slides when they are posted to the course website.

```
~$ ls
Applications/           GitTemp1/
Creative Cloud Files/   GitTemp2/
Desktop/                GitWilliams/
Documents/              Google Drive/
Downloads/              Library/
Dropbox/                Movies/
GitHub/                 Music/
GitLab/                 Pictures/
GitLohani/              Public/
GitMIT/                 VirtualBox VMs/
GitSRC/                 temp/
~$ ssh aaron@lohani.cs.williams.edu
aaron@lohani.cs.williams.edu's password:
```

```
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-42-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Wed 15 Sep 2021 12:37:22 PM EDT

  System load:  0.0                Temperature:            39.0 C
  Usage of /:   67.3% of 439.11GB  Processes:              787
  Memory usage: 2%                 Users logged in:        2
  Swap usage:   0%                 IPv4 address for ens1f0: 137.165.8.10

 * Super-optimized for small spaces - read how we shrank the memory
   footprint of MicroK8s to make it the smallest full K8s around.

   https://ubuntu.com/blog/microk8s-memory-optimisation

48 updates can be applied immediately.
8 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable


The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Int
ernet connection or proxy settings


Last login: Wed Sep 15 11:31:52 2021 from 137.165.120.103
-> ls
cs136/    Documents/  GitLabEvolene/  lecture3.txt  Pictures/  Templates/  www/
Desktop/  Downloads/  GitTemp1/       Music/        Public/    Videos/
```

`ssh` from Terminal on Aaron's Macbook (left) to the Unix environment on `lohani`.
Note that the list of files (using `ls`) are different.

nano

```
NANO(1)                    General Commands Manual                    NANO(1)

NAME
       nano - Nano's ANOther editor, inspired by Pico

SYNOPSIS
       nano [options] [[+line[,column]] file]...

       nano [options] [[+[crCR](/|?)string] file]...

NOTICE
       Since version 4.0, nano by default:

           • does not automatically hard-wrap lines that become overlong,
           • includes the line below the title bar in the editing area,
           • does linewise (smooth) scrolling.

       If you want the old, Pico behavior back, you can use --breaklonglines, --emptyline, and --jumpyscrolling (or -bej for short).

DESCRIPTION
       nano  is  a small and friendly editor.  It copies the look and feel of Pico, but is free software, and implements several features
       that Pico lacks, such as: opening multiple files, scrolling per line, undo/redo, syntax coloring, line numbering,  and  soft-wrap-
       ping overlong lines.

       When  giving  a  filename on the command line, the cursor can be put on a specific line by adding the line number with a plus sign
       (+) before the filename, and even in a specific column by adding it with a comma.  (Negative numbers count from  the  end  of  the
       file  or  line.)  The cursor can be put on the first or last occurrence of a specific string by specifying that string after +/ or
       +? before the filename.  The string can be made case sensitive and/or caused to be interpreted as a regular expression by  insert-
       ing c and/or r after the + sign.  These search modes can be explicitly disabled by using the uppercase variant of those letters: C
       and/or R.  When the string contains spaces, it needs to be enclosed in quotes.  To give an example: to open a file  at  the  first
       occurrence of the word "Foo", one would do:

           nano +c/Foo file

       As a special case: if instead of a filename a dash (-) is given, nano will read data from standard input.
```

```
                [ Welcome to nano.  For basic help, type Ctrl+G. ]
^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify
```

> tldr _ v3.8.0

nano

Star 1,419

If this web site has been useful to you, consider supporting me on Patreon

nano

Simple, easy to us
enhanced, free Pic
HTTPS://NANO-E

Open a new file in nar

nano

Open a specific file, positioning the cursor at the specified line and column:

```
nano +{{line}},{{column}} {{path/to/file}}
```

Open a specific file and enable soft wrapping:

```
nano --softwrap {{path/to/file}}
```

Open a specific file and indent new lines to the previous lines' indentation:

```
nano --autoindent {{path/to/file}}
```

Open nano and create a backup file (`file~`) when saving

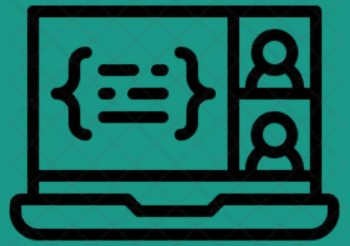
edits:

```
nano --backup {{path/to/file}}
```

`nano` is one of the the simplest **terminal-based** text editors.

