Access to CS Computing Resources

- Remotely:
  - CS student sever **agate.cs.unh.edu** accessible via **ssh/sftp/scp**
- Locally:
  - **16** linux PCs and **2** iMacs (?) in cluster in **N241**, accessible **24/7** (combo lock), open to **CS majors only**
  - **21** linux PCs in classroom **N218**, accessible during the day (when not in use by a class) to **all students** taking CS courses
  - In both cases, **home directory** from **agate** is mounted.

Java Programming Environments
(Version of reference: 1.6.0_23)

Command-line tools:

- freely available from **Oracle** (`java.oracle.com`)
- installed on **CS** cluster and server machines
- used in grading and testing
- `javac`: **Java compiler** (generates byte code)
- `java`: **Java Virtual Machine** (runs byte code)

Eclipse:

- freely available from `www.eclipse.org`
- installed on **CS** cluster and server machines
- **no help** provided in this course

emacs:

- freely avaliable from `www.gnu.org/software/emacs`
- installed on **CS** cluser and server machines
- some (limited) help provided in this course

Absolutely no help on installing / running programs on MSwindows

SML "New Jersey" (used in the textbook)

- Interactive compiler / interpreter
  - available as `/usr/local/bin/sml` on CS cluster
  - available for Unix and Windows: http://www.smlnj.org/
  - documentation on standard library: http://www.standardml.org/Basis/
  - version of reference: Standard ML of New Jersey v110.72 [built: Fri Aug 20 17:13:42 2010]
- sml-mode for emacs users http://www.smlnj.org/doc/Emacs/sml-mode.html
- probably supported by Eclipse (SMLclipse?)

Books

- Ken Arnold, James Gosling and David Holmes. "*The Java Programming Language.*", Addison-Wesley, 2005, 4th edition. ISBN: 0-321-34980-6.
- Jeffrey D. Ullman. "*Elements of ML Programming.*", Prentice Hall, 1998. ISBN: 0-13-790387-1.
- Joshua Bloch. "*Effective Java.*", 2nd edition, Addison-Wesley, 2008. ISBN 0-321-35668-3. (available online through Safari)

Homework Submission

- homework submission is done using **subversion** (`http://subversion.apache.org`)
- students indicate **which revision** to grade by sending an email to `cs671@cs.unh.edu`
- student *abcdef* uses a **repository** named `https://stsvn.cs.unh.edu/svn/cs671.abcdef`
- **passwords** have been sent by email
- repositories **must** contain subdirectories for **each assignment**: `HW1`, `HW2`, ..., `HW8`
- for **Java**, the root of the repository contains a **makefile**
- this **makefile** is used to create a `cs671` directory (bytecode) and a `html` directory (documentation)
- for **SML**, directories `HW5`, ..., `HW8` each contain a file `main.sml`
- a sample **makefile** is included in each repository
- **DO NOT** commit generated files (`html` from javadoc, `.class` from javac, ...)

Subversion

- **version control** system
- enables **collaborative work** (*NOT* to be used in this class!)
- maintains different **revisions** of **every file** in a given **repository**
- **any** revision can be retrieved from the repository
- each revision is associated with a **timestamp**
- each revision is associated with a **change log** that gives the complete **history** of a file
- many subversion **clients** exist, some are integrated with **IDEs** (Eclipse, emacs, . . .)

For student *abcdef*:

the CS–671 repository is:

`https://stsvn.cs.unh.edu/svn/cs671.abcdef`

checking out a working copy:

```
> svn checkout https://stsvn.cs.unh.edu/svn/cs671.abcdef W671
A    W671/HW1
A    W671/HW1/RandomLetters.java
A    W671/javadoc-options
A    W671/makefile
Checked out revision 2.
```

