Week 10 Lecture 1

Software Configuration Management

Software Configuration Management

Software Configuration Management

- SCM keeps versions of software
- Versions are important because people makes mistakes
  - You delete a file you didn't mean to
  - You change the program to where it doesn't work
  - …
- It is helpful to go back to an earlier state

SCM in Production

- Production programs are written by teams
- SCM is even more important in team situations
- The person who broke the code can best fix it
  - But who broke the code
  - Only SCM knows for sure.
- SCM ensures that you always have working code.

Vocabulary

- *File*: object under control
- *Version*: a particularly instance of a file that has been checked in
- *Repository*: a collection of files and their versions.
- *Check in*: putting an instance of a file in a repository.
- *Check out*: retrieve a version of a file
- *Branch*: a sequence of version

SCM Programs

- Capabilities
  - Add item to track
  - Commit item to a backup
  - Retrieve an item from backup
  - Create branch
  - Merge branches

SCM Programs

- RCS
  - Allows people to keep versions of files
- CVS, Subversion
  - Allows people to track versions of projects
  - CVS is the original and commonly used but is not currently being developed.
  - Subversion is currently most popular
  - Stores changes
- Git
  - Designed to support Linux development
  - Stores complete files
  - Allows multiple clones of repositories

Git

- We will use Git in this class
  - It is currently very popular
  - Its distributed capabilities are a little better than Subversion
  - Disk space is cheap
- There is a hosting site called github.com
  - Provide free hosting for open source
  - Source for much open source software

To use Git

- Set up your name and email in Git
  - Lets others know who you are

```
student> git config --global user.name "Student"
student> git config --global user.email "student@mru.edu.in"
```

- The values are stored in .gitconfig

```
[user]
        email = student@mru.edu.in
        name = Student
../.gitconfig (END)
```

Set up a Git repository

- Got to the directory and initialize the git repository

```
student> pwd
/home/student/project1
student> git init
Initialized empty Git repository in /home/student/project1/.git/
student> ls -la
total 20
drwxrwxr-x 3 student student 4096 Jul  8 16:36 .
drwxr-xr-x 6 student student 4096 Jul  8 16:29 ..
drwxrwxr-x 7 student student 4096 Jul  8 16:36 .git
-rw-rw-r-- 1 student student   94 Jul  8 16:16 hello.c
-rw-rw-r-- 1 student student  117 Jul  8 16:16 Makefile
```

Add the files and commit them

- Add the files to the repository using git add
  - This does not store anything; it tells git that you want to include these files when you do store.
- Commit the files using git commit
  - This creates a new version and stores the files in it.

```
student> git add hello.c
student> git add Makefile
student> git commit -m "Initial commit"
[master (root-commit) f0962cb] Initial commit
 2 files changed, 16 insertions(+)
 create mode 100644 Makefile
 create mode 100644 hello.c
```

Commit Message

- It is hard to remember what each version contains
  - When you commit, and editor opens up so you can enter a message to help you find the right version
  - The more detailed your message, the easier it is to find the version if you need it

Retrieving a file

- You can get a list of the version using git log
- To get a file back, use git checkout
-

