Source Control Lecture	CS 190
Jimmy Ho	Computer Science
03MAR99	Brown University

1 Revision Control Lecture

1.1 Goal

1.2 Problems

- Concurrent modifications
- Components in flux
- Bug tracking

1.3 RCS—Revision Control System

1.3.1 Getting Started

```
% cd /.../tetris

% mkdir RCS (or 'ln -s /../RCS RCS')

% ci *.java
```

1.3.2 Modifying Files

```
% cd /.../tetris

% co RCS/*

% co -l main.java

  <work>

% ci main.java

  <write log>

% rcsclean
```

1.3.3 Revision Numbers and Branching

1.1

1.2

⋮

1.17

`ci -r2.1 <files>`

2.1 —— 2.1.1.1

2.2   2.1.1.2

2.3

⋮

Branch Name = 2.1.1
Revision Name = 2.1.1.x
`'co -r2.1.1' yields revision 2.1.1.2`

1.3.4 References

```
% man rcsintro
```

```
% man ci
```

```
% man co
```

1.4 CVS—Concurrent Revisions System

1.4.1 Suppose...

```
nachos/
  bin/
  docs/
  src/
    threads/
    machine/
    lib/
    userprog/
    main/
```

Multiple projects?

1.4.2 Features

- Single repository
- No locks!

1.4.3 Example Session

```
% cd /.../workspace

% ls

% cvs checkout NACHOS

% ls

  nachos/

% cd nachos/main

  <modify main.cc>

% cvs commit main.cc

  <error>

% cvs update main.cc

  <resolve conflicts>

% cvs commit main.cc

% cvs release -d NACHOS

% ls

%
```

`nachos/main/main.cc`

2.3 $\longrightarrow$ you

2.4

2.5

Sample Conflict

```
<<<<<<<< main.cc
return(true);
========
return(false);
>>>>>>>> 2.6
```

1.4.4 Branching & Tagging

Can branch, tag and nest whole modules.

1.5 Conclusions

RCS	CVS
(+) simple	(+) more powerful
(–) simple	(–) more complicated
locking	merging

* Concurrent modifications.
* Stable components.
* Revision tracking/logging
* Branching
* Ease of use (e.g. getting working copies).