- this creates a `W671` directory called **working copy**
- the name of the working copy directory can be anything
- if no name is specified, the working copy is named after the repository (`cs671.abcdef`)
- after all changes have been **committed** to the server, the working copy can be deleted, but it does not have to
  (if you plan to keep working from the same computer later)

getting the history of files:

```
> cd W671/
> svn log
------------------------------------------------------------------------
r2 | abcdef | 2012-01-23 11:26:14 -0500 (Mon, 23 Jan 2012) | 2 lines

Added a given interface for assignment #1.  This file CANNOT be modified.

------------------------------------------------------------------------
r1 | abcdef | 2012-01-23 11:21:51 -0500 (Mon, 23 Jan 2012) | 2 lines

A sample makefile.  This file can be modified.

------------------------------------------------------------------------
```

- the `log` command can take **parameters** (default is '.') and **options**

```
> svn log -q makefile
------------------------------------------------------------------------
r1 | abcdef | 2012-01-23 11:21:51 -0500 (Mon, 23 Jan 2012)
------------------------------------------------------------------------
```

checking the status of the working copy:

```
> svn status
> mkdir HW1/tests
> svn status
?       HW1/tests
> svn status -v
                2        2 abcdef       .
                2        2 abcdef       HW1
?                                       HW1/tests
                2        2 abcdef       HW1/RandomLetters.java
                2        1 abcdef       javadoc-options
                2        1 abcdef       makefile
```

- by default **unmodified files/dir** are ommitted
- **?** means *not under version control* (i.e., not in the repository)

modifying and undoing:

```
> vi makefile # let's modify the makefile
> rm javadoc-options # oops!
> svn status
?       HW1/tests
!       javadoc-options
M       makefile
```

- **M** means that the local copy has been **modified**
- **!** means that a repository file is **missing**

```
> svn diff makefile
Index: makefile
===================================================================
--- makefile (revision 2)
+++ makefile (working copy)
@@ -1,3 +1,6 @@
+info:
+ @echo possible targets are 'cs671' 'html' and 'tests'
+
 cs671: HW?/*.java
  javac -Xlint -d . HW?/*.java
  @touch cs671
```

```
> svn revert makefile javadoc-options
Reverted 'makefile'
Reverted 'javadoc-options'
> svn status
?       HW1/tests
```

adding files:

```
> vi HW1/tests/TestDictionary.java # let's create a test file
> svn add HW1/tests
A         HW1/tests
A         HW1/tests/TestDictionary.java
```

- **add** is recursive by default (use -N option for non-recursive)
- files are *not* actually added to repository until **committed**

committing changes to the repository:

```
> svn commit -m 'Added a Dictionary test file'
Adding         tests
Adding         tests/TestDictionary.java
Transmitting file data .
Committed revision 3.
> svn status
```

- without -m option, an **editor** (usually vi) starts and prompts for message

```
> vi HW1/tests/TestDictionary.java # let's add more tests to the file
> svn status
M       HW1/tests/TestDictionary.java
> svn commit # this will start vi
Sending        HW1/tests/TestDictionary.java
Transmitting file data .
Committed revision 4.
> svn log HW1/tests/TestDictionary.java
------------------------------------------------------------------------
r4 | abcdef | 2012-01-23 12:41:45 -0500 (Mon, 23 Jan 2012) | 4 lines

Added more tests, especially on large dictionaries and
on dictionaries created from URLs.  There seems to be
a bug on empty dictionaries; needs to work on it.

------------------------------------------------------------------------
r3 | abcdef | 2012-01-23 12:22:26 -0500 (Mon, 23 Jan 2012) | 1 line

Added a Dictionary test file
------------------------------------------------------------------------
```

- **always** write a **meaningful** message when committing

looking at past revisions:

```
> svn cat -r 3 HW1/tests/TestDictionary.java
package tests.anagram;

import charpov.grader.*;
import static org.testng.Assert.*;
...
```