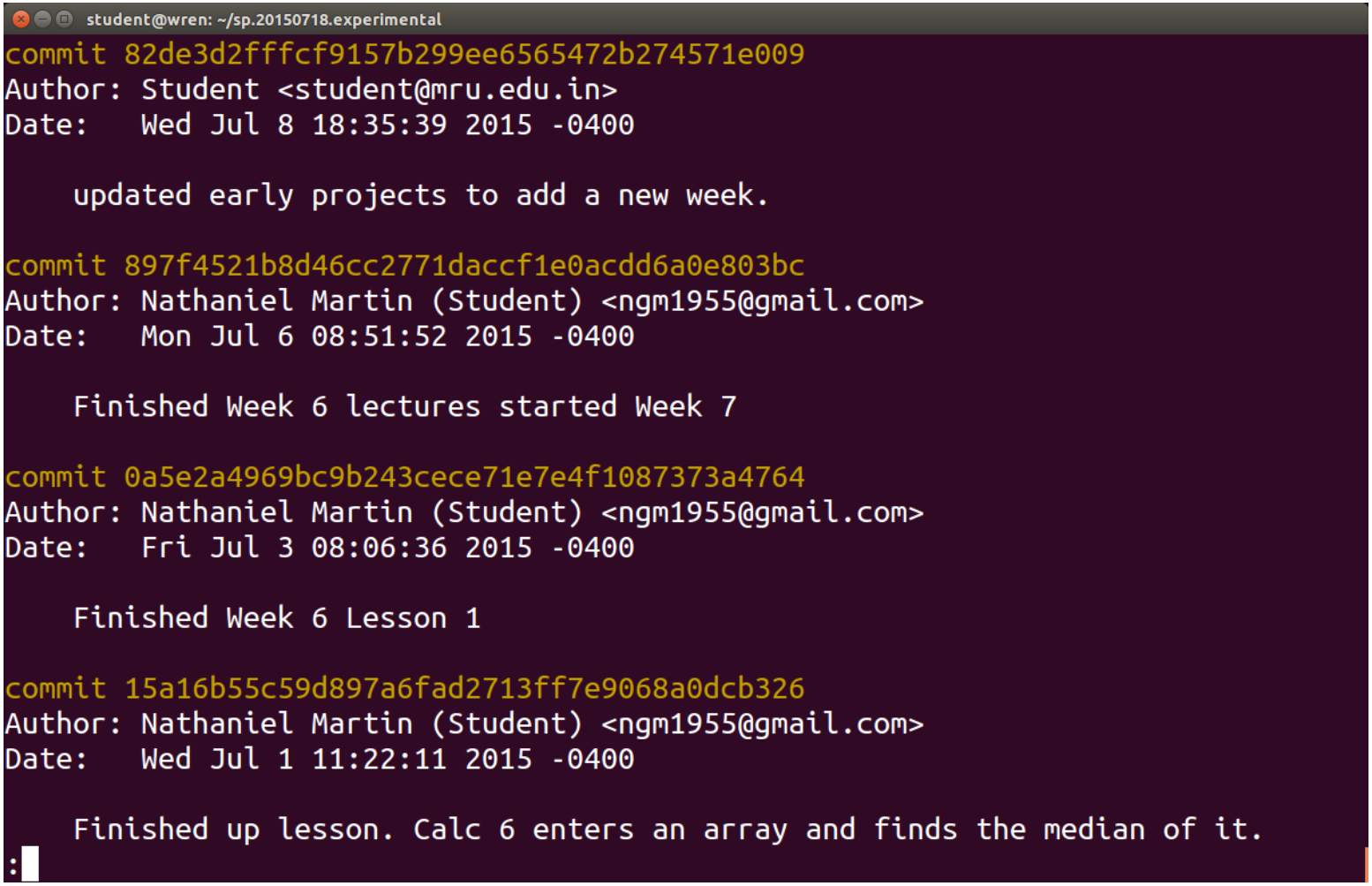
```
student> ls
hello.c  Makefile
student> rm hello.c
student> git log
commit bd1bb485ca54dfb9cbd23f97d040b9e29e0aa55b
Author: Student <student@mru.edu.in>
Date:   Wed Jul 8 16:38:18 2015 -0400

    Initial Commit
    - This directory contains the files for hello world
student> ls
Makefile
student> git checkout -- hello.c
student> ls
hello.c  Makefile
```

