CprE 288 – Introduction to Embedded Systems (Programming Style, Doxygen, SVN)

Instructors:

Dr. Zhao Zhang (Sections A, B, C, D, E)

Dr. Phillip Jones (Sections F, G, J)

Overview of This Week

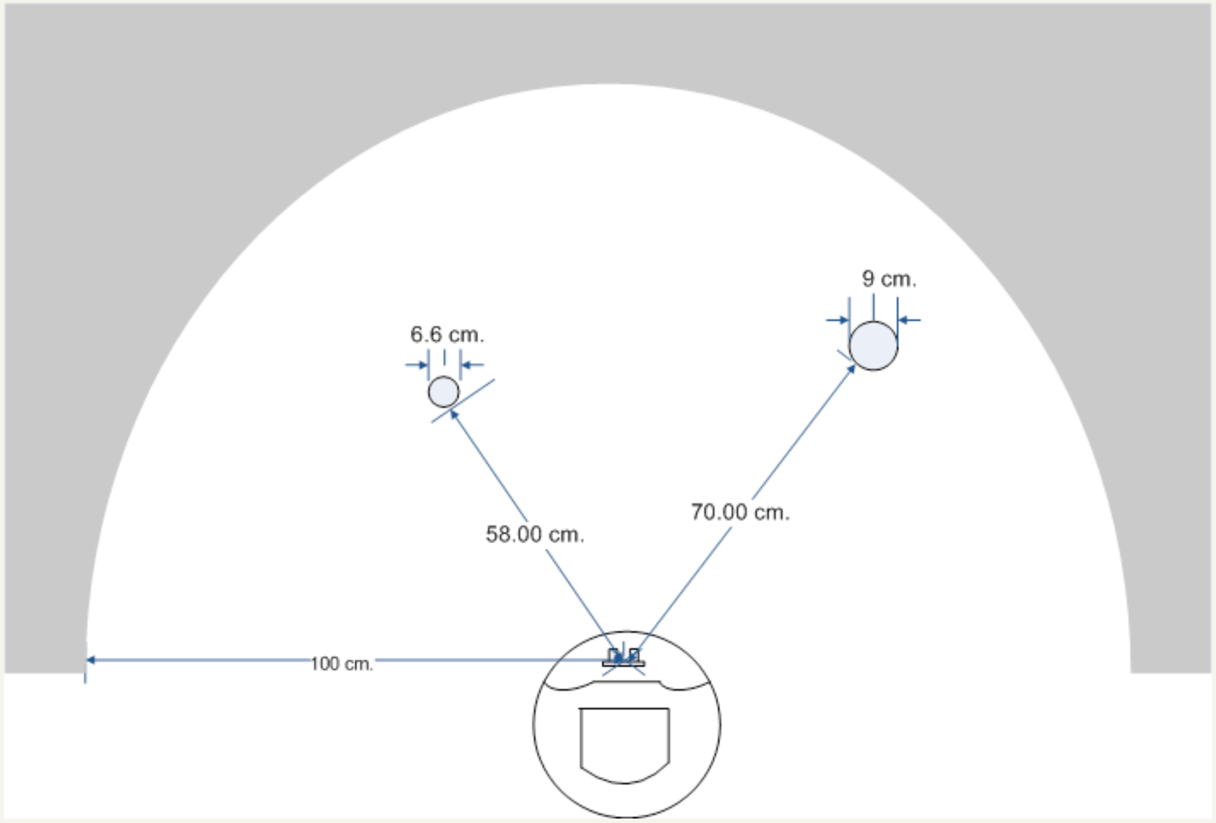
- Announcements
- Lab 9
- Homework Review
- Suggested Programming Style for Lab Project
- Doxygen: Automatic Documentation Generation
- SVN: Version Control System

Announcement

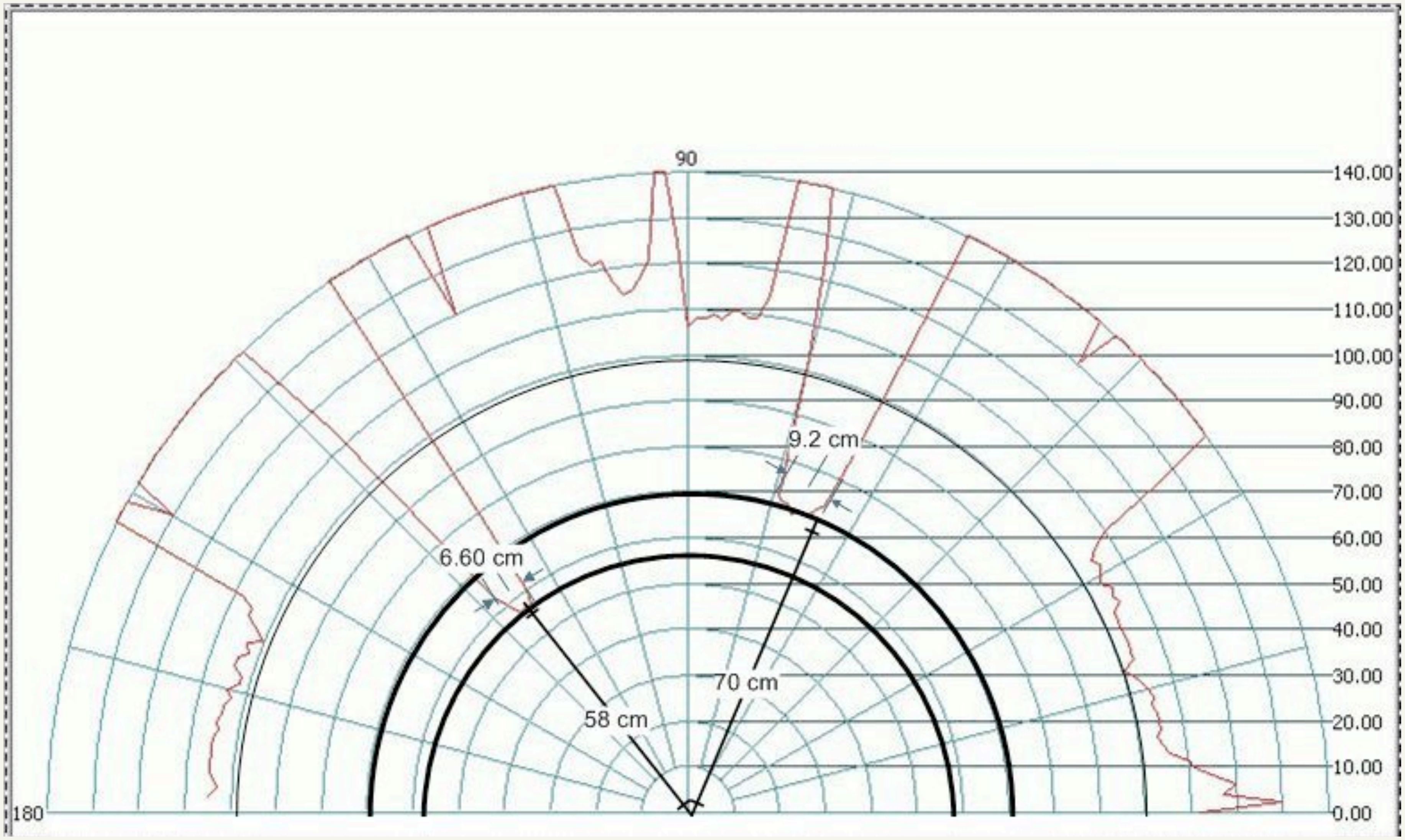
- Homework 8 due on Thursday in the class
- **Exam 2 next Thursday**
  - Open book, open notes
  - Coverage: USART, ADC, Input Capture, Output Compare

Lab 9: Object Counting and Size Discrimination

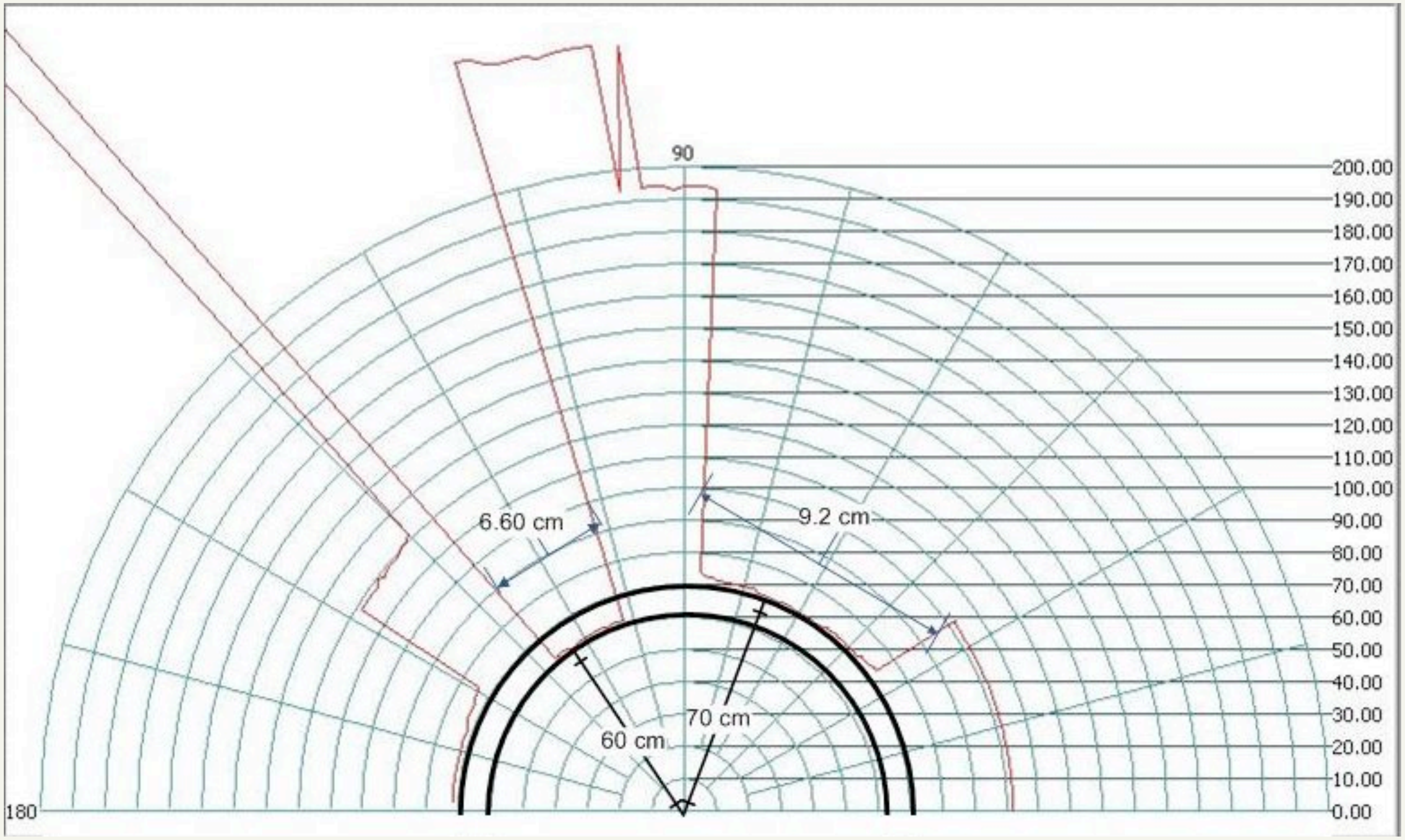
How do you distinguish two objects of different size?



Scanned Results by IR Censor



Scanned Result by Ping))) Sensor



Data Analysis

How can your program identify and distinguish different objects from the following raw data?

Degrees	IR Distance (cm)	Sonar Distance (cm)
0	120	324
2	123	330
4	119	363
6	40	40
8	40	40
10	40	41
… (more)		

Data Analysis

Step 1: Scan the array to identify gaps, convert them to **angular sizes**

- What's your algorithm?

Step 2: For each object, convert its **distance** and **angular size** into **linear size** (width)

- What's your mathematic formula?

Homework Review

- HW5, USART
- HW6, ADC
- HW7 and HW8 will be reviewed next week

Suggested Programming Style for Lab Project

References and Readings

- GNU Coding Standards. Free Software Foundation
- Proper Linux Kernel Coding Style. Greg Kroah-Hartman, Linux Journal, July 01, 2002
- Recommended C Style and Coding Standards. L. W. Cannon et al.
- Indent Style, Wikipedia

Credit: Jafar M. Al-Kofahi made contribution to an early version of 288 Lab Project Coding Style

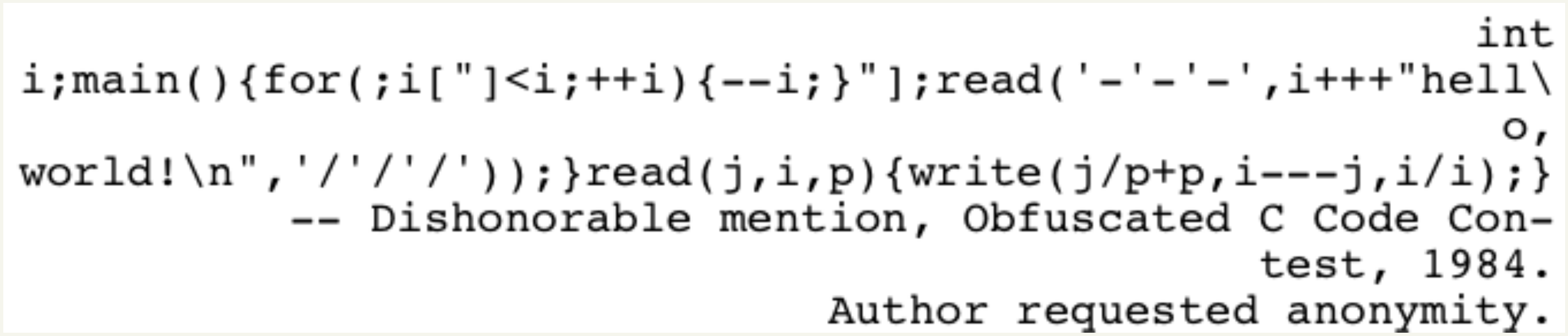
Suggested Programming Style for Lab Project

You are suggested to use the Programming style presented in this lecture

- It's a simplified version of GNU Coding Standards, with elements from the other references
- You may choose some variants, if with good reason

ALL partners of the same project team must use the same style with the same variants

