Software Development

• Development process
• Tools
  – IDE’s Integrated Development Environments
  – Software revision control
  – Compilers
Software Development

- The steps in the development of software
  - preliminary design of software
  - create source code files
  - edit/compile/run cycle
    - the repetitive process of changing the code until it works

- Other related tasks
  - create unit tests to be used during the edit/compile/run cycle
    - process of creating simple test scripts that verify code runs correctly
    - usually test a single feature
  - create documentation either as part of source code or separately
  - prepare program for distribution
Software Development

• The development environment used varies greatly
• Simple command line oriented, traditionally used in Unix systems
  – manually create files using a text editor
  – edit/compile/run - some automation is possible using *make* and scripts
  – required files are arranged into directories and sub-directories
• Integrated development environments
  – capabilities vary among IDE’s
  – generally GUI based
  – manage files in terms of projects
  – automate tasks (compile-link-run)
  – provide links to add-on software (version control)
  – may support multiple languages
  – may have support for debugging and unit testing
Software Development

- The simple command line paradigm
- The project is located in a top-level directory, components (source code, data, unit tests) are kept in sub-directories
- Files are edited by the user by entering a command for the target file
  - `vi main.c`
- The compilation process, for a large project, is usually handled through the use of a `make` program
  - a simple scripting process used to compile code based on whether changes have been made
- The process of testing is performed by using shell/command scripts
  - in more complex projects there are “software harnesses” that can be used to automate the testing process
- When project is ready to distribute it is packaged in a distribution format
  - `tar`, `zip`, `msi`
Software Development

• The IDE paradigm
• The IDE is configured to place projects in some location on the disk
• For a new project, the IDE will create a standard directory structure which may depend on the type of project being created
• The IDE will usually have it’s own text editor built-in
  – syntax highlighting
  – syntax checking
  – statement completion
• The IDE will take care of the process of compiling modules as needed
  – configure compiler to use
  – configure libraries to include
• The IDE may have a built-in debugging mode
  – checkpoint
  – watch lists
Software Development

• IDE software
• Visual Studio
  – Microsoft only
  – $$$
  – support for several languages (C++ and C# are primary languages)
  – database development tools (MS-SQL)
  – Intel Fortran can integrate into VS
  – GT has a license for institute-owned computers
• Visual Studio Express
  – Microsoft only
  – Free (registration required)
  – Languages
    • Visual Basic, C++, C#, VJ#
    • single language per install, no add-on capability
Software Development

• IDE software
• Eclipse
  – coded in Java, will run on most platforms
  – primarily for Java programming but does support other languages
  – can be extended via a plug-in mechanism
    • Photran is the Fortran support plug-in http://www.eclipse.org/photran/
  – includes project administration
  – editing
    • syntax highlight
    • syntax check
    • code completion
  – http://www.eclipse.org/
Software Development

• IDE software
• Komodo
  – capability is a step below Eclipse in functionality
  – $$$
  – runs on multiple platforms
  – multiple languages (perl, tcl, python, ruby, xml/xslt, javascript)
  – manages code using projects
  – editor
    • syntax checking
    • syntax highlight
    • code completion
    – http://www.activestate.com/
• Komodo Edit
  – edit only version, free
Software Development

• IDE software
• Notepad++
  – primarily an editor only
  – free
  – Windows only
  – editor functions
    • syntax highlight
  – http://notepad-plus.sourceforge.net/
Software Development

- IDE software
- Emacs
  - primarily an editor only
  - free
  - multiple platforms
- vi
  - editor only
  - free
  - multiple platforms
  - http://www.vim.org/
Software Development

• Version control
  – what is version control
  – why use it
  – software available
  – using Subversion
Revision Control

• What is software version control?
  – version control is a software development methodology used to manage software revisions.

