4156 9/13/16

clarification about first HW -
due Sept 20, in one week
(some students have already submitted)
ignore the "available until" date
on all assignments unless told otherwise

introduce team composition assignments
due Sept 22, next week on Thursday
(already some submissions here too)

need to create team groups
in canvas & piazza

Scott Lennon speaks about
SE & life

git handouts

everyone get account &
go through tutorial

4156 9/13/16

version control & continuous integration
mostly version control
more continuous integration
later in semester
(aka configuration management)

probably everyone in this class has
used some version control system

- many have already use git, svn, etc.
- some may only have used "weak"
  version control

for example, you have files named

foo.bar	foo.11Sep16.bar
foo.bar-v2	foo.bar.old
foo.v2.bar	foo.bar.keepme

keeps you safe
can undo
mistakes

and your operating system, or
a third party provide,
supports automate or manual
backup & restore of a previous version

dropbox, google drive, etc.
support access to older versions
& the history of changes

4156 9/13/16

let's consider "strong" version control
(file database)

purposes -

imagine that Microsoft has a folder named Windows 10 that everyone updates

- coordinate source code & other project resources among groups of developers

- "system of record" for code that goes into production

you need to know exactly what you shipped in order to reproduce & fix bugs, since users will report errors in deployed version not what you're working on now

- "logically" centralized storage independent from developers' machines

may be organized centrally or distributed

- allows for automated builds & tests (continuous integration) & sometimes deployment (continuous delivery) on demand or at preset times

4156 9/13/16

basic functionality (more details coming)

- backup & restore
- synchronization
  - two models — need way to break lock
    - LOCO - lock on checkout
    - MOM - merge on modify
- short term undo
  - return to last known good version
- long term undo
  - return to old version as of specific date, specific release, etc.
- track changes
  - commit messages
- track ownership
  - automatically tags who made each change

4156 9/13/16

sandboxing

insurance against yourself

branching & merging

fork copy of code base & track changes separately
may later merge back to main line

tagging

name the set of all file revisions contributing to some distinguished milestone, e.g., demo or release

might checkpoint separately
snapshot

4156 9/13/16

basic terminology

repository (repo) - file database

server - where the repo lives

client - developer machine

may use command line shell,
special GUI client,
snap in to file system,
plugin to IDE or editor

working set / working copy

local file directory where
developer makes changes

trunk / main

primary series of versions in
the repository

think of a tree

what to keep in repository?
source, config, tests, scripts, resources
usually <u>not</u> executables (generated files)

4156 continue here 9/13/16

basic actions

add file to repo

can't just add to local folder, need to tell VCS to track

revision number

Some VCS increment # per file, some per change set, some per whole repository

head - latest revision

checkout - download from repo

as mentioned earlier, 2 models - LOCO & MOM
pessimistic vs. optomistic

some VCS distinguish read-only vs. editable CO

problems/limitations with both

accidental vs. intentional overrides

checkin - upload changed files to repo
updates revision number
here is where most of those problems actually arise

4156 9/13/16

checkin message / commit message

changelog / history - might be integrated
with issue tracker

update / sync

get latest versions of files
be careful not to overwrite
any local changes (e.g. stash)

revert

throw away local changes +
reload latest version from repo

advanced actions

branch - both verb & noun

diff / change / delta - find differences
between 2 files, usually
sequential revisions

many different diff algorithms
also used for minimizing VCS storage

4156 9/13/16

merge/patch (automatic or manual)

integrate changes from one revision of a file into another

reverse vs. forward integration

also applies to full branches

conflict (detected automatically)

when pending changes to a file contradict each other

usually purely lexical, e.g., same line or same method

ideally semantic - needs static analysis such as slicing

resolve (usually manual)

fix conflicting changes & checkin corrected version

4156 9/13/16

now, finally, continuous integration

whenever anything is checked
in to a given branch, or
at pre-designated time (12 midnight)

external tools are hooked in +
automatically invoked

typically at least system build

developers don't have to remember commands

compile, package, etc. script

possibly also unit + system tests

maybe various static analyzers

perhaps even install (devops)
or ship (external customers)

various challenges:
do checkout/update wait for
CI to finish?
who broke the build?
some test suites take > 12 hours to run

integration or not with IDE