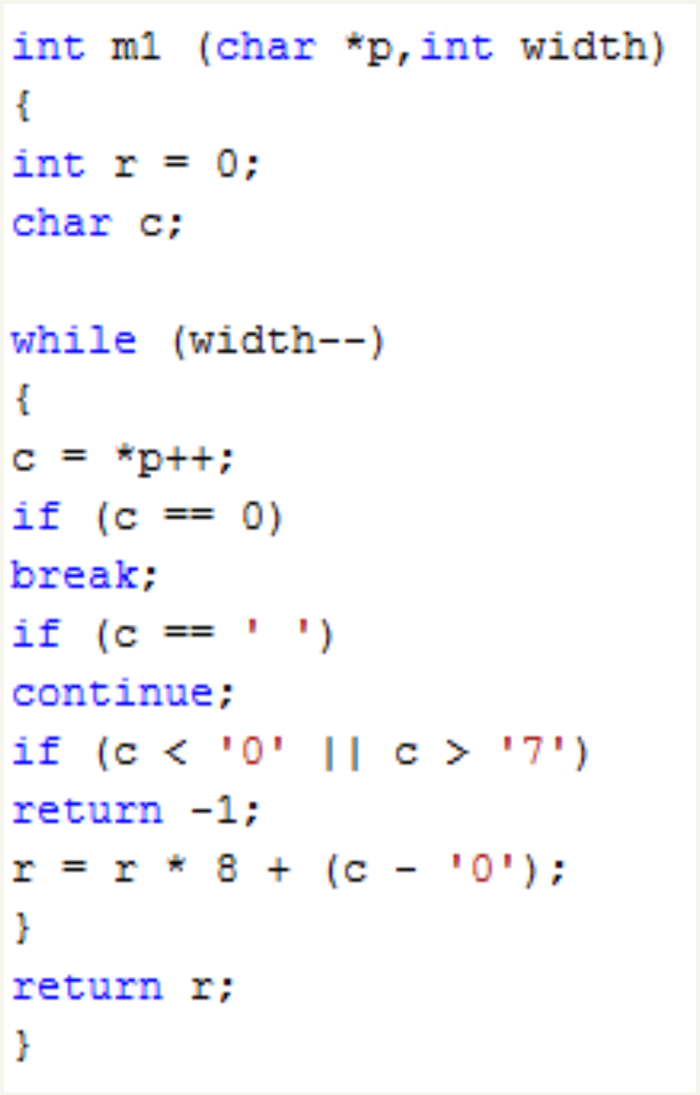
Why do we need it?



```
int
i;main(){for(;i["]<i;++i){--i;}"];read('-'-'-',i+++"hell\
o,
world!\n",'/'/'/'));}read(j,i,p){write(j/p+p,i---j,i/i);}
-- Dishonorable mention, Obfuscated C Code Con-
test, 1984.
Author requested anonymity.
```

From “Recommended C Style and Coding Standards”

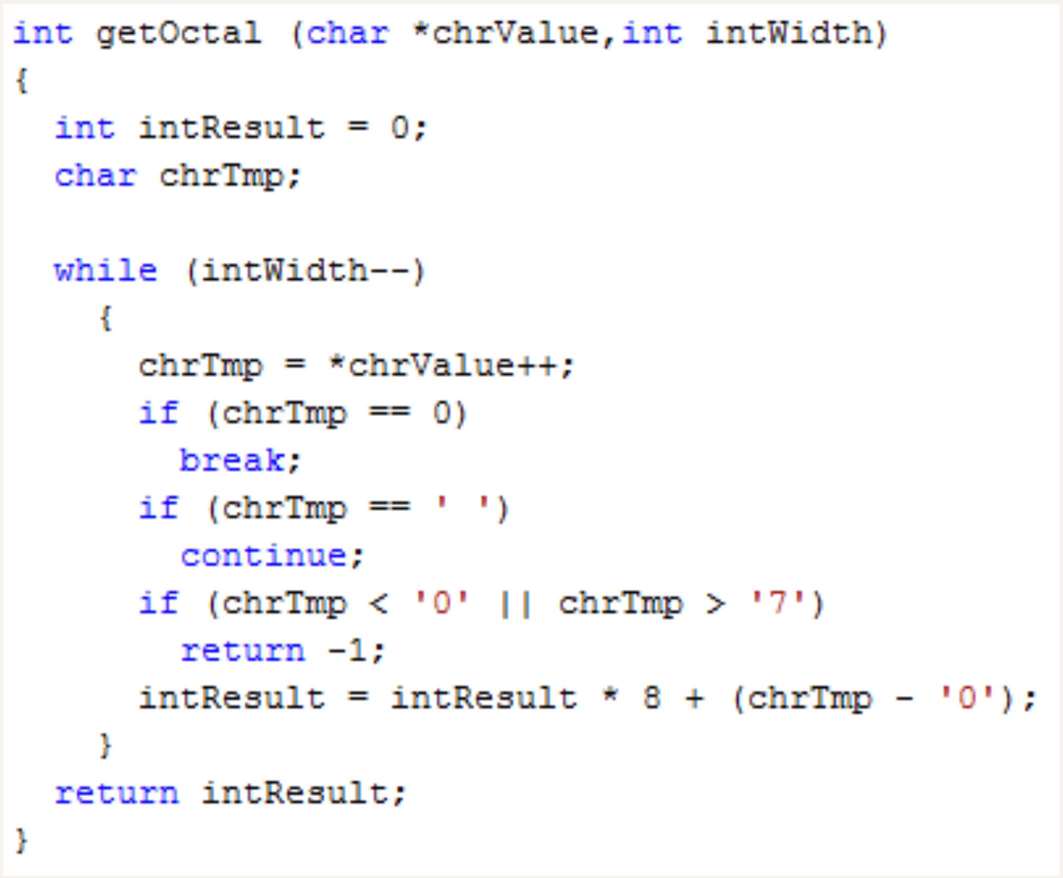
Why do we need it?



```
int m1 (char *p,int width)
{
int r = 0;
char c;

while (width--)
{
c = *p++;
if (c == 0)
break;
if (c == ' ')
continue;
if (c < '0' || c > '7')
return -1;
r = r * 8 + (c - '0');
}
return r;
}
```

Credit: Jafar M. Al-Kofahi

Why do we need it?



```
int getOctal (char *chrValue,int intWidth)
{
  int intResult = 0;
  char chrTmp;

  while (intWidth--)
    {
      chrTmp = *chrValue++;
      if (chrTmp == 0)
        break;
      if (chrTmp == ' ')
        continue;
      if (chrTmp < '0' || chrTmp > '7')
        return -1;
      intResult = intResult * 8 + (chrTmp - '0');
    }
  return intResult;
}
```

Credit: Jafar M. Al-Kofahi

Why do we need it?

We need a good coding style for many reasons

- Understand the code written by ourselves after some time
- Let others understand the code
- Reduce the number of bugs and the debugging time
- Overall, reduce the time spent on 288 Lab Project

C Programming Style

From GNU Coding Standards, Ch. 5, “Making the Best Use of C”

- Formatting: Format your source code
- Comments: Commenting your work
- Syntactic Convention: Clean use of C Constructs
- Names: Naming variables, functions, and files

Program File Layout

Suggested layout for .c files

1. A prologue that tells what is in the file
2. Any header file includes
3. Any defines and typedefs
4. Global data declarations
5. Functions, in some meaningful order

More details: Recommended C Style and Coding Standards, Sec 2.2 Program files

Program File Layout: Example

```
/*
 * ping.c: Ping))) sensor related functions
 */

#include <avr/io.h>
#include <avr/interrupt.h>
#include "servo.h"

// Number of clock cycles for 1-meter distance (single-trip) under prescalar 256
#define     TICKS_PER_METER                    735

volatile unsigned falling_time;                // captured time of falling edge
volatile unsigned rising_time;                 // captured time of rising edge

unsigned ping_read()
{
   ...
```

Prologue

Includes

Defines and defs

Global variables

Functions

Header File Layouts

Use the same layout for .c program files, for declarations visible to outside

Use C Macro def to avoid nested includes

```
#ifndef EXAMPLE_H
#define EXAMPLE_H
... /* body of example.h file */
#endif /* EXAMPLE_H */
```

Use extern for global variable visible to outside

```
extern int sound_speed;
```

Format Function

GNU Function layout

- Brace starts at column 1 of a new line
- Function name starts at column 1 of a new line

```
static char *
concat (char *s1, char *s2)
{
      …
}
```

Format Expression

Break an long expression: Split it **before** an operator and align the two parts properly

```
if (foo_this_is_long && bar > win (x, y, z)
    && remaining_condition)
```

Extra parenthesis: Add extra parentheses if they can make expressions clearer

```
max = (x > y) ? x : y;
```

Indent Style: GNU

```
int
sample_func()
{
  while (x == y)
    {
      something ();

      if (some_error)
        do_correct ();
      else
        cont_as_usual ();
    }

  finalthing ();
}
```

GNU indent style

- The opening brace occupies a line
- The opening brace is indented by 2 spaces
- The next statement is indented by another 2 spaces

Indent Style: K&R

```
int sample_func()
{
    while (x == y) {
        something();

        if (some_error)
            do_correct();
        else
            cont_as_usual();
    }

    finalthing();
}
```

K&R indent style

- The opening brace of a control body does NOT take a line
- The next statement is indented by 4 spaces

The K&R Book: The C Programming Language, Brian W. Kernighan and Dennis M. Ritchie

Indent Style: Allman

```
int sample_func()
{
    while (x == y)
    {
        something();

        if (some_error)
            do_correct();
        else
            cont_as_usual();
    }

    finalthing();
}
```

Allman indent style (ANSI style)

- The opening brace of a control body takes a line
- The opening brace is indented by 0 space
- The next statement is indented by 4 spaces

Indent Style: Simple Control Statements

GNU:

```
if (x == y)
  do_something ();
else
  do_others ();
```

K&R and Allman:

```
if (x == y)
    do_something();
else
    do_others();
```

If the control body is a single statement:

- GNU: Indented by 2 spaces
- K&R and Allman: Indented by 4 spaces

GNU function call: Note the extra space between the function name and "("

Indent Style: Lab Project

Which style to use? Your choice!

- Each style has its own rational and history

For the Lab Project

- GNU is more generous in using line space, more popular today because of GNU projects
- Allman is the most compatible, among the three, with the AVR's studio's default indentation
- K&R is the most compact, and more AVR-compatible than GNU

Everyone in the same team must use the same style!

Format Switch Statement

```
switch (expr)
  {
  case ABC:
  case DEF:
    statement;
    break;
  case UVW:
    statement;
  case XYZ:
    statement;
    break;
  }
```

GNU Style:

- Cases are aligned with the opening brace (indented by 2 spaces)
- The statements are indented by 2 spaces from case, 4 spaces from switch

Format Switch Statement

```
switch (expr) {
case ABC:
case DEF:
    statement;
    break;
case UVW:
    statement;
case XYZ:
    statement;
    break;
}
```

K&R Style

- Cases are aligned with the switch
- Statements are indented by 4 spaces from case and switch

Format Switch Statement

```
switch (expr)
{
case ABC:
case DEF:
    statement;
    break;
case UVW:
    statement;
case XYZ:
    statement;
    break;
}
```

Allman Style*

- Cases are aligned with the switch and the open brace
- Statements are indented by 4 spaces from case and switch

* This may not be the original Allman style

Format Statement

Automatic indent tool: indent

- Available on Linux, Mac or other UNIX-type systems

Format with the GNU style

```
indent -gnu sample.c
```

Format with the K&R style

```
indent -kr sample.c
```

Format with the original Berkeley style (also popular)

```
indent -orig sample.c
```

Commenting Your Work

GNU guidelines and our suggestion:

- Each program should start with a comment saying briefly what it is for
- Each function should have a starting comment saying what the function does
- Explain arguments properly, particularly if there is anything unusual
  - E.g. A string that is not necessarily zero-terminated
- Explain the return value
- Be generous in commenting, try to put a comment for every block of statements or statement with non-straightforward meaning

Commenting Your Work

More from "Recommended C Style and Coding Standards"

- Write a block of comment prologue to each function
- Make function return value have its own line, with probably a comment explain the return value (same as GNU)
- Try to align comments
- Use a blank line between local variable declarations and the function's statements

Commenting: Example

```
/* Move serve to a angular position given by degree. */
void
move_servo(unsigned degree)
{
    unsigned pulse_width;                    // pulse width in Timer/Counter cycles

    // Pulse width is (1+(degree/180))*t cycles, t is number of clock cycles per millisecond
    pulse_width = 1*MS_TICKS + (degree*MS_TICKS/180);

    OCR3B = pulse_width-1;                   // set pulse width
     wait_ms(500);                           // wait for half second for servo to settle
}
```

Commenting: Example

```
/* Start Ping))) sensor, read the pulse width, and return distance in millimeter */
unsigned                        // return distance, 0 if out of range (> 1000mm)
ping_read()
{
    send_pulse();               // send the starting pulse to PING
    state = LOW;                // now in the LOW state

    // Enable Timer1 and interrupt, with noise cancellation (ICNC=1),
    // detecting rising edge (ICES=1), and prescalar 1024 (CS=101)
    TCCR1B = _BV(ICNC) | _BV(ICES) | _BV(CS2) | _BV(CS0);
```

Commenting: Example

```
  // Wait until IC is done
  while (state != DONE)
       {}

  // Disable  Timer/Counter 1: CS=000
  TCCR1B &= ~(_BV(CS2) | _BV(CS1)| _BV(CS0));

 // Convert time difference in cycles to distance in millimeter
 unsigned dist = (falling_time - rising_time) / (2 * cycles_per_mm);

 // Out of range?
 if (dist > 1000)
    dist = 0;

 return dist;
}
```

Commenting: Doxygen

Note: You are required to use Doxygen, which has all those requirements included, plus special formats for commenting

Doxygen is a documentation generator

- It generates software reference documents automatically from your comments

Nested Control Statement

Always use braces to separate nested control statements

```
if (foo)
  {
    if (bar)
      win ();
    else
      lose ();
  }
```

The following style is bad

```
if (foo)
  if (bar)
    win ();
  else
    lose ();
```

Naming Conventions

GNU coding standards:

Use underscore to separate multiple words

```
    falling_time
    rising_tme
    init_servo
    move_servo
```

Try to use short local variable names

Naming Conventions

More from “Recommended C Style and Coding Standards”

- Avoid local declarations that override declarations at higher level, e.g. local vs. global, same local names in nested blocks
- Avoid using names started with underscore (to avoid conflicts with system/library variables)
- #define constants should be in all CAPS
- Function, typedef, and variable names, as well as struct, union, and enum tag names should be in lower case
- Avoid names close to each other, e.g. foo and Foo, foobar and foo_bar, bl and b1 and bI (with upper case I)

White Space

Use white spaces generously

```
 if ((a + b) == (c - d))
```

Split long for-loop and align the lines

```
for (curr = *listp, trail = listp;
     curr != NULL;
     trail = &(curr->next), curr = curr->next)
{
     ...
```

Program File Organization

Use multiple program files, one .c file and one .h file for each program module

Examples:

- lcd.c, lcd.h
- util.c, util.c
- ir_sensor.c, ir_sensor.h
- ping.c, ping.h
- robot.c, robot.h
- servo.c, servo.h
- main.c

Doxygen Introduction

Dr. Zhao Zhang and Dr. Phillip Jones

Based on Doxygen Tutorial by Jafar Al-Kofahi

Why Use Doxygen

To generate software reference documentation **automatically**

- So you don't have to write a separate document for your project!