• Why use version control?
  – makes it possible to track changes to software over time.
  – makes it possible to maintain multiple branches of software.
  – makes it possible to back-track to previous versions.
Revision Control

• There are generally two modes of operation
  – locked access – where a single developer checks-out a copy of a file before modifying it. After changes are made the file it is checked-in. During check-out only a single developer has access to the file.
  – concurrent access – where multiple developers modify their own copy of a file and resolve conflicts that may develop at a later time.
Software

- There are several software implementations of version control.
  - RCS (Revision Control System) is one of the earliest. It was developed along with Unix. It implements a check-in/check-out mechanism.
  - CVS (Concurrent Versioning System) is a layer over RCS and implements a concurrent access scheme. CVS is available in both unix and Windows.
  - Subversion is a recent development and is designed as a replacement for CVS. It was designed to be easily deployed as a server. It is rapidly replacing CVS as the version control system of choice. It is available for both unix and Windows.
  - Source Code Control System (SCCS) is a Microsoft product and is part of Visual Studio. It is a check-in/check-out system and only runs on Windows.
  - CVS, Subversion (svn), and SCCS are the most commonly used.
  - There are other less widely used packages, both commercial and open source.
Subversion

• Subversion (svn) is fast replacing CVS as the versioning control system of choice.
• The basic Subversion software is a line mode package. It can be installed on either unix or Windows systems which can then inter-operate, eg, a Windows client can use a Linux based server.
• Subversion manages a repository that contains the versions of files in the repository.
• The user operates on file located in a working directory. Files are copied between the repository and working directory using Subversion commands.
• Subversion repositories can be local or server based. The location of the repository is indicated by the URL.
  – file://...
  – svn://…
  – http://...
Subversion

- Subversion is designed as a line mode command system, however there are GUI interfaces.
- TortoiseSVN is a windows explorer extension
  - [http://tortoisensvn.tigris.org/](http://tortoisensvn.tigris.org/)
  - TortiseSVN is installed as an add-on to the Windows file explorer and makes some changes
    - after installation restart your system
- This demonstration is performed on a local repository.
- For use with multiple developers a server based repository should be used.
- Never share a repository with other users using a shared file
  - file corruption can occur destroying your files
  - use one of the server implementations
Subversion

• Create a pair of folders
• *work* is the working directory
• *repos* is the repository directory
Subversion

- Create the repository directory for the specific project.
- Using the TortiseSVN Explorer extension, right click the mouse on the repos folder.
Subversion

• Next a dialog box will appear asking for the type of repository to create
  – the choice is either a Berkeley Database file or a file-based repository.
  – in most cases the file-based repository is best

• The equivalent line-mode command is
  
  `svnadmin create proj1 -fs-type fsfs`
Subversion

- This results in the **repos** folder shown at the right, these files are used to maintain the files
- At this point the repository is empty
Subversion

- The files that are to be placed under version control are created in the **work** directory.
- In this example a single file is created.
- The next slide shows the `matmult.c` file contents.
- Line 5 of the source file
  - the string `$Id: $` is used to identify the revision number of the project that the file is a member of
  - the text is replaced after the file has been imported into the repository.
#include <stdio.h>
#include <time.h>
#define MAX 500

static char svnid[] = "$Id: "$;

void printmat(double *a, int imax, int jmax) {
    long int i, j;
    printf("enter printmat\n");
    for (i=0; i<imax; i++) {
        for (j=0; j<jmax; j++) {
            printf("%5.2f ", *(a + (jmax * i) + j));
        }
    }
    printf("\n");
}

int main(int argc, char **argv) {
    long int i, j, k;
    int iter;
    double a[MAX][MAX];
    double b[MAX][MAX];
    double c[MAX][MAX];
    clock_t t1, t2;

    for (i=0; i<MAX; i++) {
        a[i][i] = 2.0;
        b[i][i] = 3.0;
    }
    t1 = clock();
    for (iter=0; iter<10; iter++) {
        for (i=0; i<MAX; i++) {
            for (j=0; j<MAX; j++) {
                c[i][j] = 0.0;
                for (k=0; k<MAX; k++) {
                    c[i][j] = c[i][j] + a[i][k] * b[k][j];
                }
            }
        }
    }
    t2 = clock();
    printf("t1=%d, t2=%d\n", t1, t2);
    printf("cpu time=%8.3f\n", (float)(t2 - t1)/CLOCKS_PER_SEC);
    /*
    printmat(c,MAX,MAX);
    */
}
Subversion