- Learn more: `man nano` or `tldr nano` (online version: tldr.sh) or `Ctrl+g` in the program.
- For configuration refer to the `~/.nano` folder and the `~/.nanorc` file.
- Our Unix machines have Version 4+ but the Mac machines may only have Version 2.
- Other terminal options: `emacs` or `vi(m)`. Atom is an excellent non-terminal text editor.

Live Coding: Configuring `nano`



Let's make the following changes to the default behavior of `nano`.

- Change the length of the tabs.
  - By default a tab is 8 spaces in `nano` (yikes!).
- Tell nano to support mouse clicks.
  - By default it doesn't listen to the mouse (yikes!).
- Take a look at the local manual page (`man`) and tldr page online.

Note: Most programs in Unix / Linux can be configured in similar ways.
For example, git uses `.git` folders.

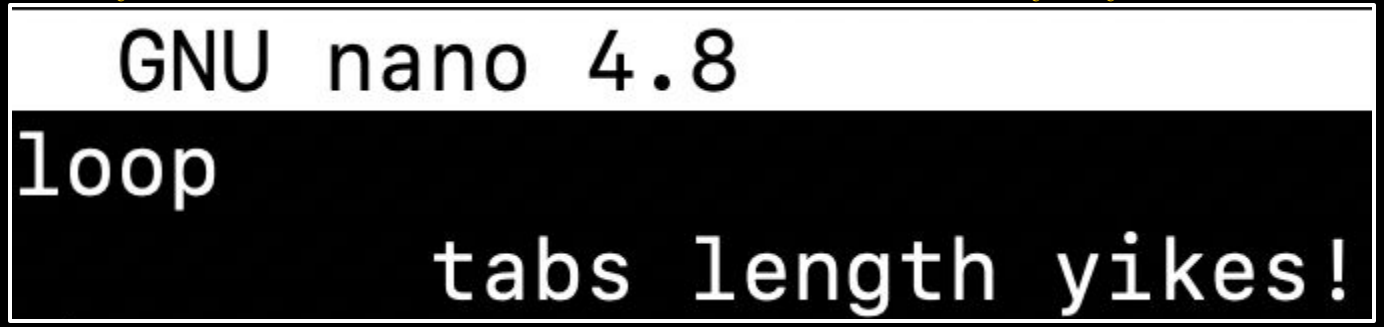
```
-> cd ~
-> nano .nanorc
-> cat .nanorc
set mouse
set tabsize 4
```

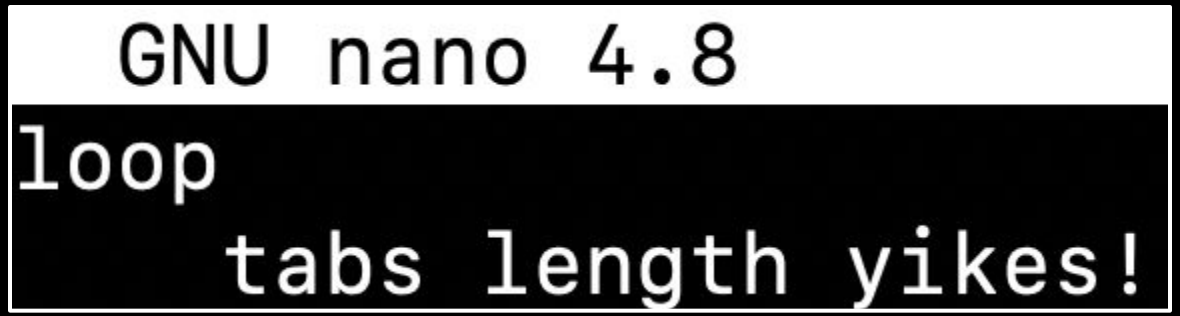
```
  GNU nano 4.8
set mouse
set tabsize 4
```

```
^G Get Help   ^O Write Out
^X Exit       ^R Read File
```

```
File Name to Write: .nanorc
^G Get Help      M-D DOS Format
^C Cancel        M-M Mac Format
```