It works with C, C++, Java, Fortran, C# and many other languages

Documentation is extracted from special comments in the source code

File Headers

Add a file header as follows

```
/**
*       @file util.h
*       @brief this header file will contain all required
*       definitions and basic utilities functions.
*
*       @author:Dr. Zhao Zhang
*
*       @date 2/28/2009
*/
```

File Headers

Doxygen commands:

@file: declares the file name

@brief: gives a brief description of the file

@author: gives the author name

@date: specifies the date of creation

Function Example

```
/**
*    Print a single character to the LCD.
*    @author        Dr.Zhao Zhang
*    @param ch      The character to print
*    @date          2/28/2009
*/
void
lcd_putc(char ch)
{
     /// Sends out the higher half to LCD
     PORTA |= ch >> 4;
     lcd_toggle_clear(1);

     /// Sends the lower half to LCD
     PORTA |= ch & 0x0F;
     lcd_toggle_clear(1);
}
```

Function Example

```
/**
*    This method will be used to print a string to the lcd.
*    @author          Jafar Al-Kofahi
*    @param str       The string to print on the LCD
*    @date            2/28/2009
*/
void
lcd_puts(char *str)
{
   /// Write characters in str to LCD one by one until NULL is met
   while (*str)
      lcd_putc(*str++);
}
```

Comment Functions

Add a comment prologue to each function

@author: Gives the author name

@date: Specifies the date of creation

@param: Describe a parameter

@return: Describe a return value

Tutorial and Example

A tutorial:

http://class.ee.iastate.edu/cpre288/lecture/Doxygen/DoxygenTutorial.pdf

Example Doxygen output:

http://class.ee.iastate.edu/cpre288/lecture/Doxygen/html/files.html

SVN Introduction and Operations

Dr. Zhao Zhang and Dr. Phillip Jones

SVN Introduction: Outline

- Introduction
- Provided functionality
- SVN Work-flow
- Common problem
- Subversion (SVN)
- SVN Limitation
- Current available systems
- Tutorial

Credit: Jafar M. Al-Kofahi prepared an early version of this lecture

Introduction

- What is it?

- SVN is a version control system that can be used for concurrent development of program files, and it manages different versions of those files in a repository.

- It is commonly used by software companies to keep track of different versions of their projects.

Introduction

What is it being used for?

- Maintaining file history.
- Managing collaboration on files.
- Managing releases.

Common Functionality

- Check-out
- Check-in (commit)
- Update
- Add/Remove
- History
- Compare

SVN – Workflow

- Get a working copy of the project/file
- Make changes in your copy
- **<u>Test</u>** them locally
- Integrate them with any changes made to the Repository
- Commit them back to the Repository
- Repeat these steps until a release is ready
- Start making changes again for next release

SVN – Commit

When you commit your changes you should do the following:

Try always to group your committed changes, for example if you have 5 new files, and 3 of them were added to support functionality A, then those 3 should be committed together and you MUST write one log for them.

For the remaining two if they were both standalone (their changes does not affect any other files) each one of them MUST get a separate log.

SVN – Commit

What to write in the log?

- Any major change you did
- Why you did it
- Any reported bugs and fixes
- Any new methods, files, sub-classes..etc
- …etc

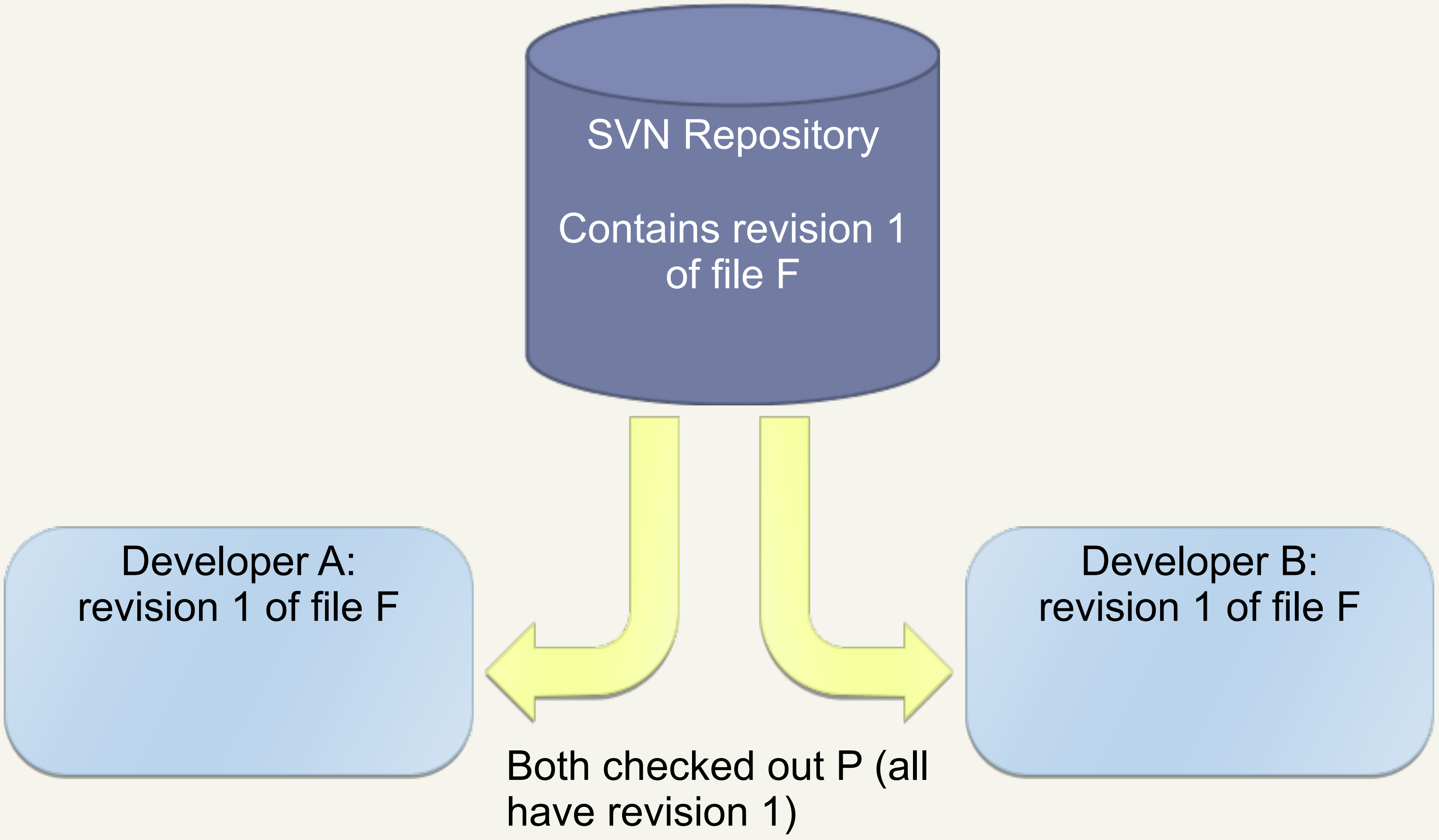
Common problem

There are couple of problem scenarios, and in the next couple of slides we will talk about one of them.

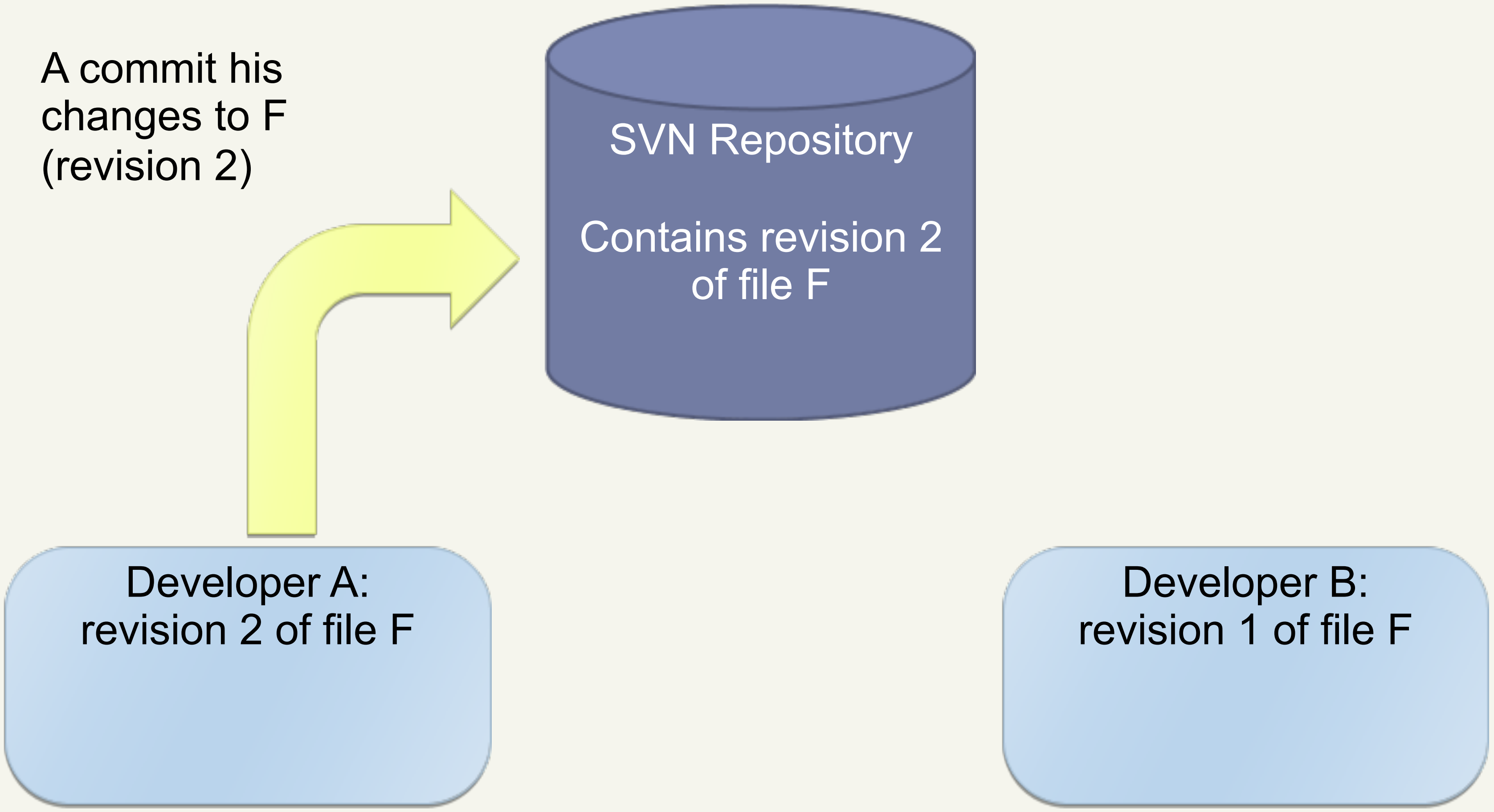
Consider the following scenario:

- We have one project called P that contain file F.
- Two new developers A and B joined the team.
- Both were assigned the task of changing file F.

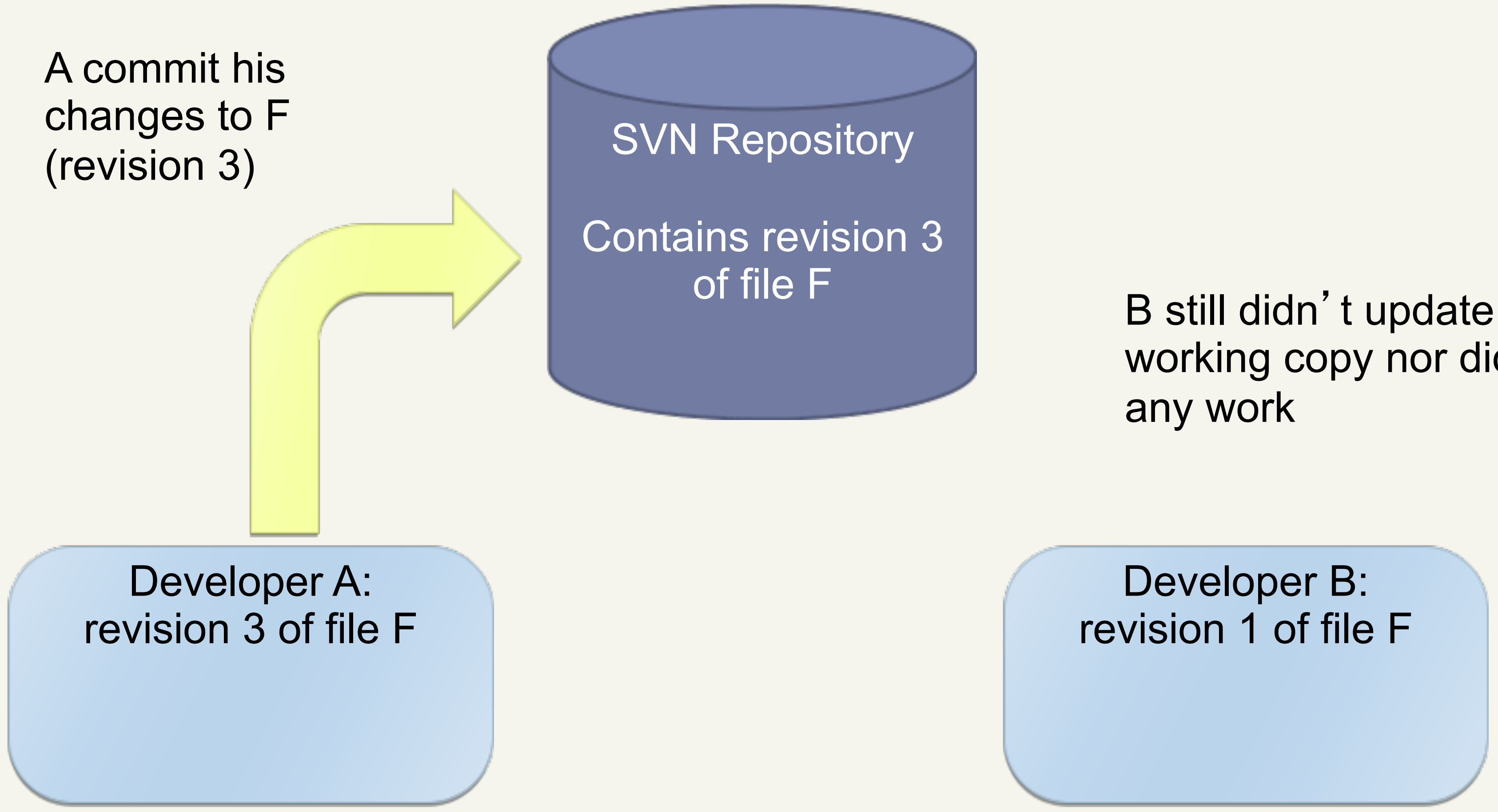
Problem - 2


SVN Repository
Contains revision 1 of file F
Developer A: revision 1 of file F
Developer B: revision 1 of file F
Both checked out P (all have revision 1)