- The next step is to import the initial set of files into the repository
- Right click the mouse on the work directory
- The work directory contains your working copy of the project, it is your copy and is not shared
- You update the project by committing your changes to the repository

```
svn import work file:///e:/ae6382/repos
```
Subversion

- The dialog box that pops up requests the location of the repository, it is entered in URL format
- In this case the repository is a local directory so use the file:// form
- Use the message area to place a comment for this revision (initial)
Subversion

- This is the result of the import operation. It shows that one file has been added to the repository and that the current revision number is 1.
Subversion

- The contents of the repository can be viewed by using the repository browser

```
svn list file:///e:/ae6382/repos
```
The next step is to create an svn working directory that is under revision control

To do this, delete the contents of the work directory
  – at least the files that were added to the repository.

Then perform an svn checkout on the work directory.

```
cd work
svn co file:///e:/ae6382/repos
```
Subversion

- This dialog box appears, it specifies the source repository and the destination working directory
Subversion

- This dialog box shows the files that have been added to the working directory and the revision number
Subversion

- Open the working directory
- Items under SVN revision control have distinctive icons

svn list
svn -R list
svn status
Subversion

- Additional information about revision controlled files can be displayed by using View/Details and then View/Choose Details…
Subversion

- The dialog box that results from View/Choose Details…
- The SVN parameters have been added
- Selecting any of these will change the View/Details display
Subversion

- The result of checking SVN Revision and SVN Status.
Subversion

- To set properties on SVN controlled files
Subversion

- The dialog box for setting properties
- Properties are meta-data associated with the file that is maintained by the SVN server
Subversion

- The drop-down box shows the available properties
Subversion

• *svn:eol-style*
  • Subversion will set the end-of-line in a text file appropriate to the destination working directory

• *svn:keywords*
  • Informs Subversion to handle the keywords, eg, `$Id: $`, which will be updated when the file is downloaded to the working directory

• *svn:executable*
  • Informs Subversion that the file is an executable file. It will set permissions on the working copy to allow execution
Subversion

- For this file svn:keywords has been set to Id and svn:eol-style has been set to native.

```
svn propset svn:keywords Id matmult.c
svn propset svn:eol-style native matmult.c
```
Subversion

- Going back and looking at the work directory notice that the icon for the file matmult.c has changed
- In this case it indicates that the svn properties for the file have changed, usually it is the result of a change in the file contents
Subversion

- Commit the changed file to the repository
Subversion

- The dialog that pops up allows you to enter a log message that describes the changes made

- `svn commit -m 'changes in the svn properties' matmult.c`
Subversion

- The final dialog box verifies the update of the repository
- The revision number is now 2
- The icon for matmult.c reverts to the green check
- Looking at the file matmult.c shows the following:

  ```c
  static char svnid[] = "$Id: matmult.c 2 2006-11-08 15:33:16Z latham "$;
  ```
Subversion

- To add a new file, `readme.txt`, to the repository. Create the file in the working directory, then use `svn add`
Subversion

- The dialog box that lists the files to be added to the repository
- And the result of the add operation.

svn add readme.txt
Subversion

- The freshly added file has a “+” icon to indicate its status
- Now set the properties of the file then commit it.

```
svn commit -m 'initial add' readme.txt
```
Subversion

- The status of the working directory after the commit operation. Note that readme.txt is at revision 3 and matmult.c is at revision 2
Subversion - Notes

• The latest revision (highest) is the head revision
• Each file may have a different revision number, the highest is returned for the head revision
• In a multi-user environment perform an update operation before starting to change a file, to get the latest revision
• There are plug-ins available for various IDE packages
• There are also GUI implementations for Linux at tigris.org
• Complete documentation on Subversion is at