Lecture #35

se Study in System and Data-Structure Design

- ibuted version-control system, apparently the most pop- currently.
- , it stores snapshots (*versions*) of the files and direc- re of a project, keeping track of their relationships, es, and log messages.
- *ted,* in that there can be many copies of a given repos- upporting indepenent development, with machinery to reconcile versions between repositories.
- n is extremely fast (as these things go).

A Little History

- Linus Torvalds and others in the Linux community when r of their previous, propietary VCS (Bitkeeper) with- e version.
- nentation effort seems to have taken about 2–3 months, he 2.6.12 Linux kernel release in June, 2005.
- ame, according to Wikipedia,
  - ds has quipped about the name Git, which is British ang meaning "unpleasant person". Torvalds said: "I'm ical bastard, and I name all my projects after myself. ux', now 'git'." The man page describes Git as "the tent tracker."
- a collection of basic primitives (now called "plumbing") scripted to provide desired functionality.
- -level commands ("porcelain") built on top of these to nvenient user interface.

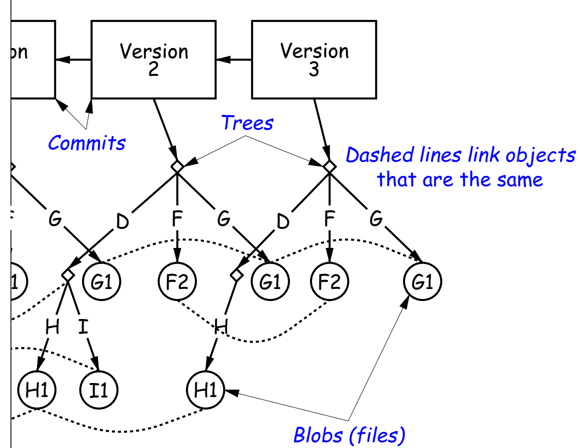
Major User-Level Features (I)

- is of a graph of versions or snapshots (called *commits*) e project.
- ructure reflects ancestory: which versions came from
- contains
  - y tree of files (like a Unix directory).
  - on about who committed and when.
  - ge.
  - o commit (or commits, if there was a merge) from which it was derived.

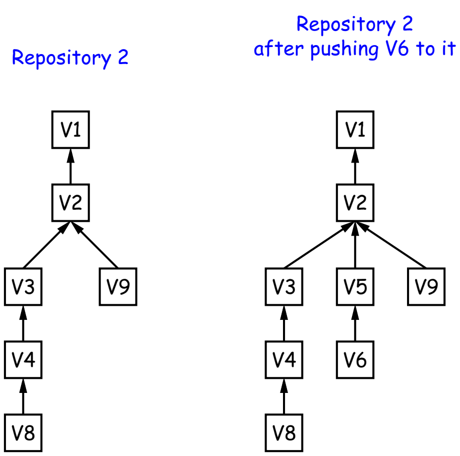
Conceptual Structure

- l components consist of four types of *object:*
  - ically hold contents of files.
  - ectory structures of files.
  - Contain references to trees and additional information r, date, log message).
  - ferences to commits or other objects, with additional on, intended to identify releases, other important ver- various useful information. (Won't mention further to-

Commits, Trees, Files



rsion Histories in Two Repositories



Major User-Level Features (II)

has a name that uniquely identifies it to all versions.

can transmit collections of versions to each other.

a commit from repository A to repository B requires nsmission of those objects (files or directory trees) not yet have (allowing speedy updating of repositories).

maintain named *branches*, which are simply identifiers commits that are updated to keep track of the most its in various lines of development.

s are essentially named pointers to particular commits. branches in that they are not usually changed.

Internals

ository is contained in a directory.

nay either be *bare* (just a collection of objects and r may be included as part of a working directory.

the repository is stored in various *objects* correspond- or other "leaf" content), trees, and commits.

e, data in files is *compressed*.

age-collect the objects from time to time to save addi-

The Pointer Problem

it are files. How should we represent pointers between

ble to *transmit* objects from one repository to another nt contents. How do you transmit the pointers?

transfer those objects that are missing in the target How do we know which those are?

counter in each repository to give each object there a But how can that work consistently for two indepen- ories?

ontent-Addressable File System

ne way of naming objects that is universal.

names, then, as pointers.

Which objects don't you have?" problem in an obvious

, what is invariant about an object, regardless of repos- *contents*.

the contents as the name for obvious reasons.

hash of the contents as the address.

at doesn't work!

a: Use it anyway!!

How A Broken Idea Can Work

o use a hash function that is so unlikely to have a colli- can ignore that possibility.

ic Hash Functions have relevant property.

ion, f, is designed to withstand cryptoanalytic attacks. , should have

resistance: given $h = f(m)$, should be computationally to find such a message m.

e-image resistance: given message m_1, should be infea- nd $m_2 \neq m_1$ such that $f(m_1) = f(m_2)$.

esistance: should be difficult to find *any* two messages uch that $f(m_1) = f(m_2)$.

properties, scheme of using hash of contents as name is likely to fail, even when system is used maliciously.

SHA1

A1 (Secure Hash Function 1).

und with this using the `hashlib` module in Python3.

ames in Git are therefore 160-bit hash codes of con-
.

commit in the shared CS61B repository could be fetched
vith

```
ckout e59849201956766218a3ad6ee1c3aab37dfec3fe
```