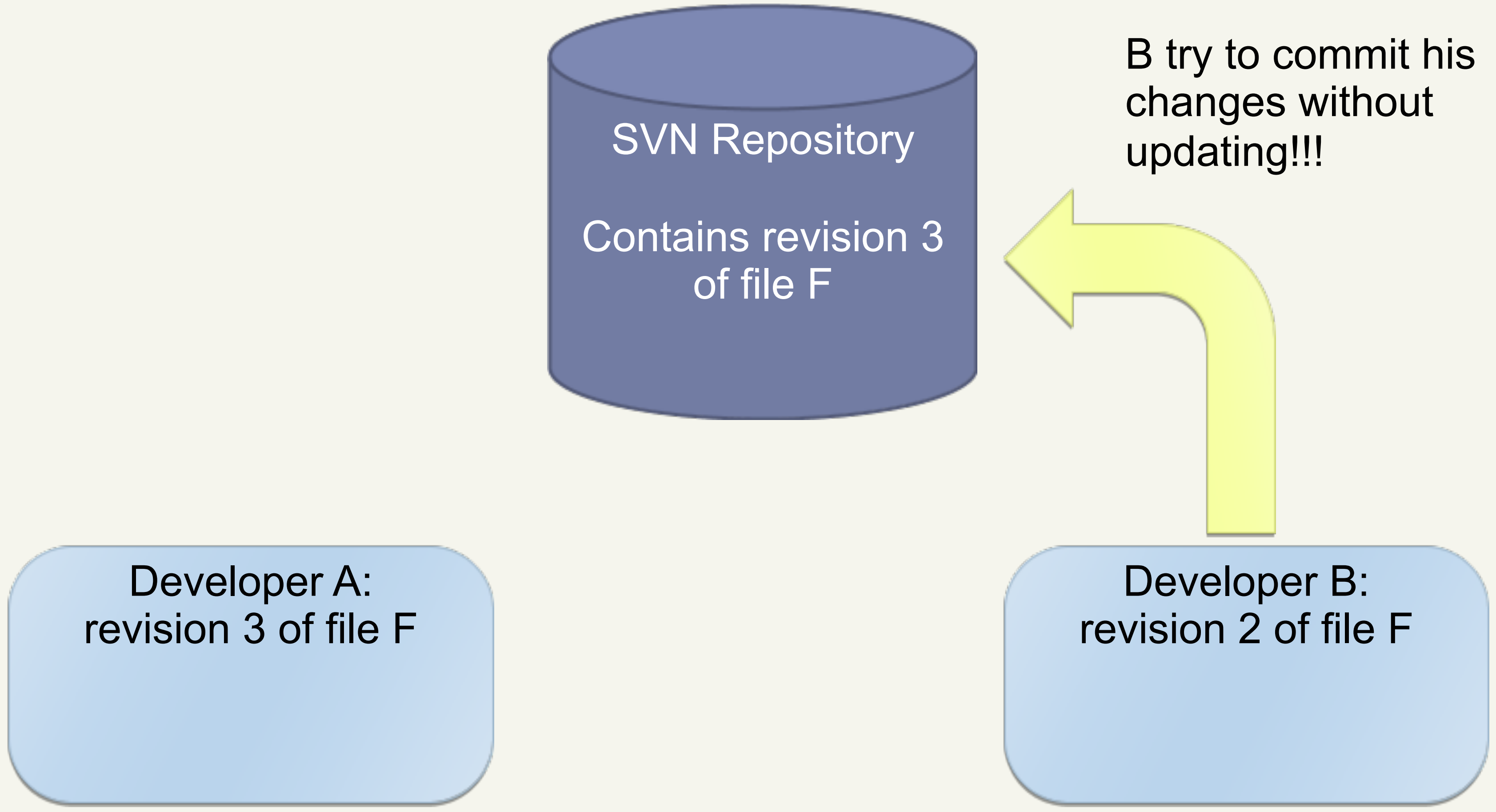
Problem - 3


A commit his changes to F (revision 2)
SVN Repository
Contains revision 2 of file F
Developer A: revision 2 of file F
Developer B: revision 1 of file F

Problem - 4


A commit his changes to F (revision 3)
SVN Repository
Contains revision 3 of file F
B still didn’t update his working copy nor did any work
Developer A: revision 3 of file F
Developer B: revision 1 of file F

Problem - 5


SVN Repository
Contains revision 3 of file F
B try to commit his changes without updating!!!
Developer A: revision 3 of file F
Developer B: revision 2 of file F

Problem - 6

Because of B commit there will be a **conflict** problem, and there are two cases:

B’s work is totally independent of A’s work (separate), which is easily solved by merging them.

B and A both modified the same code portion! What do you think should happen?

A preventive solution is Exclusive locks.

Concurrent Versioning Systems (CVS)

- CVS is very similar to SVN, and usually the differences between them is in term of architecture (different kind of repository, information stored …etc).

- SVN systems started around 2000, while CVS systems existed around 1985.

- Some argue that SVN is the new successor of CVS, that solve CVS problems, while others consider it as a separate system.

Current available systems

Some websites provides a CVS/SVN server:

- www.sourceforge.net

- There is also one for ECE department that you will be using for managing your code
- www.source.ece.iastate.edu

Some come as clients that can be used locally or can be connected to a server:

- TorToiseSVN
- Subclipse (a plug-in for Eclipse)

References

1. Ben Collins-Sussman, Brian W. Fitzpatrick, and C. Michael Pilato, Version Control with Subversion: For Subversion 1.5: (Compiled from r3305), 2008
2. Student Technology Services, Johns Hopkins University. *http://sts.jhu.edu/loader.php?page=tutorial-subversion.html*
3. Jennifer Vesperman, http://www.linuxdevcenter.com/pub/a/linux/2002/01/03/CVS_intro.html
4. www.Wikipedia.org
5. http://www.pushok.com/soft_svn_vscvs.php

SVN Operations: MyGForge and Clients Tutorial

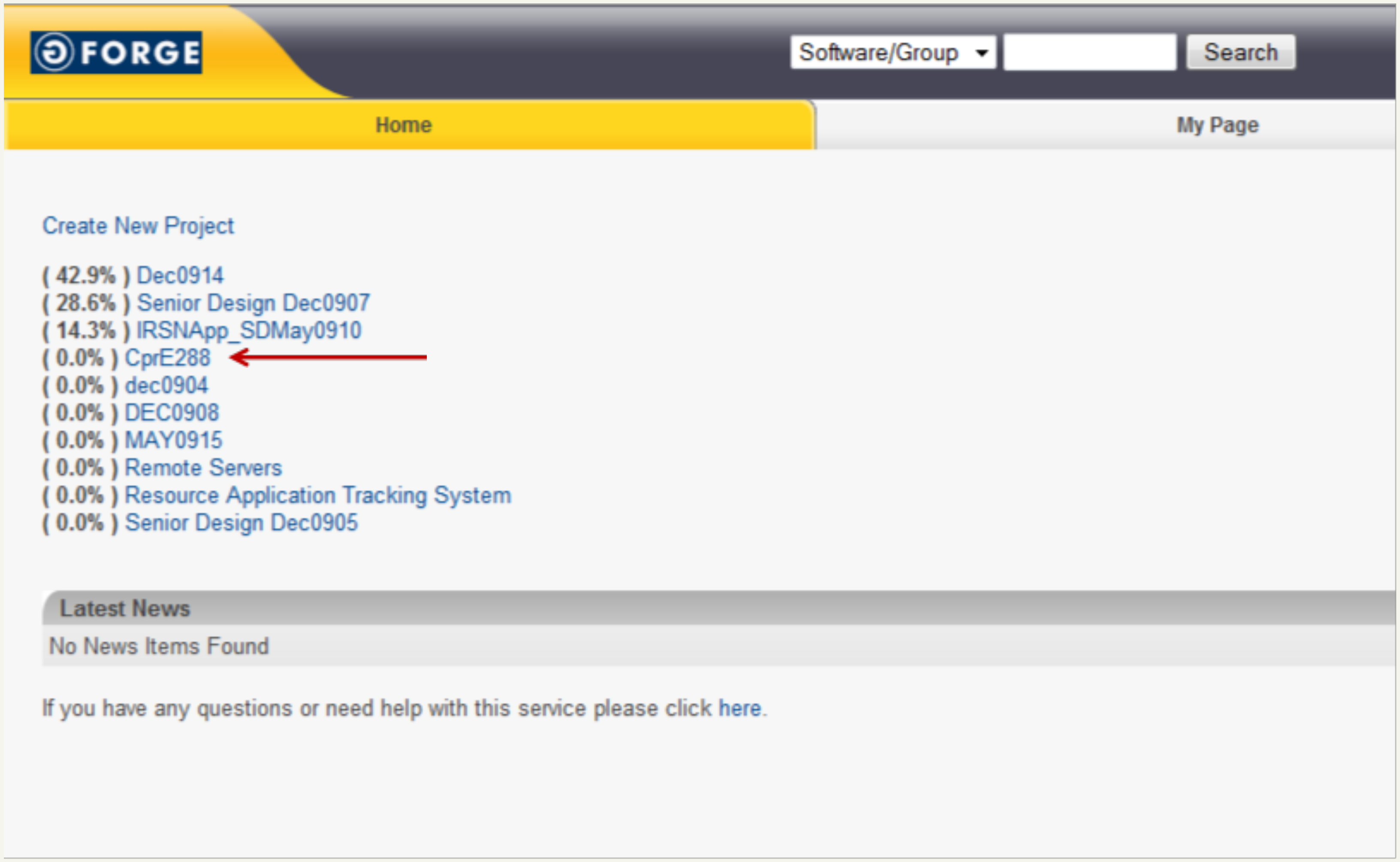
Create an account

- Before we start, you need to go to https://source.ece.iastate.edu/

- Then do a log-in using your Net-ID

- Then, under the Home tab select your project (for example "CprE288") and submit a "Request to join", type in your name and Net-ID and press the submit button.

Create an account



Create New Project

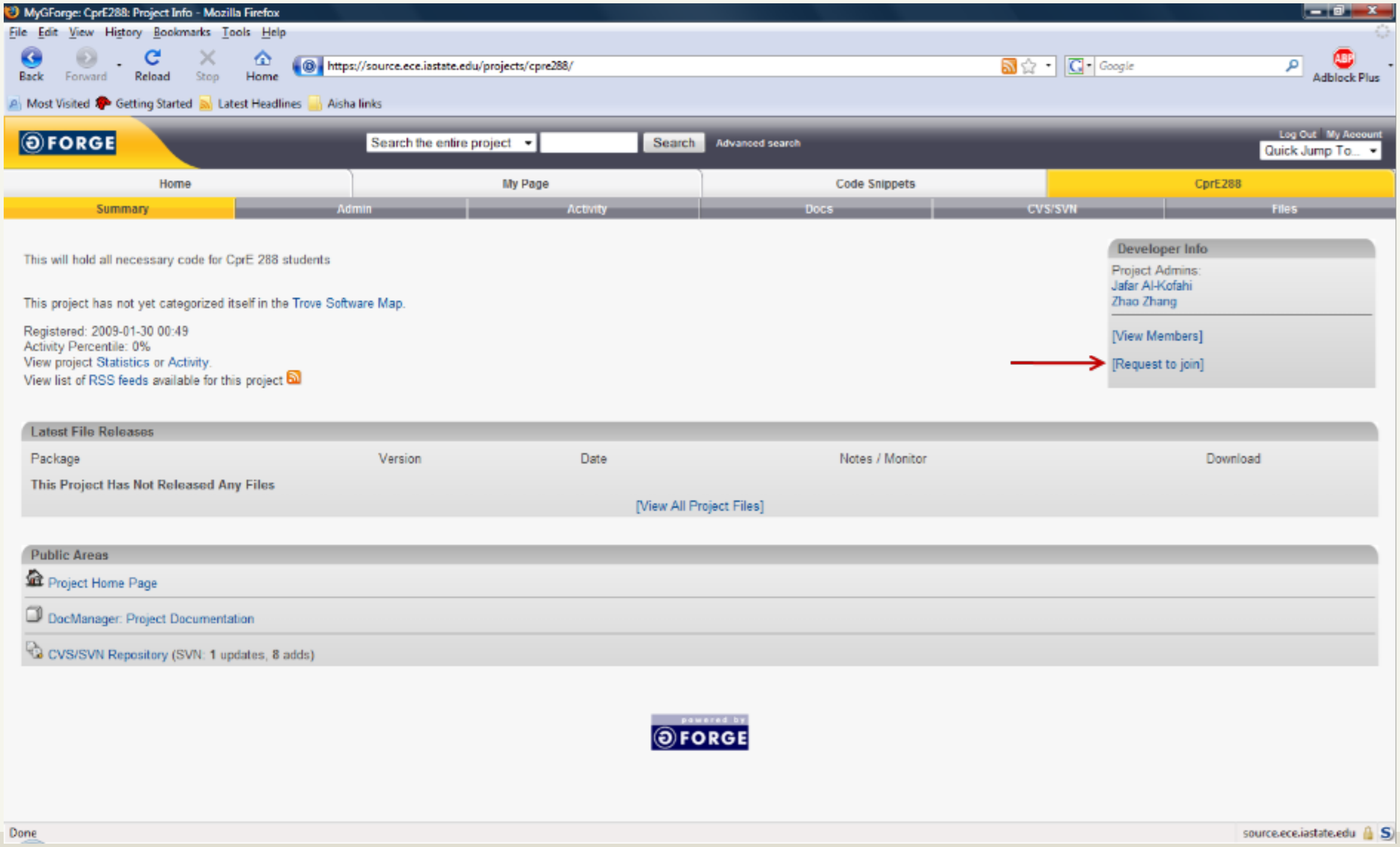
(42.9%) Dec0914
(28.6%) Senior Design Dec0907
(14.3%) IRSNApp_SDMay0910
(0.0%) CprE288
(0.0%) dec0904
(0.0%) DEC0908
(0.0%) MAY0915
(0.0%) Remote Servers
(0.0%) Resource Application Tracking System
(0.0%) Senior Design Dec0905

Latest News

No News Items Found

If you have any questions or need help with this service please click here.