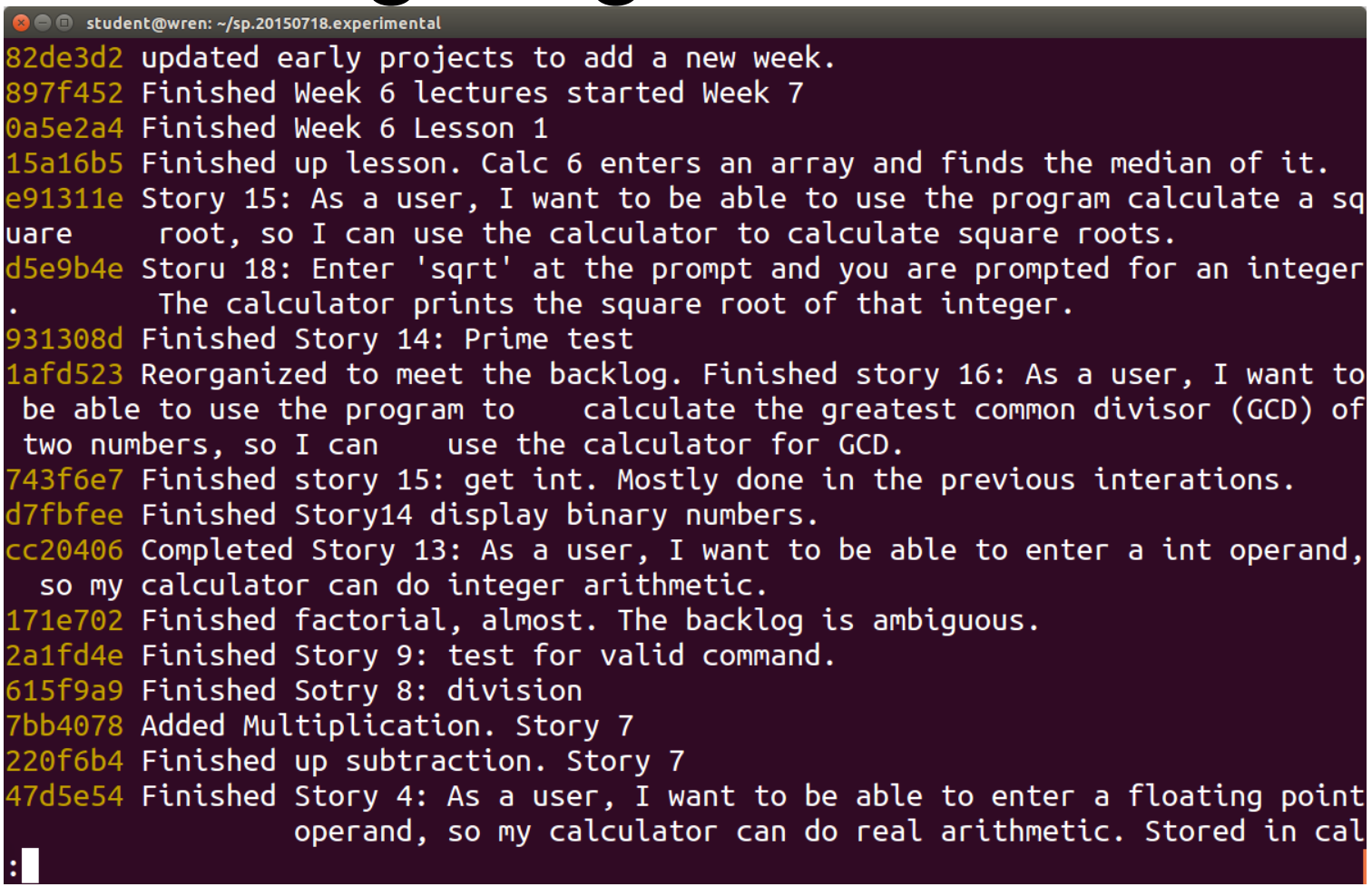
List Revisions: git log

- List revisions from most recent to oldest.
- Page though like less
  - <enter>: down one line
  - <space>: down one page
  - 'b': back one page
  - 'q': quit
- Refer to revisions by SHA-1
  - Long number following “commit”
  - Need only first four or so characters
- Easier to find stuff: git log --oneline