- this displays **revision 3** of file `TestDictionary.java`
- `svn cat HW1/tests/TestDictionary.java@3` does the same thing
- `svn cat -r {'2012-01-23 12:30'} HW1/tests/TestDictionary.ja` retrieves the file as it was on Jan 23rd at half-past noon

ignoring files:

```
> make tests html
javac -Xlint -d . HW?/*.java
javac -d . HW?/tests/*.java
javadoc @javadoc-options HW?/*.java
Creating destination directory: "html/"
> svn status
?       tests
?       cs671
?       html
> svn propedit svn:ignore . # starts vi
Set new value for property 'svn:ignore' on '.'
> svn propget svn:ignore
cs671
html
tests

> svn status
 M      .
> svn commit -m 'set svn:ignore to ignore cs671, html and tests directories'
Sending        .

Committed revision 5.
> svn status
> ls
cs671  html  HW1  javadoc-options  makefile  tests
```

configuration:

```
> cat ~/.subversion/config
...
global-ignores = *.class *.aux *.o *.lo *.la #*# .*.rej *.rej .*~ *~ .#* .DS_Store
...
enable-auto-props = yes
[auto-props]
*.sh = svn:eol-style=native;svn:executable
makefile = svn:eol-style=native;svn:keywords=Id
*.java = svn:eol-style=native;svn:keywords=Id
*.sml = svn:eol-style=native;svn:keywords=Id
...
```

- `svn:eol-style=native` allows working copies on Windows and linux to have the correct end-of-lines
- `svn:keywords=Id` allows top lines of the form:

```
// $Id: RandomLetters.java 641 2012-01-16 16:42:11Z charpov $
```

  to be **automatically updated** by subversion
- `svn:executable` sets the **executable bit** on a working file
- ...

- other subversion clients may work differently (Eclipse, emacs, . . .)
- in particular, they **store passwords** in different ways (e.g., keychain on Mac OS)
- there is **much more** to subversion
- `svn help` displays all the possible commands
- `svn help log` displays help on the `log` command
- `http://svnbook.red-bean.com` has an entire book available on subversion (free!)

- subversion tries to **merge** changes from different contributors
- it prompts for guidance when the merge is not obvious
- it could happen to you if you maintain several working copies (at home, at UNH, . . .) and forget to commit
- resolving such conflicts can be tricky (check the book)
- commit changes often

Java Testing

- uses a **custom-made** system similar to JUnit or TestNG
- system available on agate as `/usr/java/latest/jre/lib/ext/grader.jar`
- tests **fail** by **throwing exceptions**
- a nice way to throw exceptions is to use `org.testng.Assert`

```
import charpov.grader.*;
import static org.testng.Assert.*;

class TestDictionary {
  public static void main (String[] args) throws Exception {
    java.util.logging.Logger.getLogger("charpov.grader")
      .setLevel(java.util.logging.Level.WARNING);
    new Tester(MyTests.class).run(); // could specify several classes
  }
}
class MyTests {
  @Test(timeout=3000) void sillyTest () { // 3 seconds to run
    assertEquals(2 + 2, 4);
  }
}
```

- you are free to **reorganize** your repository in any way, as long as a working **makefile** is provided
- if `SampleTests.java` is in the current directory, this should work:

```
> svn co -q https://stsvn.cs.unh.edu/svn/cs671.abcdef dir
> make -s -C dir                                  # creates dir/cs671
> javac -d . -cp .:dir SampleTests.java           # creates ./tests
> java -cp .:dir tests.anagram.SampleTests
SUCCESS: tests.anagram.Sample1.testAddWords completed in 5.3 milliseconds
SUCCESS: tests.anagram.Sample2.testAllWords completed in 4.9 milliseconds
SUCCESS: tests.anagram.Sample1.testAllWords completed in 0.1 milliseconds
SUCCESS: tests.anagram.Sample2.testContainsWord completed in 0.2 milliseconds
SUCCESS: tests.anagram.Sample1.testDashes completed in 0.3 milliseconds
SUCCESS: tests.anagram.Sample2.testGet completed in 0.2 milliseconds
SUCCESS: tests.anagram.Sample1.testEmpty completed in 0.5 milliseconds
SUCCESS: tests.anagram.Sample2.testGetAll completed in 0.4 milliseconds
SUCCESS: tests.anagram.Sample1.testRemoveWords completed in 0.1 milliseconds
SUCCESS: tests.anagram.Sample3.testAddWordsFile completed in 14.1 milliseconds
SUCCESS: tests.anagram.Sample3.testAddWordsURL completed in 4.1 milliseconds
INFO:    11 tests passed; 0 failed
```