Create an account


MyGForge: CprE288: Project Info - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Back Forward Reload Stop Home
https://source.ece.iastate.edu/projects/cpre288/
Google
Adblock Plus
Most Visited Getting Started Latest Headlines Aisha links
FORGE
Search the entire project
Search
Advanced search
Log Out My Account
Quick Jump To...
Home
My Page
Code Snippets
CprE288
Summary
Admin
Activity
Docs
CVS/SVN
Files
Done
source.ece.iastate.edu

This will hold all necessary code for CprE 288 students

This project has not yet categorized itself in the Trove Software Map.

Registered: 2009-01-30 00:49
Activity Percentile: 0%
View project Statistics or Activity.
View list of RSS feeds available for this project

Developer Info
Project Admins:
Jafar Al-Kofahi
Zhao Zhang

[View Members]

[Request to join]

Latest File Releases

Package	Version	Date	Notes / Monitor	Download
This Project Has Not Released Any Files				

[View All Project Files]

Public Areas

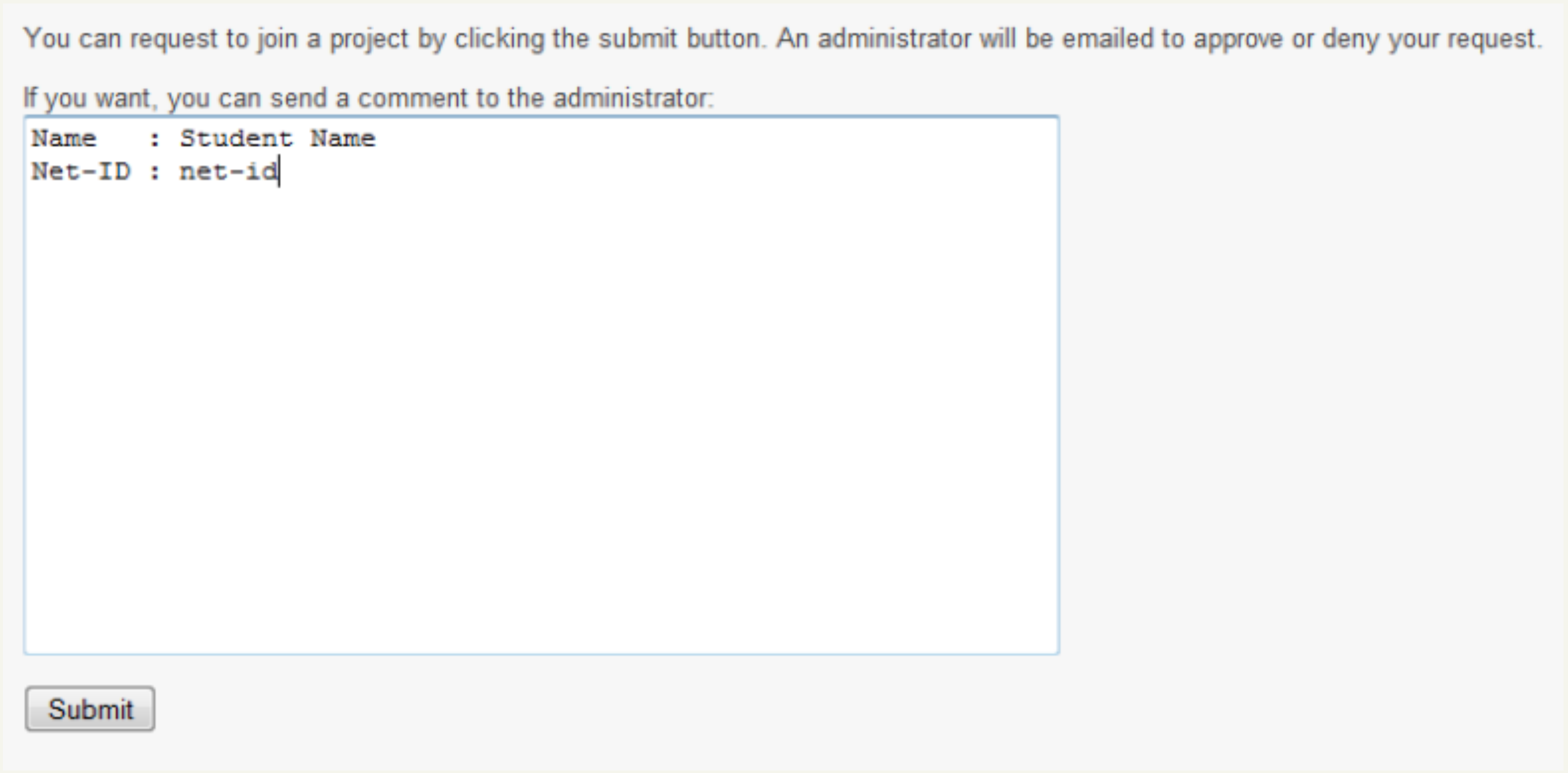
Project Home Page

DocManager: Project Documentation

CVS/SVN Repository (SVN: 1 updates, 8 adds)

powered by FORGE

Create an account



You can request to join a project by clicking the submit button. An administrator will be emailed to approve or deny your request.

If you want, you can send a comment to the administrator:

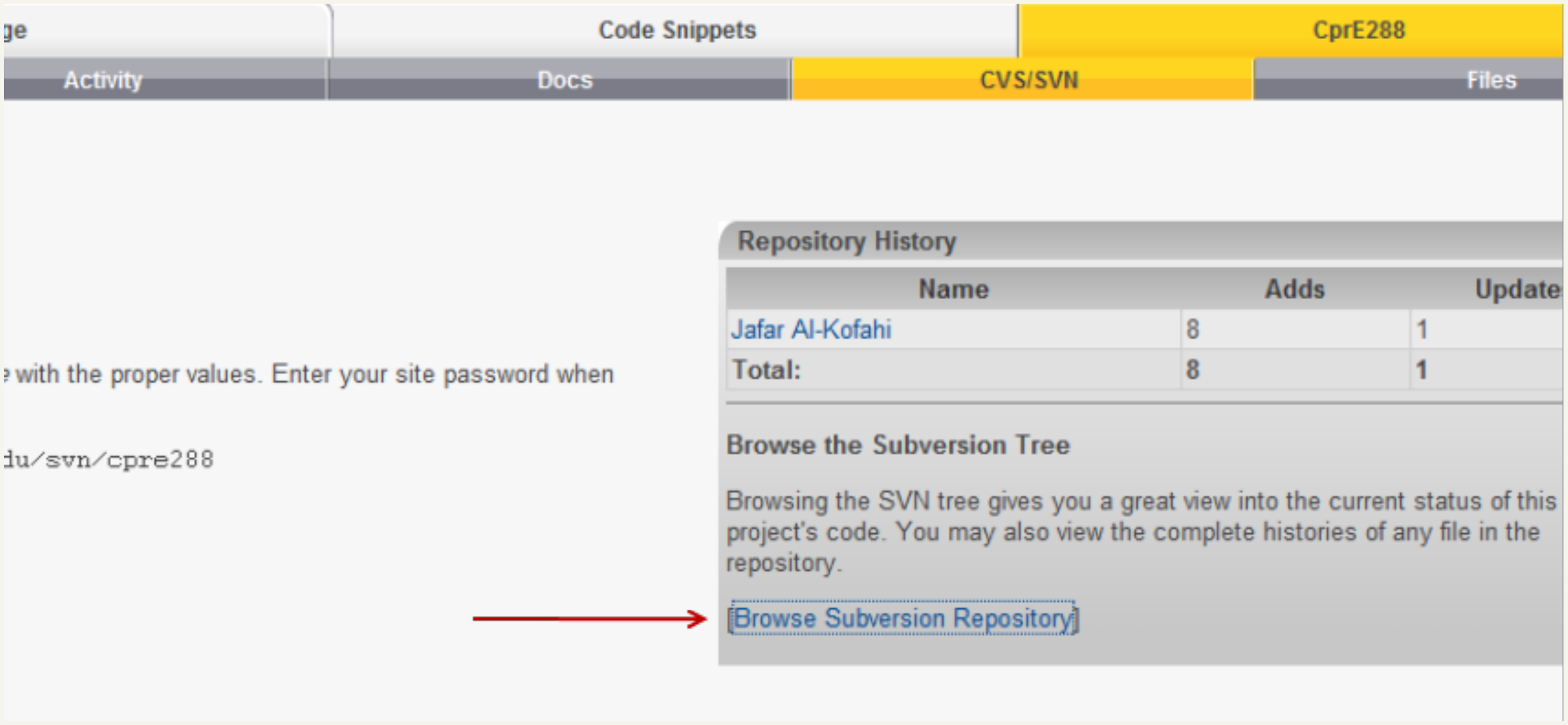
```
Name    : Student Name
Net-ID : net-id
```

Using the website

- The website will provide you with the SVN basic functionalities, such as:
    - Browse SVN files: see the contents in the SVN for any revision
    - Compare revisions: see differences between revisions
    - Recent activity: get informed by any recent changes to the project contents, reported bugs…etc

Using the website – browse SVN

- Go to the CVS/SVN tab, then click “Browse Subversion Repository”



e with the proper values. Enter your site password when

du/svn/cpre288

Repository History

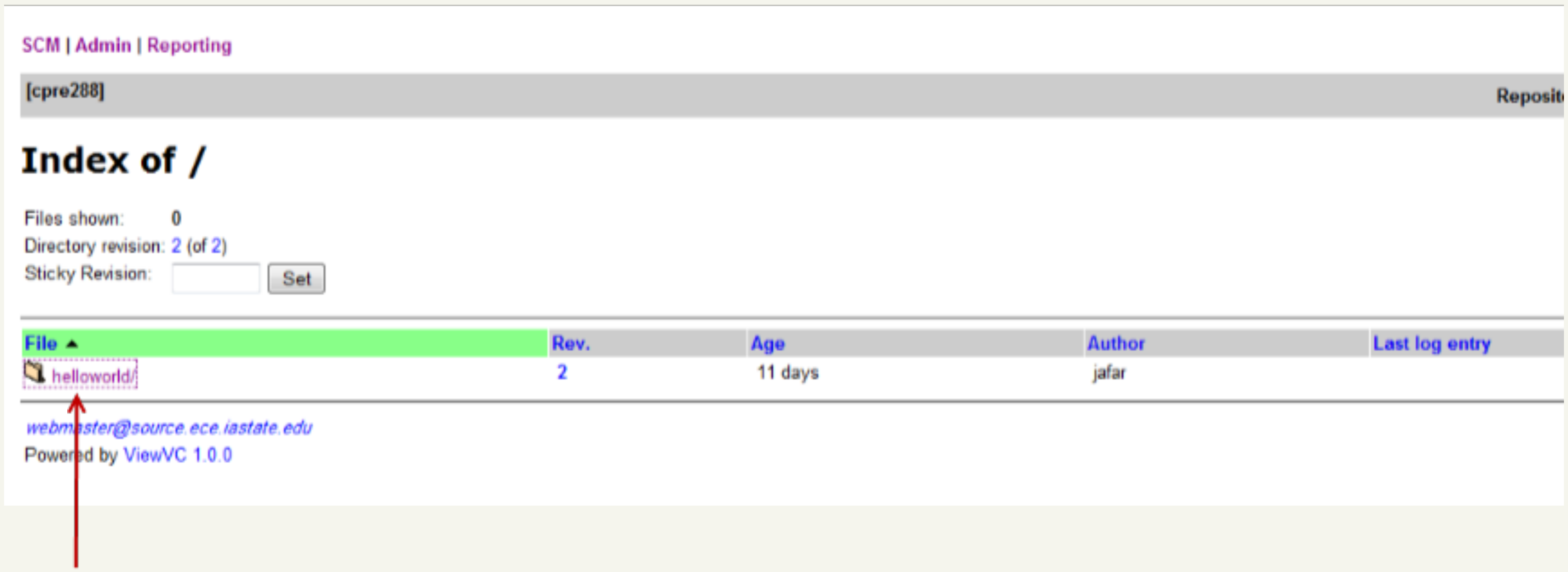
Name	Adds	Update
Jafar Al-Kofahi	8	1
Total:	8	1

Browse the Subversion Tree

Browsing the SVN tree gives you a great view into the current status of this project's code. You may also view the complete histories of any file in the repository.