Example: git log



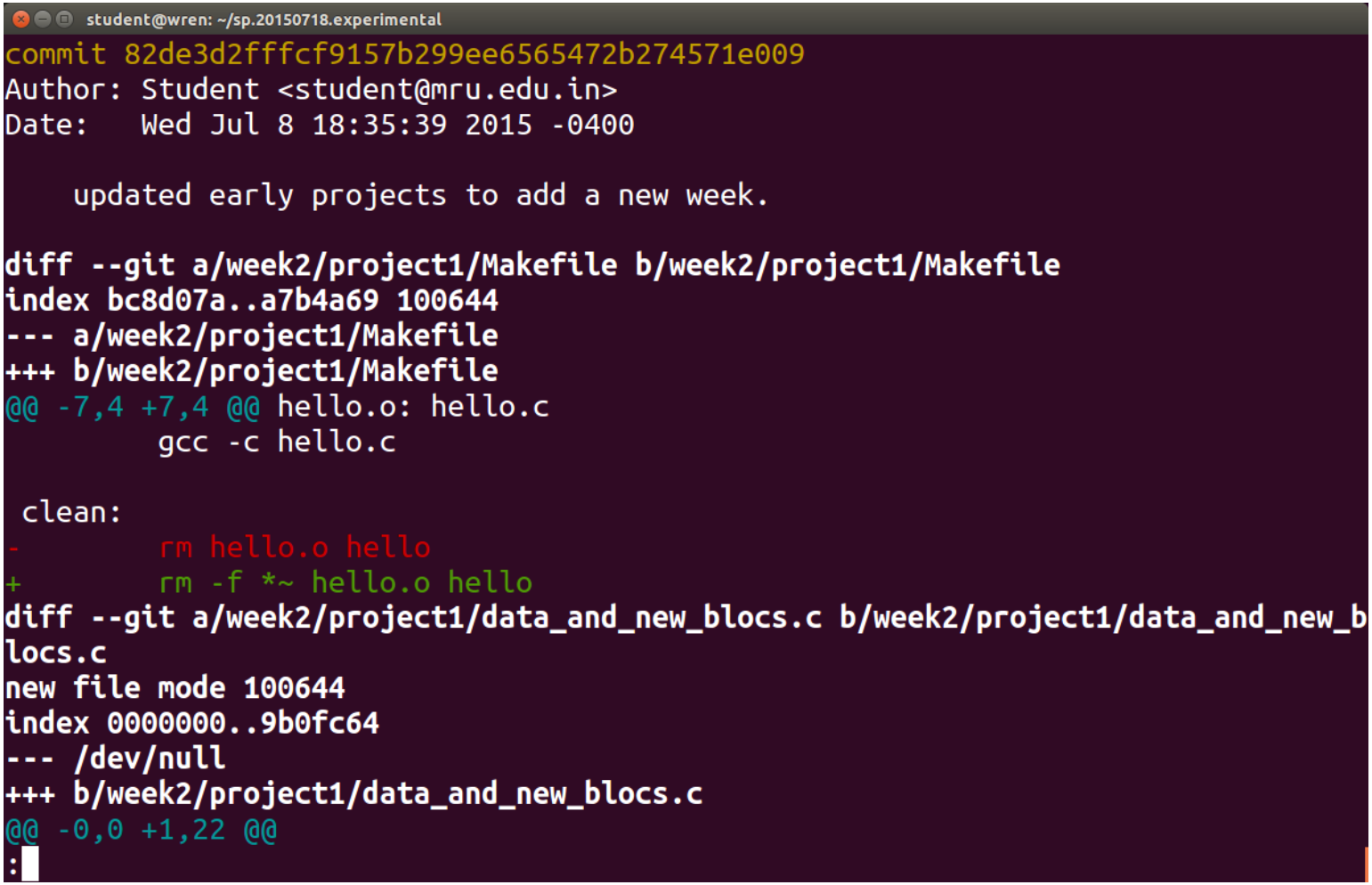
git log –oneline



List Revision Contents: git show

- Shows revision message then differences between it and previous revisiont
- Page through as with git log.

Example: git show 82de



Retrieving: git checkout

- To get work back use
  - git checkout <revision>
  - All of the files will be the like they were when you check in <revision>
  - Move the file you want to revert to a new location
- To return them to their current state:
  - git checkout master

More Information

- Best for starting out:
  - https://www.atlassian.com/git/tutorials/setting-up-a-repository
- The official manual:
  - http://git-scm.com/book/en/v2
- Reference
  - http://git-scm.com/docs