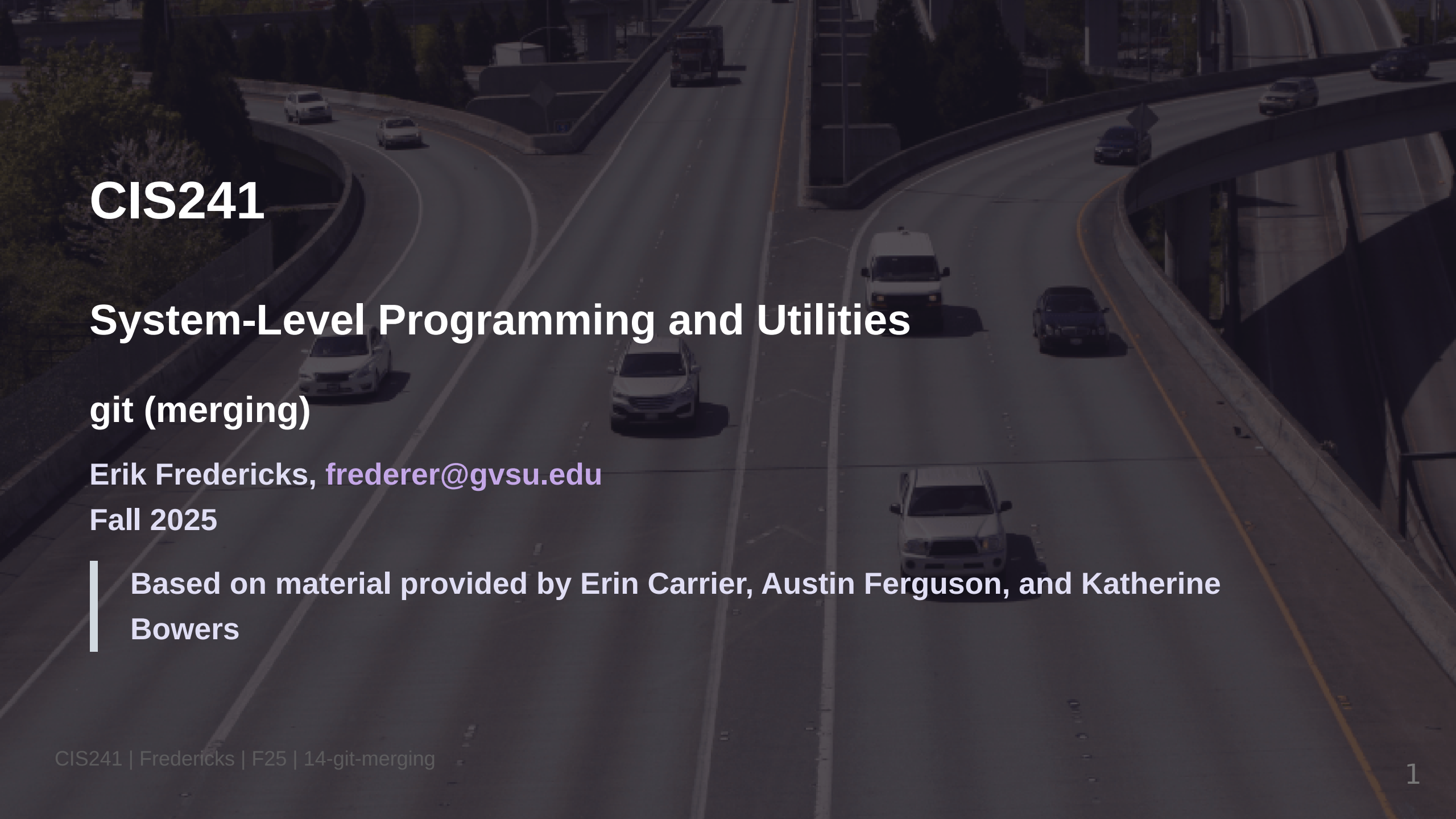




CIS241

System-Level Programming and Utilities

git (merging)

Erik Fredericks, frederer@gvsu.edu

Fall 2025

Based on material provided by Erin Carrier, Austin Ferguson, and Katherine Bowers

A background image of the character Anger from the Pixar movie Inside Out. He is a red, flame-like figure with a large, angry expression, wearing a white shirt and a striped tie. The image is darkened to serve as a background for the text.

What is a merge conflict?

Merge conflict: when two commits compete

- People editing the same line
- File both edited and deleted

Resolving

Manually edit files and decide which to keep

Markers indicate who did what

- Between `<<<<<<` and `=====`
 - local branch merging into
- Between `=====` and `>>>>>>`
 - remote branch being merged

Though, hard to read!

- Good use case for a visual diff program...

```
# No conflict
<<<<<<
This is from local branch (one we're merging into)
=====
This is from remote (one being merged in)
>>>>>>
# Conflict over
```

Regardless...

- Decide which to keep or combine in some way
- Replace with whatever decided
- Remove markers
- Repeat for all conflicts in each conflicting file
- `Add` files after resolving merge
- Run `git commit`

Resolving:

Q1: What if I just want to keep one version or the other, is there a better way?

- A1: Two options:
 - `git restore --ours filename` or `git restore --theirs filename`
 - Or, replace `restore` with `checkout`
 - `ours` = branch merging into
 - `theirs` = branch being merged in

Resolving:

Q2: I screwed up the merge - can I start over?

- A2: `git merge --abort`
 - May delete any changes in working directory!!

Resolving

Q3: This is awful - is there another way?

- A3: Yes, but not the 'ideal' way
 - As before, pull down a "clean" copy of the repo into a new folder and merge your local changes into that
 - Then decide which folder to keep

Example of inducing a merge conflict

c/o reddit:

https://www.reddit.com/r/git/comments/b3wm1j/how_to_best_intentionally_create_a_merge_conflict/

```
git init
echo 'Mr. Foo' > foo.txt
git add .
git commit -m 'foo'
```

```
echo 'Mr. Bar' > foo.txt
git add .
git commit -m 'bar'
```

Branch from HEAD~1 - a.k.a. 1 commit ago.

```
git checkout HEAD~1
git checkout -b branch
echo 'Mr. Baz' > foo.txt
git add .
git commit -m 'baz'
```

```
git checkout master
git merge branch
```

Also, visualize:

```
git log --all --date-order --graph
```