[Browse Subversion Repository]

Using the website – browse SVN



SCM | Admin | Reporting

[cpre288]

Reposit

Index of /

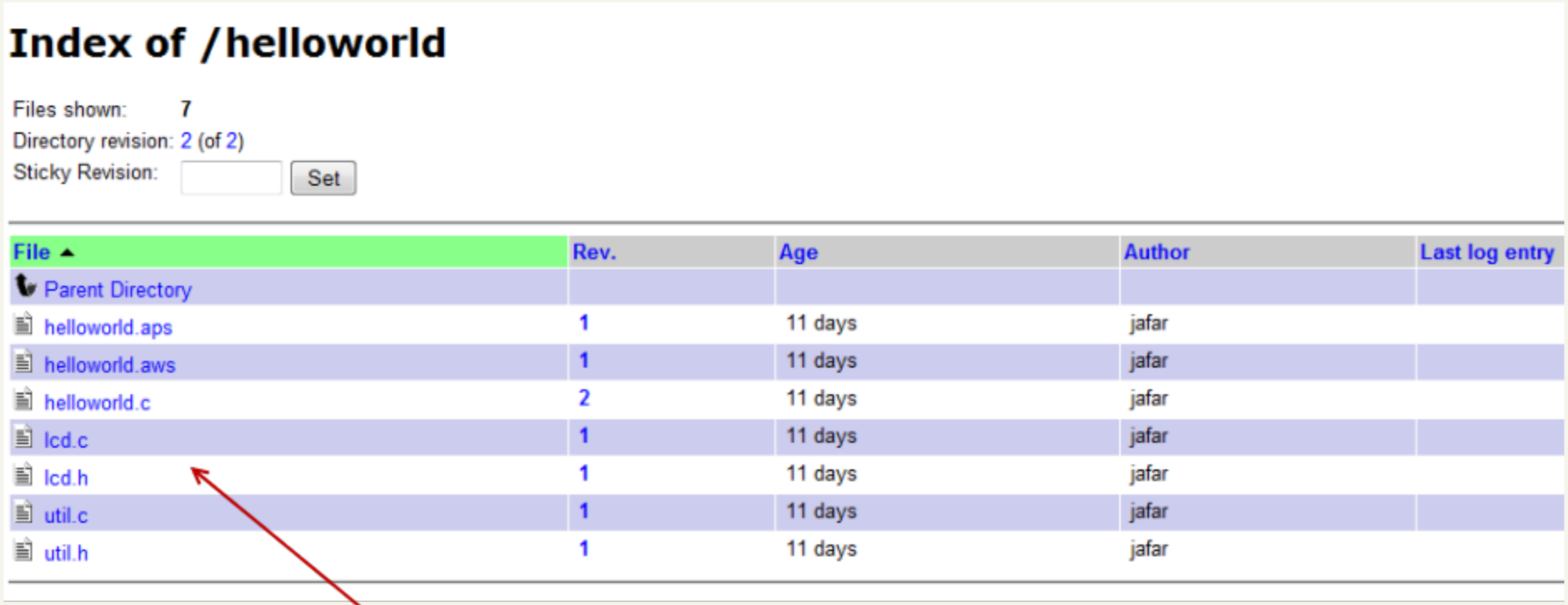
Files shown:	0
Directory revision:	2 (of 2)
Sticky Revision:	

File ▲	Rev.	Age	Author	Last log entry
helloworld/	2	11 days	jafar	

webmaster@source.ece.iastate.edu

Powered by ViewVC 1.0.0

Using the website – browse SVN



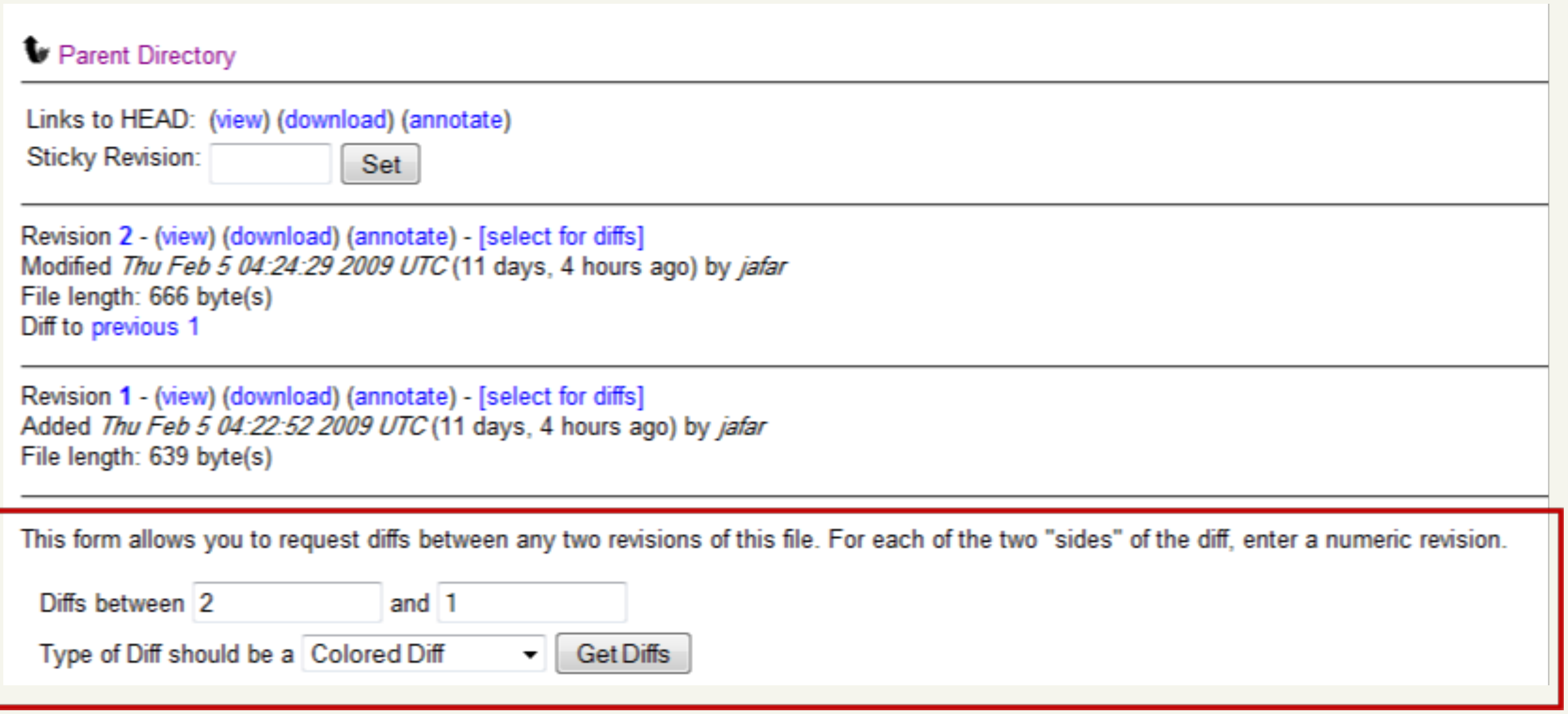
Index of /helloworld

Files shown:	7
Directory revision:	2 (of 2)
Sticky Revision:	Set

File ▲	Rev.	Age	Author	Last log entry
Parent Directory				
helloworld.aps	1	11 days	jafar	
helloworld.aws	1	11 days	jafar	
helloworld.c	2	11 days	jafar	
lcd.c	1	11 days	jafar	
lcd.h	1	11 days	jafar	
util.c	1	11 days	jafar	
util.h	1	11 days	jafar	

Click on any of the files for more info and comparing revision

Using the website – browse SVN



Parent Directory

Links to HEAD: (view) (download) (annotate)
Sticky Revision: [ ] Set

Revision 2 - (view) (download) (annotate) - [select for diffs]
Modified *Thu Feb 5 04:24:29 2009 UTC* (11 days, 4 hours ago) by *jafar*
File length: 666 byte(s)
Diff to previous 1

Revision 1 - (view) (download) (annotate) - [select for diffs]
Added *Thu Feb 5 04:22:52 2009 UTC* (11 days, 4 hours ago) by *jafar*
File length: 639 byte(s)

This form allows you to request diffs between any two revisions of this file. For each of the two "sides" of the diff, enter a numeric revision.

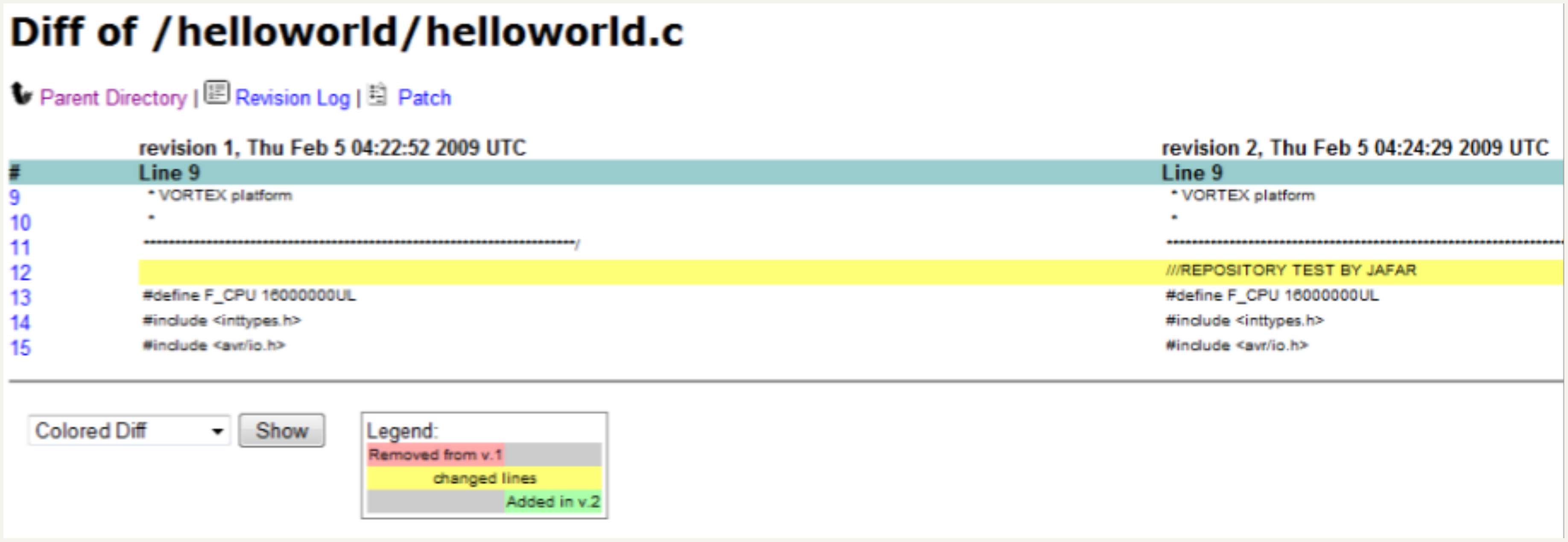
Diffs between 2 and 1

Type of Diff should be a Colored Diff

Get Diffs

Using the website – comparing revisions

Difference between revision 1 and revision 2



Diff of /helloworld/helloworld.c

Parent Directory | Revision Log | Patch

```
                revision 1, Thu Feb 5 04:22:52 2009 UTC                      revision 2, Thu Feb 5 04:24:29 2009 UTC
#               Line 9                                                       Line 9
9                * VORTEX platform                                            * VORTEX platform
10               *                                                            *
11              *******************************************************/      ********************************************************
12                                                                           //REPOSITORY TEST BY JAFAR
13              #define F_CPU 16000000UL                                     #define F_CPU 16000000UL
14              #include <inttypes.h>                                        #include <inttypes.h>
15              #include <avr/io.h>                                          #include <avr/io.h>
```

Legend:

🟥 Removed from v.1
🟨 changed lines
🟩 Added in v.2

Clients

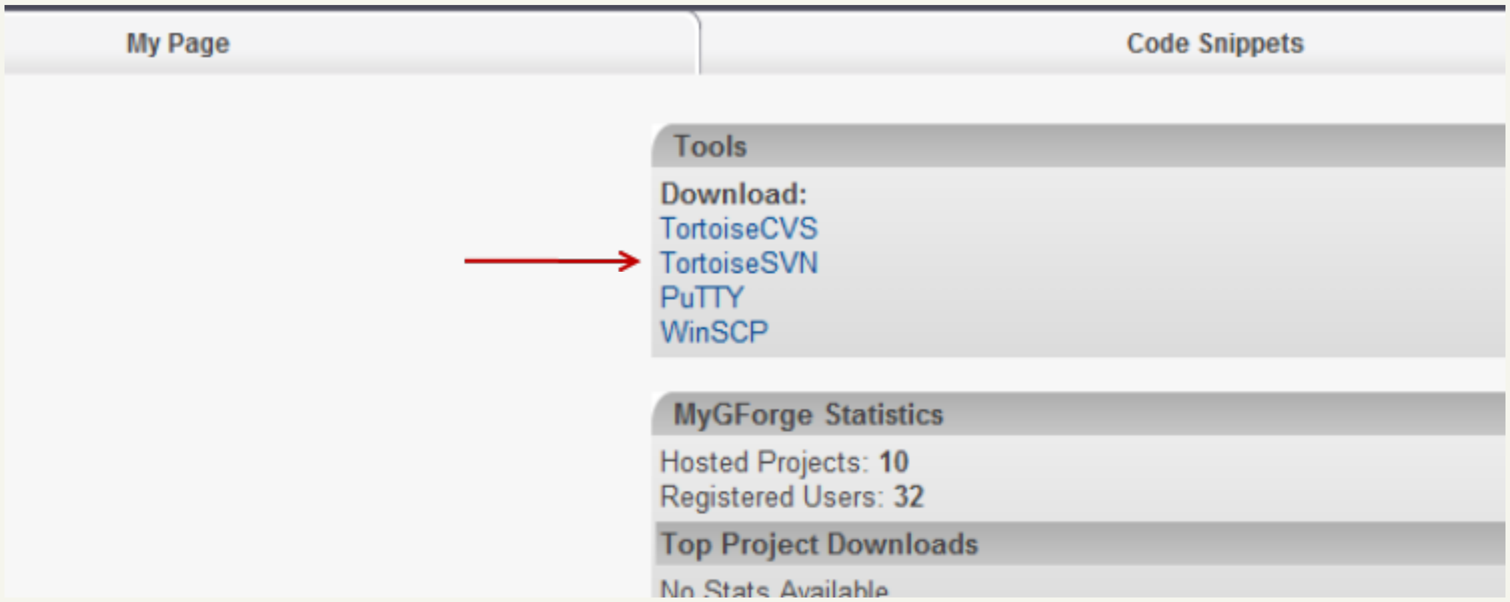
If you are already familiar with SVN and already using a client, you can use your client with MyGforge website.