Move to your home folder with `cd ~`
Then edit a file called `.nanorc` by running `nano .nanorc`
Add the `set mouse` and `set tabsize 4` lines and save it with Ctrl+O (`^O`).
Exit `nano` with `Ctrl+X` then check the file contents using `cat .nanorc`



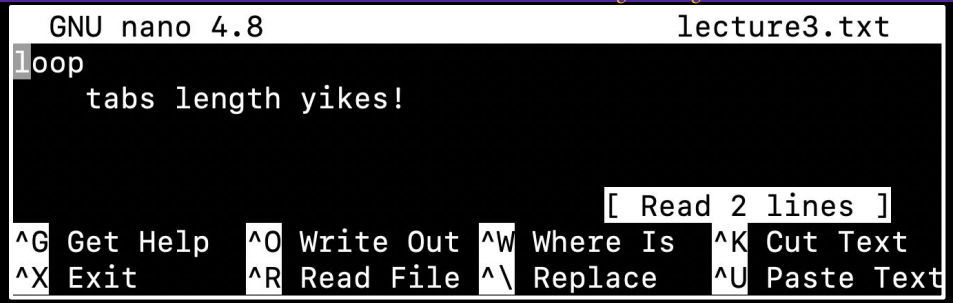


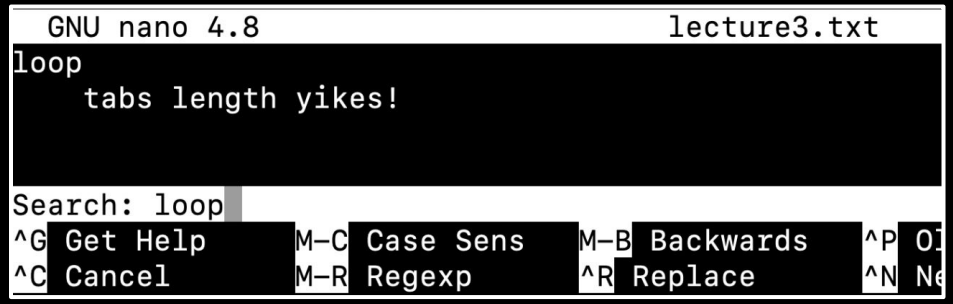
After the change, the tab length has changed from 8 spaces to 4 spaces.

```
  GNU nano 4.8
loop
        tabs length yikes!
```

```
  GNU nano 4.8
loop
    tabs length yikes!
```

After the change, `nano` displays tabs as 4 spaces instead of 8 spaces.



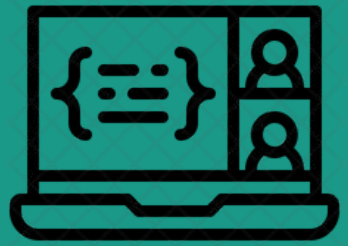



```
-> cd ~
-> cat .nano/search_history
loop
yikes
```

Search for the text `loop` and `yikes` in `nano` using `Ctrl+w`
Then the file `~/.nano/search_history` will be updated accordingly.

git

Live Coding: `git`



The basics of `git` that are used in this course.

- Configuring `git` on your machine.
  - `git config` use option `--list` to see current settings
- Creating a local copy of an existing git repository.
  - `git clone`
  - in this course we'll use username / passwords for security; GitHub now only allows ssh keys
  - side note: creating a new repository can be done with `git init` but you will not need to do this in CSCI 136
- Download changes to the repository.
  - `git pull`
- Making changes to repository.
  - `git add` on each file or file(s) that you have changed
  - `git commit -m` including a commit message
  - `git add` on each file or file(s) that you have changed
  - `git push` including `origin main` or `origin main` as needed
- Checking on the repository.
  - `evolene.cs.williams.edu` for the repositories in this course
  - `git status` for the status of a repository