CSE306 Software
Quality in Practice

Dr. Carl Alphonce
alphonce@buffalo.edu
343 Davis Hall
understanding branching


- `git branch <name>` creates a new pointer
Let's take this as our starting point.
git checkout B
create fileB
git add fileB
git commit -m "..."
git checkout master
EXERCISE
Let us first switch to branch 'a':

```
git checkout a
```
Index staging

Workspace
- README.md
- F1
- F2

Local Repository
- HEAD
  - b
    - commit
      - tree
        - F4
        - F3
        - F2
        - F1
  - a
    - commit
      - tree
  - master
    - commit
      - tree
        - README.md
Now edit and commit F2.

```sh
emacs F2 (call edited version F2')
git add F2'
git commit -m "..."
```
Create F5, add, and commit.

```bash
emacs F5
git add F5
git commit -m "..."
```
Switch to master.

`git checkout master`
Merge a into master.

git merge a
HEAD

master

F1

F2'

F6

local repository

workspace

index

staging

README.md

F1

F2'

F6

README.md
Because there was a straight line path from master to a, this is a "FAST FORWARD merge: master is simply moved to point to the same commit as a."
Non-FF merge
(three way merge)

- A new commit is created.
- The branch being merged into is the first parent of the new commit.
- 'master' should always be the first parent, hence switch to master before doing merge.
The non-fast-forward merge creates a "merge commit". The branch merged into points to that new commit.
Deleting a branch

```bash
git branch -d a
```
Switch to b, edit F2 to create conflict (edit same part of F2 as you did on branch a, but change it in a different way), merge b into master (switch to master, then ‘git merge b’).
STARTING POINT
git checkout b
edit F2 to create conflict with F2'
git add F2"
```bash
git commit -m "..."
```
git checkout master
Blob created with conflict markings.

`git merge b`
edit F'' to resolve conflict, creating F2^4
```
git commit -m "..."
```
Question

- Is there a difference between
  - merging a into b
  - merging b into a
Question

- Is there a difference between
  - merging a into b
  - merging b into a

YES!
merging b into master
master moves, b does not
merging master into b
b moves, master does not
also:
parent order differs
The parents of a commit are ordered.

A = A^0
B = A^1 = A~1
C = A^2 = A^2
D = A^1^1 = A~2
E = B^2 = A^2
F = B^3 = A^3
G = A^1^1^1 = A^3
H = D^2 = B^2 = A^2^2 = A~2^2
I = F^2 = B^3 = A^3^2
J = F^3 = B^3^2 = A^3^2

merging b into master
HEAD^1 → C, HEAD^2 → B
merging master into b
HEAD^1 → B, HEAD^2 → C
More gdb commands

C-x C-a toggle between a "graphical" and line-based UI

watch <variable>

Looking at source code:
- list line#
- list function
- disassemble /m

Looking at data:
- print
- examine (x)

https://sourceware.org/gdb/current/onlinedocs/gdb/Arrays.html#Arrays
```c
#include <stdlib.h>
#include <string.h>
#include <stdio.h>

int main(int argc, char * argv[]) {

    if (argc !=2) {
        printf("Please give one numeric argument.\n");
        return 1;
    }

    int limit = atoi(argv[1]);

    // serenity\0
    // 012345678

    char * string,* name;
    name = malloc(3 * sizeof(char));
    string = malloc(9 * sizeof(char));
    name[0] = '@';
    name[1] = '$';
    name[2] = '\0';
    string[0] = 's';
    string[1] = 'e';
    string[2] = 'r';
    string[3] = 'e';
    string[4] = 'n';
    string[5] = 'i';
    string[6] = 't';
    string[7] = '\0';
    string[8] = '\0';
    printf("string has length %d and is %s.\n",strlen(string),string);
    printf("name has length %d and is %s.\n",strlen(name),name);
    for (int i=3; i<limit; i++) {
        name[i] = (char) ('a'+((i-3)%26));
    }
    name[limit] = '\0';
    printf("string has length %d and is %s.\n",strlen(string),string);
    printf("name has length %d and is %s.\n",strlen(name),name);
    return 0;
}
```