Other clients that you can use:
1- TorToiseSVN is an SVN client for Windows
2- Unix Shell or MS-DOS

Using the TorToiseSVN

- The TorToiseSVN tool does not have an interface, it will be operated by using Windows explorer. The tool will give you the ability to Check-out the project code, commit your changes, update your working copy, and more.
- A link to the tool installer can be found at the website, under Home tab, to the right side of the screen under Tools.
- You **MUST** be registered with the project before using the client tool.

Using the TorToiseSVN



Hosted Projects: **10**
Registered Users: **32**

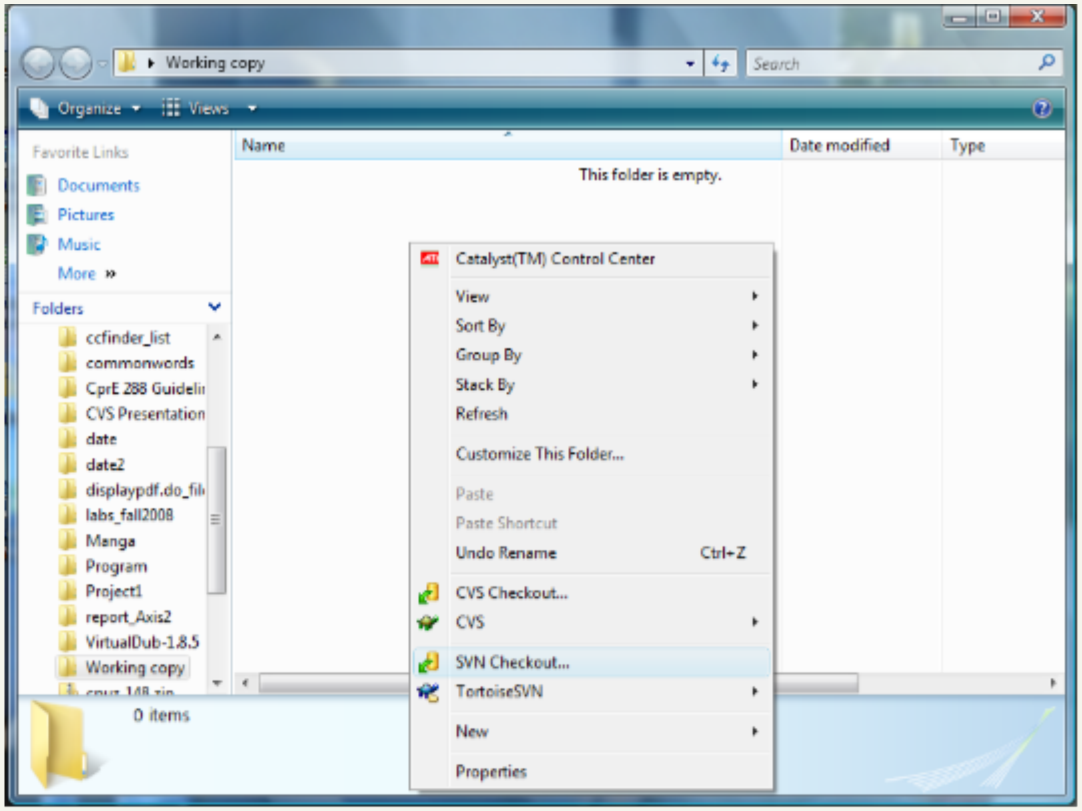
Top Project Downloads

No Stats Available

Make sure you select TorToiseSVN and NOT TorToiseCVS, the CVS tool will not work with our projects since it is CVS (Concurrent Versioning System) and not SVN (Subversion)

Using the TorToiseSVN

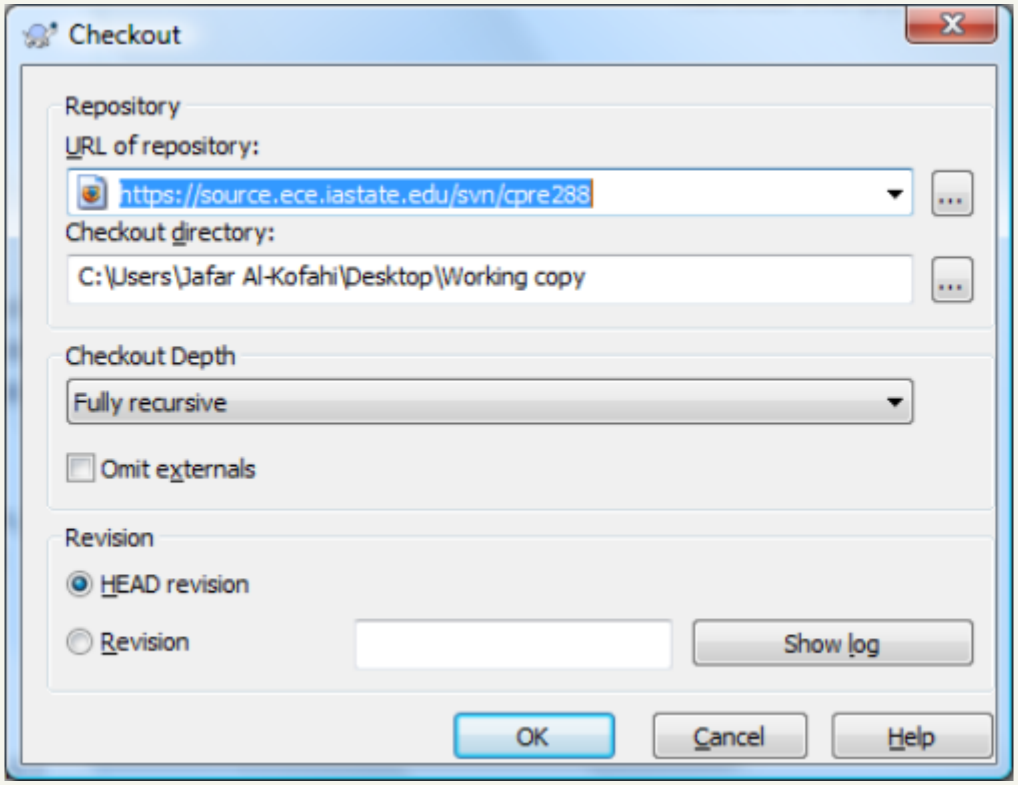
- After installing the tool, create a folder on your local drive and name it whatever you like, then open it and do a right-mouse click within it. And select SVN Checkout.



Using the TorToiseSVN

- Then put the given project URL for the repository:

(e.g. https://source.ece.iastate.edu/svn/cpre288)
And click Ok.


Checkout

Repository

URL of repository: https://source.ece.iastate.edu/svn/cpre288

Checkout directory: C:\Users\Jafar Al-Kofahi\Desktop\Working copy

Checkout Depth

Fully recursive

[ ] Omit externals

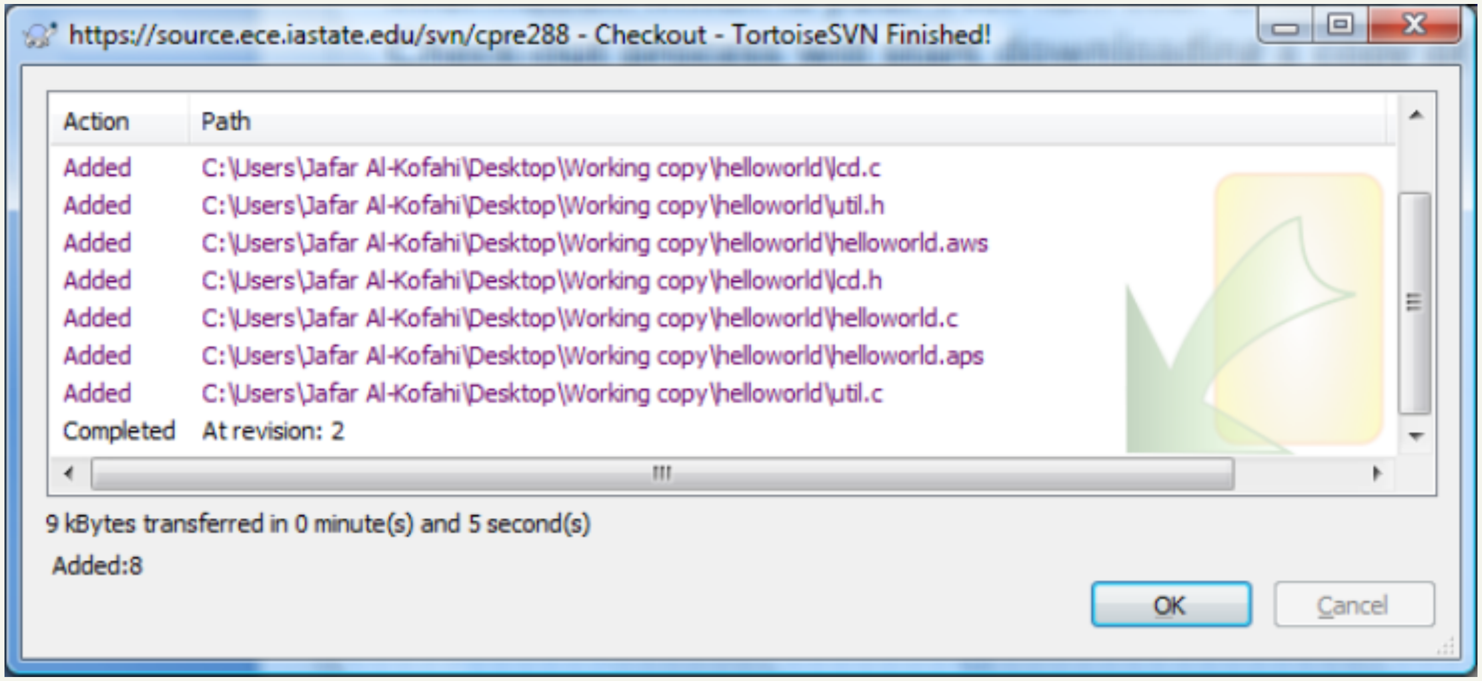
Revision

[● HEAD revision]

○ Revision

Using the TorToiseSVN

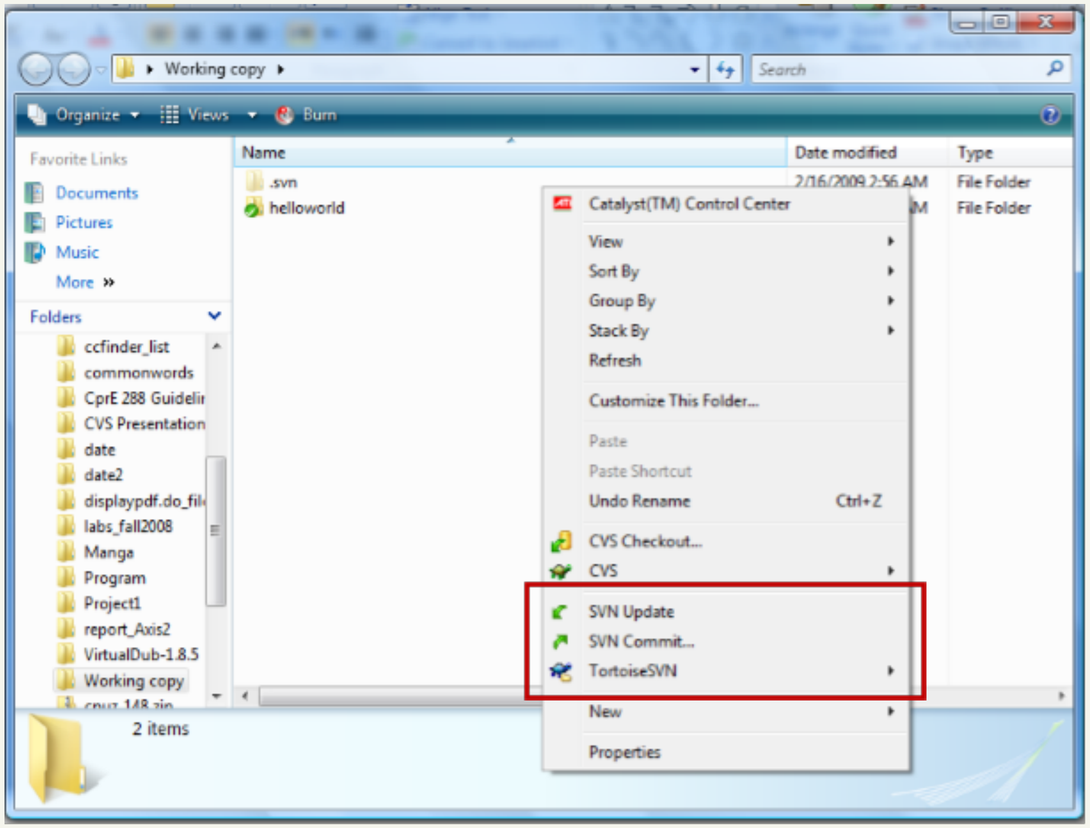
- After pressing Ok, it will ask you for your account information which is your Net-ID. After entering them the Check-out process will start downloading a copy of the project to your local folder.


https://source.ece.iastate.edu/svn/cpre288 - Checkout - TortoiseSVN Finished!

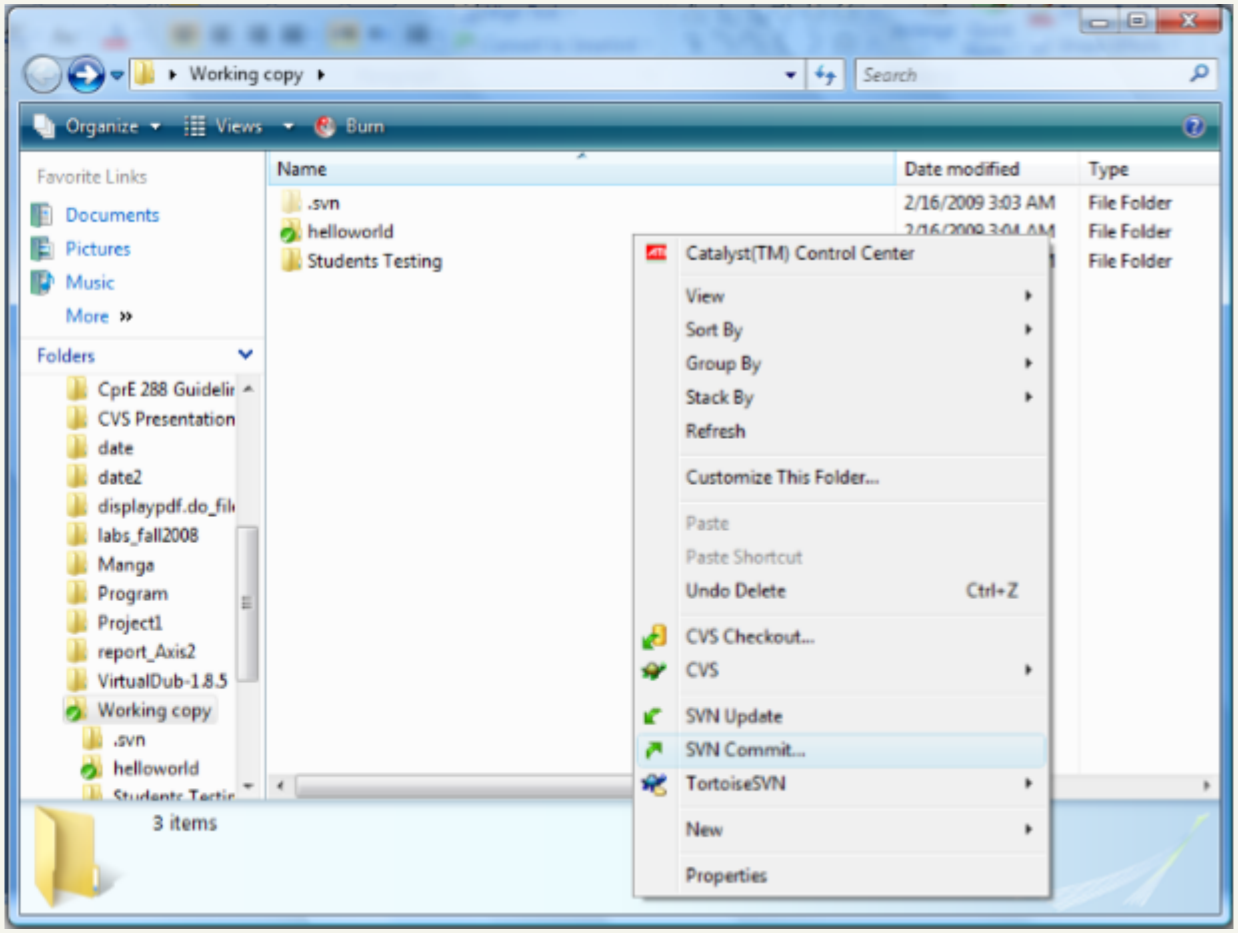
```
Action     Path
Added      C:\Users\Jafar Al-Kofahi\Desktop\Working copy\helloworld\lcd.c
Added      C:\Users\Jafar Al-Kofahi\Desktop\Working copy\helloworld\util.h
Added      C:\Users\Jafar Al-Kofahi\Desktop\Working copy\helloworld\helloworld.aws
Added      C:\Users\Jafar Al-Kofahi\Desktop\Working copy\helloworld\lcd.h
Added      C:\Users\Jafar Al-Kofahi\Desktop\Working copy\helloworld\helloworld.c
Added      C:\Users\Jafar Al-Kofahi\Desktop\Working copy\helloworld\helloworld.aps
Added      C:\Users\Jafar Al-Kofahi\Desktop\Working copy\helloworld\util.c
Completed  At revision: 2

9 kBytes transferred in 0 minute(s) and 5 second(s)
Added:8
```

Using the TorToiseSVN

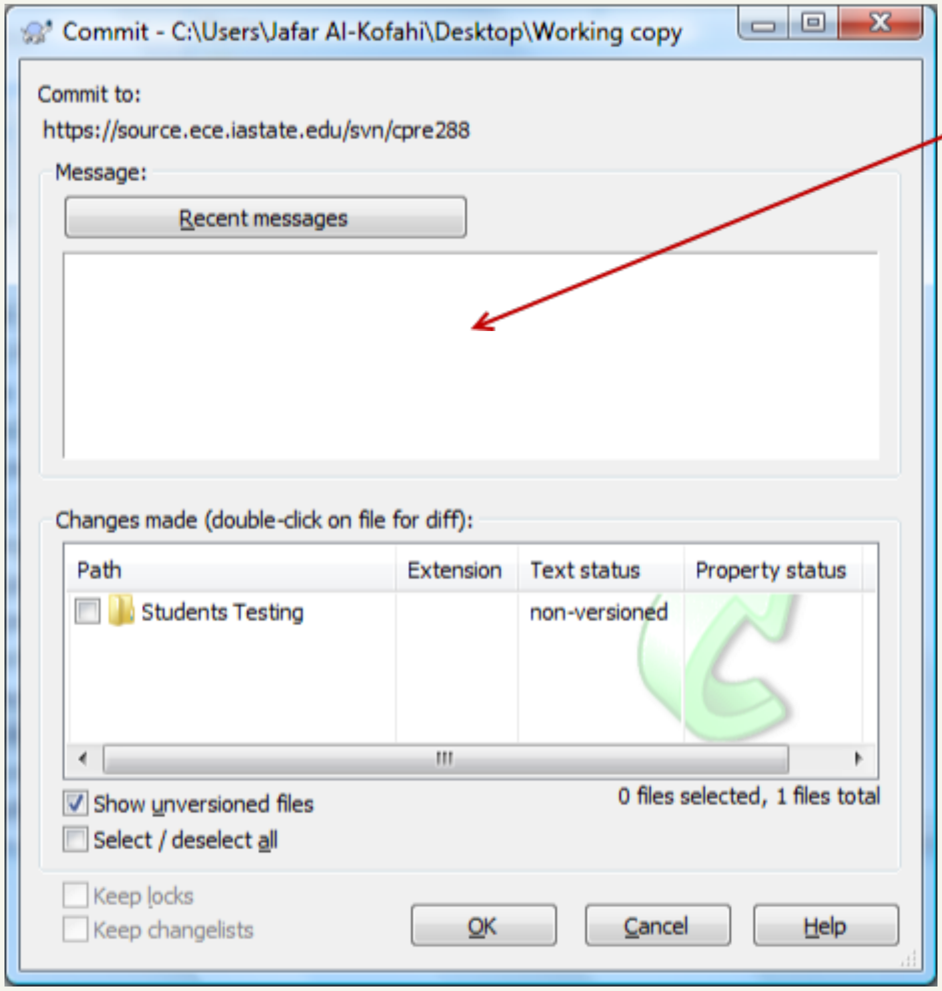
- Now after you got your copy, you can start using the SVN functionality (commit to apply changes, add files and folders, update local copy…etc)



Using the TorToiseSVN – Adding new folder



Using the TorToiseSVN - Commit



Commit - C:\Users\Jafar Al-Kofahi\Desktop\Working copy

Commit to:
https://source.ece.iastate.edu/svn/cpre288

Message:

Changes made (double-click on file for diff):

Path	Extension	Text status	Property status
☐ Students Testing		non-versioned	

☑ Show unversioned files
☐ Select / deselect all

0 files selected, 1 files total

☐ Keep locks
☐ Keep changelists

Write a log to describe your committed changes

Using Unix Shell or MS DOS

- Both use the same command and have the same options.
- To do any client operation on your SVN use the *svn* command.
- To view all available commands you can use the command: *svn help*

Using Unix Shell or MS DOS

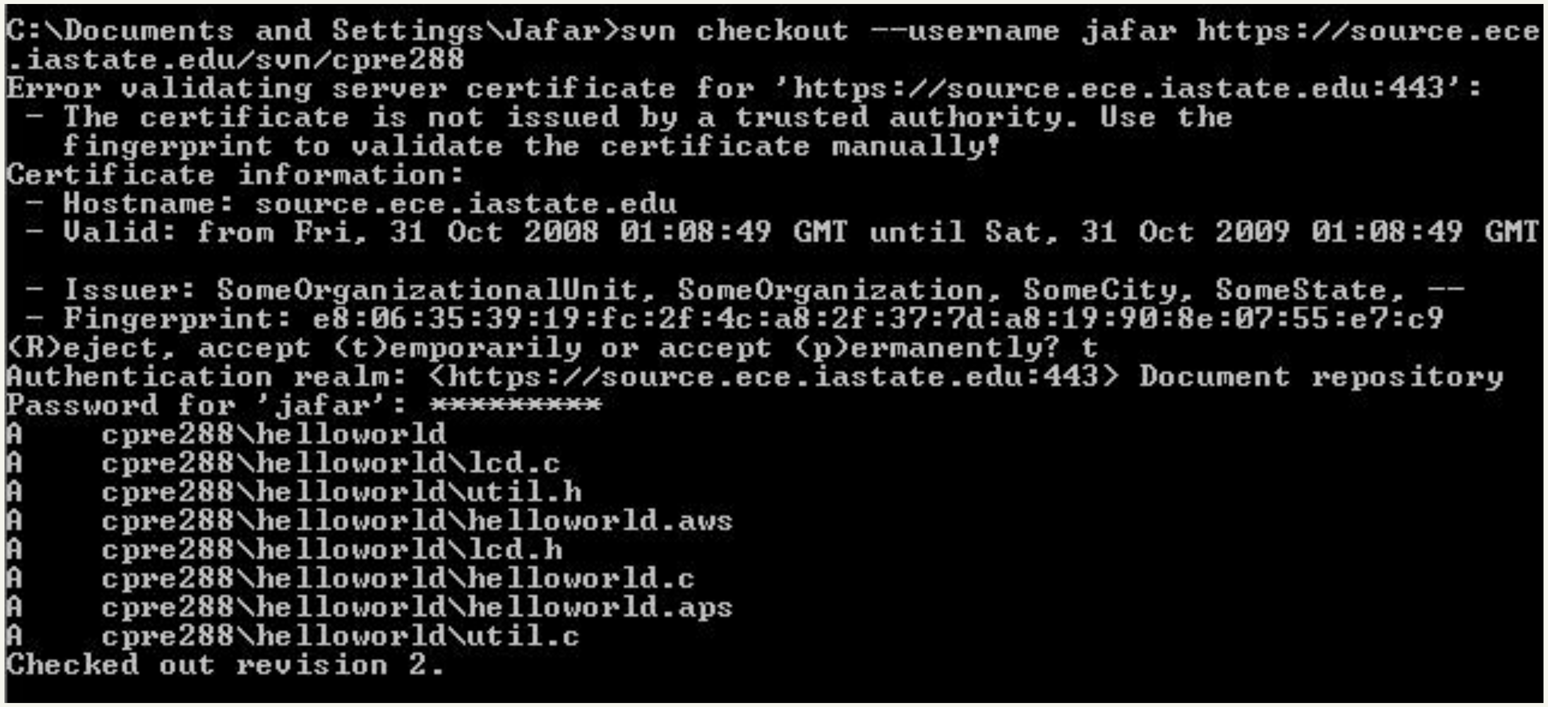
- For this course we are only interested in the following commands (to know more about each command execute *svn command -?*):
  - Check-out:
    - First go to the folder where you want your project to be checked out to (for the first time).
    - execute *svn checkout --username developername https://source.ece.iastate.edu/svn/projectname*
    - Where developername is your Net-ID and projectname is your assigned project name.

Using Unix Shell or MS DOS

- e.g. svn checkout --username jafar https://source.ece.iastate.edu/svn/cpre288
- After issuing this command you will be prompted with security confirmation message, please accept it by pressing *t* for temporarily accepting it or *p* for permanently accepting it. Notice that temporarily accepting it will keep prompting this message every time you do an operation on the SVN

Using Unix Shell or MS DOS

Screen shot of executing a check-out command on DOS



```
C:\Documents and Settings\Jafar>svn checkout --username jafar https://source.ece
.iastate.edu/svn/cpre288
Error validating server certificate for 'https://source.ece.iastate.edu:443':
 - The certificate is not issued by a trusted authority. Use the
   fingerprint to validate the certificate manually!
Certificate information:
 - Hostname: source.ece.iastate.edu
 - Valid: from Fri, 31 Oct 2008 01:08:49 GMT until Sat, 31 Oct 2009 01:08:49 GMT

 - Issuer: SomeOrganizationalUnit, SomeOrganization, SomeCity, SomeState, --
 - Fingerprint: e8:06:35:39:19:fc:2f:4c:a8:2f:37:7d:a8:19:90:8e:07:55:e7:c9
(R)eject, accept (t)emporarily or accept (p)ermanently? t
Authentication realm: <https://source.ece.iastate.edu:443> Document repository
Password for 'jafar': *********
A    cpre288\helloworld
A    cpre288\helloworld\lcd.c
A    cpre288\helloworld\util.h
A    cpre288\helloworld\helloworld.aws
A    cpre288\helloworld\lcd.h
A    cpre288\helloworld\helloworld.c
A    cpre288\helloworld\helloworld.aps
A    cpre288\helloworld\util.c
Checked out revision 2.
```

Using Unix Shell or MS DOS

- Commit (check-in):
  - Go to the project folder
  - To commit all of your changes, execute *svn commit --message "log message"* or you can use the file option if you have your log written to a text file *svn commit --file fileName*
- Add/Delete
  - You can use the command to add/delete file or folders, but you need to give the path to the file/folder you want to add/delete from your current path.
  - To execute an add command *svn add path*
  - Same thing for the delete command *svn delete path*

Using Unix Shell or MS DOS

- Update:
  - To update your local copy from the SVN
  - What will be updated is all files that do not have a conflict with the SVN (missing files will be restored, but changed files will not be restored)
  - *svn update*
- Revert
  - To revert a changed file in your local copy back to the current SVN revision. Or to restore a deleted file from your SVN version (e.g. after executing the delete command)
  - *svn revert filepath*

Using Unix Shell or MS DOS

- Status:
  - To check the difference between your local copy and the SVN. The first character will be one of the following:

Character	Meaning
?	File is not under version control
!	File is missing
A	File is scheduled for addition
C	File has textual conflicts from an update
D	File is scheduled for deletion
M	The contents of the file has local modifications

  - *svn status*

Using Unix Shell or MS DOS

- Differences:
  - To show differences between files (e.g. local copy and SVN)
  - *svn diff* : will show the differences between all files/ folders from the path the command where executed from.
  - To do it for a specific folder/file *svn diff path*
  - To output the results into a file instead of the console *svn diff > outputFile*

Reference

- Go to: http://svnbook.red-bean.com/en/1.5/svn-book.pdf
- For more information about SVN
- You MUST read **chapter 2** for more knowledge about SVN basic commands